Developing Web Applications Using Lotus Notes Designer for Domino 4.6

International Technical Support Organization
Edition Notice

First Edition (September 1997)
This edition applies to Domino R4.6.
Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to
use or distribute the information in any way it believes appropriate without
incurring any obligation to you.

© International Business Machines Corporation 1997. All rights reserved.

Note to U.S. Government Users: Documentation related to restricted rights.
Use, duplication or disclosure is subject to restrictions set forth in GSA ADP
Schedule Contract with IBM Corp.
Abstract

This redbook describes how to develop applications using Notes Designer for Domino Release 4.6, specifically for users accessing Domino from Web browsers. Domino combines the open networking environment of Internet standards and protocols with the powerful application development facilities of Lotus Notes, enabling you to develop a broad range of business applications for the Internet and intranet.

The earlier chapters of the book introduce the concepts of Domino and how to create Web applications using Domino, for those readers who may be familiar with Web application development but not necessarily with using Lotus Notes/Domino for that development.

The book then describes how to use Domino design elements such as databases, forms, subforms, views, navigators, and agents to create Web applications. Other chapters illustrate how to include traditional Web development tools, such as HTML, Java and JavaScript, in Domino applications.

Security considerations for Web applications are also discussed, as well as how to access relational databases from Domino applications using LotusScript Data Object and ODBC.

This redbook was written for Domino technical specialists and programmers — customers, IBM Business Partners, and the IBM and Lotus community — who need a good technical understanding of how to develop Web applications using Domino Release 4.6.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edition Notice</strong></td>
<td>ii</td>
</tr>
<tr>
<td><strong>Abstract</strong></td>
<td>iii</td>
</tr>
<tr>
<td><strong>Special Notices</strong></td>
<td>xi</td>
</tr>
<tr>
<td><strong>Preface</strong></td>
<td>xiii</td>
</tr>
<tr>
<td><strong>1 Domino: Architecture and Configuration</strong></td>
<td>1</td>
</tr>
<tr>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>Lotus Domino and the Web</td>
<td>1</td>
</tr>
<tr>
<td>Domino Architecture</td>
<td>2</td>
</tr>
<tr>
<td>Supported Platforms</td>
<td>3</td>
</tr>
<tr>
<td>Configuring the Domino Web Server</td>
<td>4</td>
</tr>
<tr>
<td>HTTP Setup</td>
<td>4</td>
</tr>
<tr>
<td>Summary</td>
<td>7</td>
</tr>
<tr>
<td><strong>2 Getting Started with Domino Web Applications</strong></td>
<td>9</td>
</tr>
<tr>
<td>About Domino Web Databases</td>
<td>9</td>
</tr>
<tr>
<td>Glossary of Notes Designer Terms</td>
<td>10</td>
</tr>
<tr>
<td>Creating a Database</td>
<td>13</td>
</tr>
<tr>
<td>Using an Existing Template</td>
<td>13</td>
</tr>
<tr>
<td>Copying an Existing Database</td>
<td>16</td>
</tr>
<tr>
<td>Creating a New Database</td>
<td>18</td>
</tr>
<tr>
<td>Changing the Database Properties</td>
<td>18</td>
</tr>
<tr>
<td>Setting Up Your Web Site</td>
<td>24</td>
</tr>
<tr>
<td>Designing Your Home Page</td>
<td>26</td>
</tr>
<tr>
<td>Accessing a Domino Site</td>
<td>27</td>
</tr>
<tr>
<td>Summary</td>
<td>30</td>
</tr>
<tr>
<td><strong>3 Designing Application Forms</strong></td>
<td>31</td>
</tr>
<tr>
<td>Using Forms</td>
<td>31</td>
</tr>
<tr>
<td>Specifying Form Properties</td>
<td>32</td>
</tr>
<tr>
<td>Giving the Form a Title</td>
<td>38</td>
</tr>
<tr>
<td>Form Events</td>
<td>39</td>
</tr>
<tr>
<td>Creating a Field</td>
<td>41</td>
</tr>
<tr>
<td>Performing a Test Run</td>
<td>42</td>
</tr>
<tr>
<td>Field HTML Attributes</td>
<td>43</td>
</tr>
<tr>
<td>Field Properties</td>
<td>45</td>
</tr>
<tr>
<td>Keywords Field on the Web</td>
<td>50</td>
</tr>
<tr>
<td>Sharing Design Elements with Subforms</td>
<td>51</td>
</tr>
<tr>
<td>Subform Properties</td>
<td>52</td>
</tr>
<tr>
<td>Removing Subforms</td>
<td>53</td>
</tr>
<tr>
<td>Computed Subforms</td>
<td>53</td>
</tr>
<tr>
<td>Displaying a Different Form to Web Users and Notes Users</td>
<td>55</td>
</tr>
<tr>
<td>Tables Within Forms</td>
<td>56</td>
</tr>
<tr>
<td>Table Properties</td>
<td>56</td>
</tr>
<tr>
<td>Using Tables for Layout on the Web</td>
<td>57</td>
</tr>
<tr>
<td>Merge and Split cells</td>
<td>58</td>
</tr>
<tr>
<td>Web Elements</td>
<td>59</td>
</tr>
<tr>
<td>Embedded Navigators</td>
<td>60</td>
</tr>
<tr>
<td>Embedded View</td>
<td>62</td>
</tr>
<tr>
<td>Embedded Folder Pane</td>
<td>64</td>
</tr>
<tr>
<td>File Upload Control</td>
<td>66</td>
</tr>
<tr>
<td>Other Features of Forms</td>
<td>68</td>
</tr>
<tr>
<td>Horizontal Rules</td>
<td>68</td>
</tr>
<tr>
<td>Computed Text</td>
<td>69</td>
</tr>
<tr>
<td>Using $$Return Field</td>
<td>71</td>
</tr>
</tbody>
</table>
# Domino Views and Folders

What Are Views and Why Are They Useful? ................................................. 81
Selecting Which Documents Appear in a View .................................. 82
Looking at a View Using a Notes Client ........................................ 83
Looking at a View Using a Web Browser ........................................ 85
Creating Views for the Web ......................................................... 88
View Design ............................................................................. 88
Selecting Which Documents Appear in a View .................................. 89
View Properties .......................................................................... 90
Specifying Properties ................................................................... 92
Changing a View ......................................................................... 93
Creating Column Header Sort .................................................... 96
Selecting Columns to Contain Document Links ................................ 98
Adding HTML to a View ............................................................. 100

# Domino Navigators

Building a Navigator ...................................................................... 117
Using the Design Tools for Navigators ......................................... 119
Hints and Tips for Creating Navigators ....................................... 120
Example of Embedded, Calculated Navigator .......................... 121
Targeting Navigator Links into a Frame ..................................... 123
Summary ................................................................................. 124

# Using HTML in Domino Design Elements

Part I: Adding HTML to Domino Design Elements .......................... 126
Adding HTML to a Form ............................................................ 126
Writing HTML on a Form .......................................................... 126
HTML in Computed Fields ......................................................... 128
HTML in Computed Text ............................................................ 128
Modifying the <Body> Tag Attributes ....................................... 129
Adding HTML to a Document .................................................... 130
What You Can Add to a RichText Field ..................................... 130
What You Can Modify at Document Level .................................. 131
Formatting Fields Using HTML .................................................. 132
Creating a Multi-Line Text Field on the Web ......................... 133
Masking Characters on Input Fields ......................................... 134
Mixing HTML With Field And Static Text ............................... 134
HTML for the $$Return Field .................................................... 138
HTML for the $$HTMLHead Field ............................................. 140
Adding HTML to a View or Folder ........................................... 144
Using an Image as DocLink ....................................................... 144
# Domino Security

## 9 Domino Security .......... 227

### Domino Security Structure .......... 227

### Security for Web Access .......... 228

#### Registered and Unregistered Users .......... 228

#### Authentication .......... 229

#### Defining Web Users .......... 230

#### Registering Web Users .......... 231

#### Separate Web Users and Notes Users in Public Address Books .......... 232

#### Password Considerations for Web Users .......... 233

### Protecting the Domino Server .......... 234

#### Using Firewalls with Domino .......... 235

#### Hide the Database Name List on the Server .......... 237

#### Database Access Control List .......... 237

#### Setting up the ACL for a Web Database .......... 239

### Programming Considerations .......... 241

#### Hide Content by Hide/When Attributes .......... 241

#### Encryption and Signature .......... 242

### SSL Implementation On Domino .......... 243

#### Why Use SSL? .......... 243

#### SSL Overview .......... 244

#### SSL in Domino Server Environment .......... 246

#### SSL On Domino Servers Using Domino SSL Applications .......... 246

#### Example: Web User Validation and Authentication .......... 256

### Conclusion .......... 257

### Summary .......... 257

# Java Applets, Java Agents, and JavaScript

## 10 Java Applets, Java Agents, and JavaScript .......... 259

### Overview .......... 259

#### What is a Java Applet? .......... 259

#### What are Java Agents? .......... 260

### Java Applets .......... 260

#### Displaying Java Applets .......... 261

#### Inserting Java Applets into Domino Design Elements .......... 262

#### Enabling Java Applets .......... 263

#### Importing an Applet from your File System .......... 263

#### Linking to an Applet on the Web .......... 264

#### Setting Applet Attributes .......... 265

#### Stopping a Java Applet .......... 265

#### Displaying the Java Applet Menu .......... 266

#### Setting the Applet Properties .......... 267

#### Setting Other HTML Attributes .......... 268

#### Setting the Applet Parameters .......... 269

#### Setting the Design, Reorganize, and Export Options .......... 271

### Tips and Troubleshooting for Java Applets .......... 271

#### Troubleshooting an Applet that is Not Running .......... 271

#### Accessing Resource Files .......... 272

#### Basic Error Handling .......... 272

### Java Notes Classes and Java Agents .......... 272

#### What is a Java Agent? .......... 272

#### About Java Notes Classes .......... 273

#### Java Agents and Java Applets .......... 274

#### Compiling and Running a Java Program .......... 274

#### Creating a Java Agent .......... 275

#### Example 1: Using Java Notes Classes .......... 277

#### Example 2: Java Agent .......... 281

### Using JavaScript in Domino .......... 283

#### Adding JavaScript to Forms .......... 284
11 Accessing External Data from a Domino Application ....... 297
Data Resource Access ............................ 297
   Database Access Facilities ................. 297
What Is ODBC? .............................. 299
   Using ODBC Connections ............... 299
   ODBC Access Flow .................... 299
LotusScript:DataObject (LS:DO) .......... 301
   What is LS:DO? ........................... 301
   Concepts ................................ 301
   Mobile User Queries and Updates .... 302
   Architecture ............................ 303
   When to Use LS:DO ...................... 304
Differences Between LS:DO and ODBC .... 304
   Programming Environment .......... 304
   Functionality ........................... 305
   Performance ............................ 306
   Software Requirements .............. 306
   Registering ODBC Data Sources .... 307
   USELSX Statement to Enable LS:DO ... 309
Mapping Data Types Between RDB and Notes DB .......... 309
   How to Trace and Debug LS:DO ....... 309
LS:DO Class Library ........................... 313
   ODBCConnection Class ............... 315
   ODBCQuery Class .................... 318
   ODBCResultset Class ................. 319
Using @DB Functions to Access Other Databases Through ODBC ........ 325
   When to Use ............................ 326
   How to Use @DB Functions ........... 326
   @DBColumn ................................ 327
   @DBLookup .............................. 327
   @DBCCommand ........................... 328
Summary ...................................... 328

Appendix A Notes Features to Avoid when Developing Web Applications .......... 329
Appendix B Downloading and Setting Up the Chile Pepper Site .......... 335
   Downloading the Chile Pepper Demonstration ................ 335
   Setting Up the Demo Site ................ 336
Appendix C Table of CGI Variables Supported by Domino .................. 337
Appendix D Editing The HTTPD.CNF file to Add MIME Type Mappings ............ 339
Related ITSO Publications .................. 343
   ITSO Lotus Publications .............. 343
   Other ITSO Lotus Related Publications .... 343
How To Get ITSO Redbooks ............... 345
   How IBM and Lotus Employees Can Get ITSO Redbooks ........ 345
   How Customers Can Get ITSO Redbooks ........ 347
Index ...................................... 349
Special Notices

This publication is intended to help technical professionals develop Web applications using Lotus Notes Designer for Domino 4.6. The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Notes.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

DB2/2
OS/2
The following are trademarks of Lotus Development Corporation in the United States and/or other countries.

DataLens Notes HiTest
InterNotes Notes ViP
InterNotes Web Publisher Notes Mail
Lotus NotesPump
Lotus Notes Reporter NotesSQL
Lotus Notes Notes/FX
Lotus Notes ViP Phone Notes
Lotus @SQL Phone Notes Mobile Mail
LotusScript SmartIcons
Notes Video Notes

The following terms are trademarks of other companies:

Windows is a registered trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Macintosh is a trademark of Apple Computer, Inc.

HP is a trademark of Hewlett-Packard Company.

Sybase, Powersoft and Watcom are trademarks of Sybase, Inc. or its subsidiaries.

HockWare, VisPro/Reports, VisPro/REXX, VisPro/C and VisPro/C++ are trademarks of HockWare, Inc.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.
Preface

This redbook describes how to develop applications using Notes Designer for Domino Release 4.6, specifically for users accessing Domino from Web browsers.

The earlier chapters of the book introduce the concepts of Domino and how to create Web applications using Domino for those readers who may be familiar with Web application development, but not necessarily with using Lotus Notes/Domino for that development.

The book then describes how to use Domino design elements such as databases, forms, subforms, views, navigators, and agents to create Web applications. Other chapters illustrate how to include traditional Web development tools, such as HTML, Java and JavaScript, in Domino applications.

Security considerations for Web applications are also discussed, as well as how to access relational databases from Domino applications using LotusScript Data Object and ODBC.

How This Document Is Organized

The document is organized as follows:

- Chapter 1 - “Domino: Architecture and Configuration”
  This chapter provides an introduction to Domino architecture and its server configuration options that are particularly relevant to setting up Domino as a Web server.

- Chapter 2 - “Getting Started with Domino Web Applications”
  This chapter introduces the basic Web application design elements in Domino. Creating a basic database is covered along with setting up a Web site and creating a simple home page.

- Chapter 3 - “Designing Application Forms”
  This chapter describes how to create and use Domino forms to create Web pages, and which elements can be used on forms and how to use them.
• Chapter 4 - “Domino Views and Folders”
  This chapter introduces Domino Views and covers what they are and why they are useful. The chapter also details how to create views and tailor them for Web applications.
• Chapter 5 - “Domino Navigators”
  This chapter describes how to build navigator panes for the Web using the facilities of Domino 4.6.
• Chapter 6 - “Using HTML In Domino Design Elements”
  This chapter describes where and how you can add HTML to the different Domino design elements. The chapter also covers special considerations for frames, and how to reference a Domino object using a URL.
• Chapter 7 - “Domino Agents”
  This chapter describes where and how to use Domino Agents to launch actions from the Web browser or to automate tasks within Domino.
• Chapter 8 - “Domino Web Searching”
  This chapter describes how to set up Web searching with Domino. Both view level and Web site searches are covered. An example shows how to build a custom query form enabling users to construct a query by simply pointing and clicking.
• Chapter 9 - “Domino Security”
  This chapter covers the Domino security basics, particularly as it relates to Web applications. User authentication, access control lists (ACL) and Security Socket Layer (SSL) are covered.
• Chapter 10 - “Java Applets, Java Agents, and JavaScript”
  This chapter describes how to use Java applets and Java agents in Domino. The chapter also includes some examples of how to use JavaScript in Domino applications.
• Chapter 11 - “Accessing External Data from a Domino Application”
  This chapter describes LotusScript Data Objects and ODBC, which can be used to access RDBMs from within Domino.
• Appendix A - “Features to Avoid when Developing Web Applications”
• Appendix B - “Downloading and Setting Up the Chile Pepper Demo Site”
• Appendix C - “Table of CGI Variables Supported by Domino”
• Appendix D - “MIME Types and HTTPD.CNF”
The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Cambridge Center.

Fiona Collins is a Lotus Notes/Domino Specialist at IBM’s ITSO at Lotus Development, Cambridge, Massachusetts. She manages projects whose objective it is to produce redbooks on all areas of Domino. Before joining the ITSO in 1996, she provided technical support for Lotus Notes and the AS/400 for Lotus and IBM in the UK. (Fiona_Collins@vnet.ibm.com; Notes e-mail: Fiona Collins/CAM/Lotus@Lotus)

Flemming Hansen works as a Technical Specialist for EMEA Market Intelligence. His main role is the development and deployment of Notes and Web solutions to IBMers and Business Partners. (flemming_hansen@dk.ibm.com)

Tommi Tulisalo works for IBM Finland. Working as a Lotus Notes/Domino Application Development Consultant with Workgroup Solutions Finland, he provides Lotus Notes/Domino and Web solutions for IBM customers. (tommi_tulisalo@fi.ibm.com)

Filippo Vitiello is a Technical Specialist at the IBM Italia, Milano Italy (IBM Semea). He provides technical support for Domino and Web application development, specializing in integrating workflow elements. (Fil_Vit@It.ibm.com)

Xie Li Min is an Advisory Technical Specialist at the IBM China Network Computing Center. The team he works with provides cross-industry NC solutions and technical support on Internet application development. He has over ten years experience in the area of designing, deploying and managing Client/Server application environments. (xielm@IBMCN)

A number of people have provided support and guidance throughout the production of this book. In particular, we would like to thank Chris Reckling, Notes/Domino Product Manager for Application Development, and William Aboujaoude, Internet Marketing Manager, both from Lotus Development in Cambridge. In addition, we would like to thank the following people (in alphabetical order):

- Matt Geary, Lotus Cambridge
- Janna Joyce, Lotus Cambridge
- Kimberly Ryan, Lotus Cambridge
- Tim Turley, Lotus Cambridge
- Graphic Services, Lotus North Reading
Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address: redbook@vnet.ibm.com.

Your comments are important to us!
Chapter 1
Domino: Architecture and Configuration

Overview

This chapter describes the Domino 4.6 technology. The Domino architecture is introduced and its elements are discussed.

The main areas covered in this chapter are:

- Lotus Notes and the Web
- Domino Architecture
- Configuring the Domino Web server

Lotus Domino and the Web

Lotus Notes is ideally suited for Web and intranet environments. With the Domino technology, it goes beyond today’s HTTP server to bring the Web a rich set of proven functions from the Lotus workgroup world.

Domino is server technology that transforms Lotus Notes into an Internet applications server. It combines the open networking environment of Internet standards and protocols with the powerful application development facilities of Lotus Notes, enabling you to develop a broad range of business applications for the Internet and intranet. Domino can be thought of as:

- A Web server
- A Web application development environment
- An enabler for an alternate client for Notes: the Web browser

Beyond static information dissemination and browsing, Domino has robust and industry-leading technology in:

- Programmable object services
- Information push and pull replication
- Messaging/directory services
- Database integration
- Transaction system integration
• Security and RSA authentication
• Workflow, tracking, collaboration, and conferencing
• Search engines
• Document management and linking
• Scripting and agent support
• Network and mobile support
• Administration
• HTTP, HTML, CGI, and Java support

**Domino Architecture**

The Domino Web server is a Lotus Notes server task, enabling a Lotus Notes server to simultaneously act as an HTTP server. The Domino server automatically converts all Notes design elements and documents to Web pages, allowing users to access Notes databases either through a Notes client or a Web browser.

The Domino server converts all design elements and documents to Web pages. Notes views, forms, navigators, links and documents are translated into HTML on the fly for display on any Web client. Users with Web browsers, such as Netscape Navigator, Microsoft Internet Explorer, or Notes Web Navigator, are able to participate in discussions, send e-mail and do workflow activities that are housed in Notes databases.

The Domino server task provides logging, configuration, and security management features as well as on-the-fly, dynamic integration of Notes and Web environments. It speaks the HTTP protocol, converting HTML code on
the fly, thus enabling Web clients to communicate with Domino servers and service the requests.

The Domino server basically exposes Notes design elements as an extension of the URL interface by attaching the Notes element and a command on the end of the HTTP service request to access the Notes element. For example:


Through a URL interface, Domino examines incoming HTTP requests and responds in one of two ways:

- If the request is for an HTML file in the file system, Domino acts like an HTTP server displaying HTML documents.
- If the request is for a Notes database element, the Domino engine interacts with the Notes database to retrieve the appropriate information to return to the requesting HTTP process or save information in the database.

The Notes Designer gives you a single application development environment in which to create databases for Web users, Notes users, or both. This environment provides you with the ability to rapidly develop a broad range of business applications for the Internet and the intranet.

Because the Domino Web server merges Web server technology with Notes technology, Web site designers can use Domino to build applications that take advantage of core Notes functionality, such as replication, full-text search, application development, security, and workflow.

**Supported Platforms**

Supported platforms for Notes Workstation (Desktop client, Mail client, Designer for Domino) are:

- Windows 95
- Windows NT
- UNIX (Solaris/SPARC, Solaris/Intel, AIX and HP-UX)

Supported platforms for Domino Server (Domino server, Mail server) are:

- Windows 95
- Windows NT
- NT Alpha
- OS/2
- UNIX (Solaris/SPARC, Solaris/Intel, AIX, and HP-UX)
- NLM
Configuring the Domino Web Server

The Notes system administrator is responsible for setting up and configuring the Domino Web server, but it is helpful for an application developer to become familiar with some of the settings because several may affect the display of a Web application.

The HTTP Server section of the Server document in the Public Address Book maintains several settings pertinent to Web applications, including the server home page, graphic display formats, and character set choices.

HTTP Setup

Configuration settings for the Domino Web server are stored in the HTTP Server settings section of the Server document in your server’s Address Book.

<table>
<thead>
<tr>
<th>Basics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
</tr>
<tr>
<td>Bind to host name</td>
</tr>
<tr>
<td>DNS lookup</td>
</tr>
<tr>
<td>Default home page</td>
</tr>
<tr>
<td>Allow HTTP clients to browse</td>
</tr>
<tr>
<td>databases</td>
</tr>
<tr>
<td>Maximum requests per</td>
</tr>
<tr>
<td>a single connection</td>
</tr>
<tr>
<td>Number active threads</td>
</tr>
<tr>
<td>NOTE: The following parameter is no longer used in Domino 4.6. You should use it only for servers running versions prior to 4.6.</td>
</tr>
<tr>
<td>Minimum active threads</td>
</tr>
</tbody>
</table>

Host name (default=blank)
Enter the fully qualified host name that is returned to the browser. If your PC does not have a host name registered in a DNS, enter the PC’s IP address in this field.

DNS Lookup (default=Disabled)
Specify whether you want the Domino server to look up the DNS host name of the requesting client.

Note  If you enable DNS, the server works harder to perform host name lookups. This also causes storage of long host names for log file and log filter entries.

Allow HTTP clients to browse databases (default=Yes)
Specify whether you want to allow Web users to see a list of databases when they access the server. It is recommended that this be set to No for security reasons.
Default home page (default=default.htm)
Specify the default page file name you want the Domino server to load when a client accesses a directory not followed by an explicit page name.

Number active threads (default=40)
Specify the number of threads you want to have active at one time. If the maximum is reached, the Domino server holds new requests until another request finishes and threads become available. The more power your PC has, the higher value you should use. If your PC spends too much time on overhead tasks, such as swapping memory, reduce this value.

Mapping settings
The mapping settings are stored in the HTTP Server settings section of the Server document in your server’s Public Address Book and direct Domino where to look for each of its component files.

<table>
<thead>
<tr>
<th>Mapping</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Home URL</td>
<td>/?Open</td>
</tr>
<tr>
<td>HTML directory</td>
<td>domino/html</td>
</tr>
<tr>
<td>Icon directory</td>
<td>domino/icons</td>
</tr>
<tr>
<td>Icon URL path</td>
<td>/icons</td>
</tr>
<tr>
<td>Call directory</td>
<td>domino/cgi-bin</td>
</tr>
<tr>
<td>Call URL path</td>
<td>/cgi-bin</td>
</tr>
</tbody>
</table>

Home URL (default=/?Open)
Specify the URL you want Domino to return when users enter a site name directly, but do not specify an explicit directory or page name (for example, http://domino.lotus.com).

Using the default setting, /?Open, Domino displays a list of databases on the server. This is equivalent to the File - Database - Open command in Notes. To have Domino look for and return the Welcome page in the HTML directory, leave this field blank or specify /default.htm.

Note: If you have selected option No in Allow http clients to browse databases, default setting /?Open, results in an error.

Specifying a URL that begins with a / (slash) causes Domino to return the URL information directly to the browser. The browser still displays http://hostname.domain.com/ in the location text box.

Specifying URLs that start with a protocol such as http://host.domain.com/ causes Domino to send a redirected URL to the browser. That is, the browser performs an HTTP GET request on the specified URL. The information in the browser’s location text box then changes to what is specified in this field.
Examples:

/dominodisc.nsf/By+Author
/dominodisc.nsf/$defaultnav
http://myhost.domain.com/home/myhome.html

**HTML directory (default=domino\html)**
Specify the directory location for HTML files. The directory is relative to the Notes data directory unless a full path is specified. You can copy your existing HTML files to this directory. By doing this you are able to access them on the Web using Domino.

**Icon URL path (default=/icons)**
Specify the URL path to the Domino icons directory. Note that this path relates to URLs and not the file system. In general, you do not need to modify the icons fields. However, if you have an existing icons directory, specify the path to the directory here.

**Icons directory (default=domino\icons)**
Specify the directory location for the icons directory. The directory is relative to the Notes data directory unless a full path is specified.

**CGI URL path (default=/cgi-bin)**
Specify the URL path to the CGI programs directory. Note that this path relates to URLs and not the file system.

**CGI directory (default=domino\cgi-bin)**
Specify the directory location for CGI program files. The directory is relative to the Notes data directory unless a full path is specified.

**Operation Information Settings**
Operational information settings stored in the HTTP Server settings section of the Server document in your server's Public Address Book.

**Cache directory (default=domino\cache)**
Specify the directory for the Domino server to use as a cache. Domino uses this directory to store graphic images stored as GIF files and file attachments.

**Image conversion format (default=GIF)**
Specify the image file format you want Domino to use when converting image files. The options are GIF and JPEG.

**Interlaced rendering (default=Enabled)**
If you chose GIF as the image conversion format, specify whether or not you want Domino to render the GIF images in an interlaced format.
Progressive rendering (default=Enabled)
If you chose JPEG as the image conversion format, specify whether or not you want Domino to render the JPEG images in a progressive format.

JPEG image quality (default=75)
If you chose JPEG as the image conversion format, specify the percentage numeric value for the JPEG image quality. The range is 5 to 100 percent. The larger the value, the larger the file, and the better the image quality. The lower the value, the smaller the file, the less time it takes to transmit, and the lower the image quality.

Default lines per view (default=30)
Specify the default number of lines Domino uses to display a Notes view. Note that this setting affects every database on the Domino server.

Logging settings
Domino creates an access log and an error log. Both of these are specified in the Logging section of the server document. The files created are relative to the Notes Data directory with a timestamp suffix. A log filter can help you decrease the size of your log files.

Summary
Domino is ideally suited for the Web and for Web application development. Domino is server technology that transforms Lotus Notes into an Internet application server. Besides basic HTML technology, Domino offers the powerful application development facilities of Lotus Notes, exploiting the robust Notes technology, such as replication, messaging, high security, workflow, and searching.
Chapter 2
Getting Started with Domino Web Applications

Once your Domino server is set up, you can begin to design your application elements for a system to be used over the Web. This chapter describes how to create and manage Notes databases which are the basic ‘building block’ of Domino application development.

About Domino Web Databases

On traditional Web servers, the different pages and the associated compound elements are organized in hierarchical directory structures. When you issue an HTTP request to see a page, you are opening a new HTML coded file existing in a directory.

With Domino, your Web site is structured through Notes databases designed in the Notes Object store format. When you issue an HTTP request to see a page, you are opening a Notes element through a Universal Resource Locator (URL) command and Domino is translating it for viewing as a Web page.

A Web database, in its simplest form, is identical to a Notes database. What makes it a “Web” database is the viewing mechanism — a Web browser instead of a Notes workstation — and the fact that it resides on a Domino server running the HTTP server task. If you’re accustomed to Web design, the concept of a database may be unfamiliar; think of it as a storage area for related Web pages. A Web site is a collection of databases stored on one server.

A database can represent a small amount of documents used by only a few people, or it can be an enterprise-wide database containing thousands of documents. The upper limit size of a database is four gigabytes of data.

Every Domino Web application includes at least one database. Applications of a more complex nature use several databases. For example, a workflow application may route information between databases on one or more servers. Each Domino database contains three basic components: forms, fields and views. In addition, icons, help documents, navigators, agents, sections, actions, formulas, and scripts play an important role in giving an application sophisticated automation and processing power.
Documents in a Domino database can contain any number of objects and data types including text, rich text, numerical text, structured data, images, graphics, sound, video, file attachments, embedded objects, and Java and ActiveX applets. The object store lets your Domino applications dynamically present information based on variables including user identity, user preferences, user input, and time.

**Notes Designer**
The Notes Designer features an integrated development environment that gives developers and designers access to all features of the Domino server. The Notes Designer workspace is made up of several pages or folders where the Notes databases are displayed as icons. For each page of the workspace, you can specify a title and a color.

As a default, the last tab is assigned to the Replicator, which allows you to manage replication while working away from the office. You cannot rename the Replicator tab and you cannot delete that page.

**Glossary of Notes Designer Terms**
This glossary helps you understand Notes Designer terms within the context of a Web application development environment.

**About This Database Document**
The About This Database document is a design element that introduces a database. An About This Database document can be the default opening page for a database if you design it to open automatically when a database opens. You can include text, graphics, hotspots, HTML, applets, links, embedded navigators, and embedded views in an About This Database document.

**Database**
Every Web application uses at least one database to organize and store data and design elements. Database file names end with .NSF and reside in the Notes Data directory or in a subdirectory of a Domino server, which is a networked machine to which Web users have access.

**Documents**
Documents are individual Web pages that store the data for a database. Web developers create documents to provide information about the database or Web site. Web users create documents to participate in discussions and workflow processes. Documents can contain text, graphics, fields, applets, HTML, embedded navigators, and embedded views.
To create documents from the Web, users need a button that simulates the Notes Create - <form> menu command. You can build the button into a view, a navigator, or an About This Database document. Web users fill in the necessary information on the form and then click the button to save the document in the database.

**Fields**
A field in a document contains one piece of information, such as a person’s name, a comment, a number, a date, or the results of a calculation. When developers create fields on forms, they’re creating a data holder for that piece of information. In Web applications, fields also can contain Web pages, multimedia objects, graphics, and files. The form Web users choose when they create a document determines which fields the document contains.

**Forms**
Forms are design elements that give users a framework for entering new information in a database and for viewing existing information. They are similar to Web input forms that solicit information from users, but are used not only for inputting new data, but for displaying existing data. Most databases have forms for different kinds of information. For example, a discussion database usually includes a form to begin a new topic of discussion and another form to respond to topics. You can associate forms with other design elements, such as views and navigators, to create sophisticated Web effects.

**Navigators**
Navigators provide a graphical way for users to find documents or take actions in an open database without having to open views. Domino converts navigators to HTML image maps on the Web when you select the Web Browser Compatible navigator property. You can also embed navigators in forms or documents, so that they display together. A navigator can be an entry point for a database if you design it to open automatically when a database opens.

**Views**
A view is the entry point for opening and reading documents. Web users click a link to open a document. Most database designs use several views to organize and present documents in different ways. Think of views as HTML index files that have automatic updating functions, programming features, and graphics.

**Templates**
A template is a pre-designed database that you use to create new databases quickly. When you create a database and base its design on a template, you receive a number of forms, views, and navigators that determine how this
database looks and functions. Templates have file names that end in .NTF. Domino ships with several Designer templates that are helpful for developers, including:

<table>
<thead>
<tr>
<th>Database title</th>
<th>File name</th>
<th>Audience</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion - Notes and Web (R4.6)</td>
<td>discsw46.ntf</td>
<td>General Notes users</td>
<td>Electronic conference room. Features: Threaded discussion with built-in user profiles that allow automatic mailing of links to items of interest. Anonymous responses, archiving, and public/private threads. Also, multiple navigators, alternate view templates, and hotspot actions for Web browser users.</td>
</tr>
<tr>
<td>Doc Library - Notes and Web (R4.6)</td>
<td>doclbw46.ntf</td>
<td>General Notes users</td>
<td>Document storage. Features: Review workflow (linear only) and archiving.</td>
</tr>
<tr>
<td>Frameset (R4.6)</td>
<td>framew46.ntf</td>
<td>Notes/Web designers</td>
<td>Customizable HTML frame layouts for home pages.</td>
</tr>
<tr>
<td>Search Site</td>
<td>srchsite.ntf</td>
<td>Notes/Web designers</td>
<td>Searching across any number of databases on the Web or in Notes. Features: Simple and advanced search, ability to define database/directory/server/domain wide searches.</td>
</tr>
<tr>
<td>Web Pages (R4.6)</td>
<td>pagesw46.ntf</td>
<td>Notes/Web designers</td>
<td>Quick Web-site creation. Features: Automatically linked pages, basic server entry setup, and custom page-naming.</td>
</tr>
</tbody>
</table>

The Discussion - Notes and Web (R4.6) database template is used as an example in Chapter 3: Domino Forms. The Doc Library - Notes and Web (R4.6) database template is used as an example in Chapter 4: Domino Views and Folders. The use of the Frameset (R4.6) template is explained in Chapter 6: Using HTML in Domino Design Elements, and the use of the Search Site template is covered Chapter 7: Domino Web Searching. To learn more about how to use the Web Pages (R4.6) template to create your Web pages refer to the Quick Start sample database that ships with Domino.
Creating a Database

There are several different ways of creating a database. You can:

- Use an existing template
- Use an existing database
- Create a new database

Once the database is created, you can still modify most of the settings, and access additional settings, using the Database InfoBox. We will cover these options later in this chapter in the section Changing the Database Properties.

Using an Existing Template

Notes provides a series of written applications that can be used or customized for your own needs. There are many types of popular application templates, designed to reveal the underlying technology and development capabilities within Domino. Their main intent is not to be “out-of-the-box” applications.

If your application is identical with, or similar to, an existing template provided with Domino, the most convenient way to create a new database is to use the template as your starting point since most of the design work has already been done for you. You can also copy and paste individual design elements from the templates into your applications.

Listing Available Templates

To see the list of available templates:

1. Choose File - Database - New. The list box on the New Database dialog box lists several templates.
   
   **Tip** The shortcut is CTRL and N.

2. Click the Show Advanced Templates check box. The list box at the bottom of the list displays additional templates. The templates listed are stored on your local workstation.

3. Select any template you are interested in.

4. Click the About button to display the database help document. It summarizes what the database can be used for.

To see additional templates stored on a server:

5. Click the Template Server button.

6. In the Server field, select the server you want to access. Additional templates are listed.
Creating the Database
Follow these steps to create the database:

1. Decide if the database will reside on your local workstation or on a
server so that it can be used by several users at once.

2. In the Title field, specify a meaningful title. This title will appear on
the database icon on the user’s workspace.

3. In the File Name field, specify a file name for the database. You can
also use the file name that Domino automatically provides based on
the database title.

   **Note** The extension of a database file is NSF. The extension of a
database template file is NTF.

   The following figure gives you an example of a completed New
Database dialog box:

4. If the database is local, you can encrypt it. This is useful if the database
contains confidential data, or if your users have laptops that they will
take outside their business locations.
To specify encryption, click the Encryption button. Specify the appropriate level of encryption. The following figure shows what you can do:

After encrypting the database, the only one who can access the database locally is the Notes user who was specified in the Encryption window.

**Note** Although Notes users can’t access the encrypted database locally, they still can access the database if it is on the server. This also applies to Web users.

5. If you want to keep the database within a predefined size, click the Size Limit button found on the New Database dialog box, and select the appropriate size.

**Note** The default size limit is one gigabyte.

Domino will warn you or the administrator (if the database is on the server) when the size of the database gets close to the specified limit. Make sure that the database size you specify is correct as you will not be able to change this value later on.

The Size Limit box looks like this:

6. If you want your new database design to stay synchronized with the design template, click the Inherit future design changes check box on the New Database dialog box.
Copying an Existing Database

Copying a database is similar to starting from a template, except that you will almost certainly want to change part of the design.

Listing Available Databases
To list the available databases:

1. Choose File - Database - Open.
   Tip: The shortcut is CTRL and O.
2. If required, specify the appropriate server name in the Server field to list additional databases.
3. Click the About button to browse the Help document of the database.
4. Click the Add Icon button to add the database to your workspace.

Creating the Database
To create the database:

1. On your workspace, select the database icon you want to copy.
2. Display the database pop-up menu by clicking the right mouse button.
3. Choose Database Properties.
4. Choose the Design tab. Make sure that the InfoBox shows that the design is not hidden. It should look like this:

![Design InfoBox](image)

5. Close the InfoBox.
6. Keep the database selected and then choose File - Database - New Copy.

16 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
7. Select Local as the server name if you want to store the database on your local workstation. Select a server name if you want to store the database on a server so that several people can access the database.

8. Type a title for the database.

9. Type a file name with extension .NSF for the new database.
   
   **Note** The file name cannot be changed through Notes once the database is created. It must be unique on the workstation or server where the database is created.

10. Click the Database design only option button since you do not want to copy the documents that are stored in the database.

11. Deselect the Access Control List check box because it could prevent you from modifying the database design in the future.
   
   **Note** The access you have to the copy of the database depends on the access you have to the original database.

The following figure gives an example of the Copy Database dialog box:

![Copy Database dialog box](image)

You can optionally choose the following two features:

- To encrypt the database if it is a local database. This is useful if the database contains confidential data, or if your users have laptops that they use in public environments.
  
  To encrypt the database, click the Encryption push button and select the appropriate encryption level.

- To predefine the maximum database size. Click Size Limit and specify the appropriate size.
Creating a New Database

If no template or existing database meets your requirements, you can create a completely new database. This means that you will have to create all the design elements, such as forms, views, and fields. However, you can always copy existing elements from other databases and paste them into the new database.

2. Type a title in the Title field.
3. From the list of available databases displayed at the bottom of the window, choose the Blank option.
4. Click OK. The new database has been added to your workspace.
   The database view is displayed. You are ready to start the design of the database.

For more details on how to design a database, refer to the chapters covering Forms and Views respectively.

Changing the Database Properties

One of the features of Notes is its sensitivity to context. You are very often just one mouse-click away from the properties of the object you are working on. That is true for all design elements: Fields, Buttons, Attachments, Forms, Web elements, Columns, Views, and Databases.

Opening the Database Properties Box

To display the database InfoBox:

1. Display the database pop-up menu by clicking the right mouse button.
2. Choose Database Properties.

Tip  You can also click the following InfoBox SmartIcon to display the database properties InfoBox.
Specifying the Database Type, Replication, and Encryption

The Basics tab contains information on the database, such as its title, name, location, the replication settings, and replication history. The Basics tab looks like this:

1. To set the Database type, you can select one of the following values:

   - **Standard**: Notes database used most of the time.
     
     Library: A database with type Library is generally created from the Database library template. It is used to record and store information about the databases located on a Notes server or on a workstation. If it is located on the server, it contains a list of public databases. This provides an easy way for users to browse the list of databases available to them.

   - **Personal Journal**: This allows you to store personal information. This type of database contains local personal information. It has limited design elements and is meant for individual use.

   - **Address Book**: Creates a database based on the Notes Address Book format.

   - **Multi DB Search**: Is used to specify a database type of Search Through Multiple Databases which uses the SRCHSITE.NTF template. This type of database is used to configure searches among databases that have been designated to participate in Multi Database indexing by selecting the appropriate option on the Design tab of the database InfoBox. See Chapter 8: Domino Web Searching.

2. Click the Encryption push button to display a window that enables you to specify encryption for the local version of the database.
3. Selecting Web Access: Use JavaScript when generating pages allows Domino server to use JavaScript to generate Web pages. By selecting this option you can use, for example, multiple buttons on the form. See Chapter 3: Designing Application Forms.


Displaying General Database Information
1. Click the Information tab to display some general information, such as the size of the database and the number of documents stored.

2. Click the User Activity push button to display information related to user activity.

The Information tab of the InfoBox looks like this:

![Information Tab Example](image)

Specifying Print Options
1. Click the printer tab to specify options related to printing the document.

2. Use the icons listed under the Header and Footer option buttons to define the date and time, tabs, and page numbering.

3. You can also select the font, size, and style.

**Note** Printing properties don’t take effect if you are printing from the Web.
Specifying Database Design Properties

1. Click the Design tab to display or specify information concerning the
design of the database.

   The example in the figure below shows that the design of this database is
not hidden. Also, if the template that this database is based on is
modified, the database automatically inherits all the changes made to the
template.

2. If the database you are creating is a template, check the Database is a
template check box.

3. Specify a name for the template.

4. If appropriate, select the new template to be listed as an advanced
template. This indicates that the template should be customized only by
Notes developers.

5. Deselect the List in Database Catalog and Show in Open Database dialog
check boxes if the database is located on a server, contains sensitive data,
or you do not want users to be able to list its name.

6. Select Include in multi database indexing if you want the index to be
included in Multi Database Search Database site queries.

The following figure gives an example of the Design tab:
Specifying Launch Options
1. Click the Launch tab to define what Notes users and Web users will see when they first open the database. The dialog box looks different depending on your choice of action in the On Database Open drop-down list or in the On Web Open drop-down list.

![Launch tab screenshot](image)

2. Specify an option from the On Database Open drop-down list for Notes users opening the database. A wide variety of options is possible, such as:

   ![On Database Open options](image)

3. If you choose one of the two Navigator options, you need to select a navigator from the list of navigators available for the database.

4. Choose the Open designated Navigator in its own window option if you want the navigator to be displayed in a full screen. You would typically choose this option if the navigator consists of a large map or a workflow sketch.

5. Specify an option from the On Web Open drop-down list for Web users opening the database. A wide variety of options is possible, such as:

   ![On Web Open options](image)

---

22 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
6. You can specify the properties of the preview pane by clicking on the Preview Pane Default button. You are presented with the following choices. Click on the most appropriate for the user.

![Preview Pane](image)

**Note** This option doesn’t take effect on the Web, because Web users don’t have the preview pane.

**Specifying Full-Text Indexing**
1. Display the Full Text tab to create, update, or delete a full-text index, which allows for fast retrieval of documents.
2. Specify the update frequency as required.

**Minimizing the InfoBox**
You can keep the InfoBox open while you are working on the design of the database. You will notice that the contents of the InfoBox is refreshed while you are progressing with the design.

You can also reduce the size of the InfoBox by double-clicking on its title bar. Double-clicking on the title bar or clicking on one of the tabs will restore it to its full size.
In the following example, the InfoBox is minimized and located in the upper right-hand corner of the Designer workspace.

Setting Up Your Web Site

A Domino Web site is an organized collection of information stored in one or more databases that reside on a Domino server. As you plan a Web site, think about the types of databases, views, navigators, and forms you’ll need to build the site.

**A single-database Web site**
A simple Web site includes a single database, an About This Database document that displays each time the database opens, one or two forms and views, and possibly a navigator. A simple Web site works well for sites that have only one or two main functions. To build a simple Web site, Use the Web Pages (R4.6) template (pagesw46.ntf) to create a Web site that has a home page, a Webmaster page, an upcoming events page, and a discussion forum for site visitors.

**A multi-database Web site**
A large Web site includes several databases, each with its own forms, views, and navigators, bound by document, view, and database links that make the multi-database organization invisible to users. The databases at such a site may also contain agents that process documents automatically. This approach works well for sites that have several functions or that contain interrelated databases.
A multi-database Web site might look like this in the Notes Designer workspace:

When accessing this application over the Web, you start off at a home page. The Domino Web site home page is displayed below. This page is actually the About Document of the database named ‘Domino’ displayed in the previous figure. It contains links to the other databases in the workspace.

Since this will be the default page for all users to go to, you should also set the page to display automatically even when the user accesses the server directly by its site or host name. To do this you must set the ‘Home URL’ setting in the server document’s HTTP Settings section.

Chapter 2: Getting Started with Domino Web Applications  25
In the next section we show you how to define a home page to display on your Web site using the Domino server.

**Designing Your Home Page**

A home page is a Web site element which guides Web users through your site’s information and application(s). It also serves as the site’s focal point. Although you can use an HTML file as a home page for your Domino site, the Notes Designer environment lets you create attractive home pages quickly.

Create an About This Database document or a regular document with links to views, documents, or navigators in the same database or other related databases at the site. Designers often choose to store the home page in a database whose only purpose is to store the home page, or in a database used for other, but related, purposes, such as user registration for the site.

To set the home page to launch automatically, select the Database Properties InfoBox, click the Launch tab, and select the appropriate option under On Web Open. For example, in the screen below the “About database” document has been selected:

You can specify these launch options, not only for your home page database, but for other databases as well.

**Launch Designated Doclink**

In the On Web Open selection you can also specify that Domino launch a designated doclink. You could use this, for example, if you have created your home page as a document. To do this:

1. Copy a doclink to the clipboard by selecting the desired document from a view and selecting Edit - Copy As Link - Document Link.
2. In the Launch options tab of your database select Open designated
doclink in the On Web Open section.

3. Click Paste Doclink.

When the user accesses the database over the Web, Domino will open the
document specified here.

Accessing a Domino Site

In order to access Domino Web sites, that is, Notes databases, users have to
have the authority to do so. These authorities are set in the Access Control
List (ACL) of the Domino database. Access levels assigned to users control
which tasks they can perform on that database. There are seven main levels
of access you can set up for individual users, user groups, servers, server
groups, or mixed groups of users and servers.

<table>
<thead>
<tr>
<th>Access Level</th>
<th>Description</th>
</tr>
</thead>
</table>
| Manager      | Users with Manager access can modify ACL settings, encrypt a
database for local security, modify replication settings, delete a
database, and perform tasks permitted by no other access level.
Managers can also perform all tasks allowed by other access levels.
Notes requires each database to have at least one Manager. It’s best
to assign two people Manager access to a database in case one
manager is absent. |
| Designer     | Users with Designer access can modify all database design
elements (fields, forms, views, public agents, the database icon,
Using This Database document and About This Database
document), can modify replication formulas, and can create a full
text index. Designers can also perform all tasks allowed by lower
access levels. Assign Designer access to the original designer of a
database or to a user responsible for updating the design after a
database is in use. |
| Editor       | Users assigned Editor access can create documents and edit all
documents, including those created by others. Assign Editor access,
for example, to a user responsible for maintaining all data in a
database. |
| Author       | Users assigned Author access can create documents and edit
documents they create. Assign Author access to allow users to
contribute to a database but not edit documents created by others.
When possible, use Author access rather than Editor access to
reduce Replication or Save Conflicts. |

continued
Access Level | Description
--- | ---
Reader | Users assigned Reader access can read documents in a database but cannot create or edit documents. For example, assign Reader access to users who must be able to read the contents of a reference database such as a company policies database. Anyone with at least Reader access to a database can create personal agents in the database if the database manager selects the ACL option “Create personal agents.” However, users can only run agents that perform tasks allowed by their access levels. For example, someone with Reader access can create a private agent that deletes documents, but the agent won’t delete documents when the user runs it.
Depositor | Users assigned Depositor access can create documents but can’t see any documents in the database views, even the documents they create. For example, assign Depositor access to allow users to contribute to a mail-in database or to a database used as a ballot box.
No Access | Users assigned No Access cannot access the database. For example, assign No Access as the default access to prevent most users from accessing a confidential database.

In addition to adding people or groups to a database ACL and assigning them an access level, you can also fine-tune their access by selecting or deselecting certain sub-access options.

The table below summarizes these sub-levels:

<table>
<thead>
<tr>
<th>Access Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Documents</td>
<td>Select this option to allow Authors to create documents. Managers, Designers, Editors, and Depositors are permanently assigned this access. You normally select this option for all users with Author access; you may want to deselect this option after a period of time to prevent Authors from adding any more documents but to allow them to read and edit ones they’ve already created. By default, the Create documents option is not selected for new Authors that you add to the access control list.</td>
</tr>
<tr>
<td>Delete Documents</td>
<td>Select this option to allow Managers, Designers, Editors, or Authors to delete documents. Authors can delete only documents that they created or if the document contains an Author Names field, they can delete documents if their name or a group or role that contains their name appears in the Author Names field.</td>
</tr>
</tbody>
</table>

continued
<table>
<thead>
<tr>
<th>Access Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Personal Agents</td>
<td>Select this option to allow Designers, Editors, Authors, or Readers to create personal agents. Managers are permanently assigned this access. Since personal agents on server databases take up server disk space and processing time, you may want to deselect this option to prevent some users from creating them. Note: A Notes administrator can use the Agent Manager Restrictions section of a server document to prevent people from running personal agents on a server; people denied this server access can’t create personal agents on the server, regardless of the ACL setting.</td>
</tr>
<tr>
<td>Create Personal Folders/Views</td>
<td>Select this option to allow Editors, Authors, or Readers to create personal folders and views in a database on a server. Managers and Designers are permanently given this access. Personal folders and views created on a server are more secure and available on multiple servers. Also, administrative agents can operate only on folders and views stored on a server. If this option is not selected, users can still create personal folders and views but the views and folders are stored on their local workstations. Deselect this option to save disk space on a server.</td>
</tr>
<tr>
<td>Create Shared Folders/Views</td>
<td>Select this option to allow Editors to create shared folders and views. Managers and Designers are permanently assigned this access. Deny this access to save disk space on a server and to maintain tighter control over database design.</td>
</tr>
<tr>
<td>Create LotusScript Agents</td>
<td>Select this option to allow Readers, Authors, Editors, or Designers to create LotusScript agents. Managers are permanently assigned this access. Since LotusScript agents on server databases have the potential to take up significant server processing time, you may want to restrict which users can create them. Note: A Notes administrator can use the Agent Manager Restrictions section of a Server document in the Public Address Book to prevent people from running restricted and/or unrestricted LotusScript agents on a server. If you select “Create LotusScript Agents” for a name in the ACL, a Server document can nevertheless prevent people assigned this access from creating LotusScript agents.</td>
</tr>
<tr>
<td>Read Public Documents</td>
<td>This option is to support the Calendaring and Scheduling delegation function. It enables an application developer to assign reader access to a database without giving the user the full Reader Role. A Notes application designer can add a $PublicAccess field to a document. When this field is in the document, it means that the document can be read by any user with Public Reader access and can be modified by anyone who has Public Writer access. A Notes application designer can check the [x] Available to Public access user’s check box on the Forms and Views Properties InfoBoxes to allow the object to be visible and usable by public access users.</td>
</tr>
<tr>
<td>Write Public Documents</td>
<td>See Read Public Documents above.</td>
</tr>
</tbody>
</table>
**Note** Some of these options are disabled depending on the level of access you have given a user. For example, if you give someone Manager access, you will not be able to remove their ability to create documents.

In order for Web users to access a database you have to add the name Anonymous to the ACL of the database. This will allow Web users to access a database anonymously. If you don’t want to allow anonymous access, you should create a registration application to allow users to register and then access databases with their own identity. If you don’t have name Anonymous in the ACL of the database, Web users get the Default access to the database.

To learn more about ACLs, how to set up access for Web users, and user registration applications, see Chapter 9: Domino Security.

---

**Summary**

With Domino, your Web site installation and design is structured through Notes databases. Web pages and design elements are stored in the Notes object store, which acts as a storage area for all the Web pages making up the Web site. When Web users access a Domino site they are accessing Notes databases. Domino then translates the documents and design elements to HTML on the fly.

Every Domino application includes at least one database. Each Notes database contains three basic components: forms, fields, and views. The Notes Designer is used to create and design the Web site databases and their components.

Access to a Domino site is managed through the Notes database Access Control List. Access levels assigned to users control which tasks they can perform on individual databases.
Chapter 3
Designing Application Forms

In this chapter you will learn about Domino forms — what they are, the different types of form and how to create, design and modify them. This chapter also discusses the design elements of the form, such as fields, subforms, buttons, images, Web elements, horizontal rules, and computed text.

In this chapter you will also learn how to create Web pages using forms and how to lay out Web pages using Notes Designer. The chapter also covers how to show different information to Web users and to Notes users and how to return information to users using HTML.

When you have completed this chapter, you will also know how to capture CGI variables into a form.

Using Forms

The form is the skeleton you provide to users to create documents and to view existing documents in a Notes database. Users can enter data by typing it into fields in the form. There is always at least one form in a database. Most often though, business applications have more than one form, each form being targeted to the type of information the user wants to save in the database.

The form contains all the design elements: fields to store the user’s information, static text, buttons, sections, and subforms that help the user gain access to the information.

You can associate forms with other design elements, such as views and navigators, to create sophisticated Web effects.

When the user creates a new document through the Web using a Web browser, Domino creates an empty document from the form design which the designer has created. After completing the document, it is saved into the Notes database.

When the user opens the document through the Web with a Web browser, Domino automatically converts the Notes document to an HTML page. The HTML page then contains design elements, for example fields, text, graphics, and the information users have entered.
To create a new form, choose Create-Design-Form. Or, you can copy and paste a form from the Design Form pane and then customize it to suit your needs.

In this next section we will explain how to edit the properties of an existing form, the Main Topic form from the Discussion template which comes with Domino, to give you an idea of what is available.

This chapter also describes, later on, how to create Web pages using forms.

**Specifying Form Properties**

The Form InfoBox contains all the information related to forms.

To look at the form properties, do the following:

1. From the standard navigator expand the Design option.
2. Choose Forms.
3. To create a form select Create-Design-Form.
4. Click the Properties SmartIcon.
5. In the InfoBox displayed, click on the Properties for: combo box and select Form. An InfoBox is displayed that allows you to set the properties of the form. It consists of six tabs:
   - Basics
   - Defaults
   - Launch
   - Background
   - Print
   - Security

**Using the Basics Tab**

The Basics tab stores general information about the form.

1. In the Form name field, specify a name and an alias for the form.
   
   By default, the form name appears as an item in the Create option on the menu bar. Since it is the name the user sees, make it as meaningful as possible.

   It is recommended that you create an alias for each name. This is the name you will use in your code. Specifying an alias enables you to leave your code unmodified if the user requests to have the name of the form changed, for example.

2. Try to keep the first character unique. This is because Notes will use the first unique character as an accelerator key under OS/2, Windows, and UNIX (Macintosh does not use accelerator keys).
Note  The accelerator key is used only by a Notes client; the feature is not available from a Web browser.

If desired, add an underscore next to one of the letters of the name to be used as an accelerator key.

3. Specify the form type. This form is a Document type, which means that it is a main document. Another type of document, the Response type, is linked to a Document type of document, and it cannot exist without this parent. A third type is the Response to Response type, which adds a third level to the document hierarchy.

4. If you want to include the form in the Create option on the menu bar, you can select:
   - Create menu if there is only a small number of forms. Up to nine forms can be displayed in the Create option.
   - Create-Other dialog if there are many different forms. Putting the least used forms under the Create-Other dialog option is recommended.

   Note  Create options are not applicable for Web users, because Web users don’t have the Create menu.

5. Click the Search Builder check box to add the form to the list of forms that users can search.

6. In the Versioning field, specify whether or not you want version control. The following options are possible:
   - None
   - New versions become responses
   - Prior versions become responses
   - New versions become siblings
7. You can decide to keep track of the different versions of the document that have been created. The current version can be set as a response to the previous version of the document, or vice versa.

8. Check the Anonymous Form check box if you want authors or editors to anonymously put documents based on this form into the database.

You only have to use the Anonymous Form property to allow Web users to create documents anonymously if you require users to log in to the database. This is because Web users don’t automatically have Notes user identities.

To allow Web users to create documents anonymously, assign Author access to the name Anonymous in the access control list of the database.

Anonymous forms could be used for an opinion survey, for example.

**Note** Documents created with an anonymous form do not contain the $UpdatedBy field but have a $Anonymous field with a value of 1.

Of course, you need to make sure that the author’s or editor’s name does not appear in any other field of the form.

9. If desired, check the Merge replication conflicts check box. Notes then merges conflicting edits into a single document whenever possible. If two users edit different fields in the same document, Notes saves the changes to each field in a single document.

However, if two users edit the same field in the same document, Notes saves one document as a main document and the other as a response.

**Note** Checking Merge replication conflicts has no effect on the Web, because Domino can’t merge replication conflicts made by Web users.

**Using the Defaults Tab**
The Defaults tab lets you specify details regarding the usage of the form.
1. Notes uses a default form to open documents whenever their associated forms have been dropped from the database design. You should select this option for the main form of the database.

2. Leave the Store form in document check box deselected.
   
   You must store the form into the document if, for example, a user who has no access to the database receives a document but is not able to access the design of the form used to create the document.

   **Note** Selecting the Store form in document option increases the amount of disk storage required to store each document based on the form.

   **Note** On the Web, storing forms in documents is supported only in Read mode, but do not use for documents that need to be created or edited on the Web.

The following features are not applicable or available on the Web, or are only partially supported from a Web browser:

- The Disable Field Exchange check box, which is normally left deselected to enable data exchange with Notes/FX compliant applications is not supported on the Web.

- The option to Automatically refresh fields is not supported on the Web.

- In the On Create section, if you select the first check box the data contained in the fields of the parent document is copied to the fields of the document or response document. This is supported for Rich Text fields on the Web, but not for other types of fields.

- The Inherit Entire Selected Document Into Rich Text Field option, which defines how the fields of the parent document are displayed in the response document, is supported for Rich Text fields on the Web, but not for links or collapsible rich text fields.

- The Show Context Pane check box has no effect on the Web.

- The Present Mail Send dialog check box is not supported on the Web.

- For Web Access: Treat Document Contents as HTML. This option has no effect on the Web. See Chapter 6, Using HTML in Domino Design Elements, for more information.
Using the Launch Tab
The Launch tab enables you to specify what happens when the document is opened.

**Note**  Launching features are not available for Web users.

1. In the Auto Launch field, specify the type of action to take place when the document is opened. These are the available options:
   - None
   - First Attachment
   - First Document Link
   - First OLE Object

2. Before you leave the Launch tab, make sure that None is selected in the Auto Launch field, because launch features are not supported on the Web.

Using the Background Tab

1. Click the Background tab to specify the background options on the form.

2. Specify the background color for the form using the Background Color drop-down combo box.
3. If desired, click the Paste Graphic button to paste a graphic image into the form. If the image is smaller than the form, Notes tiles the image to conform to the size of the form.

**Note** You need to copy an image to the clipboard before you click the button.

**Tip** Keep in mind that the cursor could be difficult to see on some displays if you choose a color such as yellow, for example.

4. You can also import a graphic image into the form by selecting the image and clicking the Import Graphic button. Supported graphic formats are BMP, GIF, JPEG, PCX and TIFF 5.0. If desired, you can select Hide graphic in design mode. Select Hide graphic on 16 color displays, if your images have more than 16 colors.

**Note** If the image doesn’t fill the entire page, it’s tiled automatically.

**Tip** It’s better to import than paste a graphic into the background, because imported graphics have better quality than pasted graphics. It’s also easier to import a graphic than copy the graphic to the clipboard and then paste it in.

5. By selecting Allow users to override background properties, you give users a chance to change background properties of the document the designer has made with this form.

**Using the Print Tab**

1. Click the print tab to specify options related to printing a document based on the form.

2. Use the icons listed under the Header and Footer option buttons to define the date and time, tabs, and page numbering.

3. You can also select the font, size, and style.

**Note** Printing properties don’t take effect if you are printing from the Web.
Using the Security Tab

Use the security tab to define which users or user groups are authorized to use the form.

1. Deselect the All readers and above check box. This activates the small button to the right of the list.
2. Click the button. A window is displayed that allows you to select users and groups from the different address books you have access to.
3. Specify who can create documents with this form. The default is All authors and above. If required, deselect the check box and click the small button to the right of the list. A window is displayed that allows you to select individual users and groups.
4. If required, select Disable printing/forwarding/copying to clipboard. This makes it more difficult for users to distribute the documents created with this form to other users. However, it is recommended that you limit this option to confidential data.
   **Note** Selecting this option does not prevent the usage of other software to copy data to the clipboard.
5. Select Available to Public Access users, if required.

Giving the Form a Title

You can define actions to be performed when users trigger events as they compose, edit, or browse a document created with a form. The events can be defined in the programming pane at the bottom of the Form Builder window.
One event you can define is the window title. When you design a form, make sure you give it a title. The title will then appear on the Notes title bar when a document is created or edited based on the form in Notes and on the browser title bar when a document is created or edited in the browser.

**Note** You should pay special attention to creating a window title when the application is to be used through the Web because some of the functions used in Notes don’t work on the Web.

Here is an example of a window title that works when used from either a Notes client or a Web browser:

```plaintext
@if(@isNewDoc;"New Document";"Document: " + Subject + " created by " + @Name([CN];@UserName))
```

*Subject* is a field of the form. *@UserName* is a function that returns the whole name of the user. *@Name([CN];@UserName)* returns common name part of the user.

If the document is a new document (@isNewDoc), the title is set to New Document. If the document already exists in the database, the title is set to Document: *Subject* created by *Username*.

**Form Events**

In the programming pane, you can define additional actions. You can code processes to be performed if the document is accessed from a Notes Client:

- When the document is initialized
- Before or after the document is opened (Queryopen and Postopen)
- Before or after the document changes its mode from browse to edit or from edit to browse (Querymodechange and Postmodechange)
- When the contents of document fields have been recalculated (Postrecalc)
- When the document is saved (Querysave) or closed (Queryclose)
- When the document terminates

The following events are the only ones available for Web users accessing the form:

- Before the document is opened (Webqueryopen)
- Before the document is saved (Webquerysave)

With Release 4.6, both Web events have the formula:

```plaintext
@Command([ToolsRunMacro];"<Your agent goes here>")
```

in the programming pane. You can change the text `<Your agent goes here>` with the name of the agent you want to run.

**Chapter 3: Designing Application Forms  39**
**Note** In Release 4.5 and earlier, this function was achieved using `$\$QueryOpenAgent`. This method is still supported in Release 4.6, as well as the new command.

See also Chapter 7: Domino Agents.

A form also has an event called HTML Attributes. Here you can change the background properties. The following table shows you some of the background properties you can use.

<table>
<thead>
<tr>
<th>Body Tag</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGColor</td>
<td>The color of the background</td>
</tr>
<tr>
<td>Text</td>
<td>The color of the document’s text</td>
</tr>
<tr>
<td>Link</td>
<td>The color of the document’s hotspots</td>
</tr>
<tr>
<td>Vlink</td>
<td>The color of the visited links</td>
</tr>
<tr>
<td>Alink</td>
<td>The color of the active link (the color it appears while the user is selecting it)</td>
</tr>
<tr>
<td>Background</td>
<td>Background image</td>
</tr>
</tbody>
</table>

You can set the color of the desired tag by typing

```
"TAG="XXXXXX"
```

where TAG is the desired tag and XXXXX is the color, to the HTML Attributes event. For example:

```
"BGColor="green"
```

sets the page background to green. Colors you can use are from the original Windows VGA palette. You can also use a 6-digit hexadecimal code.
Creating a Field

In this next section we will take a look at how to create a field in a form as well as how to change the properties of the field.

We will add a field of type keyword in the form. This field will demonstrate how to use keyword fields on the Web. The keyword field will let you choose one keyword. Selecting Keyword will then change the color of the document’s background on the Web.

1. From the standard navigator, choose Design, then Forms, and open the Main Topic form listed in the view pane.

2. Type the static text as shown in the following example. The static text is Background Color.

3. Under the static text, create the field Color. To do so, choose Create - Field.
   
   Tip   You can also display a pop-up menu by clicking the right mouse button and choosing Create - Field.

4. On the InfoBox for Field, type a name for the field, for example, Color. The new name is now also shown in the Define field in the programming pane.

5. In the Type field, choose Keywords.

6. In the field next to Type leave Editable.
7. In the Choices field leave Enter choices (one per line).
8. In the Choices list box type Red, Green and Blue. To separate values, press Enter.
9. In the programming pane leave the Default Value empty.
10. Save the modified form by pressing the Esc key, and confirm that you want to save the form.
11. Close the information box that is displayed.

Performing a Test Run

To test your modification from a Notes client:
1. From the standard navigator, select All documents.
2. In the view pane, click the New Main Topic button on the Action Bar.
3. In the document, you see that there is an additional field. The document will look like this:

4. Type in a subject and select a color for the background.
5. Click Close button on the Action Bar and save the document.

The document will now appear in the list of documents.

To test your modifications from a Web browser interface you can either use the Preview in Web browser option in Notes or:
7. Find your application.
8. Click the New Main Topic button.
9. In the document you can see that there is a new field. The document will look like this:

10. Type in the subject and choose the color for the background.
11. After you have completed the form, click Submit.

**Note** If you want to preview your form without creating a document in Notes, you can use the Preview in Web browser function in the Notes client. To do this choose Design - Preview in Web browser. If your database is on a Domino Server, Notes will then use the browser specified in your location document to preview the form. If your database is on a local machine, Notes will start the Local Web Preview Process.

**Note** You can preview only local databases that are in the Notes data directory.

**Field HTML Attributes**

When you click on the events on the programming pane you will notice that field has an event called HTML Attributes. In this event you can specify attributes for this field, which then affects how the field is shown on the Web.

Some of the attributes are very useful, because you can’t specify them in the Field Properties InfoBox in Notes. You can specify, for example, the size of the field or the maximum number of characters that are accepted as input. The attribute for specifying the size of the field is SIZE. In the HTML Attributes event you type, for example, “SIZE=30” and the field will then be
30 characters wide. The attribute for specifying the maximum number of characters is MAXLENGTH. These two attributes are appropriate only for fields of type text.

It’s also possible to write JavaScript to HTML Attributes. For example, you might want something to happen when a user selects a value in the keyword field.

Earlier we created a Color field for changing the background color. We will now make some further updates to that form to add new functions:

1. From the standard navigator, choose Design, then Forms.
2. Double-click the Main Topic form in the view pane to open it.
3. Click on the Color field.
4. Select the HTML Attributes event from the programming pane.
5. Type the following JavaScript code:

   "OnChange=""document.bgcolor=(this.form.Color.options[this.
   form.Color.selectedIndex].text)"

   The code has to be enclosed in quotes. OnChange is a method which occurs when the value in the field is changed. Object document represents the current Web page and it has a property bgcolor which is the background color of the page.

   this.form.Color.selectedIndex

   returns an integer specifying the option selected in the field. Now we know which of the options is selected, we can use it to return a text string with

   this.form.Color.options[this.form.Color.selectedIndex].text

6. After you have typed code the design should look like this:

   ![Design view of the Color field with JavaScript code]

   **Note**  Unlike HTML and LotusScript, JavaScript is case sensitive.

7. Preview form in your browser.
8. When you select the value in the Color field the background color of the page should change.
Field Properties

Now that you have created a field, we are going to look at some of the properties of fields contained in the Main Topic form.

The WebCategories Field

Now, let’s look at the WebCategories Field

1. From the standard navigator, choose Design, then Forms.
2. Double-click the Main Topic form in the view pane to open it.
3. Double-click the WebCategories field. The Field InfoBox is displayed. It looks like this:

The Basics Tab

On the Basics tab, Notes displays the field format. This field is of type Editable Keywords field. There are different ways of displaying the list of keywords from which users can make their selections. In our example, an @DbColumn formula checks all documents in the current database for categories and retrieves them for display in a keyword list.

The Basics tab also shows how the data is actually put into the field. The following types of field are available:

- **Editable**: The user enters the data, or the data is created when the user selects a button performing a formula or script written by the developer.
- **Computed**: The field is computed each time the document is created, edited and saved.
- **Computed when composed**: The field is computed only when the document is created. This type of field is especially useful for storing the name of the author of the document, the creation date or a document reference number.
Computed for display: The field is computed each time the document is opened in browse or edit mode. The contents of the field are visible only while the document is open. It is not saved into the database and is not visible in a view.

For example, this type of field is used to display the current time and date or work variables, such as the server name where the database is stored.

There are also two check boxes:

1. Allow values not in list. By choosing this option you give the user a chance to enter a value which is not on the list.

   Note This feature is not available for Web users, but there is a rather simple way to do this. We are going to see it later in this chapter.

2. Allow multi-values. This option allows users to select multiple values.

Keyword Options

1. Go to tab 2 of the InfoBox. It looks like this:

2. In the Interface section, you can specify how the keywords will be displayed. Three options are available:

   - Dialog List

      The keywords are displayed in a dialog list. If you want to enable users to specify additional keywords not listed, check the Allow values not in list check box on the Basics page of the InfoBox.

      If you choose Dialog List and look at it on the Web it looks like this:
**Note** The check boxes, Don’t display entry helper button, Refresh fields on keyword change, and Refresh choices on document refresh are not applicable to Web users.

- **Check box**
  This is a multiple choice field. If you want the check boxes displayed vertically, leave 1 in the Columns field. If you want the check boxes displayed as a table, set the number of columns to a value greater than 1.
  To display all check boxes on one line, set the number of columns to the number of check boxes available.
  If you choose Check box and look at it on the Web it looks like this:

   - Red
   - Green
   - Blue

**Note** Choosing frame type has no effect on the Web.

- **Radio Button**
  Only one radio button can be selected at any given time. The radio buttons can be displayed vertically, horizontally, or as a table.
  If you choose Radio Button and look at it on the Web it looks like this:

   - Red
   - Green
   - Blue

**Note** Choosing frame type has no effect on the Web.

Next, take a look at the programming pane:

The Event combo box enables you to define other actions:

- **Input Translation**: It can be used to modify the data entered by the user, to trim blanks, and to change users’ names into uppercase or proper case.

- **Input Validation**: This requires a choice from a list, which is done using an @If formula. The same validation can be written using LotusScript. To use LotusScript, you need to choose the Exiting option in the Event combo box. The Script option button is automatically selected.
Options Tab
Go to tab 3 of the InfoBox. It looks like this:

This tab enables you to specify:

- Field help
  Note  Field help is not available for Web users.
- Whether the entry field will have the initial focus when the form is opened. You must specify this option if you want to place the cursor in an entry field that is not the first one in the form. You can also use this option if you want to paste data in a particular entry field before placing the cursor in its final position.
  Note  This option is not available for Web users.
- Multi-value separators
- Security options, such as Enable encryption for this field
  Note  Security options are not available for Web users.

Fonts and Colors Tab
Tab 4 of the InfoBox lets you specify fonts and colors for the field data. The tab looks like this:
**Alignment Tab**
Tab 5 lets you specify the alignment of the field. For example, you should use this option if you define a field to be used as the title. If you choose to align it in the center of the form, it will stay in the center independent of the screen resolution used.

**Print Option Tab**
1. Go to tab 6 to set the printing defaults related to paragraphs. For example, you can select to keep a paragraph on one page, or to insert a page break before a paragraph.
2. Specify the tab settings for the fields. You must use that option if you have multiple fields on the same line and want to keep them in the same place where you have put them in the form.
   Although you can set the tabs manually, it is better to set them directly in the form using the ruler.

**Note** Printing properties don’t take effect when printing from the Web.

**Hiding the Field**
1. Go to the Hide When tab. It looks like this:

   ![Hiding the Field](image)

   2. Several check boxes are available to hide the paragraph on predefined conditions. Or, you can specify other conditions using an @Function.
   The field is hidden when the document is opened for reading. It is also hidden if the document is accessed from Notes (4.6 or later).

   **Note** The InfoBoxes of all the design elements found in a form provide a tab that allows you to specify hide-when conditions.

3. Return to the standard navigator.
Keywords Field on the Web

In the Basics tab of the Keywords field, selecting Allow values not in list doesn’t have any effect on the Web. Here is an easy way to provide Web users with a similar feature:

1. You should still see the Form Builder window of the form Main Topic. If not, from the standard navigator, choose Design, then Forms, and open the Main Topic form listed in the view pane.

2. Double-click the WebCategories field. The Field InfoBox is displayed. You can see that Allow values not in list is not checked.

3. Click the NewCats field. The Field InfoBox is still displayed. You can see that the field is an editable text type of field.

4. Go to the Hide When tab.

5. You can see that this field is hidden from Notes Clients. This field is made for Web users to provide a way to enter values not in the list of the WebCategories field.

6. Click the Categories field. Select Input Validation event from the programming pane.
   The validation formula looks like this:

   \[
   \text{FIELD WebCategories := @If(NewCats = "": WebCategories; @Trim(@Unique(Categories : WebCategories : NewCats)))};
   \]

   \[
   \text{FIELD NewCats := @DeleteField;}
   \]

   @Success

   WebCategories, NewCats and Categories are fields of the form. @Trim is a function which removes leading, trailing, and redundant spaces from a text string. @Unique removes duplicate values from a text list.

   If the field NewCats is empty, this formula will ensure that the WebCategories field will hold its value. This happens when a user has chosen a value from the list, instead of entering a new one. If the field NewCats is not empty, field WebCategories will get a unique list from values of fields Categories, WebCategories and NewCats. Finally the field NewCats is deleted and the formula will return 1 (the boolean value true) to indicate that the value meets the validation criteria.

7. Return to the standard navigator.
Sharing Design Elements with Subforms

Subforms provide a way to share fields between groups of design elements. All design elements that are added to forms can also be put into subforms. This includes:

- Static text and pictures
- Fields, whatever their type and format
- Hotspots as buttons or links
- Tables
- Action Bar
- Web Elements

When you modify an existing subform, the changes are immediately reflected in all the forms that use the modified subform.

**Note** You cannot create subforms within subforms or inside tables.

A subform is provided with the Discussion template. You can work with its design in one of the following ways:

1. From the standard navigator, choose Design, then Subforms. The list of subforms is displayed. Double-click on the SharedResponseHeader subform. This subform is used to share hidden fields which are common to both forms. The Subform Builder window is displayed.

2. Or, from the standard navigator, choose Design, then Forms. The list of forms is displayed. Double-click the Response or Response To Response form. Once the form is open, double-click the subform part of the form. The Subform Builder window is displayed.

**Tip** You might have to scroll up to the top of the window to see the subform part.

The following figure shows you that the Subform Builder window is identical to the Form Builder window. It contains:

- The form in the design pane.
- The actions linked to the subform in the action pane. When a form and a subform are displayed, the action bars of both the form and the subform are shown.
The field definition in the programming pane. In subforms as in forms, both @Functions and LotusScript can be used.

Subform Properties

To display the subform properties:

1. On the subform pane, click the right mouse button.
2. Select Subform Properties. The InfoBox is shown. It looks like this:

3. If required, check the Hide Subform from R3 Users check box.

   **Note** You must hide the subform from Release 3 users if the subform contains features that are not available in Notes Release 3, such as layout regions.

4. Close the InfoBox.
5. Close the subform.
Removing Subforms
You can remove subforms from the design of a form or from the design of a database.

Removing Subforms From the Form Design
If the subform is no longer needed in a particular form:

1. Open the design of the form.
2. Click on the subform area.
3. Choose Edit - Clear on the menu bar.

Removing Subforms From the Database Design
As for all design elements, you can remove subforms from the design of the database. This may be necessary if all the fields contained in the subform are no longer needed in any of the database forms.

However, if the database contains documents using the deleted subform, you have to make sure the users can still access the documents by having an empty subform using the same name as the deleted subform.

Computed Subforms
You can use computed subforms to show different elements, for example, to different users. Which subform is loaded is based on formula, so you can load different subforms for Web users than for Notes users. Here is how you do this:

1. First you have to create some subforms, at least two.
2. When your subforms are ready, open the Main Topic form.
3. Create a computed subform by choosing Design - Insert Subform... A Dialog box is displayed:

4. Choose Insert Subform based on formula and click OK.
5. A computed subform is created on the form and it looks like this:

![Computed Subform](image)

6. You have to specify a formula in the programming pane. The formula has to return a text string which is the name of the subform to be loaded.

**Looking at a Computed Subform**

1. You should still be in the Main Topic form.
2. Scroll down to the bottom of the form.
3. Click once on the Computed Subform.
4. You can see the following formula in the programming pane:

   ```
   @If(@ClientType="Notes";"";"Thread")
   ```

   `@ClientType` is a formula which tells if the user is using Notes or a Web client. The result of the formula above is a text string Thread, if the user is using a Web client, and the Thread subform is loaded. If the user is using a Notes client, the formula returns an empty string and no subform is loaded.

   Using a Computed subform is a good way to show some elements only to one kind of client or to users who have different roles. For example, one user can read information or provide content to the Web site while another user may have the authority to approve information for the Web.
Displaying a Different Form to Web Users and Notes Users

Maybe the easiest way to show different things to different users is to use different forms. This is most useful when the information you want to show to Web users and Notes users differs considerably. To use different forms do the following:

1. First of all you have to have two forms: one for Web users and one for Notes users.
2. Make sure that both forms have the same alias name. The form InfoBox looks like this:

3. The names of the forms can also be the same, but it’s much more useful to give them different meaningful names. In this way you can determine easily if the form is for use by Web users or by Notes users. For example, you could name them as Main Topic (Notes) and Main Topic (Web).

After creating two forms make them available only for Web users or Notes users by:

4. From the standard navigator, choose Design, then Forms. The list of forms is displayed. Click on the form and click the Properties SmartIcon. The InfoBox is displayed. Go to the Design tab. It looks like this:

Chapter 3: Designing Application Forms  55
5. On the Hide design element from section, select Web browser if the form is used only in Notes and select Notes R4.6 or later clients if the form is used only on the Web.

**Note** All Notes elements can be hidden from a Notes client or a Web client. When the user opens a document, whether in Notes or on the Web, the right form is always used to display the document.

---

**Tables Within Forms**

Tables are added to forms to format data as columns and rows. Within the table columns and rows, you can create design elements such as fields and buttons.

Tables are especially useful for applications used on the Web. Using tables you can make sure that fields are aligned correctly and that the images are in the right places.

To create a table, follow these steps:

1. In your form, click where you want to create the table.
2. Choose Create - Table.
3. Specify the amount of rows in the Rows field.
4. Specify the amount of columns in the Columns field.

The table is then created on the form.

**Table Properties**

1. There are three available Table border styles: Standard, Extruded and Embossed.

**Note** Choosing a Table border style doesn’t have any effect on the Web. Table borders are always the same on the Web.


2. You can set the thickness of the borders from 0 to 10. You can set all borders to 0 or 1 by clicking the button below. If you have selected more than one cell you can outline them by clicking the button below.

**Note** For the Web, there are only two options for borders, on or off. This is dependent on the borders of the upper left cell.

**Layout Tab**

In the Layout tab you can select Cell Width, Space between Rows and Columns and Left Margin. Only Cell Width has effect on the Web. To fit the table in the window regardless of size, select Fit Table Width to Window.

**Colors Tab**

In the Colors tab you can choose background color for the cell. To have the same background color in the whole table, click Apply to Entire Table.

**Using Tables for Layout on the Web**

If you try to lay out text strings and fields on a form using indents, Domino ignores them. This is how it looks on the form design:

![Form Design](image1)

and this is how it looks on the Web:

![Web Design](image2)

You can see that Domino has ignored the indents. However you can easily achieve the layout you want by using a table:

1. Choose Create - Table.
2. In the Rows field, type 2.
3. In the Columns field, type 2.
4. Click in the table area.
5. Click the Properties SmartIcon to display the InfoBox.
6. On the Layout tab, deselect Fit to Window.
7. Type the text string in the upper left cell.
8. Create a field in the lower left cell.
9. Type the text string in the upper left cell.
10. Create a field in the lower left cell.

The form design in the Notes Designer looks like this:

and on the Web:

11. If desired, table borders can be set to 0.

Note  If a table cell is empty, it is not displayed on the Web.

**Merge and Split cells**

Table cells can also be merged into one cell. Domino translates this into the proper rowspan and colspan attributes. If you want to simulate tables within tables, merging cells is the method to use. To create dynamic tables, you have to use HTML. See Chapter 6: Using HTML in Domino Design Elements.

In this example you are going to create a table of four cells and merge two of them into one.

1. Create a blank form by choosing Create - Design - Form on the standard navigator.
2. Create a table with two rows and two columns.
3. Highlight the two leftmost cells.
4. Choose Table - Merge Cells.
You can see that these two cells are now merged into one cell.

5. Type some text in all cells and preview the form on the Web.

6. To split cells, click on the merged cell and choose Table - Split Cells.

**Web Elements**

There are some elements in Notes Designer which can be used only on the Web. These elements are:

- Embedded Navigator
- Embedded View
- Embedded Folder Pane
- File Upload Control

Web elements are especially useful when they are used in a form which is a View Template. A View Template is a form with a special name, for example $$ViewTemplateDefault (the $$ViewTemplateDefault is used for all views, by default, unless another template is specified for a view). The concept of the View Template is covered more thoroughly in Chapter 4: Domino Views and Folders.
Embedded Navigators

An embedded Navigator is an element that makes it easy to show a Notes Navigator on the Web. You can have multiple Navigators in one form. Let’s have a look at one of the View Templates in the Discussion database.

1. From the standard navigator, choose Design - Forms. A list of forms is displayed:

2. You can see that there are four View Templates. The three first are for the particular view which is mentioned in the form name and the last one is the default for all the other views. Double-click the $$ViewTemplateDefault form. The form is then opened.
3. The bitmap shown on the left of the figure below is an Embedded Navigator.

4. Click the embedded navigator to highlight it.

5. You can see in the programming pane that the All Documents navigator is highlighted.

6. Try to change the embedded navigator by choosing another navigator in the programming pane to see what happens.

   Note  You can also embed a navigator based on a formula.
Creating a New Embedded Navigator

To create a new navigator you must already have the navigator ready.

Note The navigator must have been made Web Browser compatible. See Chapter 5: Domino Navigators.

1. Click somewhere on the form.
2. Choose Create - Web Element - Embedded Navigator. A dialog box is displayed:

![Insert Embedded Navigator dialog box](image)

3. You can choose a navigator from the list or you can specify a formula to choose a navigator. Click OK.
4. The navigator is displayed on the form.

As well as embedding the navigator in a View Template, you can also embed it in any form. This is very useful if you want to include, for example, button bars, headers, or footers on the form.

Embedded View

An embedded View is an element that makes it easy to show a Notes view on the Web. You can have only one view in one form.

1. You should still be in the $$ViewTemplateDefault form. If not, from the standard navigator, choose Design - Forms and open $$ViewTemplateDefault form.
2. You can see the gray rectangular object just to the right of the embedded navigator.

3. The embedded view is highlighted and you can see a list of views in the programming pane.

4. You can preview the form in Web browser by choosing Design - Preview in Web Browser.
Creating a New Embedded View
To embed a view into the form, do the following:

**Note** You can embed only one view per form.

1. Open a form.
2. Choose Create - Web Element - Embedded View. A dialog box is displayed:
   ![Insert Embedded View dialog box]
3. You can choose a view from the list or you can specify a formula to choose a view. Click OK.
4. The embedded view is displayed on the form.

If you take a closer look at the `$ViewTemplateDefault` form, you can see that design elements are inside a table. The table has three cells, the leftmost cell is merged from two cells.

As well as embedding a view in a View Template, you can also embed a view in any form.

**Embedded Folder Pane**

An embedded Folder Pane is an element that makes it easy to show a list of Notes views on the Web. You can have only one Embedded Folder Pane within one form.

1. You should still be in the `$ViewTemplateDefault` form. If not, from the standard navigator, choose Design - Forms.
2. First, delete the embedded navigator element.
3. To do this, click on the embedded navigator element and press Delete.
4. Then choose Create - Web Element - Embedded Folder Pane. After embedding the folder pane element, the form layout looks like this:

5. You can preview the form by choosing Design - Preview in your Web browser. It looks like this:
6. You can also change, for example, the font, font size, color of the font, and alignment of the folder pane.

7. Close the form and return to the standard navigator.

**File Upload Control**

A File Upload Control is an element that makes it easy to upload file attachments to the Web. You can have multiple file upload control elements in one form. To create a file upload control element, do the following:

1. From the standard navigator choose Design - Forms. A list of forms is displayed.
2. Open the Main Topic form.
3. Click below the Body field.
5. The File Upload Control element is displayed on the form:
6. You can preview the form by choosing Design - Preview in your Web browser:

7. Click the Browse button. A File Upload Dialog box is displayed:

8. Select the type of file and the file you want to upload. Click Open. The name and path of the file are added to the File Upload Control field.

9. Click Submit and go to All documents. You should see your document on the list. If not, try to Reload the page.
10. Go to the document and click Edit Document. Scroll down to the bottom of the document where you will see the file attachment.

![Attachment screenshot]

11. You can delete the attachment by selecting the check box next to file attachment. The attachment is deleted after clicking Submit.

**Note**  Web browsers which support File Upload Control are Netscape Navigator 3.x or later and Microsoft Internet Explorer 3.02 or later. To be able to upload files with Microsoft Internet Explorer you also have to install the Microsoft Internet Explorer File Upload Control add-on.

---

**Other Features of Forms**

**Horizontal Rules**

It is not necessary to write HTML to insert and define horizontal rules. To create a horizontal rule:

1. From the standard navigator choose Design - Forms. A list of forms is displayed.
2. Open the Main Topic form.
3. Click on the form, where you want to create a horizontal rule.
5. A horizontal rule is created on the form. You can change the settings of the horizontal rule from the Infobox:

6. By default, rule is set to fit the window. You can set the height and color, and either fit the width to the window or specify a width.

   **Note** Experiment with your browser to see if your browser supports colors and width attributes for rules.

**Computed Text**

Computed Text can be used to create text on the form or in the field of type rich text in a document, based on Notes @formulas. Creating Computed Text is similar to creating field of type text, which is computed for display.

Computed Text is not stored in a document and it is computed every time the document is opened, reloaded, or refreshed. Computed Text is especially useful when used in a document. Authors who don’t have designer privileges can create personalized Web pages. To create Computed Text on the form:

1. From the standard navigator choose Design - Forms. A list of forms is displayed.
2. Open the desired form.
3. Click where you want to create Computed Text on the form.
4. Type some static text.
5. Choose Create - Computed Text.
6. Computed Text is created on the form.

7. When Computed Text is highlighted, click on the programming pane and type the desired formula. In this case the formula is:

@Name ([CN]; @UserName)

Formula returns the common name part of the username.

8. Preview the form on the Web by choosing Design - Preview in Web Browser.

Note If the user has not logged in, the value of the Computed Text is Anonymous.

Computed Text in Documents
To create Computed Text in the rich text field, do the following:

1. In Notes, go to the All Documents view.
2. Click Main Topic to create a new Document.
3. Go to the Message Contents field. This is a rich text field.
4. Type some static text.
5. Choose Create - Computed Text.
6. In the programming pane, type desired formula.

Tip  If desired, you can type HTML into Computed Text.

Using $$\text{Return Field}$$

The $$\text{Return field}$$ is used for creating messages, after the user has submitted a document on the Web. Without the $$\text{Return field}$$, Domino responds with the default response “Form processed”. To customize this response message you can include HTML code as part of the formula for the $$\text{Return field}$$.

You can also use a $$\text{Return field}$$ to run a custom CGI (Common Gateway Interface) program immediately after the user submits the form and Domino will create the document. For example, you may run a CGI program that uses the Notes API to further process the input data. The Web client displays the output of the CGI program to the user.

To run a CGI program, include the URL to the CGI program file and enclose it in brackets. Note that you can pass arguments, for example, values from fields in the form, to the CGI program.

Returning to Another Page

You can display another Web page to the user once their form has been processed instead of leaving them with a blank, gray screen. This example displays the Lotus home page to the user, but you can direct the user, for example, to the main view of the database instead.

"[http://www.lotus.com]"

Adding a Link to Another Web Page

In the response message, you can include links to other Web pages. In this instance an HTML page will be created with a link to the Lotus home page. The user will see a blank screen with a link to the specified Web page once their form has been processed.

"<a href="http://www.lotus.com">Lotus</a>"

Adding links is useful, for example, if you want to provide the user with a choice of Web pages to display once their form has been processed.
Personalized Messages
You can create a personalized message for the user who submits a form. For example, the following $$Return formula returns the response “Thank you for your document,” and appends the user’s name. <h2> </h2> is an HTML tag and means that the text between those tags is a second level header. <hr> creates a horizontal rule.

"<h2>Thank you for your document, "+@Name([CN];@UserName) +"!
</h2><hr>"

$$Return Example from Discussion Template
The discussion template has a rather complex $$Return field, but going through the code gives you some good examples of how you could use the $$Return field.

First simple strings are assigned to variables to make it easier to write and to read formulas.

REM "resource strings..";
PrevDoc := "Previous Document";
AllDoc := "All Documents";
ByCat := "by Category";
ByAuth := "by Author";
Archiving := "Archiving";
MainTopic := "topic";
Response := "response";
IntProfile := "Interest Profile";
ArcProfile := "Archive Profile";
Message := "message";
ThankYou := "Thank you for your";
REM "Get the name of this database."
The following sentence gets the name of the current database. @DbName is a formula which returns the name of the current Domino server and the name and the path of the database.

DB := @ReplaceSubstring(@ReplaceSubstring(@Subset(@DbName; -1);" ";"+");"\";"/");
REM "Thank the user, personalize based on his name.";
This part is for personalization. Because the $\$Return$ field is a shared field in this database, the formula first determines which form was used.

\[
\text{FormName} := @If(\text{Form} = "MainTopic"; \text{MainTopic}; \text{Form} = "Response"; \text{Response}; \text{Form} = "ResponseToResponse"; \text{response}; \text{form} = "Interest Profile"; \text{IntProfile};
\]
\[
@Contains(\text{form};"Archive"); \text{ArcProfile}; \text{Message});
\]

\[
\text{Thanks} := "<h2>" + \text{ThankYou} + " " + \text{FormName} + ", " + @Name([CN];@UserName) + "! </h2><hr><br>
\]

REM "Anchors to discussion views."

This sentence creates a link to this document so that the user can return to this document. \text{@Text(@DocumentUniqueID)} returns the unique ID of this document.

\[
\text{existingdoclink} := "<b><a href="/+db+/($All)="/+@Text(@DocumentUniqueID )="/+?OpenDocument>" + \text{PrevDoc} + "</a></b> | ";
\]

\[
\text{TopicView} := "<b> <a href="/+DB + "/(All)?OpenView>" + \text{AllDoc} + "</a></b> ";
\]

\[
\text{CategoryView} := " | <b><a href="/+DB + "/by+Category?OpenView>" + \text{ByCat} + "</a></b> | ";
\]

\[
\text{AuthorView} := "<b><a href="/+DB + "/by+Author?OpenView>" + \text{ByAuth} + "</a></b> | ";
\]

\[
\text{ArchivingView}" = "<b><a href="/+DB+/Archiving?OpenView>" + \text{Archiving} + "</a></b> <hr>";
\]

REM "Assemble the HTML to be returned"

Finally variables are concatenated to one text string, which is then returned as a response to the user.

\[
\text{OkMsg} := \text{Thanks} + \text{existingdoclink} + \text{TopicView} + \text{CategoryView} + \text{AuthorView} + \text{DateView} + \text{ArchivingView};
\]

\[
\text{OkMsg}
\]
**Using JavaScript in $$Return Field**

Besides HTML, you can write JavaScript in a $$Return field. Here is a small useful example:

Normally in a Discussion database when the user clicks the Submit button they get the response message seen above. But if you want the user to return to the view where they came from, you type the following line into the $$Return field.

```javascript
Back2 := "<script>history.go(-2);</script>";
Back3 := "<script>history.go(-3);</script>";
@if(@isNewDoc;Back2;Back3)
```

The first line assigns a temporary variable which causes the browser to jump back a couple of windows. The second line assigns a temporary variable which causes the browser to jump back three windows in the history list. The third line checks whether it is a new document which implies it was composed off the action button in the view. If that is the case, it takes the "Back2" temporary variable which tells the browser to move back two windows. If it is not a new document, then it causes the browser to move back three windows.

**Buttons, Action Bar Buttons and Hotspots**

Web browsers don’t have a prebuilt Graphical User Interface (GUI) for Domino applications, so you must provide Web users with a way to perform common actions such as creating new documents, saving a document, and changing to another view.

Multiple buttons can be created on Notes forms to be displayed on the Web. Button formulas are run when the user clicks the button. With buttons or view actions that display in the action bar, Web users can perform functions equivalent to those that Notes users are able to perform by using menus and the Action Bar.

To create multiple buttons and to Web-enable all buttons in an database:

1. The browser must support JavaScript.
2. Select the database property Web access: Use JavaScript when generating pages. Without this property set, Domino recognizes only the first button in a document and treats it by default as a Submit button that closes and saves the document. If there are no buttons in the form, Domino places a Submit button at the bottom of the form.
Guidelines for Using Button Formulas

Notice the following guidelines and restrictions:

- Action bar buttons and hotspots support only part of the actions, See Appendix A: “Notes Features to Avoid When Developing Web Applications.”

- In Release 4.5 you were not able to use images as buttons, but in Release 4.6, with the Use JavaScript database property set on, you can now use images as buttons.

- You should create only one Submit button per form. You can customize this button, even if you haven’t selected the database property: Web access: Use JavaScript when generating pages.

- Formulas on buttons are run when the user clicks the button, while formulas on hotspots and action buttons are run when the hotspot or action button is displayed.

**Tip** If your browser doesn’t support JavaScript and you don’t want any buttons on the form, write the following HTML code on the bottom of the form: `<form>`. Remember to mark text as pass-thru HTML and hide it from Notes users.

Images Within Forms

There are several ways to add images to your Web pages:

- Copying images through clipboard
- Importing images
- Using pass-thru HTML
Copying Images

1. Copy desired image to the clipboard.
2. Open the form in design mode.
3. Click on the form where you want to place the image.

Importing Images

1. Open the form in design mode.
2. Click on the form where you want to place the image.
3. Choose File - Import.
4. Choose type of file, click the file to import, and click OK.

**Tip** Often the quality of the image is better when the image is imported rather than pasted.

Using Pass-Thru HTML to Attach Images in Web Pages

In some cases you might want to use pass-thru HTML to add images in Web pages. For example, if you want to keep all images in the same directory, you add a pass-thru HTML reference in the form. For example:

```html
<IMG SRC=http://www.company.com/images/image.gif>
```

**Note** Importing animated images is not supported in Domino. If you want to add an animated image to a Web page, use the pass-thru HTML reference shown above.

You might want to include images inside the database to allow designers to update, replicate, and copy Web images without needing access to the server’s file system.

1. Create a form with two fields: a text field for the document title and a rich text field.
2. Store images as attachments in the document in the rich text field or create one document per image and attach in the rich text field.
3. Create a view that identifies the document(s).

To reference an image stored in the database, use the Domino URL syntax for opening documents from a view by key name. For example:

```html
<IMG SRC= http://www.company.com/view/doctitle/file$/image.gif>
```

**Note** If you attach more than one image in the document, you must use unique file names.

**Tip** Use tables to lay out images in Web pages.
Domino URL syntax is explained in Chapter 6: Using HTML in Domino Design Elements.

**Alternate Text**

Adding alternate text to graphical hotspots allows Web users with text-only Web browsers to see text on the form where the graphic should be. Users with a Web browser that supports graphics will see the alternate text while the Web browser is loading the graphic. To add alternate text:

1. Select the graphic and choose Picture - Properties
2. In Alternate text for Web and deferred loading, type the text describing the graphic.

---

**Special Form Types**

The most common Domino forms are input forms, which we have used this far. Besides input forms, there are several special form types.

**Using a Form as a Page Template**

By using these forms Domino associates the form with the view or navigator:

<table>
<thead>
<tr>
<th>Form name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$ViewTemplate for viewname</td>
<td>Associates the form with the view specified in form name.</td>
</tr>
<tr>
<td>$$ViewTemplateDefault</td>
<td>Associates all Web views which are not associated with special form.</td>
</tr>
<tr>
<td>$$NavigatorTemplate for navigatorname</td>
<td>Associates the form with the navigator specified in the form name.</td>
</tr>
<tr>
<td>$$NavigatorTemplateDefault</td>
<td>Associates all Web navigators which are not associated with special form.</td>
</tr>
<tr>
<td>$$Search</td>
<td>For customizing search input forms.</td>
</tr>
<tr>
<td>$$SearchTemplate for viewname</td>
<td>Associates the form with the view specified in the form name. Used for customizing search results.</td>
</tr>
<tr>
<td>$$SearchTemplateDefault</td>
<td>Associates all Web views that are not associated with special form. Used for customizing search results.</td>
</tr>
</tbody>
</table>

The use of View and Search templates is explained thoroughly in Chapter 4: Domino Views and Folders. The use of Navigator templates is explained thoroughly in Chapter 5: Domino Navigators.

Chapter 3: Designing Application Forms  77
Customizing Error Messages

To customize error messages that are returned to the user after an error, you can customize the following forms:

<table>
<thead>
<tr>
<th>Form name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$ReturnDocumentDeleted</td>
<td>The user successfully deletes documents.</td>
</tr>
<tr>
<td>$$ReturnAuthenticationFailure</td>
<td>The user’s name and password can’t be verified.</td>
</tr>
<tr>
<td>$$ReturnAuthorizationFailure</td>
<td>The user doesn’t have a high enough access level to access the database.</td>
</tr>
<tr>
<td>$$ReturnGeneralError</td>
<td>Any other errors occur.</td>
</tr>
</tbody>
</table>

To create an association between the error message and the form:

1. Create a form with one of the names above.
2. Add text, graphics, HTML or links to the form, that you want to display on the message, or create an editable text field to hold the error message.

When an error condition occurs, Domino displays an error message using the customized form.

Tip  You can copy these forms from the Discussion template, paste them into your database, and customize them.

Using CGI Variables

CGI, Common Gateway Interface, is a standard for interfacing external applications with HTTP servers. When a Web user saves a document or opens an existing document, the Domino Web server uses CGI variables to collect information about the user, including the user’s name, the browser, and the user’s Internet Protocol (IP) address.

To capture this information in a Web application you have two options:

1. Create a field with the same name as a CGI variable.
2. Use LotusScript agents.

Using a Field to Capture CGI Variables

When a field has the same name as a CGI variable, Domino copies the field value from the CGI environment and places it in the field. There are two things to note:

1. You don’t have to specify a formula if the field is editable.
2. Make the field hidden when previewed for editing and opened for editing.
For example, you can create a field named HTTP_USER_AGENT. This field captures information about which browser the user is using. You can use this field to see if the user’s browser supports features in your application.

The general format of the HTTP_USER_AGENT variable is software/version library/version. The eleven leftmost characters tells you browser version. You can use this information in formulas to display different items to the user depending on the version of their browser. For example:

@If(@Left(HTTP_USER_AGENT;11)="Mozilla 3.0";"Netscape Navigator"; @Left(HTTP_USER_AGENT;11)="Mozilla 2.0";"Microsoft Internet Explorer";"")

Some of the values returned by HTTP_USER_AGENT

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla 2.x</td>
<td>Microsoft Internet Explorer 3.x</td>
</tr>
<tr>
<td>Mozilla 3.x</td>
<td>Netscape Navigator 3.x</td>
</tr>
<tr>
<td>Mozilla 4.x</td>
<td>Netscape Navigator 4.x</td>
</tr>
</tbody>
</table>

Using a LotusScript Agent to Capture CGI Variables

You can use the DocumentContext property of the NotesSession class to capture CGI variables. The property returns a Notes document that contains all CGI variables that are applicable to the session. You can use these values to collect or process information for the current session.

The following example demonstrates how to access CGI variables:

Dim session As New NotesSession
Dim doc As NotesDocument
Dim CGIValue As String
Set doc = session.DocumentContext
CGIValue = doc.HTTP_USER_AGENT

*CGIValue* string value now has information about the user’s browser.

For more information about CGI environment variables, see Appendix C: “Table of CGI Variables Supported by Domino.”
The form is the skeleton you provide to users to create documents. Users can enter data by typing it into fields. The form contains all the design elements: fields to store the user’s information, and static text, buttons, sections, and subforms. When the document is opened through the Web, Domino automatically converts it to an HTML page.

When you develop forms think of the different behaviors of a form on a Notes platform and on a Web platform. Web users don’t have access to form menus for database activities such as saving the document. Therefore, when you design your Web forms you must supply actions to enable your users to interact with the application correctly.
Chapter 4  
Domino Views and Folders

In this chapter we take a look at views, one of the key design elements in Domino. The following topics are covered:

- What are views and why are they useful?
- Creating views for the Web.
- How to use View templates to display views on the Web.
- Views and URLs.

What Are Views and Why Are They Useful?

A view is the entry point for opening and reading documents in a Notes database. When a user opens a view, a list of the documents in the database is displayed. The view acts as a table of contents for the database. As such, views provide a convenient, easy, and meaningful way for users to navigate a database. If you are accustomed to Web design, think of views and folders as index files, only in Notes the indexes are dynamic, have automatic updating functions, programming features, and graphics.

Every database must have at least one view, although most databases have several views that organize and present documents in different ways. Views provide alternate tailored entries to a database.

In many cases a database will also have views that are hidden from the users, but are used for programming purposes to look up field values. If you are developing for both Notes and Web clients, you might need to create alternate views for the respective clients to enhance the display on each platform.

There are three types of views in Notes:

1. Standard views
2. Calendar views
3. Folders
The standard view is the most commonly used. It displays the documents in a database organized in rows and columns.

Calendar views group the documents within date blocks and are generally used for organizing schedules, meetings, and appointments, etc. Web users can see all calendar entries. Domino converts calendar views to HTML tables, so restrictions for tables also apply to calendar views. For example, some schedule displays, such as conflict bars, the clock, creating new appointments, and scrolling through entries within a single day, are not supported on the Web.

Folders have the same design elements as a standard view, and are designed in much the same way as views. The difference between folders and views is that a folder remains empty until a user drags and drops documents into it or a program adds documents to it. On the Web you cannot drag and drop documents into folders.

**Selecting Which Documents Appear in a View**

A view can display all the documents in a database, or it can display only a subset of documents, depending on the selection formula defined for the view. Most databases have one view that shows all documents in the database and then a number of views that show a subset of documents tailored to different audiences and tasks.

When you publish a database on the Web, the views you display on your Web site will typically have a selection formula that selects only the documents whose Status field is equal to Approved and other views used by content providers that show documents still in progress. In this way you can use the selection formula to tailor your views to different tasks and user groups.
Looking at a View Using a Notes Client

Let us have a look at the various elements of a standard Notes view. When you open a view in Notes you might see something like this:

This is a Notes view in its most basic form. The action bar is displayed at the top of the window. It contains buttons that perform either simple actions or @Formulas. The Navigator button toggles the display between a Navigator view and the default standard navigator, also known as the outline hierarchy. The outline on the left-hand side shows available views. In our example we are looking at the By Author view.

View Pane

The view pane consists of: the selection column and a listing of the documents. The selection column uses icons to convey information about the documents in the view. Our example shows that two documents have not yet been read, one document is marked for deletion and one document has been selected for some later action.

A standard view organizes documents in rows and columns. Each row in a view represents data from a single document in the database. The row is displayed as one or more columns. Each of the columns represents a field or combination of fields in the document, or the computed result of a column formula. The above view has eight rows each representing a document in the database. It has three columns entitled Author, Date, and Topic. Each of the columns display a field value extracted from the document. In the Author column the value from the document’s Author Name field is displayed. The Date column displays the value of the Creation Date field in the document. The Topic column’s value is derived from the Subject field.
Sorting
Every view needs a sorting method to list the documents in a way that makes sense to the user. In our view above, for example, the second column containing the creation date is sorted in descending order to display newly created documents at the top of the list. Views can be either auto-sorted or user-sorted. Users see a small triangle next to the column title whose value can be sorted. Users click the column title to sort the view in a different order. In our example the view is sorted in ascending order after clicking the Date column header.

Categories
To better organize a view and help users find related information, you can choose to have one of the columns categorize the documents in the view. The categorized column will then appear in the view as a heading to the documents belonging to the same category.

The graphic below shows the by Category view from the same database used in the example above. Documents with the same value in the documents’ Category field appears underneath a heading displaying the category.

You can expand and collapse a categorized view by clicking the twistie (the small triangle icon) next to the category. When collapsed, the view shows only the category title. In our example, the URLs category is collapsed and the others are expanded. When you click the triangle next to the URLs category it will expand and show you the documents in the category. Categorized views enable users to easily find related documents in a view.
Looking at a View Using a Web Browser

Domino dynamically creates Web pages out of the Notes views in a database, including URL links to the documents in the view. Using a Web browser, a user can navigate, expand, and collapse the view in much the same way as in Notes.

When using a browser the view is split into pages of 30 lines per page. This is to avoid a view containing hundreds of documents being presented as one page containing all the documents. Limiting the lines per page in this way improves performance and makes navigation of the database more manageable.

**Note**  The default of 30 lines per page can be changed in the HTTP section of your server document.

When Domino generates the HTML page for a view, it maintains the column and row format of standard Notes views. There are, however, a number of differences you need to be aware of and take into consideration when you develop applications for the Web.

View Outline

The above view shows the previous example of a categorized view as seen through a Web client. The first thing you will notice is that there is no outline of available views on the left. Domino does not include a view outline by default on your Web page. If you want to include a view outline you must design a special form, a Viewtemplate, to display your view and imbed a folder pane on the form. Only the menu action for creating a new document is displayed on the Web. Domino ignores any menu actions not supported on the Web.
Viewtemplates are forms with embedded views, used to enhance the appearance of views on the Web. In most cases you will probably embed your view in a Viewtemplate form with a Navigator or folder pane included. This will give you a nicer layout and help your users navigate your Web site.

The graphic below shows a view embedded in a Viewtemplate with an embedded Navigator on the left and menu actions at the top. Later in this chapter we will cover how to imbed folder panes and add menu actions to a Viewtemplate.

Selection Column
The next thing you will notice is that there are no selection column icons. The concept of a selected document in a Notes view does not apply to a Web view. You cannot use the Response and Response to Response buttons on a Web view for the same reason, since they act on selected documents in a view. To enable users to create response documents on the Web, you must include Response buttons on the document form.

Each time a Web user opens, expands, or collapses a view, Domino converts it to a new HTML page. Each “snapshot” of the view is a newly generated HTML page and basically a new session with the Domino server.

Web Navigation Bar
You will notice that Domino has automatically generated a Web navigation bar at the top and bottom of the screen. This navigation bar contains buttons that users click to expand, collapse, scroll, and search the view. Web users don’t have action menus for database activities in their browsers, like Notes.
users do. If you create your own Viewtemplate to display the view you must supply form and view actions to enable your users to interact with the application.

**Document Link**
On the Web, users open documents by clicking a document-link column rather than clicking anywhere in the row as you do in Notes. The application designer can specify which column(s) should include a URL link to the document. By default, it is the first non-categorized column.

**Expanding Categories**
You will also notice that all categories are expanded. On the Web you can expand only one category at a time or expand all categories.

**Summary**
To summarize, views are user-defined ways of looking at information in a database. They act as an entry point to the data in your database. Views are the main navigational element of a Notes database and should be designed so as to provide users with a convenient and meaningful way to navigate a database.

Different views may be created based on the viewing requirements of the application. Think of your audience when you design your views, what will they be looking for in the database, what are the tasks they will perform using the view? Tailor your views accordingly.

When you develop views think of the different behaviors of a view on a Notes platform and on a Web platform. For example, Web users don’t have access to view menus for database activities and there is no Web equivalent of a selected document in a Notes view. Therefore, when you design Web views you must supply form and view actions to enable your users to interact with the application.

We will now take a more detailed look at designing views for users accessing a Notes application using a Web browser.
Creating Views for the Web

You will be using the Notes Designer from a workstation when you design your Web views. The Web application resides in the data directory (or a subdirectory) on your Domino Web server. To see your work, either use the Preview in Web browser option in Notes or start your Web browser and type the URL command to open the database on the Domino Web server.

**Note** The HTTP task must be running on the Domino server.

We will start by having a look at the Document Library template in Notes 4.6. The template includes a number of design features aimed specifically at Web client access. We will be making some changes to the By Category view.

**Note** You need at least Designer access in the database ACL in order to change a view.

View Design

1. Create a Document Library database based on the Doc Library - Notes & Web (R4.6) template. Deselect Inherit future design changes. You might lose your design changes if this option is not deselected.
2. Open the database and choose Design, then Views. The available views are listed in the view pane.

![View Design](image)

The ($All) view is a reserved name for Notes to create the default view, All Documents, with its associated icon. Except for the reserved ($All), enclosing a view between parentheses means that the view is hidden from users and is used solely for programming purposes.
3. Double click on the By Category view to open its design and click on the Properties SmartIcon to display the View Properties.

In the top half you can see the design pane, with the view columns listed under the SmartIcons. Only the Topic column has a title. In the lower half you can see the programming pane. Notice that the View Selection pane is empty, meaning that all documents in the database are included in the view.

**Selecting Which Documents Appear in a View**

The default selection for a new view is the formula “SELECT @All” which means “include all database documents in this view.” To narrow down the kinds of documents the view displays, you can program which particular documents are selected. For example, in a task-tracking database, the Work in Progress view could select only those documents whose Status field does not contain “Complete.” In a brainstorming database, the Design Ideas view could select all documents categorized as “Design Suggestion.”

With a view in design mode, you program the document selection in the design pane. Choose “View Selection” in the Define list box, click the button for the type of programming you want to add, then build the program in the window below.

**Easy**

Easy allows you to create a document selection without knowing a programming language. In the design pane, click Easy, then click Add Condition for each selection you want to include. To delete a condition, click it in the formula window, and choose Edit - Clear.
**Formula**
Formula allows you to create a program for document selection that uses the @function formula language. In the design pane, click Formula and write the formula in the window below. In an Action Items database for example, to select all documents that were created using the Action Item form and whose Status field contains the value “Open,” you enter the following formula:

```plaintext
SELECT Form = "Action Item" & Status = "Open"
```

If a database contains a form called Action Item, you can select all documents that were created with the Action Item form, as well as responses to these documents, using the formula:

```plaintext
SELECT Form = "Action Item" | @IsResponseDoc
```

The Databases by Title view in the Database Library template selects all documents that were not created with the Librarian form using this formula:

```plaintext
SELECT Form != "Librarian"
```

**View Properties**

1. Click on the Properties SmartIcon to display the View Properties.
   In the properties box you will notice that this is a standard Notes view. The name of the view as well as its alias is displayed. Always include aliases for your views and refer to the alias in the URL for your views. This is particularly important when you are developing Web applications.
   The View name is displayed in the browser’s title bar. Since links point to an alias, using an alias rather than the view name allows you to change the name of a view later without breaking your links. This gives you great flexibility when designing your Web site.

   **Tip** It is helpful if the alias name does not contain a space.

2. Click on the leftmost data column.
   Notice how the Properties box changes to Properties for the Column. Click the Sorting tab in the Properties box. You will notice that the view is sorted and categorized by this column. If you go back to the basic properties for the column, you can see the column width has been set to 2.
A categorized column acts as a heading to the documents in the category, this is why you can set a small column width. Categorized views can be expanded or collapsed at the heading level. You can choose to “Show twistie when row is expandable,” to put a small triangle next to the expandable column. This setting has no effect on the Web, all expandable views show twisties on the Web, you cannot remove them.

In the Programming pane, the “field” radio button is selected. A list of the available form fields are displayed. The Categories field has been selected and will be displayed in the column.

If you click on the Topics column, you will see a formula used to display a value in the column. Formulas allow you to create a program for a column using the @function formula language. This is useful when you need to process values in a field before displaying it or when calculating a value. Typical uses for formulas include writing a formula that displays an icon instead of a value, adding text to field values and adding HTML coding when developing for the Web.

Tip To improve Web performance, avoid time-sensitive column formulas with @Functions such as @Now, @Created, @Modified, and so on. Since the Domino Web server generates Web views as HTML pages on the fly, time-sensitive formulas recalculate every time a Web user works in the view, for example, opening, scrolling, or expanding the view, etc. Instead, create a field in the form for the formula and refer to the field in your view. Refer to Chapter 3: Designing Application Forms for more information on how to create field formulas.
Specifying Properties

We have already touched upon a few of the View and Column properties. We will now take a closer look at the properties and their relevance to Web applications. Most properties work on the Web. The ones that don’t work are generally related to features in the Notes client interface and don’t have a Web equivalent.

View Properties Supported on the Web

- Background color.
- Show column headings.
- Lines per heading.
- Lines per row.
- Collapse all when database is first opened.

Web Only View Properties

- Treat view contents as HTML. The entire view is treated as pass-thru HTML. There is an example of using this feature in the “Using HTML in Domino Design Elements” chapter.

View Properties Not Supported on the Web

- Show in View menu. As mentioned earlier Web applications do not have a View menu. This option is used by Notes applications to hide a view from the view list. To hide a view to Web users only, use the Design Properties box and select the Hide design element from Web browsers option.
- View indexing options are not applicable. Views can be re-indexed on the server.
- On Refresh options.
- Style options for Unread rows and Alternate row colors are not supported.
- Show selection margin.
- Beveled column headings.
- On Open, Go To options are not supported. The feature can be simulated on the Web using an &StartKey argument in the URL. For example, Domino will accept an argument of &StartKey=P, which means opening the view at the first sorted entry that begins with “P.” You can use a URL like:

```
/names.nsf/people?Openview&StartKey=P
```

which will open the People view in the Address Book positioned at the first person whose name begins with “P.”
Column Properties Supported on the Web
- Click on column header to sort (this feature is newly supported in Release 4.6).
- Display values as icons.
- Hide column.
- Show response only.
- Justification.

Web Only Column Properties
- Showing values in this column as links enables you to select which column(s) should act as a link to the documents in the view.

Column Properties Not Supported on the Web
- Click on column header to change to view.
- Resizable columns are not supported.
- Show Twisties When Row Is Expandable is not available. On the Web triangles are always shown for expandable columns.

Changing a View
You can create a view from scratch by selecting Create - Design - View from the Notes menus. However, in most cases, to save time, you will modify an existing view that you have copied and pasted either from the current database or from another database. Let’s go back and copy an existing view that we can then modify. We will start by copying and pasting the By Category view. The new view will be called My Category View. We will add some icons to the rows, add a column header icon, change the background color, add some HTML coding to enhance the display on the Web, and create a column header link to another view.

Copying & Pasting an Existing View
1. Start by opening the Design - Views pane.
2. Click once on the “By Category” view to select it. Select Edit - Copy and Edit - Paste.
3. Double click on the new “Copy of By Category” view to open the design.
4. Open the properties box and change the name to “My Category View” and the alias to “MyCategory.”
Changing Colors
1. In the view Properties box, click the Style tab and change the background color to a light blue.
2. Deselect Alternate rows.
3. Click on the leftmost column.
4. In the properties box, click the font tab and change the font.

Inserting an Icon Column
1. Click once on the fourth column from left and select Create - Insert New Column. A new column is inserted to the left.
2. Leave the title field blank. Change the column width to 1 and select Display values as icons.
3. In the Programming Pane click on Formula and then delete the @DocNumber in the Formula area.
4. Insert the following formula:
\[
@If(@Attachments; 5; 0 )
\]
This formula determines whether a document has an attachment and, if so, displays the attachment icon (number 5). Use 0 as the ‘false’ case when you want to leave the column blank. The formula above returns 0 when the document has no attachments, so nothing is displayed.

An icon column has two requirements:

- The column property “Display values as icons” must be selected.
- The column must use a formula that results in a number that corresponds to the icon you want to display.

**Note** All the number references for the available 170+ icons are listed in the Table of column icons document in the Help database.

To create your own custom icons for Web views do the following:

- Create an icon of the appropriate size using a graphics tool.
- Save the icon as a .gif file with a name of “vwicn” plus a three digit number: vwicn800.gif for example.

**Note** The first 172+ numbers are used by the default icons in Domino. The attachment icon we used above for instance has the filename vwicn005.gif. Just to be safe, use a high number like 800 to start with.

- Place the file in the Domino “icons” subdirectory on the server.
- In the column formula you can now reference the new icon with the number 800. The above attachment formula for example, would look like this using the new icon: @If(@Attachments; 800; 0 ).

**Note** These custom icons will be visible only on the Web. Also, if you replicate the database to another server, you will need to add the icon to that server.
Creating Column Header Sort

1. Double click on the leftmost column. This opens the property box for the column.

2. Select the sorting tab and select Click on column header to sort.

3. Select Descending as sort order. The default sort order for the view is Ascending. When the user clicks the new sort tab it will sort on the first column in Descending order.

4. Save and close your view.

Create some documents in the database. Add attachments to some of the documents. Your view should now look something like this:
Use the Preview in browser feature in Notes or type the URL for the view in your browser to look at the view from the Web.

(\texttt{http://host/databasename.nsf/MyCategory?OpenView})

You will see that the various changes we have made, like color and font specifications, are all rendered on the Web. The formula-based column icons are automatically displayed on the Web, as well as the click on column header to sort (on the Web you have to actually click the small triangle). Clicking the category triangles expands and collapses the category. Domino has automatically generated document links for all the documents in the view. You will also notice that the top part of the screen on the Web is occupied by the navigation buttons and the view title that is automatically generated by Domino.
Selecting Columns to Contain Document Links

You can designate any column or columns in a view to display as links to the documents. You may want to let all columns in a view act as links.

Setting the column links is only a click away in Domino. For each column you want to display with links, choose Show values in this column as links from the advanced tab of the column’s Properties box:

We will append a new column to the far right and display the name of the document author in the column and then make the new column act as a document link on the Web. We will remove the display of the author name and creation date from the Topic column.

1. Open the view in design mode again and click once on the Topic column to select it.
2. Select Create - Append New Column.
3. Click once on the new column to select it and click the Formula radio button in the programming pane. Write the following formula in the formula window (you can copy and paste it from the formula for the Topic column):
   \[ @Name([CN];From) + @If(@IsAvailable(From);" ";"") \]
4. Go back to the Topic column and delete the last part of the formula. Your formula should look like this when you are done:
   \[ @If(readers!="";"PRIVATE:";ExpireDate !="";"EXPIRED:";" ") + @If(Subject = "";"Untitled"; Subject) \]
5. Double click on the new column header. This will bring up the Properties box for the column.
6. Click the Advanced tab (the tab with the small hat icon). Select Show values in this column as links.

Close the Properties box and press Esc. Select Yes to save your changes. Look at the view from your browser:

Notice how both the author column and the topic column act as a link to the documents. You can select as many columns as you like to act as a document link. Also notice how the column width defined in Notes is not maintained on the Web. By default, Domino displays the width of a column in a view as the length of the longest entry in the column, regardless of what the column width is set to in the Column Properties box.
Tip To force a column to be limited to a certain width when viewed from the Web, use a column formula to retrieve the field value, for example:
@Left(FieldName;50)
This formula will set the widest width of the column to 50 characters.

To avoid long columns being pushed to the right, use the Style tab in the View Properties box and specify a number greater than 1 in the Lines per heading setting. This causes the lines to wrap on the Web. If you specify 1, the lines will not wrap. The same guidelines apply to column headings.

Tip As well as selecting multiple columns to contain document links you can choose not to have any document links in a view, thus creating a ‘read only’ view.

Adding HTML to a View

You can also add HTML coding to the view in order to enhance the display on the Web. HTML embedded in views is a browser-only feature. If you embed HTML in a view, the view is accessible by a Notes client, but the client interface will be unattractive (due to the exposed HTML tags). There is no option to hide a column from Notes users, as in the case of forms and views. A column is either hidden or visible to both Web and Notes users. Although you can see the HTML tags from the Notes client, the features that the HTML coding provides, such as linking, are not available to Notes clients.

Unlike in forms and documents there is no pass-thru HTML option in the design interface. To write HTML in views you have to include the HTML code in square brackets “[ ]”. Domino treats everything between square brackets as pass-thru HTML.

We will make the following changes to the view we have been working on:

- Insert a blank .gif file between the Topic column and the author column in order to get a little more space between the two columns on the Web.
- Add a horizontal ruler spanning the entire view, for each document category.
- Add a couple of icons and a URL link to the home page in the column headers.
- Include an HTML statement in the formula for the Topic column to display a blinking text string if it is a newly created document.
- Finally we will include a URL in a couple of documents and jump directly to the URL from the view.
Adding a Space Between Columns.
1. Create a new column to the left of the author column and set the column width to one. Deselect the Show values in this column as links if not already done.
2. Select Formula in the programming pane and insert the following HTML code, including the quotation marks, as this is a text string:

   \[ \text{"[<img src=/icons/ecblank.gif>]"} \]

**Note** The “ecblank.gif” is one of the standard icons that come with Domino. It is located in the icons subdirectory on the Domino server.

Adding HTML and Icons to Column Headers
You can use the column title field in the column properties to add graphics and pass-thru HTML to your column headings. The only limitation is that you can fit only 64 characters into the column title.
1. Open the Properties box for the leftmost column (our categorized column) and in the title field add the following line (no quotation marks):

   \[ \text{Sort \[<img src=/icons/vwicn108.gif>]} \]

   This will add a small green arrow pointing out where to click to sort the column on the Web. Non Notes users might not be familiar with the little triangle.
2. Open the Properties box for the author column. In the title field enter:

   \[ \text{[<A HREF="/">
   <img src=/icons/vwicn069.gif border=0>Home</A>]} \]

   This will add an icon with a doclink to the home page from the column header.
3. Click the Title tab and select Right Justification.

Adding HTML to Column Formulas
1. Click the leftmost column. In the programming pane click the Formula button and enter the following formula for the column:

   \[ \text{Categories + "[<hr>]"} \]

   This will display the value of Categories field and then add a horizontal rule. Notice that the + sign is used to append values.
2. Click on the Topic column and append the following formula to the existing formula:

   \[ \text{@If(@Now>@Adjust(@Created;0;0;7;0;0;0);"";"[<blink><font color="Red">New</b> New </b></blink>]"} \]
The entire formula for the column should look like this:

```plaintext
@If(readers!="";"PRIVATE:";ExpireDate !="";"EXPIRED:";""")+
@If(Subject =""; "Untitled"; Subject)+
@If(@Now>@Adjust(@Created;0;0;4;0;0;0);"";"[<blink><font color="Red"><b> New </b></blink>]<b> New </b></blink>])
```

The formula checks if the document was created within the last week, if so it will add a blinking “New” text string next to the document title.

**Note**   The HTML blink tag is not supported on all browsers.

3. Press Esc and save your changes.

**Creating URL Links at View Level**

You can add HTML to the document fields displayed in the view columns in the same way. By adding a URL address to a document, you can display URL links in a view, enabling users to jump directly from the view level to a URL address.

**Note**   You could also calculate the HTML in the view, which would be useful if you wanted to use the same document but with different views for Notes and Web clients.

We will create a couple of new documents linking to the Lotus home page and the Notes.Net Web site.

1. Create a new document and include the following URL in the Document title. The pass-thru HTML text style option is available only in rich text fields; you will have to include the URL in square brackets:

   ```plaintext
   ```

2. Give the category the name Links.
3. Save your document and create a new one. Enter the following URL:
   
   \[<A\text{HREF=\http://notes.net}>Notes.Net</A>]  

4. Save your document and take a look at your view from the Web.

![Image of My Category View](image)

Your view should now look something like the figure above. We have added a couple of icons to the column headings and a URL link to the home page. A horizontal ruler now helps divide the different document categories. The two new documents are clearly marked by a blinking “New” text string. Both documents take you directly to the respective Web sites.

**Note**  In the column properties we defined two columns to show values as document links, the Topic and the Author column. Therefore, the two new ‘Link’ documents each contain two links. In the Topic column we override the automatically generated document link with pass-thru HTML. The Author column still contains the document’s URL generated by Domino and still links to the document.

To further enhance the display of a view on the Web you can embed the view in a form. When you embed a view in a form you can use all the available form elements such as styled text, graphics, tables, hotspots, etc. to design the form.

When you display a view through a form, Domino disregards the view title and uses the background colors defined in the form. If you want to include the view title you will need to do it in a field formula on the form.

**Providing Menu Replacements for Web Users**

Domino combines form and subform actions with embedded views. However, Domino does not display the default view navigation bar with functions such as, Next, Prev, Expand, Collapse, and Search, for embedded
views. You may also want to add actions for creating documents and switching to views. You do this by creating Hotspot Actions or links using one of the following:

<table>
<thead>
<tr>
<th>Action</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Page</td>
<td>@@Command(“Domino”;ViewNextPage)</td>
</tr>
<tr>
<td></td>
<td>Creates a link to the next page in a view. Can only be used on the Web with the above syntax.</td>
</tr>
<tr>
<td>Prev Page</td>
<td>@@Command(“Domino”;ViewPreviousPage)</td>
</tr>
<tr>
<td></td>
<td>Creates a link to the previous page in a view. Can only be used on the Web with the above syntax.</td>
</tr>
<tr>
<td>Page Down</td>
<td>@@Command(ViewPageDown)</td>
</tr>
<tr>
<td></td>
<td>Scrolls an embedded view down one page.</td>
</tr>
<tr>
<td>Page Up</td>
<td>@@Command(ViewPageUp)</td>
</tr>
<tr>
<td></td>
<td>Scrolls an embedded view up one page.</td>
</tr>
<tr>
<td>Expand</td>
<td>@@Command(ViewExpandAll)</td>
</tr>
<tr>
<td></td>
<td>Expands all categories in a view.</td>
</tr>
<tr>
<td>Collapse</td>
<td>@@Command(ViewCollapseAll)</td>
</tr>
<tr>
<td></td>
<td>Collapses all categories in a view.</td>
</tr>
<tr>
<td>Search</td>
<td>@@Command(ViewShowSearchBar)</td>
</tr>
<tr>
<td></td>
<td>Opens either the default or a customized search form.</td>
</tr>
<tr>
<td>Change view</td>
<td>@@Command(ViewChange);“viewname”</td>
</tr>
<tr>
<td></td>
<td>Changes to another view in the database. Equivalent to the Column sort property “Click column header to Change to View.”</td>
</tr>
<tr>
<td>Open View</td>
<td>@@Command([OpenView];“viewname”)</td>
</tr>
<tr>
<td></td>
<td>@@Command([OpenView];“database”;“viewname”)</td>
</tr>
<tr>
<td></td>
<td>Opens a view in the current database or another database.</td>
</tr>
<tr>
<td>Create Document</td>
<td>@@Command([Compose];“formname”)</td>
</tr>
<tr>
<td></td>
<td>@@Command([Compose];“database”;“formname”)</td>
</tr>
<tr>
<td></td>
<td>Creates a new document in the current database or another database.</td>
</tr>
</tbody>
</table>

Keep in mind that Domino cannot translate commands based on a selected document in a view because there is no notion of a “selected” document on the Web. For actions such as Create Response Document, you must add a form action to the Main Topic form for opening a Response document.
Embedding a View in a $$ViewTemplate

You can embed only one view in a form and the name of the form must be one of the following two:

1. $$ViewTemplate for <viewname>
   This form name associates the form with a specific view. The view name is the alias for the view, or if no alias exists, the view name.

2. $$ViewTemplateDefault
   This form name will associate the form with any views that aren’t already associated with a specific view template.

Creating a $$ViewTemplate

In this next section we will create a view template for our view. We will copy an existing view template form in the database, add the MyCategory view to it and embed a folder pane which will create a “frame-like” effect on the Web.

1. Open the Forms Design.
2. Copy and paste the $$ViewTemplate for ByCategory.
3. Double click on the new $$Viewtemplate form to open it in design mode.
4. In the Properties Box rename $$ViewTemplate to $$ViewTemplate for MyCategory.

The ViewTemplate consists of a table. In the left table cell, there is an embedded navigator, in the top right cell there are four hotspot icons for interacting with the view and in the bottom right cell a view is embedded.

We will delete the existing embedded navigator and view. Instead we will embed our own view and a folder pane. The embedded folder pane displays...
a list of views and folders laid out as the standard Folders navigator. The folder will have URL links to the displayed views. Only one embedded folder pane is allowed on each form.

5. To delete the embedded navigator, click once on the gray rectangle and press Delete.

6. Do the same for the embedded view.

7. To embed the folder pane, put the cursor in the left table cell and click Create - Web element - Embedded Folder Pane.

8. To embed the view you created earlier, place the cursor in the lower right table cell and click Create - Web element - Embedded View.

9. In the view selection box, select the My Category View and click OK.

Tip You can use formulas to select which view to embed. In this way you can create very dynamic view templates. The following formula, for example, inserts one view during weekends and another view on workdays:

```
@If(@Weekday(@Today)=1 | @Weekday(@Today)=7;
"WeekEnd";"WorkDay")
```

To see the new view template on the Web, click Design - Preview in Web browser.
All the non-hidden views in the database are displayed on the left; the view is on the right. Notice that the default Domino navigation bar and view title are no longer generated on the view and the background color setting for the view is lost. At the top, menu actions for creating a new view, full text search, etc. are included, but the view template doesn’t have navigation buttons for expanding or collapsing the view. We will have to add these actions to the view template form.

**Adding Navigation Elements to the Viewtemplate**

1. Open the view template in design mode again.
2. Add a couple of icons that we can use as action hotspots to collapse and expand the view. The two default Domino icons are called expview.gif and colview.gif. You can either copy and paste or import the image files, or you can use pass-thru HTML to reference them on the Domino server.
3. Mark the expand icon and select Create - Hotspot - Action hotspot. In the programming pane click the @Commands button and select the ViewExpandAll command.
4. Do the same for the collapse icon and select the ViewCollapseAll command.
5. Double click on the folder pane. This brings up the Properties box for the folder pane. Change the font to bold to make the view list stand out.
6. Place the cursor in the table cell containing the view. Select Table Properties and change the background color.
7. Save your changes and open the view on the Web.

The views in the folder pane are displayed in alphabetical order and you cannot remove the bullets in the view list. Another method of including a
view list on your form is to use a combination of fields, formulas and pass-thru HTML. With this option you can exclude some views from the list, display them in any order you want, and include an icon pointing out the view currently displayed.

Creating a Customized “Folder Pane”
1. Open the $$ViewTemplate for MyCategory in design mode.
2. Delete the embedded Folder Pane.
3. Create a hidden text field and name it ViewsToList.
4. In the Formula enter the names of the views you want to include in the list.

5. In the left table create a Computed for display field and name it Views.
6. Enter the following formula for the field:

```plaintext
views := ViewsToList;
xs := "x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":"x":
ncurr := @Member(@LowerCase(@ViewTitle);@LowerCase(views));
vmask := @Explode(@Implode)
@If(ncurr=0;@Subset(xs;@Elements(views));
@If(ncurr>1;@Subset(xs;ncurr-1); ""):
"y":
@If(ncurr<@Elements(views);@Subset(xs;
    @Elements(views) - ncurr); ")
```

[108 Developing Web Applications Using Lotus Notes Designer for Domino 4.6]
The example code above generates a URL for the views listed in the ViewsToList field. First it finds the number of elements (Views) in the field, and locates the current view. It then generates the URL for the views and the HTML code for the icon. Finally it includes everything in a pass-thru HTML statement.
To test the new view you have to include the two fields in the other $ViewTemplates and remove their navigators.

Embedding Web Elements in Documents

As with forms, you can embed Web elements like Embedded Views and folder panes in the rich text field of a document you create in the database. This is useful if you are using documents rather than forms to design your Web site. Another advantage is that users can gain access to some of the Web design elements in Notes.

1. Create a new Document in our database. Give it a category and a title and move the cursor to the rich text field.
2. Select create Web element - Embedded view.
3. As with forms you can pick a view from the list or use a formula to select the view to embed in the document.
4. To change the properties for an embedded object, click on the gray rectangle and select pane properties.
5. Center the view on the page and check the Hide paragraph from Notes 4.6 or later.
6. Choose Actions - Preview in Web Browser to see how the page you are creating in Notes will display on the Web.

Note Since we selected Hide Paragraph from Notes 4.6 or later, you have to use the View - Show - Hidden from Notes option in order to see the embedded view in Notes, for example, if you want to change the properties of the view.

Using Keys to Identify Documents

When you open a view on the Web, Domino has dynamically generated the URLs for each document in the view. The document URL always includes the view object, because Domino displays the document using the views form formula if one exists. You may, for example, specify a form formula to display the document using one form when the document is read and another form when the document is being edited. If there is no form formula, Domino will use the form specified by the document’s Form item, or it will use the form stored with the document.

Domino uses universal identifiers (UNIDs) to reference the objects. The UNID is assigned by Notes when the view or document is created. The UNID is unique and consists of a string of 32 hexadecimal characters, resulting in very cryptic URLs like this one:

http://fcmobile/catalog.nsf/85255e6f0052055e85255d7f005ed8bc/3b
b3f0f15c1116f6852564d60060f978?OpenDocument

Chapter 4: Domino Views and Folders  111
The first UNID represents the view name and the second UNID a document. The first UNID in the above URL could be substituted with the actual view name:

http://fcmobile/catalog.nsf/Hardware+Products/3bb3f0f15c1116f6852564d60060f978?OpenDocument

However, the dynamic URL creation still poses two problems for the Web developer. First, the URL doesn’t convey any meaningful information about the document. A user has no way of knowing what hardware product is referenced in the URL. Secondly, the document UNID points to a specific document in the database. Suppose a user has ‘bookmarked’ a specific document in a product catalog database. If the document is then deleted and a new document describing the product is created, the new document will get a new UNID, and the user’s bookmark will point to a document that has been deleted.

To overcome these problems you can take advantage of a feature in Domino called open-by-key. Domino allows you to use the first sorted or categorized column of a view as a key to the documents, and then replace the document UNID in your URL with the a document name (the key). The key column value must be text. Domino returns the first document in the view whose value exactly matches the key in the URL. There may be more documents that match the key, but Domino always returns the first match. The syntax for the URL is:


You can use this syntax to open, edit, or delete documents. The key (documentname) must match completely for Domino to return the document. However the match is not case-sensitive or accent-sensitive.

**Example:**
Using a product catalog database as an example, we will create a small application enabling users to access the product catalog using keys instead of UNIDs. This way they will get a relative address to a product description document in the catalog instead of the absolute UNID.
First we create a view in the product catalog which selects all the hardware products and sorts the documents by product name. The view name will be Hardware Products.

We now have a view we can use to open a document by key without using the UNID. For example, to open the product description for the TP755CSE from a browser, you could use the URL:


This URL conveys more meaningful information to your users, but it still doesn’t give them an easy way to open a document in the view. They have to know both the view name and the key (product name) to compose the URL. To enable users to dynamically select a document they want to read, we need to create a form, from where the user can select a product. In our form we will have a keywords field that uses an @DbColumn formula to select all
the product names in the view and a $$Return field that constructs the URL based on the product selected.

The Product field is populated using the following formula to select every value in the first column of the Hardware Products view:

```
@DbColumn ("":""; @DbName; "Hardware Products"; 1)
```

In the $$Return field we use the following formula to generate a URL that will cause the browser to immediately open the requested document:

```
DBName:=@Subset (@DbName; -1);
URL:="/"+DBName+/Hardware+Products/
    +Product+"?OpenDocument";
@ReplaceSubstring (URL; " "; "+"
```

The document key value is pulled from the product choice made by the user. For example, if the user selected TP760C, the URL string generated by the $$Return field would be:

```
"[/catalog.nsf/Hardware+Products/TP760C?OpenDocument]"
```

We use the @ReplaceSubstring function to replace any blank spaces in the URL with plusses, in case, for instance, a product name consists of more than one word.

The Open Document button is a submit button. If you have not selected Web access: Use JavaScript when generating pages, Domino ignores any formula value for the button, otherwise include the following formula for the button:

```
```
Normally when a Web user submits a form, Domino creates a document in the database. However, in our case the form is simply used to generate a URL; we don’t want to save the document. To prevent documents from being created, we use the special SaveOption field and set its value to 0. When a Web user clicks the submit button the Web server will perform all the usual actions involved in saving a document (evaluating formulas, mailing documents, etc.) except actually saving the document.

We have now achieved what we wanted. With the combination of the @DbColumn field and the $$Return field we have managed to create a form our users can use to dynamically open any product description document in the database, by simply pointing an clicking, without having to bother about UNIDs, view names or document names. All that is left to do is to add a URL to point users to the new form.

To enhance the application further, add a keyword field to the form and name it Action. Enter the following two values: Open and Edit. Change the formula for the $$Return field to:

```
DBName:=@Subset (@DbName; -1);
URL:="/"+DBName+"/Hardware+Products/"
    +Product+"?"+Action+"Document");
@ReplaceSubstring (URL; " "; "+")
```

Users are now able to select an action on the form to open the select document in either Edit or Read mode when they click the submit button.
Summary

Views are the entry point to the data stored in a database. When users open a view, a list of documents in the database is displayed, each row presenting pieces of information from a document. As such, views give users a logical and organized overview of information available in Notes databases.

The dynamic nature of views allows application developers to design highly flexible entries to databases and Web sites, based on the users’ needs or access levels. Application developers have to keep the differences between the Notes UI and a browser UI in mind when developing applications for the Web. Web users don’t have access to Notes menu actions or the ability to act on selected documents in a view. These facilities must be provided by the developer by adding links and buttons, usually in a viewtemplate form that holds the embedded view.

The different behavior of the Notes and browser platforms has to be considered as well. Each time a Web user opens, scrolls, expands, or collapses a view, Domino converts the view to an HTML page. When developing views for the Web, developers should keep the following guidelines in mind:

- Use only two or three columns. Remove all unnecessary columns. Browsers tend to have narrower viewing windows than Notes workstations.
- Omit column headings when possible to remove clutter.
- Use a background color for a distinctive look.
- Be aware that views show 30 rows at a time by default. The setting is determined by the number chosen for Default lines per view in the Public Address Book Server document (HTTP Server - Operational Information settings).
- Avoid time-sensitive column formulas with @Functions such as @Now, @Created, @Modified, and so on. Domino Web server generates Web views as HTML pages on the fly. Time-sensitive formulas recalculate every time a Web user works in the view, for example, opening, scrolling, expanding, etc.
Chapter 5
Domino Navigators

A navigator is a ‘map’ or ‘image’ composed of several graphic items. Each item may be an activation point by which a user can launch an agent, open another navigator, open a view, etc.

A navigator is usually associated with a view or a folder.

This chapter describes what a navigator is and how to build one. It also includes some hints, tips, and examples on how to make best use of navigators in Domino Web applications.

Building a Navigator

To build a navigator, open the Property Box for the navigator and select the desired default view. This setting causes the view to be displayed on part of the screen when the navigator is opened using a Notes Client. On the Web you will see the navigator only. To achieve the same effect on the Web as the user sees when using a Notes client, you need to embed the navigator and the associated view in a form and set that form as the default form for that navigator. See Chapter 3: Designing Application Forms for more information.

In Domino Release 4.5, a Web-enabled navigator should be built using only the following items:

- A single image, pasted as the graphic background.
- Rectangular hotspots.

In Domino Release 4.6 you can use all the graphic objects that you can use to build a navigator suitable for Notes Clients, that is:

- A single image pasted or imported as background (now optional).
- Several images pasted or imported, as graphic buttons.
- Rectangular, circular, elliptic, polygonal hotspots.
- Rectangular (also rounded), circular, elliptic and polygonal shapes.
- Polygonal lines.
- Static text boxes (including those with transparent backgrounds).
- Rectangular 3D buttons (not only gray buttons).
All of the items above can perform one of the following actions:

- Open a navigator.
- Open a view.
- Alias a folder (not available on the Web because you cannot drag and drop a document using a browser).
- Open a URL. See also Chapter 6: Using HTML in Domino Design Elements, the section on Domino URLs.
- Open a link. See also Chapter 6: Using HTML in Domino Design Elements, the section on Domino Links.
- Run a formula (available on the Web, with limitations).
- Run a LotusScript Click Event (not available on the Web).

Here is a snapshot that shows all the graphic elements:

Note The difference between hotspots and shapes is that the shapes also have a background property and a text caption.
Using the Design Tools for Navigators

This section outlines the basic steps to follow when building a navigator.

Open the database and go to Design. The design is the collapsible section with a triangular icon that you should find below the views and folders pane.

**Note**  If you can’t find the Design section, verify that your access to that database is Designer or Manager. If so, choose View - Show - Design. If you still don’t have access to the design, this means that your UserID doesn’t contain a designer license, so you will need to contact your Notes administrator.

Now select Navigators to see how many navigators are already built.

To create a new navigator you can:

- Copy and Paste an existing navigator and then modify it.
- Create a navigator from scratch by choosing Create - Design - Navigator.

The design panel displays as follows:

- The Create Menu. This menu allows you to create any of the graphic objects that make up your navigator.
- The Navigator Layout Pane. This is where you can place the graphic objects in the navigator to fix their position and size.
The Properties Box. This panel opens automatically when you create an item. It allows you to set different attributes for different objects within the navigator, such as text color and character, background color (also transparent), position and size, and hotspots behavior. The Properties box can also be set to display the properties for the navigator itself. Simply select the navigator from the drop-down list at the top of the box.

**Tip** To show the Properties box for an object, put the focus on it and double click or click the diamond icon on the SmartIcons bar.

The Design Pane. Here you can set the action that a graphic object should perform when clicked. You have three possibilities:

- **Simple actions:** the kind of action you can select is to display another Domino object (see list in picture) whose name you can specify using the right drop-down list.
- **Formula:** you may enter a formula but this formula must contain @Formulas that are valid also for the Web.
- **Script:** you may write a LotusScript procedure but this doesn’t work on the Web.

**Important** To run a LS agent you can use the @Command([ToolsRunMacro];"AgentName") which will work on the Web. In this case the LotusScript procedure must be a shared agent and the client or the server must have access to run the agent on the server.

To create a graphic background or a graphic button you can:

- Import it by choosing File - Import.
- Copy and Paste it. In this instance you have to copy the image onto the clipboard and then choose Create - Graphic Background or Create - Graphic Button.

### Hints and Tips for Creating Navigators

1. To see the Hotspots boundaries on the Web page:
   - Open the Property box for the object.
   - Go to the HiLite pane.
   - Select Highlight when clicked.
2. Do not try to put HTML code into text boxes or captions. The Domino HTTP server translates the entire navigator into an image so your HTML code will not be treated as text!
3. Importing or pasting animated .gifs results in only a static image appearing on the Web.
4. Importing rather than pasting images improves color fidelity. However, Device Independent Bitmap support is introduced with Release 4.6. This improves color fidelity when pasting images from the clipboard; so it is worth testing both methods to see which gives the better result.

**Note** Support for Transparent .gifs, is also available.

5. If you need to have dynamic links, use hotspots that run an agent or a formula, paying attention, of course, to the limitations of using @formulas on the Web.

To run an agent you can use:

```
@Commands( 
[ToolsRunMacro];"<Name of your Agent here>" );
```

The agent can terminate with @URLOpen() or @Command([OpenView]) and so on, after evaluating a formula.

The agent may also be written in LotusScript. (To see how a LotusScript agent can interact with a Web browser see Chapter 6: Using HTML in Domino Design Elements, the section on agents and also Chapter 7: Domino Agents.)

6. Remember that more navigators can be embedded in a form or in the rich text field of a document. The selection of the embedded navigators to display inside the form can be controlled by a formula.

Using a form together with navigators is also useful because you can add to the form the HTML that you cannot add to the navigators to achieve special effects, for example.

---

**Example of Embedded, Calculated Navigator**

In this example our aim is to open a view from a Web browser with the related navigator displayed to the left of the view. To do this you must build a form, embed the view in it and then save the form naming it $$ViewTemplate for ViewName. Using this special name forces the display of the View Template form every time a user opens the view.

We will also include another navigator in the same form, but this will be a different navigator, depending on the language selected by the Web user.

Let us suppose that, on a previous Web page, the user has selected the language and then activated a link as in the following example:

```
http://host/dbFileName.nsf/ViewName?OpenView&Language
```

This means that on the former page we should have provided a way to compute this link based on the user selection. All this could be done using a keywords field (let’s name it SelectedLanguage) on a form that gives users the option to select a language from a list.

---

**Chapter 5: Domino Navigators** 121
Then on the form we must put a $$Return field computed as follows:

"[/dbFileName.nsf/viewName?OpenView&" + SelectedLanguage +"]"

This special field tells the browser to go to the view page immediately after the user has submitted the form.

**Caution** The square brackets are needed otherwise the user will not jump automatically to the view page after submitting.

The graphic below shows how the form would look on the Web:

![Form Example](image)

A solution to our problem is summarized in the following picture:
Notice that:

1. The Embedded navigator is based on the following formula:
   \[ @If(Langauge="Italian";"Main_Italian";
   Language="English";"Main_English";
   "Main_English") ; \]

2. The field Language is computed in order to extract the language argument string, as follows:
   \[ @Right(Query_String;"\&") ; \]

3. The field named Query_String is a special field that captures, as a CGI variable, the entire string entered into the previous URL, after the question mark (“OpenView&Italian”, for instance).

4. The Embedded view is the view we want to display with this form together with the relative navigator in the appropriate language version.

**Caution** The Query_String and Language fields must be hidden.

---

**Targeting Navigator Links into a Frame**

Using navigators with frames allows you to open several views and to display those views in a side frame while still displaying the navigator on the left of the screen. To do this:

1. Open the Properties Box for the navigator.
2. Go to the Navigator Title and change it in this way:
   \[ NavigatorTitle \langle/TITLE\rangle\langle BASE Target="Right"\rangle \] | NavigatorAlias
3. Use the NavigatorAlias in the frame content coding.

**Note** We have supposed that the target frame has the name Right.

For more information about using frames, refer to Chapter 6: Using HTML in Domino Design Elements, the section on Domino and Frames.
This is the effect that you should see on the Web:

Of course, you must always use the same navigator for the left side frame. In fact, if you try to open another navigator by activating a link on the same navigator, the new one would display in the right side frame. If you need to change the navigator depending on the displayed view, the best way is to define a template form for each navigator and embed in it the associated view. See Chapter 3: Designing Application Forms and Chapter 4: Domino Views and Folders for more information.

Summary

A navigator is a graphic image map containing several ‘active areas’ called hotspots or buttons. These hotspots and buttons can be links to other Domino objects or Web pages located outside Domino and can also have the ability to launch agents to perform specific tasks on documents stored in Domino databases.

This chapter covered how to create a navigator that will display effectively on a Web browser and also how a navigator can be used in combination with a form (an embedded navigator) to achieve a more attractive result on the Web.
Chapter 6
Using HTML in Domino Design Elements

This chapter will cover the integration of HTML in Domino applications. A Domino application consists of at least one database structured with different design elements and we will discuss adding HTML code into those elements. In the previous chapters you have discovered how to use Domino forms, views, folders, navigators, and agents. Now we are going to see how they can be enhanced using HTML.

There are two main reasons to add HTML to Domino objects:

1. To customize the end user interface for Web browser users, enhancing the standard interface provided for those users by Domino in order to make the applications more attractive and easy to use for Web users.

2. To reuse old HTML pages, either porting them into Domino documents, or linking to them leaving their code on the file system.

The first part of this chapter is composed of several sections; each one is focused on a different design object and describes how to add HTML to that particular item. As you will see, the way to insert HTML varies from one design element to another.

The second part of this chapter covers other subjects of interest for Web developers such as how to use frames in Domino applications. This part also includes a description of Domino URLs, with some examples of how to access Domino objects using keys and a paragraph about the different ways to build links in Domino.

We have assumed that readers are already familiar with HTML and so will concentrate on how HTML can be used within Domino.
Part 1: Adding HTML to Domino Design Elements

This section describes how to add HTML to Domino design elements, giving some examples with explanations. Some description of how Domino translates documents and forms into Web pages is also included at the end of this first part of the chapter.

This section is structured as follows:

- Adding HTML to a form
- Adding HTML to a document
- Formatting fields using HTML
- Adding HTML to a view or folder
- Adding HTML to other design items
- Using HTML in agents
- Testing how Domino generates HTML pages.

Adding HTML to a Form

Adding HTML to forms provides developers with a number of advantages:

- Enables the ‘look’ of the form to be enhanced for Web users by adding attractive Web effects, for example, a GIF animation.
- Enables the <BODY> attributes of the Web page to be modified, for example, to set a different background image.
- Enables input fields to be sized.
- Enables HTML links to be added, for example, to link to a non-Domino object like an “.htm” file stored on the server file system.
- Enables frames to be built into Domino applications.

The following section describes how you can insert HTML code on a form. Since a form usually contains other elements, you will also see how to use some of them (fields and computed text in particular) to store HTML.

Writing HTML on a Form

This is the easiest way to add HTML since you simply write the HTML code as static text on the form, then select it and choose:

Menu - Text - Pass-thru HTML.

In this way the code will be sent, as is, to the browser.
Note In release 4.5 you could achieve the same effect in two ways:
1. Including HTML code in square brackets “[...]”
2. Defining an “HTML” style for the text and applying it to the selected text.

You may also mix HTML and Domino elements. In the following example we have some text included between two pass-thru HTML paragraphs.

The result is a hypertext link that has a size and a character similar to what we see in design mode. If the text had been in pass-thru HTML style, the only way to format its color and size would have been to add other HTML tags.

If you want the entire contents of the form to be considered as pure HTML by Domino, do the following:

- Open the Property Box on the form properties.
- Choose the Defaults pane.
- Flag “Treat document contents as HTML.”

In this way Domino treats every document created with that form as a flat “.htm” file. Domino also will not generate the default <Head> and <Body> tags.

Note See also the section at the end of this part of the chapter on Testing How Domino Generates HTML.
HTML in Computed Fields

The value in computed fields can be an HTML string. This facility is useful when you need to have different HTML depending on a formula.

You can use Text or RichText computed fields.

The following example shows how to display different links for different users. The formula below can be the value of a computed for display, text or rich text field. The link calculated with that formula addresses a user to the view related to his role in that database.

**Note** For more information on user’s roles, please refer to the Notes Help or other Notes developer’s guides.

```html
View:=
@if(@Contains(@UserRoles;"Supervisor");"AllDocuments";
    @Contains(@UserRoles;"Operator"); "MyDocuments";
    "PublicDocuments");
"<a href=/dbName.nsf/+View+/?OpenView>Go to Documents</a>"
```

In the next example, a different pass-thru image is displayed depending on the creation date of the document. If the document has been modified today, a special icon will distinguish this document when opened.

```html
iconFileName:=
@if(@Date(LastModified)=@Today; "New.gif";"Archive.gif");
"<img src=/dbName.nsf/ImageView/Icons/$file/" +
    iconFileName + ">"
```

**Note** The image of the former example is a file attachment in a separate document. To access that image we use a sorted view (ImageView) and a key value (Icons). This allows all images that we need to be stored in a Notes document so that we don’t need to replace them in several forms each time those images need to be modified; in fact, we simply reference to an image using a URL (see also “The Domino URLs” section of this chapter).

HTML in Computed Text

Computed text is a new feature of Release 4.6 and allows you to have pieces of text calculated using a formula.

**Note** Do not mistake this feature with a normal computed text field. Computed Text is not a real field because it is not saved in a document like data stored in Notes fields. Computed Text is really a new kind of text hotspot and not a new kind of field.
To use this function choose:

Menu - Create - Computed Text

A possible use of computed HTML using computed text could be the following: imagine that you need to broadcast certain general news to all users logged onto your site; you might add a computed text at the bottom of every form with the following formula:

@DbLookUp("":"NoCache"; "":"Broadcast.nsf"; "BroadcastMessages"; "LastNew";"HTMLDescription");

Where:

- **Broadcast.nsf** is a repository of news documents.
- **BroadcastMessages** is a view containing some of this news.
- **LastNews** is a keyword that identifies one document inside that view.
- **HTMLDescription** is a text field containing the HTML code (included in square brackets) to display as a broadcast message on all forms. This means that the message could also be an elaborate HTML page instead of a simple string.

Of course, we have assumed that the document LastNews contains the news to send to all users.

Modifying the <Body> Tag Attributes

You may want to add dynamic behavior to some properties of your Web page, such as the background color or text color for example. With Release 4.6 you can access the <Body> tag and modify its attributes in the following way:

1. Select the form.
2. From the design pane, select the Event List.
3. Choose **HTML Attributes**.

Here you can add a formula to dynamically define the attributes of the <Body> tag.

Example:

AccLev:=@TextToNumber(@Subset (@UserAccess (@DbName);1));
@If(AccLev<=2; "BGCOLOR=\"Grey\""; "")

In this way a gray background is set when a user has at most Reader access to the document.

See also Chapter 3: Designing Application Forms, for more examples.
Adding HTML to a Document

A document is not the same as a form. A document is a set of data created using a form or by an agent. When a user opens a document he actually applies a form to a set of data, so the form is the user interface that displays that data.

Some of the data can itself be HTML code so that every document has its own customized design when seen through a Web browser.

All this gives more flexibility to Web content creators, because they can introduce different presentation logic elements (links, animation, navigators and so on) to each different document, without altering the design (that is the form structure).

On a document you may use RichText fields to store HTML and also modify the <BODY> attributes of that particular Web page (the document is actually a Web page when opened from a browser).

What You Can Add to a RichText Field

RichText can contain many different items. The most useful are:

- HTML code (using pass-thru HTML style).
- Links to other objects (see the sections on “Domino Links” and “Domino URLs” later in this chapter for more information).
- Embedded views, navigators, folder panes.
- Computed text (which could also be HTML strings of text as discussed earlier).
- Imported or copy/pasted graphics.
- Attached files.
- Embedded OLE objects, for example, a “1-2-3” spreadsheet.
- Text formatted with font, color, and hide-when formulas.
- Expand/Collapse Notes sections.

As an example, suppose that a Web content creator (that is, a user who can create Web pages but has no access to the design), has saved a document to advise people about a new Web application that is now available.

Imagine that this application is replicated on several servers on different sites around the world, so that it might be useful to recommend users to access one site during the morning and another one during the evening.
This user can include a computed text that evaluates a link, as follows:

```plaintext
host :=
@if(CriticalTime-@Now<0; host1; host2);
"<a href=http://"+ host +"/dbName.nsf?OpenDatabase>
   Access the Application through this link</a>"
```

Where host1 and host2 are DNS entries or IP addresses in string format.

**What You Can Modify at Document Level**

We have already seen how to modify the `<BODY>` tag attributes at form level or to simply set the background for a form. Similarly you can act at a single document level overriding what is set at form level.

**Changing the Background for Both Notes and Web Users**

1. Open a Document in edit mode.
2. Pick the Property Box of the document.
3. Use the background pane to change the color or import an image.

**Modifying the `<Body>` Tag Attributes**

This only takes effect for Web users.

1. Open a Document in edit mode.
2. Choose Menu - Document.
3. Choose Edit HTML Body Attributes and insert the attributes in the bottom pane, as in the following example:
The operations described above require some design skills, but a user can perform them without having Designer access to the database. We can consider all these tools as extended authoring instruments, useful for users playing the role of Web content creators.

Formatting Fields Using HTML

Input fields on the Web are text boxes in which a user can enter data. Domino gives default values for the size and other attributes of these boxes which developers might wish to change for a particular application.

The following steps explain how to change the default size and attributes that an input field may display on a Web page:

1. Select an Editable Field on a form.
2. From the design pane, select the Event List.
3. Choose HTML Attributes.

In this area you can put the attributes available for an <Input> tag and also for a <TextArea>, if the field is a RichText Field.

Look at this table for a quick reference.

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Syntax &amp; Example</th>
</tr>
</thead>
</table>
| All editable Fields     | “NAME=...... TYPE=.......”  
ex: “NAME=CREDITNUMBER TYPE=Password”                                           |
| Normal Editable Field   | “SIZE=..., MAXLENGTH=...”  
ex: “SIZE=40 MAXLENGTH=60” defines to 40 char. the video length of the field and to 60 the max number of char. considered. |
| Editable RichText Field | “ROWS=.... COLS=..... WRAP=.....”  
ex: “ROWS=5 COLS=60 WRAP=VIRTUAL” defines the dimension of the input box and tells the browser to wrap the text without inserting any line feed or carriage return. |
| Keywords Field          | “SIZE=...”  
ex: “SIZE=6” over 6 rows, a scroll-down control appears on the list. |

Note In Release 4.5 you would perform these changes using the Help Area of the Properties Box for the Field as there is no option for using different attributes based on a formula as there is in Release 4.6.
In the HTML Attributes, you can use a formula to define the size that a field should have on the Web page. For example, you might adjust the length of a text field to the size of its content as in the following example:

\[
\text{Len} := @\text{Length}(\text{PhoneNumber}); \\
\text{LenStr} := @\text{if}(\text{Len}>0; @\text{text}(\text{Len}); "25") ; \\
"\text{SIZE}="+\text{LenStr}+" \text{MAXLENGTH}=60"
\]

In this other example, a list box is sized to the number of the possible choices if they are less than 10:

\[
\text{NumRows} := @\text{Min}(\text{@Elements}(\text{CountryList}); 10) ; \\
"\text{SIZE}="+@\text{Text}(\text{NumRows})
\]

**Creating a Multi-Line Text Field on the Web**

If you try to use the ROWS and COLS keywords in the HTML Attributes of a normal text field, they will have no effect. In fact those attributes are typical of an HTML <TextArea> but Domino doesn’t convert a simple text field in a <TextArea>.

To size a Text Field for the Web, follow this next example:
1. Set the “Comments” text field to be editable but hidden for browsers.
2. Create a Computed for Display text field, hidden for Notes users but visible to browsers. Call it, for instance, “htmlComments.”
3. Put this field in pass-thru HTML style by selecting the field like you do for a piece of text and then choose
   
   Menu – Text – Pass-thru HTML

4. Insert the following Value formula in the “htmlComments” field:
   
   `<TEXTAREA NAME="Comments" ROWS=5 COLS=30>" + Comments + 
"</TEXTAREA>"`

5. To delete all the carriage returns and line feeds returned from the browser, set the Translation formula to the following:

   `@Trim(@Text(Comments));`

   In this way Domino uses “htmlComments” as input fields to collect data from the browser and then places all the data in the “Comments” field.

   **Tip** If you need a Multi-Line Multi-Value text field just set the “Comments” field as Multi-Value and decide which separator should be used to distinguish the values (a NewLine, probably).

**Masking Characters on Input Fields**

Sometimes you may need to hide the characters typed by a user in a field. For example, a field called *CreditCardNumber* should have its content hidden while a user is typing his code.

It is possible to replace each character typed on the screen with an “*” simply by adding the following string to the HTML Attributes of the field:

"TYPE=Password"

**Mixing HTML With Field And Static Text**

Sometimes you may need to add some HTML code but not want the whole form content to be in HTML. In the following examples we have mixed Domino elements with HTML code in order to obtain Web effects without writing the Web page entirely in HTML.

If you select pass-thru HTML as the style for a paragraph, Domino will not translate that text into HTML so that text is sent directly on the Web.
Between two HTML paragraphs you may put other Domino objects, for example, fields or static text, and use another paragraph style for them. These elements will be formatted for the Web by Domino and the resulting HTML code will be mixed with that of the two HTML paragraphs.

See the following example:

In this picture the two <BLINK> tags are in pass-thru HTML style, while the central sentence is formatted using another Notes text style. The “Click here...” sentence is also an Action Hotspot and its formula is displayed in the design pane.

**Note** The “Name” field is Computed for Display and its Value formula is: @Name([CN];@UserName));

On the Web, the above form shows a blinking link that brings the user directly to the user’s own personal document.
Another example of mixing HTML and Notes elements is shown in the following picture:

The image on the left shows the browser view while the image on the right shows how the table is defined in a Notes document (or in the form) that creates that image. The bottom right table is defined by HTML coding written inside a cell of a Notes table.

To create the HTML code for the inner table and put it into the main table, do this:

1. Create the inner table in Notes using the facilities that Notes gives.
2. Once the table is exactly as you want it, preview it in a browser.
3. Go to the HTML source code, select all the text included between <Table> and </Table> and copy it to the clipboard.
4. Return to Notes, go to the main table and paste the text in the cell where it is required.
5. Select the pasted text and put it in pass-thru HTML style.

For this kind of effect you could also create a 4 x 4 table in Notes and then merge the cells of the upper left, upper right, and bottom left corners (to merge cells, select a group of cells and choose Table - Merge Cells).

However, if you need to change the inner table depending on the value of a formula, you need to use HTML to define the inner table. Instead of static text, you can put a computed field or computed text in the cell of the main table. These computed elements might evaluate into different HTML table definitions depending on a formula. In this way you can create dynamic tables. This is illustrated by the following example.
In this example we have a computed HTML table inside a Notes table; the computed table is simply selected between two alternative tables and then displayed in the bottom right cell of the main table.

The following picture shows the sample as seen from a browser:

This form allows you to select the kind of table to display and to refresh the form in order to see the effect of your selection.

Below is the design of the above sample form:

Explanation:

- **ComputedTable** is a Computed for Display text field with the following value formula:

  ```plaintext
  @If(TableToShow="1x2"; Table1x2;
  TableToShow="2x2"; Table2x2; "");
  ```

Chapter 6: Using HTML in Domino Design Elements  137
So the value of this field is Table1x2 or Table2x2 depending on the value of TableToShow. If this last has value “”, no inner table is displayed.

Caution This file must be in pass-thru HTML style, so select it and apply this text style to it. If you don’t do this, you must add square brackets to the HTML strings (see the formulas in the Table1x2 and Table2x2 fields).

- TableToShow is a Keyword field with value list “1x2” and “2x2”. Select radio buttons style in tab 2 of the Properties Box for this field.
- Table1x2 and Table2x2 are Editable Text fields and their Default Value formula is a string containing the HTML definition of the inner table. To create these strings, just follow the procedure shown in the previous example but, before pasting the HTML into the formula (Step 4), do the following:
  1. Temporarily paste the HTML text on the form (or in a text editor) and select it.
  2. Choose Edit - Find/Replace and replace all double quotes with backslash + double quotes (that is, “ becomes \”). Now the text is ready to be pasted inside a formula and copied to the clipboard.
  3. Take the field Table*x2 put in its value formula two double quotes (“ ”) and paste the HTML text between them.
- The button, Refresh Fields performs the following action:

  @Command([ViewRefreshFields]);

  In order to have this button working you must do the following:
  1. Open the Properties Box for the Database

Note This was a simple example but you may find other ways to compute tables. For example, you might store tables in other documents and create a sorted view on those documents. In this way you have a tables database that you can access using @DbLookUp formulas to select the table you need.

HTML for the $$Return Field

There are some special fields in Domino, used for specific purposes. Usually their names begin with “$$”. In this section we are going to look at one of them: the $$Return field. This field is used to define what the browser should do after a form is submitted.

This field usually contains the HTML page that you want to display after a user has submitted a form. By default, if you don’t add a $$Return field to your form Domino sends a message (for example, “Form Submitted”) to the browser, or an error message if the operation has encountered a problem.
Although many $$ fields are no longer used in Release 4.6, the $$Return field is still needed.

Typically you use a computed $$Return to perform one of the following functions after the user has submitted a form from a browser:

1. To show a customized “Form Submitted” message.
2. To send the user directly to another Web page.
3. To launch another URL.

Usually this field is computed and you would enter a formula for the value that evaluates to an HTML page.

**Caution** Don’t use squared brackets for the HTML text unless you need it to be immediately executed by the browser.

Here is an example:

The formula below is used in a $$Return field of a form and gives a ‘thank you’ message to the author of the document.

```plaintext
NSFName := @ReplaceSubstring(@Text(@Subset(@DbName; -1));"\";"/"});
who := @If(@Left(From; " ") = ""; From; @Left(From; " "));
@Return("<body bgcolor="#FFFFFF"><center><h2>Thank you, +who +"</h2><br><h4><a href=WebRecipes>Return to the site</a>"
```

**Important** Don’t mistake this “Submit Message” for a “Document Saved” message. A form could be successfully submitted but this does not mean that a document has been saved or created. In fact, try to add a field named SaveOptions to a form and use @False as a value for this field; in this way no document is saved even if you request that it is. You would still receive the $$Return message or page but the document would not be saved. If you need a confirmation that the document has been saved then you should use the WebQuerySave Event. See Chapter 3: Designing Application Forms and the section on Using HTML in Domino Agents later in this chapter for more information.
If you want to immediately launch another URL after a user has submitted the form, follow the examples below:

**To return the same document in read mode:**
"[/" + @Subset (@DbName; -1) + 
"/"+$defaultview/"+
@Text (@DocumentUniqueID) + "]"

**Note** Instead of $defaultview you may use another view that has a sorted column on @Text (@DocumentuniqueID).

**To send to another Web site:**
"[http://"+ URL + "]"

where “URL” must be computed earlier in the code.

**To launch an agent:**
dbNameNSF:=@Subset (@DbName; -1)
DocUID:=@Text (@DocumentuniqueID)
"[/" + dbNameNSF + "/Register+User?openAgent&"+DocUID+ "]"

**Note** In this example an agent takes the personal registration document submitted by the user and, using the argument DocUNID, provides the effective user registration.

**Important** If you are using a button (not the default Submit button), for example, an action button, to save the document, make sure that the following are included in the formula:
@Command ([FileSave]);
@Command ([FileCloseWindow])

If you omit the second command, the $$Return will have no effect, and no additional page is displayed after submitting. The [FileCloseWindow] will cause another Web page to be displayed after submitting and, in this case, $$Return actually defines that HTML page.

**HTML for the $$HTMLHead Field**

Like the $$Return field, the $$HTMLHEAD field is another special field. It allows you to put code inside the <HEAD> ... </HEAD> section; everything you write as a value for this field is included inside the header and therefore it comes before the <Body> tag.

This field is mostly used to define <META> tags, to declare JavaScript functions and also for <FRAMESET> code. See the Domino and Frames section later in this chapter for more information.
In most cases you would use this field (Rich Text or Text) as a computed or computed for display field. In either case, remember to hide this field to any client. If you don’t hide it, the HTML coding on the page will be displayed to the user.

**Caution** You don’t need to select pass-thru HTML style for this field or to put the code in square brackets.

**Note** If you have selected “Treat this Document as HTML,” you shouldn’t use the $$HTMLHead field as you must enter the HTML code directly into the form. The $$HTMLHead field has no effect in this situation.

**Example: Auto-reloading A Web Page**
The form in picture below comes from the “Chile Pepper Direct Site” (See the source list in Appendix B for more information on downloading the Chile Pepper demo).

![Form screenshot](image)

This form, opened by a browser, shows a page that reloads its content every two seconds. Each time a new image (an advertisement banner) is displayed on the screen. The image is randomly selected from a previously built collection.

**Note** Performance may be an issue with this method as the page is reloaded from the server every two seconds. Another method would be to use an animated GIF file.
The $$HTMLHead field contains, as a value formula:

"<META HTTP-EQUIV="Refresh" Content=2>"

The Body field contains:

BannerNames:= @DbColumn("":"NoCache";";"Banners";1);
NumberOfBanners:= @Elements(BannerNames);
RandomBanner:= (NumberOfBanners - 1) * @Random + 1;
RandomBannerName:=
@Subset(@Subset(BannerNames;RandomBanner);-1) ;
AttachmentName :=
@DbLookup("":"NoCache";";"Banners";RandomBannerName;2);

fromList := ";
TempURL  := "/" + @Text(@Subset(@DbName; -1)) + ";/
HomeURL  := @ReplaceSubstring(TempURL; fromList; ";/");
"[<img src="" + HomeURL + "Banners/" + RandomBannerName +
"/$file/" + AttachmentName + ""]"

Be aware of the following points:

- A sorted view called “Banners” contains one document per banner; each banner is an image attached into a RichText field of the document. The view shows in column 1 the names of banners and in column 2 the file name of the attachment (see below).

- @Subset(@DbName; -1) returns the file name of the database.
The last string is a URL that opens an image stored in a document.

Tip This time the HTML is included in square brackets but you might also
select pass-thru HTML style for the Body field, and avoid using brackets.
Refer to the Frames and Domino section later in this chapter and the
JavaScript section of Chapter 10: Java Applets, Java Agents, and JavaScript
for more examples involving the use of the $$HTMLHead Field.
Example: Generating “Follow-Me Links”
Imagine you have changed the location of a certain site or more simply, that
you have moved a database to a sub-directory. You need to have a page that
helps people find the new location.
The following HTML can be used in a $$HTMLHead field:
URL:="<your new http://.......>":
"<META HTTP-EQUIV=\"Refresh\"
CONTENT=\"5; URL="+ URL +
"\"></HEAD> <h1>
This database now is at " + URL +
" and you will be there in a few seconds</h1>"

Chapter 6: Using HTML in Domino Design Elements 143


The “Refresh” keyword tells the browser that the page must be reloaded after 5 seconds (determined by the “Content” keyword). On reload, the URL specified is used instead of the original one. The result is that a page is displayed showing the new address for a few seconds, and then the browser automatically reaches the new location.

Adding HTML to a View or Folder

The standard way used by Domino to render a view on the Web is actually efficient but also esthetically poor. You may want, for instance, to replace text links with image links, or display blinking text under particular conditions and so on. To customize a view or a folder you can add HTML in:

1. Column formulas.
2. Column labels (only 64 characters of HTML code).
3. Action Button labels (only 64 characters of HTML code).

So you can use column formulas that evaluate to HTML code in order to obtain more interesting Web effects.

Important All the HTML text must be included between square brackets “[..]” and remember that, since that text is a string computed in a formula, you need to use “\" to insert double quotes and "\" to use backslashes.

Here are some examples.

Using an Image as DocLink

Putting the following code in a column formula will result in an image being shown to a browser user to enable them to open a document instead of a simple hypertext link.

```
"[<img src=/dbname/ViewName/"+ @Text(@DocumentuniqueID) + "/$file/" + @AttachmentNames +"]"
```

The column must be set as the default link column so that Domino adds the href coding to the image.

Caution We have presumed that each document has only one attached image file. If not, @AttachmentNames returns a list of names.

Note Remember that with Domino 4.6 you may select more than one column as a link column so that you can open a document by clicking on different places on the same row. (See Chapter 4: Domino Views and Folders for more information).
Using HTML to Emphasize Some Documents

In the following example different animated GIF files are used to indicate documents 24 hours old, 7 days old and all others, respectively. This is the formula:

```plaintext
@if ( @now > @adjust(releasedate; 0; 0; 0; 24; 0; 0);
@if(@now > @adjust(releasedate; 0; 0; 7; 0; 0; 0);
"[<!—-]
"[<img border=0 src="public/images/$file/chilebullet.gif"]
"[<img border=0 src="public/images/$file/chilebulletflame.gif"]
"
)
```

Note  ReleaseDate is a document field computed when the document has been saved. Images are attached to the same document; this is accessed through the “Public” View using “Images” as a key.

The example below shows what you would see from a Web browser using the above column formula. New documents are marked with a different icon while old documents are not marked.

Chapter 6: Using HTML in Domino Design Elements 145
Using HTML in Column and Action Buttons Labels

Since the amount of HTML code that you can display in labels is very limited, we are not going to give many examples but just some possible ideas for their use.

In order to improve the appearance of a view, you might, for instance, use a GIF image in place of the column label text.

The same technique could be used for action button labels. For example, you can use action buttons to switch to other views and put a descriptive GIF animation on each button or a &lt;MARQUEE&gt; scrolling text describing the content of the next view.

**Note**  A &lt;MARQUEE&gt; is a feature available only for the Internet Explorer browser. It makes text strings scroll on the screen. Unfortunately it is not possible to use very long strings in buttons and column labels and the scrolling text, also, cannot be very long. However, if you use &lt;MARQUEE&gt; in column formulas, there are no limits on text length, so you might, for example, show document descriptions as scrolling text instead of multi-line static text.

The formula for these buttons is:

``` @Command([OpenView]; "ViewName"); ```

The labels are pass-thru HTML image references and images can be stored in a particular document as attachments. The syntax of the label string is:

```
[/dbName/IconsView/Buttons/$file/imageName]
```

**Note**  In this case there is no way to use a formula for the button label.

Treating an Entire View as HTML

If you want to display a view in a completely different layout from the standard way, you can create a full HTML view or folder.

With Release 4.6 a new flag is available on the Properties Box for a View. You can open the tab with the hat icon and check “Web Access: Treat view contents as HTML.”

In this way non-formatting operations will be performed by the Domino HTTP Server so you must provide the HTML code that is necessary to display documents and hypertext links.

**Important**  You must not use square brackets for your HTML text.
Example 1
In the following example we have a view that displays its documents on rows composed of 3 items each.

To build such a view you must:

- Have an attached image file for each document, inside a RichText field of the document.
- Have a field named Progressive that contains the progressive number of the document. This field must be computed after composing the document and must uniquely identify the document in the view.
- Sort the view, using a hidden column, on the progressive number.
- A column with a formula like this:

  ```
  NSFName:=
  @ReplaceSubstring(@Text (@Subset (@DbName; -1));"\";"/");
  DirName:=
  @Text (@Subset (@Explode (@Explode (@DbName;"\");-1);"."));1));
  DocID := @LowerCase (@Text (@DocumentUniqueID));
  AttURL := "\"WebBoutique/"+ DocID +"/$file/"+ViewIcon+"";
  ```

Chapter 6: Using HTML in Domino Design Elements 147
DestinationURL := "\"WebBoutique\/+DocID+\"";
spaces:= "&nbsp&nbsp&nbsp&nbsp&nbsp&nbspnbsp&nbspnbsp&nbsp&nbspnbsp&nbsp;"]
endTag:= @If(@Modulo(Progressive;3)=0;"<P>"; spaces) ;
@If(ViewIcon !="";

"<a target="\"Products\" href=" +
DestinationURL +
"><img border=0 src=" +
AttURL+
" alt="" + Subject +
""></a></b>
endTag ;

"<!—" +Subject+"—>"

Where:
- **ViewIcon** is a field containing the name of the image file.
- **WebBoutique** is the name of the view.
- **AttURL** stands for Attachment URL (the image to show).
- **DestURL** stands for Destination URL (the document to open).
- **endTag** is spaces or new paragraph depending on the result of the division Progressive/3.
- **Spaces** is a set of HTML spaces to display between items on the same row.
Example 2
In the following example, we create a view that displays a table for each document as in the following image:

Clicking the link on the left activates a multimedia item.
Clicking the link on the right launches an agent. The agent reads the parameter (DocUID) and adds that item to the shopping list.

Note For more information on URL syntax, see Part 2, “Domino URLs” later in this chapter.

Each document contains:

- A RichText field with an attached multimedia file (a “.wav” file for instance).
- A field called FileName containing the name of the attached file.
- A Description field.
The view is composed of a single column with this title:

\(<\text{CENTER}>\text{Buy a piece of this Collection}</h2></\text{CENTER}>\)

and the following formula:

```
MediaFileHref:=
"<a href=All/" + @Text(@DocumentUniqueID) + "/file/" +
    @ReplaceSubstring(FileName;" ";"+") +
      "?OpenElement>Click for music</a>"<br>;
```

```
AgentBuyHref:=
"<a href= AddToShoppingList?OpenAgent&DocUID=" +
    @Text(@DocumentUniqueID) +
      ">Buy this item</a>"<br>;
```

```
"<TABLE WIDTH="100\%" BORDER=1>
  <TR VALIGN=top>
    <TD WIDTH="50\%">
      \(<\text{CENTER}>+MediaFileHref+</\text{CENTER}>\)
    </TD>
    <TD WIDTH="50\%">+
      \(<\text{CENTER}>+AgentBuyHref+</\text{CENTER}>\)"+
      
  </TR>
  <TR VALIGN=top>
    <TD WIDTH="100\%" COLSPAN=2">+
      \(<\text{Center}>+Description+</\text{Center}>\)"+
  </TR>
</TABLE>"
```
To create the table, it is easier to use a form and the tools for building tables. This allows you to preview it in a browser, copy the HTML source that Domino has generated, and paste it into the formula. (This procedure is also explained in the second example of “Mixing HTML With Field And Static Text,” earlier in this chapter).

Finally you need to edit the pasted formula in order to insert the variables MediaFileHref, AgentBuyRef, and the field Description in the correct places (shown above).

For this sample we have designed a table in Notes, as shown below:

```
<table>
<thead>
<tr>
<th>ListenAudio</th>
<th>BuyThis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>
```

Then, once you have the HTML code ready to be pasted in a formula, you just need to replace the ListenAudio, BuyThis and Description strings, with the above mentioned variables in order to have a computed string for the formula.

Another use for HTML views, is to make every document appear with a different character size or color within the view. The color and size of a columns entry may depend, for instance, on the level of importance of the different documents.

**Note**  This is not possible using Notes clients, only browsers.

To do this you have to use a formula in a column that evaluates to different HTML text styles based, for instance, on the value of a field “LevelOfImportance.”

**Note**  If you need to create a read-only view, that is, a view without hypertext links, it is not necessary to treat the view as HTML, you can just deselect the property Show values in this column as links for each column.

For more about Views and Folders see Chapter 4: Domino Views and Folders.
Adding HTML to Other Design Elements

In this paragraph we will explore how to use HTML in other Domino design elements.

“About this Database” and “Using this Database”

These two elements are actually the same as Notes documents so you can perform the same functions with these documents as other Notes documents. The only differences are:

1. You cannot Edit HTML Body Attributes.
2. You cannot set Treat this Document as HTML since no form is provided for these special documents.

Navigators

Even if you add text boxes to a Navigator they are rendered to the browser just as image maps (actually the whole Navigator is translated as a single image map). This means that all the HTML you put in those text boxes, will be ignored because it is not treated as text.

You can add HTML only for special purposes like instructing the navigator to target to a frame, using the field of the properties box reserved for the Navigator Name (see Chapter 5: Domino Navigators, for more information).

Subforms

Whatever is valid for a form, is also valid for a subform, except for the following differences:

1. You cannot select Edit HTML Body Attributes from the events pane of the Subform.
2. You cannot set Treat this Document as HTML since this is a property of a Form.

Shared Fields

All the considerations for using HTML in fields, also apply to shared fields. The use of shared fields could be a good way to define standards for the appearance and behavior of some fields, either in Notes or on the Web.
Using HTML in Agents

This section is not a guide to agent design and its importance in Domino applications since these topics are covered Chapter 7: Domino Agents, and in more detail in several other Domino Guides for developers.

Here we want to show how you could use an agent to dynamically build entire Web pages, where the pages do not exist on the file system but are created as an output of the agent running.

You can achieve this using the following statements in your LotusScript agents:

\[ \text{Print "<Your HTML code here>"} \]

Using several print statements containing HTML code, you can send an entire Web page to the browser. You don’t have to include the HTML code in square brackets unless you want to force the browser to go to another page. In this case, provide the new URL (within the “[...]”). For example, if you want to send the user to the HomePage database after the agent has run, type this at the end of your agent:

\[ \text{Print "[http://HostName/HomePage.nsf?OpenDatabase]"} \]

By default, Domino generates a <BODY> tag before the agent output, and there is no way to override this. This means that you cannot use agents to create frames as the tag <FRAMESET> must come before the <BODY> tag when creating frames. See Part 2 of this chapter, in the Frames and Domino section for more information.

**Note**  Notes clients will see the HTML strings, scrolling on the bottom messages bar when the agent is running.

**Note**  Don’t try to use @Prompt() to achieve the same result using an @Formulas agent; this kind of agent cannot be used to send output to browsers.

Once the agent has been defined, you have several ways to run it from a Web page:

1. Use @Command([ToolsRunMacro]; “AgentName”) in hotspots or buttons.
2. Use this URL: http://host/database/AgentName?OpenAgent. You can use this in hypertext links and it then enables you to pass some arguments to the agent as in this example:

   "<a href=/BuyThisProduct?OpenAgent"&"+ProdId +"&"+Quantity+"/></a>"
We have presumed that this formula is entered in a hotspot on a form; ProdId is a field whose value is the Universal ID of the product document and Quantity another field with the selected quantity to buy. The agent can then access the list of arguments using the DocumentContext property of the NotesSession class. For more examples, see Chapter 7: Domino Agents.

3. There are, finally two special agents that can be triggered by a form. The WebQueryOpen and the WebQuerySave agent. To use these agents you must enter the formula that is provided in the event drop-down list of a form. These agents can also access all of the document’s fields using DocumentContext. See Chapter 7: Domino Agents, for more information.

Caution  Print statements are not sent to the browser when you use a WebQueryOpen agent. This means that you cannot use this agent to create dynamic Web pages. The WebQuerySave agent can be used for this purpose but make sure that the button triggering the save action also contains @Command([FileCloseWindow]), otherwise no output will be sent to the browser. (If you are using the default Submit button, you don’t have to check for this).

Example
The following agent could be used in any database and provides a list of all the views in that database with relative document scores. To launch this agent directly from the browser command line, enter a URL with the following syntax:

http://hostName/databaseNSF/ListAllViews?OpenAgent

Where ListAllViews is the name for this agent.

This agent must be shared and set to be launched “Manually From Action Menu.” Below is the LotusScript code for the agent:

Sub Initialize
' Get the document context. Since this agent is invoked by
' OpenAgent, the context will be a temporary document
' that exists only while the agent is running.
    Dim sess As New NotesSession
    Dim db As NotesDatabase
    Set doc = sess.DocumentContext
    ' The CurrentDatabase is the database the agent is running in
    Set db = sess.CurrentDatabase
' The file name and the title are used often in this example so
' it is better to store them into a local variables.

    dbFileName=db.FileName
    dbTitle=db.Title

' Format the header for the HTML page

    Print "<html><body text="000000" bgcolor="f7f7f7">"
    Print "<center><b><a href="/" & dbFileName & "?OpenDatabase">"
    Print "Return to Database Main Page</a>
    </b></center><br>"
    Print "<b><font size=5>List All Views in " &
    dbTitle & "</font>"
    Print "<br><br>" "<font color=#ff0000>"_ & dbFileName & 
    " :&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</font><br>"

' Loop over all the views in the database. For each view,
' construct a link to the view using its UNID, and count up
' and display the number of documents in the view.

    Dim viewdoc As NotesDocument
    For all view In db.Views
        Print "<hr><b>The <a href="/" &
        dbName & "/" & view.UniversalID & 
        "?OpenView">" & view.Name & 
        "</a>"
        NumDocs% = 0
        Set viewdoc = view.GetFirstDocument
        While Not viewdoc Is Nothing
            NumDocs% = NumDocs% + 1
            Set viewdoc = view.GetNextDocument(viewdoc)
Testing How Domino Generates HTML Pages

This section documents a number of tests designed to show how the Domino HTTP Server translates forms or documents into HTML pages. These might help you understand why sometimes you don’t get the results you are expecting on the Web.

Note All these tests will also be useful when you are using frames and Domino (documented later in this chapter).

156 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
This set of simple tests should be easy to reproduce on your server.

1. Create a blank form, add two text rows as in the following example, name the form and save it.

   ![Form Example](image)

   **This Red Bold Text text is formatted by Domino**

   **This text is In Pass-thru HTML Style, so it is not formatted by Domino**

   **Important** The second row of text must be in pass-thru HTML Style.

   Now launch the form in Preview in Browser and look at the HTML source code of the page you have received on the browser.

   You should find the following at the top of the page:

   ```html
   <HTML>
   <!— Lotus Domino Web Server (Build 147.18 - Jul 7 1997 on Windows NT/Intel) —>
   <HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
   </HEAD>
   <BODY TEXT="000000" BGCOLOR="ffffff">
   <FORM METHOD=post ACTION="/WSamp.nsf/fbf95d6409b33b56852564de005f75f3?CreateDocument" ENCTYPE="multipart/form-data" NAME="_DominoForm">
   <B><FONT COLOR="ff0000">
   This Red Bold Text is formatted by Domino
   </FONT></B><BR>
   This text is In Pass-thru HTML Style, so it is not formatted by Domino
   <P>
   <INPUT TYPE=submit VALUE="Submit"></FORM>
   ```

   **Note** If you have selected Use JavaScript when generating pages in the Properties Box of the database, you will find some more code in the <HEAD> </HEAD> segment.
2. Starting from the same blank form, launch Preview in Notes first and then, once the document is opened, save the document.

Then launch the document in Preview in Browser and look at the HTML source code of the page you have displayed on the browser.

You should find basically the same result as in step 1; the only difference is in the absence of the coding that defines the submit action. This is just because the form is opened in read mode.

3. Using the same form:
   - Open the Properties Box for the form.
   - Select the Defaults pane.
   - Choose the flag Treat Document content as HTML.
   - Repeat step 1.

This is the resulting code:

```
<HTML>
<!-- Lotus Domino Web Server (Build 147.18 - Jul 7 1997 on Windows NT/Intel) -->
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
</HEAD>
<BODY TEXT="000000">
<Form METHOD=post ACTION="/WSamp.nsf/fbf95d6409b33b56852564de005f75f3?CreateDocument" ENCTYPE="multipart/form-data" NAME="_DominoForm">
This Red Bold Text is formatted by Domino
This text is In Pass-thru HTML Style, so it is not formatted by Domino
</p>
<input TYPE=submit VALUE="Submit"></FORM>
Note All the text has been treated as pure HTML.
Domino still adds <BODY> and <HEAD> tags but does not define the background.
```
4. Starting from the form of step 3, repeat step 2.

This is the resulting code:

This Red Bold Text is formatted by Domino

This text is In Pass-thru HTML Style, so it is not formatted by Domino

Note  Nothing has been generated by Domino.

These experiments show that Domino treats forms and documents in different ways. In particular, you cannot ask Domino to consider a form as pure HTML; only a document can be processed as a simple flat HTML file and sent directly to the browser.

---

**Part 1 Summary**

HTML code can be used inside Domino applications to make them more attractive and easier to use from a Web browser. To add HTML to Domino databases you must know how to use it with the different design elements that are the building blocks of those databases. Every design object has different ‘areas’ suitable for adding HTML and you must respect some rules in order to see your code working on the Web.

Part 1 explained how to add HTML to forms, views, fields, agents, etc. giving basic rules and showing several examples that suggest possible ways to add value to a Domino application using HTML.
Part 2: Frames, Links and Domino URLs.

This second part of the HTML chapter covers a number of HTML related topics and is divided into three main topics:

- Frames and Domino
- Domino Links
- Domino URLs.

Frames and Domino

The Notes client gives users different framed windows, for example, a three pane window that allows navigator, a view, and a preview of a selected document to be displayed at the same time, in different parts of the screen. Domino doesn't convert these presentation logic features automatically into HTML frames.

**Note** In fact, you can use the 3PaneUI argument of a Domino URL but you are not able to customize the resulting layout at all. (See the section on Domino URLs later in this chapter for more information.)

To solve this problem you can insert `<FRAMESET>` and `<FRAME>` coding, into forms or documents.

Since browsers like Netscape ignore frame coding if it follows the `<BODY>` tag, you may see your frames working only if you follow some special rules while adding the HTML coding to a Notes design element.

Below are some exercises to help you understand how frames can be used with Domino.

**Tip** You might find it useful to read Testing how Domino Generates HTML Coding at the end of Part 1 if you have not already done so.

**Caution** Many of the following examples require you to write or copy/paste some JavaScript or HTML code, inside a formula. The formula might be, for example, a value formula of a field or an HTML Attributes formula. In all cases you should remember the following rules:

1. Inside the text you are going to put in the formula, every double quote ("), every single quote (’), and every backslash (\), must be preceded by a backslash (\).

   For example:
   
   `<a href="http://www.ibm.com"> click here </a>`
must become:

```html
<a href="http://www.ibm.com"> click here </a>
```

2. The same text must be included between a couple of double quotes, before pasting it into the formula. For example:

```html
"<a href="http://www.ibm.com"> click here </a>"
```

**Exercise: How To Create a Simple Two Frame Web Page**

The following HTML coding defines two frames of the same size placed one below the other. The entire Web page should look as if it is divided into two sections; one showing the “People” view of the Address Book and the other one displaying the “Help using this Database” document.

You could, of course, replace the URL of the two frames in order to reference two objects inside your Notes system (or outside, if you prefer). To do this, follow the suggestions you will find in “Domino URLs” later in this chapter.

**Important** The following four tests are useful to understand how Domino treats frames but some of them end with a negative result. You can go directly to Step 4 to see a solution that works with both Netscape and MS Internet Explorer browsers.

1. Enter the HTML shown below on a blank form.

```html
<frameset rows="50%,*"
    frameborder="No" frameborder="0"
    framespacing=3 border="5" >
<noframes>
    This page requires a browser that supports frames.
    <a href="/names.nsf">Click here to use non-frames pages.</a>
</noframes>
<frame src="/names.nsf/People?OpenView"
    name="UP" frameborder="Yes" scrolling="Auto" >
<frame src="/names.nsf/$Help?OpenHelp"
    name="Down" frameborder="No" frameborder="0"
    scrolling="Auto" noresize>
</frameset>
```

2. Select the text and do Text - Pass-Thru HTML.
3. Save the form but don’t close it.
Here is a short explanation about the `<FRAMESET>` code.

- `<FRAMESET>`..`</FRAMESET>` is the definition of a set of frames.
  - `rows="50%,*"` means that the first frame takes 50% of the browser window height, while other frames take the remaining 50%. Since no `cols=" ..,.."` is used, all frames will be the width of the entire window.
  - `frameborder`, `framespacing` and `border` are attributes valid for the entire frameset. For more details please refer to any HTML guide. Be aware that the `frameborder="No"` and `border` attributes are only supported by Netscape; `framespacing` is an MS InternetExplorer feature.

- `<FRAME...>` is the definition of a frame that composes the frameset.
  - `Name=".."` sets a name for the frame; the name is used especially when you need to target a Web page to a specified frame.
  - `scrolling="Auto"` to automatically enable the frame scroll bars when a frame content is bigger than the frame’s dimensions.
  - `noresize` to inhibit frame resizing.

- `<NOFRAMES>...</NOFRAMES>` defines the message to be displayed when a browser doesn’t support frames. The message contains a hypertext link in order to help users reach a page that does not contain frames.

The following example shows what we want to have on the Web.
Testing the Form
Now try the following experiments:

Preview the Form in a Browser
- Choose Design - Preview in Web Browser.
On MS Internet Explorer you will see your frames working.
On Netscape the resulting Web page will display the <NOFRAMES> message.
If you look at the HTML coding you will find your <FRAMESET> instructions but they are ignored by the browser since they come after the <Body> tag. So the problem is how to inhibit Domino generating this tag or how we can put our HTML coding before that tag.

Preview the Form in Notes, then Preview the Document in a Browser
To do this, follow these steps:
- Choose Design - Preview in Notes.
- Choose File - Save, in order to create a document.
- Choose Design - Preview in Browser, to preview the document.
The resulting page is the same as step 1.
On MS Internet Explorer frames are working.
On Netscape the <NOFRAMES> message is displayed; the cause is the same as in step 1.

Select “Treat the Document as HTML” and Repeat Step 1
To do this, follow these steps:
- Open the Property Box for the form, on the defaults page.
- Check Treat document contents as HTML and save the form.
- Repeat step 1.
The resulting Web page will display the following message for both browsers:
“Error 500 : HTTP Web Server: Application Exception-Documents treated as HTML cannot be edited”.
Try also to change the URL displayed in the browser replacing the ?OpenForm with ?OpenRead. You will get the <NOFRAMES> message if you are using Netscape, while you will get the frames if using MS Internet Explorer.
This means that Domino doesn’t treat a Form as HTML; in fact Domino generates automatically a <BODY TEXT="ffffff"> tag.

**Select “Treat the Document as HTML” and Repeat Step 2**

To do this, follow these steps:

- Open the Property Box for the form, on the defaults page.
- Check Treat document contents as HTML and save the form.
- Repeat step 2.

This time the resulting Web page is as we desired for both browsers.

**Important** You could also have used the HTML coding as a value in a text field, perhaps a computed value; the results would have been the same.

It may seem that the only way to introduce frames is just to store them in a Notes document. Actually this is the most common way and you may find many examples on several Web sites powered by Domino. This method allows you to build applications using hidden design, but still allowing the user to customize the frames since they are simply Notes documents and not design elements.

Nevertheless, a way exists to use forms to store frames. This might be needed when you want to add some special dynamic behavior to your frames.

**How To Store Frames in a Form**

There are two ways to use a form to store frames instead of a document:

**Writing HTML into a $$HTMLHead Field**

1. Create a RichText Field (or a simple text field) on a blank form and name it $$HTMLHead.
2. Make it Computed (or Computed for Display).
3. Copy the HTML frames coding into the Value formula for that field, remembering to include it all between double quotes before saving the formula.
4. Save the form but don’t close it.
5. Just do as we did in step 1 of the previous set of experiments.

This time the resulting Web page is as we desired in both Netscape and MS Internet Explorer.

In fact this time the <FRAMESET>...</FRAMESET> text has been inserted in the <HEAD>..</HEAD> section of the Web page, as you can see by accessing the HTML source from the browser. In this instance the <BODY> tag comes after our coding and so doesn’t ‘disturb’ the frames.
Writing HTML into a $$HTMLHead Field
1. Create a RichText Field (or a simple text field) on a blank form and name it $$HTMLHead.
2. Make it Computed (or Computed for Display).
3. Into the Value formula for that field, write: “</HEAD> <!—”.
4. On the form and exactly on a line immediately following the field, write: “—>” (without quotes!). Select pass-thru HTML style for this text.
5. Now, on a line immediately following the “—>”, write the <FRAMESET> coding (You don’t have to put it into the $$HTMLHead field). Select pass-thru HTML style for all this text.
6. Save the form but don’t close it.
7. Preview the form into the Web browser.
The frames will work again this time in both browsers.
Take a look at the source of the Web page below and you will see that steps 3 and 4 are used to put a remark in the <Body> tag.

<HTML>
<!— Lotus Domino Web Server
(147.21 - Jul 28 1997 on Windows NT/Intel) —>
<HEAD>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html; charset=ISO-8859-1">
</HEAD><!—</HEAD>
<BODY TEXT="000000" BGCOLOR="ffffff">
<FORM METHOD=post ACTION="/WSamp.nsf/b972338f07a85a3985256d8006e9c40?CreateDocument" ENCTYPE="multipart/form-data" NAME="_DominoForm"><BR>
</HEAD><!—>
Now let’s see some other examples.

Storing Frames In Documents
For this example you need to access a demo called “Chile Pepper Demo Site” that you can find on the Web, at the site suggested in Appendix B.
The URL of this page is:
http://host/ChilesDirect/ChilePepperSite.nsf/WebFrameset/Retail+Store

Chapter 6: Using HTML in Domino Design Elements 165
Check this with what is on your browser command line; instead of host you should have your server host name (as an IP address or a DNS entry).

First enter the site with your Web browser and reach the “RedMarket-Retail Store” section. Now we are going to analyze the techniques used to obtain the frame effects you can see on your screen.

**Note**  All the other sections of the site are structured in more or less the same way.

The example below shows the Web page divided into four frames but the real structure is the following:

- Navigation frame on top, targeting to the Content Frame (frame name =“NavBar”)
- Content Frame in the middle (frame name =“ContentFrame”), ‘split’ into:
  - Body frame (frame name =“Body”) divided into:
    - Product document frame, showing the description of the selected product (Frame Name =“Products”)
    - Product selection frame, showing a view on the product list (Frame Name =“ProductList”)
- Advertisement/Help frame at bottom (Frame Name =“Help”)

---

166 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
Here is a snapshot of the Notes client opened on the “Chile Pepper Demo Site” database:

Looking at the design of the database you will first discover that:

- “WebFrameset” is the alias for “Web\Frameset” view.
- “Retail Store” is a key value that allow us to access a document in that view. (See accessing documents by key in the Domino URLs section later in this chapter for more information).
- The WebFrameset view shows documents using the “HTMLFrameset” form (title HTML\Frameset). To see this, open the Properties Box for the view at tab 4, and look at the value of the Form Formula.

Open the HTMLFrameset form. It is composed simply of a $$HTMLHead field, computed for display with the following formula:

```
"<title>" + Subject + "</title>" + @NewLine +"</head>" + Frameset + @NewLine
```

“Frameset” and “Subject” are fields actually stored on the document. The first of them contains the HTML <FRAMESET> definitions. Note that since the code is passed to the $$HTMLHead; it isn’t necessary to use pass-thru HTML style for the text.
Tip  Documents have been created using another form called “WebMaster” that is structured like a wizard helping users to create frame pages; it is not our objective to study that form but bear in mind that it just generates a computed “Frameset” field, based on selections made by the composer.

If we open the “Retail Store” document in Notes, using the “Web/Frameset” view, we can see the frameset coding:

```html
<title>Retail Store</title>
</head>

<frameset frameborder="0" border="0" framespacing="0" rows="94,*">
   <frame src="../WebNavbar/Retail+Store"
      name="Navbar" frameborder="0"
      border="0" scrolling="no"
      noresize marginheight="0"
      marginwidth="4" framespacing="0">
   <frame src="../WebContentFrame/Retail+Store"
      name="ContentFrame" frameborder="0"
      border="0" scrolling="no"
      noresize marginheight="0"
      marginwidth="0" framespacing="0">
</frameset>

Two frames are defined:

1. “NavBar” with its initial content being the document “Retail Store” that is under the view “WebNavBar.” This document contains the graphic navigation pane entirely built with HTML.
2. “ContentFrame” with its initial content being the document “Retail Store” in the view “WebContentFrame.”

Important  We are talking about documents, named “Retail Store,” that are not necessarily the same document! They just might happen to have the same key word.

Tip   In this case the first “Retail Store” document is actually the same as the previous one but the view “WebNavBar” uses another form to show it: “HTMLNavBar.” This form treats the document as HTML and shows only the content of the “FrameBody” field.

168  Developing Web Applications Using Lotus Notes Designer for Domino 4.6
The second “Retail Store” document (on the “ContentFrame”) shows as follows, if you open it using the “Web\ContentFrame” view.

```
<title>Retail Store</title>
</head>

<frameset rows="*,20%">
  <frame src="../WebContentFrame/Retail+Store+Items"
    name="Body" marginheight="0"
    marginwidth="0" framespacing="0">
  <frame src="../Billboard"
    name="Help" scrolling="no"
    marginheight="10" marginwidth="10"
    framespacing="0">
</frameset>

Two other frames are stored in this document:
3. “Body” with initial content the document “Retail Store Items” that is under the view “WebContentFrame.”
4. “Help” with initial content the form “BillBoard.” This form displays several advertisement banners chosen at random. (See the example in the section on “$$HTMLHead” for more information.)

The “Retail Store Items” contains:

```
<title>Retail Store Items</title>
</head>

<frameset frameborder="0" border="0" framespacing="0" rows="*,33%">
  <frame src="../WebBoutique/$first"
    name="Products" frameborder="0"
    border="0" marginheight="0"
    marginwidth="0" framespacing="0">
  <frame src="../HTMLBoutique?ReadForm"
    name="ProductList" frameborder="0"
    border="0" marginheight="0"
    marginwidth="0" framespacing="0">
</frameset>
```

Chapter 6: Using HTML in Domino Design Elements 169
These are the two central frames:

5. “Products,” with initial content as the first document in the “WebBoutique” view.

6. “ProductsList” showing the form “HTMLBoutique” opened for reading. This form takes an HTML column of the “WebBoutique” view and displays it horizontally through a RichText field.

This was an example of storing frames into documents. This way of developing Web pages is much more similar to the traditional one; the only difference is that you no longer need to reference the file system since all pages are Notes documents.

In the next example you will see frames stored in forms so that documents are used only to store real data and not to hold the <FRAMESET> coding.

**Storing Frames In Forms**

In this example two forms are used to create frame effects. The aim is to have a Web page which displays a view of a Technical Forum database in one frame, and to display the selected document in another frame, as shown below:

Let’s call the top frame “View” and the bottom one “DocumentPane”. When a user clicks a document in the view, this is displayed in the “DocumentPane” frame.
If a user selects a response (Child) document, we would like this displayed, along with its main topic (Parent) document as in the figure below:

Let’s call the left bottom frame “Parent” and the right bottom one “Child.”

To achieve this layout, you must have two forms; one stores the “View” and “DocumentPane” frames, the other defines the “Parent” and “Child” frames, or only one of the two depending on the document to display.

This last form will be used as a form for the document selected and hence it must be introduced in the view form formula.

We will now go, step by step, using the view “WebForum” (view title is “Web/Forum”) through part of the Chile Pepper Demo Site, creating our own design elements in that database in order to achieve this effect.

**Caution** This sample is based on the Chile Pepper Demo Site so it would be useful if you have a copy of this database installed on your server. Even if you haven’t you can still follow this example using another database; the only things you must change are the URLs and the name of the view that you want to use.
1. Create a form as in the following example and name it “MainFrame.”

![Image of Lotus Notes form]

It is better to hide the $$HTMLHead Field to avoid duplication of HTML on the Web page.

As you can see we are instructing the browser to display the “WebForum” view at the top and an initial “Using this Database” document into the bottom frame.

The $$HTMLHead contains this value formula:

"</Head>"</!

Here is a copy of the code that follows the $$HTMLHead in the figure above:

```html
<frameset frameborder="No" frameborder="0" framespacing=3 border="5" rows="20%,*">

<noframes>
This page requires a browser that supports frames.
<a href="/ChilesDirect/ChilePepperSite.nsf/">Click here to use non-frames pages.</a>
</noframes>
</frameset>
```

172 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
<frame src="/ChilesDirect/ChilePepperSite.nsf/MainForum?OpenView"
     name="View" frameborder="Yes" scrolling="Auto"/>

<frame src="/ChilesDirect/ChilePepperSite.nsf/$help?OpenHelp"
     name="DocumentPane"
     frameborder="Yes"
     scrolling="Auto" />

</frameset>

Tip Here we used a trick to comment the <BODY> tag. Take a look at the value of $$HTMLHead and the following “—>” characters (without quotes). Remember that these characters must be in pass-thru HTML style! (See also “Writing HTML into a $$HTMLHead Field” earlier in this section for more information).

2. Create a second form and name it “ParentChild”.
   This form must have only a $$HTMLHead Computed for Display text field, with the following formula:
   
   hasParent:= @IsResponseDoc;
   cols:= @If(hasParent;"40%,*";"100%,*");
   srcChild:= "/ChilesDirect/ChilePepperSite.nsf/WebForum/"+
     @Text(@DocumentUniqueID)+"?OpenDocument";
   frameChild:= @If(srcChild="" ; "" ; "<frame src="+srcChild+
     " name="Child" frameborder="Yes" scrolling="Auto" />");
   
   srcParent:= @If(hasParent;
   "/ChilesDirect/ChilePepperSite.nsf/WebForum/"+
     @Text($Ref)+"?OpenDocument" ; "");
   frameParent:=@If(srcParent="" ; "" ; "<frame 
   src="+srcParent + " name="Parent" frameborder="Yes"
   scrolling="Auto" ">");
   
   Frameset:=
   "</Head>"+@NewLine+
   "<frameset frameborder="No" frameborder="0"
   framespacing=3 border="5" cols=""+cols+"">"+ @NewLine +
   "<noframes>This page requires a browser that supports frames." + @NewLine +

Chapter 6: Using HTML in Domino Design Elements  173
Frameset;

Here is some additional explanation about the above formula:

- **hasParent** is “True” if the document is a response document; this means that a Parent document exists.
- **cols** is used in the FRAMESET string to define the width of the inner frames; if the document has a parent, two frames must be created otherwise only one is needed.
- **srcChild** contains the URL for the document.
- **srcParent** contains the URL for the parent document if this exists, otherwise its value is an empty string. The URL is built using the UniversalID of the parent, which is stored in the $$Ref field (a special field) of the child document.
- **frameChild** is the HTML <FRAME> definition for the document.
- **frameParent** is the HTML <FRAME> definition for the parent; if this doesn’t exist.
- **Frameset** is the string that defines the entire HTML <FRAMESET> code that we need.

Note the following, when analyzing the formula:

- This form is used to display documents because it will be inserted in the **form formula** of the view “MainForum.”
- The frames are computed differently depending on the type of document. In particular one frame is not allocated if the document is a main topic.
- The document(s) that are then put into the frame(s), are accessed through the original view “WebForum;” if we use the “MainForum” view, documents are displayed inside each frame using the form “ParentChild” and hence we have an undesired frame-within-frame effect.

**Note**  We are storing frames into a computed $$HTMLHead field in order to have the frame coding before the <Body> tag generated by Domino.

174  Developing Web Applications Using Lotus Notes Designer for Domino 4.6
3. Create a copy of the view “WebForum” and call it “MainForum.”

The column formulas for showing main topics and responses must be changed as follows:

************************
Main Topic column
************************
Name := " " + @Name([CN]; From);
Descends :=
   @DocDescendants(""; ", % response)"; ", % responses)"
ViewName := @ReplaceSubstring(@Subset(@ViewTitle ;-1);" ";"+");
DocID := @LowerCase(@Text(@DocumentUniqueID));
DestinationURL := ""+ViewName+""+DocID+""
"[<b><a Target="DocumentPane" href=""+DestinationURL +">" + Subject + "</a></b>" + Name +
Descends +"<!——>"
************************
Response column
************************
Date := ", " + @LowerCase(@Text(ReleaseDate; "D2T1"));
Name := " " + @Name([CN]; From);
ViewName := @ReplaceSubstring(@Subset(@ViewTitle ;-1);" ";"+");
DocID := @LowerCase(@Text(@DocumentUniqueID));
DestinationURL := ""+ViewName+""+DocID+""
"[<b><a Target="DocumentPane" href=""+DestinationURL +">" + Subject + "</a></b>" + Name +
Date +"<!——>]"
Note that:

- Actually the significant difference from the original formulas, is the added “TARGET="DocumentPane"” attribute to the <a>..<a> link.
- The form formula for the view must be set to: “ParentChild.”

Now you can go to the browser and launch:

http://host/ChilesDirect/ChilePepperSite.nsf/MainFrame?OpenForm

or preview in browser the form “MainFrame.”

Overview Of The Domino 4.6 FRAMESET Template

A set of templates for Web application developers is available with Notes Designer 4.6. One of these is a tool for building framesets and storing them in documents.

To understand this template, first open the sample database that should be under the “samples” directory. Take a look at this sample, first using a browser and then using a Notes client.

The view All Pages helps you understand how the application (actually a small Web site) has been built.

- **Frames Sets** are built using the form Frame Set. This is where the frames code is stored.
- **Frames with Content** are built using the form named Page or HTML. These pages contain real information for the site.
- **Frames with Links** are built using the form FrameDesignF (column of Links To TargetF Frame) or FrameDesignD (Site Map Links to Target Frame). These documents contain the definitions of several 'bars of links hotspots' that can be used as a content in some frames. FrameDesignD contains the site map that is actually the Site Map view.

Looking at the design pane, you will see that for some forms there is a corresponding form for the Web that has the same alias. One form is hidden to Notes while the other one is hidden to browsers so that Domino 4.6 can use the appropriate form for the right client.

The view Site Map usually includes all the documents that store real information for the site, such as the Home Page and the other areas of the site. You may decide what to add to this view and, usually, you store Frames with Content in this instance.

Let us see how to create these documents. Since the forms function provides on-line explanations, we recommend that you use that; we will not go into too much detail here.
Below is a screen-shot of the *Create* Menu:

1. Create a link list that is used, as a default, every time you create another “Column of Links to Target Frame.”

2. Create a page that holds a column of links to be used elsewhere. For example, take the graphic buttons displayed in the next picture. These buttons are actually pre-built navigators but you can substitute them referencing images using pass-thru HTML (See the instructions available on the form used to build these documents for more information on how to do this).
   - Create a Site Map that targets its links where you decide.
   - Create a Web Page without using HTML (actually if you want to use it here you can include it between “[..]”)
   - Create an HTML Web Page. In this case you don’t need to use square brackets and you must provide all the HTML code since Domino will send this page directly to the browser.

   **Note**  For this kind of document, a field called HTML has been used to store the code; although this is equivalent to having a form that treats the document content as HTML, this feature is not used here, probably in order to maintain compatibility with Release 4.5

3. Create frame set for storing sets of frames in documents.

For this last step the form used acts like a wizard so that you don’t need to know HTML FRAMESET code. You could always avoid using the wizard, and type the code directly if you wanted.

The figure below will give you an idea of the kind of wizard you might use to create your frames. In this case the form-wizard helps you build four frames and for each of them you simply select some properties, such as dimensions, scrolling behavior and the initial content. You can give a name
to the frameset and decide its positioning on the site map. The wizard also helps you to introduce <META> tags in the header of the Web page (look at the two buttons at the bottom of the example).

Web Page Management

Site Map
Include this page in the site map?  □ Yes  □ No
The site map is a view that provides easy navigation to pages on your site.
The site map view is divided into categories, which you can specify. E.g., “Company Info”, “Products”, etc.

Advanced HTML Features
□ Specify META tags
META tags can help search engines index your site.
These tags will be placed in the HTML header.

□ Preview HTML Header

178 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
This is what you get on the Web when you open the document described earlier:

![Image of frameset with links to different sections](image)

**Note** From the *(Images)* view you can create documents to store images as attachments. These documents are also accessible by key using the name of the image file.

All pages are included in a hidden view called *(LookUp)*. That view is used to build URLs by keys. The *(LookUp)* view has a sorted column first that shows the unique name that identifies a page. This name is used as a key for accessing the page. (See also the Domino URLs section later in this chapter.)
The (LookUp) view looks like this:

Frameset documents are displayed to the browser using the form: “Step 3: Frame Set-Web.” Looking at this form you may see what use has been made of the $$HTMLHead field together with the field that actually contains the frameset statements. In this way you can understand which forms and shared fields you should copy in another database in case you want to use the FRAMESET Template only to build frames and not to store them.

An alternative and easier way is to copy the HTML source from the browser and paste it into your documents or into a form just as we have seen previously in this section.

Since you can also display objects stored in other databases in frames and not only in documents inside the FRAMESET database, you may keep this database only for administering frames that you need for other applications. In this way the application databases will contain only forms and documents needed for that application, while framesets are kept in another database.

**Note** Since frames are often the entry points for Web applications, you must adjust the Access Control List for the frameset database in order to reflect that of the application database.
Targeting to a Frame

How can you define in which frame the next Web page should be displayed?

This kind of instruction is not inside the frame set definition but is an attribute of hypertext links. The following `<a>`..`</a>` link makes the referenced page display in the “Left” frame where “Left” is the name of a Frame.

```
<a href="http://www.ibm.com/" Target="Left"> IBM </a>
```

Hence it is links that contain the target statements.

If you need all links on a page to have the same target frame you must use the following code inside the `<HEAD>...</HEAD>` of that page:

```
<BASE Target="Left">
```

**Note** Remember that to add code to the header of a form you can use a hidden `$$HTMLHead` text field (see Part 1 “Adding HTML to `$$HTMLHead Field”.

This is useful if you need to target all entries of a view to a specific frame, without having to modify all column formulas, in order to add the Target attribute to all document links. In this case do the following:

1. Embed the view in a form called `$$ViewTemplate for ViewName`; in this way the view is always displayed using that form.
2. Put a hidden `$$HTMLHead` field on the form and write this for its value formula:

```
"<BASE Target="Left">"
```

All documents in the view will be displayed in the frame you have specified but also any other link that is inside that view. For example, if you click to expand a category of the view, you will find the expanded view in the same frame you have specified in the `<BASE>` tag, and not in the frame that originally displays the view. Therefore, it is better to use this targeting method only for non-categorized views.

---

Domino Links

Domino allows you to create several kinds of links; some of these work with both a Notes client and a Web browser. All HTML links work only on the Web, while just the HTML coding is visible in Notes, so remember to hide those links to Notes.

---

Chapter 6: Using HTML in Domino Design Elements  181
Notes Links

These links are defined in Notes but will also work on the Web.

- Database Links.
- View Links.
- Document Links.
- Anchor Links (only in Release 4.6).

To create them:

- Select a document from a view.
- Choose Menu - Edit - Copy as Link, and choose the kind of link.
- Open the target document in edit mode, move into a RichText field.
- Choose Edit - Paste and save the document.

**Note** Instead of using a document as the target in the final step above, you could have a form, an “About This Database” or a “Using This Database.”

An Anchor Link allows you to jump to a specified row (actually a label) inside a specific document. To create an Anchor link the procedure differs:

- Open in Edit mode the document to link.
- Go to the area you want to link to and put the mouse focus in it.
- Choose Menu - Edit - Copy as Link, and choose the Anchor Link.
- Name the link then save the document.
- Open the target document in edit mode, move into a RichText field.
- Choose Edit - Paste and save the document.

**Note** The Anchor Link is one-way; this means that you cannot use them to navigate backwards.

At the end of this procedure, an icon like the one shown in the figure above, appears on both a Notes client and a browser. Using a browser, the icon is a hypertext link whose URL is based on the Universal ID of the Notes objects (See “Domino URLs”).
Hotspot Links

You can create Hotspot areas by selecting a piece of text, an image in a rich text field of a document or a part of a form, and then choosing:

Create - HotSpots - choose one of the following:

Link Hotspot
This operation requires that you have previously copied a link to the clipboard. This causes the entire selected area to become a hypertext link on the Web and a Notes Hotspot Link if you are using a Notes client.

URL Link
A property box opens automatically. You can write a URL in absolute notation or relative notation. For instance:


Caution  Don’t use quotes or @Formulas.

For relative addressing the assumed base URL is: http://host/ where host is the DNS entry of the Domino server (or its IP address).

Domino databases are supposed to stay within the /Notes/Data/ directory while HTML text files should stay under the /Notes/Data/Domino/HTML/ directory.

The use of relative, in place of absolute, URLs is suggested because it enhances the portability of Domino applications. If you use relative addressing exclusively in all your hypertext links, you can move the database from one server to another without having to change anything.

Action Hotspot
You can introduce a formula using this type of Hotspot. The formula then decides which Domino object must be opened.

Remember that only a subset of @Formulas work also on the Web (see Appendix A on Features to Avoid When Developing Web Applications). The most useful are:

@Command([OpenView];"ViewName");"key");
@Command([OpenDocument]);

or

@Command([FileOpenDatabase];"";"DatabaseName";"ViewName");"key");
@Command([OpenDocument])

Chapter 6: Using HTML in Domino Design Elements  183
Examples:

@Command([OpenView];"Registered Users";@UserName);
@Command([OpenDocument])

or

@Command([FileOpenDatabase];"":"Register.nsf";"Registered Users";@UserName);
@Command([OpenDocument])

**Important** The above commands must always be used in pairs.

To refine the @Command, append the optional argument for [OpenDocument], ("1" to edit, “0” to read).

**Caution** Only the @Command combinations listed above are valid. For example, you cannot combine:

@Command([OpenView];"ViewName";"key");
@Command([EditDocument])

To create documents you can use the following commands.

**Note** The second command is for a form stored in another database. The first parameter is the server name but it has no use on the Web.

@Command([Compose]; "FormName");
@Command([Compose];":"; "Database"; "FormName");

To open Navigators use:

@Command([OpenNavigator]; "NavigatorName");

To open a URL:

@URLOpen("Absolute or relative URL");

**Caution** Text and Formula Pop-Up are not useful for links and are not supported on the Web.

**Button Hotspot**
In Release 4.5 all buttons created on a form had no effect on the Web; only the first one on the form was a visible button and it worked as a Submit button no matter what formula it contained.
With Release 4.6 you can use multiple buttons also on the Web. The only thing you must do is:

1. Open the Properties Box for the database.

The rules as to which formula to use in these buttons are the same as for Action Hotspots.

**HTML Links**

You can use pass-thru HTML to write HTML anchors on Notes objects (documents, forms, views, button labels,...) as shown in the previous sections of this chapter.

Remember to select Text - Pass-thru HTML style so that your HTML coding is sent directly to the browser.

If you cannot select that style (in a view, for example), you can force Domino to consider your text as HTML, by including it in square brackets.

Example:

```
"[<a HREF="/database/FormName?OpenForm"> Click here to create a document </a>"
```

**Note** The escape sequence \" is necessary only if you are writing a formula; you don’t have to use it if you are typing in a document.

**Important** We recommend again that you use relative URLs instead of absolute URLs for the reasons we have discussed in the URL Links section of this chapter.

**Explicit URL**

Every URL typed as is, is automatically transformed into a hypertext link. This works in Notes and, in Release 4.6, also on the Web. You simply have to insert a string beginning with: "http://.............." inside Notes documents, forms or columns headers and when the user clicks on it, it will activate the URL.

**Note** You don’t need to include the URL in square brackets, nor use pass-thru HTML style.
Domino URLs

Domino uses URLs to access servers, databases, and other components of a Web site and display them to Web users. Knowing Domino URL commands allows you to design links or enter commands directly into a browser to navigate a Domino site or reach specific components quickly.

**Domino URL Command Syntax**

Domino URL commands have the syntax:

```
http://Host/Database/NotesObject?Action&Arguments
```

where:

- **Host**
  - DNS entry or an IP address

  **Caution**  Do not use the Notes Server name unless it has a DNS entry.

- **Database**
  - Can be one of the following
    1. The *database file name* with the path relative to notes/data.
    2. The database *Replica ID*.

- **NotesObject**
  - A Notes construct (a view, document, form, navigator, agent, etc.)

- **Action**

- **Arguments**
  - A qualifier of the action. For example, Count = 10 combined with the ?OpenView action limits the number of rows displayed in a view to 10.

Use the following guidelines when working with Domino URLs:

1. Special identifiers used in Domino URL commands include: $defaultView, $defaultForm, $defaultNav, $searchForm, $file, $icon, $help, $about. These special identifiers are described in the following sections of this chapter.

2. **NotesObject** can be any of the following:
   - for a database, it is the database name or replica ID itself.
   - for other objects, the Notes object’s name or alias, universal ID, NoteID or special identifier. For example, to specify a view in a URL,
you can use any of the following: the view name, view universal ID, view Note ID, or $defaultView.

3. A Notes object’s name and universal ID are identical in all replicas of a database, but the Notes object’s NoteID will probably change in database replicas. Therefore, it is best to use the Notes object name or universal ID in URLs.

4. Action can be explicit or implicit.
   - If you do not specify an action, Domino defaults to the ?Open action.

5. Append the Login argument to any Domino URL to require user authentication.
   - Because URLs may not contain spaces, use the + (plus sign) as a separator. For example,
     http://www.mercury.com/discussion.nsf/By+Author

6. Separate arguments with & (ampersands). For example:
   http://www.mercury.com/leads.nsf/By+Salesperson?OpenView&ExpandView

7. Separate hierarchical names with / (slashes). For example, to open a view named Docs\By Author in a database named Discussion, enter
   http://www.mercury.com/discussion.nsf/Docs/By+Author

**Opening Servers, Databases and Views**

The following commands access servers, databases, views, About documents, help documents, and database icons.

**OpenServer**
Syntax:  
http://Host

Example:
http://www.mercury.com/?OpenServer

**OpenDatabase**
Syntax:  
http://Host/DatabaseFileName?OpenDatabase
http://Host/DatabaseReplicaID?OpenDatabase
Examples:
http://www.mercury.com/leads.nsf?OpenDatabase
http://www.mercury.com/852562F3007ABFD6?OpenDatabase

Arguments
3PaneUI This argument tells the HTTP server to use a 3 frames window to display the navigator, the view and the preview pane. The resulting effect is a simulation of the Notes client as in the image below.

Caution This argument is officially still unsupported and there is a way to inhibit this feature to the Domino server; you just have to set it in the notes.ini file, DominoEnable3PaneUI = 0.

OpenView
Syntax:
http://Host/Database/ViewName?OpenView
http://Host/Database/ViewUniversalID?OpenView
http://Host/Database/ViewNoteID?OpenView
http://Host/Database/$defaultview?OpenView

Examples:

188 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
Arguments (Optional)
Append optional arguments to refine the URL. Combine any of the following arguments for the desired result.

- **Start = n** Where \( n \) is the row number to start with when displaying the view. The row number in a hierarchical view can include sub indexes, for example, Start=3.5.1 means the view will start at the third main topic, subtopic 5, document 1.
- **Count = n** Where \( n \) is the number of rows to display.
- **ExpandView** Displays the view in expanded format.
- **CollapseView** Displays the view in collapsed format.
- **Expand = n** Where \( n \) is the row number to display in expanded format in a hierarchical view. Do not combine this argument with the ExpandView or CollapseView arguments.
- **Collapse = n** Where \( n \) is the row number to display in collapsed format in a hierarchical view. Do not combine this argument with the ExpandView or CollapseView arguments.
- **StartKey = string** Open a view starting from the first document that matches the key. The key is selected on the first sorted column.

Examples:


**Note** To open the first document in a view, use keyword $First and the following syntax: http://host/database/view/$First.

**OpenAbout**
Use the OpenAbout command to access the “About database” document.

Syntax:

http://Host/Database/$about?OpenAbout
OpenHelp
Use the OpenHelp command to access the Help document.
Syntax:
  http://Host/Database/$help?OpenHelp
Examples:

OpenIcon
Use the OpenIcon command to access the database icon.
Syntax:
  http://Host/Database$/icon?OpenIcon
Examples:
  http://www.mercury.com/leads.nsf$/icon?OpenIcon

Login Argument
Append the Login argument to any Domino URL to force user authentication regardless of the database access control list. This ensures that anonymous Web users who weren’t initially prompted for a name and password when they entered the site are required to supply a name and password to complete tasks that require user identity.

Note Do not use this argument to let a Web user switch login. In fact if a user has already logged in with a certain UserID and Password, the login argument will be ignored. The only way to re-login as a new user is to close and then restart the browser.

Syntax:
You can use this argument with any URL, but the most common are:
  http://Host?OpenServer&Login
  http://Host/Database?OpenDatabase&Login
Examples:
Opening Forms, Navigators And Agents

The following commands open forms, navigators, and agents in a database.

**OpenForm**

Syntax:

- `http://Host/Database/FormName?OpenForm`
- `http://Host/Database/FormUniversalID?OpenForm`
- `http://Host/Database/FormNoteID?OpenForm`
- `http://Host/Database/$defaultform?OpenForm`

**Note** FormName can be also an alias of the form.

Examples:


Arguments (Optional):

**ParentUNID** = The *Universal ID* of the parent document, to respond to or to inherit from. Remember that if you are composing a response document or you want to inherit formulas from another document on the Web, you cannot select the parent document.

Syntax for using this argument:

- `http://Host/Database/FormUniversalID?OpenForm&ParentUNID`

Examples:

- `http://www.mercury.com/products.nsf/40aa91d55cle4c8285256363004dc9e0?OpenForm&ParentUNID=6bc72a92613fd6bf852563de001f1a25`

**Note** You can also use the ?OpenRead or ?ReadForm command to open a form only in read mode. This is useful when you don’t want to display the submit button.

**OpenNavigator**

Syntax:

- `http://Host/Database/NavigatorName?OpenNavigator`
- `http://Host/Database/NavigatorNoteID?OpenNavigator`
Examples:
http://www.mercury.com/products.nsf/7B5BC17C7DC9EB7E85256207004F8862?OpenNavigator

Note  $defaultNav opens the folders navigator in a database.

OpenAgent
Syntax:
http://Host/Database/Agentname?OpenAgent
http://Host/Database/AgentUniversalID?OpenAgent
http://Host/Database/AgentNoteID?OpenAgent

Examples:
http://www.mercury.com/sales/leads.nsf/0000021A?OpenAgent

Opening, Editing and Deleting Documents
The following commands manipulate documents in a database:

OpenDocument
Syntax:

DocumentKey is the contents of the first sorted column in the specified view. For more information, see the section on Opening documents by key later in this chapter.

Note  The View is a necessary parameter because Domino uses the Form Formula of a view to determine the form to use when displaying the document (either by using a Notes client or a browser). If this formula is set to nothing, Domino uses the form written in the “Form” field of the document.
Examples:

**EditDocument**
Syntax:
Example:

**DeleteDocument**
Syntax:
Example:

**CreateDocument**
The CreateDocument command is used as the POST action of an HTML form. When the user submits a form, Domino obtains the data entered in the form and creates a document.
Syntax:
http://Host/Database/Form/?CreateDocument
Example:

**SaveDocument**
The SaveDocument command is used as the POST action of a document being edited. Domino updates the document with the new data entered in the form.
Syntax:
Opening An Anchor Link

As seen in the previous section on Domino Links, a new syntax exists for opening a document at a specified area of its text.

Syntax:


Example

http://localhost/RedBook.nsf/66aa0bd809ee8316852564d8004e7ddc/503d4ee771078042852564e400598a8e?OpenDocument#Paragraph+3

The Anchor label is created when you build the anchor inside the linked document. If no name had been entered for that link, a default number is added (Example: “#_0”).

Opening Documents By Key

The following commands allow you to open a document by key, or to generate a URL to link to a document by key.

Using Domino URLs to Access a Document

To open a document by key, create a sorted view with the sort on the first key column. Then you can use a URL to open the document:

Syntax:


Example:


Where View is the name of the view, and DocumentName is the string, or key, that appears in the first sorted or categorized column of the view. Use this syntax to open, edit, or delete documents, and to open attached files. Domino returns the first document in the view whose column key exactly matches the DocumentName.

There may be more than one matching document; Domino always returns the first match. The key must match completely for Domino to return the document. However, the match is not case-sensitive or accent-sensitive.
Note that view can be a view Note ID, UNID, or view name. In addition, the implicit form of any of these commands will work when appropriate. (EditDocument and DeleteDocument must be explicit commands.)

Advantages of Using Keys Instead of Universal ID

Let’s suppose that the following two URLs refer to the same document, which is the personal document of "Jay Street" (for instance):

```
http://www.mercury.com/register.nsf/
Registered+Users/Jay+Street?OpenDocument

http://www.mercury.com/register.nsf/
a0cefa69d38ad9ed8525631b006582d0/
4c95c7c6700160e2852563df0078cfeb?OpenDocument
```

The first URL is of course much more understandable than the second URL and so, easier to remember. But this is not the only advantage of using keys to refer to Domino objects.

Imagine that the “Jay Street” document has been deleted and replaced by a new copy. The reasons for this event might be several; for example, all documents of that kind are deleted and rebuilt every night by an agent that takes updated data from an external data source. In this situation the first URL continues to reference the “Jay Street” document while the second URL fails because the Universal ID of the new document will be different from the former. All this means is that if a user has stored the first URL as a Bookmark he will have no problem finding his document again and so will avoid repeated searches.

However, you should also consider the following implications of using URL by Keys:

- A developer must use @Formulas to calculate every URL using keys, and must hide or inhibit all the URLs automatically generated by Domino. This means that you don’t have to copy and paste links to Domino objects.
- The access to documents is faster when using Universal ID than when using keys because the server must read the view index to reach the document when using keys.
Example
This example shows how to use a form to select a document in a view. The form helps the user to compose the URL of the document he is looking for. The URL generated doesn’t contain any Notes ID number since documents are accessed by key.

This is the form as seen on a browser:

As you can see, the key used for searching documents is multiple; in fact an expense report is uniquely determined by the employee’s name, plus the month, plus the year.

This is the form design:

Where:
- **EmpName** is a keyword field whose keywords list is computed as follows:
  ```
  @Unique(@DbColumn("";"":"; "ExpenseReports";2));
  ```
ExpenseReports is a view (hidden if you want) having the following column formulas:

- Column 1 (Sorted):
  
  \[
  \begin{align*}
  \text{Key} & := \text{EmpName}+\text{Month}+\@\text{Text}(\text{Year}); \\
  \text{WebKey} & := \@\text{ReplaceSubstring}(\text{Key}; "\\"; "); \\
  \text{WebKey};
  \end{align*}
  \]

- Column 2 (Sorted):
  
  \[
  \text{EmpName};
  \]

In this way the triple unique key is transformed in one string. The character “/” is replaced with “.” because the key we will use to compose the URL cannot contain “/”.

- Year and Month keywords containing all values you need.
- SaveOptions computed with value “0”, so that this form cannot generate documents.
- MultipleKey computed as follows: \text{EmpName}+\text{Month}+\@\text{Text}(\text{Year})
- $$\text{Return}$$ computed for display and its formula is:

\[
\begin{align*}
\text{dbName} & := \@\text{Subset}(\@\text{DbName}; -1); \\
\text{WebKey} & := \@\text{ReplaceSubstring}(\text{MultipleKey}; "\\"; "\\+\":".;");
\end{align*}
\]

"[/"+\text{dbName}+"/\text{ExpenseReports}="/"\text{WebKey}="/?\text{OpenDocument}]

In this way documents are opened after clicking the button (simply a ‘Submit’ button) and if you look at the URL you will find something like this:

http://localhost/WSamp.nsf/ExpenseReports/

Even this URL is still long, but it is still more understandable than a URL based on a Universal ID.

**Opening Image Files, Attachments, and OLE Objects**

The following commands open files and objects within a document:

**OpenElement**

Use the ?OpenElement command to access file attachments, image files, and OLE objects. This is very useful when writing pass-thru HTML; for example when you need to display an image that is attached to another Notes document you can use the URL described here.
Using OpenElement with File Attachments
Syntax:

InternalFileName/Filename?OpenElement

Example:

http://www.mercury.com/lproducts.nsf/
By+Part+Number/SN156/$File/spec.txt?OpenElement

If more than one attached file has the same name, the URL includes both the internal file name as well as the external name. Since the internal file name is not easily determined, make sure all attached files have unique names.

Domino treats all file attachment OpenElement commands as implicit commands, because some browsers require that the URL ends with the attached file name.

Using OpenElement with Image Files
It is used to retrieve an image that is an imported image into a Notes field.

Syntax:

http://Host/Database/View/Document/FieldName/
FieldOffset?OpenElement&FieldElemFormat=ImageFormat

FieldOffset is represented by xx.yy, where xx is the field number, and yy is the byte offset into the field.

ImageFormat is either GIF or JPEG. If the FieldElemFormat is not entered, Domino assumes the image file format is GIF.

To see an example, try to import a GIF image into a RichText field and launch the preview in browser. Then look at the HTML source for that image; you should find something like this:

<IMG SRC="/database/view/document/FieldName/xx.yy
?OpenElement&FieldElemFormat=GIF>

Using OpenElement with OLE Objects
Syntax:

http://Host/Database/View/Document/FieldName/
FieldOffset/$OLEOBJINFO/FieldOffset/obj.ods?OpenElement

Note  The current URL syntax for referencing images and objects in Notes documents — specifically the FieldOffset — makes it impractical to create these URLs manually. As an alternative, you may paste the actual bitmap or object in place of the reference, create URL references to files stored in the file system, or attach the files to the documents.
Searching For Text With Domino Search URLs

The following commands allow you to search a Domino site or to search individual databases within a Domino site:

**SearchSite**

Syntax:

```
http://Host/Database/$SearchForm?SearchSite[ArgumentList]
```

Where $SearchForm and ArgumentList are optional arguments. The special identifier $SearchForm indicates that Domino will present a search site form for search input. If this identifier is provided, ArgumentList is ignored.

Examples:

```
http://www.mercury.com/mercsrch.nsf/$SearchForm?SearchSite
http://www.mercury.com/mercsrch.nsf/?SearchSite&Query=product+info+requests;1;;0;FALSE
```

**SearchView**

Syntax:

```
```

Where $SearchForm and ArgumentList are optional arguments.

The special identifier $SearchForm indicates that Domino will present a search view form for search input. If this identifier is provided, the ArgumentList is ignored.

Examples:

```
http://www.mercury.com/products.nsf/By+Product+Number/?SearchView&Query=PC156;3;;0;TRUE;
```

**Arguments (Optional)**

ArgumentList = Query; SearchOrder; SearchThesaurus; SearchMax; SearchWV

Where:

- **Query** = the search string
- **SearchOrder**=[1,2,3] default = 1.
  - 1 = “By Relevance”
  - 2 = “By Date Ascending”
  - 3 = “By Date Descending”
- **SearchThesaurus** = [TRUE, FALSE], default = FALSE
- **SearchMax** = [n], 0 default = 0 (meaning all)
- **SearchWord Variants** = [TRUE, FALSE], default = TRUE
  The ArgumentList must contain the Query argument; in addition, it may contain any or all of the other arguments.
- **OldSearchQuery**
  Repeats the last query.

---

**Part 2 Summary**

Most Web sites make use of frames and the effects are not only more captivating but also make the Web pages easier to use. Therefore, adding frames to Domino Web applications, is more of a necessity than an option. To accomplish these effects, special rules must be respected, otherwise frames will not work. The first section of Part 2 focused on frames and gave different possible solutions through different examples.

Part 2 also included a section on the possible ways to add links between Domino objects and to external objects. The last section explained the syntax of the URLs you must use to reference a specific Domino object with special instructions about how to access them using key words instead of Universal IDs.
Chapter 7
Domino Agents

This chapter describes Domino Agents; what they are, what they do, where you can use agents, and how to create them. The chapter also describes the special Web agents, WebQueryOpen and WebQuerySave, and how to enable agents for Web users.

In this chapter you will also learn how to write agents using LotusScript, how to access CGI variables, and the difference between restricted and unrestricted agents.

The chapter also includes examples of how to use profile documents with agents using LotusScript.

What is an Agent?

Agents allow you to automate many tasks within Notes. They operate in the background to perform routine tasks automatically for the user, for example, filing documents, sending mail, looking for particular topics, and archiving older documents.

There are several ways to activate an agent: by a schedule, when new mail arrives, or if a document has been pasted. The user can also activate an agent by clicking on a button or hotspot or selecting a form or view action. These buttons, hotspots and form and view actions use the @Command([ToolsRunMacro]) or the OpenAgent URL command to run agents.

Agents are more commonly used in Notes for the Notes client user rather than the Web browser user. From a Notes client you can do various things with agents. In this chapter we will concentrate on agents which can be activated by a user using a Web browser.

As an application developer you will most likely create two sets of agents in order to perform same operations from a Notes client and from a Web browser. The most important reason for this is the difference in the way an application interacts with the user in the two environments. In Notes, the applications can interact with users using message boxes or by prompting information, for example, to change the values in the fields of the currently open document. On the Web the only way to show information to users,
without using JavaScript, is by using HTML to create Web pages. If you want to change the current document on the Web, you can only do it before the document is loaded, using the WebQueryOpen event, or before it is saved, using the WebQuerySave event. These agents are covered later in this chapter.

Agents for Web users are most often written using LotusScript or Java, since simple actions are not available on the Web and @formulas do not allow you to return information to users.

**Creating an Agent**

To create an agent:

1. From the standard navigator select Agents.
2. Choose Create - Agent.
3. A new agent is created and the Agent Builder window will be displayed. It looks like this:

![Agent Builder Window](image)

**Naming the Agent**

The first thing to do is to give the agent a name. A descriptive name is especially important if the agent will be selected from the Actions pull-down menu by the user. Also, try to keep the first character unique. This is because, as with forms and views, Notes will use the first unique character as an accelerator key under OS/2, Windows, and UNIX (Macintosh does not use accelerator keys).

Also, click Shared Agent if you want this agent to be used by other users.

**Caution** Once you have saved an agent you cannot change a shared agent to a private agent or vice versa.
Allowing the Web User to Run Agents
When agents run, they automatically check the identity of a Notes user against any Server document or ACL restrictions. Manually run agents run with the identity of the Notes user; scheduled agents run with the identity of the person who created or last modified the agent. Since Web users don’t have a Notes ID that identifies them you have to allow Web users to run agents:

1. Choose Agent - Agent Properties.
2. Click the Design tab.
3. Select For Web Access: Run Agent as Web user.

Scheduling the Agent
You have several options to define when the agent should run.

1. First select when the agent will run. The following list of options is available:
   - Manually from Actions Menu. This option is not applicable for Web users, because Web users don’t have the Action menu.
   - Manually from Agent List. If you select this option, Domino encloses the name of the agent in brackets.
   - If new mail has arrived.
   - If documents have been created or modified.
   - If documents have been pasted. This option is not applicable on the Web since users can’t paste documents on the Web.
   - Scheduled Hourly/Daily/Weekly/Monthly/Never.
2. Select On Schedule Hourly from the pull-down menu and click the Schedule button to schedule the specified run time. The following dialog box is displayed:

   ![Schedule Dialog Box]

Chapter 7: Domino Agents  203
The example shows an hourly scheduled macro. You can specify for the agent to run every 30 minutes, hour, two, four, or eight hours.

3. Specify the start and end time each day.

4. Specify the start and end date for the agent to run, and whether or not to run on weekends.

5. Specify the server on which the agent is to run. This is only applicable for databases which are replicated across a number of servers. If the agent is modifying data in the database, it should run only once on one server. Then the changed data is replicated to the other replicas of the database.

   **Note** You can also type an asterisk (*) to indicate that the agent runs on any server. Choosing this may result in replication conflicts if several servers run the same agent and change documents.

**Selecting Documents to Be Processed**
This selection defaults intelligently depending on the option selected for scheduling the agent. For example, if the agent is scheduled to run if new mail has arrived, this option is set to Newly Received Mail Documents, and it cannot be changed.

   **Note** All other selection options are supported on the Web, except the Select documents option since selecting documents is not possible on the Web.

**Specifying What the Agent Should Do**
There are four ways of specifying what the agent should do:

**Simple Actions**
Simple actions are pre-defined actions which perform a sequence of actions without requiring any programming.

   **Note** Simple actions are not supported on the Web.

**Formulas**
Formulas can use the full range of @Functions available for the Web.

You cannot use most @commands in Web applications, since they are based on the Notes workstation user interface.

**LotusScript**
Agents can also be written in LotusScript. This is the most useful way, when launching agents through the Web.

1. In the Agent Builder window, click on the Script option button.
2. Start entering a LotusScript program in the programming pane.

Control is always passed to the agent using the Initialize event, so this is where the program should begin.
If you want to declare global variables or objects you can declare them in the (Declarations) event. Declared variables or objects are then available for all events.

**Java Agents**
Agents can also be written in Java. To attach a Java program to an agent, write the program in a Java development environment. Then, in Notes, click Java and then click Import Class Files to import the files into the agent. See Chapter 10: Java Applets, Java Agents, and JavaScript for more information.

**Agent Properties**
Agents have only a few properties:
1. From the standard navigator select Agents.
2. Select one of the agents.
3. Select Properties, an InfoBox is displayed:

4. You can choose to hide the agent from Web browsers or from Notes R4.6 or later clients.
5. In the For Web Access section selecting Run Agent as Web user causes the agent to be run with the Web user’s privilege. Otherwise the agent is run with the privilege of the user who has signed the agent.

**Note** The user who signs the agent has to have the authority to run agents on that server.
WebQueryOpen and WebQuerySave Agents

There are two special events in all Domino Forms: WebQueryOpen and WebQuerySave.

WebQueryOpen Event
A WebQueryOpen event runs the agent before Domino converts a document to HTML and sends it to the browser. Domino ignores any output produced by the agent in this context.

Examples for using this agent include performing large computations that aren’t possible with @commands or collecting statistics about who opened documents and when.

WebQuerySave Event
A WebQuerySave event runs the agent after field validations and before Domino saves the document in the database. The agent can run any operations with the document’s data or can modify this document.

An example of a WebQuerySave agent could be an agent which creates another document in the Notes database but doesn’t save the current document.

To perform error checking, field validation, and other processing before Web users open or save documents, create a shared agent that runs manually. Then write a formula that uses @Command([ToolsRunMacro]) to run the agent and attach it to the WebQueryOpen or WebQuerySave form events. This simulates the LotusScript QueryOpen and QuerySave form events that aren’t supported on the Web.

Restricted and Unrestricted Agents

In the server document of the public address book you can determine who can run unrestricted and/or restricted agents. Using unrestricted agents, users have full access to the server’s system. By using certain LotusScript methods, properties and classes in an agent, you can determine that the agent can run only if the user has the authority to run unrestricted agents. However, be aware that this does not apply on the Web. Web users can run any agent as long as the agent is not hidden from Web users.

Using LotusScript Agent to Capture CGI Variables

CGI, Common Gateway Interface, is a standard for interfacing external applications with HTTP servers. When a Web user saves a document or opens an existing document, the Domino Web server uses CGI variables to collect information about the user, including the user’s name, the browser, and the user’s Internet Protocol (IP) address.
You can use the `DocumentContext` property of the `NotesSession` class to capture CGI variables. The property returns a Notes document that contains all CGI variables that are applicable to the session. You can use these values to collect or process information for the current session.

The following example demonstrates how to access CGI variables:

```livescript
Dim session As New NotesSession
Dim doc As NotesDocument
Dim CGIValue As String

To create an instance of the `DocumentContext`:

Set doc = session.DocumentContext

Once you can handle this object, you can access every CGI variable and store it in a LotusScript variable. For example:

```livescript
CGIValue = doc.HTTP_USER_AGENT
```

The `CGIValue` string now has information about the user’s browser.

Examples:

- To find out the Web identity use:
  ```livescript
  Set webUserName=docContext.remote_user(0)
  ```

- To read the arguments passed, that is, the string “&............” that ends some URLs, use:
  ```livescript
  Set args=docContext.query_string(0)
  ```

---

**Examples of Agents**

In this section we will look at some examples of how to use agents on the Web.

**Example 1: Using Profile Documents with Agents**

In the Discussion database, many agents are related to two profile documents: Interest Profile and Archive Profile. Profile documents can be used to store data that is user dependent. For example, using the Interest Profile you can choose to be informed if a document is created in selected categories. We will now take a look at how the Interest Profile form works.

**WebInterestProfileEdit Agent**

WebInterestProfileEdit agent is an example of how you can open a profile document. This agent is run from a view when the user clicks the Edit Profile hotspot.
Sub Initialize
    Dim item As notesitem
    InstantiateObjects
    dbpath = getdbpath
    
    'Find the authenticated web user name
    Set webuser = New notesname(note.user(0))
    Set view = db.GetView("($Profiles)")
    The agent finds the profile document by key, for example, Interest Profile Joe User.
    
    'Create the appropriate key for finding the correct Interest profile
    key = "Interest Profile" & webuser.common
    Set profile = view.GetDocumentByKey(key)
    If profile Is Nothing Then
        If the agent doesn’t find an Interest Profile, it creates a new one.
        
        'If an Interest Profile hasn’t been created already, display it to the user
        Print "/"+dbpath+"/InterestProfile?OpenForm"
        Exit Sub
    End If
    
    If the Interest Profile is found, the agent opens it using the Print statement.
    
    'If an interest profile has been found for the user, present it to them
    Print "/"+dbpath+"/($All)/Lcase(profile.universalid)+?EditDocument"
End Sub

208 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
WebInterestProfileOpen Agent
The Interest Profile form includes both WebQueryOpen and WebQuerySave agents. The WebQueryOpen agent is called WebInterestProfileOpen. This agent resets the WebProfileCategories field and WebProfileStrings field and loads values from the ProfileCategories field and the ProfileStrings field, so that the values don’t get duplicated.

On Error Goto StandardError
  InstantiateObjects
  'remove all of these so they can reset with default values each
time the doc is opened
  If note.isnewdoc Then Exit Sub
  note.WebProfileCategories = note.ProfileCategories
  note.WebProfileStrings = note.ProfileStrings
  Exit Sub
  StandardError: 'This is for errors which we are not
  specifically handling
  Print "<h2>" & GetString(22) & "</h2>"
  Exit Sub

WebInterestProfileSave Agent
The WebQuerySave agent is called WebInterestProfileSave. This agent sets the document readers and authors.

  InstantiateObjects
This agent is only for Web users.
  IsWebUser = True
Set webuser to hold user’s name.
  Set webuser = New notesname(note.personname(0))
This is only done for a new interest profile.
  If note.isnewnote Then
    note.Form = "Interest Profile"
The agent creates a field in the interest profile document. This field is of the type, Authors. The field contains the user's name and this user is then the only one who can edit the document.
    Set item = New NotesItem(note, "PersonName",
webuser.common, AUTHORS)
Set the subject to the document.

```vbnet
note.Subject = "Interest Profile for " & webuser.common

note.ProtectFromArchive = 1

ReaderNames = webuser.common+": LocalDomainServers"
```

The agent creates a field in the interest profile document. This field is of the type, Readers. The field contains the users name and the names of the Local Domain Servers. After this, the user is the only one who can read the document. The names of the servers have to be in the Readers field, otherwise the document can't be replicated to other servers.

```vbnet
Set item = New NotesItem(note, "Readers", ReaderNames, READERS)

item.IsSummary = True

End If
```

**Example 2: Deleting a Profile Document Using an Agent**

This example is from the Discussion template, WebDelete. The agent deletes the profile document and returns an HTML page to the user.

```vbnet
Sub Initialize
  InstantiateObjects
dbpath = getdbpath

InstantiateObjects is a function call to sub in ScriptLibrary. InstantiateObjects sub looks like this:

Sub InstantiateObjects
  Set s = New NotesSession
  Set db = s.CurrentDatabase

  The DocumentContext property returns the values of the CGI variables supported by Domino. The CGI variables appear as items in the NotesDocument object (note in this case) returned by the DocumentContext property.

  Set note = s.documentcontext

End Sub
```
This is again the Initialize event of WebDelete agent. The following sentences are for creating HTML links to database views.

'------------------------------------------------------------
TopicView = "<hr><b> <a href="/\$All?OpenView">"+GetString(2)+'</a></b>"

CategoryView = " | <b><a href="/By+Category?OpenView">"+GetString(3)+'</a></b> | "

AuthorView = "<b><a href="/By+Author?OpenView">"+GetString(4)+'</a></b> | "

ArchivingView ="<b><a href="/Archiving?OpenView">"+GetString(5)+'</a></b> <hr>"

'------------------------------------------------------------

Getting the document with GetDocumentByUnid method. The DocumentID is captured using the Query_String CGI variable.

'Get the document

   OriginalUNID = Mid(note.Query_String(0),
   Instr(note.Query_String(0), ";")+1, 32)
   Set originalNote = db.getdocumentbyunid(originalunid)
   Subject = originalNote.Subject(0)

The document is deleted.

'remove the document

   originalNote.Remove(True)

An HTML page is returned to the Web browser using the Print statement.

'tell the user

   Print "<b>"+GetString(11)+topicview+categoryview+
   authorview+favorites+archivingview

End Sub

It is also possible to delete a common document, for example, a discussion main topic document, by using @Command([EditClear]). By using this @Command, Domino returns an HTML page with text “Deleted.” You would probably want to customize this message. You can do this by using an agent, as shown above, or by creating a $$ReturnDocumentDeleted form. See Chapter 3: Designing Application Forms, for more information.
Agents allow you to automate many tasks within Notes. They operate in the background to perform routine tasks automatically for the user, for example, filing documents, sending mail, looking for particular topics, and archiving older documents.

On the Web you can use agents to perform operations before the document is opened and before the document is saved. You can also create agents for using profile documents. Using LotusScript you can also access CGI variables to capture information about the user.
Chapter 8
Domino Web Searching

This chapter will cover setting up Web searching with Domino. Topics covered are:
- Search related URLs.
- Full text indexing of databases.
- Customizing search forms.
- Customizing search results.
- Setting up a site search database.

Adding Search Capabilities to Your Web Site

As the number of databases making up a Web site increases, and the amount of data grows, you will want to add search capabilities to the site. Enabling your users to search for text throughout a Web site or within a view helps them to quickly and easily find the information they are looking for.

Domino provides a full-text search engine that acts on text within documents in a Notes database as well as searching ASCII text in file attachments within those Notes documents. However, the search engine does not work on HTML files or other file types.

You can set up your site to allow text searching within a single view or in multiple databases. Creating an index for multiple databases, requires creating a search site database on a server. The person creating the search site database specifies a search scope that includes all the databases to be included in the desired index plus the type of information to be included. Each database in the search scope must have the “Include in multi database indexing” design property enabled.

You include buttons or links within your views or forms, to open a search form when users click on them. The users fill in the search criteria on the form and Domino returns a list of links to the documents that match the criteria.
Search Related URLs

You can use the @Command([ViewShowSearchBar]) in the formula for your buttons or links to bring up the search form, or you can define a URL to display the search form. The search form can be either a customized search form or the default search form. Search related URLs are available for both view searches and search site searches.

Search View URLs

You use search view URLs when you want to limit a search to documents displayed in one database view.

The search view URL enables you to control the search results. If you want users to be able to search on all documents in a database your search view URL points to a view displaying all documents, or you can limit the search, for example, to a view that shows only documents whose status is Completed.

To display a search form for user-defined searches, include the $SearchForm object in the URL:

http://<site>/<database>/<view>/$SearchForm?SearchView

The ?SearchView action can also be used with arguments if you don’t want users to enter their own search arguments:

http://<site>/<database>/<view>?SearchView&<arguments>

Search Site URLs

You use search site URLs for text searches in multiple databases. The URL requires the name of a search site database:

To display a search form for user-defined searches, include the $SearchForm object in the URL:

http://<site>/<search site database>/$SearchForm?SearchSite

The ?Searchsite action can also be used with arguments:

http://<site>/<search site database>?Searchsite&<arguments>
Full Text Indexing

It is a prerequisite for Web searching with Domino that the databases you want to act on are full text indexed. Use the Full Text tab in the database properties box to create a full text index on a database.

A full text index is a collection of files that lets a user search one or more databases for information. After creating a full text index, a user can enter a word or phrase and locate all the documents containing that word or phrase.

To create an index for a single database stored on a server, you must have Manager or Designer access. You can index any database stored on your workstation without regard to access levels.

The full text index for a search site database lets a user search all the databases in the search scope. Creating and maintaining a search site database can greatly affect system resources and performance. The system administrator should administer search site databases. As a database manager, you should know if any of your databases are associated with a search site database.

Security Issues
If the default option Index Encrypted Fields is selected when the index is created, the following three things happen:

- Any Notes user with access to the database can search for phrases within the encrypted fields without the encryption key. For example, the Employee form in the Personnel database contains the encrypted field Salary. With the full text index, any user can search on "50,000" and documents containing that figure are returned; however, the user cannot read the contents of the field without the encryption key.
- The full text index file is unencrypted plain text, and anyone with access to the server can read the file. Encrypted text may now be unencrypted.
• The encryption key is part of the server ID, and is active for all databases on that server. If you index a different database and do not deselect Index Encrypted Fields, any fields using that encryption key are compromised.

Location of a Full Text Index
When you create a full text index, Notes creates a subdirectory in which to store the index files. Notes names this subdirectory using the name of the indexed database and the file extension .FT. When indexing multiple databases, the search site database is the indexed database.

For example, if you index a database named LOG.NSF, Notes creates the subdirectory LOG.FT. Notes places the subdirectory in the same directory as the database, usually the Notes data directory.

Size of a Full Text Index
The size of a full text index depends on two factors:

• The ratio of text to non-text elements (such as bitmaps, buttons, agents) in the indexed database(s). The percent of text typically ranges from 25% to 75%. If you choose to index attachments (including OLE objects) or encrypted fields, the text associated with them increases the percentage of text available for full text indexing.

• The index options chosen. The Word Breaks Only option creates a full text index that is about 50% of the space used by text in the database. The Word, Sentence, and Paragraph Breaks option creates an index that is approximately 75% of the space used by text.

Full text indexes take up a significant amount of disk space and you need to be sure adequate system resources are available to store an index.

Full Text Index Updates
Since the contents of most databases change over time, full text indexes associated with them need to be periodically updated to keep in sync with the changes. You must manually update indexes on databases stored on your workstation. Notes automatically updates indexes on server databases according to frequencies selected by the database manager for individual databases or the system administrator for a search site database.

Note Documents added to the database since the database’s full text index was last updated will not be returned by the search.
Customizing Search and Result Forms

You can customize searching at your site by adding your own design touches to search-input and search-result forms. A customized search form must be named $\$Search — either as actual name or alias. When a user clicks on a link to open up a search form, Domino looks in the current database or in a search site database for a form of that name. If the form exists, Domino opens it. If it doesn’t find it, Domino uses the default search.htm file found in the icons directory on the Domino server.

You can customize either the search.htm file, create a new search form of your own or copy and paste the search forms provided in the Search Site database. There are two Web search forms in the Search Site database:

- **Web Search Simple.** This is used for simple searches. It includes a link to the Web Search Advanced form.
- **Web Search Advanced.** This form allows users to create more advanced searches.

The forms are designed to work in a Search Site database. To use the forms in a database that allows view level search you must change the formula in the $\$\text{Return}$ field.

1. Copy the Web Search Simple form to the database where you want to use it.
2. Open the form in design mode.
3. Go to the $\$\text{Return}$ field. The beginning of the original formula looks like this:
   \[
   \text{DBName} := \text{@Subset (@DbName; -1);}
   \]
   \[
   "\[/" + \text{DBName} + "?\text{SearchSite}\&\text{Query}=\" + \text{Query} + ";\text{SearchOrder}=\"
   \]
4. You need to change the ?SearchSite to ?SearchView and add a view name after the DBName. The start of the new formula should look like this:
   \[
   \text{DBName} := \text{@Subset (@DbName; -1);}
   \]
   \[
   "\[/" + \text{DBName} + "/\text{ViewName}/?\text{SearchView}\&\text{Query}=\" + \text{Query} + ";\text{SearchOrder}=\"
   \]
   where ViewName is the name of a view in your database. The remainder of the formula should remain the same. Since you are creating a URL, the view name must start and end with a forward slash (/), and spaces in the view name must be replaced with plusses (+).

Make the same changes to the Web Search Advanced form if you want to include the advanced search in your database.
Creating a Customized Point & Click Search Form

In the following example we will be creating a simple-to-use customized search form. The form allows users to create ad-hoc queries by simply pointing and clicking. Based on user selections, the form generates the search view URL and returns a list of documents found. Users could create the same queries by using the default search form and enter a query, like the one shown below, in the full text search field:

\[(\text{FIELD } \text{AssignedTo}=\text{Alice Duffy} \text{ OR FIELD } \text{AssignedTo}=\text{Bill Smith} \text{ OR FIELD } \text{AssignedTo}=\text{Eileen Sawyers}) \text{ AND (FIELD } \text{Status}=\text{Open} \text{) AND (FIELD } \text{DueDate}=01/01/96)\]

To avoid forcing users to learn the syntax for creating complex queries like the one shown above, you can create a simple-to-use interface for building complex queries.

We will use an action items database as an example. Each action item tracks the assigner, assignee, the date assigned, due date, status of the action, subject, etc. We want to enable users to search for action items based on assignee, the status of the item (open, completed or approved), and due date.
1. Start by creating a new form and give it the alias, $$Search.

2. Create the following non-hidden fields. All fields are keyword fields, except for the QueryDate field.
   - **AnyAll** is an editable Keyword field with the following values:
     - Any | 0
     - All | 1
   - **AssignedTo** is an editable Keyword field that uses a @DbColumn formula to populate the drop-down list:
     - @DbColumn ("Notes": "NoCache"; @DbName; "AssigneeItems"; 1)
     - "Allow multi-values" is selected and the HTML Attribute is set to "SIZE=3"
   - **Status** is an editable Keyword field with the values:
     - Open
     - Completed
     - Approved
   - **BeforeAfter** is an editable Keyword field with the values:
     - on|0
     - after|1
     - before|-1
   - The QueryDate field is an editable Time field showing only the Date.
3. Create the following hidden fields:

- The Query field is computed. It is used to generate query arguments. Enter the following formula for the field:

  ```
  sep:=@If(AnyAll="1";" AND ";AnyAll="0";"OR";"");
  dsep:=@If(BeforeAfter="0";" = ";BeforeAfter="1";" >";
                 BeforeAfter="-1";" < ";"");
  wdate:="FIELD DueDate";
  assign:="FIELD AssignedTo="+AssignedTo;
  state:="FIELD Status="+Status;
  dstring:=@If(QueryDate="";wdate+dsep+@Text(QueryDate);"
        @If(AssignedTo="";"("+@Implode(assign;"OR")")
        @If(AssignedTo="" & Status=""; sep + "("+
            state")")
        @If(AssignedTo="" & Status=""; "("+
            state")")
        @If(dstring="";"AND"+"("+dstring+")")
  
  The sep variable is set to the literal string “AND” or “OR” depending on whether a user wants documents containing all or any of the search criteria. In the @If statement we insert the sep value if both AssignedTo and Status are used for the search.

  If a user selects more than one assignee on the search form, the variable assign becomes a list of several values: “FIELD AssignedTo=Henry Moore” and “FIELD AssignedTo=Alice Walker”, etc. We use the @Implode function in the @If statement to turn assign into a string separated by the literal string “OR”.

  If the QueryDate field is not blank we prepend it by the literal string “AND”.

- We are using the $$Return field to generate our search URL with the search arguments, and have the browser immediately return the result of our query. To return a URL, the $$Return field must specify a URL in square brackets.

  ```
  DBName:=@Subset(@DbName;-1);
  
  "[[/"+DBName+"/ByAssignee/?SearchView&Query="+Query+"]]
  
  220  Developing Web Applications Using Lotus Notes Designer for Domino 4.6"
When a user enters data in a form, like our search form, Domino normally acknowledges the submittal. To overrule the submittal functionality of the $$Return field and go directly to the results page, we enclose the URL in double square brackets.

- Normally, when a Web user submits a form, Domino will create a document in the database. In our case we do not want to create documents with the query form. To prevent this from happening, we use a special Notes field called SaveOptions. If the field value for the SaveOptions field is “0” Domino will not save the document.

We have now accomplished what we wanted. Our users can now construct complex queries in the database by simply pointing and clicking, without having to worry about the correct syntax. All that is left to do is to add a link from our Web site to open the customized search form using the URL:

```
</<database>/<view>/$$Search>
```

### Customizing Search Result Forms

To customize the Search Results page, create a form and assign it one of the form names below:

- **$$SearchTemplate for <viewname>**
  Associates the form with a specific view. The form name includes `<viewname>`, which is the alias for the view, or, if no alias exists, the name of the view.

- **$$SearchTemplateDefault**
  This form is used as the default search result form for all Web searches that aren’t associated with a specific form.
Both forms require a $$ViewBody field on the form. The value of the field is ignored by Domino, but the field is mandatory on a search result form.

**Note** The following features in the default Search results form provided with Domino cannot be built into a customized Search results page:
- A count of the number of documents found.
- The ability to restate the search query.
- A search bar that allows users to reset search results.

## Search Site Databases

You can create search site databases that enable users to search multiple databases for information. To set up a search site database, you should configure a search scope which defines the databases and the information in them that users can search. The databases included in the search scope can span more than one server and more than one domain, but must have the property Include in Multi Database Indexing selected.

You then create a full text index that indexes all the databases in the configured search scope. You don’t need to create full text indexes for individual databases included in the search scope unless you want users to be able to perform view searches as well.

You can create more than one search site database, each indexing a group of related databases. For example, you can create a search site database for marketing databases, another for sales databases, and another for customer service databases.

To search the databases for information, users fill out a search form in the search site database and specify search criteria. Domino returns each document found in a search results form, as in the case of single database view searches.

### Creating a Search Site Database

2. Select server location.
3. Enter a title and file name for the search site database.
4. Select Show Advanced Templates.
5. Select Search Site (SRCHSITE.NTF) as the template.
6. Click OK.

**Note** The default access of the database created is Author. There is a role named [SearchSiteAdmin] in the access control list, which is associated with
the Search Scope Configuration form and all of the views except the Private Searches view. Database managers should assign this role to anyone who is authorized to create configuration documents in the database.

**Defining a Scope**
Create Search Scope Configuration documents:

- To specify a domain as a search scope, click Domain and enter the name of the Domain.
- To specify a server as a search scope, click Server and enter the name of the server.
- To specify a directory on a server as a search scope, click Directory then enter the name of a server and a directory on the server relative to the Notes data directory.
- To specify a specific database as a search scope, click Database and enter the name of the server and the file name for the database.

**Refining a Search Scope**
When you modify a database entry to refine a search scope, for example, change the full text index option to No Index to exclude the database from searching, Notes creates a search scope configuration document for the database that reflects the change.

A search scope may be refined automatically according to which databases have the design property Include in multi-database indexing. For example, if this design property is selected for only ten databases on a server, you can create a search scope configuration with the server as a scope but only those ten databases are indexed.

**Caution** When you change configuration options, you have to delete the full text index on the search site database and recreate it for the changes to take effect.

**Database Views**
You use any of the following views to see the databases included in a search scope:

- All By Server.
- Databases By Category.
- Databases By Replica ID.
- Databases By Server.
- Databases By Title.
Selecting an Indexing Option

The index option you select affects the size of the full text index for the search site database.

- To exclude databases for the selected scope from a search, click No Index. This option excludes a database or databases and therefore adds nothing to the index size.
- To index only summary data that appears in views and not rich text or attachments, click Index Summary Data (No RTF). If a search scope is wide-ranging, use this option as a way to limit the size of the full text index.
- To index only summary data and rich text but not attachments, click Index Full Document.
- To index summary data, rich text, and attachments, click Index Full Document and Attachments. This indexing option increases the index size the most. Notes indexes only the ASCII text in attachments and OLE objects. If an attachment or OLE object is compressed, it is not indexed.

Multi-Database Full Text Indexes

A full text index created for a search site database is just like a full text index created for a single database except that it indexes multiple databases. The index is stored in a subdirectory to the Notes data directory that stores the search site database. The subdirectory name is SEARCHSITE.FT, where SEARCHSITE represents the name of the search site database without the .NSF extension.

You should plan carefully which databases and data to include in a search scope and make sure the server that stores the search site database has adequate space. Including many databases in a search scope may require a server which stores only the search site database. You can help keep the index size to a minimum by selecting the index options Word Breaks Only and Exclude words in stop word file, when you create the index and by selecting the option Index Summary Data when you configure a search scope, which prevents rich text from being indexed.

Because updating the full text index of a search site database can take some time, you should select Daily as an update frequency so that the index updates once a day during off-hours.
How Users Search Using a Search Site Database

Incorporate SiteSearch URLs in your site by adding them where you want them. The search site database contains two Web forms you can use as is or modify:

1. Web Search Simple.
2. Web Search Advanced.

The search criteria the forms use is similar to the criteria used for a single-database search. You can search for a specific word or phrase or perform advanced searches which allow searches for multiple words, word variations, synonyms defined by the thesaurus, and documents by date and category. The search results can also be sorted.

Domino returns each document found in a Search Results document. Each document entry includes a document link and summary information about the document from the default view of the database the document comes from.

Caution All people and groups in the ACL for the databases included in a search scope must have at least Reader Access (including the OtherDomainServers group). If not, the document entry in the search results form will show only the document link as the database name, it will not display any summary information about the document. If you are using ReaderNames fields in a database, those documents will not appear in a search.

Note If the property Show in Open Database dialog has not been selected for a database in the search scope, then a Search Results document for that database shows a document link but no text.

Customizing Search Site Result Forms

To customize the Search Results page in a search site database, create a form and assign it the following name:

- **$SearchSiteTemplate**
  
  This form is used as the default search result form for all Web searches in the search site database.

The form requires a $ViewBody field. The value of the field is ignored by Domino, but the field is mandatory.
Summary

Domino provides a powerful full-text search engine that acts on text within documents in Notes databases. With Domino you can combine text searching within a single view or in multiple databases.

To enable view level searching, databases must be full text indexed. The search site database indexes all databases in the search scope. A database included in the search scope does not need to be full text indexed itself, unless you want to allow view level searching on the database as well.

Search and result forms are fully customizable, enabling you to utilize all the forms design features such as graphics, buttons, tables, navigators, etc. when you design search facilities for your Web site.
Chapter 9
Domino Security

This chapter will introduce Domino security from the viewpoint of Web application development and deployment. We will focus on the following topics:

- Overview of the Domino security structure.
- Administering Web users’ access.
- Special programming considerations for Web applications.
- The SSL implementation within Domino.

Domino Security Structure

Domino gives you many ways and tools to help you secure your server and applications. Domino uses powerful encryption technology and replication to offer one of the most secure computing environments available today. Domino security covers multiple levels, these levels are complementary; using one level reinforces the security above and below it. These levels include:

- Internet security
- Network security
- Hierarchical naming for servers and users
- Domino server security
- Notes client security
- Web browser client security
- Database security
- Database design element security
- Mail security

Within this chapter we will concentrate on those areas that specifically affect the Web application environment. For more detail on Domino security in general, please refer to ITSO redbook, The Domino Defense: Security in Lotus Notes and the Internet (SG24-4848).
When you set up a Domino environment with communications over the Internet, there are many aspects of security to consider. The table below lists some of the security features Domino provides to secure its applications over the Web.

<table>
<thead>
<tr>
<th>Types and levels of security</th>
<th>Domino Internet/intranet feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>User identification</td>
<td>Internet/intranet users are registered through Person documents in the Public Address Book.</td>
</tr>
<tr>
<td>Authentication</td>
<td>Basic password authentication occurs when users supply a name and password and Domino checks the name and password in the Public Address Book. If you enable SSL, client authentication occurs when clients and servers exchange certificates.</td>
</tr>
<tr>
<td>Database access control</td>
<td>Access control list. For TCP/IP connections, an additional Advanced setting “Maximum Internet name &amp; password access.”</td>
</tr>
<tr>
<td>Secure Sockets Layer (SSL)</td>
<td>An SSL-enabled server offers client and server authentication through public/private key exchange. The SSL option adds greater security by encrypting transactions to and from the server.</td>
</tr>
<tr>
<td>Encryption</td>
<td>SSL-encrypted transactions only.</td>
</tr>
<tr>
<td>Server access lists</td>
<td>Not available.</td>
</tr>
<tr>
<td>ACL control in databases reached through directory link (.DIR) files</td>
<td>ACLs work in databases reached with directory links. You can control whether browser users can access directory links using a NOTES.INI setting.</td>
</tr>
<tr>
<td>Electronic signatures</td>
<td>Not available.</td>
</tr>
</tbody>
</table>

### Security for Web Access

#### Registered and Unregistered Users

There are two types of Domino Web users, registered users and unregistered users. A registered user has a person document in the Public Address Book with his/her name and HTTP password. An unregistered user does not have a person document in the Public Address Book and is given the user name Anonymous. No password is associated with the Anonymous ID.
Authentication

The method Domino uses to authenticate the user depends on whether the user accesses the server using TCP/IP or Secure Sockets Layer (SSL). Authentication of Web users via TCP/IP protocol is done by a technique known as basic authentication.

If you set up basic password authentication for Internet/intranet users, they are authenticated only when they attempt to do something for which access is restricted. For example, when users try to open a database whose default access is No Access, they are challenged by the server to supply a valid user name and password.

Authentication succeeds if the user name and password supplied by the user matches the appropriate fields in the Person document of the Public Address Book on the Domino server and if the user is listed individually or as part of a group in the database access control list (ACL).

The following is an example of how a client (ClientA) connects to a server (ServerB) using the TCP/IP protocol.

1. ClientA tries to access a database on ServerB.

2. If the server has basic password authentication and anonymous access enabled and the database allows access to anonymous users in the database access control list (ACL), ServerB gives ClientA access using the access specified for anonymous users in the database ACL.

3. If the server has basic password authentication and anonymous access enabled and the database does not allow anonymous access in the database ACL or if the server has only basic password authentication enabled, ServerB requests ClientA’s name and password. Andrew sends his name and password to ServerB. ServerB checks the name and password in ClientA’s Person document in the Public Address Book. If the name and password do not match, ClientA cannot be verified.

   **Note** After a user name and password is entered, the Web browser remembers the information and re-sends it automatically in response to Domino’s requests. In effect, the Web session holds the user name and appropriate access throughout the session.

4. If the server has anonymous access enabled, but not basic password authentication, ClientA connects anonymously to the server using the access specified for anonymous users in the database ACL.

After authentication is done, the user’s session holds the user name as the CGI. After a user name and password is entered, the Web browser remembers the information and re-sends it automatically in response to Domino’s requests. In effect, the Web session holds the user name and appropriate access throughout the session environment variable.
REMOTE_USER. Domino interprets the REMOTE_USER variable as a user name within the Notes world, and Notes applications receive the interpreted user name. For example, the @Username function receives the Web user’s name from REMOTE_USER.

If the REMOTE_USER value is undefined (or null), in other words a user name has not yet been entered, Domino gives the special name Anonymous to the user session and Notes applications recognize the user as Anonymous.

**Note** Domino applications designed for the Web may handle the CGI REMOTE_USER variable directly. Domino applications that are designed for both Notes users and Web users can use @Username to receive the Web user’s name.

### Defining Web Users

You may register new Web users from scratch, or let existing Notes users have access to the Domino Web server.

**New Users**

To register the user from scratch, follow these steps:

1. Create a new Person document.
2. Enter the required information. The User name and HTTP password fields are the only fields used for Domino Web user authentication. The other fields are optional with the exception of the Last name field. Even though this field is not used in Domino authentication, the form won’t be saved without it.
3. Save and close the document.

**Existing Notes Users**

To give an existing Notes user Web access:

1. Open the Person document of the target user in edit mode.
2. Put a password into the HTTP Password field.
3. Save and close the document.

**Caution** Make sure you are aware of the implications of giving Notes users a Web password. You may be inadvertently increasing your security exposure. The authentication offered by the HTTP user ID and password is not as secure as that of the Notes ID. For example, Notes users must have physical access to their notes.id file but a Web user’s password may be stolen, particularly if it is sent unencrypted over the network.
Registering Web Users

Using A Registration Application
With Domino R4.5 and R4.6, a Domino registration sample application program is included with the Domino server. The registration process is very simple, it asks users for a name, company name, phone number, and e-mail address. The form contains a randomly generated password, which users can change at registration time. When the user submits the form, the password is encrypted, and the triggered LotusScript agents add or modify a person document in the Public Address Book on the Domino server. The application also takes advantage of Notes features such as input validation formulas, readers fields, and password encrypting.

The registration program has some built-in error checks. It uses an input validation formula to verify that the user’s e-mail address is valid when the form is submitted. Also, if the user name already exists in the Domino server’s Public Address Book, it asks the user to pick another user name.

Customizing the Registration Application
You can create your Web users’ registration program by copying some features from the sample program. If you intend to do this, remember the following points:

- The sample is designed to allow anyone with Web connectivity to your server to create a user name and password for themselves on your server. This will not allow them to use Notes to access your data, but it will allow them to use their Web browser through Domino to access, and possibly change data. Be very careful in deploying this application or modified versions of it as the application modifies your Public Address Book.

- Modify the server variables to specify how often an agent is run. The variables to modify are located in the NOTES.INI. The default is every half hour. If you want the agent to run more frequently change the following parameters, for example:

  AMgr_DocUpdateEventDelay=2
  Amgr_DocUpdateAgentMinInterval=5

- Use the Agent - Log menu item with the Handle Requests agent selected to see what happened the last time the agent ran.

- Add a special group for Web users, and add the group to the database ACL.

- Assign this group an appropriate access level, for example, Author.
Separate Web Users and Notes Users in Public Address Books

As a Webmaster, you may not want to place Internet public users in your domain public address book. For security and ease of management it may be better to place those users in a separate Name and Address book.

You can utilize Domino’s cascading Public Address books feature in your application. Cascading Public Address books is one of the multiple public address books features. It is set up by using the NOTES.INI setting NAMES= on servers, to point to secondary Public Address books. With this approach, users can use the addressing dialog to select names from secondary Public Address books.

Note  Searches through the secondary Public Address books are in the order they are listed in the NAMES= setting.

For example, create a secondary public address book named webusers.nsf from template pubnames.ntf, and modify your Notes.ini file as follows:

NAMES = names, webusers

Caution  Make sure your domain’s public address book continues to be called names.nsf and that it is the first listing in the names= line.

After setting up webusers.nsf, you can modify your registration program to associate webusers.nsf with Web users’ registration.
Password Considerations for Web Users

The password placed in the HTTP password field of the Person document is one-way encrypted (or hashed) before saving. Only the hashed form is stored, so you cannot extract a user’s password if it is forgotten, you can only replace it. You can see the hashed value by re-opening the Web user Person document.

Hashing is done by the @Password built-in function. The logic is not published but every Notes user can test the @Password function on their Notes client. Once you get someone’s hashed password, you can test whether your guess is correct or not by comparing the results of @Password.

For example, the word password results in:
355E98E7C7B59BD810ED845AD0FD2FC4 every time. This opens up the possibility of a dictionary attack, in which the attacker tries hashing a list of trivial passwords in an attempt to find one with a hashed value that matches the stored value.

Unfortunately, the hashed password is not fully protected. The design of the Public Address Book hides the field, so it does not appear in read mode. However, it is visible in edit mode.
Note Editor access to the Person document is not required to view the HTTP password field for any user; reader access is enough. If you can read the Person document, you are also able to open the Properties InfoBox and browse all fields, including the HTTP password field. Usually every Notes user has at least reader access to their domain’s Public Address Book, therefore, they can see any other user’s HTTP password.

How serious is this exposure?

- First, note that a Web user cannot use the Properties InfoBox and therefore cannot see the hashed password; only a Notes client user can see it.
- Secondly, the hashed field is only of value if you can guess what the password is. Users should be educated about the need for non-trivial passwords. If you are an administrator, it is also a good idea to attack your own system from time to time. Use a program to test for trivial passwords and reprimand users who do not follow the rules.

In summary, you should be aware that giving a Web password to an existing Notes user can possibly weaken security, unless you are careful in your choice of passwords.

---------------

Protecting the Domino Server

If you run a Domino server inside an enterprise network you do not usually have to think too much about how it may come under attack. By contrast, a Domino server providing access over the Internet is in a very exposed position.

There are several aspects to consider when protecting a server in this situation:

- How to place it within a firewall so that it gets the best possible protection, but can still do its job.
- How to harden the underlying operating system, so that it does not offer a target to a would-be hacker.
- How to set up the Notes Access Controls so that the server allows only the access we want, and does not expose information about Notes objects that we want to hide.
- How best to set up logging and monitoring.
Using Firewalls with Domino

A firewall is a system that is designed to control unauthorized access to a private network from the public Internet.

A proxy is a system that can understand the type of information transmitted, communicate on behalf of the requester, and communicate information back to the requester. A proxy can provide detailed logging information about the client requesting the information and the information that was transmitted. It can also cache information so you can quickly retrieve information again.

You can set up three types of firewall software and hardware configurations.

Packet-Filtering Firewall

A packet is a piece of information that is transmitted over the network, sometimes called a datagram. A packet-filtering firewall examines where the packet is going and what type of information it contains. It then checks whether your network allows delivery to that destination and allows that type of information to enter or exit your network. If the packet passes these tests, it is allowed to continue to its destination. Typically, packet filtering is implemented with router software.

You can use a packet filter to filter specific ports that determine which services are allowed through your firewall. Packet filters control Notes and Domino traffic at the network level. Below is a table that contains the service that a client requests and the default port used when making the request.

<table>
<thead>
<tr>
<th>Service</th>
<th>Default port number without SSL</th>
<th>Default port number using SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes and Domino Remote Procedure Calls (RPC)*</td>
<td>1352</td>
<td>n/a</td>
</tr>
<tr>
<td>HTTP (Web server and Server Web Navigator)</td>
<td>80</td>
<td>443</td>
</tr>
<tr>
<td>IMAP</td>
<td>143</td>
<td>993</td>
</tr>
<tr>
<td>LDAP</td>
<td>389</td>
<td>636</td>
</tr>
<tr>
<td>POP3</td>
<td>110</td>
<td>995</td>
</tr>
<tr>
<td>NNTP</td>
<td>119</td>
<td>563</td>
</tr>
</tbody>
</table>

Application-Proxy Firewall

An application-proxy firewall is a server program that understands the type of information that you are transmitting. Application proxies control the flow of information between internal and external clients and servers. An application-level proxy provides greater and more configurable security than a packet-filtering solution because it resides in the actual application and acts as an intermediate node communicating on behalf of the requester.
You can use an application-level proxy for communicating Notes and Domino Remote Procedure Calls (RPC) and HTTP format information. RPC is the architectural layer of Notes and Domino that lets you use Notes and Domino services, such as replication and Notes mail.

**Note** You cannot use application-level proxies for the other Internet protocols — IMAP, LDAP, POP3, and NNTP — unless the protocol is set up to use Secure Sockets Layer (SSL).

When setting up an application-level proxy, make sure the following Domain Name Services (DNS) are correctly configured:

- db.DOMAIN and db.ADDR, which are the databases DNS uses to map Host names to IP addresses, must contain the correct Host names and addresses.
- Host files must contain the fully qualified domain name.
- If you are using Network Information Service (NIS), you must use the fully qualified domain name and make sure NIS can coexist with DNS.

You can use the following types of application proxies:

- **Domino passthru**: Lets you use a passthru server to communicate using Notes and Domino RPCs.
- **HTTP proxy**: Lets you use an HTTP proxy server to communicate using the HTTP protocol, Internet protocols secured with SSL (for example, IMAP, LDAP, or NNTP), or Notes and Domino RPCs.

For more information refer to Lotus Notes Administration Help.

**Circuit-Level Proxy Firewall**

Circuit-level proxy servers (specifically SOCKS) work outside the application layers of the protocol. These servers allow clients to pass-thru a centralized service and connect to whatever TCP port the clients specify. SOCKS servers can authenticate the source address of connection requests and can block unauthorized clients from connecting to the Internet.

Domino has built-in support for SOCKS Version 4.2. (Most SOCKS Version 5 servers are backward-compatible with SOCKs Version 4.2 servers.) By leveraging this native SOCKS support, you can utilize centralized SOCKS services that may exist on the corporate network.

If desired, you can configure the communications between the local Notes workstations and Domino servers to bypass the SOCKS server so all local traffic does not need to pass-thru the SOCKS server. However, if passing through a SOCKS server is a requirement, you can configure your Notes workstations to do so by using TCP vendor stacks that support SOCKS transparently for all applications.
Note  The Domino SOCKS implementation does not include support for the Identification protocol. Support of this protocol is available from other sources and works in conjunction with the Domino SOCKS implementation. For more information on the Identification protocol, read the Internet Request for Comments (RFC) 1413 available at:
http://ds.internic.net/rfc/rfc1413.txt

Notes workstations and Domino servers need to know how to access the SOCKS server in order to use it as an circuit-level proxy. To set up a workstation or server to use the SOCKS server, you must specify the proxy information in the Location document for the workstation and in the Server document for the server.

Hide the Database Name List on the Server

Server access lists for Notes servers provide security, however, server access lists do not work for Web users. You can control access on each database, but the existence of a database on a server cannot be completely hidden from Web users.

To hide the database name list, do the following:

1. From the administration panel, click Servers and choose Servers View.
2. Open the Server document for this server, and then go to the Security section.
3. In the Allow HTTP Clients to Browse Databases field, select No.
4. Save the document.

When this field is set to No, Web users can’t see a list of databases, although they can still open individual databases for which they have access. Hiding the list of databases is useful if you have virtual servers on one machine or if some databases aren’t for Web use.

Database Access Control List

At the heart of every Domino database’s security lies the Access Control List, commonly referred to as the ACL. This list contains information on the users and groups that can have access to the database, and the level at which these people are allowed to access.

With Domino Release 4.6 there are seven main levels of access that a database administrator can assign to a person or group and eight sublevels below each of these.
<table>
<thead>
<tr>
<th>Access Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>Users with Manager access can modify ACL settings, encrypt a database for local security, modify replication settings, delete a database and perform tasks permitted by no other access level. Managers can also perform all tasks allowed by other access levels. Notes requires each database to have at least one Manager. It’s best to assign two people Manager access to a database in case one manager is absent.</td>
</tr>
<tr>
<td>Designer</td>
<td>Users with Designer access can modify all database design elements (fields, forms, views, public agents, the database icon, Using This Database document and About This Database document), can modify replication formulas, and can create a full text index. Designers can also perform all tasks allowed by lower access levels. Assign Designer access to the original designer of a database or to a user responsible for updating the design after a database is in use.</td>
</tr>
<tr>
<td>Editor</td>
<td>Users assigned Editor access can create documents and edit all documents, including those created by others. Assign Editor access, for example, to a user responsible for maintaining all data in a database.</td>
</tr>
<tr>
<td>Author</td>
<td>Users assigned Author access can create documents and edit documents they create. Assign Author access to allow users to contribute to a database but not edit documents created by others. When possible, use Author access rather than Editor access to reduce Replication or Save Conflicts.</td>
</tr>
<tr>
<td>Reader</td>
<td>Users assigned Reader access can read documents in a database but cannot create or edit documents. For example, assign Reader access to users who must be able to read the contents of a reference database such as a company policies database. Anyone with at least Reader access to a database can create personal agents in the database if the database manager selects the ACL option “Create personal agents.” However, users can only run agents that perform tasks allowed by their access levels. For example, someone with Reader access can create a private agent that deletes documents, but the agent won’t delete documents when the user runs it.</td>
</tr>
<tr>
<td>Depositor</td>
<td>Users assigned Depositor access can create documents but can’t see any documents in the database views, even the documents they create. For example, assign Depositor access to allow users to contribute to a mail-in database or to a database used as a ballot box.</td>
</tr>
<tr>
<td>No Access</td>
<td>Users assigned No Access cannot access the database. For example, assign No Access as the default access to prevent most users from accessing a confidential database.</td>
</tr>
</tbody>
</table>
Protecting Databases from Anonymous User Access
To protect databases from unregistered (Anonymous) Web users, create an entry in each database’s ACL called Anonymous and grant it No Access.

In a database ACL, Anonymous can be listed as an individual or as a member of groups or roles, the same as registered Web users. An entry named Anonymous in the ACL specifies the access level of unregistered Web users. If Anonymous isn’t listed in the ACL, Domino grants the unregistered Web user access based on the database’s default access level.

You should place an Anonymous entry in all database ACLs on the Domino server. In a mixed Notes and Domino environment, a database administrator may assume the Default entry is for representing a Notes user. Anonymous users should be more restricted than that.

Maximum Internet Browser Access
The database ACL gives you the option to set an access level for Web users that will set an upper limit to Web access to the database, regardless of individual ACL settings. In the database ACL advanced panel, the field Maximum Internet Browser Access can be set to determine the maximum access allowed for the database from Web browsers.

ACL of Directory Link
A directory link is a pointer file with .DIR, which allows you to take advantage of server space in another location. Using a directory pointer, you can store databases in a directory outside of the Notes data directory. You can also restrict access to the directory and allow only people with access to the directory to view databases in it.

Users see the pointer file as a subdirectory to the Notes server. For example, the pointer file MKTG.DIR appears to users as the Notes subdirectory [MKTG]. Users are unaware that [MKTG] points to another location.

Domino does not support ACLs in databases reached from Web users through directory links, so it is necessary to closely review databases accessed via directory links (.DIR files). Anything stored in this manner should be open to the public.

Setting up the ACL for a Web Database
1. Select the database and choose File - Database - Access Control.
2. Assign yourself and at least one other Manager to the access control list and give yourselves access to all tasks.
3. Set an access level for the - Default - entry.
4. Add entries for Notes and Web users, groups, and servers and set their access levels.
5. Add an entry called Anonymous to set a specific access level for Web users who haven’t registered at the site, that is, for Web users who don’t have Person documents in the Public Address Book.

6. Refine the entries by restricting or allowing individual tasks.

7. Click Advanced and accept or change the Web access level in the Maximum Internet Name & Password Access list.

8. Click OK.

**Example: Setting up the Access Control List for a Web Database**
The following tables give some recommendations for setting up the ACL for various types of databases:

**ACL for high-security confidential database:**

<table>
<thead>
<tr>
<th>Role</th>
<th>Access Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>No access</td>
</tr>
<tr>
<td>Anonymous</td>
<td>No access</td>
</tr>
<tr>
<td>Authorized readers</td>
<td>Reader</td>
</tr>
<tr>
<td>Contributors</td>
<td>Author</td>
</tr>
<tr>
<td>Supervisors</td>
<td>Editor</td>
</tr>
<tr>
<td>Maximum Internet name &amp; password access</td>
<td>Reader</td>
</tr>
</tbody>
</table>

**ACL for medium-security project database:**

<table>
<thead>
<tr>
<th>Role</th>
<th>Access Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Reader</td>
</tr>
<tr>
<td>Anonymous</td>
<td>No access</td>
</tr>
<tr>
<td>Authorized readers</td>
<td>Reader</td>
</tr>
<tr>
<td>Project team members</td>
<td>Editor</td>
</tr>
<tr>
<td>Maximum Internet name &amp; password access</td>
<td>Editor</td>
</tr>
</tbody>
</table>

**ACL for low-security informational database:**

<table>
<thead>
<tr>
<th>Role</th>
<th>Access Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Reader</td>
</tr>
<tr>
<td>Anonymous</td>
<td>Reader</td>
</tr>
<tr>
<td>Contributors</td>
<td>Author</td>
</tr>
<tr>
<td>Supervisors</td>
<td>Editor</td>
</tr>
<tr>
<td>Maximum Internet name &amp; password access</td>
<td>Editor</td>
</tr>
</tbody>
</table>
Programming Considerations

The Web browser does not have the same powerful abilities as Notes clients. The Web browser user can see only what was transmitted to it from Domino. In Domino R4.6, a new @ClientType is added to @function which enables the designer to easily find out which client is accessing the application in order to tailor the functionality and screens best for that type of client.

The following few paragraphs are taken from Mr. Howard Greenberg’s article, “Designing a Secure Domino App.” For the complete article, please refer to the Web site:

notes.net/Today.nsf.

Note We have modified the information from the article slightly to include information on some new R4.6 features.

Hide Content by Hide/When Attributes

To control what information in a document is visible to Web users, both static text and field contents should be hidden. This is accomplished using the hide/when properties that control when information is presented to the user.

Using a Notes client, the hide/when capability is not considered a security feature because a Notes client can always use the document properties InfoBox to see the contents of all fields on a document, including hidden fields, but the Web users can see only what was transmitted to it from the server and information hidden with a hide/when attribute is not sent to the Web browser.

There are several ways to use the hide/when properties.

- Hide the paragraph.
- Hide by writing a formula which controls when the paragraph appears.
Hide the Paragraph
The smallest level of information that can be hidden is one line, which if followed by a carriage return is considered a paragraph. Information can be hidden when the document is in read mode or edit mode. The application developer can also choose to hide the information from the Web user completely by checking the check box, Hide Paragraph from Web Users. The hide/when properties can be set in the properties InfoBox by clicking on the Window Shade tab.

Below is a screen shot of the hide/when settings. The Previewed for Reading, Previewed for Editing, and Printed check boxes do not apply to a Web client.

![Hide/when settings](image)

Hide by Writing a Formula
The formula goes into the Hide Paragraph if Formula is True edit box and is saved when the syntax check box is selected.

For example, if a paragraph should be visible only when John Smith is viewing the document, then the following formula could be used to hide the paragraph.

```
!@UserName="John Smith"
```

**Note**  The exclamation mark means Not.

Encryption and Signature
Electronic signatures do not work from the Web. A designer can choose to have a section signed when the document is saved but this feature will not work from the Web because there is no public key/private key for a Web user.

If field encryption was used to encrypt any fields, those fields will not be visible by a Web user because they do not have the encryption keys necessary to decrypt the field.
SSL security does not have the capability to encrypt or sign individual parts of a document. So if SSL is invoked, the whole of a document is encrypted, including any artwork or other media files that may be embedded.

SSL Implementation On Domino

Why Use SSL?

The most commonly used protocol on the Web is HTTP. HTTP is a lightweight, stateless protocol based on TCP/IP that does not lend itself to a strong authentication method. There is a vulnerability in HTTP basic user ID/password authentication: the user ID and password are included in the packet header, which means that they be captured by anyone with a network sniffer or trace tool at any place in the transmission session path.

How serious is this exposure?

Within a corporate network it may not be a big problem, but on the Web it is a different story. The Web is not a single network, it is comprised of a number of IP networks linked by backbone routers/switchers. This network is accessed by the public and there is no central point of management control. There is a whole community of hackers who use this free-and-easy environment as a playground in which to test their system-cracking skills.

How to deal with this situation?

The answer is to use Secure Sockets Layer (SSL) protocol. The SSL was originally created by Netscape Inc., and is now implemented in Web servers and browsers from many vendors. SSL makes use of a number of cryptographic techniques, such as public key and symmetric key encryption, digital signatures, and public key certificates.

SSL has two main objectives:

1. To ensure confidentiality by encrypting the data stream that a client and server send.
2. To provide authentication of the session partners, using RSA public key methods.

SSL is also adopted by many e-commercial products, but SSL is not just for e-commerce. The most important function that SSL brings to standard TCP/IP and HTTP sessions is the ability to mutually authenticate servers and clients.
By enabling SSL, you can establish trusted relationships between clients and servers using certificates and digital signatures. While this ability has long been available to Notes users, it was a groundbreaking development in security across TCP/IP. In other words, SSL should be understood as a simple, general, useful specification that developers can implement to protect any bi-directional network communications.

SSL offers the following security benefits:

- **Privacy.** Data is encrypted to and from clients, so privacy is ensured during transactions.
- **Message validation.** An encoded message digest accompanies data to detect any message tampering.
- **Server authentication.** The server certificate accompanies messages to assure the client that the server identity is authentic.
- **Client authentication.** The client certificate accompanies messages to assure the server that the client identity is authentic. Client authentication is optional, and may not be a requirement for your organization.

You can use SSL for the following Domino server tasks:

- Web server (HTTP server task).
- Server Web Navigator (Web server task).
  
  **Note** You cannot authenticate client certificates with the Server Web Navigator.
- Network News Transfer Protocol (NNTP server task).
- Post Office Protocol 3 (POP3 server task).
- Internet Message Access Protocol (IMAP server task).

**SSL Overview**

- SSL security is based on certificates used by both the client and server. Domino uses the X.509 format for SSL certificates, which is currently the most widely-recognized certificate format. This allows servers with certificates created by Domino certificate applications to exchange certificates easily between Domino and other applications.
- Certificates are stored in the key ring files, which contain a public key, a name, an expiration date, and a digital signature. A key ring file is a binary file that is protected by a password and stores one or more certificates on the client and server hard drives. Public and private keys are a unique pair of mathematically-related keys that are used to initiate SSL-encrypted transactions.
Certification Authority (CA) issues certificates to both sides: server and client. CA vouches for the identity of a server and client by being stamped with the CA’s digital signature in the certificate. The digital signature ensures the client and server that both the client certificate and the server certificate can be trusted. If the client and server can identify the digital signature on the certificate, then a secure SSL session can be established. Otherwise, the client and server cannot authenticate each other, and the session cannot be established.

Merging a CA’s certificate into a server or client key ring file marks CA as a trusted root. Clients and servers identify digital signatures by comparing them against the trusted root certificate in their key ring files.

A CA can be an external, commercial certifier, such as VeriSign, or an internal certifier that you create at your organization. An external CA can create both server and client certificates. The Domino internal CA application can create server certificates, but cannot create client certificates.

Domino includes several trusted root certificates from external commercial CAs like VeriSign and RSA by default when you create a server key ring file. The most popular browsers have external CA certificates issued from more than ten commercial CA organizations. For example, Netscape Navigator Gold 3.0 has 16 CA certificates. The browser can connect to an SSL-enabled Web site without any difficulty.
SSL in Domino Server Environment

You can enable SSL on a server using one of two methods:

1. **Enable SSL for a server only.** This secures transactions, validates messages, and authenticates server identity, but does not authenticate client identity. You can control access to databases on the server by user name if you set up an Internet password for the user in the Person document.

2. **Enable SSL for clients and server.** This method secures transactions, validates messages, and authenticates both the server and client identities. You can set database access control using individual client user names.

To enable SSL for servers only:
- The servers must have certificates from an internal or external Certification Authority (CA).
- The clients must have certificates from an external CA.
- (Optional) To set up SSL to require a user name and password, clients must have a Person document in the Public Address Book that contains an Internet password.

To enable SSL for clients and servers:
- The servers must have certificates from an internal or external CA.
- Clients must have certificates from an external CA.
- Servers must have the client’s CA certificate marked as a trusted root.
- Every client must have a Person document in the Public Address Book that contains the SSL public key.

The client obtains a certificate issued from an external CA. The Certificate Server application cannot create client certificates.

SSL On Domino Servers Using Domino SSL Applications

To set up SSL for an intranet, use the Certificate Authority (CA) application (CERTCA.NSF) and the Server Certificate Administration (CERTSRV.NSF), both shipped with Domino server R4.6 software.

The Server Certificate Administration application lets you request server certificates from either an internal or external Certification Authority and manage your server certificates in a key ring file, add a CA’s certificate as a trusted root, and create a self-certified certificate for testing purposes.
The Certificate Authority Application lets you become an internal Certification Authority, which includes creating a CA certificate and a key ring file, signing server certificates with the CA’s digital signature, and adding client certificates to the Public Address Book. Server administrators also send their server certificate requests to this database and use a browser to pick up approved certificates.

The Certificate Authority application does not allow you to issue certificates for clients; however, a server can authenticate clients if the client obtains a client certificate from an external CA and the server obtains the client CA certificate.

Setting up an internal CA simplifies the process of creating and managing certificates when your users do not need to communicate with external servers or when external users do not need to access your servers. In addition, setting up an internal CA lets you avoid unnecessary fees that external CAs charge to create and recertify certificates.

When accessing the CA Application and the Server Certificate Administration Application, make sure you have a network connection to the server and are working at the server machine. The Server Certificate Administration Application must access the database locally. The CA Application must access files on the server hard drive over the network.

Setting Up SSL On Domino Servers

Setting up the server involves the following tasks:

1. Setting up the Server Certificate Administration application.
2. Setting up the Certificate Authority application.
3. Creating the Certification Authority key ring file and certificate.
4. Creating the server key ring file and submitting certification requests to an internal Certification Authority (CA).
5. Merging the CA’s certificate into the server key ring file, which marks the CA’s certificate as a trusted root.
6. Configuring the Server document for the server in the Public Address Book to enable SSL for server tasks.
7. (Optionally), forcing all connections to a server or to an individual database to connect using SSL.

Do not set up SSL on a Web server that hosts multiple sites. Domino supports SSL on servers that host only a single site.
Setting Up the SSL Application Database
Assuming that you are working at a Domino CA server:

1. On the server machine using a Notes workstation, (server local client), create two databases
   - one using the CERTSRV.NTF template and naming it CERTSRV.NSF.
   - one using the CERTCA.NTF template and naming it CERTCA.NSF.
   **Note** You must select the option Advanced templates when creating the database to view the CERTSRV.NTF and CERTCA.NTF template.

2. In the access control list (ACL) of the Server Certificate Administration database, set the access to No access to prevent others from using the database.

3. In the access control list of the Certificate Authority database, do the following:
   - Add the names of the people who you want to administer certificates and give them Editor with Delete access.
   - Set the default access to Author with Create access.

4. (Optional) To hide this database when users open databases using File - Database - Open, deselect Show in ‘Open Database‘ dialog in the Database properties box.

Creating A CA Key Ring and Certificate
You use the Certification Authority certificate to sign server certificates. The request is created using standard Public-Key Cryptography Standards (PKCS) format, which is a format Domino CAs as well as other popular CAs recognize.

1. Open the CERTCA.NSF database and select, CERTIFICATION AUTHORITY CONFIGURATION — INITIAL CERTIFICATION AUTHORITY SETUP.

2. Click Create Certificate Authority Key Ring & Certificate.

3. Enter a name for the key ring file in the Key Ring File Name field. The default name is CAKEY.KYR, it is helpful to use the extension .KYR to keep the key ring file name consistent.

4. Enter a password to secure the key ring file contents in the Key Ring Password field.

5. Enter the server’s distinguished name.

6. Click Create Key Ring and Certificate.
7. After you read the information about the key ring file and distinguished server name, click OK.

8. Make a backup copy of the Certificate Authority key ring file and store it in a secure location.

Modifying the Profile for the Certificate Authority Application

The Certificate Authority (CA) profile identifies the CA’s key ring file and specifies the server name to use for the URLs mailed to the server administrators notifying them to pick up the server certificate.

1. Open the CERTCA.NSF database and select, CERTIFICATION AUTHORITY CONFIGURATION — INITIAL CERTIFICATION AUTHORITY SETUP.

2. Click Configure Certificate Authority Profile.

3. If necessary, enter the key ring file name.

4. Enter the host name or IP address of the server.

   **Note**  Domino uses this name when sending e-mails to server administrators indicating the location of the signed server certificate that they need to pick up.

5. Save the profile document.
Creating a Key Ring File and Certificate Request for the CA Server

Follow the steps below to create a key ring file and a certificate request that can be sent to a CA.

1. Open Domino Server Certificate Administration (CERTSRV.NSF).
2. Click Create Key Ring.
3. Enter a name for the key ring file and password in the Key Ring File Name and Key Ring Password fields. The default key ring file name is Keyfile.kyr.
4. Specify the components of your server’s distinguished name.
5. Click Create Key Ring.
6. After you read the information about the key ring file and distinguished server name, click OK.
7. Click Create Certificate Request.
8. From the Server Certification Administration main panel, click Create Certificate Request.
9. Enter the name and password of the key ring file that you specified in step 3.
10. If you want to log information about this request in the Server Certificate Administration application, select Yes in the Log Certificate Request field. Otherwise, select No.
11. Click Paste into Form on CA’s Site.
12. Click Create Certificate Request.
13. Highlight the text in the dialog box, press CTRL+C to copy it to the Clipboard, and click OK.
   **Note** You must select all the text in the second dialog box, including Begin Certificate and End Certificate.
**Signing Server Certificates by CA**
When you sign a server certificate, you add the CA’s digital signature to the server certificate. The server certificate is made available to the CA in the Certificate Authority application. On the Certification Authority Application main panel, there are specific choices for CA server application certificates. Under Configuration Utilities, one is Submit Server Certification Request, another is Pick Up Certification Authority Certificate.

1. Open Domino Certificate Authority (CERTCA.NSF) on the local machine to the server.
2. Click Submit Server Certification Request.
3. Fill in Name and E-mail address by pasting text from the clipboard to the field Certificate.
4. Click Submit Server Certification Request button.
5. Go back to the Certification Authority Application main panel and click Pick Up Certification Authority Certificate.
6. Review the user information and distinguished name. Copy the Certificate content text to the clipboard, then click Done.

**Merging A Server Certificate**
When the Certification Authority (CA) approves your request, merge the signed certificate into the server key ring file.

1. Open the Domino Server Certificate Administration database (CERTSRV.NSF) on the local machine.
2. Click Install Trusted Root Certificate into Key Ring.
3. Enter the file name and password for the key ring.
4. Select Clipboard in the Approved Certificate is Supplied In field. Paste the Clipboard contents into the next field.
5. Click Merge trusted root Certificate into Key Ring and click OK twice to approve the merge.
From the server Public Address Book, choose Server and select the Internet Port and Security Configuration section.

1. In the SSL Protocol Version field, select one of the following:
   
   **Note**  This setting applies only to NNTP, IMAP, POP3, and LDAP connections.
   
   - R2.0 only. Allows only SSL 2.0 connections.
   - Negotiated (default). Attempts an SSL 3.0 connection. If this fails, attempts to connect using SSL 2.0. Use this setting unless you are having connection problems due to incompatible protocol versions.
   - R3.0 handshake. Attempts an SSL 3.0 connection. If this fails and the requester detects SSL 2.0, then attempts to connect using SSL 2.0.
   - R3.0 only. Allows only SSL 3.0 connections.
   - R3.0 and R2.0 handshake. Attempts an SSL 3.0 connection, but starts with an SSL 2.0 handshake, which displays relevant error messages. Makes an SSL 3.0 connection if possible.

2. In the SSL Key File field, enter the name of the key ring file that the server uses.
3. In the column for the protocol for which you want to enable SSL, select Enabled in the SSL Port Status field and, if necessary, change the SSL port number. For example, to enable SSL for the Domino Web server, select Enabled in the SSL Port Status field in the Web column.

4. Select YES in the Client certificate field for the protocol.

5. To enforce SSL for all databases on a server, select Redirect to SSL in the TCP/IP Port Status field for the protocol for which you want to force SSL connections.

   **Note** The Redirect to SSL setting is not supported by NNTP and POP3.

6. Do the following to add a Person document to the Public Address Book:

   - Open the Public Address Book.
   - Create a Person document and enter the person’s first, middle, and last names in the First name, Middle initial, and Last name fields.
   - Enter the user’s common name on the certificate in the User name field. Domino uses the first name listed in the User name field when verifying names in database access control lists.
   - (Optional) Enter additional information about the user in the Work and Home sections.
   - Save the document.
You can set up access to an individual database so only those who have the appropriate SSL certificates can access it. The server denies access to clients connecting using TCP/IP. To enforce SSL on an individual database, select the database for which you want to force clients to use SSL and open the Database properties box. On the Basics tab, select “Force SSL Connection on the Web.”

**Getting the Client Certificate from the Server**

When the client tries to connect to the server, Domino looks for the client’s SSL common name in the User name field in the Public Address Book. If Domino finds a match, then the first name listed in the User name field is used to check access control lists. If Domino finds multiple matches for the user name in the Public Address Book, it compares the public key in the Person document to the public key in the client’s certificate.

If you have Client Certificate and Name & Password selected, Domino first checks the client for a certificate. If the server cannot find a certificate, Domino asks the client for a name and password. If the name and password do not match the Person document, the client is denied access to the server. Similarly, if you have Client certificate and Anonymous selected, Domino checks for the client certificate and if not, connects the user to the server anonymously. If you do not want clients that do not have SSL certificates to access a server using basic password authentication or anonymous access, select No in the Name & password and Anonymous fields for the protocol in the Server document.

You must have the Administration Process set up to add client certificates to the Public Address Book. Check the Administration Requests database for errors when adding the user’s certificate. The client cannot use the certificate until the Administration Process completes the request.

The client must do the following to submit the certificate to the server administrator:

- Use a browser to open the Certificate Authority application (CERTCA.NSF) and click Register Browser Certificate in Address Book.
- Enter your name, e-mail address, phone number, and any comments. This information is used on the request entered in the Certificate Authority application.
- Click Submit Certificate.

The CA must do the following to add the certificate to the Person document:

- On the server machine from the administration panel, click System Databases and choose Open Domino Certificate Authority (CERTCA.NSF) using a network connection to the server.
- Click Certificate Registration.
Open the request document.

If you do not want to approve the request, click Reject and enter a reason for rejecting the request. Otherwise, click Accept and click OK.

Domino sends the client an e-mail using the address specified in the Email address field indicating that you denied or accepted the request and, if you denied the request, the reason why you denied it. You can disable e-mail notification by deselecting Send a Notification Email to the Requester.

**Example: Web User Validation and Authentication**

SSL uses a public-private key pair for users and servers to encrypt a transaction. The public key certificate is available to everyone and contains identification information for the user, a public key value, the name of the certification authority, and a digital signature by a certification authority. The user must use the public-private key pair to decrypt the message.

When a client initiates a connection with the server, the server attempts to authenticate it. The server determines whether a client certificate is required or the client should be challenged for a name and password.

The following is an example of how a client (ClientA) connects to a server (ServerB) using the SSL protocol.

1. ClientA sends a request to ServerB specifying information about the SSL connection, such as supported encryption algorithms and certificate expiration date.

2. ServerB sends ClientA its certificate that contains ServerB’s public key. ClientA checks the certification authority’s digital signature on ServerB’s certificate to verify the identity of ServerB. If the digital signature in ServerB’s certificate has been tampered with, ServerB cannot be verified.

3. ClientA uses an algorithm to create a secret encryption key pair, encrypts the secret key using the public key stored in ServerB’s certificate, and sends it to ServerB. This secret key changes for every session, which makes it difficult for users to trace.

4. ServerB decrypts the secret encryption key using ServerB’s private key and uses the secret encryption key to encrypt data passing between ClientA and ServerB after this point.

5. If client-side certification is enabled, ServerB requests ClientA’s certificate.

6. ClientA sends ServerB his certificate. ServerB checks the certification authority’s digital signature on ClientA’s certificate to verify the identity of ClientA. If the digital signature in ClientA’s certificate has been tampered with, ClientA cannot be verified.
Conclusion

The Domino server and most browsers are shipped with an SSL feature, and with several default CA certificates. It is now much easier to set up the SSL security “wall” than in previous releases to protect message transactions and consistency. With internal CA setup on the intranet, you can also authenticate client identity for Web browser clients.

Summary

With open standard Internet security technology emerging, Domino adopts these new standards into its architecture. From this chapter, you should now have an awareness and basic understanding of:

- Domino security features for Web applications.
- How to implement SSL on the Domino platform.
- How to deploy a security strategy in a mixed Web users and Notes users environment.
Chapter 10
Java Applets, Java Agents, and JavaScript

This chapter will cover the topic of the Java technology implementation on the Domino platform. We will focus on Java applets and Java agents.

We will discuss:
- How to import Java applets and Java programs into Domino design elements.
- The Java Notes classes.
- How to use Java Notes classes in your application.

At the end of the chapter there is a section covering JavaScript. JavaScript is a scripting language, similar in syntax to Java, used for scripting Web applications. The section explains how to use JavaScript in Domino, through several useful examples.

Overview

As the Web evolves, more and more Web pages contain interactive elements, such as animation, sound, and multimedia. Using the Web Navigator, you can view the various kinds of interactive Web pages that include Java applets.

Since Domino Release 4.5, the Notes client has had the ability to run Java applets. Now, with Release 4.6, you can insert a Java applet directly on a form so that the applet is included in each document created from the form, or you can insert the applet in a rich text field of a document to include it only in that document.

**Note** You cannot create Java applets directly in Notes, but you can use Lotus BeanMachine™ or another authoring tool to create the applets.

What is a Java Applet?

A Java applet is a Java program that runs in the context of a browser and is included in an HTML page with the <APPLET> tag.

The *base class* is a Java applet file, which contains the main class, and is where the applet begins execution. Generally Java applets are not one file...
but a whole collection of files. As well as the base class file, the applet can also contain image files, archive files, and Java source files.

If the files for the applet are stored in your local file system, Notes provides tools to collect all of the files needed to run the applet. Notes imports Java applet files from your file system and Java applets are stored as a special kind of attachment to a form or document. They differ from regular Notes attachments in that regular Notes attachments are displayed as file icons, but Java applets are displayed as running applets.

You can link to a Java Applet stored on the Web by creating a URL link in your Domino application form or document.

**What are Java Agents?**

You can create and run Java Agents on Domino servers. Java agents differ from Java applets in that Java agents are written explicitly for Notes, in order to build Notes agents using Java programs.

When you write a Java agent for Notes, you must create the Java class files using a Java development environment, such as IBM VisualAge for Java. In these Java programs you use the Java Notes classes to access Notes databases in the same way that you use LotusScript. Both languages share the same object model.

Domino supports programs written in Java 1.1.x. and above. To learn more about Java Notes classes, please refer to The Java Programmer’s Guide which is available as a database (javapg.nsf) and installs with other Domino Version 4.6 documentation databases in the \notes\data\doc subdirectory.

---

**Java Applets**

On Notes 4.5, there is no UI mechanism for inserting applets. To insert them in a Domino form or document you use a Web navigator to display the Web page, and then copy and paste the applet from the Web page into the Notes form or document.

With Domino Release 4.6, it is much easier to use Java applets in programming. In the Notes Designer environment you can create Java applets from the Create menu in edit mode, as shown below:
The Java applet can be created on forms, subforms and documents, including the “Using” and “About” documents. In order to position the applet in the document or form, it is best to place it in a table. The applet is just like a layout region, hotspot, or picture in that it is a design element in your programming.

Java applets cannot be used in encrypted databases (or in encrypted fields within a database).

Java applets created with Domino Release 4.6 will not work with any earlier version. Java applets created with Domino version 4.5 will work with Release 4.6, however, if you modify the applet parameters using 4.6, the applet is then saved as a 4.6 applet and ceases to work in 4.5. In order to run in both Domino versions, use HTL to reference the applet in the file system.

Displaying Java Applets

When a Java applet is displayed via a Web browser, the display and debugging are under the control of the Web browser. For example, Web clients can wrap text around an applet while the Notes client cannot. If the document/page designer takes advantage of the wrapping capabilities of the Web client, the display will differ.

When the applet is displayed via the Notes client, if the applet can’t be loaded, the Alt-Text for the applet is displayed. If no Alt-Text is provided,
for example, if the alt text formula evaluates to an empty string, a gray box of the picture with the correct size is displayed. Information about why the applet could not be run is displayed in the status area.

If the current edit session is set to View - Show - Java Applets Running (the default), the applet is running (is loaded) as soon as the document is opened (or the page is downloaded).

When the applet runs, the status bar displays “loaded, initialized, started.” If the applet does not run as expected or does not load properly, a dotted gray rectangle is displayed on the screen. Choose File - Tools - Show Java Debug Console to examine the applet and determine the problem.

When an applet is turned off, it behaves like a non-activated OLE object. To delete a Java applet from your document, simply select it and delete it. When you print a document or Web page containing an applet, a gray box of the correct size is printed in place of the applet.

---

**Inserting Java Applets into Domino Design Elements**

You can insert Java applets in several different ways:

- Import a copy of an applet into a form or document.
- Copy an applet from an existing document into a different form/document.
- Reference an applet via a URL to the location (on a Web server) where the applet is located.

Inserting an applet in a Domino form or document is a two-step process:

1. Attach the applet files from your file system or create a link to an applet on the Web.
2. Use the design pane and the Properties box to set the applet attributes, such as its size and parameter values.

Some applets will run with no modifications (without the applet attributes being modified) but most require that the attributes be set before the applet can be successfully run.

By default, applets are “on” (running) when inserted into a document or form. If the applet is running, you cannot click the applet to select it. Instead, click to the right of the applet, and use the left arrow key to select the applet. Choose View - Show - Java Applets Running to stop all applets in a form or document, one click to select an applet, double click to restart an applet.
Enabling Java Applets

Before inserting an applet you must enable Java applets in your Notes environment:

If you are linking to an applet on the Web, make sure your Location document specifies a valid Web proxy. See your system administrator if you have questions about setting up a Web proxy.

2. Select Enable Java applets from the list of Advanced options.
3. Click OK.

Note If this option does not display as a preference, check with your system administrator to make sure that the Java files necessary for running applets are installed on your workstation.

Importing an Applet from your File System

If you import an applet, you should specify the pathname of the directory containing the base class file (or .jar file) of the applet, and the name of the applet’s base class (this would be the CodeBase if you were writing the HTML applet tag for the applet).

1. Open a form in design mode or click in the rich text field of a document.
2. Choose Create - Java Applet.

3. Select Import an applet from the file system.
4. Enter the path for the applet files in the Base directory field. Enter the name of the main class for the applet in the Class name field. Click the folder icon to locate the main class file for the applet.
5. Click Locate to see all related files for the applet.
6. Select any additional applet files. Use the Add, Add All, Clear, and Clear All buttons to add or remove applet files.

7. Click OK twice.

8. (Optional) Set applet parameters and properties using the design pane and the Properties box.

   In most cases you must select all applet class and resource files. Include the source files if you plan to send the applet to another user, so that they can then edit the applet.

**Linking to an Applet on the Web**

Many very good Java applets are located in the Web. To use any of these you just need to provide the location of the required Java applet by specifying the correct URL in the Notes document or form.

**Note** You cannot insert a Java applet into a form or document until Enable Java Applets is selected in the User Preferences box.

1. Open a form in design mode or click in the rich-text field of a document.
2. Choose Create - Java Applet.
3. Select Link to an applet on a Web server.
4. Type or paste the name of the URL where the applet files are located, and then type or paste the name of the main class file for the applet.
5. Click OK.

6. (Optional) Set applet parameters and properties using the design pane and the Properties box.

Setting Applet Attributes

Applets have display attributes that control, for example, the height and width of the applet. Some also have parameters that pass values to the applet on start-up. After inserting an applet, set the applet attributes to control how the applet displays in your application.

Some applets have default value parameters. These applets can run successfully without further work. For those applets without default values you must set parameter values, otherwise the applet will not run. You can manually enter parameters and values, or you can paste all of the parameters into the parameter design pane and edit the associated values.

There are three main options where the attributes of a Java applet are set; all can be found from the Java Applet pull down menu in Notes Designer:

2. Java Applet Parameters.
3. Design, Reorganize, and Export options.

We will first explain how to stop a Java applet and then how to display the Java applet menu before discussing each of these options.

Stopping a Java Applet

Before setting or changing the attributes for the applet, the applet should be stopped. Any action which causes the page to be recalculated will cause an applet to be stopped.

To stop all applets in a form or document:

1. Click the EDIT PAGE action button to bring the document page into EDIT mode.
2. Choose View - Show - Java Applets Running. Deselect it to remove the check mark.

The applet area will become a grey color and the applet is then stopped.

Displaying the Java Applet Menu

Follow the steps below to display the Java Applet menu:

1. Move the cursor over the applet and click the left mouse button. The JavaApplet menu appears on the menu bar.
2. Move the cursor over the applet and click the right mouse button to bring up the Java Applet Properties and Java Applet Parameters menu.

**Setting the Applet Properties**

Selecting Applet Properties pops up the Applet Infobox. The Infobox is used to set all properties of an applet that are not computed with a formula. Properties that are computed by formula are set using events in the formula pane.

Applets have six Infobox panels:

1. **General Applet Properties** - unique to applets.
2. **Alignment Properties** - standard alignment properties.
3. **Files Properties** - unique to applets.
4. **Pagination/Tabs Properties** - standard options.
5. **Hide/When Properties** - in Release 4.6 these include the Hide from Notes/Web additions.
6. **Security Properties** - standard options.
We will discuss the panels that are unique to Java applets:

**General Applet Properties Tab**
- Base class is the name of the applet’s main class. When editing make sure that there is a class with the new name either in the list of files or in a .jar file.
- DocumentBase is provided for information only (it is not editable). It contains the results of a GetDocumentBase call from within the applet.
- If the applet is a link to a Web server, CodeBase is the URL of the directory containing the applet. The field contains the results of a GetCodeBase call from the applet.
- The Height and Width fields are used to set the (pixel) height and width of the applet display area. Changing the Height/Width causes the applet to be resized.
- Alternate text contains the text which is displayed if the applet fails to load.

**Files Properties Tab**
This is a non-editable list box (horizontal and vertical scrolling enabled), that lists the files that compose the applet.

**Setting Other HTML Attributes**

**HTML attributes**
This event identifies the HTML that is inserted between the `<applet>` tag and `</applet>` tag. Users can insert any attributes that Domino does not automatically generate or that are custom/specific to a browser.

You might also want to set the alignment attributes using these events.

**Alternate HTML**
This event identifies the HTML that is inserted between the `<applet>` tag and the `</applet>` tag. This HTML is displayed by browsers that do not support Java. This is different from Alt-Text (defined in the Infobox) which is displayed by browsers that do understand Java applet tags but can’t show the applet.

HTML for non-Java-enabled clients can contain any valid basic HTML, for example, links, text, and image maps.
Setting the Applet Parameters

From the Java Applet Properties and Parameters menu, select Java Applet Parameters.

When an applet is created the formula pane is displayed, and closed when focus returns to the document window.
Enter the parameters (names and values) to be passed to the applet on start up. The value for a parameter is in a Notes formula format, so if you enter simple text be sure to put it in quotation marks.

If the parameter value is left blank a null value is generated for the applet tag.

To Set Individual Applet Parameters
1. Select the Java Applet Parameters event in the design pane.
2. Determine the parameter and value you need to set. You may find needed parameters by viewing the HTML file in a text editor.
3. Click the Add button in the Parameter name window of the design pane.
4. Enter each parameter name.
5. Enter the corresponding value for each parameter in the Parameter Value window. The parameter value is a formula format, therefore, you must enclose text values in double quotes. In addition to text strings, a parameter value can be an @function, or a field name.

To Set All Applet Parameters
1. Open the HTML source file for the applet in a browser or text editor.
2. Select all of the parameter text in the HTML file and copy it to the clipboard.
3. In the JavaApplet design pane, click the Paste button in the Parameters window. All of the parameters from the HTML file are pasted into the Parameter name and the Parameter value window.
4. Click a parameter name to display its current value in the Parameter value window.
5. Set each parameter value.

You can set three applets properties by selecting them from the Event pull down menu.
Setting the Design, Reorganize, and Export Options

There are three other Java Applet menu items that enable you to change the attributes of a Java applet, Refresh, Reorganize, and Export and we will cover each, briefly, here.

**Note**  These three items are only enabled if the applet is not a link.

**Refresh Dialog**

Refresh is used to reload the applet files from the file system. It replaces (refreshes) existing files and also can be used to add additional files.

Selecting Refresh brings up the same dialog box as the “Locate Applet Files” dialog box brought up when first creating the applet.

**Reorganize Dialog**

Reorganize is used to change the order or delete the files that make up an applet. You edit, reorder, and delete the file list using the reorganize dialog box.

**Export Dialog**

Selecting Export brings up a dialog box. The Export dialog box identifies the root directory where the class tree is exported. All files that are saved as pieces of the applet are exported with their current file names and subdirectory structures.

---

Tips and Troubleshooting for Java Applets

**Troubleshooting an Applet that is Not Running**

Here are a number of suggestions to check if your applet is not running as you expected:

- If the applet is linked to a Web site, check that your Web proxy is running.
- Get the name of your Web proxy from your Location document. If you are running Windows, go to the DOS prompt and use the ping command followed by the proxy name to determine if your connection is valid. If ping returns an error rather than a reply, contact your system administrator for help restoring your Web proxy.
- Make sure that you have included all the necessary applet files. Choose Java Applet - Refresh to add additional files to the applet.
- Make sure you have correctly specified parameters required by the applet.
Accessing Resource Files

Java applets frequently use resource files such as images and audio files. There are three common ways that applets access these files:

- By specifying a full URL for the file, for example:
  

- By using the applet class getDocumentBase method to construct a URL relative to the location of the document in which the applet is found.

- By using the applet class getCodeBase method to construct a URL relative to the code base of the applet, that is, the location from which the applet’s files were loaded.

Depending on what method you use, Notes may not be able to locate resource files for the applet. The `getCodeBase` method is the most reliable method for specifying resource files. If you are experiencing problems with applets not finding resource files, modify your applet to use getCodeBase. Recompile the files and choose Java Applet - Refresh to replace these files in the document.

Basic Error Handling

- Simple understandable errors are reported to the Web browser/Notes client for display in the status bar.

- More complex errors (e.g., null class) are reported via a status bar/dialog box (Web browser) or Java Console (Notes).

If a class fails to load, information about why this may have happened is reported in the Debug console of Notes.

Java Notes Classes and Java Agents

What is a Java Agent?

Java agents are Notes API programs written using Java. The Java agent’s function is the same as the traditional Lotus Notes agent and all the features of a Lotus Notes agent can be found in a Java agent. Java agents also have full agent security, such as signing, and restricted/restricted authors. They also have the full agent scheduling and triggering ability from both the client and the server. Java agents can be run in the foreground or the background. Java agents can also be replicated.

Through the Java agent, the Notes object interface is exposed to Java. Java applications can access and manipulate Notes database objects.

Domino supports Java programs written in Java 1.1.x.
About Java Notes Classes

A number of Java Notes classes are shipped with Domino Release 4.6 and provide new opportunities for Domino developers.

Java Notes classes are created by modifying some of the LotusScript Extension (LSX) architecture to include a Java “adapter” to compose the new Java Notes classes. The Java Notes classes have similar functions to some of the LotusScript Notes back-end classes. You can use these classes from any Java program (within the Notes Designer environment or outside of it) as long as Notes Release 4.6 is installed on the machine.

Internally, Java Notes classes execute the same C++ code as the Lotus Notes back-end classes, only the language syntax is different. The table below shows how some of the new Java Notes classes correspond to LotusScript classes:

<table>
<thead>
<tr>
<th>Notes Java class</th>
<th>LotusScript class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lotus.Notes.Session</td>
<td>NotesSession</td>
</tr>
<tr>
<td>Lotus.Notes.Database</td>
<td>NotesDatabase</td>
</tr>
<tr>
<td>Lotus.Notes.View</td>
<td>NotesView</td>
</tr>
<tr>
<td>Lotus.Notes.Item</td>
<td>NotesItem</td>
</tr>
</tbody>
</table>

The imported Java files for agents can be of the following types:

- Class - *.class
- Archive - *.jar

A Java program is generally made up of a number of files. You must designate one as the Base Class, which is the starting point for the agent program. There can also be archive files and source files, stored either on your local workstation or on a server.

When you write a Java agent, the class you write must extend the class *lotus.notes.AgentBase*. The code you want to execute when the agent runs is in the *NotesMain()* method. The Java Programmer’s Guide describes the Java Notes classes and is available as a database (*javapg.nsf*). It installs with other Release 4.6 documentation databases in the \notes\data\doc subdirectory.
Java Agents and Java Applets

The model for Java agents differs from Java applets in a number of ways:

- Java agents are written explicitly for Notes. Applets are often designed to be served up by any Web server.
- Java agents behave in the same way as LotusScript agents but Java applets behave like Java applets in any Web-authoring environment.
- Java agents run only within a Notes-supplied Java runtime environment while Java applets run in both Notes-supplied Java runtimes and browser-supplied runtimes.
- Java agents are structured in the same way as Java applications (not as applets). They run within a Notes-supplied context as opposed to applets whose context is provided in part by the browser and in part by the codebase parameter specified as part of the applet tag. For agents, CodeBase and DocBase are not meaningful. Instead, like other Java applications, classes and resources are located within .jar files and the class path.
- Java agents can access Notes databases directly using the Java Notes classes. Applets can only access Notes objects within Notes using URLs. Note that nothing precludes a Java agent from using URLs to access a Notes object in Notes.
- Agents do not have a UI (and consequently do not use resources as much as applets).

Compiling and Running a Java Program

A Java program using the Notes classes has the following requirements:

- Lotus Domino (server or client) must be installed on the machine for both compilation and execution.
- The CLASSPATH environment variable must include Notes.jar in the Notes directory. For example:
  
  ```
  set CLASSPATH=c:\notes\Notes.jar
  ```
  
- The program must import the lotus.notes package (the files under the Notes directory are notes.jar, i18n.jar, rt.jar, icsclass.jar).
A stand-alone program must use the lotus.notes.NotesThread class, which extends Java.lang.Thread. You can either extend NotesThread or implement the Runnable interface. If you extend NotesThread, the entry point to the functional code must be public void runNotes(). If you implement Runnable, the entry point must be public void run().

A Domino or Notes agent program must extend the lotus.notes.AgentBase class, which extends lotus.notes.NotesThread. The class that contains the agent code must be public. The entry point to the functional code must be public void NotesMain().

The lotus.notes.Session class is the root of the Notes back-end object containment hierarchy. For stand-alone programs, use the method newInstance() to create a Session object. For agents, use the AgentBase method getSession().

System.exit must not be used to terminate a program using the NotesThread class (and by extension the AgentBase class). In an agent, System.exit throws SecurityException. In a stand-alone program, System.exit may cause corruption problems.

For foreground agents, System.out and System.err output goes to the Java debug console. For locally scheduled agents, System.out and System.err output goes to the Notes log.

Creating a Java Agent

A Java agent is created by attaching a Java program (.class file) to an agent in a Notes client environment. The Java program must be written in a Java development environment first, and then included in the Notes environment using the agent design pane to import the Java files into the agent.

1. With the agent in Design mode, click Java in the design pane.
2. Click Import Class Files.
3. Select the base directory. Files must be in the base directory or in a subdirectory of the base directory.

4. Select the files you need and click Add or Add All.
5. Select each file in the Selected Files list and use the up and down arrows to designate the order in which Domino should load them at run time.
6. Click OK.
7. If you want a Base class that is not the first in the files list, type the name in the Base class list.

To attach new or changed files later, click Reimport Class Files.

To change the order of selected files later or to delete files, click Reorganize Class Files.

For writing and running Java agent programs, Domino Release 4.6 (server or client) must be installed, the CLASSPATH environment variable must include Notes.jar in the Notes directory. For example:

```
set CLASSPATH= c:\notes\notes.jar
set PATH = c:\notes ( need nlsxbe.dll in the directory )
```

In the Java program, the lotus.notes package must be imported, the statement "import lotus.notes.*" must be in lowercase.

---

276 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
Example 1: Using Java Notes Classes

This sample Java program is from the Lotus Technology Learning Center. The Java code is commented to help you understand how the Java Notes class is implemented.

This program creates an instance of NotesThread, a class which extends the Java Thread class. It allows Notes to properly initialize and terminate per thread in a convenient way for the programmer.

This sample program does the following things:
1. Creates a new Notes session.
2. Opens a database (in this case, the local name and address book).
3. Accesses the People view.
4. Searches the People view for the entered name.
5. Accesses the document that matches the search criteria.
6. Pulls the Spouse field out of the document.
7. Prints the Spouse field in a message output.

To Run This Sample:
1. Add a person John Smith and his spouse Mary Smith into the local Address Book. John Smith will be used as a parameter to the command to run the Java program.
2. Write the following code into a Java program (.java), set Windows 95 PATH and CLASSPATH as follows:
   
   ```
   PATH = c:\jdk1.1.3\bin;c:\notes;
   CLASSPATH = c:\jdk1.1.3\lib\classes.zip;c:\notes\notes.jar;
   ```
3. Compile the Java program.
4. Type the command:
   ```
   javac myjavafile.java
   ```
   The output is a file named abe.class.

Chapter 10: Java Applets, Java Agents, and JavaScript  277
5. Run this class file at a DOS command prompt:
   C:\jdk1.1.3\bin> java abe.class  John Smith
   The output will be like this:
   Creating Notes session...
   User name = CN = John Smith OU=CAM O= Lotus
   Spouse of John is Mary Smith
   Date Created : 08/15/97 16:00:00 PM EDT
The sample program is listed below for your information:
Copyright 1997, Iris Associates, Inc.
Sample Java program, for illustrative purposes only.
*/
import lotus.notes.*;
import java.lang.*;
import java.util.*;

class abe implements Runnable
{
    public String g_name;

    // if you run the class from the command line...
    public static void main(String argv[])
    throws Exception
    {
        // print out a message, then exit, no args provided
        if (argv == null || argv.length == 0)
            System.out.println("Usage: java abe <user name>");
        else
            { // create new instance of abe
                abe t = new abe();
            
278 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
// store name to look up in the instance
t.g_name = argv[0];

// make sure the Notes lsx is loaded
NotesThread.load(true);

// create a thread instance for running abe, start it
NotesThread nt = new NotesThread((Runnable)t);

// start the thread, call our runNotes()
nt.start();
}
}

// this would get called if we ran it from java.lang.Thread instead
public void run()
{
    runNotes();
}

public void runNotes()
{
    int i;
    try
    {
        System.out.println("Creating Notes session...");
        Session s = Session.newInstance();

        // show off, print the current user's name
        System.out.println("User name = " + s.getUserName());
    }
}
// get db instance for the name and address db
Database db = s.getDatabase("","names.nsf");

// find the "People" view
View view = db.getView("People");

// search for the name provided
view.FTSearch(g_name);

// for now, ignore multiple matches
Document doc = view.getFirstDocument();

// look up contents of the "spouse" field
String name = doc.getItemValueString("Spouse");
System.out.println("Spouse of " + g_name + " is " + name);

// also print out the date the document was created
System.out.println("Date created: " + doc.getCreated());
}
catch (Exception e)
{
    e.printStackTrace();
}
}
Example 2: Java Agent

This example shows an agent that runs on newly created and modified documents since the agent was last run. The program works on the unprocessed documents, prints each Subject item, and marks each document as processed. The first time the agent runs, the agent returns all of the documents in the database. Thereafter, the agent returns those documents that updateProcessedDoc has not touched.

To Run This Sample
1. Write the following sample code into a Java program named myagent.Java.

2. If you use (SUN Java JDK V 1.1.3), set Windows 95 PATH and CLASSPATH as follows:
   
   PATH = c:\jdk1.1.3\bin;c:\notes\

   CLASSPATH = c:\jdk1.1.3\lib\classes.zip;c:\notes\notes.jar;

3. Create an agent inside your mail database:
   - Name the agent.
   - Select When should this agent run = Manually from Actions Menu.
   - Which documents should it act on = All documents in database.
   - Select Java as your source code.
   - Then import file myagent.class into agent builder.
   - Save it.
The following is the record in the agent log after the agent is first run.

The Java agent program is shown below for information:

```java
import java.util.*;
import lotus.notes.*;

public class myagent extends AgentBase
{
    public void NotesMain()
    {
        try
        {
            Session s = getSession();
            AgentContext ac = s.getAgentContext();
            DocumentCollection dc = ac.getUnprocessedDocuments();
            Document doc;
            int size = dc.getCount();
            System.out.println("Count = " + size);
            for (int i = 1; i <= size; i++)
            {
                doc = dc.getNthDocument(i);
                System.out.println
                    (i + " *** " + doc.getItemValue("Subject"));
            }
        }
    }
}
```

282 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
Using JavaScript in Domino

Using JavaScript it is possible to modify the standard behavior of Web pages by adding some “client side” functions to them. Client side means that no “server side” activity is requested by JavaScript functions so that their result comes quickly to the end user. In fact, JavaScript is executed on the browser and since some differences exist between different browsers it is recommended that you test your JavaScript code using more than one Web browser.

JavaScript allows you to handle events such as onLoad (for a Web page), onClick (for an input button on form), onChange, onBlur, onFocus (for input fields) and so on. You can use these events to trigger JavaScript functions that can also perform some complex operations. JavaScript functions can access all elements on a Web page (like input fields) as well as properties and methods that control the status and the behavior of the Web browser window itself.

This section contains:

- Basics about adding JavaScript to forms.
- Some examples.
- Some considerations about cookies.

ac.updateProcessedDoc(doc);

} catch (Exception e) {
   e.printStackTrace();
}
Adding JavaScript to Forms

JavaScript functions are usually defined inside the <Head> </Head> because you often need those functions loaded before any other element of the Web page. The JavaScript is then included between <SCRIPT> </SCRIPT>.

To achieve this in a Domino form you must use the $$HTMLHead field. The JavaScript functions are stored as formulas so make sure that you replace all double quotes with " and all single quotes with ' before pasting the script as a value formula for that field.

Caution Remember to hide the $$HTMLHead using the Hide-when formulas of the Property box for Text.

If you have a long JavaScript library of functions it might be better to reference a single document storing that library instead of copying it in each form or design element that needs those functions. In order to do this, place on a form (or inside the $$HTMLHead) the following HTML:

<SCRIPT LANGUAGE="JavaScript"

SRC="/dbName/JSview/Module1/$file/module1.js">

Note In this example we have assumed that we have a view (JSView) that collects the document whose attachments are JavaScript files. We access a document using a key value (Module1) and so that we reach its attached file (module1).js. The Domino server will recognize the .js files only if the HTTPD.cnf file contains:

AddType .js application/x-javascript binary 1.0 #JavaScript

See Appendix D for more information on Editing the HTTPD.cnf to add MIME types.

Adding JavaScript to Domino forms and fields is particularly useful as it allows you to create forms with a more dynamic behavior without adding workload to the Domino server. For example, using JavaScript, field values can be validated locally, on the browser, instead of on the Domino server, after submitting.

The next section contains some examples of using JavaScript in forms and fields. Each of the following examples shows some effects you can implement in JavaScript.

Caution Many of the following examples involve cutting and pasting some JavaScript or HTML code into a formula. The formula might be a value formula of a field or an HTML Attributes formula. You should remember the following rules:
Example 1: Auto-Refresh, Field Validation and Help Fields

The form in the picture below allows a user to insert a percentage value in one of the two fields and the other field is then computed as a complementary percentage. For example, if a user inserts 70 in A, he will see 30 appearing in B immediately after changing the focus. If a number higher than 100 is entered, a JavaScript alert message is displayed. Also, when a user puts the mouse inside a field, a help message is displayed on the browser’s bottom bar.

Note All validation operations are performed locally, without calling any server tasks.

To create this form do the following:

1. Create two Numeric Editable fields, PercentA and PercentB.
2. In the HTML Attributes event of PercentA write:

   ```javascript
   onBlur="if(this.form.PercentA.value<=100)
   {
   this.form.PercentB.value=100 - this.form.PercentA.value;
   }
   ```
else
{
    alert('Invalid Percentage A!');
};
window.status='';  

onFocus="window.status=
window.status=
    'Insert Percentage A and look to Percentage B' 

We have already discussed the use of the HTML Attributes in Chapter 6: Part 1, Adding HTML to a Form. In this case you will find all the above code added inside the <INPUT> block that is the HTML translation of a Domino Editable field. To check this, look at the Web page HTML source code.

onBlur is an event rising each time we move the focus out from a field and here it is used to trigger the validation and auto-refresh ‘procedure’ (if()...else block) and to set the message bar to blank (window.status="''). Fields accessed in these procedures are those on the Web form so you cannot access hidden or computed Domino fields using JavaScript because these are not fields on the Web page.

The onFocus event occurs when a user enters an input field, and here it is used to set the status bar message in order to use it as a help field.

3. In the HTML Attributes event of PercentB write:

    onFocus="window.status="window.status=
    'Insert Percentage A and look to Percentage B' 

    onBlur="if(this.form.PercentB.value<=100)
    {
        this.form.PercentA.value=100 - this.form.PercentB.value;
    }
    else
    {
        alert('Invalid Percentage B!');
    };
window.status='';  

286 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
This JavaScript code perfectly mirrors that in step 2, so it doesn’t need further explanation.

**Caution** The code in steps 2 and 3, must be converted in order to replace all “ with \” and all ’ with \’. To replace those characters use a text editor’s “find and replace” tool. We have shown the original JavaScript code here to make it easier to read.

An alternative to onBlur is to use **onChange** as a triggering event for the validation. This event occurs only when the value of a field is actually different from the previous value.

**Note** There are many solutions for the problem solved by this example. Another possible way is, instead of doing the validation inside each field it is possible to store a unique JavaScript function for the validation of the entire form, in the Header of the Web page and then call it using the following syntax for an event like onChange or onBlur:

```javascript
onBlur="validateForm()"
```

In the same way, you can call a procedure defined in the Header of the form that refreshes fields on a form.

If you prefer to use the validation and translation formulas of the Domino form, you can use JavaScript to force the execution of these formulas every time a value is changed in a field. To do this:

- Put the following in the **HTML Attributes** of a triggering field:
  ```javascript
  <onChange="this.form.submit()">
  ```

- Put this formula in the **@Return** field in order to go back immediately to the form after ‘refreshing’:
  ```javascript
 ="/"+@Subset(@DbName;-1) + "/defaultView/
  @Text(@DocumentUniqueID)+"?EditDocument"
  ```

In this way you can also have Hide-when formulas working every time the value changes in that field.

Keep in mind that this is only a simulation of field refresh, since the document is actually saved.

To avoid this you can use a button that perform the formula: @Command([ViewRefreshFields]). In this way you have form refresh without document saved but the user must click the button to see the results.
Example 2: Setting Field Values

The field in the picture below can be reset to today’s date value, by clicking the button. Though this example is very simple, it illustrates a function which could be very useful — having a button that resets all the fields on a form to their original values. Again, this action is performed without calling the server.

This panel shows the design to achieve this function:

To create this sample:

1. Define a button using this HTML:

   ```html
   <INPUT TYPE="button" VALUE="Set Default Date"
   onClick="this.form.Date.value = this.form.DefaultDate.value;"/>
   ```

   The action behind the button assigns the value of the ‘DefaultDate’ field to the ‘Date’ field.

2. Create an Editable Date field and call it ‘Date’

3. Create an Editable Date field and call it ‘DefaultDate’ and put in its default formula: @Today.
4. In HTML Attributes of ‘DefaultDate’ write:

   "TYPE="Hidden""

Caution  This field doesn’t appear on the Web but it is defined on the Web form so it really exists on the browser. If you use the Domino hide-when formulas instead of TYPE="Hidden", the field would not be sent to the browser so the sample would not work.

Example 3: Using JavaScript with Keyword Fields

In this example the input field is completed automatically when the user selects a value from the list. Since the selected field is separated by the keyword list, the user may also introduce a value that is not on the list.

Here is the solution adopted:

To create this sample:

1. Create **Choices** as a keyword field and in its HTML Attributes event insert:

   "SIZE=4

   - **SIZE=4** simply sets the size of the list frame to 4.
   - **onChange**, triggers the assignment of the selected value to the field named **Country**. **SelectedIndex** is the number of the selected item of the list.

2. Create **Country** as an Editable Text field.
Example 4: Changing an Image on Mouse-Over or Mouse-Out

An image displayed can be changed when the mouse floats in and out of it. The following example alternates two different logos depending on the position of the mouse pointer.

To build the sample:

1. Create a form with a hidden $HTMLHead field and put the following JavaScript as the value formula for that field:

```javascript
<SCRIPT LANGUAGE="JavaScript">

logoIBM=new Image(100,40);

logoIBM.src = "/ChilesDirect/ChilePepperSite.nsf/Banners/IBM/$file/IBM.gif";

logoLotus=new Image(100,40);

logoLotus.src= "/ChilesDirect/ChilePepperSite.nsf/Banners/Domino/$file/DominoSquareLogo.gif";

function showLogo(logoName)
{
    logo = eval(logoName+".src");
    document.images["Banner"].src=logo
}
</SCRIPT>

Caution  Remember to replace double quotes with " before using it in the field formula.

This JavaScript will be inside the Header of the Web page so that all objects of this JavaScript are allocated before other elements on the Web page.

- `logoIBM` and `logoLotus` are the definitions of two images.
- `logoIBM.src` and `logoLotus.src` store the URL of the images so you have to replace their values with the URL of your sample images.
- `showLogo()` is a function that replaces the content of the “Banner” image with that of one of the two above defined; the selection of the image depends on the value of `logoName` which is a parameter (a string) passed to the function.
2. Add to the form the following HTML code, using pass-thru HTML style:

```html
<a href='http://www.ibm.com'
   onMouseOver="showLogo('logoIBM'); return true;"
   onMouseOut="showLogo('logoLotus')">
   <IMG NAME="Banner"
        SRC="/ChilePepperSite.nsf/Banners/Entrevision/$file/Entrevision.gif">
</a>
```

Explanation:

- `onMouseOver` is the event that triggers the function call `showLogo('logoIBM')`.
- `onMouseOut` is the event that triggers the function call `showLogo('logoLotus')`.
- `'Banner'` is the name of the image that is replaced each time the mouse enters or exits its area.

**Example 5: Updating Frames Using JavaScript**

This short sample can be used when you need to change the content of two frames at the same time.

1. Insert this code in the $$HTMLHead field.

   ```html
   <SCRIPT Language=JavaSript>
   function changeFramesContent(URL1,URL2)
   {
       top.Frame1.location=URL1;
       top.Frame2.location=URL2;
   }
   </SCRIPT>
   ```

   We have names the two frames Frame1 and Frame2.

2. Use the following button to update the frames:

   ```html
   <INPUT TYPE="button" VALUE="Change Frames"
           onClick="changeFramesContent(URL1,URL2);">"
   ```

   The form containing this button and $$HTMLHead can initially be displayed in a third frame. See the Frames section in Chapter 6: Using HTML in Domino Design Elements for more information.
Cookies and Domino

This section will provide information on:

- What is a cookie?
- Cookie syntax.
- Cookie security.
- An example.

What is a Cookie?
Cookies enable a server to send information to a client and then store it locally on the client’s file system.

This is useful when you need to simulate a session between a browser and a client; you can store some status variables on the client so that, on the next connection, the server can read them and re-establish the ‘session’ starting from the last status.

As an example you can store the users’ preferences about the layout of some Web pages in a cookie variable so that the Web pages are customized for each user, every time the user connects to the site.

Cookie Syntax
Cookies are simply text strings of the following format:

"CookieName=Value; expires=Date; path=Path;
    domain=DomainName; secure;"

where:

- **CookieName=Value** is the name and value of the variable sent to the browser.
- **expires=Date** contains a string, for example: “01-Apr-1998 08:00:00 GMT”. If omitted, the cookie is deleted at the end of the browser session.
- **path=Path** and **domain=DomainName** if the URL doesn’t match path and domain, the cookie is not sent to or retrieved from the browser.
- **secure** this is optional and determines if the cookie must be sent exclusively on a secure connection or not.
**Cookie Security**
The following list summarizes the security features for cookies:

- A cookie cannot be an executable file so it cannot alter the file system.
- A cookie variable can be read and altered only by the server that has created it.
- An expired cookie cannot be used.
- A cookie can also be sent and retrieved on a secure connection.

Since cookies are stored on the user’s machine, everyone who has access to that machine can alter the cookies’ information. Therefore, it is better to use them to store non-sensitive information only.

*Note*  The way that you use cookies should be similar to how a Notes developer would use *Environment Variables*.

An alternative to using cookies is to use Profile Documents. These special documents can store user information and, since they are Domino objects, the level of security is much higher than that of cookies. Profile documents are centrally located on the server so that if you are building an application for large numbers of users, it might be too expensive in terms of performance, space etc. to store so many user profiles on the server. For many intranet applications, profile documents provide a good method for keeping track of client information.

*Note*  To see an example of profile documents go to Chapter 7: Domino Agents.

**Example**
The following example comes from the Domino 1.5 User’s Guide.

The program uses JavaScript and cookies to load a page into the browser and save a cookie called “*Cookie_Man*” in the user’s cookie file. When Domino loads the page, it includes the number of times that the site has been visited before.

To create this sample, prepare a form with a $$HTMLHead field, then:

1. Write in the value formula of a hidden $$HTMLHead field the following JavaScript, but remember to use a \ before double quotes.

   `<script>
   
   cookieName = "Cookie_Man";
   
   function doCookie()
   
   {
       if(document.cookie)
   
   Chapter 10: Java Applets, Java Agents, and JavaScript  293`
{
    index = document.cookie.indexOf(cookieName);
} else
{
    index = -1;
}

if (index == -1)
{
    document.cookie = cookieName+"=1;
    expires=Tuesday, 01-Apr-1998 08:00:00 GMT";
} else
{
    countbegin =
        (document.cookie.indexOf("=", index) + 1);
    countend =
        document.cookie.indexOf(";", index);
    if (countend == -1)
    {
        countend = document.cookie.length;
    }
    count = eval(document.cookie.substring(countbegin,
        countend)) + 1;
    document.cookie=cookieName+"="+count+";
    expires=Tuesday, 01-Apr-1998 08:00:00 GMT";
}
}

function getTimes()
{
    if(document.cookie)
    {

294 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
index = document.cookie.indexOf(cookieName);
if (index != -1)
{
    countbegin =
    (document.cookie.indexOf(“=”, index) + 1);
    countend = document.cookie.indexOf(“;”, index);
    if (countend == -1)
    {
        countend = document.cookie.length;
    }
    count =
    document.cookie.substring(countbegin, countend);
    if (count == 1)
    {
        return (“<b>”+count+”</b> time before.”);
    } else
    {
        return (“<b>”+count+”</b> times before.”);
    }
}
return (“<b>0</b> times before.”);
</script>

Explanation:
- **cookieName** is the name of the cookie; its value is “Cookie_Man”.
- **doCookie()** updates the cookie incrementing its value. The cookie string has a format like “cookieName=Value; expirationParameters” so some string operations are done to extract the value, increment it and then rebuild the cookie.
- **getTimes()** reads the cookie value and returns a message string with the cookie’s value.
2. Write in the HTML Attributes of the form:
   "onLoad="doCookie()"

   The procedure doCookie() is called every time the form is opened.

3. Write on the form, using pass-thru HTML style, the following script:

   <script>
   document.write("You have visited this site "+ getTimes());
   </script>

   This JavaScript write calls the getTimes() function and writes a message containing the number of times the user has opened this form.

---

Summary

Java is a universal language on the Web, and is included in a rapidly growing number of products.

From this chapter you will have learned that:

- At present, Domino is not a Java programming environment. Java programming should be performed with other special authoring tools, like Lotus BeanMachine or IBM VisualAge for Java.

- Java Notes classes are an extension to the LSX architecture; they provide a new method for integrating Java with Notes back-end data objects.

- JavaScript is a different language from Java and is especially useful for adding some client-side logic to Web pages. The advantage of using JavaScript is mainly in the improved dynamic behavior of Web pages.
Chapter 11
Accessing External Data from a Domino Application

This chapter describes tools and techniques that can be used to access data resources from a Domino application. Since any Domino application can then be made available on the Web, the facility to integrate Domino with external data makes for a powerful tool in enhancing the Web applications created by Domino.

The following areas are covered in this chapter:

- LS:DO (LotusScript:Data Object).
- @DBCommand, @DBLookup, @DBColumn.

The above features are based on Open Database Connectivity (ODBC) technology.

When you are developing your Notes application, you need to decide what your data access needs are and which products best meet those needs. Each product has different functionality as well as performance and programmability. When you have completed this chapter, you should know when to use which tool, and how to use it.

Data Resource Access

When you develop a Notes application, you often need to implement data integration between Notes and other data resources such as RDBMS, spreadsheet data, and ASCII delimited text files. In enterprise Notes application development, this becomes even more important as you will surely have to integrate legacy database resources in your design.

Database Access Facilities

The following tools enable Notes applications to connect to data resources through ODBC or native database access.

1. LS:DO (LotusScript:Data Object)
   This is a LotusScript Extension (LSX) which provides additional LotusScript classes for accessing other data resources via ODBC.
2. \texttt{@DBCommand}, \texttt{@DBLookup}, \texttt{@DBColumn} using ODBC

These are \texttt{@functions} for ODBC data access. The functions \texttt{@DBLookup} and \texttt{@DBColumn} are frequently used to access Notes databases as well as ODBC-compliant databases.

The choice of tool to use in a particular situation depends on the functionality and performance of the particular tool. The following table summarizes the characteristic differences between the tools covered in this chapter:

<table>
<thead>
<tr>
<th></th>
<th>LS:DO</th>
<th>\texttt{@DBLookup}</th>
<th>\texttt{@DBCommand}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on ODBC</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Available in LotusScript</td>
<td>X</td>
<td>*1</td>
<td>*1</td>
</tr>
<tr>
<td>Has a Class</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Only</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>64Kb Data Limit</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

*1 Technically, LotusScript can perform \texttt{@Functions} under the Evaluate function.

Furthermore, we must consider some ease of use versus programming functionality and flexibility. For example, \texttt{@Functions} are useful to retrieve small data on the fly without complex sequences, but they are limited in the number of ways to access data.

LS:DO can be used in more complex situations with much more flexibility from a programming perspective, for example, result set handling to read and update records queried by SQL. LS:DO is also easy to use if you are familiar with LotusScript. The figure below, may help in deciding which tool to use in a particular situation.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{relationship_diagram.png}
\caption{Relationship among Data Access Features}
\end{figure}

298 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
What Is ODBC?

The ODBC (Open Database Connectivity) standard is a set of functions established by Microsoft to access Relational Database Management Systems like Oracle, DB/2, Informix and others. There are two software components required to use ODBC:

1. **ODBC Driver Manager** - is a set of APIs in the ODBC dynamic link library. Those APIs are called by client programs like LS:DO, NotesSQL, and so on, in order to access an RDBMS via ODBC.

2. **RDBMS ODBC driver** - is the driver for specific RDBMSs like NotesSQL, DB2, Oracle etc. The ODBC driver allows you to issue any SQL statements in DDL (Data Definition Language), DCL (Data Control Language) and DML (Data Manipulation Language) using SQLExecute or SQLExecDirect with the ODBC API. In addition, other ODBC Drivers enable you to get information about columns attributes, index, privileges of column, drivers, foreign keys of tables, and other RDBMS entities.

Using ODBC Connections

You can use ODBC:

1. APIs in your C, C++, Basic, LotusScript, or any other programming language programs.
   
   **Note** The programming language you use must support calls to a DLL (Dynamic Link Library) as all of the ODBC functions are in the ODBC DLL.

2. Compliant high level tools such as LS:DO, Lotus Spreadsheet Component in LotusScript, and Data Access Object in Visual Basic.

ODBC Access Flow

The process by which a program accesses a database through ODBC is shown below:

1. The program makes a call to the ODBC API.
2. The ODBC driver manager parses the requested command.
3. The ODBC driver manager decides which ODBC driver is required according to database resource information registered in advance through the operating system.
4. The requested command is passed to the specific ODBC driver for the database being accessed.
5. The ODBC driver composes a series of commands for the particular RDBMS and sends them to the RDBMS.
6. The results, if available, are sent to the calling routine.

7. There are many ODBC drivers. Usually they are provided by RDBMS vendors but others come from independent software vendors like InterSolv or Visigenic. The following figure shows conceptually how LS:DO makes connection paths to RDBMSs as an example. It also shows other ODBC drivers which are capable of accessing ASCII delimited text files, spreadsheets, and other types of data resources other than RDBMS.
Note  There are two types of ODBC driver managers in the Windows environment: the 16-bit ODBC driver manager and the 32-bit ODBC driver manager. You must ensure that the one you use matches the application environment you are in. For example, when you use the 32-bit Windows version of Lotus Notes, you need a 32-bit ODBC driver manager and a 32-bit RDBMS driver.

LotusScript:DataObject (LS:DO)

What is LS:DO?

The LotusScript:Data Object (LS:DO) provides full read and write access to external ODBC data sources using the complete control and flexibility of a structured programming language: LotusScript.

The LS:DO consists of a set of three classes: ODBCConnection, ODBCQuery, ODBCResultSet. These classes come complete with a powerful set of properties and methods and full SQL capabilities. Yet at the same time, the LS:DO is easy to learn and use because its design is consistent with LotusScript’s BASIC syntax and other LotusScript Notes classes.

Concepts

The LS:DO is available on both the Notes client and the Domino server. LS:DO is excellent for real-time data access from any LotusScript event in Notes, such as clicking a button, exiting a field, or opening a document. LS:DO real-time data access is the best choice for:

- Optimizing data entry.
  - On-the-fly lookups.
  - Immediate updates.
  - Input validation.
  - Avoiding duplicate entries.
- Mobile user queries and updates.
Optimizing Data Entry

Many designers use Notes as the data entry point for an application, which may synchronize that data with a DBMS or use the DBMS for long-term data storage and archiving. The LotusScript Data Object can provide the following functionality on the fly:

- **On-the-fly lookup**
  Once a user enters a customer name and exits the field with the TAB key or a mouse click, LotusScript code can immediately perform an SQL query to one or several external back ends, retrieve the customer record matching that name, and fill the remaining fields in the form, such as address, city, phone, and contact name.

- **Immediate updates**
  LotusScript gives you the flexibility to update the information in the relational DBMS the moment the user saves a new document in Notes or in batches at scheduled intervals. When another document is created in Notes, you can be sure that document will access the most current information in the relational DBMS.

- **Input validation**
  Is the right salesperson assigned to that customer in the Notes form? Is the regular salesperson for that region currently overloaded with assignments, indicating that a backup person should be assigned to the task? The LotusScript Data Object can retrieve that information from the DBMS that indicates these conditions, and LotusScript’s fully structured programming constructs enable you to evaluate that data and act accordingly.

- **Avoiding duplicate entries**
  Once a user enters a customer’s name, the LS:DO can query the back end for variations on that customer name, for example, to ensure that the same customer is not entered with an “Inc.” as opposed to a “Co.” in the DBMS.

Mobile User Queries and Updates

One of the most exciting results of the intersection between DBMSs and Notes is that the mobile Notes user can take their access to the DBMS with them on the road. For example, when sales representatives are on the road, they often find themselves with last-minute opportunities to visit customers in different cities. If they are on the road and if that customer information is contained in the mainframe DBMS, they are forced to call someone in the office, and ask them to look up the information, which is out of the question from a hotel room outside business hours.
The LotusScript Data Object’s ability to run on Domino servers as well as Notes clients, coupled with Notes native replication capabilities, solves the problem. With an integrated Notes/DBMS application, a user can do the following:

1. Compose a query request within an application on their mobile Notes client, such as “What are the customer contacts and activity in this city?”
2. Replicate the query to the Notes server, where a waiting LS:DO agent sees the new document, authenticates and performs the query, stores the results in that document, and saves it.
3. Replicate the query results back to their laptop in moments, even during the same dial-up connection if they choose, for analysis and review.

Architecture

In addition to allowing users to issue SQL statements to relational DBMSs, the LS:DO also offers data manipulation capabilities. The LS:DO supports and manages result sets as well as provides an interface for directly using SQL when appropriate. The result set management takes the form of caching result sets, supporting navigation through the result set, and managing individual row updates regardless of the underlying driver’s cursor or ODBC conformance capabilities.

The following diagram is a schematic representation of the components in the LS:DO framework that allow a Notes application to access a database:
When to Use LS:DO

LS:DO is best suited to handle the following situations:

- **LotusScript programming environment**
  If you develop an application with the LotusScript environment, you can easily utilize ODBC access through LS:DO classes.

- **Low-volume data transfer**
  LS:DO is more suited for low volume access to data resources. From a performance perspective LS:DO is not well suited to moving large volumes of data.

- **Easy data access**
  When your application needs to both read and update data in an RDBMS, LS:DO is an easier way than the ODBC API or the @DBCommand because of the classes allowing you to work with result sets.

- **Real-time direct access**
  LS:DO is integrated directly in a Notes application and so on.

Differences Between LS:DO and ODBC

LS:DO is a high-level abstraction of the ODBC feature, which enables you to make more complicated operations toward RDBMS, but requires more detailed knowledge about the ODBC architecture and ODBC APIs. Let’s look at three aspects of both methods:

- **Programming Environment.**
- **Functionality.**
- **Performance.**

**Programming Environment**

Calling ODBC APIs requires passing many arguments. You have to be careful with the different argument types. A wrong argument type may cause unexpected severe errors and may make your system unstable.

The LS:DO is more intuitive and at a higher level of abstraction. Also, the LotusScript development environment checks syntax on the fly.

LS:DO is available only in LotusScript and some development environments which are compliant with OLE clients, such as Visual Basic. LS:DO is one of the LotusScript Class Libraries (LSXs). This enables you to benefit from the object-oriented and event-driven programming environment provided by the Notes Integrated Development Environment.

304 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
Functionality

Through LS:DO classes, you can update data in a result set, which is then automatically reflected to the original table. It is much easier to update data using LS:DO methods than using an SQL statement.

ODBC functions are calls from C or C++ programs to the Dynamic Link Library. There are three conformance levels:

- Core Level
- Extension Level 1 and
- Extension Level 2.

There are more than 50 functions depending on the version number of the ODBC driver manager and the ODBC driver.

The following table conceptually shows which method in LS:DO calls which ODBC APIs. Each LS:DO method corresponds to a combination of some ODBC APIs:

<table>
<thead>
<tr>
<th>Functions</th>
<th>LS:DO Method</th>
<th>ODBC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>List all data sources</td>
<td>Connection.ListDataSources</td>
<td>SQLAllocEnv, SQLDataSources, SQLFreeEnv</td>
</tr>
<tr>
<td>registered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Establish a connection to DB</td>
<td>Connection.ConnectTo</td>
<td>SQLAllocConnect, SQLBrowseConnect, SQLFreeConnect, SQLAllocConnect, SQLConnect</td>
</tr>
<tr>
<td>List all tables in a database</td>
<td>Connection.ListTables</td>
<td>SQLAllocStmt, SQLTables, SQLFetch, SQLGetData</td>
</tr>
<tr>
<td>Execute an SQL</td>
<td>ResultSet.Execute</td>
<td>SQLSetStmtOption, SQLExecDirect</td>
</tr>
<tr>
<td>Fetch data from a result set</td>
<td>ResultSet.GetValue</td>
<td>SQLNumResultCols, SQLColAttributes, SQLFetch, SQLGetData</td>
</tr>
</tbody>
</table>
**Performance**

ODBC provides better performance than LS:DO in some cases because the C++ program can directly access the ODBC driver manager, whereas LS:DO has some overhead due to the language architecture. Although LS:DO can make it easy to retrieve and update records in a result set, ODBC API calls are more powerful allowing the use of more complex and efficient record data handling using fetch and retrieve of records, parameterized SQL, and cursor features.

**Software Requirements**

The software requirements are:

- **ODBC driver manager 2.0 or later** - You need to install the appropriate ODBC driver manager as required by your operating system and by the applications that use the ODBC features. The 32bit ODBC driver manager comes with Visual Basic 4.0, Lotus SmartSuite 97, Office95, Visual C++ 4.0, and others. The 16bit ODBC driver manager comes with Windows 3.1 and Windows 95.

- **ODBC drivers for specific RDBMSs** - For example, if you create a program to access the Oracle DB server, you must install an Oracle ODBC driver which corresponds to the ODBC driver manager type already installed. There are many drivers provided by many software companies for RDBMSs and other data resources as well.

Some of them are listed in the following table:

<table>
<thead>
<tr>
<th></th>
<th>Microsoft</th>
<th>Intersolv</th>
<th>Visigenic</th>
<th>Lotus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1-2-3</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>dBASE</td>
<td>X</td>
<td></td>
<td></td>
<td>X*1</td>
</tr>
<tr>
<td>Excel</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FoxPro</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Informix</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ingres</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL Server</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X*1</td>
</tr>
<tr>
<td>Paradox</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sybase</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Text File</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X*1</td>
</tr>
</tbody>
</table>

**Note**  
*1 This driver is bundled in Lotus SmartSuite.
Registering ODBC Data Sources

To register ODBC data sources, follow these steps. Our example is based on an Oracle DB Server connection for Windows 95. The basic operations are practically the same on the other platforms.

1. Double-click the ODBC driver manager icon in the Control Panel.

   ![32-bit Administrator]

   **Note** The icon image and icon title displayed depends on the driver manager you installed on your system.

2. The Data Sources dialog box displays. You can see all data sources previously defined. To add a new data source, click the Add... button.
3. The Add Data Source dialog box displays. You can see all the ODBC drivers installed on your PC. Select the ODBC driver for your application and click OK. In our example, we selected the Oracle 72 ODBC driver:

4. If you select the Oracle ODBC driver, you’ll need to fill in the appropriate information as follows. This dialog box will vary depending on the driver you choose.

The Data Source Name you enter will be the one you specify in your programs, such as the ConnectTo method in LS:DO, whenever you connect to the Oracle Database.

The Description field is just an explanation for this data source.

The SQL*Net Connect String is important to establish the connection. In our case, the “T” means that the TCP/IP protocol is used during communication. The rest of the string “OracleITSO” is a TCP/IP hostname for the Oracle DB server on Windows NT. Optionally, you can specify the Oracle instance ID here. For a full description and different connect string examples, click on the Help button of the Oracle7 ODBC Setup dialog box.
USELSX Statement to Enable LS:DO

The following statement must be specified in the Define (Globals) Event (Declarations) within Lotus Notes.

\texttt{Uselsx "*LSXODBC"}

\textbf{Note} The leading "*" tells LotusScript to use the class registry to look up the path of the LS:DO dynamic library being loaded. This is a platform-independent way of loading LS:DO since each operating system uses different methods.

Mapping Data Types Between RDB and Notes DB

The following diagram shows the data type mapping between an Oracle database and a Notes database through the LS:DO:

How to Trace and Debug LS:DO

In this section, we will briefly touch on a few ways to debug and trace applications that employ the LS:DO.

The structure for a connection between a Notes application using the LS:DO and the target RDBMS is the same as in all ODBC-compliant systems.
The following diagram shows each connection layer and the respective component. If a connection cannot be established for some reason, the configuration of each of the components must be verified one by one. The appropriate debugging or tracing tool is indicated by the caption circle for each component.

**Using the ODBC Trace Option**

Trace information issued by the ODBC API can be gathered using the trace function in the ODBC Administrator program. To do this, follow these steps:

1. Run the ODBC Administrator program and click Options....
2. Select the Trace ODBC Calls check box. Click OK, and leave the ODBC Administrator program.

Trace descriptions of the ODBC API calls in your program(s) using LS:DO are saved in the “C:\sql.log” text file as a default. You may change the log file name by clicking the Select File button in the dialog box above. An example of some of the trace output is shown below:

```sql
SQLAllocConnect (henv004993F0, phdbc00483E0C);
SQLConnect (hdbc00483E0C, "dBaseDB1", -3, ",", -3, ",", -3);
SQLGetInfo (hdbc00483E0C, 11, rgbInfoValue, 4, pcbInfoValue);
SQLGetInfo (hdbc00483E0C, 21, rgbInfoValue, 4, pcbInfoValue);
SQLAllocStmt (hdbc00483E0C, phstmt00488110);
```

**Using the ODBC Test Tool**

If you are using the Microsoft ODBC driver SDK, you can use the ODBC Test tool which allows you to issue ODBC API function calls. In the following example, we will test whether the connection to the database is configured properly and, if so, try other calls to debug each of the layers.

1. Double-click the following icon to start the ODBC Test program.

```
Odbcte32.exe
```
2. Choose Connect - Full Connect from the menu bar.

3. You are prompted for the connection specifics. From the list box, choose the Data Source you want to test the connection to. These are the data sources you specified with the ODBC driver manager. In our example, we chose the Millennia Oracle7 instance data source. If security is enforced, you must specify the User ID and/or a password.
4. You will see the results of the connection. If successful, you can select from the other ODBC API function calls from the menu to test the functionality of the connection and the results of the calls:

![ODBC Test (32-bit)](image)

**LS:DO Class Library**

The classes that make up the LotusScript:Data Object provide you with the following benefits:

- **Connection sharing**
  Connections are cached to avoid the added overhead of establishing a connection. In addition, since it is defined as an independent object, one connection object can be used by multiple LotusScript SQL calls.

- **Multiple query and result sets**
  You can define multiple query objects to generate multiple result set objects which can all be executed against the same connection, and manipulated from the same script.

- **Bi-directional scrolling over result set**
  The ODBCResultSet object provides a scrolling cursor with methods for navigating to the next, previous, first and last rows.

- **Result set search**
  The LocateRow method of the ODBCResultSet object provides the ability to search for specific rows within the result set based on specified criteria. This search capability executes faster than multiple queries or comparing values from multiple rows in LotusScript.
• **Cached results**
  The query result in the ODBCResultSet object is optionally cached in memory (default setting), so it can be later accessed by other events in the form, increasing performance and reducing DBMS connection time. In addition, the cached result set gives you the ability to later locate records using LocateRow.

• **Update services**
  Updates to back-end DBMSs through a generic ODBC interface are limited to SQL statements, where the user must ensure that the row to be updated contains a unique record reference or can otherwise be uniquely accessed through a cursor. LS:DO extends this capability by permitting individual items in a result set to be modified without use of an SQL statement using the SetValue method. These changes are then updated to the back-end database all at once.

• **Driver transparency**
  Although different vendors’ ODBC drivers support varying conformance levels, the LotusScript:Data Object assesses these differences and often provides the same level of behavior across all drivers and databases. The developer does not have to write separate scripts for separate drivers.

The following figure represents the manner in which a LotusScript program would use each class in an application access in a database.
Relationship Among Classes
The three classes in the LS:DO are tightly related to one another as shown in the following diagram.

Event Handling
If needed, you can create event handling subroutines for some ODBC methods. An event handling subroutine you create is called according to the behavior of an appropriate ODBC method, after the On Event statement is issued.

In the following example, an event handler, named presub1 is called, before the ListDataSources method is called.

1. On Event statement
   ```vba
   Dim connection As New ODBCConnection
   On Event BeforeListDataSources From connection Call presub1
   ```

2. Event handler
   ```vba
   Sub presub1(Source As ODBCConnection)
       '** Write your event handling script here
   End Sub
   ```

   **Note** Your event handler must be in the scope where the event occurs.

ODBCConnection Class
The ODBCConnection class allows you to establish a connection. It also allows you to access some database catalog information, such as data source lists, table lists, procedures lists and so on.
**Property**
The following table shows the properties of the ODBCConnection Class:

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Read/Write</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSourceName</td>
<td>String</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>DisconnectTimeOut</td>
<td>Integer</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>Exclusive</td>
<td>Boolean</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>IsConnected</td>
<td>Boolean</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>IsSupported(option)</td>
<td>Boolean</td>
<td>R</td>
<td>option:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB_SUPP_ASYNCHRONOUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB_SUPP_CURSORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB_SUPP_PROCEDURES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB_SUPP_READONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB_SUPP_SILENTMODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB_SUPP_TRANSACTIONS</td>
</tr>
<tr>
<td>IsTimedOut</td>
<td>Boolean</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>SilentMode</td>
<td>Boolean</td>
<td>R/W</td>
<td></td>
</tr>
</tbody>
</table>

**Note**  
Boolean is not a pre-defined data type in LotusScript. But you can use a constant value (TRUE and FALSE) as a Boolean data type.

**Method**
The following table shows the ODBCConnection methods with the corresponding arguments and events.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Error Constant</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectTo(source$ [, userID$, password$ ] )</td>
<td>Boolean</td>
<td>DBstsCANF</td>
<td>BeforeConnect</td>
<td>AfterConnect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsSVRQ</td>
<td></td>
<td>AfterConnectTo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsCCON</td>
<td></td>
<td>BeforeConnectTo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsACCS</td>
<td></td>
<td>AfterConnectTo</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Boolean</td>
<td>DBstsNCON</td>
<td>BeforeDisconnect</td>
<td>AfterDisconnect</td>
</tr>
<tr>
<td>ExecProcedure(name$, arg$ )</td>
<td>Boolean</td>
<td>DBstsNCON</td>
<td>BeforeExecProcedure</td>
<td>AfterExecProcedure</td>
</tr>
<tr>
<td>GetError</td>
<td>Constant</td>
<td>*1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetErrorMessage( [ error% ] )</td>
<td>error%:</td>
<td>DB_LASTERROR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>or Constants *1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued
<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value</th>
<th>Error Constant</th>
<th>Event Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetExtendedErrorMessage( [error% ])</td>
<td>error%:</td>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB_LASTERROR or Constants *1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetRegistrationInfo(source$)</td>
<td>String</td>
<td>DBstsCANF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ListDataSources</td>
<td>Array of String</td>
<td></td>
<td>BeforeListDataSources</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AfterListDataSources</td>
<td></td>
</tr>
<tr>
<td>ListFields( [tableName$ ] )</td>
<td>Array of String</td>
<td>DBstsNCON</td>
<td>BeforeListFields</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsNCOL</td>
<td>AfterListFields</td>
<td></td>
</tr>
<tr>
<td>ListProcedures( [source$ [, userID$, password$ ] )</td>
<td>Array of String</td>
<td>DBstsNCON</td>
<td>BeforeListProcedures</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsACCS</td>
<td>AfterListProcedures</td>
<td></td>
</tr>
<tr>
<td>ListTables( [source$ [, userID$, password$ ] )</td>
<td>Array of String</td>
<td>DBstsNCON</td>
<td>BeforeListTables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsACCS</td>
<td>AfterListTables</td>
<td></td>
</tr>
</tbody>
</table>

*1 Error number list is shown in ODBCResultSet section.

**Note** Most of the pre-defined data types in LotusScript are represented by the following suffix types:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currency</td>
<td>@</td>
</tr>
<tr>
<td>Double</td>
<td>#</td>
</tr>
<tr>
<td>Integer</td>
<td>%</td>
</tr>
<tr>
<td>Long</td>
<td>&amp;</td>
</tr>
<tr>
<td>Single</td>
<td>!</td>
</tr>
<tr>
<td>String</td>
<td>$</td>
</tr>
</tbody>
</table>

**Sample Uses of the ODBCConnection Class:**

1. To get a data source list registered by the ODBC administrator, use the ListDataSources method of the ODBCConnection class.

```vbnet
Dim con As New ODBCConnection
Dim dl As Variant
dl = con.ListDataSources
'** Keywords is a field name in which a data source list is saved
keyDoc.Keywords = dl
```
2. To get a table list owned by a database, use the ListTables method of the ODBCConnection class.

   ```vba
   Dim con As New ODBCConnection
   Dim tl As Variant
   '** sampleDB1 is a database name registered in this example
   tl = con.ListTables("sampleDB1")
   '** Keywords is a field name in which a table list is saved
   keyDoc.Keywords = tl
   ```

   **Note** When the ListTables method is issued, the SQLConnect ODBC API is called in LS:DO before getting a table list. So you don’t need to execute the ConnectTo method.

3. To get a column name list owned by a table, use the ListFields method of the ODBCConnection class.

   ```vba
   Dim con As New ODBCConnection
   Dim Clist As Variant
   Dim status As Variant
   '** sampleDB1 is a database name registered in this example
   status = con.ConnectTo("sampleDB1")
   '** courses is a table name in the sampleDB1 database
   CList = con.ListFields("courses")
   '** Keywords is a field name in which a column list is saved
   keyDoc.Keywords = Clist
   ```

**ODBCQuery Class**

The ODBCQuery class is used to hold the ODBCConnection object in which a connection is established, and to hold an SQL statement you want to use to perform the inquiry. The SQL statement is parsed through the ODBC driver which your application requires.

**Property**

The three properties of this class are shown below:

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>ODBCConnection Object</td>
<td>W</td>
</tr>
<tr>
<td>QueryExecuteTimeOut</td>
<td>Integer</td>
<td>R/W</td>
</tr>
<tr>
<td>SQL</td>
<td>String</td>
<td>R/W</td>
</tr>
</tbody>
</table>
Method

The `ODBCQuery` class provides the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetError</td>
<td>Constant</td>
<td></td>
</tr>
<tr>
<td>GetErrorMessage( [ error% ] )</td>
<td>error%: DB_LASTERROR or Constants *1</td>
<td>String</td>
</tr>
<tr>
<td>GetExtendedErrorMessage( [ error% ] )</td>
<td>error%: DB_LASTERROR or Constants *1</td>
<td>String</td>
</tr>
</tbody>
</table>

*1 Error number list is shown in ODBCResultSet section.

Sample Uses of the ODBCQuery Class:

This sample shows a sample execution of an SQL statement using the Execute method in ODBCResultSet. The following steps are needed before executing the Execute method. The Connection method and the SQL property in ODBCResultSet class are also used in this example.

```vba
Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
'** sampleDB1 is a database name registered in this example
status = con.ConnectTo("sampleDB1")
Set qry.Connection = con
'** courses is a table name in the sampleDB1 database
qry.SQL = "select * from courses"
Set res.Query = qry
```

ODBCResultSet Class

The `ODBCResultSet` class has many functions used to handle records which are termed result sets. A result set holds the retrieved records of an SQL query which is specified with the `ODBCQuery` object.

Property

The following table shows the properties available with the `ODBCResultSet` class:

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>Boolean</td>
<td>R/W</td>
</tr>
<tr>
<td>AutoCommit</td>
<td>Boolean</td>
<td>R/W</td>
</tr>
<tr>
<td>CacheLimit</td>
<td>Integer</td>
<td>R/W</td>
</tr>
<tr>
<td>CommitOnDisconnect</td>
<td>Boolean</td>
<td>R/W</td>
</tr>
<tr>
<td>CurrentRow</td>
<td>Integer</td>
<td>R/W</td>
</tr>
<tr>
<td>FetchBatchSize</td>
<td>Integer</td>
<td>R/W</td>
</tr>
</tbody>
</table>
### Methods

The methods of the ODBCResultSet class can be categorized into the following areas:

- SQL execution and transaction control
- Result set row navigation and location
- Accessing column values
- Result set row modification operations
- Column attributes operations
- SQL parameter operations

The following tables show the methods based on the above categories.

#### SQL Execution and Transaction

These methods are used to issue an SQL statement and to commit or roll back a transaction.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Error Constant</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close(option)</td>
<td>Option:</td>
<td>Boolean</td>
<td>BeforeClose</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_CLOSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_ROLLBACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute( [ option ] )</td>
<td>Option:</td>
<td>Boolean</td>
<td>BeforeExecute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_CANCEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBstsODBC</td>
<td>AfterExecute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AsynchOperationComplete</td>
<td></td>
</tr>
<tr>
<td>Transactions(option)</td>
<td>Option:</td>
<td>Boolean</td>
<td>BeforeTransactions</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_ROLLBACK</td>
<td>AfterTransactions</td>
<td></td>
</tr>
</tbody>
</table>

---

320  Developing Web Applications Using Lotus Notes Designer for Domino 4.6
### Result Set Row Locating Operations

These methods are used to locate a cursor on a result set which is produced by the Execute method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Error Constant</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstRow</td>
<td>Boolean</td>
<td>DBstsINVR</td>
<td>BeforeFirstRow</td>
<td>AfterFirstRow, BeforeRowPositionChange, AfterRowPositionChange</td>
</tr>
<tr>
<td>LastRow</td>
<td>Boolean</td>
<td></td>
<td>BeforeLastRow</td>
<td>AfterLastRow, BeforeRowPositionChange, AfterRowPositionChange</td>
</tr>
<tr>
<td>LocateRow</td>
<td>column is Integer or String.</td>
<td>Boolean</td>
<td>DBstsCARR</td>
<td>BeforeLocateRow, AfterLocateRow, BeforeRowPositionChange, AfterRowPositionChange</td>
</tr>
<tr>
<td>NextRow</td>
<td>Boolean</td>
<td>DBstsINVR, DBstsEOFD</td>
<td>DBstsCNVR</td>
<td>BeforeLocateRow, AfterLocateRow, BeforeRowPositionChange, AfterRowPositionChange</td>
</tr>
</tbody>
</table>

### Accessing Column Value Operations

These methods are used to access specific column values and to check column properties.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Error Constant</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetValue</td>
<td>column is Integer or String.</td>
<td>Variant</td>
<td>DBstsINVC</td>
<td>BeforeGetValue, AfterGetValue</td>
</tr>
<tr>
<td>IsValueAltered</td>
<td>column is Integer or String.</td>
<td>Boolean</td>
<td>DBstsINVC</td>
<td></td>
</tr>
<tr>
<td>IsValueNull</td>
<td>column is Integer or String.</td>
<td>Boolean</td>
<td>DBstsINVC</td>
<td></td>
</tr>
<tr>
<td>SetValue</td>
<td>column is Integer or String.</td>
<td>Boolean</td>
<td>DBstsRDON, DBstsRDEL, DBstsINVC, DBstsCNVR, DBstsNODA</td>
<td>AfterSetValue, BeforeSetValue</td>
</tr>
</tbody>
</table>

Chapter 11: Accessing External Data from a Domino Application 321
### Result Set Row Modification Operations

These methods enable you to dynamically add and delete rows from within the result set. Furthermore, you can retrieve the row status and you can update the altered result set in the database.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Error Constant</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRow</td>
<td>Boolean</td>
<td>DBstsAHVR</td>
<td>DBstsRDON</td>
<td>BeforeAddRow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsNOEX</td>
<td>AfterAddRow</td>
</tr>
<tr>
<td>DeleteRow(tableName$)</td>
<td>Boolean</td>
<td>DBstsINVR</td>
<td>DBstsNUNQ</td>
<td>BeforeDeleteRow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsRCHG</td>
<td>AfterDeleteRow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsRDON</td>
<td>RowContentsChanged</td>
</tr>
<tr>
<td>GetRowStatus</td>
<td>DB_UNCHANGED</td>
<td>DBstsNODA</td>
<td></td>
<td>TransactionsPending</td>
</tr>
<tr>
<td>RefreshRow</td>
<td>Boolean</td>
<td>DBstsNUNQ</td>
<td>DBstsINVR</td>
<td>BeforeRefreshRow</td>
</tr>
<tr>
<td>UpdateRow</td>
<td>Boolean</td>
<td>DBstsRDON</td>
<td>DBstsRDEL</td>
<td>BeforeUpdateRow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsCXIN</td>
<td>AfterUpdateRow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsNUNQ</td>
<td>TransactionsPending</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsRCHG</td>
<td>RowContentsChanged</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsRUNC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBstsUPDB</td>
<td></td>
</tr>
</tbody>
</table>

### Column Attributes Operations

These methods allow you to access information about the column attributes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Error Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldExpectedDataType (column [, dataType ] )</td>
<td>column is Integer or String.</td>
<td>DB_TYPEUNDEFINED</td>
<td>DBstsINVC</td>
</tr>
<tr>
<td></td>
<td>dataType:</td>
<td></td>
<td>DB_CHAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_TYPEUNDEFINED</td>
<td>DB_SHORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_CHAR</td>
<td>DB_LONG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_SHORT</td>
<td>DB_DOUBLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_LONG</td>
<td>DB_DATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_DOUBLE</td>
<td>DB_TIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_DATE</td>
<td>DB_BINARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_TIME</td>
<td>DB_BOOL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_BINARY</td>
<td>DB_DATETIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_DATETIME</td>
<td></td>
</tr>
</tbody>
</table>

Continued
### SQL Parameter Operations
These methods are used to define new SQL parameters and to retrieve the values of those already existing.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetParameter(parameter)</td>
<td>parameter is Integer or String</td>
<td>Variant</td>
<td>BeforeGetParameter</td>
</tr>
<tr>
<td>GetParameterName</td>
<td></td>
<td>String</td>
<td>BeforeGetParameterName</td>
</tr>
<tr>
<td>NumParameters</td>
<td></td>
<td>Integer</td>
<td>BeforeSetParameter</td>
</tr>
<tr>
<td>SetParameter(parameter, value$)</td>
<td>parameter is Integer or String</td>
<td>Boolean</td>
<td>AfterSetParameter</td>
</tr>
</tbody>
</table>

### Error Operations
These methods are used to deal with error messages.

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Return Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetError</td>
<td></td>
<td>Constant *3</td>
</tr>
<tr>
<td>GetErrorMessage( [ error% ] )</td>
<td>error%: DB_LASTERROR or Constants *3</td>
<td>String</td>
</tr>
<tr>
<td>GetExtendedErrorMessage( [ error% ] )</td>
<td>error%: DB_LASTERROR or Constants *3</td>
<td>String</td>
</tr>
</tbody>
</table>
The following table shows return value constants of the FieldInfo method.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nDB_INFO_AUTOINCREMENT</td>
<td>DB_INFO_NULLABLE</td>
</tr>
<tr>
<td>DB_INFO_CASESENSITIVE</td>
<td>DB_INFO_PRECISION</td>
</tr>
<tr>
<td>DB_INFO_COLUMNID</td>
<td>DB_INFO_READONLY</td>
</tr>
<tr>
<td>DB_INFO_COLUMNNAME</td>
<td>DB_INFO_SCALE</td>
</tr>
<tr>
<td>DB_INFO_COMPUTED</td>
<td>DB_INFO_SEARCHABLE</td>
</tr>
<tr>
<td>DB_INFO_DISPLAYSIZE</td>
<td>DB_INFO_SETTABLE</td>
</tr>
<tr>
<td>DB_INFOEXPECTED_DATATYPE</td>
<td>DB_INFO_SQLDATATYPE</td>
</tr>
<tr>
<td>DB_INFO_LENGTH</td>
<td>DB_INFO_TABLENAME</td>
</tr>
<tr>
<td>DB_INFO_MONEY</td>
<td>DB_INFO_UNSIGNED</td>
</tr>
<tr>
<td>DB_INFO_NATIVE_DATATYPE</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows return value constants of the FieldNativeDataType method.

<table>
<thead>
<tr>
<th>SQL_TYPE</th>
<th>SQL_TYPE</th>
<th>SQL_TYPE</th>
<th>SQL_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_CHAR</td>
<td>SQL_FLOAT</td>
<td>SQL_TIMESTAMP</td>
<td>SQL_LONGVARCHAR</td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td>SQL_REAL</td>
<td>SQL_VARCHAR</td>
<td>SQL_BIGINT</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>SQL_DOUBLE</td>
<td>SQL_BINARY</td>
<td>SQL_TINYINT</td>
</tr>
<tr>
<td>SQL_INTEGER</td>
<td>SQL_DATE</td>
<td>SQL_VARBINARY</td>
<td>SQL_BIT</td>
</tr>
<tr>
<td>SQL_SMALLINT</td>
<td>SQL_TIME</td>
<td>SQL_LONGVARCHAR</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows error constants.

<table>
<thead>
<tr>
<th>Const</th>
<th>Const</th>
<th>Const</th>
<th>Const</th>
<th>Const</th>
<th>Const</th>
<th>Const</th>
<th>Const</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBstsSUCCESS</td>
<td>DBstsINVC</td>
<td>DBstsDSTY</td>
<td>DBstsTMPL</td>
<td>DBstsRDOM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsFAIL</td>
<td>DBstsNCOL</td>
<td>DBstsDRVN</td>
<td>DBstsBROW</td>
<td>DBstsRCHG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsMEMF</td>
<td>DBstsBADP</td>
<td>DBstsFITY</td>
<td>DBstsCANF</td>
<td>DBstsRUNC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsNCON</td>
<td>DBstsODBC</td>
<td>DBstsFILT</td>
<td>DBstsCNVR</td>
<td>DBstsCXIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsCCON</td>
<td>DBstsLIBM</td>
<td>DBstsINST</td>
<td>DBstsCNVD</td>
<td>DBstsAHVR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsNOEX</td>
<td>DBstsSNFD</td>
<td>DBstsNODR</td>
<td>DBstsHSTMT</td>
<td>DBstsCPAR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsINVR</td>
<td>DBstsINTR</td>
<td>DBstsNAUT</td>
<td>DBstsSQLP</td>
<td>DBstsNIRC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsCARR</td>
<td>DBstsACCS</td>
<td>DBstsNOSV</td>
<td>DBstsINTE</td>
<td>DBstsRDEL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsNODA</td>
<td>DBstsTYPE</td>
<td>DBstsNAPE</td>
<td>DBstsUPDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBstsEOFD</td>
<td>DBstsENTR</td>
<td>DBstsSVRQ</td>
<td>DBstsNUNQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

324  Developing Web Applications Using Lotus Notes Designer for Domino 4.6
Sample Programs Using the ODBCResultSet Class

In this example we get a column value list.

Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
'** sampleDB1 is a database name registered in this example
status = con.ConnectTo("sampleDB1")
Set qry.Connection = con
'** courses is a table name in the sampleDB1 database
qry.SQL = "select * from courses"
Set res.Query = qry
Call res.execute
Dim num As Integer
num = 0
Dim vl As Variant
Redim vl(num)
Do
'** name is a column name in the courses table
    vl(num) = res.GetValue("name")
    num = num + 1
    Redim Preserve vl(num)
Loop While res.NextRow
'** CValue is a field name in which a value list is saved
gDoc.CValue = vl

Using @DB Functions to Access Other Databases Through ODBC

@DBCommand, @DBLookup and @DBColumn are Notes functions that enable you to access RDBMSs which use the underlying ODBC interface. The @DB formulas are read-only.

The basic purpose of these functions is to create value lists for keyword fields. @DBLookup and @DBColumn can be used to query a relational database; @DBCommand is used only for executing stored procedures. @DBCommand does not return result sets. If you need a more customized and more complex query, LS:DO is a better option.
When to Use

Lotus Notes provides fast and easy-to-use read access to ODBC-compliant DBMSs via @DB functions. Notes @DB functions give developers the power of three frequently-used query tasks:

- **Generating Keyword Lists**
  The @DBColumn function in the Notes formula language generates Notes keyword lists from internal as well as external data sources. The same function supports keyword value lookups in tables stored in a DBMS through ODBC. For example, a Notes @DBColumn field formula can present a keyword list of customer names stored in a DBMS table when composing a document in a Notes customer contact tracking database.

- **Performing Lookup Operations**
  The @DBLookup function looks up a value in one field based on the value of a related field. For example, it will look up a customer phone number in a DBMS when given a customer name in Notes. Like @DBColumn, @DBLookup works both with other Notes databases and with external data sources through ODBC. The @DBColumn and @DBLookup functions can be used in other Notes formula contexts as well, such as input validation or translation formulas.

- **Launching External DBMS Stored Procedures**
  Database procedures and insert statements can be triggered with the @DBCommand function.

**Note** Some of these functions inherently involve a delay before they complete and so, in order to set users expectations, it is sometimes a good idea to code the functions behind a button so that the user expects some delay before the function is completed.

How to Use @DB Functions

The @DB functions are summarized in the following table:

<table>
<thead>
<tr>
<th>Functions</th>
<th>Descriptions</th>
<th>Equivalent SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DBColumn</td>
<td>Generates a keyword list. Returns a specified column for all rows in the specified table.</td>
<td>SELECT DISTINCT column_name FROM table_name</td>
</tr>
<tr>
<td>@DBLookup</td>
<td>Performs a lookup. Returns a specified column value in the row that matches the specified condition.</td>
<td>SELECT column FROM table WHERE condition</td>
</tr>
<tr>
<td>@DBCommand</td>
<td>Triggers stored procedures in the external database.</td>
<td>(any SQL statement)</td>
</tr>
</tbody>
</table>

326 Developing Web Applications Using Lotus Notes Designer for Domino 4.6
@DBColumn

The @DBColumn syntax is:

@DBColumn( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ;
Password1 : Password2 ; TableName ; ColumnName : NullHandling ;
Distinct : Sort )

Parameters:

<table>
<thead>
<tr>
<th>@DBColumn(&quot;ODBC&quot;)</th>
<th>Description</th>
<th>Choice</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache,</td>
<td>Inquiry Cache</td>
<td>“Cache” (Default)</td>
<td>X</td>
</tr>
<tr>
<td>DataSource,</td>
<td>Database resource name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserID1:UserID2,</td>
<td>User IDs</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Password1:Password2,</td>
<td>Passwords</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>TableName,</td>
<td>Table Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ColumnName:</td>
<td>Column Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NULLHandling,</td>
<td>Null Handling</td>
<td>“Fail” (Default)</td>
<td>X</td>
</tr>
<tr>
<td>Distinct:</td>
<td>Remove duplicate values</td>
<td>“Distinct”</td>
<td>X</td>
</tr>
<tr>
<td>Sort</td>
<td>Sort Direction</td>
<td>“Ascending”</td>
<td>X</td>
</tr>
<tr>
<td>Sort</td>
<td>Sort Direction</td>
<td>“Descending”</td>
<td></td>
</tr>
</tbody>
</table>

@DBLookup

The @DBLookup syntax is:

@DBLookup( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ;
Password1 : Password2 ; TableName ; ColumnName :
NULLHandling ; KeyColumn ; Key ; Distinct : Sort )

Parameters:

<table>
<thead>
<tr>
<th>@DBLookup(&quot;ODBC&quot;)</th>
<th>Description</th>
<th>Choice</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache,</td>
<td>Inquiry Cache</td>
<td>“Cache” (Default)</td>
<td>X</td>
</tr>
<tr>
<td>DataSource,</td>
<td>Database resource name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserID1:UserID2,</td>
<td>User IDs</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Password1:Password2,</td>
<td>Passwords</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>TableName,</td>
<td>Table Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KeyColumn</td>
<td>Key Column</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distinct:</td>
<td>Remove duplicate values</td>
<td>“Distinct”</td>
<td>X</td>
</tr>
<tr>
<td>Sort</td>
<td>Sort Direction</td>
<td>“Ascending”</td>
<td>X</td>
</tr>
<tr>
<td>Sort</td>
<td>Sort Direction</td>
<td>“Descending”</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 11: Accessing External Data from a Domino Application  327
### Summary

This chapter covered the LotusScript Data Object (LS:DO). In the chapter, the registration of ODBC data sources was covered. The differences between LS:DO and ODBC were also briefly discussed. Finally, @DB functions, i.e. @DBCommand, @DBLookup and @DBColumn, which are provided in Lotus Notes for accessing third party databases through ODBC and their use, were described.
Appendix A
Notes Features to Avoid when Developing Web Applications

The following tables list all the features which are not available or applicable for Web users.

### Notes Form Properties Not Supported on the Web

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basics properties</strong></td>
<td></td>
</tr>
<tr>
<td>Version control</td>
<td></td>
</tr>
<tr>
<td>Anonymous forms</td>
<td></td>
</tr>
<tr>
<td>Merge replication conflicts</td>
<td></td>
</tr>
<tr>
<td><strong>Defaults properties</strong></td>
<td>Read mode is supported, but do not use for documents that need to be created or edited on the Web.</td>
</tr>
<tr>
<td>Store form in document - edit mode</td>
<td></td>
</tr>
<tr>
<td>Disable field exchange</td>
<td></td>
</tr>
<tr>
<td>Automatically refresh fields</td>
<td></td>
</tr>
<tr>
<td><strong>Launch properties</strong></td>
<td>Other On Create Inherit options are supported.</td>
</tr>
<tr>
<td>Auto Launch options (First Attachment, First Document Link, First OLE object)</td>
<td></td>
</tr>
</tbody>
</table>

*Continued*
<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security properties</strong></td>
<td></td>
</tr>
<tr>
<td>Default encryption keys</td>
<td>Not applicable to Web.</td>
</tr>
<tr>
<td>Disable printing/forwarding/copying to clipboard</td>
<td></td>
</tr>
<tr>
<td><strong>Form elements</strong></td>
<td></td>
</tr>
<tr>
<td>Layout regions</td>
<td>Use tables to align form components instead.</td>
</tr>
<tr>
<td>Pop-up hotspots</td>
<td></td>
</tr>
<tr>
<td>ActiveX components, OLE, and OCX objects</td>
<td>Not supported on Macintosh, UNIX, and OS/2 platforms. Supported for display on Windows NT and Windows 95 platforms, but users can’t save changes made to objects.</td>
</tr>
<tr>
<td>Border controls for tables</td>
<td>If the top left cell of a Notes table has a border, the entire table is displayed with a border; otherwise, there is no border because of limitations with HTML.</td>
</tr>
</tbody>
</table>
## Notes Field Properties Not Supported on the Web

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes/FX fields</td>
<td></td>
</tr>
<tr>
<td>Basics properties</td>
<td></td>
</tr>
<tr>
<td>Use Address dialog for choices</td>
<td>Not applicable to the Web. Web user access to databases is based on the authenticated name.</td>
</tr>
<tr>
<td>Use Access Control List for choices</td>
<td></td>
</tr>
<tr>
<td>Use View dialog for choices</td>
<td></td>
</tr>
<tr>
<td>Compute after Validation</td>
<td></td>
</tr>
<tr>
<td>Keywords styles</td>
<td></td>
</tr>
<tr>
<td>Don’t display entry helper button</td>
<td>Not applicable to the Web.</td>
</tr>
<tr>
<td>Options properties</td>
<td></td>
</tr>
<tr>
<td>Help description</td>
<td></td>
</tr>
<tr>
<td>Give this field default focus</td>
<td></td>
</tr>
<tr>
<td>Enable encryption for this field</td>
<td>Web users can read data in encrypted fields.</td>
</tr>
<tr>
<td>Signed if mailed or saved in section</td>
<td></td>
</tr>
</tbody>
</table>

## Notes Text Styles Not Supported on the Web

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style properties</td>
<td></td>
</tr>
<tr>
<td>Default fonts (Helv and Times Roman in Windows).</td>
<td></td>
</tr>
<tr>
<td>Alignment and Tab properties</td>
<td></td>
</tr>
<tr>
<td>Full Justification and No Wrap</td>
<td></td>
</tr>
<tr>
<td>alignment styles</td>
<td></td>
</tr>
<tr>
<td>Interline spacing</td>
<td></td>
</tr>
<tr>
<td>Formatting aids, such as tabs,</td>
<td>HTML doesn’t support these formatting styles. Use tables to align form components. Use the text style pass-thru HTML to use HTML formatting styles.</td>
</tr>
<tr>
<td>indents, outdents, and extra spaces</td>
<td></td>
</tr>
<tr>
<td>Text effects properties</td>
<td></td>
</tr>
<tr>
<td>Shadow, Emboss, and Extrude text</td>
<td></td>
</tr>
<tr>
<td>styles</td>
<td></td>
</tr>
</tbody>
</table>
### Notes View Properties Not Supported on the Web

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Views and folders</strong></td>
<td></td>
</tr>
<tr>
<td>Options properties</td>
<td></td>
</tr>
<tr>
<td>Collapse all when database is first opened.</td>
<td>Web views do not expand or collapse all; instead they expand or collapse only one category at a time (equivalent to Expand/Collapse Selected Level).</td>
</tr>
<tr>
<td>Show in View menu</td>
<td>Web applications do not have a View menu. To exclude a view from the folders navigator, use the Design - Design Properties box to hide the view from Web users.</td>
</tr>
<tr>
<td><strong>On Open: Go To... options</strong></td>
<td></td>
</tr>
<tr>
<td><strong>On Refresh options</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Style properties</strong></td>
<td></td>
</tr>
<tr>
<td>Unread rows</td>
<td></td>
</tr>
<tr>
<td>Alternate rows</td>
<td></td>
</tr>
<tr>
<td>Show selection margin</td>
<td></td>
</tr>
<tr>
<td>Beveled column headings</td>
<td></td>
</tr>
<tr>
<td><strong>Advanced properties</strong></td>
<td></td>
</tr>
<tr>
<td>Refresh index options</td>
<td></td>
</tr>
<tr>
<td>Discard index options</td>
<td>Views can be re-indexed at a Notes server.</td>
</tr>
<tr>
<td><strong>Columns</strong></td>
<td></td>
</tr>
<tr>
<td>Basics properties</td>
<td></td>
</tr>
<tr>
<td>Resizable</td>
<td></td>
</tr>
<tr>
<td>Show twistie when row is expandable</td>
<td>Triangles are always shown.</td>
</tr>
</tbody>
</table>

### Notes Navigator Properties Not Supported on the Web

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basics properties</strong></td>
<td></td>
</tr>
<tr>
<td>Auto adjust panes at runtime</td>
<td></td>
</tr>
<tr>
<td><strong>Objects</strong></td>
<td></td>
</tr>
<tr>
<td>Polyline objects</td>
<td>Polyline objects display on the Web, but clicking them has no effect.</td>
</tr>
<tr>
<td><strong>HiLite properties</strong></td>
<td></td>
</tr>
<tr>
<td>Highlight when... options for navigator objects</td>
<td>Dragging and dropping to folders is not available on the Web.</td>
</tr>
<tr>
<td>“Alias a folder” simple action</td>
<td></td>
</tr>
</tbody>
</table>
### Notes Actions and Agent Properties Not Supported on the Web

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agents</strong></td>
<td>Web users do not have an Actions menu. Run agents with actions or buttons that use <code>@Command([ToolsRunMacro])</code> or through the WebQuerySave and WebQueryOpen form events.</td>
</tr>
<tr>
<td>Manually from Actions Menu</td>
<td>“If Document Has Been Pasted” and the document selection option “Selected documents.” The concepts of “pasted documents” and “selected documents” don’t apply to Web applications.</td>
</tr>
<tr>
<td>Form and view actions</td>
<td>Web users do not have an Actions menu. Use the option “Include action in button bar” instead.</td>
</tr>
<tr>
<td>Include action in Action menu</td>
<td>Use supported <code>@commands</code> to create the equivalent actions.</td>
</tr>
<tr>
<td>Default form and view actions that use system commands (such as <code>*Edit Document, *Categorize</code>) Use supported <code>@commands</code> to create the equivalent actions.</td>
<td></td>
</tr>
<tr>
<td>Programs for actions and agents</td>
<td>Supported for agents only.</td>
</tr>
<tr>
<td>Simple actions</td>
<td>LotusScript for forms, actions, and buttons</td>
</tr>
<tr>
<td>LotusScript for forms, actions, and buttons</td>
<td>LotusScript for forms, actions, and buttons</td>
</tr>
</tbody>
</table>

### Notes @Functions Not Supported on the Web

<table>
<thead>
<tr>
<th>@Function</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Certificate</td>
<td>Available only with the syntax <code>@DbCommand(&quot;Domino&quot;,&quot;ViewNextPage&quot;)</code> and <code>@DbCommand(&quot;Domino&quot;,&quot;ViewPreviousPage&quot;)</code> to create a link to the next/previous page in a view. Not available in other contexts.</td>
</tr>
<tr>
<td>@DbCommand</td>
<td>Available only with the syntax <code>@DbCommand(&quot;Domino&quot;,&quot;ViewNextPage&quot;)</code> and <code>@DbCommand(&quot;Domino&quot;,&quot;ViewPreviousPage&quot;)</code> to create a link to the next/previous page in a view. Not available in other contexts.</td>
</tr>
<tr>
<td>@DDEExecute</td>
<td></td>
</tr>
<tr>
<td>@DDEInitiate</td>
<td></td>
</tr>
<tr>
<td>@DDEPoke</td>
<td></td>
</tr>
<tr>
<td>@DDETerminate</td>
<td></td>
</tr>
<tr>
<td>@DocMark</td>
<td></td>
</tr>
<tr>
<td>@DeleteDocument</td>
<td></td>
</tr>
</tbody>
</table>

*Continued*
<table>
<thead>
<tr>
<th>@Function</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DocChildren</td>
<td></td>
</tr>
<tr>
<td>@DocDescendants</td>
<td></td>
</tr>
<tr>
<td>@DocLevel</td>
<td></td>
</tr>
<tr>
<td>@DocNumber</td>
<td></td>
</tr>
<tr>
<td>@DocParentNumber</td>
<td></td>
</tr>
<tr>
<td>@DocSiblings</td>
<td></td>
</tr>
<tr>
<td>@IsCategory</td>
<td>Not available except in view and column formulas.</td>
</tr>
<tr>
<td>@IsExpandable</td>
<td></td>
</tr>
<tr>
<td>@Responses</td>
<td></td>
</tr>
<tr>
<td>@DialogBox</td>
<td></td>
</tr>
<tr>
<td>@PickList</td>
<td></td>
</tr>
<tr>
<td>@Prompt</td>
<td></td>
</tr>
<tr>
<td>@IsModalHelp</td>
<td></td>
</tr>
<tr>
<td>@GetPortsList</td>
<td></td>
</tr>
<tr>
<td>@Environment</td>
<td>Use predefined field names to gather information about the Web user’s</td>
</tr>
<tr>
<td>@SetEnvironment</td>
<td>environment by requesting Common Gateway Interface (CGI) environment</td>
</tr>
<tr>
<td>ENVIRONMENT keyword</td>
<td>variables.</td>
</tr>
<tr>
<td>@MailSend</td>
<td></td>
</tr>
<tr>
<td>@Domain</td>
<td></td>
</tr>
<tr>
<td>@MailDbName</td>
<td></td>
</tr>
<tr>
<td>@MailEncryptSavedPreference</td>
<td></td>
</tr>
<tr>
<td>@MailEncryptSendPreference</td>
<td></td>
</tr>
<tr>
<td>@MailSavePreference</td>
<td></td>
</tr>
<tr>
<td>@MailSignPreference</td>
<td></td>
</tr>
<tr>
<td>@IsAgentEnabled</td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingMailed</td>
<td></td>
</tr>
<tr>
<td>@URLGetHeader</td>
<td></td>
</tr>
<tr>
<td>@URLHistory</td>
<td></td>
</tr>
<tr>
<td>@UserPrivileges</td>
<td></td>
</tr>
<tr>
<td>@Platform for user’s platform</td>
<td>Returns server’s platform only. Use @ClientType to distinguish between</td>
</tr>
<tr>
<td></td>
<td>Web and Notes users.</td>
</tr>
</tbody>
</table>
Appendix B
Downloading and Setting Up the Chile Pepper Site

This section describes how to download and set up the Chile Pepper Site.

Downloading the Chile Pepper Demonstration

The Chile Pepper Marketing Board Demo is available for download from the Chile Pepper home site. The URL is:

http://chilepepper.lotus.com

Before downloading, you have to be registered. You can register by clicking the Register link on the home page. After registration you can download the demo.

The Chile Pepper Marketing Board demo is made up of two Notes databases. You have to download both databases.

<table>
<thead>
<tr>
<th>Database</th>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chile Pepper Web Site</td>
<td>ChilePepperSite.nsf</td>
<td>The main database containing most of the site’s content. Notes users should start here.</td>
</tr>
<tr>
<td>Chile Pepper Registration</td>
<td>ChilePepper.nsf</td>
<td>The registration/guest book database. This also functions as a Notes Address Book to hold usernames and passwords for the site.</td>
</tr>
</tbody>
</table>

To maximize the design options, the Chile Pepper Marketing Board demo site was developed for Netscape Navigator 3.0 or Microsoft Internet Explorer 3.01.
Setting Up the Demo Site

Complete the following steps to set up Chile Pepper demo:

1. Install both databases into a subdirectory within the Notes data directory to be called /ChilesDirect.

2. Sign the databases with your ID:
   - Select File - Tools - Server Admin.
   - Select Database Tools in the bottom right.
   - Select the databases and then Sign a Database in the Tools menu.
   - Sign every design note so that the authorizations work properly.

   **Note** Make sure that the ID that signs the agents has access to run agents in the server document.

3. Set the default in the ACL in the Public Address Book to be editor.

4. If desired, use the Chile Pepper Web Site database as the main database.

5. To enable Java Applets to run:
   - Select File - Tools - User Preferences.
   - In the Basics, select Enable Java Applets.
   - Click OK.

   **Note** To enable your changes, Notes must be restarted.
Appendix C
Table of CGI Variables Supported by Domino

The following table lists all the CGI (Common Gateway Interface) variables which are supported by Domino.

Domino captures the following CGI variables through a field or a LotusScript agent. You can also capture any CGI variable preceded by HTTP or HTTPS. For example, cookies are sent to the server by the browser as HTTP_Cookie.

<table>
<thead>
<tr>
<th>CGI Variable</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth_Type</td>
<td>If the server supports user authentication and the script is protected, this is the protocol-specific authentication method used to validate the user.</td>
</tr>
<tr>
<td>Content_Length</td>
<td>For queries that have attached information, such as HTTP POST and PUT, this is the content type of the data.</td>
</tr>
<tr>
<td>Content_Type</td>
<td>The version of the CGI spec with which the server complies.</td>
</tr>
<tr>
<td>Gateway_Interface</td>
<td>The version of the CGI spec with which the server complies.</td>
</tr>
<tr>
<td>HTTP_Accept</td>
<td>The MIME types that the client accepts, as specified by HTTP headers.</td>
</tr>
<tr>
<td>HTTP_Refferer</td>
<td>The URL of the page the user used to get here.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Indicates if SSL mode is enabled for the server.</td>
</tr>
<tr>
<td>HTTP_User_Agent</td>
<td>The browser that the client is using to send the request.</td>
</tr>
<tr>
<td>Path_Info</td>
<td>The extra path information (from the server’s root HTML directory), as given by the client. In other words, scripts can be accessed by their virtual path name, followed by extra information that is sent as PATH_INFO.</td>
</tr>
<tr>
<td>Path_Translated</td>
<td>The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.</td>
</tr>
</tbody>
</table>

Continued
<table>
<thead>
<tr>
<th>CGI Variable</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query_String</td>
<td>The information that follows the ? in the URL that referenced this script.</td>
</tr>
<tr>
<td>Remote_Addr</td>
<td>The IP address of the remote host making the request.</td>
</tr>
<tr>
<td>Remote_Host</td>
<td>The name of the host making the request.</td>
</tr>
<tr>
<td>Remote_Ident</td>
<td>This variable will be set to the remote user name retrieved from the server. Use this variable only for logging.</td>
</tr>
<tr>
<td>Remote_User</td>
<td>Authentication method that returns the authenticated user name.</td>
</tr>
<tr>
<td>Request_Method</td>
<td>The method used to make the request. For HTTP, this is “GET,” “HEAD,” “POST,” and so on.</td>
</tr>
<tr>
<td>Script_Name</td>
<td>A virtual path to the script being executed, used for self-referencing URLs.</td>
</tr>
<tr>
<td>Server_Name</td>
<td>The server’s host name, DNS alias, or IP address as it would appear in self-referencing URLs.</td>
</tr>
<tr>
<td>Server_Protocol</td>
<td>The name and revision of the information protocol accompanying this request.</td>
</tr>
<tr>
<td>Server_Port</td>
<td>The port to which the request was sent.</td>
</tr>
<tr>
<td>Server_Software</td>
<td>The name and version of the information server software running the CGI program.</td>
</tr>
<tr>
<td>Server_URL_Gateway_Interface</td>
<td>The version of the CGI spec with which the server complies.</td>
</tr>
</tbody>
</table>

For more information about CGI see:

[http://hoohoo.ncsa.uiuc.edu/cgi/env.html](http://hoohoo.ncsa.uiuc.edu/cgi/env.html)
Appendix D
Editing The HTTPD.CNF file to Add MIME Type Mappings

HTTPD.CNF is the configuration file for your Internet system and can be used to specify MIME (Multimedia Internet Mail Extension) type mappings, a process that maps an attachment’s file extension, for example, .WAV for sound files, to an external viewer or a helper application. When you include MIME type mappings in HTTPD.CNF, Domino reads the extension of the attached file, and sends the appropriate information to the browser.

The syntax for specifying MIME type mapping is:

```
Addtype .extension type/subtype/ encoding [quality[character-set]]
```

The following table contains a sample of MIME type settings in a HTTPD.CNF file:

<table>
<thead>
<tr>
<th>AddType</th>
<th>extension</th>
<th>type/subtype</th>
<th>encoding</th>
<th>quality</th>
<th>character-set</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.mime</td>
<td>www/mime</td>
<td>binary</td>
<td>1.0</td>
<td># Internal MIME is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.bin</td>
<td>application/octet-stream</td>
<td>binary</td>
<td>1.0</td>
<td># Uninterpreted binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.oda</td>
<td>application/oda</td>
<td>binary</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.pdf</td>
<td>application/pdf</td>
<td>binary</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.ai</td>
<td>application/postscript</td>
<td>8-bit</td>
<td>.5</td>
<td># Adobe Illustrator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>application/postscript</td>
<td>8-bit</td>
<td>.8</td>
<td># PostScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.eps</td>
<td>application/postscript</td>
<td>8-bit</td>
<td>.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.ps</td>
<td>application/postscript</td>
<td>8-bit</td>
<td>.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.rtf</td>
<td>application/x-rtf</td>
<td>7-bit</td>
<td>1.0</td>
<td># RTF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following LotusScript agent may be used to update the HTTPD.CNF. The example comes from the Chile Pepper Demo Site. See Appendix B for more information on downloading and setting up this demo site.

Option Public

' To make all string comparisons not sensitive to case.

Option Compare Nocase

%INCLUDE "LCONST.LSS"

Sub Initialize

' This agent updates the Domino HTTPD.CNF file
" to include a mime type definition for shockwave

Dim session As New NotesSession

Dim envNames As Variant

Close

' Read the 'Notes/Data' path.

notesdatadir$ =

    session.GetEnvironmentString("Directory", True)

Print "Checking httpd.cnf in" + notesdatadir$

' See if the entry is already there

f1% = Freefile()

found = False

Open notesdatadir$ + "\httpd.cnf" For Input As f1%

While Not Eof(f1%)

    Line Input #f1%, buffer$

    If Instr(buffer$, "application/x-director") > 0 Then

        Found = True

    End If

Wend

Close f1%
If not, then add it to the HTTPD.CNF file. The content of the file is copied into a new file named HTTPD.NEW together with the added rows that are needed for Shockwave. Then HTTPD.CNF is renamed as HTTPD.$$ for backup, while HTTPD.NEW becomes HTTPD.CNF.

If Not Found Then

    Print "Updating httpd.cnf"
    c%=0
    f1% = Freefile()
    Open notesdatadir$ + "\httpd.cnf" For Input As f1%
    f2% = Freefile()
    Open notesdatadir$ + "\httpd.new" For Output As f2%
    While Not Eof(f1%)
        c%=c%+1
        Line Input #f1%, buffer$
        Print #f2%, buffer$
        If Instr(buffer$,"www/mime") Then
            Print #f2%,
                "AddType  .dcr      application/x-director
                    binary  1.0 # Shockwave for Director"
            Print #f2%,
                "AddType  .dir      application/x-director
                    binary  1.0 # Shockwave for Director"
            Print #f2%,
                "AddType  .dxr      application/x-director
                    binary  1.0 # Shockwave for Director"
        EndIf
    Wend
    Close

Appendix D: Editing The HTTPD.CNF file to Add MIME Type Mappings 341
' Rename the new httpd.cnf file and keep a backup

On Error Resume Next

Err = 0

Kill notesdatadir$ + "\httpd.$$$"

If Err<>0 Then
    Err = 0
End If

On Error Goto 0

Name notesdatadir$ + "\httpd.cnf" As notesdatadir$ + "\httpd.$$$"
Name notesdatadir$ + "\httpd.new" As notesdatadir$ + "\httpd.cnf"

End If

Print "Done checking httpd.cnf"

End Sub
Related ITSO Publications

This section lists other Lotus related publications produced by the International Technical Support Organization (ITSO). For information on ordering these ITSO publications see “How To Get ITSO Redbooks.”

ITSO Lotus Publications

- Lotus Notes 4.5: A Developers Handbook, IBM form number SG24-4876-00, Lotus part number AA0425
- LotusScript for Visual Basic Programmers, IBM form number SG24-4856-00, Lotus part number 12498
- Secrets to Running Lotus Notes: The Decisions No One Tells You How to Make, IBM form number SG24-4875-00, Lotus part number AA0424
- Enterprise Integration with Domino Connect, IBM form number SG24-2181-00, Lotus part number 12913
- Deploying Domino in an S/390 Environment, IBM form number SG24-2182-00, Lotus part number 12957

Other ITSO Lotus Related Publications

The publications listed in this section may also be of interest:

- The Domino Defense: Security in Lotus Notes and the Internet, IBM form number SG24-4848-00, Lotus part number 12967
- Lotus Solutions for the Enterprise, Volume 1. Lotus Notes: An Enterprise Application Platform, IBM form number SG24-4837-00, Lotus part number 12968
- Lotus Solutions for the Enterprise, Volume 3. Using the IBM CICS Gateway for Lotus Notes, IBM form number SG24-4512-00
- Lotus Solutions for the Enterprise, Volume 4. Lotus Notes and the MQSeries Enterprise Integrator, IBM form number SG24-2217-00
- Lotus Domino Server Release 4.5 on AIX Systems Installation, Customization and Administration, IBM form number SG24-4694-00, Lotus part number 12969
• *IBM PC Server and Lotus Domino Integration Guide*, IBM form number SG24-2102-00, Lotus part number 12970
• *Using Lotus Notes on the IBM Integrated PC Server for AS/400*, IBM form number SG24-4779-00
• *Mail Integration for Lotus Notes 4.5 on the IBM Integrated PC Server for AS/400*, IBM form number SG24-4977-01
• *Managing a Notes Environment with TME 10 M*, IBM form number SG24-2104-00
• *Image and Workflow Library: Integrating IBM FlowMark with Lotus Notes*, IBM form number SG24-4851-00
• *Lotus Notes Release 4 In a Multiplatform Environment*, IBM form number SG24-4649-00
• *Implementing LAN Server for MVS in a Lotus Notes Environment*, IBM form number SG24-4741-00
• *Using ADSM to Back Up Databases*, IBM form number SG24-4335-02
• *NetFinity V5.0 Database Support*, IBM form number SG24-4808-00
• *An Approach to ODBC: Lotus Approach to DB2*, IBM form number SG24-4685-00
How To Get ITSO Redbooks

This section explains how both customers and IBM/Lotus employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM and Lotus Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

IBM Employees

- PUBORDER - to order hardcopies in the United States.
- GOPHER link to the Internet - type gopher.wtscpok.itso.ibm.com.
- Tools disks
  - To get LIST3820s of the redbooks listed in the Other ITSO Lotus Related Publications section, type one of the following commands:
    - tools sendto ehone4 tools2 redprint get sg24xxxx package
    - tools sendto canvm2 tools redprint get sg24xxxx package (Canadian users only)

Note The ITSO Lotus publications, and this redbook Developing Web Applications Using Lotus Notes Designer for Domino 4.6, are not available in LIST3820 or BookManager format. However, PDF format is available for those books from http://www.lotus.com/redbook

- To get lists of redbooks:
  - tools sendto wtscpok tools redbooks get redbooks catalog
  - tools sendto usdist mkttools mkttools get itsocat txt
  - tools sendto usdist mkttools mkttools get listserv package
To register for information on workshops, residencies, and redbooks:
  tools sendto wtscpok tools zdisk get itsoregi 1997
For a list of product area specialists in the ITSO:
  tools sendto wtscpok tools zdisk get orgcard package
• Redbooks Home Page on the World Wide Web
  
  Note  The ITSO Lotus Related Publications are listed on this site, but not
  the ITSO Lotus redbooks, which are available in Adobe Acrobat format
  from http://www.lotus.com/redbook
• IBM Direct Publications Catalog on the World Wide Web
  IBM employees may obtain LIST3820s of redbooks from this page.
• ITSO4USA category on INEWS
• Online - send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at
  IBMMAIL.

Lotus Employees
• This redbook, and those listed in the ITSO Lotus Publications section of
  this book, can be ordered from the Internal Orders Database using Lotus
  part numbers.
• In addition PDF versions of those books can be downloaded from:
  http://www.lotus.com/redbook
• The other ITSO publications listed can be accessed on the Redbooks
  Home Page on the World Wide Web at:

All Employees
• Internet Listserver
  With an Internet e-mail address, anyone can subscribe to an IBM
  Announcement Listserver. To initiate the service, send an e-mail note to
  announce@webster.ibmlink.ibm.com with the keyword subscribe in the
  body of the note (leave the subject line blank). A category form and
detailed instructions will be sent to you.
How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:
  - IBM Mail
    - In United States: uisb6fpl at ibmmail
    - In Canada: cahlmbkz at ibmmail
    - Outside North America: bookshop at dkibmbsh at ibmmail
  - Internet
    - usib6fpl@ibmmail.com
    - lmannix@vnet.ibm.com
    - bookshop@dk.ibm.com

- **Telephone orders**
  - United States (toll free): 1-800-879-2755
  - Canada (toll free): 1-800-IBM-4YOU
  - Outside North America (long distance charges apply):
    - (+45) 4810-1320 - Danish
    - (+45) 4810-1020 - German
    - (+45) 4810-1420 - Dutch
    - (+45) 4810-1620 - Italian
    - (+45) 4810-1540 - English
    - (+45) 4810-1270 - Norwegian
    - (+45) 4810-1670 - Finnish
    - (+45) 4810-1120 - Spanish
    - (+45) 4810-1220 - French
    - (+45) 4810-1170 - Swedish

- **Mail Orders** - send orders to:
  - IBM Publications
    - IBM Direct Services
  - Publications Customer Support
    - 144-4th Avenue, S.W.
    - Sortemosevej 21
  - P.O. Box 29554
    - Calgary, Alberta T2P 3N5
    - DK-3450 Allerod
  - Raleigh, NC 27626-0570
    - Canada
    - Denmark

- **Fax** - send orders to:
  - United States (toll free): 1-800-445-9269
  - Canada (toll free): 1-800-267-4455
  - Outside North America (long distance charge): (+45) 48 14 2207

- **1-800-IBM-4FAX (United States) or (+1) 415 855 43 29 (Outside USA)** — ask for:
  - Index #4421 Abstracts of new redbooks
  - Index #4422 IBM redbooks
  - Index #4420 Redbooks for last six months

- **Direct Services** — send note to software@software@ibm.com

- **On the World Wide Web**
  - Redbooks Home Page: http://www.redbooks.ibm.com

- **Internet Listserv**
  - With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserv. To initiate the service send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).
Symbols

$HTMLHead, 140, 164, 165, 167, 172
and frames, 164
to auto-reload a Web page, 141
$HTMLHead field, 284
$return, 71, 122, 138, 217
changing Submit messages, 139
to go back to document, 140
to launch an agent, 140
to send user to another page, 140
$returnAuthenticationFailure, 78
$returnAuthorizationFailure, 78
$returnDocumentDeleted, 78, 211
$returnGeneralError, 78
$search, 217
$searchSiteTemplate, 225
$searchTemplate, 221
$searchTemplateDefault, 221
$viewBody, 222
$viewTemplate, 121
creating, 105
$viewTemplateDefault, 59
& arguments separator, 187
<a> tag, 185
$about, 189
@AttachmentNames, 144
<BLINK>, 135
<BODY>, 126
<BODY> tag, 160, 164, 173
@ClientType, 54, 241
@Command
ToolsRunMacro, 120, 206
@Command([Compose]), 184
@Command([FileCloseWindow]), 140
@Command([FileOpenDatabase]), 183
@Command([FileSave]), 140
@Command([OpenDocument]), 183
@Command([OpenNavigator]), 184
@Command([OpenView]), 121, 183
@Command([ToolsRunMacro]), 153
@Command([ViewRefreshFields]), 138
@DB Functions, 326
@DBColumn, 327
@DBCommand, 328
@DBLookup, 327
@DBColumn, 297, 325, 327
@DBCommand, 297, 325, 328
@DBLookup, 297, 325, 327
@DBName, 142
$defaultform, 191
$defaultNav, 191
$defaultview, 140
$File, 198
$First, 189
<FRAME> tag, 160
<FRAMESET> tag, 160
@function formula language, 90
@Functions, 298
<HEAD>, 140
$help, 190
.htm file, 127
$icon, 190
<Input> tag, 132
<Marquee> tag, 146
<Meta> tag, 140
$OLEOBJINFO, 198
@Password, 233
$PublicAccess
in form design, 29
$searchForm, 199
<table>, 136
<textarea>, 132, 133
@URLOpen(), 121, 184
@Username function, 230

Numbers

3PaneUI, 188

A

About This Database Document, 10
accelerator key, 32, 202
Accessibility Control List (ACL), 17, 27, 34, 228, 229, 237
ability to create LotusScript
agents, 29
ability to read public documents, 29
Anonymous, 240
anonymous access, 30
creating documents, 28
deleting documents, 28
groups, 239
levels of access, 27, 28, 237
personal agents, 29
personal folders/views, 29
roles, 239
setting up, 239
share personal folders/views, 29
accessing applications over the Web, 25
accessing databases, 27
accessing RDBMS from Notes, 297, 298
accessing SSL application, 247
accessing Web sites, 27
action bar buttons, 74
add Java applet, 262
adding HTML to design elements, 126
advanced templates, 21
agent builder, 202
Agents, 201
ability to create LotusScript
agents, 29
ability to create personal, 28, 237
allowing web users to run, 203
capturing CGI variables, 206
database, 205
listing views in a database, 154
LotusScript, 204
naming, 202
personal (ability to create), 29
properties, 205
restricted, 206
running as a Web user, 205
running on any server, 204
running with signer's privilege, 205
select documents to be processed, 204
unrestricted, 206
using formulas, 204
webqueryopen, 206
webquerysave, 206
what should the agent do, 204
alias, 32
All readers and above, 38
Allow HTTP clients to browse databases, 237
specifying to Domino, 4
alternate text, 77
animated images, 76
Anonymous, 34, 228, 230, 239, 240
anonymous access, 30
Anonymous Form, 34
applet parameters, 270
applet with document, 262
application templates, 13
Arguments, 186
attachments, 66
authentication, 229
authentication between server and client, 256
author access, 27, 237
Auto Launch, 36
auto refresh a field, 285
automatically refresh fields, 35
B
background color of forms, 36
basic authentication, 229
Basics settings
specifying for Domino in server document, 4
BODY tag, 40
buttons, 74, 184
to activate agents, 201
C
CA issue certificate, 252
Cache
specifying directory for Domino, 6
specifying shutdown options for Domino, 6
specifying size for Domino, 6
calendaring and scheduling
$PublicAccess field, 29
ability to read public documents, 29
write public documents, 29
cascading names and address books, 232
certificate add to person documents, 255
certificate clients, 255
certificate merged into server key ring file, 252
certification process, 256
CGL, 71, 206, 229
directory, specifying for Domino, 6
URL path, specifying for Domino, 6
CGI environment variable
REMOTE_USER, 230
CGI variables, 78
check box, for keywords, 47
Chile Pepper Demo Site, 141, 165, 171
class library, LS:DO, 313
client access to server, 256
client certificate, 255
client submit certificate requirement, 255
clients access SSL enabled server, 255
CollapseView, 189
COLS, 132
command
editclear, 211
ToolsRunMacro, 39, 201
computed fields, 45
computed for display, 46
computed subforms, 53
computed text, 69, 129
computed when composed, 45
config SSL protocol, 253
Configuring the Domino Web Server, 4
conformance levels, 305
context sensitivity, 18
Conversion format
specifying for images (Domino), 7
cookie
description of, 292
features, 293
syntax, 292
Copy Database dialog box, 17
copying a database, 16
Count, 189
Create-Other menu, 33
create CA key ring file, 248
create CA server key ring file, 250
create Java agent, 281
Create menu, 33
creating
an agent, 202
computed text, 69
documents, 28, 31
embedded folder pane, 65
embedded navigators, 62
embedded views, 64
fields, 41
file upload control, 66
horizontal rules, 68
LotusScript agents, 29
messages to users, 71
multiple buttons, 74
new database, 13, 18
tables, 56
views, 88
customized search form
creating, 218
customizing error messages, 78, 211
D
data type mapping, 309
Database, 186
database access facilities, 297
database icon
ability to create, 27, 237
database
accessing, 229
accessing anonymously, 228
accessing encrypted, 15
ACL, 228, 237
Address Book type, 19
anonymous access, 30, 239
controlling access, 237
copying existing, 16
creating new, 13, 18
default size, 15
definition, 9
encryption, 15, 17, 19
file extension, 14
inherit design, 21
library type, 19
logging, 234
monitoring, 234
multi database search type, 19
opening in Notes, 22
opening on the Web, 22
personal journal type, 19
properties, 18
require SSL connection, 20
setting maximum access for Internet browser, 239
setting up the ACL, 239
size, 20
size limit, 9
standard type, 19
using JavaScript when generating pages, 20
viewing, 81
default form, 35
default link column, 144
delete
database, 27, 237
documents (ability to), 28
manager access, 27, 237
deleting a profile document example of using an agent, 210
depositor access, 28, 237
design, inherit, 21
design elements, 10, 31
adding HTML, 126
in subforms, 51
Designer Access, 27, 237
Dialog List, 46
DIB: Device Independent Bitmap, 121
directory links, 239
Disable Field Exchange, 35
Discussion Template, 32
display Java applet menu, 266
DNS lookup
specifying for Domino, 4
document fields
$Anonymous, 34
$UpdatedBy, 34
document links, 98, 182
document retrieval, 23
DocumentContext, 79, 207
documentcontext property, 210
documents
computed text, 70
converting to HTML, 31
creating anonymously, 34
edit mode, 242
embedding Web elements in, 110
overriding form background properties, 37
previewing in Notes, 42
previewing on the Web, 42
read mode, 242
Domain Name Services, 236
Domino Security
levels of, 227
Domino Architecture, 1
examples
cookie, 293
Java agent, 281
JavaScript, 285, 288, 289, 290, 291
using Java Notes classes, 277
Web users validation and authentication, 256
Expand, 189
ExpandView, 189
F
field encryption, 242
field help, 48
FieldOffset, 198
fields
$$Return, 71
ability to create, 27, 237
Alignment tab, 49
capturing CGI variables, 78
creating of, 41
definition, 11
Dialog List, 46
derector values not in list, 50
Fonts and Color tab, 48
Hiding tab, 49
HTML attributes, 43
JavaScript, 74
keyword check box, 47
keyword field on the Web, 50
Option tab, 48
Print Option tab, 49
properties of, 41, 45
Radio Button, 47
type keyword, 41
validation formula, 50
writing JavaScript, 44
file upload control, 66
firewalls, 234
application proxy, 235
circuit-level proxy, 236
controlling information flow, 235
circuit-level proxy, 235
packet-filtering, 235
SOCKS, 236
folders
ability to create personal, 29
ability to share personal, 29
folders vs views, 82
fonts, 20
footer, 20
form builder, 38
form with Java applet, 262
forms, 31
ability to create, 27, 237
351
allow users to override background properties, 37
available for public access, 38
background properties, 40
Background tab, 36
customizing error messages, 78
Defaults tab, 34
definition, 11
disabling printing, forwarding, copying, 38
ever messages, 78
events for, 38, 39
formatting, 56
hiding from Notes client, 55
hiding from the Web browser, 55
how to name, 32
HTML attributes, 40
Inherit entire selected document, 35
Launch tab, 36
layout, 56
On Create option, 35
Present mail send dialog, 35
previewing on the Web, 43
Print tab, 37
properties Basics tab, 32
properties, 32
Security tab, 38
show context pane option, 35
special types of, 77
title of, 39
type of, 33
using tables for layout, 57
formulas
@DBName, 72
@DocumentUniqueID, 73
@Text, 73
ClientType, 54
frames, 160, 176
creating, 161, 176
stored in documents, 164, 165
stored in forms, 170
FRAMESET Template, 176
Frameset wizard, 177
full-text indexing, 23
full-text search, 23
full-text index
location, 216
multi-databases, 224
size, 216
updates, 216

G
Garbage collection
specifying interval for Domino, 6
GIFs
specifying settings for Domino, 6
graphs
importing, 37
pasting into the background, 37
supported formats, 37
graphics background, 120

H
title, 20
Header and Footer, 37
hide-when conditions, 49
hiding,
a line, 242
a paragraph, 49
content, 241
content by formula, 242
database name list, 237
from Notes clients, 49
from the Web browser, 49
paragraphs, 242
subforms, 52
home page
creating, 26
Home URL, 25
specifying for Domino, 5
horizontal rules, 68
Host, 186
Host name
specifying for Domino, 4
hotspot actions, 104
hotspots, 74, 117, 183
to activate agents, 201
HTML
$HTMLHead, 140
<Body>tag, 129, 131
and $Return field, 138
and agents, 153
and computed fields, 128
and computed text, 128
and documents, 130
and Forms, 126
and navigators, 152
and RichText, 130
and shared fields, 152
and static text, 134
and subforms, 152
and tables, 136
attributes for documents, 131
attributes for fields, 132
attributes for forms, 129
directory, specifying for Domino, 6
for Multi-Line fields, 133
for Views and Folders, 144
in columns formulas, 145
in labels, 146
in square brackets, 127
Pass-Thru HTML style, 126
testing Web pages generation, 156
to mask fields, 134
using HTML to format fields, 132
HTML attributes, 40, 43, 129
HTTP_USER_AGENT, 79
HTTP password, 228, 230, 233
HTTP protocol, 243
HTTP servers, 1, 2
settings, 4

I
directory, specifying for Domino, 6
URL path, specifying for Domino, 6
Image as DocLink, 144
ImageFormat, 198
Images
interlaced rendering setting for Domino, 6
JPEG image quality setting for Domino, 6
progressive rendering setting for Domino, 6
specifying conversion format for Domino, 6
images, 75
alternate text, 77
animated, 76
copying, 76
importing, 76
in text-only browsers, 77
using pass-thru HTML, 76
IMAP, 236, 244
importing graphics, 37
indexing, full-text, 23
InfoBox for Java applet, 267, 268
Inherit entire selected document, 35
Initialize event, 204
input forms, 38
input translation, 47
input validation, 47
Internet applications server, 1

J
Java, 2, 202
base class file, 259
import from local file system, 260
overview, 259
Java agent, 260, 274, 276, 281
creating an agent, 275
what is a Java agent, 272
Java agents, 205
Java agents compare with Java applets, 274
Java applet, 274
add to Domino design elements, 262
applet properties, 267
created with version 4.5, 260, 261
created with version 4.6, 260
design pane, 269
display status, 262
displayed via browser, 261
export dialog, 271
general applet properties tab, 268
how to select applet, 262
imported from network, 264
Java applet menu, 266
parameter value, 270
refresh dialog, 271
reorganize dialog, 271
set applet parameters, 270
stop a Java applet, 265
troubleshooting, 271
User Preference setting, 263
Java attributes, 265
Java debug console, 262
Java files, 260, 273
Java Notes classes, 273, 275, 277
Java program, 274
JavaScript, 44, 74, 284
add to form or field, 284
elements, 288, 289, 290, 291
function, 285
overview, 283
JPEGs
specifying settings for Domino, 6

K
key ring file with CA, 248
key ring file with CA server, 250
key ring files, 244

L
launch options, 22
LDAP, 236, 244
library database, 19
links, 181
anchor link, 182
database link, 182
explicit URL link, 185
hotspots, 183
HTML link, 185
relative vs. absolute notation, 183
URL link, 183
using action hotspots, 183
view link, 182
list of public databases, 19
local security, 27, 237
Local Web Preview Process, 43
Log files
specifying Error log for Domino, 7
Logging settings
specifying for Domino in the server document, 7
Login, 190
Login argument, 187
LotusScript, 47, 52, 202, 204, 273
ability to create agents, 29
DocumentContext property, 79, 207
NotesSession class, 79
LotusScript:Data Object (LS:DO), 297, 298, 301
architecture, 303
class library, 313
tracing and debugging, 309
when to use, 304, 325
LS:DO and ODBC, differences, 304

M
macros
see also agents
mapping data types, 309
Mapping settings
specifying CGI directory, 6
specifying CGI URL path, 6
specifying for Domino in server document, 5
specifying HTML directory, 6
specifying icon URL path, 6
specifying Icons directory, 6
maximum Internet browser access, 239
MAXLENGTH, 132
Merge replication conflicts, 34
merging certificate, 252
Microsoft Internet Explorer File Upload Control, 68
Microsoft ODBC driver, 311
modify CA profile, 249
modifying notes.ini, 231
multi-database full text indexed, 224
multi-value separators, 48
multiple buttons, 74

N
NAME, 132
names and address book, 232
naming
an agent, 202
navigators, 22
actions, 118
and Frames, 123
and GIF images, 120, 121
and HTML, 120, 121, 123
and views, 117
calculated, 121
create menu, 119
creating, 119
creating a graphic, 120
definition, 11
design, 119, 120
embedding in a form, 121
hotspots and @Formulas, 121
how to build, 117
importing vs. pasting images, 121
list of graphic objects, 117
properties box, 120
template form, 124
New Database dialog box, 14
NNTP, 236, 244
no access (in ACL), 28, 237
NOTES.INI, 231, 232
Notes Designer, 88
Notes Designer terms, 10
NotesObject, 186

353
programming a Java program, 275
  properties
  - of agents, 205
  - of databases, 18
  - of fields, 45
  - of forms, 32
Properties SmartIcon, 18
proxy, 235
public/private key, 228
Public Address Book, 228, 232
  person document, 234
public agents
  ability to create, 27, 237

Q
Query, 199
Query_String, 123
QueryOpen form event, 206
QuerySave form event, 206

R
Radio Button, 47
RDBMS, accessing from Notes, 297, 298
  read mode, 35, 242
  read public documents, 29
reader access, 28, 237
real-time data access, 301
registered users, 228
registering ODBC data sources, 307
registering Web users, 231
registration application
  customizing, 231
removing subforms, 53
replication conflicts, 34
replication formulas
  ability to create/modify, 27, 237
replicator, 10
responses, 33
restricted agents, 206
RichText, 130
ROWS, 132
RSA, 243
search scope, 223
search site
  database, 222
  index, 215, 222
  $rcrsite.ntf, 222
searching, 213
  $$ SearchSiteTemplate, 225
  $$Return, 217, 220
  $$Search, 217, 219
  $$SearchTemplate, 221
  $$ViewBody, 222
default search form, 217
full text index, 215, 224
search result form, 217
search site index, 215
search site URLs, 214
search view URLs, 214
views, 213, 226
Web Search Advanced, 217, 225
Web Search Simple, 217, 225
Web site, 213, 226
SearchMax, 200
SearchOrder, 199
SearchThesaurus, 200
SearchWordVariants, 200
Secure Sockets Layer (SSL), 228, 243
  authentication, 243
  Certificate Authority Application, 247
  client authentication, 244
digital signatures, 243, 244
  encrypting transactions, 228
  ensuring confidentiality, 243
  internal Certification Authority, 247
  key ring files, 244, 246
message validation, 244
privacy, 244
public key, 243, 244
public key certificates, 243
request server certificates, 246
server authentication, 244
Server Certificate Administration application, 246
setting up for an Intranet, 246
symmetric key encryption, 243
X.509, 244

S
SaveOptions, 139
scheduled agents, 203
search, 23
Search Builder, 33
search related URLs, 214
column, 83
copying, 93
definition, 11
description of, 83
document link, 87
embedded, 105, 110
expand, 84
folders, 81
full HTML view, 147, 149
hidden, 88
link columns, 99, 100
naming, 90
on the Web, 85
properties, 90, 92
selection column, 83
selection formula, 82, 89
sorting, 84
standard, 81, 83
treating entire view as HTML, 92
types of views, 81
UNID, 111
URLs, 111
view pane, 88
Web navigation, 103
Web navigation bar, 86
what is a view, 81
Viewtemplates, 85, 87
adding navigation elements, 107
associating a view with a form, 103
embedding a folder pane, 106
embedding a view, 105, 106
menu actions, 103
Visual Basic, 304

creating a new, 230
defining, 230
existing Notes users, 230
hiding content from, 241
registering, 231, 232
separating from Notes users, 232
WebInterestProfileEdit agent, 207
WebInterestProfileOpen agent, 209
WebInterestProfileSave agent, 209
WebQueryOpen, 154, 206
WebQuerySave, 139, 154, 206
Welcome page
specifying for Domino, 4
window title, 39
WRAP, 132
write public documents, 29

W
Web, 1
Web application
creating, 13
Web browser
holding user’s identity, 229
Web elements, 59
embedded folder panes, 64
embedded navigators, 60
embedded views, 62
file upload control, 66
Web session, 229
Web site
multi-database site, 24
setting up, 24
single database site, 24
structure, 9
Web users

356
IBM Redbook Order Form

Please send me the following:

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

☐ Please put me on the mailing list for updated versions of the IBM Redbook Catalog.

First name ___________________________ Last name ___________________________

Company ______________________________

Address ______________________________

City ___________________________ Postal code ___________________________ Country

Telephone number _____________________ Telefax number _____________________ VAT number _____________________

☐ Invoice to customer number ___________________________

☐ Credit card number ___________________________

Credit card expiration date ___________________________ Card issued to ___________________________ Signature ___________________________

We accept American Express, Diners Club, Eurocard, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.