Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 169.

First Edition (April 2001)

This edition applies to Version4, Release 4 of OS/400 (5796-SS1).

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

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.

© Copyright International Business Machines Corporation 2001. 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.
## Contents

Preface .......................................................... vii
The team that wrote this redbook .......................... vii
Comments welcome ........................................ viii

### Chapter 1. Introduction to Patterns for e-business .......................... 1

1.1 Patterns for e-business .................................. 1
   1.1.1 Design patterns .................................. 2
   1.1.2 Components of the Patterns for e-business .... 2
   1.1.3 Defined Patterns for e-business ................. 3
   1.1.4 How to use these patterns ....................... 5
   1.1.5 Patterns for e-business Web site ............... 6
1.2 The User-to-Business pattern ............................ 6
1.3 The IBM Framework for e-business ................... 8
1.4 Structure of this redbook .............................. 8

### Chapter 2. A four-stage approach to Patterns for e-business ............. 9

2.1 Overview .................................................. 9
   2.1.1 Advantages of the four-stage methodology .. 11
   2.1.2 The phased approach ........................... 12
2.2 Stage zero: Preparing for e-business systems .......... 13
   2.2.1 What you should know before advancing to stage one . 13
   2.2.2 Preparing to move to stage one ............... 14
   2.2.3 When to move to the next stage ............... 16
2.3 Stage one: Application server .......................... 17
   2.3.1 Servlets .......................................... 18
   2.3.2 JavaServer Pages (JSP) ......................... 19
   2.3.3 JavaBeans ....................................... 19
   2.3.4 Enterprise JavaBeans (EJB) .................... 20
   2.3.5 eXtensible Markup Language (XML) .......... 20
   2.3.6 What you should know about stage one ....... 21
   2.3.7 Preparing to move to stage two .............. 24
   2.3.8 When to move to the next stage .............. 24
2.4 Stage two: User security approach ................... 25
   2.4.1 Directories: Base for authentication and personalization .... 25
   2.4.2 Personalization .................................. 27
   2.4.3 What you should know about stage two ....... 28
   2.4.4 Preparing to move to stage three .......... 29
   2.4.5 When to move to the next stage ........... 29
2.5 Stage three: Enterprise approach .................... 30
   2.5.1 Legacy definition ............................... 30
   2.5.2 Enterprise application design ............... 30
2.6 What the next stage should be ........................................ 32

Chapter 3. Modifying the PDK application for OS/400 .................... 33
3.1 Pattern Development Kit (PDK) ......................................... 33
3.2 Using legacy resources .................................................. 33
3.3 Using the Pattern Development Kit in VisualAge for Java .......... 34
   3.3.1 Communicating with RPG programs ............................ 35
   3.3.2 Using EJB technology ............................................ 39
3.4 Using the PDK in WebSphere Studio ................................. 47
   3.4.1 Using Pagelist with RPG ........................................ 47
   3.4.2 Adding data using the RPG program ......................... 62

Chapter 4. Setting up the AS/400e server, running the application ... 79
4.1 System environment ...................................................... 79
   4.1.1 AS/400e server .................................................... 79
   4.1.2 PC workstation .................................................. 80
4.2 Technology overview .................................................... 80
   4.2.1 Application servers .............................................. 80
   4.2.2 Directory services ............................................... 85
4.3 System setup tasks ....................................................... 88
   4.3.1 Setting up a configuration file for WebSphere Application Server 88
   4.3.2 Configuring the HTTP server .................................. 96
   4.3.3 Configuring LDAP with Operations Navigator ............... 103
   4.3.4 Installing WAS Administrator Console ...................... 108
   4.3.5 Testing your configuration .................................... 109
4.4 Restoring the database and sample e-business application ........ 113
4.5 The sample e-business application .................................. 114
   4.5.1 Application scenario ............................................ 114
   4.5.2 Restoring the project on WebSphere Studio .................. 116
   4.5.3 Changing the connection properties ......................... 119
   4.5.4 Publishing the project on the server ......................... 122
4.6 Configuring the sample e-business application environment ....... 130
   4.6.1 Manual configuration .......................................... 130
   4.6.2 Automatic configuration ....................................... 153
4.7 Running the sample e-business application on a Web browser .... 155
   4.7.1 Stage zero ....................................................... 156
   4.7.2 Stage one ....................................................... 158
   4.7.3 Stage two ....................................................... 160
   4.7.4 Stage three ..................................................... 162

Appendix A. Using the additional material ............................... 167
A.1 Locating the additional material on the Internet ................. 167
A.2 Using the Web material ............................................... 167
Preface

Patterns for e-business are a group of proven, reusable models that can help speed the process of developing applications. The pattern discussed in this book, the User-to-Business pattern, is the general case for users that interact with enterprise transactions and data. In particular, it is relevant to enterprises that deal with goods and services that cannot be listed and sold from a catalog.

This redbook discusses two application topologies of the User-to-Business pattern. Application topology 1 describes a situation where you build an application that has no need to connect to back-end or legacy data. Topology 2 extends topology 1 to describe a situation where you need to access existing data on legacy or third-party systems.

For practical purposes, this redbook walks you step-by-step through the implementation of an e-business application using application topology 1 and the OS/400 system. The sample e-business application and associated files are available for download from the IBM Redbooks Web site.

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 Rochester Center.

Masahiko Hamada is an Advisory International Technical Support Specialist for the iSeries and AS/400e servers at the International Technical Support Organization, Rochester Center. He writes extensively and teaches IBM classes worldwide on all areas of iSeries e-business. Before joining the ITSO in 2000, he worked in the AS/400 Field Support Center in Japan as an AS/400 System Specialist.

Kohjiroh Hayashi is an IT Specialist in IBM Japan, Global Services. He has over eight years of experience with IBM midrange systems. He specializes in applications using Lotus Notes, WebSphere, RPG, and C/S tools. He currently provides e-business solutions to customers mainly using WebSphere Family products.

Vicente Moredo is the Sales Director of MD Systems Informatica located in Sao Paulo, Brazil. He has 10 years of experience in the IT solutions area, dealing with networking projects and support, database, development, and work management. In the last five years, he worked with AS/400e servers as
an IT manager and previously as a sales specialist. He currently provides comprehensive IT solutions for customers.

Thanks to the following people for their invaluable contributions to this project:

Jonathan Adams
Stephen P. Young
IBM UK

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

• Fax the evaluation form found in “IBM Redbooks review” on page 181 to the fax number shown on the form.
• Use the online evaluation form found at ibm.com/redbooks
• Send your comments in an Internet note to redbook@us.ibm.com
Chapter 1. Introduction to Patterns for e-business

Rapid advances in computer hardware have been driven by the use of standards and well-specified components for assembly. The desire to apply these same approaches to software construction has given rise to object-oriented software, design patterns, and component-based development.

The concept of design patterns has gained acceptance by software designers and developers. Design patterns enable efficiency in both the communication and implementation of software design, based on a common vocabulary and source. Information technology architects, who face challenges in systematic and repeatable descriptions of systems, are encouraged by the success of design patterns and have explored the idea of architectural patterns.

Enterprise Solution Structure (ESS) looked at patterns for complete end-to-end system architectures. ESS is now part of the IBM Global Services methodology. For more information about ESS, see “Enterprise Solutions Structure” in IBM Systems Journal, Volume 38, No. 1, 1999, which is available on the Web at: http://www.research.ibm.com/journal/sj38-1.html

The following publications offer more information on design patterns and their background:

- Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, et al
- Pattern-Oriented Software Architecture, Volume 1: A System of Patterns, Buschmann, et al
- Pattern Hatching: Design Patterns Applied, Vlissides, John

1.1 Patterns for e-business

The purpose of Patterns for e-business is to communicate (in a highly accessible fashion) the business pattern, systems architecture (application and runtime topologies), product mappings, and guidelines required for different classes of applications. For some patterns, there is also an associated Pattern Development Kit (PDK), which provides sample application code to illustrate the effective use of those patterns.
The goal is to provide the smallest number of Patterns for e-business that will allow IT architects, in 80% of cases, to quickly develop 80% of their required infrastructure by the reuse of proven:

- Architecture patterns
- Design patterns
- Runtime patterns
- Application development and systems management patterns
- Design, development, and deployment guidelines
- Code

1.1.1 Design patterns

Design patterns describe the atomic elements that are reused to create architectural structures. These structures can make up part or all of an application. Some structures are so common that their patterns are reusable.

The Patterns for e-business fall into this category of patterns. They are made up of atomic elements and represent 80% of the applications that are made up of a common set of elements.

1.1.2 Components of the Patterns for e-business

The components that make up Patterns for e-business are shown in Figure 1. Each component is explained here:

- **Business patterns**: Describe the interaction between participants in an e-business solution.
- **Application topology**: Illustrates the various ways to configure the interaction between users, applications, and data. Choosing an application topology leads to a basic runtime topology.
- **Runtime topology**: Uses nodes to group functional requirements. The nodes are interconnected to solve a business problem. An application topology leads to a basic runtime topology.
- **Product mappings**: Show possible combinations of products used to instantiate the runtime topology.
- **Guidelines**: Outline and define the processes used to build the e-business application.
1.1.3 Defined Patterns for e-business

There are currently six defined Patterns for e-business:

- **User-to-Business**: This is the general case of users (internal or external) who interact with enterprise transactions and data. In particular, this pattern is relevant to enterprises that deal with goods and services that cannot be listed and sold from a catalog. You can also think of it as covering all user-to-business interactions that are not covered by the User-to-Online Buying pattern.

- **User-to-Online Buying**: This pattern is used to describe the special case (a subset of the User-to-Business pattern) where packaged goods, for example, are sold through a catalog using a shopping cart, a wallet, etc. This includes both consumers purchasing goods or online buyers purchasing goods from a single supplier. It also include links to back-end systems to allow for inventory updates and credit checking.

- **Business-to-Business**: This pattern is used to describe two styles of inter-business-to-business activity. Intra-business-to-business is covered
under the Application Integration pattern. The two styles of inter-business-to-business activity are:

- **B2Bi**: This style covers programmatic links between arms-length businesses (where potentially a trading partner agreement may be appropriate). A good example of this would be supply/chain applications.

- **B2M2B**: The second style covers the eMarketPlace. The “M” represents the eMarketPlace that supports multiple buyers and suppliers. The buying function may be performed online or programmatically.

- **User-to-User**: This pattern describes users who are collaborating with one another by e-mail, shared documents, etc.

- **User-to-Data**: This pattern describes users who need to use tools to extract useful information from large volumes of data, text, images, video, etc.

- **Application Integration**: This pattern links applications together within a business (such as Enterprise Resource Planning (ERP) with existing applications). It can be used within a business pattern or between business patterns. As shown in Figure 2, IBM views e-business as an integration of many application domains into systems that connect a business with its customers, partners, and suppliers.

![Figure 2. e-business integration](image-url)
These systems are not confined to Web interfaces. However, increasingly many of the user interfaces to the combined system use Web technology.

The common set of node descriptions in the Patterns for e-business enables communication between architects and designers from different application domains. This set suggests areas for shared nodes and infrastructure. This is similar to the process of using design patterns to solve a programming design problem, where classes in the composed pattern play multiple roles, derived from the source patterns. However, it is also different, because design pattern composition is based on class diagrams and a white box, by nature. Composing architectural patterns are more component-based.

The Patterns for e-business may be applied to e-business solution areas. Table 1 offers a guide to which areas you may find they apply most.

Table 1. Patterns for e-business and e-business solutions

<table>
<thead>
<tr>
<th>e-business solution area</th>
<th>Business pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer relationship management</td>
<td>User-to-Business pattern</td>
</tr>
<tr>
<td>e-commerce</td>
<td>User-to-Online Buying pattern</td>
</tr>
<tr>
<td>Supply chain management</td>
<td>Business-to-Business pattern</td>
</tr>
<tr>
<td>Collaboration</td>
<td>User-to-User pattern</td>
</tr>
<tr>
<td>Business intelligence; knowledge</td>
<td>User-to-Data pattern</td>
</tr>
<tr>
<td>management</td>
<td></td>
</tr>
<tr>
<td>Business application integration</td>
<td>Application Integration pattern</td>
</tr>
</tbody>
</table>

1.1.4 How to use these patterns

The Patterns for e-business are particularly focused on addressing common business application problems and providing answers to frequent architecture, design, and implementation questions. You can use the Patterns for e-business in a number of ways according to your needs:

- As a starting point for an end-to-end system architecture
- As a detailed example and prescriptive approach, following the product mappings and guidance provided
- As a way to design more complex, multi-channel systems, when several patterns are used together

As with the design patterns and ESS, architects and designers may want to combine these patterns to compose solutions to more complex system architectures.
We recommend that you use the Patterns for e-business together with an appropriate development methodology that considers the full set of requirements that are to be understood and implemented. These requirements may concern the function of the solution or its operational characteristics such as availability, scalability, or performance.

1.1.5 Patterns for e-business Web site

The Patterns for e-business are published on IBM developerWorks, a portal for developers. This Web site is located at:
http://www.ibm.com/developerworks/patterns

This interactive patterns site acts as a guide to aid you in the selection of the pattern and topologies that are most relevant to your needs. You can also register at this site for pattern-related updates.

While you can navigate via shortcuts to the information you most need, the site is structured to enable you to “drill down” into the material. On this site, you should:

1. Select a business pattern.
2. Select an application topology.
3. Review runtime topologies.
4. Review product mappings.
5. Review guidelines.

At the time this redbook was written, the Web site supported material for the User-to-Business, Business-to-Business, User-to-Data, and User-to-Online Buying patterns. It also offered material for other business patterns in the process of development.

1.2 The User-to-Business pattern

In the User-to-Business pattern, users who are either internal or external to the enterprise can interact with enterprise transactions and data. In some cases, there may be a need to access back-end applications and data. This pattern is relevant to enterprises that deal with goods and services that are not normally listed in and sold from a catalog. It encompasses all user-to-business interactions that are not covered by the User-to-Online Buying pattern. Many, but not all, of the functions supported by the User-to-Business pattern relate to Customer Relationship Management (CRM) systems.
The following industry examples indicate where the User-to-Business pattern would provide the appropriate application and runtime topologies to fit each particular need:

- **Insurance Industry**
  - Locate a nearby office
  - Locate brokers or agents
  - Financial planner and insurance needs analysis tool
  - Portfolio summary
  - Policy summary and details
  - Claims submission and tracking
  - Online billing

- **Discount Brokerage**
  - Portfolio summary
  - Detailed holdings
  - Buy and sell stocks
  - Transaction history
  - Quotes and news

- **Convenience Banking**
  - View account balances
  - View recent transactions
  - Pay bills/transfer funds
  - Stop payments
  - Manage bank card

- **Telecommunications and Wireless Industry**
  - Review account statements
  - Paying bills online
  - Change personal profile
  - Add/change/remove services (for example, call waiting or caller ID)
  - Submitting service requests

- **Government**
  - Submit tax returns
  - Renew automobile licenses
  - Download forms/applications
  - Submit forms/applications

- **Manufacturing**
  - Review required parts/services
  - Locate service centers
  - Register for training classes
  - Submit/track orders
1.3 The IBM Framework for e-business

The advent of e-business, with the requirement for interoperability that it brings, has been a major catalyst for the rapid adoption of standards by the industry. The IBM Framework for e-business establishes:

- A recommended approach for building systems, embodied in the Patterns for e-business
- Innovative technology delivered in a rich product portfolio
- Cross-platform standards, including Java and XML

The Framework, with the standards it prescribes for e-business systems and their components, can be applied to:

- Custom application code
- Application packages
- Software products

The Patterns for e-business are an integral part of the IBM Framework for e-business. The patterns make it easy to apply the technologies, standards, and products of the IBM Framework to provide an e-business solution.

1.4 Structure of this redbook

The remainder of this redbook follows this structure:

- Chapter 2, “A four-stage approach to Patterns for e-business” on page 9, introduces the concepts to quick start e-business on an AS/400e server. It describes the important stages to reach the latest technology for e-business systems. It also gives considerations about each stage and what you need to move to the next level.

- Chapter 3, “Modifying the PDK application for OS/400” on page 33, discusses how to modify the sample code and technology that is available and renominated. You can find a sample application that demonstrates this concept in Chapter 2, “A four-stage approach to Patterns for e-business” on page 9.

- Chapter 4, “Setting up the AS/400e server, running the application” on page 79, explains how to set up the sample e-business application provided in this redbook. As a result, you will achieve the knowledge you need to configure your own environment.
Chapter 2. A four-stage approach to Patterns for e-business

This chapter explains all the concepts you need to quickly start e-business on the AS/400e server. It describes the important stages to achieve the latest technology for e-business systems. It also offers considerations about each stage and what you need to move to the next level.

The main approach is to explain the concepts and provide best practice standards for developing and design techniques. For detailed information and source code samples for each technology, see Chapter 3, “Modifying the PDK application for OS/400” on page 33.

2.1 Overview

When IBM created the concept of Patterns for e-business, it intended to give you a best practice method to construct and deploy e-business applications. This redbook examines User-to-Business (U2B) applications. In fact, if you look at a typical U2B application, you will see a lot of characteristics that are, in essence, the same as other U2B applications. That's why understanding Patterns for e-business can help you to create a state-of-the-art e-business application.

Using Patterns for e-business makes it easy to understand all the components found in U2B applications. They also help you create e-business applications by using the best tools, best knowledge, and best analytical view.

The deployment of e-business applications occurs in stages. Each stage is made up of important tasks that you must complete before you move to another level of the application. This is similar to the cycle concept in all traditional applications, but with consideration to U2B applications. Figure 3 on page 10 helps to demonstrate this cycle.
Figure 3 shows a sequence of stages and a time direction looking forward to the future. Stage zero has the most common and primary way to introduce a company to the Internet. It is called stage zero because it is not an e-business stage yet, but is more like an entry-level stage.

In stage one, you can see a more mature and logical structure and, in fact, the first stage of an e-business system. It contains dynamic content, offers access to database, and supports separation of logic and presentation, which allows more robust and reusable code.

In stage two, your e-business system must be secure and deploy personalized information to users, probably personal data or monetary transactions. It is the natural upgrade of step one, where it doesn’t matter who is using the information or how the information must be deployed to each user.

Stage three is called a “last stage”, because it is difficult to see another way to expand U2B applications. In this stage, your system interacts with enterprise applications, for example, legacy systems, ERPs, Enterprise Java Beans (EJB), and any other kind of data processing available that is not exclusive to the e-business system. This stage opens endless options to
handle your e-business system and to make all the components work
together and in an intelligent way.

Each stage has its own life cycle and path to move to the next stage. Because of this, it is impossible to move to the next stage if you don’t perform the proper tasks. Figure 4 shows the life cycle of a stage.

![Stage life cycle](image)

*Figure 4. Stage life cycle*

It is important for you to achieve all the steps in each stage. When you move to the next stage, you make choices about the technology, infrastructure, people to support the application and the business, and application design. A wrong choice may incur a huge loss of time and money for a company. Be sure you understand each stage and its characteristics.

### 2.1.1 Advantages of the four-stage methodology

Based on the stage life cycle shown Figure 4, you can imagine how difficult it is to begin an e-business system from scratch. For example, when you create traditional systems with traditional languages, you have a solid knowledge from which you can decide, manage, and calculate risks. In e-business systems, everything is new and changing every day, so wisdom for developers has not achieved a mature stage.

It is important for you to understand the methodology stages, because when you do, you will achieve these advantages:
• **High quality source code**
  You can implement your own application by using the source code examples and sample e-business application provided in this redbook. These examples offer a proven stable source code and extend functions to your scenarios. At the same time, you can take advantage of the design used to develop the source code.

• **Productivity**
  You can reuse, maintain, and most importantly, prepare for the next stages using a methodology and samples that have already been tested. Usually developers adopt a “scrap and build” solution on traditional languages, because it’s very fast and old source code is not a factor. However, in e-business systems, if reusability is not important to you, your code will start to increase and become difficult to maintain. That's because the changes are very fast. This continues until the day you can't manage the changes any longer and you have to scrap everything. Reusing the sample code in this redbook will save you time.

• **Perspective**
  This redbook provides standards to help you create your e-business application and reduce investment risks. By understanding the sample code, the sample e-business application, the concepts, and the products described on this redbook, you can achieve the benefits of the standards. This is important when you consider the future, because the better you understand standards, the easier it will be to handle challenging new technology. Also, you can grow your business at any time by reusing almost all the tools you already have.

  The next section explains each stage in our methodology. It gives you the information and knowledge you need to make your own decisions about U2B e-business systems. Keep in mind that you must follow the stages in sequence for best practice in creating U2B e-business systems.

### 2.1.2 The phased approach

The redbook *AS/400e e-business Handbook: A Technology and Product Reference*, SG24-5694, describes the evolution of e-business sites using a phased approach. The three different phases in this approach are:

- **Phase one:** Web presence
- **Phase two:** Dynamic site
- **Phase three:** Transactional site
The four-stage approach described in this redbook is similar to the phase approach in the AS/400e e-business Handbook. The main difference is that the approach in this book spreads phase two and phase three into stage one, stage two, and stage three.

Both methods are similar in regard to the evolution of e-business systems. That is, each redbook explains a growth pattern to provide solutions to handle publishing an enterprise or transaction solution.

2.2 Stage zero: Preparing for e-business systems

Stage zero involves preparing your e-business systems. It could also be referred to as the “pre-stage”, because you gather all the necessary information and technology to move to stage one. For example, companies publish static information on Web pages to provide their customers with data about facilities, product information, and marketing. This may seem helpful at first, but is limited in terms of growth.

To advance to the next stages, you need to follow each step within each stage. Do not take shortcuts or bypass stages because this is not beneficial to e-business systems. Make sure you give serious thought to your goals and consider the importance of the unique tasks within each stage. Then, you will surely create a successful, high-quality e-business system.

2.2.1 What you should know before advancing to stage one

When you are at stage zero and plan to move to stage one, be sure to isolate the functions of a stage zero system, because it is not good to create relationships with other applications. For example, consider the workload to keep information updated on Web sites. You can easily update information with batch processing or other tools. However, there may be a lack of organization and control in the way information should be distributed. When you move to stage one, uncontrolled relationships between applications create a huge risk for conflict with resources running on the application server. Information becomes impossible to update when you have many sources of data. What is easy to update today may not be easy in the near future. Figure 5 on page 14 shows the correct way to manage your information.
Integrated management for information sources is fundamental to achieve success in stage one. You need to define a responsibility for managing the information resources.

As you progress, use an integration tool, such as WebSphere Studio, to help you easily manage the contents in an integrated environment.

2.2.2 Preparing to move to stage one

When you move to stage one, you need to pay special attention to your:

- Security policy
- Capacity plan
- Application scenario
- Application design

This will help facilitate a good transition from stage zero. This section briefly explains each of these areas and provides a point of reference for developing your e-business system.

2.2.2.1 Security policy

A security policy is a written document that defines the security controls that you institute for your computer systems. It also defines the risks that these
controls are intended to minimize. In addition, it defines the actions you should take if your security controls are breached.

The most important rule that your security policy should express is: Anything that is not explicitly permitted is denied by default. In other words, you must automatically disallow any actions that you do not specifically allow. This ensures that new types of attacks are not likely to go past your defenses. However, you may have no knowledge of them and have nothing in your security controls to defend specifically against them.

A security policy contains rules, such as who can access certain services or which services can be run from a given computer. The policy also contains information about the processes and controls that are instituted to enforce these rules. If you are connecting to the Internet, your security policy should stipulate that you install and use a firewall to control access to and from the Internet.

Once you create a policy, you must ensure that it is put into effect. This may involve establishing more restrictive password rules, installing and running virus protection software, holding classes to educate users on security rules, and so on.

Security is mandatory on every e-business system. However, keep in mind that security implementation must be kept from business logic. Because security is related to the applications environment, it can be changed every time your environment changes. But, business logic is related to the company’s business, so it must be portable across other platforms and easy to maintain. Therefore, we recommend that you never mix security logic with business logic.

You can find further information about security on the AS/400e server at the IBM Web site: http://www.as400.ibm.com/tstudio/secure1/secdex.htm

2.2.2.2 Capacity plan

Capacity planning is the process of estimating the system requirements used to forecast the growth of existing workloads and the addition of any new workloads. This involves reviewing historic data, obtaining forecasts of business growth, and making a judgement based on simple mathematics to predict future resource requirements. You could also use a modeling tool, such as BEST/1, to help collect performance data and information on anticipated growth.

While working in stage zero, you can gather information in advance to build your capacity plan. In a U2B environment, as well as any other Internet
environment, we recommend that you gather such information as page average hit ratio, page size in bytes, maximum users access, database size, line bandwidth, and peak time.

The purpose of a capacity plan is to define an estimate for the investment in system, people, and time resources.

You can find more information about capacity planning in the redbook AS/400 HTTP Server Performance and Capacity Planning, SG24-5645.

2.2.2.3 Your application scenario
You must establish a strategy for your target application. Consider asking such questions as “What kind of applications will we need to work in the system?” or “What kind of service will we supply to the user?” These examples show you how important it is to have an understanding of what application will run on your new e-business system.

Your application scenario is the only step in the application development cycle.

2.2.2.4 Application design
Technology is always changing and doing so rather quickly. On the other hand, concept and application design changes are more uncommon. The life cycle of an application could be as short as the business it was created to support. But, an application design can have a very long life time, because it is independent of environment changes. Therefore, a good design concept will result in an e-business system that will last a long time.

We recommend you use a standard design methodology such as Unified Modeling Language (UML), which is implemented on Java. For more information, see the redbook Design and Implement Servlets, JSPs, and EJBs for IBM WebSphere Application Server, SG24-5754.

2.2.3 When to move to the next stage
There are several indicators that you can use to identify whether you are ready to advance to stage one. Some of these indicators include:

- Static information updates demand too much maintenance workload.
- You want to supply dynamic content.
- You want to use business logic to supply information.
- You want to identify individual user access.
- You want real-time updates from databases that are maintained by other systems.
2.3 Stage one: Application server

This stage introduces the application server, which is the basis for stable e-business development. Application servers deploy well-designed, robust, and optimized e-business applications. They are the foundation for the development and implementation of U2B Patterns for e-business. They are also a strong platform for the implementation of all other Patterns for e-business, such as Business-to-Business (B2B) and User-to-Online-Buyer (U2OB).

An application server mediates between user requests and system resources. This type of server can be described as “middleware”, or a middle tier in a three-tier e-business environment, where the first tier is the HTTP server and the highest tier is the database and business logic (traditional business applications such as order processing). An example of this kind of e-business system is a dynamic Web site environment that involves automatic linking between the Web catalog and price changes in the customer’s price database.

The server has a set of routines or software that allows a user to run server-side applications, such as servlets, JavaServer Pages (JSP), and Enterprise JavaBeans (EJB). These applications can serve as a link to the existing legacy applications or database information.

The application server concepts are shown in Figure 6.

![Application Server Diagram](image)

**Figure 6. Application server concepts**
This redbook concentrates on WebSphere Application Server, which is the IBM solution to application servers. The sample e-business application that was created and the tests that were made at the time this redbook was written are based on this technology.

2.3.1 Servlets

Servlets are Java classes that run on Web servers to provide dynamic HTML content to clients. They take the HTTP request from the client as input and output dynamically generated HTML.

Why use servlets

At first sight, implementing applets may appear to be a good idea. However, servlets are more powerful than applets by design. Servlets run on the server and eliminate the class download time. Since an HTTP servlet can output HTML, relatively small amounts of data need to be transferred between the Web server and the Web browser. A high-speed link between the Web server and the data server should provide reasonable performance for servlet users that are connected using modems.

On the contrary, applets running in a browser cannot access system resources. They can only open a socket to the server that served the applet. This restriction is relaxed in newer browsers, which allow the user to grant authority to signed applets to perform restricted tasks. However, this is up to the individual user, not the administrator. The user may not have the necessary knowledge to know if this authority should be granted. To simplify the decision process for the end user, the administrator may choose to store the applets, dependent classes, and pages on the same system on which the data is stored. In a business where the data is stored on multiple servers, the administrator maintains applets, classes, and HTML pages on each server.

Servlets offer these advantages:

- Have full access to local resources.
- Are easy to develop.
- Are portable.
- Are multi-threaded.
- Run on a server, and therefore, no downloading is necessary.
- Can be dynamically loaded and unloaded without shutting down the HTTP server.
- Once installed, they can be compiled using a just-in-time (JIT) or native compiler. For the AS/400e server, they are transformed into executable
Java program objects. This eliminates the requirement to interpret Java class files at run time and results in better performance.

For more information, see the Java Servlet Technology page at:

2.3.2 JavaServer Pages (JSP)
JavaServer Pages are HTML source files that include Java extensions to provide dynamic content and increased functionality. JSP are compiled into servlets before they are deployed.

Why use JSP
The typical e-business application generates large amounts of user interface content such as HTML. Some of this content is dynamic depending on the results of computations, but much of the content is static.

On a servlet mechanism, you can output data to a Web browser. However, this practice does not adequately address user-interface design and program-logic design. In addition, the presence of many print statements are required to generate the HTML that makes servlets harder to understand and develop. To solve these problems, Sun Microsystems created JavaServer Pages.

JavaServer Pages are HTML pages that have been extended with a number of mechanisms to allow dynamic content to be added to the page as it is sent to the client. All processing of JSP HTML extensions is done on the server. No additional client support is required, so you don’t need to worry about Web browsers.

For more information, read about JSP on the IBM Application Framework for e-business site at: http://www-4.ibm.com/software/ebusiness/pm.html

2.3.3 JavaBeans
JavaBeans are Java components that are designed to be used on client systems. They are Java classes that conform to certain coding standards. They can be described in terms of their properties, methods, and events. JavaBeans may be packaged with a special descriptor class called a BeanInfo class and special property editor classes in a JAR file. JavaBeans may or may not be visual components.

Why use JavaBeans
With the JavaBeans API, you can create reusable, platform-independent components. JavaBeans (also called “beans”) are one way for developers to
have a standard, highly integrated, and highly interoperable infrastructure to build applications. Without such standards, developers are forced to invent their own mechanisms, which ultimately limit the inter-operability and portability of the components they develop.

For more information, see the JavaBeans Component Architecture Documentation site at: http://www.javasoft.com/beans/docs

2.3.4 Enterprise JavaBeans (EJB)

Despite the name, Enterprise JavaBeans (EJB) are not the same as JavaBeans. Enterprise JavaBeans are server-side Java components that are designed for distributed environments. They do not exist in isolation, but instead, are deployed in containers that provide such services as security, naming and directory services, and persistent storage. WebSphere Application Server acts as this type of container for EJB.

Why use EJB

You should use EJB-based logic because it can be distributed across multiple servers and can be accessed directly from clients via remote method call technology. EJB technology also reduces the cost of developing enterprise scale applications, while protecting your existing investment in IT due to the fact that EJB can interact with other resources like legacy applications.

EJB is a powerful tool to create a solid business logic that is available for the entire enterprise. Its implementation lets you be concerned only with the information you need from system resources, not how to get this information.


2.3.5 eXtensible Markup Language (XML)

XML (a subset of SGML) is a platform- and application-independent way of describing data using tags. It is similar to HTML in that it uses tags to describe document elements. It is different in that the tags describe the structure of the data rather than how the data is to be presented to a client. XML has the facility to allow data providers to define new tags as needed to better describe the data domain being represented.

Why use XML

XML’s strongest point is its ability to perform data interchange. Because different organizations (or even different parts of the same organization) rarely normalize on a single set of tools, it takes a significant amount of work for two groups to communicate. XML makes it easy to send structured data across applications so nothing becomes lost in translation. XML also makes it
easier for two computers to exchange data with each other. Your data is described using tags that explain what each piece of data is. XML doesn't replace HTML, since they're designed for different purposes. Where XML is the Web's language for data interchange, HTML is the Web's language for rendering.

Consider this analogy for applications when using XML. Application A can receive XML-tagged data from application B and vice versa. Neither application has to know how the other's system is organized, only that information on application A is the same type of information as on application B.

Another example is regarding search engines. One major problem today is that search engines can't process HTML intelligently. If you search on "chip" when you're looking for someone named Chip, you might find pages on chocolate chips, computer chips, and people named Chip. But, if there was a description for name and address records, searching for someone named Chip could generate much more accurate and useful search results.

The advantages of XML include:

- It gives you a standard way to output data.
- It separates data from the presentation.
- It's a standard for any kind of browser.
- It is core to the future implementation of other technologies.

For more information, visit the developerWorks XML Zone on the Web at:
http://www.software.ibm.com/xml

2.3.6 What you should know about stage one

This section explains a recommended structure for e-business application development in stage one.

2.3.6.1 Model-View-Controller (MVC)

Model-View-Controller is implemented to produce maintainable applications that can be efficiently developed. However, this structure is not required. Developers can follow it fully, or they can adapt it to their special requirements, always looking for the best quality e-business application.

Figure 7 on page 22 shows the major elements of an MVC structure and their relationships. This structure allows various skills and tools to be used for different elements of the application. It also allows for the development sequence of the various parts to be largely separated. The idea of this structure is that servlets process HTTP requests and coordinate the rest of
the application elements. The primary application logic is separated into Java components (command beans) outside the servlet and independent of the details of HTTP access. This allows these components to be used to support other styles of Web applications such as those based on direct applet-to-server communication. JSPs are assigned the responsibility of generating the user interface. They take data provided by the servlet, format it, and then insert it into an HTML data stream to be sent to the client. Generally, JSPs should not contain application logic.

![Model-View-Controller components](image)

**Figure 7. Model-View-Controller design architecture**

### 2.3.6.2 Business model design

When you develop e-business systems in stage one, keep in mind that everything, in general, should be designed. For example, there should not be a design only for one specific system, not only for one Web system, not only for one kind of browser, nor one kind of presentation. It should be fully re-usable, no matter how the technology is implementing your application. Computer systems are always renewing technology, so your e-business system should be fully suitable to new technologies.

### 2.3.6.3 Three-tier application design

A number of buzzwords were coined recently to describe the basic design for enterprise applications. In particular, the terms *three-tier* and *n-tier* are used to describe applications reside on clients and servers. As mentioned for stage zero, two-tier applications might connect a client to a Web server to request
information. Extending to a three-tier application allows programmers to leverage the use of resources such as processing routines on the server side. Figure 8 shows examples of two-tier and three-tier applications.

![Two-tier example](image)

**Two-tier example**

The application requests static information that remains on the Web server.

![Three-tier example](image)

**Three-tier example**

The application requests a Web server for processing, which requests routines on WebSphere Application Server, which requests data from a database server.

Figure 8. Two-tier and three-tier application examples

### 2.3.6.4 Recommended implementations

Up to this point, we explained the concepts behind e-business system stage one. We strongly recommend that you use a technology implementation to achieve a state-of-the-art e-business application.

**Session object**

A session is a connection between a client and a server where information is exchanged. You can follow the correct work flow of your Web application by using only parameters to track data between transactions. This could be a problem in maintaining source code when your application becomes bigger or has several interactions to track. You can prevent this by using a session object component of the Java specification, which allows you to use a standard method to track sessions. Session objects are not scrapped by the garbage collection function of the Java Virtual Machine (JVM) environment.
**Connection pools**

Connection pools are a common feature of application servers. The implementation described here is specific to the WebSphere Application Server. Connection pools are commonly used because creating a new Java Database Connectivity (JDBC) connection is time consuming and uses system resources. With a connection pool, WebSphere Application Server creates a connection object when the server starts. When a bean needs a connection to the database, it retrieves a connection object from the pool. When the bean is done with the connection, it returns the object to the pool.

2.3.7 Preparing to move to stage two

Before you move to stage two, you need to verify whether you are ready to make such a move. As mentioned before, it’s important to always implement technology with a good analysis and design method. If you don’t have the proper methodology, it's difficult to maintain and extend the functionality of your e-business system. In stage two, you deal with user information and personalized data that must be well understood and have a good platform, in terms of design, for successful implementation. If you don’t have a good application design, you could be forced to scrap the entire application and begin from scratch.

Make sure a security policy is in effect and working. It should be designed to deal with multiple devices and handle future security needs. U2B applications are available generally to different kinds of users, who are sometimes spread around the world.

Access growth and your internal business process are directly proportional to the quality of your e-business system. Be sure you have a good capacity plan and knowledge about your actual resources. If you have substantial access growth on your e-business system, be ready to attend to such a demand. Scalability is the best word to define e-business system needs. Choose technologies that can implement scalability easily.

Try to implement functions in your e-business system that use industry standards. Specifically-developed functions may appear to be good at first, but they could become a problem when technology vendors implement new products. This may result in twice the work to keep your e-business system updated.

2.3.8 When to move to the next stage

If you need to leverage your e-business system to the next level, you may need to:
• Protect, or make confidential, some information from users
• Provide dynamic contents changed for each user
• Collect information related to users
• Provide statistics about individual user's access
• Update data that will be accessed by other systems or users that have their own authentication, so you need a common authentication process to validate users

2.4 Stage two: User security approach

Your e-business application will almost always need specific user authentication or personalized information delivery. You can look at stage two as a preparation for stage three, in which you consider how you intend to control security and user information.

In stage two, you need to implement technology that allows you to share user information for authentication purposes and deliver personalized data before your e-business system is ready to share resources with other enterprise applications. This is especially true in regard to U2B e-business systems, where user information is the most important.

Currently, there are many e-business solutions that provide directory services and LDAP protocol to deal with user authentication and personalization. The solution that this redbook refers to is AS/400 Directory Services, which implements LDAP protocol as an IBM solution for directory services.

2.4.1 Directories: Base for authentication and personalization

Directories are the base of stage two. They can handle all the common information that one e-business system should have about users. They also share this information among other applications.

A directory is often described as a database. Rather, it is a specialized database that has characteristics that set it apart from general purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) more often than they are updated (written). For example, hundreds of people might look up an individual's phone number, or thousands of print clients might look up the characteristics of a particular printer. But, the phone number or printer characteristics rarely change.

Because directories must be able to support high volumes of read requests, they are typically optimized for read access. Write access might be limited to system administrators or to the owner of each piece of information. A general purpose database, on the other hand, needs to support applications, such as
airline reservations and banking applications, with relatively high update volumes.

Your directory information doesn’t always need to be available for anyone to access. You can use an Access Control List (ACL) to control user access to the directory.

2.4.1.1 What is an Access Control List (ACL)?
An Access Control List (ACL) allows you to manage who can access directory information in your network directory. In many cases, you may not want to restrict access to data on your LDAP directory server. For example, an LDAP server on your company intranet might contain a telephone directory of company employees. You might want all employees to be able to view the data in this directory. Imagine, however, that the president of your company does not want all employees to access their telephone number. In that case, you could create an ACL. With this ACL, you could restrict access to the president’s server entry to only those employees from which the president wanted to receive calls.

Why use directories
You can currently find several applications, using specific user data, on enterprises. In such an environment, each application creates and manages its own application-specific directory, which quickly becomes an administrative nightmare. The same e-mail address stored by the calendar application may also be stored by a mail application and by an application that notifies system operators of equipment problems. Keeping multiple copies of information up-to-date and synchronized is difficult especially when different user interfaces and even different system administrators are involved.

Besides that, your e-business system should use a common repository for user information that could be used by another enterprise application. To do that, you need a standard access method. AS/400 Directory Services allows you to share information among several applications with only one point of administration, as shown in Figure 9.
2.4.2 Personalization

Personalization can be defined as tailoring the contents and behavior of your e-business application directly to a specific user or group of users. You can achieve this goal by asking the user to provide information for your e-business application. Or you can track the user's behavior when they use the application, so the e-business system can modify the contents based on this information. In addition, you can stipulate business rules to tell your e-business system how to behave with specific users.

2.4.2.1 Actual personalization techniques

There are four approaches to personalization. Each is identified and explained in Table 2.

Table 2. Personalization techniques

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookies</td>
<td>A tag on the Web browser identifies users to the e-business system</td>
</tr>
<tr>
<td>Check box</td>
<td>Users check their interests from a checklist and receive proper information based on that information</td>
</tr>
<tr>
<td>Collaborative filtering</td>
<td>Tracks user behaviors and compares them to others with similar behavior</td>
</tr>
</tbody>
</table>
You can design your e-business system to allow a combination of personalization techniques. U2B e-business systems are becoming more personalized as technological improvements allow it.

2.4.2.2 Why use personalization
Here are two reasons why you should employ personalization in your e-business systems:

- Personalization technologies enable companies to collect and correlate data that constitutes new knowledge about emerging interests. It's one way to advance actions that handle user needs. The advantage is that all information is generated and updated by the user themself.
- Security can be enhanced with personalization. For example, a different user profile may access and update different information. This could be a system administrator user who accesses the same page but with different data than a user with standard access.

2.4.3 What you should know about stage two
User information is very important for User-to-Business applications. Consider the important points in the following sections.

2.4.3.1 Information on directories
Directories are meant to store relatively static information and are optimized for that purpose. They are not appropriate for storing information that changes rapidly. For example, the number of jobs currently in a print queue should not be stored in the directory entry for a printer because that information would have to be updated frequently to be accurate. Instead, the directory entry for the printer can contain the network address of a print server. The print server can be queried to provide the current queue length if desired. The information in the directory (the print server address) is static, where the number of jobs in the print queue is dynamic.

2.4.3.2 Personalization doesn’t mean less information
One good example is the weather forecast. For 95% of the time, people want the weather forecast for the area where they live (so it's unchanging). However, 5% of the time they want the weather for a different area.
Personalization doesn’t free the designer from providing a way to choose another location.

2.4.3.3 Scalability
If you create an e-business system based on technologies that don’t allow scalable growth, you can’t handle all the customer data and personalized information. In this case, delays are created, and your users won’t be happy with your system.

2.4.3.4 Flexibility
If you customize your e-business application too much, you can lose your application’s personality. Users increase productivity when applications have similar screens, with similar controls.

2.4.3.5 Group information
Don’t focus too much on creating many individual customizations. Design personalizations around groups that share common needs. If a group becomes too large, you should divide into a more precisely tuned assembly of user needs.

2.4.4 Preparing to move to stage three
Before you move to stage three, your e-business system must have:

- A well designed security policy (must be implemented already)
- An application design that reflects modular implementation
- A security foundation

In stage three, it is very important that your e-business system have improved efficient integration with different kinds of resources. Optimized use of such resources is only possible if you have a good application design. The design must isolate the functionality of application presentation, security, business rules, and database processing in a way that any stand-alone running function can interface with another system. If the previous stages are not important to you, it may be difficult to interface your e-business system with legacy applications.

At this point, all user information, access control rights, business rules, and user-related information should be sharable. It must also be applied to an e-business system design that can manage and maintain its integrity.

2.4.5 When to move to the next stage
If you need to leverage your e-business system to the next level, you may have such concerns as:
• You need to update data shared with another application, and the application has its own business rules.
• You need to run a function, and one of the steps of execution is owned by a legacy application.
• Your e-business system takes tremendous resources to process data that could be processed better by another application.
• Your application design demands a distributed workload.

2.5 Stage three: Enterprise approach

In stage three, the enterprise approach refers to e-business systems that, by their own design, need to access data over the entire enterprise. In this redbook, the e-business system must have direct access to the legacy applications and access to business logic using EJBs. This section provides definitions for legacy applications and tips to help you integrate older technologies with newer technologies.

2.5.1 Legacy definition

In information technology, legacy applications and data have been inherited from languages, platforms, and techniques prior to the current technology. Most enterprises that use computers have legacy applications and databases that serve critical business needs. Typically, the challenge is to keep the legacy application running, while converting it to newer, more efficient code, which uses new technology and programmer skills. Theoretically, this will make it easier in the future to update applications without having to rewrite them entirely.

In addition to moving to new languages, enterprises redistribute the locations of applications and data. In general, legacy applications continue to run on the platforms for which they were developed. Typically, new development environments are responsible for continuing to support legacy applications and data.

2.5.2 Enterprise application design

The AS/400e server is particular about development language. Most source code in use today was created on RPG. This redbook provides information about the RPG interface and EJB interface, which may be a target for most developers.

RPG is a typical business-oriented language, which is the same for the AS/400e server. That's why, even today, a lot of applications running on the
AS/400e server have been written in RPG. They continue to run because the programs that have been written are very stable and satisfy most business needs. However, with the advent of such new technologies as the Internet, Java, and application servers, corporations face the challenge of keeping strong business control over most recent technologies.

EJBs complement the MVC model, so you can design your application to be reusable for the entire enterprise. Figure 10 can help you to understand how a high quality e-business application design can take advantage of the MVC model.

Figure 10. Enterprise application design

Figure 10 clearly shows the division between each component:

- The View layer contains the HTML requests and the JSPs that are responsible for showing the presentation. This layer invokes a servlet through the HTTP protocol and waits for a JSP response.

- In the Controller layer, a servlet that is responsible for the workflow of executions always implements a SET method (input values), an EXECUTE method (processing), and a GET method (output values), which make a standard interface for others components. It instantiates a command to start communication with other resources. It is independent
of the kind of communication method you are using and makes a powerful implementation to develop enterprise applications.

- In the Model layer, the command has three kinds of interfaces to implement communication:
  - The JDBC communication method for direct access to the database
  - The EJB communication method to invoke an EJB
  - The RPG communication method to invoke an RPG program

These communication methods allow you to process enterprise data that implements the MVC model, leaving all code reusable for other applications. The Model layer command can run processing tasks and access the database only through JDBC. However, if you are thinking about reusability for the entire enterprise, the EJB model offers a better solution.

To access legacy applications, Java implementation on the AS/400e server provides Toolbox for Java, which enables you to interface with RPG programs. If your RPG program design has a standard interface similar to the SET, EXECUTE, and GET command interface, you can run RPG programs without any modification. Otherwise, you can make a few adaptations on your source code and enable RPG logic to receive and send information to your e-business application.

### 2.6 What the next stage should be

By understanding the concepts in this chapter, you can implement U2B e-business applications. The next stage should entail integrating enterprise applications with another enterprise application. Such integration may be as simple as passing data to a particular application, or as sophisticated as a workflow involving several applications and user interactions.
Chapter 3. Modifying the PDK application for OS/400

To better understand the Pattern Development Kit (PDK), you should use the appropriate tools to look inside the source code for all of the Patterns Development Kit's artifacts. This chapter explains how to modify the PDK. It gives you a high-level look at working with the PDK using VisualAge for Java and WebSphere Studio. You can find detailed information on using these tools in *WebSphere Studio and VisualAge for Java Servlet and JSP Programming*, SG24-5755.

### 3.1 Pattern Development Kit (PDK)

The Pattern Development Kit is designed to provide an overview of WebSphere Advanced Edition in a working environment. The PDK provides a single-machine setup with applications and product code. The kit is divided into sections, each showing a different aspect of the Web application:

- Section A shows Web access to a read-only DB2 database and a writable DB2 database.
- Section B shows custom authentication using an LDAP directory.
- Section C shows Web access to a read-only DB2 database. This section generates a dynamic menu based on a user profile stored in the LDAP directory.
- Section D shows Web interactivity with MQ and CICS back end applications.
- Section E shows Web access to an IMS and DB2 back end.
- Section F shows the EJB implementation of the command manager pattern.

For the sample code in this redbook, we modified an application from the IBM Pattern Development Kit (PDK). You can find more information about PDK at: [http://www.ibm.com/developerworks/patterns](http://www.ibm.com/developerworks/patterns)

### 3.2 Using legacy resources

Using the sample application, there are several ways to connect the OS/400 resources (database and RPG programs). Many customers want to build a Web application that communicates with legacy applications. This section explains how to communicate with legacy applications and the database.
On the OS/400 system, there are many programs written using the RPG or CL language. An application written to WebSphere Application Server (WAS) can use these resources as well as Java programs. To access these resources, OS/400 provides the Java environment development tool – AS/400 Toolbox for Java (5769JC1). AS/400 Toolbox for Java is a library of Java classes that give Java programs easy access to AS/400e data and resources. The Toolbox does not require additional client support over and above what is provided by the Java Virtual Machine and JDK.

When you use AS/400 Toolbox for Java, you can communicate with the OS/400 commands and programs. AS/400 Toolbox for Java provides support that is similar to functions that are available when using the Client Access/400 APIs. It uses the OS/400 host servers that are part of the base OS/400 operating system to access data and resources on an AS/400e server. Each of these servers runs in a separate job on the AS/400e server and communicates with a Java client program using data streams on a socket connection. The socket interfaces are hidden from the Java programmer by the AS/400 Toolbox for Java classes. Since the socket connections are managed by Toolbox classes, Client Access/400 is not needed on the workstation.

For more details about the AS/400 Toolbox for Java, refer the Web site: http://www.iseries.ibm.com/toolbox/

3.3 Using the Pattern Development Kit in VisualAge for Java

If you are interested in the Java source code (servlets, beans, and other classes) of the PDK, use VisualAge for Java to view and modify the code.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>You must complete the following steps to access the PDK’s Java source code inside the IDE of VisualAge for Java.</td>
</tr>
</tbody>
</table>

1. Add the following features to the workspace:
   - Data Access Beans 3.0
   - IBM Common Connector Framework 3.0
   - IBM EJB Development Environment 3.0
   - IBM Enterprise Access Builder Library 3.0
   - IBM Enterprise Data Access Libraries 3.0
   - IBM IDE Utility class libraries 3.0
   - IBM Java Record Library 3.0
Chapter 3. Modifying the PDK application for OS/400

- IBM WebSphere Test Environment 3.0
- IBM Enterprise Toolkit for AS/400 3.0
- Sun Servlet API 2.1

2. Locate the VisualAge for Java repository file, called PDK400.dat.

3. Import the Pattern Development Kit project from the PDK400.dat repository file into the VisualAge for Java repository.

4. Add the PDK400 project to the workspace.

3.3.1 Communicating with RPG programs

RPG offers high productivity and optimizes high performance running in an ILE environment. OS/400 customers have many RPG resources for their business applications.

This section explains the RetrieveHistoricalDataCommandRPG class. This class implements the command model and has an interface to the RPG program RTVHSTDTA. This class has the same properties as the RetrieveHistoricalDataCommandJDBC class. This command can be exchanged without any change on the previous controller and views. This is one advantage for using the MVC architecture. Figure 11 shows the class diagram for the RetrieveHistoricalDataCommandRPG class.

![Class diagram for the RetrieveHistoricalDataCommandRPG class](image)

The RTVHSTDATA RPG program communicates with the Java servlets pass parameter. This parameter is described in the command interface class (RetrieveHistoricalDataCommandRPG). In this sample, RTVHSTDATA declares the parameter list in the RPGLE program:
In this sample, we use the ProgramCall class to connect to the RPG program as shown in the following code example. You can use the data queue and Program Call Markup Language (PCML) to communicate with the RPG program using AS/400 Toolbox for Java.

```java
public void execute() throws com.ibm.rochester.sg245999.common.CommandException {
    System.out.println("execute method start!!!");
    HistoricalData temp = null;
    try{
        System.out.println(CommonVar.getSystemName() + CommonVar.getUserID() +
            CommonVar.getPassword());
        AS400 as400 = new AS400(CommonVar.getSystemName(),CommonVar.getUserID(),CommonVar.getPassword());
        AS400.disconnectAllServices();
        System.out.println(as400.getRelease());
        as400.connectService(AS400.COMMAND);
        as400.setCcsid(65535);
        System.out.println("Connection Get!!!");
        ProgramCall pgm = new ProgramCall(as400);
        System.out.println("Create Program Object");
        AS400Text planet_name = new AS400Text(30);
        AS400Text str_date = new AS400Text(10);
        AS400Text end_date = new AS400Text(10);
        AS400Text names = new AS400Text(5000);
        AS400Text temperature = new AS400Text(5000);
        AS400Text humidity = new AS400Text(5000);
        AS400Text windspeed = new AS400Text(5000);
        AS400Text local_time = new AS400Text(5000);
        AS400Text local_date = new AS400Text(5000);
        ProgramParameter[] parmlist = new ProgramParameter[9];
        parmlist[0] = new ProgramParameter(planet_name.toBytes(this.getPlanetName()),30);
        parmlist[1] = new ProgramParameter(str_date.toBytes(this.getStartDate()),10);
        parmlist[2] = new ProgramParameter(end_date.toBytes(this.getEndDate()),10);
        parmlist[3] = new ProgramParameter(names.toBytes(""),5000);
        parmlist[4] = new ProgramParameter(temperature.toBytes(""),5000);
        parmlist[5] = new ProgramParameter(humidity.toBytes(""),5000);
        parmlist[6] = new ProgramParameter(windspeed.toBytes(""),5000);
        parmlist[7] = new ProgramParameter(local_time.toBytes(""),5000);
        parmlist[8] = new ProgramParameter(local_date.toBytes(""),5000);
        System.out.println("Ready to call");
        pgm.setProgram(CommonVar.getRpgPgm(),parmlist);
        System.out.println( "pgm.run()" );
        AS400Message[] messagelist = pgm.getMessageList();
        System.out.println( "messagelist.length");
        for (int i=0;i < messagelist.length ;i++) {
            System.out.println(messagelist[i]);
        }
    }
    catch (Exception e) {
        System.err.println("Exception: "+e.getMessage());
    } finally {
        System.out.println("execute method end!!!");
        as400.disconnectAllServices();
    }
}
```

User-to-Business Patterns for e-business
pgm.setProgram(CommonVar.getRpgPgm(), parmlist);
System.out.println(pgm.run());
AS400Message[] messagelist = pgm.getMessageList();
System.out.println(messagelist.length);
for (int i=0; i < messagelist.length; i++) {
    System.out.println(messagelist[i]);
}
System.out.println(String.valueOf(names.toObject(parmlist[3].getOutputData(), 0)));
String names_arr[] = RetrieveHistoricalDataCommandRPG.reOrg(String.valueOf(names.toObject(parmlist[3].getOutputData()[0])));
System.out.println(names_arr.length);
for(int i=0; i < names_arr.length; i++){
    System.out.println("names" + names_arr[i]);
}
System.out.println((String)valueInObject(parmlist[4].getOutputData(), 0));
String temp_arr[] = RetrieveHistoricalDataCommandRPG.reOrg((String)valueInObject(parmlist[4].getOutputData()[0]));
System.out.println(temp_arr.length);
for(int i=0; i < temp_arr.length; i++){
    System.out.println("temp" + temp_arr[i]);
}
System.out.println(String.valueOf(humidity.toObject(parmlist[5].getOutputData(), 0)));
String humidity_arr[] = RetrieveHistoricalDataCommandRPG.reOrg(String.valueOf(humidity.toObject(parmlist[5].getOutputData()[0])));
System.out.println(humidity_arr.length);
for(int i=0; i < humidity_arr.length; i++){
    System.out.println(humidity_arr[i]);
}
System.out.println(String.valueOf(windspeed.toObject(parmlist[6].getOutputData(), 0)));
String windspeed_arr[] = RetrieveHistoricalDataCommandRPG.reOrg(String.valueOf(windspeed.toObject(parmlist[6].getOutputData()[0])));
System.out.println(windspeed_arr.length);
for(int i=0; i < windspeed_arr.length; i++){
    System.out.println(windspeed_arr[i]);
}
System.out.println(String.valueOf(local_date.toObject(parmlist[7].getOutputData(), 0)));
String local_date_arr[] = RetrieveHistoricalDataCommandRPG.reOrg(String.valueOf(local_date.toObject(parmlist[7].getOutputData()[0])));
System.out.println(local_date_arr.length);
for(int i=0; i < local_date_arr.length; i++){
    System.out.println(local_date_arr[i]);
}
System.out.println(String.valueOf(local_time.toObject(parmlist[8].getOutputData(), 0)));
String local_time_arr[] = RetrieveHistoricalDataCommandRPG.reOrg(String.valueOf(local_time.toObject(parmlist[8].getOutputData()[0])));
System.out.println(local_time_arr.length);
for(int i=0; i < local_time_arr.length; i++){
    System.out.println(local_time_arr[i]);
}
historicalData = new Vector();
for(int i=0; i < local_time_arr.length; i++){
    temp = new HistoricalData();
    System.out.println(i);
    temp.setHumidity(humidity_arr[i]);
    historicalData.add(temp);
}
temp.setLocalDate(local_date_arr[i]);
temp.setLocalTime(local_time_arr[i]);
temp.setPlanetName(names_arr[i]);
temp.setTemperature(temp_arr[i]);
temp.setWindSpeed(windspeed_arr[i]);
historicalData.addElement(temp);
}
as400.disconnectAllServices();
}catch(Exception e){
e.printStackTrace();
}
}

3.3.1.1 RPG program component test

As you know, the RPG program can be tested with a traditional method. The easiest way is to use a CL program, instead of a Java program. The following code shows the sample CL program source for the RTVHSTDATA RPG program. This CL program declares the parameter list and sends the output to the program message.

DCL VAR(&USER) TYPE(*CHAR) LEN(10)
DCL VAR(&PLANET) TYPE(*CHAR) LEN(30) VALUE('Moon')
DCL VAR(&STR_DATE) TYPE(*CHAR) LEN(10) +
   VALUE('2000-01-01')
DCL VAR(&END_DATE) TYPE(*CHAR) LEN(10) +
   VALUE('2001-01-01')
DCL VAR(&NAME) TYPE(*CHAR) LEN(5000) VALUE(' ')
DCL VAR(&TEMP) TYPE(*CHAR) LEN(5000) VALUE(' ')
DCL VAR(&HUMIDITY) TYPE(*CHAR) LEN(5000) VALUE(' ')
DCL VAR(&WINDSPEED) TYPE(*CHAR) LEN(5000) VALUE(' ')
DCL VAR(&LOCAL_DATE) TYPE(*CHAR) LEN(5000) VALUE(' ')
DCL VAR(&LOCAL_TIME) TYPE(*CHAR) LEN(5000) VALUE(' ')
RTVJOBA USER(&USER)
CALL PGM(RTVHSTDATA) PARM(&PLANET &STR_DATE +
   &END_DATE &NAME &TEMP +
   &HUMIDITY &WINDSPEED &LOCAL_DATE &LOCAL_TIME)
SNDMSG MSG(\$SUBSTRING(&NAME 1 50)) +
   TOMSQQ(QUSRSYS/USER) 
SNDMSG MSG(\$SUBSTRING(&TEMP 1 50)) +
   TOMSQQ(QUSRSYS/USER) 
SNDMSG MSG(\$SUBSTRING(&HUMIDITY 1 50)) +
   TOMSQQ(QUSRSYS/USER) 
SNDMSG MSG(\$SUBSTRING(&WINDSPEED 1 50)) +
   TOMSQQ(QUSRSYS/USER) 
SNDMSG MSG(\$SUBSTRING(&LOCAL_DATE 1 50)) +
   TOMSQQ(QUSRSYS/USER) 
SNDMSG MSG(\$SUBSTRING(&LOCAL_TIME 1 50)) +
   TOMSQQ(QUSRSYS/ITSCID60) }

You can test the RTVHSTDATA function using this CL program. Figure 12 shows the result of running the CL program.
3.3.2 Using EJB technology

Sun Microsystems Enterprise JavaBean (EJB) technology provides the means to make Java a viable language for AS/400e business application development. It focuses on the server-side Java. EJB technology returns the focus of the application developer to the business application. The developer does not deal with system-level services. The AS/400e server provides a secure and scalable environment for running Java applications, while EJB application servers provide server and container support.

The EJB architecture provides a component model for server applications. With EJB, you achieve rapid application development and the rich graphical interfaces of a client. And you don’t have to sacrifice thin client manageability and the security of a server. EJB accomplishes this by making it easy to partition an application to a user interface and the business logic. The user interface can be specified in a Java applet, in a client application (written in Java, Visual Basic, and so on), or in HTML using a Java servlet. The server-side business logic is packaged as EJB components. EJB are easily deployed anywhere on the network, reused within other business applications running on disparate platforms, and easily managed from a remote console.
EJB technology is not a tool to build applications. Rather, it is the architecture for defining components that can be used with a variety of tools. A stated goal of the EJB specification is to be the standard component architecture for building distributed object-oriented business applications in the Java programming language. EJB makes it possible to build distributed applications by combining components that are developed using tools from different vendors.

3.3.2.1 WebSphere Application Server and EJB

The IBM implementation of the Enterprise JavaBeans (EJB) specification enables users of WebSphere Application Server Advanced Edition to integrate their Web-based systems with their other business systems. A major part of this implementation is the WebSphere EJB server and its associated components, which are illustrated in Figure 13.

![Figure 13. Components of the EJB environment](image)

The WebSphere EJB server environment contains the following components:

- **EJB server**: A WebSphere EJB server contains and runs one or more enterprise beans, which encapsulate the business logic and data used and shared by EJB clients. The enterprise beans installed in an EJB server do not communicate directly with the server. Instead, an EJB
container provides an interface between the enterprise beans and the EJB server. This provides many low-level services such as threading, support for transactions, and management of data storage and retrieval.

- **Data source**: There are two types of enterprise beans:
  - *Session beans*: Encapsulate short-lived, client-specific tasks and objects.
  - *Entity beans*: Encapsulate permanent or persistent data. The EJB server stores and retrieves this persistent data in a data source, which can be a database, another application, or even a file.

- **EJB clients**: There are two general types of EJB clients:
  - *HTTP-based clients*: Interact with the EJB server by using either Java servlets or JavaServer Pages (JSP) via the Hypertext Transfer Protocol (HTTP).
  - *Java applications*: Interact directly with the EJB server by using Java remote method invocation over the Internet Inter-ORB Protocol (RMI/IIOP).

- **Administration interface**: The administrative interface allows you to manage the EJB server environment.

The WebSphere Application Server also provides the integrated development test and deployment.

For more information, see the Web site at: [http://www.ibm.com/websphere](http://www.ibm.com/websphere)

### 3.3.2.2 EJB consideration

This section explains a consideration for the EJB develop environment using VisualAge for Java.

**Database mapping**

For database maps, you can use schema maps and then run EJB in any environment more easily.

A *schema* defines how to persist any data using a relational database (RDB). The EJB specification allows two ways to persist entity beans:

- Container-managed persistence (CMP)
- Bean-managed persistence (BMP)

With BMP, the developer is responsible for writing code (usually JDBC) to store, load, create, and remove the bean’s state. You must make sure all tables and fields are read and written properly. Also, make sure your bean accurately reflects the data in the persistent store. This gives you control over...
the inner workings of your bean, but it can be a tedious, boring, and error-prone task.

Of course, anytime you have a tedious and error-prone task, you should look for someone or something to perform that task for you. In EJB, that “something” is the container. When you allow the container to take responsibility for maintaining the database (with CMP), you give relief to the developer from having to do the task.

An advantage of EJB is that it gives you the ability to separate application logic from deployment and any implementations that depend on the environment.

VisualAge for Java provides the map browser, as shown in Figure 14, to browse database maps.

Figure 14. Schema map browser

The sample application uses CMP. You need to set up the environment for EJB in the WebSphere application server administrative console as shown in 4.6.1.5, “EJB” on page 146. After this is setup, you can see the default container as shown in Figure 15.
You can test any EJB using the VisualAge test environment. You can use the local database for the test. You can create the test database more easily by using the schema explained in “Database mapping” on page 41.

EJB use the RMI/IIOP to communicate with the servlet, JSP, or Java application client. You can access anywhere you want. It is good for porting an application, can be extended to a distributed environment, and test all functions of EJB from your development workstation. VisualAge for Java provides a powerful test environment. You can test EJB deployed on WebSphere and EJB on VisualAge using your own test environment.

This section explains how to test EJB from the sample application. Once you generate the deployed classes for your enterprise beans, you can create an EJB server configuration. This sample application provides a U2B EJB server configuration. Follow this process:

1. In the EJB server configuration, select U2B in the Enterprise Beans pane of the EJB display. Right-click U2B, and select Open To -> Server configuration. The window that appears is shown in Figure 16 on page 44.

Figure 15. Container configuration

Unit test environment for EJB
You can test any EJB using the VisualAge test environment. You can use the local database for the test. You can create the test database more easily by using the schema explained in “Database mapping” on page 41.

EJB use the RMI/IIOP to communicate with the servlet, JSP, or Java application client. You can access anywhere you want. It is good for porting an application, can be extended to a distributed environment, and test all functions of EJB from your development workstation. VisualAge for Java provides a powerful test environment. You can test EJB deployed on WebSphere and EJB on VisualAge using your own test environment.

This section explains how to test EJB from the sample application. Once you generate the deployed classes for your enterprise beans, you can create an EJB server configuration. This sample application provides a U2B EJB server configuration. Follow this process:

1. In the EJB server configuration, select U2B in the Enterprise Beans pane of the EJB display. Right-click U2B, and select Open To -> Server configuration. The window that appears is shown in Figure 16 on page 44.
2. After you verify the test environment, start the test client. Select **U2B** in the Enterprise Beans pane of the EJB page. Right-click **Planets**, and select **Run Test Client** as shown in Figure 17.

3. After you connect the IIOP server, you see the Planets test client window (Figure 18). Select **findByPlanetName(String)** in the method pane, and type **Mars** in the parameter pane.
Figure 18. Testing the CMP home interface

4. Click the **Send** button, and you see the planet ID in the information area.

You can use HistoricalDataManager as an independent application to use Test Client (Figure 19). It means the EJB model is isolated from other layers and can be used from any other type of client.

Figure 19. Testing the EJB business models as an independent application
### 3.3.2.3 Design tips

When you develop EJB, consider the responsibility that each class should have. In this sample application, there are three EJBs:

- **Planet**: This is an entity bean that is responsible for managing the planet database. It can respond to the planet name by using the `findByPrimaryKey` method. It can also respond to the planet ID by using the `findByPlanetName` method.

- **Weather**: This is an entity bean that is responsible for managing the weather database. It can respond to the weather information by using the weather key (this method is `findByPrimaryKey()`). It can also respond to the weather information by using planet Name, startDate, and endDate (this method is `findTargetData()`).

- **HistoricalDataManager**: This is a session bean. HistoricalDataManager only has business logic.

![Figure 20. EJB (business logic) class diagram](image-url)
3.4 Using the PDK in WebSphere Studio

If you are interested in the Web site source code of the Pattern Development Kit (the HTML files, JSP files, and image files), use WebSphere Studio as the tool to view and modify the code.

3.4.1 Using Pagelist with RPG

This section offers tips for using PageListServlet to connect to the RPG program.

IBM provides the PageListServlet class in the com.ibm.servlet package. This is a subclass of HttpServlet, which provides a callPage method to invoke JSP. Servlets generated by the WebSphere Studio wizards are subclasses of the PageListServlet class. Such a servlet must have an associated servlet configuration file (.servlet) that specifies all the possible JSP that the servlet may invoke.

A typical call to invoke a JSP from a PageListServlet is:

```java
callPage("retry", request, response);
```

The name of the JSP can be a short name (alias) that is assigned to the real file name of the JSP in the servlet configuration file, which is the hstdataRPG.servlet file shown in the following example:

```xml
<?xml version="1.0"?>
<!-- This file was generated by IBM WebSphere Studio using C:\WebSphere\Studio\BIN\GenerationStyleSheets\AppServerV3\JSP1.0\WebSphere\ServletConfig.xsl-->
<servlet>
  <init-parameter name="commandType1" value="EJB" />  
  <init-parameter name="commandType2" value="RPG" />
  <page-list>
    <default-page>
      <uri>/pdk400/stage3/RetrieveHistoricalDataServletResults.jsp</uri>
    </default-page>
    <error-page>
      <uri>/pdk400/stage3/RetrieveHistoricalDataServletError.jsp</uri>
    </error-page>
  </page-list>
</servlet>
```

Note

If you need to use any user-defined classes, you must include its class definition to the deployment JAR file. In this sample program, the vector array is used to pass to the Command Bean (RetrieveHistoricalDataCommandEJB) from EJB.
Because the JSP names used in the callPage method of the servlet are aliases, a change of directory can be accomplished by changing the servlet configuration file, without touching the servlet code.

In this sample application, the PageListServlet class calls RPG programs. The following section explains how to develop this sample application using WebSphere Studio. The steps to develop the application are:

1. Create the input form, servlet as the iteration controller, Result form, and servlet control files using the JavaBean Wizard.
2. Change the RetrieveHistoricalDataServlet.java file.
3. Change the view.

3.4.1.1 Using the WebSphere Studio JavaBean Wizard

This sample application has a ResultBean, ViewBean, command interface, and application sequence. Using the JavaBean Wizard, you can create the input form, servlet as the iteration controller, Result form, and servlet control files from ViewBean (TableView) as explained here:

1. In the WebSphere Studio window, select stage3 under PDK400. In the menu bar, select Tools -> Wizards -> JavaBean Wizard (Figure 21).

![Figure 21. Starting the JavaBean Wizard](image)
2. You can see the JavaBean Wizard display as shown in Figure 22. Select the JavaBean that you want to add to the Web page. In this sample application, we selected TableView.class.

![JavaBean Wizard](image)

*Figure 22. JavaBean Wizard*

3. Click **Next** on the JavaBean Wizard display. You can see the Web Pages display (Figure 23). In this display, select the pages that you want to create. In this sample, we selected all options. Click **Next**.

![Web Pages](image)

*Figure 23. Web Pages display*
4. In the Input Page display (Figure 24), click the Next button to skip this display. In this case, no date is required for the Input Page.

![Input Page display](image)

**Figure 24. Input Page display**

5. In the Result page display (Figure 25), select the table to appear in Results Page. Click **Next**.

![Result Page display](image)

**Figure 25. Selecting a table in the Results Page display**

6. In the Standard Error Page display (Figure 26), create a new error page with a comment. Click **Next**.

![Standard Error Page display](image)

**Figure 26. Standard Error Page display**
Figure 26. Creating the Error Page

7. In the Methods display (Figure 27), click Next to skip this display. In this sample, we do not need to call any method.

Figure 27. Selecting the method

8. On the Session display (Figure 28 on page 52), select No. Specify the name as tableView for using this bean on more than one Web page. Click Next.
9. Click the **Rename** button. Change the package name and the prefix to `RetrieveHistoricalServlet` on the Finish display as shown in Figure 29.

After this step, you can see the new Java source, class file, and servlet configuration file under the stage3/servlet directory, and one HTML file and two JSP files under the stage3 directory as shown in Figure 30.
3.4.1.2 histdataRPG Controller development

The steps in 3.4.1.1, “Using the WebSphere Studio JavaBean Wizard” on page 48, provided the skeleton code (RetrieveHistoricalDataServlet.java) shown in the following example. You can change this code using the VisualAge for Java IDE.

```java
public void performTask(HttpServletRequest request, HttpServletResponse response)
{
    try
    {
        // instantiate the beans and store them so they can be accessed by
        // the called page
        com.ibm.rochester.sg245999.common.JTable tableView = null;
        tableView = (com.ibm.rochester.sg245999.common.JTable)
            java.beans.Beans.instantiate(getClass().getClassLoader(),
                "com.ibm.rochester.sg245999.common.JTable");
        setRequestAttribute("tableView", tableView, request);
        // Call the output page. If the output page is not passed
        // as part of the URL, the default page is called.
        callPage(getPageNameFromRequest(request), request, response);
    }
    catch (Throwable theException)
    {
        // uncomment the following line when unexpected exceptions are occurring
    }
}
```

Figure 30. Results WebSphere Studio window
The following steps explain how to move the source to VisualAge for Java:

1. **Send RetrieveHistoricalDataServlet.java and the class file that you will change using VisualAge for Java.**
   
   To do so, select `RetrieveHistoricalDataServlet.java` and the class in the left frame of WebSphere Studio. In the menu bar, select **Projects -> VisualAge for Java -> Send to VisualAge** as shown in Figure 31.

   ![Exporting to VisualAge for Java](image)

   **Figure 31. Exporting to VisualAge for Java**

2. **The first time you send this code to VisualAge for Java, you need select the project to which you want to add this code (Figure 32).** In this sample, we chose PDK400. After this step, you can modify this code by using VisualAge for Java.
After you change the code, update it by using the update function in WebSphere Studio. We explain this process in the following section.

3.4.1.3 Changing the view
The JavaBean Wizard created one HTML source (input page) and two JSP sources (result and error). You can change this output by using WebSphere Page Designer. In this sample application, we change the input HTML page and result JSP.

In the left frame of WebSphere Studio, select **RetrieveHistoricalServletResults.jsp**. From menu bar, select **Tools -> Edit with -> PageDesigner**. You can see the Page Designer window as shown in Figure 33 on page 56.

![Figure 32. Selecting a project](image)
For this JSP, you delete the table text as indicated in Figure 33. The Designer source code for the RetrieveHistoricalServletResults.jsp file is shown here:

```html
<br>
<% tableView.setRequest(request); %>
<TABLE border="0">
<%=tableView.getTable() %>
</TABLE>
</BODY>
</HTML>
```

The input form is created as an HTML file in our sample. For the Results form, you start Page Designer for the RetrieveHistoricalServletInput.html file. An example of the window is shown in Figure 34.
In our sample application, we gave conditions for searching the weather information. We changed the HTML file to the JSP file as shown in Figure 35 on page 58. The Designer source code for the RetrieveHistoricalServletInput.jsp file is shown on page 56.
3.4.1.4 Publishing and setting up for WebSphere Application Server

After you change the sources that were created by the JavaBean Wizard, you should publish the files to the AS/400e server. This section explains how to publish the files and set up the WebSphere Application Server.

WebSphere Studio does not support a WebSphere application alias. You should change the servlet configuration file by yourself. Also you have to register the servlet to your WebSphere Application Server. Follow these steps:

1. The servlet configuration file (RetrieveHistoricalDataServlets.servlet) that was created by the JavaBean Wizard is shown here as the PageList Control file (in XML format):

```xml
<?xml version="1.0"?>
<!-- This file was generated by IBM WebSphere Studio using C:\WebSphere\Studio\BIN\GenerationStyleSheets\AppServerV3\JSP1.0 \WebSphere\ServletConfig.xsl-->
<servlet>
<page-list>
<default-page>
<uri>/stage3/RetrieveHistoricalDataServletResults.jsp</uri>
</default-page>
<error-page>
```

Figure 35. Using Page Designer for the Input form
2. This control file is configured to work on the default server root path. In this sample application, we located the servlet under the PDK400 directory. We registered the RetrieveHistoricalDataServlet as hstdataRPG. We changed the file name as hstdataRPG.servlet and the path name as /pdk400 as shown here:

```xml
<?xml version="1.0"?>
<!-- This file was generated by IBM WebSphere Studio using
 C:\WebSphere\Studio\BIN\GenerationStyleSheets\AppServerV9\JSP1.0\WebSphere\ServletConfig.xsl-->
<servlet>
    <page-list>
        <default-page>
            <uri>/pdk400/stage3/RetrieveHistoricalDataServletResults.jsp</uri>
        </default-page>
        <error-page>
            <uri>/pdk400/stage3/RetrieveHistoricalDataServletError.jsp</uri>
        </error-page>
    </page-list>
    <code>com.ibm.rochester.sg245999.stage3.servlet.RetrieveHistoricalDataServlet</code>
</servlet>

3. Add a new page in this configuration file.

In this sample application, we added a retry form. It is named “retry” as shown in the following example. Most of the retry form is the same as the input form. In this sample application, the input form and retry form are shared. This is the reason to change the name from RetrieveHistoricalDataServletInput.html to RetrieveHistoricalDataServletInput.jsp.

```xml
<?xml version="1.0"?>
<!-- This file was generated by IBM WebSphere Studio using
 C:\WebSphere\Studio\BIN\GenerationStyleSheets\AppServerV9\JSP1.0\WebSphere\ServletConfig.xsl-->
<servlet>
    <page-list>
        <default-page>
            <uri>/pdk400/stage3/RetrieveHistoricalDataServletResults.jsp</uri>
        </default-page>
        <error-page>
            <uri>/pdk400/stage3/RetrieveHistoricalDataServletError.jsp</uri>
        </error-page>
        <page>
            <uri>/pdk400/stage3/RetrieveHistoricalDataServletInput.jsp</uri>
            <page-name>retry</page-name>
        </page>
    </page-list>
</servlet>
```
Now, you can call these pages from servlets using these methods:

- default page
- callPage(getPageNameFromRequest(request), request, response);
- error page
- handleError(request, response, theException);
- Using page name
- callPage("page name", request, response)

4. Before you publish the file, move the hstdataRPG.servlet file to pdk400/servlet as shown in Figure 36. The JavaBean Wizard creates the original file of hstdataRPG (name is RetrieveHistoricalDataServlet.servlet) that was under the pdk400/servlet/com/ibm/rochester/sg245999/stage3/servlet directory.

5. You can see the relationship in the right frame of the WebSphere Studio window as shown in Figure 37.
At first, you see the relationship in the physical location. After the configuration, you see the URL relationships. Note that the new page information is automatically related and displayed.

6. You receive the RetrieveHistoricalDataServlet.java and class file that you changed using VisualAge for Java in 3.4.1.3, “Changing the view” on page 55. Select RetrieveHistoricalDataServlet.java and the class in the left frame of WebSphere Studio. In the menu bar, select Projects -> VisualAge for Java -> Update from VisualAge as shown in Figure 38.

7. Right-click the pdk400 projects. Select Publish whole Project as shown in Figure 39 on page 62.
3.4.2 Adding data using the RPG program

This section explains how to enhance the application using the WebSphere Studio wizard. You can see the benefit of the MVC model. In your environment before you develop the application, you should complete the MVC design and prepare the business model. This section defines the business model as explained in 3.4.2.1, “Adding the model” on page 62.

3.4.2.1 Adding the model

The AddHistoricalDataCommandRPG class shown in Figure 40 is already prepared as the business model. Some classes invoke the execute() method of AddHistoricalDataCommandRPG with the value of the parameters (access_level, humidity, local_date, planet_name, temperature, and windspeed). This class calls the ADDHSTDTA RPG program, and this program adds the date to the database.
Chapter 3. Modifying the PDK application for OS/400

Figure 40. AddHistoricalDataCommandRPG class diagram

You have to define the interface between the controller and the view. The ViewBean is suitable for this work. In the following steps, create the AddHistoricalDataView as ViewBean:

1. In the VisualAge for Java workbench, select `com.ibm.rochester.sg245999.common` under the PDK400 projects. Right-click, and select Add -> Class... as shown in Figure 41.
2. The Create Class SmartGuide window appears. In this window, specify the class name and super class as shown in Figure 42. Click **Finish** to continue.

![SmartGuide window](image.png)

*Figure 42. Creating AddHistoricalDataView class using the SmartGuide*

3. You see the **AddHistoricalDataView** class under `com.ibm.rochester.sg245999.common` in the workbench. Right-click this class, and select **Add->Field** as shown in Figure 43.
4. In the SmartGuide for Create Field (Table 3), specify the field name and type as much in the ADDHSTDTA RPG program as shown in Figure 44 on page 66.

Table 3. Adding the field name and type

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field type</th>
</tr>
</thead>
<tbody>
<tr>
<td>planet_name</td>
<td>String</td>
</tr>
<tr>
<td>temp</td>
<td>String</td>
</tr>
<tr>
<td>humidity</td>
<td>String</td>
</tr>
<tr>
<td>windspeed</td>
<td>String</td>
</tr>
<tr>
<td>local_data</td>
<td>String</td>
</tr>
<tr>
<td>local_time</td>
<td>String</td>
</tr>
<tr>
<td>access_level</td>
<td>String</td>
</tr>
<tr>
<td>reading_id</td>
<td>int</td>
</tr>
</tbody>
</table>
Figure 44. Creating a field using SmartGuide

The class diagram is shown in Figure 45.

Figure 45. AddHistoricalDataView (class diagram)

3.4.2.2 Creating the view component

As explained in 3.4.1.1, “Using the WebSphere Studio JavaBean Wizard” on page 48, you can create the input form, servlet as iteration controller, Result form, and servlet control files from ViewBean (AddHistoricalDataView). To do so, follow these steps:

1. Before you use the JavaBean Wizard, import the AddHistoricalDataView class file to WebSphere Studio. In the WebSphere Studio window, select
the servlet under the PDK400 project. Right-click, and select **Insert -> File...** as shown in Figure 46.

![Figure 46. Importing AddHistoricalDataView](image)

2. The Insert File dialog appears. In this dialog, click the **From External Source** tab, and select **VisualAge for Java** as shown in Figure 47. Click the **Browse** button on the display that appears.

![Figure 47. Insert File dialog](image)

---

**Chapter 3. Modifying the PDK application for OS/400**
3. From the VisualAge dialog, select **AddHistoricalDataView**, and click **OK** as shown in Figure 48.

![Figure 48. Browsing the target class](image)

4. In the WebSphere Studio window, select **stage3** on the **PDK400** project. In the menu bar, select **Tools ->Wizards->JavaBean Wizard**.

5. The JavaBean Wizard display appears as shown in Figure 49. Select your JavaBean that you want to add to the Web page. In this sample application, we selected the AddHistoricalDataView class.

![Figure 49. Selecting AddHistoricalDataView](image)
6. Click **Next** on the JavaBean Wizard display. You can see the Web Pages display (Figure 50). In this display, select the pages that you want to create. In this sample, we selected all options. Click the **Next** button.

![Web Pages Display](image)

**Figure 50. Selecting the form in the Web Pages display**

7. On the Input Page display (Figure 51), click the **Next** button to skip this display. In our sample application, we selected all options, except `reading_id`. Click **Next**.

![Input Page Display](image)

**Figure 51. Selecting the fields for the Input Page**
8. In the Results Page display (Figure 52), select the reading_id table. Click Next.

![Figure 52. Selecting the fields for the Results Page](image)

9. In the Standard Error Page display, create a new error page with a comment. Click Next.

10. On the Methods display, click Next to skip this display. In our sample, we do not need to call any method.

11. On the Session display shown in Figure 53, select No. Specify the name as addHistoricalDataView to use this bean on more than one Web page. Click Next.
Chapter 3. Modifying the PDK application for OS/400

12. On the Finish display, click **Rename**. Change the name prefix to `AddHistoricalDataServlet`. The package name is shown in the **Rename** dialog in Figure 54.

![Session configuration](image1)

**Figure 53. Session configuration**

After you complete these steps, you can see the new Java source, class file, and servlet configuration file under the `stage3/servlet` directory. You can also see three JSP files under the `stage3` directory as shown in Figure 55 on page 72.

![Package name, class name, and form name](image2)

**Figure 54. Package name, class name, and form name**
3.4.2.3 AddHstData controller development
The JavaBean Wizard provides the skeleton code (AddHstoricalDataServlet.java). You can change this code using VisualAge for Java IDE. Follow these steps:

1. Send AddHistoricalDataServlet.java and the class file that you want to change using VisualAge for Java. Select AddHistoricalDataServlet.java and the class in the left frame of WebSphere Studio. In the menu bar, select **Projects -> VisualAge for Java -> Send to VisualAge** as shown in Figure 56.
2. The first time you send this code to VisualAge for Java, you need to select the project that you want to add to this code (see Figure 32 on page 55). For this sample, we chose PDK400.

   In VisualAge for Java, add the following function code in the performTask member of AddHistoricalDataServlet.java:

   ```java
   /* from here it is for customize */
   // Prepare Command bean
   addCommand = new com.ibm.rochester.sg245999.stage3.command.AddHistoricalDataCommandRPG();
   // Set required values from view bean
   addCommand.setAccess_level(addHistoricalDataView.getAccess_level());
   addCommand.setHumidity(addHistoricalDataView.getHumidity());
   addCommand.setLocal_date(addHistoricalDataView.getLocal_date());
   addCommand.setLocal_time(addHistoricalDataView.getLocal_time());
   addCommand.setPlanet_name(addHistoricalDataView.getPlanet_name());
   addCommand.setTemperature(addHistoricalDataView.getTemperature());
   addCommand.setWindspeed(addHistoricalDataView.getWindspeed());
   //execute
   addCommand.execute();
   //set return value to View Bean
   addHistoricalDataView.setReading_id(addCommand.getReading_id());
   /* end of customize */
   
   3. You receive the AddHistoricalDataServlet.java and class file. Select AddHistoricalDataServlet.java and the class in the left frame of WebSphere Studio. In the menu bar, select Projects->VisualAge for Java->Update from VisualAge as shown in Figure 57 on page 74.
Developing a view

JavaBean Wizard created three JSP sources (input, result, and error). You can change this output using WebSphere Page Designer. In this sample application, we change the input JSP file. Follow these steps:

1. In the left frame of WebSphere Studio, select `AddHistoricalDataServletInput.jsp`. From the menu bar, select `Tools->Edit with->PageDesigner`. You can see the Page Designer window as shown in Figure 58.

![Figure 57. Update from VisualAge for Java](image)
2. Select the form in the right frame of the Page Designer. Select Edit->Attribute... from the menu bar. You see the Attribute dialog as shown in Figure 59 on page 76. Change Action to /pdk400/stage3/addhstData.

3. Click OK to finish.
3.4.2.4 Publishing and setting up WebSphere Application Server

After you change the sources that were created by the JavaBean Wizard, you should publish the files to the AS/400e server. This section explains how to publish the files and set up the WebSphere Application Server.

WebSphere Studio does not support a WebSphere application alias. You should change the servlet configuration file by yourself. You also have to register the servlet to your WebSphere Application Server. Follow this process:

1. The servlet configuration file (AddHistoricalDataServlets.servlet) that was created by the JavaBean Wizard is shown here:

```xml
<?xml version="1.0"?>
<!-- This file was generated by IBM WebSphere Studio using
C:\WebSphere\Studio\BIN\GenerationStyleSheets\AppServerV3\JSP1.0\WebSphere\ServletConfig.xsl-->
<servlet>
  <page-list>
    <default-page>
      <uri>/stage3/AddHistoricalDataServletResults.jsp</uri>
    </default-page>
    <error-page>
      <uri>/stage3/AddHistoricalDataServletError.jsp</uri>
    </error-page>
  </page-list>
</servlet>
```
2. This control file is configured to work on the default server root path. In this sample application, we located the servlet under the PDK400 directory. We also registered the AddHistoricalDataServlet as addhstdata. We changed the file name as addhstdata.servlet and path name as /pdk400 as shown in the code example in step 2 on page 59.

```xml
<?xml version="1.0"?
<page-list>
    <default-page>
        <uri>/pdk400/stage3/AddHistoricalDataServletResults.jsp</uri>
    </default-page>
    <error-page>
        <uri>/pdk400/stage3/AddHistoricalDataServletError.jsp</uri>
    </error-page>
</page-list>
</servlet>
```

3. Before you publish the project, move the addhstdata.servlet file to pdk400/servlet as shown in Figure 60. The JavaBean Wizard creates the original file of addhstdat (that name is RetrieveHistoricalDataServlet.servlet) under pdk400/servlet/com/ibm/rochester/sg245999/stage3/servlet.

![Figure 60. Renaming and relocating the PageList configuration file](image)

4. Right-click the pdk400 projects. Select Publish whole Project as shown in Figure 61 on page 78.
Figure 61. Publish whole Project
Chapter 4. Setting up the AS/400e server, running the application

The AS/400e server offers an incredible advantage with its integration features. All the services you need to set up your environment to run the sample e-business application are integrated directly on the operating system. This chapter explains how to setup the AS/400e server and run the sample e-business application provided in this redbook. By following this process, you will gain the knowledge you need to configure your own environment.

This chapter provides a path with tips to help you setup the sample e-business application. It is not a quick-start guide. Prior to reading this chapter, you should have some experience in OS/400 administration and configuration. You should also have TCP/IP running and all the appropriate PTF and licensed programs correctly installed.

The following sections explain:

- The system environment that was used to write this redbook
- A technology overview about the products involved
- The system setup tasks
- Everything you need to run the sample e-business application

4.1 System environment

This section explains the AS/400e server and PC workstation environment, which we used to write this redbook, to run the sample e-business application. The hardware that is described here is not a pre-requisite or the ideal machine for delivery Web applications.

For more information about AS/400e server and WebSphere Application Server prerequisites, see the Web site at:
http://www.as400.ibm.com/products/webSphere/docs/as400v302/docs/instprrq.html

4.1.1 AS/400e server

The AS/400e server that was used when researching and testing the sample e-business application was a Model 170 (feature 2385) with 1.5 GB RAM and 150 GB disk (10 disk units). The performance index of this server is about 460 CPW in batch mode and 50 CPW in interactive mode. The operational system version was V4R4 and the WebSphere Application Server was Advanced Edition Version 3.02 (5733-WA2/WA3).
The AS/400e server currently supports WebSphere Application Server Standard Edition and Advanced Edition. The Standard Edition is a scaled-down version of the Advanced Edition. They both have the same look and feel in regard to installation, configuration, and runtime. However, the Advanced Edition also offers these features:

- Enterprise JavaBean support
- Workload management support for increased scalability support

In the sample application using EJB support, our environment used:

- OS/400 V4R4 (Cumulative PTF package C0252440)
- WebSphere Application Server Advanced Edition for AS/400 (Group PTF SF99029-01)

For a detailed PTF related to WebSphere Application Server Advanced Edition for AS/400 V4R4, go to:

4.1.2 PC workstation

The PC workstation in our sample environment consisted of a Netfinity 3000 with 256 MB RAM, 8 GB SCSI disk unit, and Microsoft Windows NT 4 Workstation (Service Pack 4).

In your setup, you must have installed Microsoft Service Pack 4, JDK 1.1.7, WebSphere Studio 3.0X, and an Internet browser, such as Microsoft Internet Explorer 5 or Netscape Communicator 4.7, with Java capabilities. You also need to configure the WebSphere Application Server Administrator Console (WAS Administrator Console) during the setup process. We explain how to do this in the following sections. Before you start the setup process, be sure to gather all the necessary media.

4.2 Technology overview

Prior to learning how to setup the system environment, there are important concepts that you need to understand that were used as a basis for writing this book. Understanding these concepts will help you to create a new Web application using state-of-the-art tools.

4.2.1 Application servers

*An application server* is a set of routines or software that links new or existing legacy applications and data to Web applications. It doesn’t require the use of
screen-scrapers or a 5250 HTML Gateway to provide a Web interface to your application. It gives you a new way to develop computer programs, using small sets of code that run on the application server and can be reused by another applications. IBM offers WebSphere Application Server to perform the job of an application server.

4.2.1.1 WebSphere Application Server

As the popularity of Java continues to increase, more people are expected to move to WebSphere Application Server development. It's a Java-based servlet engine that's built on top of the native Java virtual machine on the AS/400e server. It provides a Java servlet API that is defined by Sun. If you write to this Java servlet API standard, your application is portable across any operating system and any environment that supports servlets. This is a big reason why writing with the Java servlet API is becoming a popular interface in which to write. IBM WebSphere Application Server (WAS) ties this all together.

Most applications access data, so there are a lot of capabilities in WebSphere Application Server to help you access the DB2/400. The Java Database Connector (JDBC) feature allows you to access the database in a native way. There is a package in this product with data access beans that make accessing the database even easier for Java programmers.

WebSphere Application Server also comes with JavaServer Pages (JSP) support, which is a scripting language for servlets. As Net.Data was the scripting language for CGI, JSP are the scripting language for servlets. With JSP, the programming logic is typically Java source code, but you don't have to compile it. You simply place it on the HTTP server. Then, a browser accesses the server and data, which is then compiled dynamically for you.

Figure 62 on page 82 shows the details of the WebSphere Application Server environment.
There are two WebSphere Application Server editions for the AS/400e server. The Standard Edition comes with 56-bit encryption support (5733-AS2) for international usage or a 128-bit encryption support (5733-AS3) for North America and restricted usage outside North America. The Advanced Edition has only 56-bit encryption support (5733-WA2). In both versions, you only need the IBM HTTP Server for AS/400 (5769-DG1) to run. All the components used to implement WebSphere Application Server 3.0 depend on JDK 1.1.7, and the administration client does not run on the AS/400e server. A Java application runs on the PC workstation.

The Standard Edition is a no-charge product. On the other hand, the Advanced Edition is a cost-based product, with the advantage that you benefit from support for Enterprise JavaBeans (EJB), more scalability, and workload management. The Advanced Edition was used in writing this redbook due to the use of EJB in the sample e-business application. Apart from EJB support, all the features of the Advanced Edition were designed to work on the Standard Version too. For more information, see: http://www.as400.ibm.com/products/websphere

Note
4.2.1.2 Execution environment
The WebSphere Application Server Advanced Edition 3.0 environment runs independently of the HTTP server. On a single AS/400e server, this independency enables any number of HTTP server instances to use the same resources of a single WebSphere Application Server as shown in Figure 63.

In Figure 63, notice how WebSphere Application Server Advanced Edition 3.0 changes the sequence of communication between a browser and a servlet. There appears to be an additional step in the process, which is the Web server interface to the application server. This interface allows multiple, different Web server instances to run and access the same application server.

WebSphere Application Server Advanced Edition 3.0 implements Version 2.1 of the Sun Java servlet class libraries. The JavaServer Web Development Kit (JSWDK) 1.0 is the reference implementation for JSP 1.0 technology and the Java servlet API 2.1. The JSWDK is available at no charge from the Sun Microsystems Web site. For a complete definition of the Sun Java servlet API specification, refer to the Sun on-line documentation at: http://java.sun.com/products/servlet/2.1/

4.2.1.3 Multiple administrative servers support
The WebSphere Application Server for the AS/400e server comes with an additional feature. It supports more than one administrative server. That
means it supports different environments running different configuration files. For example, in the Microsoft Windows NT and AIX operating systems, you can have only one WebSphere Application Server environment. On OS/400, you can configure the number of administrative servers you need for testing or production systems. This could be a powerful feature for developers and system engineers, because the WebSphere Application Server jobs run independently of other jobs and independently of HTTP server jobs. In this situation, you can create your own administrative server, with your own WAS Administrator Console and your own security information. Figure 64 shows how the AS/400e server works with this configuration.

As shown in Figure 64, each administrative server indicates a new, completely different, and independent instance to run the applications. You can manage, start, stop, delete, and create different configurations, during production time, without risk to other administrative servers.
4.2.2 Directory services

A directory is a listing of information about objects arranged in some order that gives details about each object. Common examples are a city telephone directory and a library card catalog. In computer terms, a directory is a specialized database, also called a data repository, that stores typed and ordered information about objects. Directories allow users or applications to find resources that have the characteristics needed for a particular task.

For example, a directory of users can be used to look up a person's e-mail address or fax number. A directory can be searched to find a nearby PostScript color printer. Or a directory of application servers can be searched to find a server that can access customer billing information. Searching a directory is similar to looking up a name in the white or yellow pages of a telephone directory. If the name of a particular individual object is not known, the directory can be searched for a list of objects that meet a certain requirement. However, directories stored on a computer are much more flexible than the yellow pages of a telephone directory. They can usually be searched by specific criteria, not just by a predefined set of categories.

Although directories may have originally been used for databases of personal information, such as a telephone number or e-mail address, the number of directory applications has recently increased considerably. Directories are now being used to hold all of the information about a person and are being used to authenticate a user to network services. Systems management applications have recently started exploiting the directory for profile-based management of system resources, such as bandwidth utilization of a network.

4.2.2.1 LDAP concepts

In 1988, the Consultative Committee on International Telephony and Telegraphy (CCITT) created the X.500 standard. This standard became the ISO 9594, Data Communications Network Directory. In 1990, a recommendation was made for it to change from X.500 to X.521. However, it is still commonly referred to as X.500.

The X.500 standard organizes directory entries in a hierarchical name space capable of supporting large amounts of information and specifies that communication between the directory client and the directory server uses the directory access protocol (DAP). However, as an application layer protocol, the DAP requires the entire Open System Interconnection (OSI) protocol stack to operate. Supporting the OSI protocol stack requires more resources than are available in many small environments.
Therefore, an interface to an X.500 directory server using a less resource-intensive or lightweight protocol was desired. LDAP was developed at the University of Michigan (USA) as a lightweight alternative to DAP (therefore, the name LDAP). LDAP requires the lighter weight and more popular TCP/IP protocol stack rather than the OSI protocol stack. LDAP also simplifies some X.500 operations and omits some esoteric features.

LDAP defines a communication protocol. That is, it defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. However, when referring to a directory that can be accessed using LDAP, the directory is usually called an LDAP directory. Therefore, LDAP directories can be implemented in many different ways. IBM implements cross-platform LDAP directories using DB2 and Lotus Domino.

An LDAP client is a software application that accesses an LDAP server using a TCP/IP connection. The client accesses the LDAP directory using an industry standard API, such as the LDAP C API, and does not need to know how the LDAP server stores the information. A client may log on anonymously to an LDAP server. In this case, they would see only “public” information. Or, they would authenticate as a specific user, in which case, information to which that particular user is allowed access is also made visible.

LDAP, which has become the Internet standard for directory operations, is starting to replace the more familiar HTTP in some Internet applications. An Internet URL may specify the address of an LDAP server, instead of an HTTP server. When specifying an LDAP URL, parameters can be specified to perform searches of the directory. For example, using a Web browser, you can search a directory to display personal information by specifying:

Ldap://ldap_server/c=rochester,o=ibm??sub?(cn=John Smith)

LDAP directories may be located on a single server or configured across multiple servers. LDAP servers can replicate information between servers, making information more accessible. Synchronization of LDAP directories with non-LDAP directories provides for a “meta-directory” and a consistent method for all directory access. You can deploy applications that provide the synchronization, and therefore, extend your LDAP directory to meet your specific needs.

4.2.2.2 AS/400 Directory Services

AS/400 Directory Services provides Lightweight Directory Access Protocol (LDAP) on the AS/400e server. AS/400 Directory Services is part of the IBM SecureWay Directory family of products and services and is sometimes
referred to as SecureWay Directory for OS/400. LDAP runs over Transmission Control Protocol/Internet Protocol (TCP/IP) and is gaining popularity as a directory service for both Internet and non-Internet applications. You perform most setup and administering tasks of the LDAP directory server on the AS/400e server through the graphical user interface (GUI) of AS/400 Operations Navigator. To administer AS/400 Directory Services, you must have Operations Navigator installed on a PC that is connected to your AS/400e server. You can use AS/400 Directory Services with LDAP-enabled applications, such as mail applications that look up e-mail addresses from LDAP servers. Since V4R3, LDAP has been included free in OS/400 (option 32) as part of OS/400 Directory Services. It includes an LDAP server and a complete set of LDAP clients and utilities.

Figure 65 shows a typical scenario of LDAP usage on AS/400 Directory Services.

Figure 65. AS/400 Directory Services typical scenario implementing LDAP

The OS/400 implements an LDAP client that supports the ability to access any LDAP server from all OS/400 ILE programming languages: C, COBOL, and RPG. An LDAP client for Microsoft Windows is included with OS/400 Client Access, and a Java client is included in the OS/400 support of Java Naming and Directory (JNDI).

Command line utilities are provided to access an LDAP server from Microsoft Windows and OS/400. These utilities are compatible with LDAP utilities.
provided for other operating systems and allow you to search, add, modify, and delete directory information.

LDAP is the standard for Internet directories in a heterogeneous network. Regardless of the directory’s implementation, LDAP provides applications with a consistent view and access method for information in the network. Initially LDAP directories focused on personal information, such as names and addresses. However, LDAP directories are becoming the foundation for network operating systems and systems management. As explained here, you can use OS/400 support of LDAP in your network to simplify access to information.

4.3 System setup tasks

This section explains how to perform the following processes:
- Set up a new administrative server for the WebSphere Application Server environment
- Configure an HTTP server to support a new instance
- Enable AS/400 Directory Service to provide LDAP
- Install the WAS Administrator’s Console on your PC workstation
- Test whether the configuration works

4.3.1 Setting up a configuration file for WebSphere Application Server

In our sample, we used some of values shown in Table 4 to configure and run WebSphere Application Server, WAS Administrator Console, and the sample e-business application. You do not need to use the same values that are shown in the table.

Note that if you change the values, be sure you make the same modifications for any other places as indicated in this redbook. The port numbers were already generated. Therefore, if you are already using these numbers, you can change them at your convenience. You can change all the values shown in Table 4 without affecting the behavior of our WebSphere Application Server configuration and HTTP server instance.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General project name</td>
<td>PDK</td>
<td></td>
</tr>
<tr>
<td>AS/400e server user ID</td>
<td>PDKUSER</td>
<td>The user should have *SECOFR user class</td>
</tr>
<tr>
<td>AS/400e server password</td>
<td>PDKPASS</td>
<td></td>
</tr>
</tbody>
</table>
### Variable | Value | Description
--- | --- | ---
AS/400e server name | AS26 | Replace with your system name
Project library name or admin.dbschema | PDKLIB | Collection library where all the system journals, database tables, and internal SQL control tables will be stored. This library provides all necessary SQL interfaces. It is modified in the admin.properties file.
WebSphere Application Server configuration root directory or admin.instance.root | \QIBM\UserData\WebASAdv\pdkdata | This is the folder on the integrated file system (IFS) that contains all the data about the specific instance we will create to run our sample e-business application. It is modified in the admin.properties file. This setting must be the same as the one we use when running the `crtnewinst` command on the Qshell interface.
WebSphere Application Server admin port number or admin.bootstrapPort | 7760 | This is the port number used to open the WAS Administrator Console Java application. It is modified in the admin.properties file.
WebSphere Application Server OSE port number or ose.srvgrp.ibmappserve.clone1.port | 8860 | That's the port number used internally by WebSphere Application Server to handle servlet requests. It is modified in the bootstrap.properties file.
WebSphere Application Server location server port number or admin.lsdPort | 9960 | The TCP/IP port for the Location Service Daemon, which is used internally by WebSphere Application Server. It is modified in the admin.properties file.
HTTP server port number | 6660 | Port number used to access the application server through the HTTP server. It is modified in the HTTP setup process.
Setting up the new administrative server requires this process:

1. Copy the default WebSphere Application Server configuration files into a new directory.
2. Modify these files to match the information listed in the following process for your unique instance.
3. Start the new administrative server of WebSphere Application Server.
4. Create a new HTTP Server configuration with the correct properties.
5. Create and start a new HTTP server instance based on this configuration.

---

**Tip**

Read the information contained in the configuration files under the properties folder. After the configuration files are copied, they should be located in the /QIBM/UserData/WebASAdv/pdkdata/properties directory.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP instance name</td>
<td>PDKINST</td>
<td>The name given to the HTTP server instance that handles all the requests for our specific HTTP configuration. It is modified in the HTTP setup process.</td>
</tr>
<tr>
<td>HTTP config name</td>
<td>PDKCONF</td>
<td>The configuration file name used to store all the specific data about the new instance and configuration of HTTP server. This file can be accessed by entering the WRKHTTPCFG &lt;filename&gt; command on the 5250 terminal. It is modified on the HTTP setup process.</td>
</tr>
<tr>
<td>WebSphere Application Server job name</td>
<td>PDKJOB</td>
<td>The name given to the submitted job that handles all the transactions and communication under the WebSphere Application Server subsystem. It is used when starting one administrative server.</td>
</tr>
</tbody>
</table>
Let's begin by following these steps:

1. Copying the configuration files is fast and easy with the Java program that comes with WebSphere Application Server. Enter Qshell Interpreter. Then, run the script that creates all new server directories, and set up the correct authorities. On a 5250 terminal, type:

   STRQSH

   This command starts the Qshell interface. Then, type:

   /QIBM/ProdData/WebASAdv/bin/crtnewinst /QIBM/UserData/WebAsAdv/pdkdata

   This command creates a folder on the OS/400 Integrated File System (\QIBM\UserData\WebAdAdv\pdkdata) that contains all the data you need to start a new instance on WAS (Figure 66).

![Figure 66. WebSphere Application Server configuration files on the IFS](image)

2. Edit the properties files. These files are located in the directory you specified as a parameter to the Qshell `crtnewinst` command. Edit the `bootstrap.properties` file in the `/QIBM/UserData/WebAsAdv/pdkdata/properties` directory in the AS/400 IFS.
3. Use your network drive to access the AS/400 IFS. This is necessary to edit the configuration files with an editor like Microsoft Wordpad.

4. Replace the default values of the properties listed in Table 5 with the corresponding new values.

**Table 5. Bootstrap.properties files**

<table>
<thead>
<tr>
<th>Property to edit</th>
<th>New value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.root</td>
<td>/QIBM/UserData/WebASAdv/pdkdata</td>
<td>Where the WebSphere Application Server looks for configuration data when it starts a job.</td>
</tr>
<tr>
<td>ose.tmp.dir</td>
<td>/QIBM/UserData/WebASAdv/pdkdata/temp</td>
<td>This directory stores temporary files for the internal work of the WebSphere Application Server.</td>
</tr>
<tr>
<td>ose.logs.dir</td>
<td>/QIBM/UserData/WebASAdv/pdkdata/logs</td>
<td>This directory stores execution log files of the WebSphere Application Server.</td>
</tr>
<tr>
<td>ose.srvgrp.ibmappserve.clone1.port</td>
<td>8860</td>
<td>The port used to establish communications between the application server and Web server.</td>
</tr>
</tbody>
</table>

5. Edit the admin.properties file in the pdkdata/properties directory. Then, change the values according to those shown in Table 6.

**Table 6. Admin.properties file**

<table>
<thead>
<tr>
<th>Property to edit</th>
<th>New value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mntr.admin.name</td>
<td>PDKMNTR</td>
<td>This is the name of the admin server job running under the WebSphere Application Server subsystem. This job is responsible for handling the WAS Administrator’s Console tasks.</td>
</tr>
</tbody>
</table>
### Property to edit | New value | Description
--- | --- | ---
install.initial.config | True | This setting controls whether the sample configuration is created when the administrative server is started, for example the `snoop` and `hello` servlet. This value is changed to `false` automatically when the administrative server is started the next time.

admin.dbSchema | PDKLIB | This setting must be a unique database collection name. A new database collection is created if it does not already exist.

admin.bootstrapPort | 7760 | Enter an unused port number. The WAS Administrator Console uses this port number when connecting to an instance of the application server. The default value, 900, which is used by the administrative server, should not be used.

admin.lsdPort | 9960 | Enter an unused port number. The default value, 9000, which is used by the administrative server, should not be used.

admin.classpath | Replace the reference to “default” with `pdkdata` | These parameters indicate where the application server searches information when running the administrative job. In this sample, you only need to change one line that refers to the default path.

admin.instance.root | `/QIBM/UserData/WebASAdv/pdkdata` | This setting is the fully-qualified path name of `pdkdata` directory, where all the specific configuration data is stored.

java.properties | Replace the reference to “default” with `pdkdata` | This is internal information needed to run the administrative server.
6. Edit the sas.server.props file in the  
/QIBM/UserData/WebASAdv/pdkdata/properties directory. Modify the  
values as shown in Table 7.

Table 7. The sas.server file values

<table>
<thead>
<tr>
<th>Property to edit</th>
<th>New value</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.CORBA.keytabFileName</td>
<td>Replace the reference to “default” with pdkdata.</td>
</tr>
<tr>
<td>com.ibm.CORBA.bootstrapRepositoryLocation</td>
<td>Replace the reference to “default” with pdkdata.</td>
</tr>
</tbody>
</table>

For more information about the port numbers that are currently in use  
on your AS/400e server, enter the following command on a 5250  
command line:

```
NETSTAT *CNN
```

Press F14 to view the assigned port numbers.

7. Start the WebSphere Application Server subsystem. This will increase the  
application server and the WAS Administrator Console instance. On a  
5250 terminal, type the following command:

```
STRSBS SBSD(QEJB/QEJBSSBS)
```

After a while, depending on your system performance, you should see a  
screen like the example in Figure 67. It indicates the jobs that are started  
with a status of JVAW and EVTW.
Chapter 4. Setting up the AS/400e server, running the application

8. Verify that everything is okay. Start the new EJB admin server by typing the following command on your 5250 screen. Be sure you're not in the Qshell Interpreter.

```
SBMJOB CMD(CALL PGM(QEJB/QEJBMONTR) PARM('-p' '/QIBM/UserData/WebASAdv/pdkdata/properties/admin.properties')) JOB(PDKJOB) JOBD(QEJB/QEJBJOBD) JOBQ(QEJB/QEJBJOBQ) USER(QEJB)
```

9. Type the following command:

```
WRKACTJOB
```

Then, you should see your administrative server jobs start in the QEJBBSBS subsystem. There will be two jobs. It will take some time for your server jobs to become ready because this is the first time you are starting your server and a default configuration must be created. When it

---

**Tip**

If you need to restart the QEJBBSBS subsystem for any reason, do this after the subsystem ended. If you don’t wait until the subsystem ends, some jobs may remain locked on the QSYSWRK subsystem, and then when you start again, it will not work properly.
is done, you should see a screen like the example shown in Figure 68. Now you can configure the HTTP server instance.

![Figure 68. PDKJOB started](image)

### 4.3.2 Configuring the HTTP server

On the AS/400e server, it is easy to configure the HTTP server to run the administrative instance and create a new configuration for the sample e-business application. This section takes you through the steps to check whether the HTTP configuration is okay.

#### 4.3.2.1 Starting the HTTP administrative server

To start the HTTP administrative server, follow these steps:

1. Sign on as a user with the necessary security access. Then, enter the command:
   
   ```
   GO TCPADM
   ```

2. Choose option 2 (Configure TCP/IP Applications), then option 14 (Configure HTTP), and then option 1 (Change HTTP Attributes). You will see a screen like the example shown in Figure 69.
Chapter 4. Setting up the AS/400e server, running the application

3. Start the admin HTTP server job to enable the configuration through a Web browser. Type the following command on the 5250 terminal:

   STRTCPVR SERVER(*HTTP) HTTPSVR(*ADMIN)

   This command starts the ADMIN instance and allows you to create the sample e-business application instance.

   \-----
   **Tip 1**
   \-----

   If the AUTOSTART parameter is set to *YES, every time you start TCP on the AS/400e server, all the HTTP server instances will start too.

   \-----
   **Tip 2**
   \-----

   For the Number of server threads attribute shown in Figure 69, usually the standard value is good. However, after working for some time with the HTTP server, you can discover the actual threads that your system is using. If the number of threads used by the job is higher than the number defined on the HTTP attributes, you should change to the same number of threads your job is using.

   Figure 69. Checking the HTTP attributes
4. Type the `WRKACTJOB` command to check whether the job successfully started. You should see a screen like the one shown in Figure 70.

![Work with Active Jobs](image)

**Figure 70. HTTP started jobs**

5. Note that five jobs were started on the QHTTPSVR subsystem. Check whether the status is exactly the same as the screen in Figure 70. If it's okay, the HTTP server and administrative instance are already running, and you can continue.

4.3.2.2 Configuring a new HTTP server instance

You need to access the HTTP ADMIN server on the AS/400e server. This is a special server instance that is capable of allowing administrators to modify the server configuration and start, stop, and restart instances of the HTTP server remotely over the Web.

**Tip**

At this point, you must check whether your Internet browser has a 0 KB memory cache setting. This setting helps to avoid strange data on your browser by always refreshing the information.

1. The ADMIN server usually runs on port 2001. Open a Web browser, and enter the URL:
http://MyAs400:2001

Here MyAs400 is the hostname of the AS/400e server that is described in Table 4 on page 88. Enter your logon and password information when prompted. Then, a page appears like the example in Figure 71.

![Figure 71. HTTP Administration server](image)

2. Click **IBM HTTP Server for AS/400**. Then, click **Configuration and Administration**. Next, create a configuration for your new server instance. Click **Configurations** and then **Create configuration**. The right panel prompts you for a configuration name, which can be anything. For the purpose of this redbook, use the same name (**PDKCONF**) as the one listed in Table 4 on page 88. Leave the **Create Empty Config** box selected. Click **Apply**. Remember to click Apply whenever you make changes in this environment. Otherwise, the changes will not be saved.

3. The server configuration file is now created. Select the name of your configuration in the drop-down box at the left, and click the **Basic** link.
4. In the **Default port** field, enter the port number given to you for the HTTP server instance, as shown in Table 4 on page 88.

5. In the **Host name**, enter the name of the AS/400e server. Then, you should see a page like the one in Figure 72. Be sure to click **Apply** so the changes take effect.

![Configuration and Administration](image)

**Figure 72. Configuring the HTTP server**

6. Click the **Java servlets** link. Choose **Websphere Version 3**. Select the **Servlets and JavaServer Pages (JSP)** check box. Choose **pdkdata** for the WebSphere domain. Then, click **Apply** to save the changes as shown in Figure 73.
7. Enable the HTTP methods that are required. This allows the HTTP server to process CGI (and then pass it to WebSphere Application Server). Click Request Processing and then Methods. Select the GET and POST check boxes. You may leave the other methods “as is”. Once again, click Apply.

8. Add the WebSphere Application Server request routing entries. Click the Request routing link. Select an Index number (choose the last option), and select Insert after to choose the placement of an entry.

9. Fill in the fields as shown here, and then click Apply.

   - Action: Pass
   - URL template: */*
   - Replacement file path:
     /QIBM/UserData/WebASAdv/pdkdata/hosts/default_host/pdk400/web/*

You should now see a page like the example shown in Figure 74 on page 102. Check your new entries to make sure that they are correct.
10. Create an HTTP server instance to use this configuration. In your browser window, click **Server instances** and then **Create server instance**. Name your instance *pdkinst*. Select your configuration from the drop-down box, and click **Create**.

11. Start the HTTP server by clicking **Work with server instances**. Select your server instances, and click **Start**. On the 5250 screen, type:

```
WRKACTJOB
```

You should see the PDKINST jobs running as shown in Figure 75.
Now, you should configure the AS/400 Directory Services to enable the LDAP interface.

### 4.3.3 Configuring LDAP with Operations Navigator

In order for you to use AS/400 Directory Services, Operations Navigator must be configured on your PC workstation and started on the AS/400e server. However, our sample e-business application uses a specific library to hold the LDAP database (PDKLDAP) and the entry `ibm` for organization and `uk` for context.

1. Open Operations Navigator as shown in Figure 76 on page 104. Right-click **Directory** in the Server Name panel. If you have not configured the LDAP server, you should see a “Configure” entry in the menu. Otherwise, you should see a “Reconfigure” entry. Choose one of them and you will see a display like the example in Figure 77 on page 104.
2. In the Library name field, type:

/QSYS.LIB/PDKLDAP.LIB

Figure 76. Operations Navigator LDAP server configuration

Figure 77. Specifying a database library
Click the **Next** button. Choose Yes to confirm the creation of the new library, as shown in Figure 78.

![Figure 78. Confirming the new library creation](image)

3. Specify in which auxiliary storage pool (ASP) you want the library to be created (Figure 79). Choose the correct ASP, and click **Next** to continue.

![Figure 79. Specifying an ASP to create the new library](image)

4. Choose an administrator name for the LDAP database. Then click **Next** as shown in Figure 80 on page 106. Note that the name you use here has no relation to the AS/400e server user profiles. This administrator name will only be used to manage the LDAP database.
5. Add directory suffixes to run the sample e-business application. Complete the fields as shown in Figure 81. When you are finished, click **Add** and then click **Next**.

6. On the next display, select **Yes, start this server when TCP/IP is started**, and click **Next** (see Figure 82).
7. You should see a display like the example in Figure 83. Click **Finish**. Now your LDAP service is configured.

At this point, you can start configuring the interface to access and manage WebSphere Application Server.
4.3.4 Installing WAS Administrator Console

WebSphere Application Server provides centralized administration of application servers, servlets, and other resources. An administrative server tracks a domain's contents and activities by maintaining an administrative repository. The repository is the database of information about an administrative domain and can be shared by several administrative servers on multiple nodes in the domain.

Administration occurs through method calls to resource beans in the administrative server. A graphical administrative client (WAS Administrator Console) makes requests to an administrative server to access or modify a resource in the domain. An administrative server also communicates with other remote administrative servers to delegate tasks and to respond to requests.

To install the WAS Administrator Console, follow these steps:

1. Insert the WebSphere Application Server media in the CD-ROM drive, and start the setup program.
2. Choose your language, and select the Administrator Console check box. Click Next.
3. Click the JDK 1.1.7 option, and click Next again. If you don't see a window like the example shown in Figure 84, your PC may not be configured properly.
Chapter 4. Setting up the AS/400 server, running the application

4.3.5 Testing your configuration

Now the WebSphere Application Server instance is configured and started on the server, and the WAS Administrator’s Console is installed at the PC workstation. In this section, you learn how to test it. In this case, we are going to ping the “hello” EJB and run the “snoop” servlet. Complete these steps:

1. Make sure your WebSphere Application Server admin jobs are running and in the correct states. To verify that they are in EVTW and JVAW status, as shown in Figure 85 on page 110, type the following command:

\[ \text{WRKACTJOB} \]
2. Start WAS Administrator Console. To do this, bring up a DOS prompt on your local machine, and type the following command:

```
C:\>adminclient AS26 7760
```

Be sure to type the host name *exactly* the same way as your AS/400e server, whether it is uppercase, lowercase, or mixed case. Use the admin server port defined in Table 4 on page 88. The application will take some time to start, so be patient.

3. Wait until the message **Console Ready** appears in the bottom pane of the admin client window.

4. Configure the WebSphere Application Server virtual hosts to recognize your HTTP port number. To do this, click the **Topology** tab. Then, select **default_host**. In the right pane, click the **Advanced** tab, and a window appears like the example in Figure 86. For the HTTP port number, enter **:6660**, as defined in Table 4 on page 88, to every hostname listed in the Host Aliases table. You may need to scroll down. Click **Apply** to save your changes.

---

**Figure 85. PDK jobs running on the QEJBBSBS subsystem**

```
<table>
<thead>
<tr>
<th>Opt</th>
<th>Subsystem/Job</th>
<th>User</th>
<th>Type</th>
<th>CPU %</th>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QEJBBSBS</td>
<td>QSYS</td>
<td>SBS</td>
<td>.0</td>
<td>DEQW</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DEFAULT_SE</td>
<td>QLB</td>
<td>BCI</td>
<td>.0</td>
<td>JVAW</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PDKJOB</td>
<td>QLB</td>
<td>BCI</td>
<td>.0</td>
<td>PGM-QEJBMTNR</td>
<td>EVTW  -</td>
</tr>
<tr>
<td>4</td>
<td>PDKMTNR</td>
<td>QLB</td>
<td>BCI</td>
<td>.0</td>
<td>JVAW</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>QEBJADMIN</td>
<td>QLB</td>
<td>BCI</td>
<td>.0</td>
<td>JVAW</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>QEBJMTNR</td>
<td>QLB</td>
<td>ASJ</td>
<td>.0</td>
<td>PGM-QEBJMTNR</td>
<td>EVTW</td>
</tr>
</tbody>
</table>
```

Parameters or command
```plaintext
F3=Exit  F5=Refresh  F7=Find  F10=Restart statistics
F11=Display elapsed data  F12=Cancel  F23=More options  F24=More keys
```
5. Right-click **Default Server** to start the application server. Click **Start** as shown in Figure 87 on page 112. A pop-up window appears telling you that the server has started. If the default server is already started, stop it and then restart it.
6. Expand **Default Server** and **Default Container**. Right-click the **Hello** bean inside the default container, and select the **Ping** command to run against the bean. A dialog appears like the example in Figure 88.

Figure 87. Starting Default Server

Figure 88. Command ping OK

7. Start a Web browser, and run the snoop servlet. Enter the URL:

   http://MyAs400:6660/servlet/snoop

   Here, **MyAs400** is the hostname of your AS/400e server. If everything is running okay, you will have an environment on the AS/400e server like the example shown in Figure 89.
Once you successfully configure your WebSphere Application Server instance, you can install the sample e-business application source on it.

### 4.4 Restoring the database and sample e-business application

The entire database and sample e-business application source are ready to work. You simply need to restore the data to your AS/400e server. The file named SG245999.zip (see Appendix A, “Using the additional material” on page 167) stores all the libraries and folders needed to run the application. After the process, this file creates two structures on the AS/400e server:

- The HSTDATA library, which contains the database, control tables, and some programs.
- WebSphere Studio archive file (.WAR)
- VisualAge for Java file (.Dat)
- LDAP export file (.ldif)
4.5 The sample e-business application

After you configure your system and restore all necessary data, it's time to look at the sample e-business application. This sample Web site uses the Interplanetary Weather Data application as a vehicle to demonstrate the User-to-Business pattern. The weather data is used for demonstration purposes only and to help you understand the pattern. The sample e-business application was created and may be modified with IBM WebSphere Studio 3.0x or with VisualAge for Java 3.0. For the purposes of this redbook, we use WebSphere Studio as a tool to publish the source code that is already provided for you.

Section 4.5.1, “Application scenario” on page 114, describes the sample application scenario. Then, 4.5.2, “Restoring the project on WebSphere Studio” on page 116, explains how to restore and publish the project on WebSphere Studio. This involves opening an archive file, restoring the original structure of the sample e-business application, and then publishing the data to the AS/400e server.

4.5.1 Application scenario

Figure 90 shows the structural details of the User-to-Business topology 1 and 2 Pattern Development Kit sample Web site. The legend shows how the program modules apply to the stage concepts explained in Chapter 2, “A four-stage approach to Patterns for e-business” on page 9.
The scenario shown in Figure 90 is explained here:

1. A splash screen is initially displayed in the user's browser.

2. A link on the splash screen leads the user to a new browser window with two frames that contain a menu on the left and a contents page on the right.

3. The contents page displays basic information about the sample site. In a production site, this page could be replaced with company information.

4. The frame on the left of the browser window contains the menu page. By clicking a menu item, the user can navigate to other areas of the site. The menu controls what appears in the right frame. For example when the Topology 1 menu item is clicked, a JSP form appears in the right frame.

5. The Topology 1 menu item leads the user to a JSP data entry page. The pages that lead from this menu option are contained in one WebSphere Application Server application called topologyone. This Web application is an implementation of the User-to-Business topology 1 pattern.

6. The Topology 2 menu item leads the user to a logon page. The page that leads from this menu option and all associated pages form the
User-to-Business Patterns for e-business

topologytwo WebSphere Application Server application. The topologytwo Web application is an implementation of the User-to-Business topology 2 pattern.

7. On the logon page, the user is prompted for a user ID and password. Once the user is authenticated, a new menu is displayed in frame 2. This dynamic menu contains a horizontal list with hyperlinks of planets to visit. The menu that is generated is profile based and depends on the type of user that logged on.

8. From the links on the generated menu, the user can gather data from an RPG-compiled program on the AS/400e server.

9. A link on the menu also allows the user to gather data from an EJB interface.

10. Another link on the horizontal menu permits the user to add new records on the database through an RPG command on the AS/400e server.

4.5.2 Restoring the project on WebSphere Studio

The sample e-business application is contained in an archive file called pdk400.war (war is the file extension for archived WebSphere Studio project files). Opening an archive creates a WebSphere Studio project with all the necessary folders and files.

1. To open the archive, from the WebSphere Studio workbench, select File and then Open Archive…. Then, you see a new dialog as shown in Figure 91. Select pdk400.war file, and click the Open button.

![Figure 91. Opening an archive file](image)

2. Click the Extract tab. Choose Create a new project for the entire archive.
3. Click the **Destination** tab. Select **Custom location** for your project. In the Project folder field, WebSphere Studio suggests a standard path for you, but you can change it to suit your needs. Click **Extract** to extract the sample site to your project folder, as shown in Figure 92.

![Figure 92. Extracting an archive file to a new project](image)

WebSphere Studio creates a new project that contains all the sample e-business application folders and files. The extract process could take a long time to finish, but after that, you should see a display like the example shown in Figure 93 on page 118.
The WebSphere Studio's workbench (its main window) has a two-pane graphical user interface (GUI) with many features common to other desktop applications. You can use the keyboard and the mouse to move around the interface, select objects, and perform actions.

The objects you will maintain in WebSphere Studio are the various files that make up your Web sites and applications. WebSphere Studio shows them in three views:

- The File view represents the physical organization of your Web site source files.
- The Relations view represents the hypertext links in your files.
- The Publishing view represents the physical organization of your published Web site files.

Figure 93 shows the file view on the left pane and the publishing view on the right pane.
4.5.3 Changing the connection properties

The sample e-business application uses hard-coded information to connect the application to the database. You need to change this information to make it possible for the application to connect to your AS/400e server.

1. Expand the Servlet folder on the left pane so you can see the CommonVar.java source. It is located under the /servlet/com/ibm/rochester/sg245999/common directory, as shown in Figure 94 on page 120.
2. Double-click the **CommonVar.java** source file to edit it. Modify the **userID**, **password**, and **URL** variables with the correct information for your AS/400e server (Figure 95). Then, save the file.
Chapter 4. Setting up the AS/400e server, running the application

This sample application uses the values shown in Table 8.

Table 8. Values of the sample application

<table>
<thead>
<tr>
<th>Variables</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>PDKUSER</td>
</tr>
<tr>
<td>password</td>
<td>PDKPASS</td>
</tr>
<tr>
<td>URL</td>
<td>jdbc:db2:AS26</td>
</tr>
<tr>
<td>systemName</td>
<td>AS26</td>
</tr>
<tr>
<td>providerURL</td>
<td>iop:as26:7760</td>
</tr>
</tbody>
</table>

3. Now you need to compile the source code to a Java class file. Right-click the CommonVar.java file. Choose Compile from the menu as shown in Figure 96 on page 122. Confirm your action so WebSphere Studio compiles your Java source into a class file. If there are no error messages, everything is okay and there are no problems.
4. Before you publish the entire site, right-click the **CommonVar.class** file, and choose **Check In** from menu. Now this class is ready to be published with the other files of the application.

### 4.5.4 Publishing the project on the server

To actually run the sample and see it working, you have to publish it to an HTTP server and to WebSphere Application Server. WebSphere Studio makes this task simple. You simply need to complete some steps to publish the information in the right place.

To publish the project, you need to define at least one publishing stage with an accessible publishing server. WebSphere Studio must be able to connect to that server and place the files there. Of course, this is the simplest case. The larger the development environment is, the more likely you are to have projects that are published to more than one server. In such cases, you may need additional publishing stages and different server configurations. There
are several other options that affect how the files are published, when they are published, and where they go on the Web server.

For the purpose of this redbook, we used FTP to copy the project files from WebSphere Studio directories to the Web server directories. Here are the tasks to do this:

1. When the archive file was restored, it also restored the AS/400e server IP address. Therefore, you need to change your system. Right-click **http://as20:6662**, and choose **Delete...** from the pop-up menu. Confirm the deletion (Figure 97).

![Figure 97. Deleting the old IP address](image)

2. Right-click **Test** stage in the right pane. Then choose **Insert** from the menu and **Server...** to create a new server as shown in Figure 98 on page 124.
3. A dialog box appears and asks for the new server name. Type:

   MyAS400:PORT

   Here MyAS400 is your AS/400e server name, and PORT is your HTTP server port number as defined in Table 4 on page 88.

4. Include the project files into this new server to make it possible to publish the entire project. Select all the folders on the left pane. Then, drag and drop them into the server folder on the right pane (Figure 100).
Now all the files included in the publishable server are in the right pane. Next, you need to set up additional information on the publishing server.

5. To setup a publishing server, in the publishing view, right-click the server icon. Select Properties from the menu as shown in Figure 101 on page 126.
6. Click the **FTP publish** radio button. Then type the user name and password in the correct field. Your display should look like the example shown in Figure 102.
Chapter 4. Setting up the AS/400e server, running the application

7. Click the **Define Publishing Targets...** button. A dialog box appears that asks you for the publishing targets. The targets are paths for the publishing directories on the Web server and the WebSphere Application Server. By default, WebSphere Studio has two publishing targets:

- Servlets for the servlets publishing directory on WebSphere Application Server
- Html for the document root publishing directory

Each project has a servlet folder. Only the files in the servlet folder go to the directory defined for the servlet publishing target. All other files go to the directory defined for the html publishing target.

For the sample e-business application, enter the following targets:

a. For the **html** entry, type:

   /QIBM/UserData/WebASAdv/pdkdata/hosts/default_host/pdk400/web

b. For the **servlet** entry, type:

   /QIBM/UserData/WebASAdv/pdkdata/hosts/default_host/pdk400/servlets

See Figure 103 on page 128.
Confirm all the information. Now your project is ready to be published on your AS/400e server.

8. Right-click the http://as20:6662 server. Choose Publish this Server from menu as shown in Figure 104.

A dialog box should appear like the example in Figure 105. Confirm all the information, and click the OK button to publish the project. Confirm all the subsequent dialog boxes to keep the publication process running.
WebSphere Studio opens a new Web browser page that shows the results of your publication. The page should look like the example shown in Figure 106.

Figure 105. Confirm publishing options

Figure 106. Publishing results
Now the sample e-business application is published to your AS/400e server. It's time to configure the WebSphere Application Server environment to handle the processing of the application.

4.6 Configuring the sample e-business application environment

Because the sample e-business application implements EJB, which is only supported on the WebSphere Application Server Advanced Edition, this section is dedicated exclusively to Standard Edition users.

This section describes two ways to configure your WebSphere Application Server environment:

- **Manual**: With all steps described, so you can choose the best way to implement the sample e-business application.
- **Automatic**: Can be used after you create your first environment.

The automatic configuration way is an important strategy for managing your WebSphere Application Server environment. For the purpose of this redbook, we show only one example for importing and exporting the configuration files.

4.6.1 Manual configuration

You can perform the application setup tasks manually. Basically you need to configure one JDBC driver, one data source, and the servlets used on the application and EJBs.

Start WAS Administrator Console. To do this, bring up a DOS prompt on your local machine, and type the following command:

```
C:\>adminclient AS26 7760
```

Be sure to type the *host name exactly the same* (uppercase, lowercase, or mixed case) as the name of your AS/400e server. Use the admin server port defined on Table 4 on page 88. The application will take some time to start, so be patient.

4.6.1.1 JDBC driver

To configure the JDBC driver, follow these steps:

1. On WAS Administrator Console, choose the **Topology** pane and right-click the **WebSphere Administrative Domain**. Click **Create...**, and then **JDBC Driver**. See Figure 107.
2. You see a dialog box asking for the JDBC driver properties. Type the values in the fields as shown in Figure 108. Click the Create button to continue.
After that you should see a dialog box (Figure 109), which informs you that your JDBC driver configuration was created on your WebSphere Application Server instance.

![Figure 109. JDBC driver created](image)

3. Install the JDBC driver into the application server. Right-click the **PDK400** JDBC driver and choose **Install...** from the menu, as shown in Figure 110.

![Figure 110. Installing JDBC into the application server](image)

4. You see a dialog box (Figure 111) that asks for which node you want to install your JDBC driver. Choose your AS/400e server name, which is shown in Figure 108 on page 131.
5. Click the **Browse** button to choose a .JAR file.

6. On the dialog box, choose the **db2_classes.jar** file in the `\QIBM\ProdData\Java400\ext` folder. Click the **Open** button. Then, click the **Install** button. You should see a pop-up window like the one in Figure 113.
4.6.1.2 **DataSource**

To configure the DataSource, follow these steps:

1. On the WA Administrative Console, right-click **WebSphere Administrative Domain**. Click **Create...** and then **DataSource** (Figure 114).

![Figure 114. Creating a new DataSource configuration](image)

2. A dialog box appears and asks for the DataSource properties. Type the values in the fields as shown in Figure 115. In the **Database name** field, type your AS/400e server name. Click the **Create** button to continue.
Another dialog box (Figure 116) appears and informs you that your DataSource configuration was created on your WebSphere Application Server instance.

After you complete these steps, you should see a window like the example shown in Figure 117 on page 136. Now, you can configure the application servlets.
4.6.1.3 Web application

To configure a Web application on WebSphere Application Server, follow the steps that are outlined here. You should use a wizard tool to help you perform the setup tasks.

1. Click the Tasks pane.
2. Click Configure a Web application
3. Click the Start Task button (green button at the top of the window).
4. In the Web application name field, type:
   
   pdk400

   You should see a window like the example in Figure 118.
5. Click the **Next** button.

6. On the window (Figure 119 on page 138) that appears, select the node and click the **Next** button.
7. On the next window that appears, confirm the information that is displayed.

8. Type a description for your Web application as shown in Figure 120.

9. Make sure that \pdk400 is entered in the **Web Application Path** field. Click the **Next** button.
10. On the next window, add two new classpaths to make your environment work. The wizard has already created one classpath for you. Type the other two classpaths as shown here:

- /QIBM/ProdData/HTTP/Public/JT400/lib/JT400.jar
- /QIBM/UserData/WebASAdv/pdkdata/hosts/default_host/pdk400/servlets/u2bclient.jar

11. Confirm the information shown in Figure 121 on page 140, and click the **Finished** button.
After all the wizard steps are complete, you should see a dialog box like the example in Figure 122. Your Web application is now created with standard servlets.

12. Restart the Web application. Right-click the pdk400 Web application, and choose Restart Web App from menu, as shown in Figure 123.
Chapter 4. Setting up the AS/400e server, running the application

Figure 123. Restarting the Web application

If everything is okay, you should see a dialog box like the one shown in Figure 124.

Figure 124. Web application created and started

4.6.1.4 Servlets

Now it's time to create the servlets information in the Web application. The sample e-business application uses five servlets, so you need to create a pointer on WebSphere Application Server to the IFS directory structure.

1. Right-click the pdk400 Web application. Choose Create... and then Servlet... as shown in Figure 125 on page 142.
2. A new dialog box appears. Type the servlet information. Then, click the **Add** button to add a new entry on the servlet path list. Click the **OK** button to confirm the new servlet path. Click the **Create** button to finish the servlet creation. See Figure 126.
After the servlet is created, you should see a dialog box like the one in Figure 127.

3. To check the servlet information, right-click the `stage1.RetrieveHistoricalData` servlet, and choose Properties from menu. You should see a window like the one shown in Figure 128 on page 144.
4. Repeat steps 1 through 3 (including Figure 125 on page 142 to Figure 128) to create all five servlets that contain the information described in Table 9, Table 10, Table 11, Table 12, and Table 13. The configuration is case sensitive so be sure to enter the information exactly as it is shown.

Table 9. RetrieveHistoricalData

<table>
<thead>
<tr>
<th>Name</th>
<th>stage1.RetrieveHistoricalData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>for pdk400 sample application</td>
</tr>
<tr>
<td>Servlet Class Name</td>
<td>com.ibm.rochester.sg245999.stage1.servlet.RetrieveHistoricalDataServlet</td>
</tr>
<tr>
<td>Servlet Web Path List</td>
<td>default_host/pdk400/stage1/hstdata</td>
</tr>
</tbody>
</table>
Table 10. SecurityServlet

<table>
<thead>
<tr>
<th>Name</th>
<th>stage2.SecurityServlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>for pdk400 sample application</td>
</tr>
<tr>
<td>Servlet Class Name</td>
<td>com.ibm.rochester.sg245999.stage2.servlet.SecurityServlet</td>
</tr>
<tr>
<td>Servlet Web Path List</td>
<td>default_host/pdk400/stage2/logon</td>
</tr>
</tbody>
</table>

Table 11. MenuOptionServlet

<table>
<thead>
<tr>
<th>Name</th>
<th>stage2.MenuOptionServlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>for pdk400 sample application</td>
</tr>
<tr>
<td>Servlet Class Name</td>
<td>com.ibm.rochester.sg245999.stage2.servlet.MenuOptionServlet</td>
</tr>
<tr>
<td>Servlet Web Path List</td>
<td>default_host/pdk400/stage2/menuOptions</td>
</tr>
</tbody>
</table>

Table 12. hstdataRPG

<table>
<thead>
<tr>
<th>Name</th>
<th>hstdataRPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>for pdk400 sample application</td>
</tr>
<tr>
<td>Servlet Class Name</td>
<td>com.ibm.rochester.sg245999.stage3.servlet.RetrieveHistoricalDataServlet</td>
</tr>
<tr>
<td>Servlet Web Path List</td>
<td>default_host/pdk400/stage3/hstdataRPG</td>
</tr>
</tbody>
</table>

Table 13. addhstData

<table>
<thead>
<tr>
<th>Name</th>
<th>addhstData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>for pdk400 sample application</td>
</tr>
</tbody>
</table>
You should now have a Web application named pdk400 with three standard servlets and five customized servlets, as shown in Figure 129.

![Servlets configuration on the Web application](image)

To enable servlets for running, right-click the pdk400 Web application and choose **Restart Web App** to restart it.

### 4.6.1.5 EJB

To configure the EJB environment on WebSphere Application Server, you need to associate a DataSource and user name to the EJBCoontainer. Follow these steps:

1. Click **Default Container**. On the right pane, you see the properties of this container.

<table>
<thead>
<tr>
<th><strong>Servlet Class Name</strong></th>
<th>com.ibm.rochester.sg245999.stage3.servlet.AddHistoricalDataServlet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Servlet Web Path List</strong></td>
<td>default_host/pdk400/stage3/addhstData</td>
</tr>
</tbody>
</table>
2. Click the **DataSource** tab, and then click the **Change** button. Choose the **HistoricalData** DataSource, and click **OK**.

3. Complete the other fields as shown in Figure 130. Click the **Apply** button to store the modifications.

4. Now a new **DataSource** is created, and you can install the EJBs. Right-click **Default Container**, and choose **Create...** from menu. Click **EnterpriseBean** as shown in Figure 131 on page 148.
5. A new dialog box appears (Figure 132). Click the Browse button to select the .JAR file provided with the sample e-business application. The file is located in the directory \QIBM\UserData\WebASAdv\pdkdata\hosts\default_host\pdk400\servlet.

6. Select the U2B.jar file as shown in Figure 133. Don’t click the Select button yet. Double-click the U2B.jar file.
7. Now you can see all the EJB contained in the .JAR file. Choose the first EJB, and then click the Select button as shown in Figure 134.

8. All the information of EJB was restored from the .JAR file as shown in Figure 135 on page 150. Click the Create button.
A new EJB should be created as indicated by the dialog in Figure 136.

9. Repeat the steps described from step 4 through step 8 (Figure 131 to Figure 136) for each EJB in the .JAR file. For the sample e-business application, you only need three EJBs. When you are finished, you should see a window like the example in Figure 137.
Chapter 4. Setting up the AS/400e server, running the application

10. Start each new EJB by right-clicking it and choosing **Start** from the menu. There is a small red circle on each EJB indicating that it is stopped. After the start command, this circle should turn blue.

At this point, you should have all the configurations needed on WebSphere Application Server to run the sample e-business application.

### 4.6.1.6 LDAP database

To see the implementation of stage two in our sample e-business application, you need to add some user information into the LDAP database. You can do this through an LDIF script file that is provided together with the source code.

1. Open Operations Navigator.
2. Expand your AS/400e server
3. Click **Network->Servers->TCP/IP**.
4. Select **Directory** and right-click. If its status is **Stopped**, you can continue. Otherwise, choose **Stop** from the menu and wait until the status of
directory service is stopped. Right-click Directory again. Then, click Tools and then Import File as shown in Figure 138.

Figure 138. LDAP importing file tool

5. Choose the sectiona.ldif file name. Confirm the import by clicking OK as shown in Figure 139.

Figure 139. Choosing the file name for import data to the LDAP database

After some time, you will see a progress display like the example in Figure 140.
6. Start the directory service. Right-click **Directory**, and choose **Start** from menu. Your LDAP service is now ready to work with our sample e-business application.

### 4.6.2 Automatic configuration

WebSphere Application Server provides an XMLConfig command that runs on the Qshell interface on a 5250 terminal. With this command, you can export and import configurations of your WebSphere Application Server environment. It creates an XML source file with all actual configurations of your environment. This could be helpful to backup and restore configurations faster and to compare manual configurations with the information generated in an XML file. Figure 141 on page 154 shows the command line parameters that are available for the XMLConfig command.
4.6.2.1 Exporting your environment
To export your environment to an XML file, open a 5250 terminal and type:

```
STRQSH
```

This command starts the Qshell interface. Then, type the following command:

```
/QIBM/ProdData/WebASAdv/bin/XMLConfig -export YourWAS.cfg -adminNodeName AS20 -nameServicePort 7762
```

Here, replace “AS20” and “7762” with your own AS/400e server name and admin port number as described in Table 4 on page 88.

After some time, the command creates the YourWAS.cfg file on your root IFS directory. Open this file with an editor and examine your configuration.

For example, the JDBC diver configuration should appear on an XML file as shown in Figure 142.
4.6.2.2 Importing an environment

The importing process runs the same way as the exporting process. You only need to change the -import parameter.

Be careful when you run this command. Otherwise, your entire WebSphere Application Server environment configuration will be replaced.

4.7 Running the sample e-business application on a Web browser

The sample e-business application shows the concepts of the U2B topology 1 and U2B topology 2 business patterns. In the PDK concept, topology 1 is further divided into sub-section Section A. Topology 2 is subdivided into Section B, Section C, Section D, Section E, and Section F. The purpose of this redbook is only to provide a application that shows the implementation of Sections A, B, C, and D. Each section deals with a separate Web interaction:

- Section A is an example of Web access to a read-only DB2 database and a writable DB2 database.
- Section B is an example of custom authentication using an LDAP directory.
- Section C is an example of Web access to a read-only DB2 database. This section generates a dynamic menu based on a user profile stored in the LDAP directory.
- Section D is an example of Web interactivity with EJB and ILE RPG/400 programs on the back end.
Table 14 shows an analogy of the development stages.

Table 14. Mapping between PDK sections and stage number

<table>
<thead>
<tr>
<th>PDK section</th>
<th>Stage number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section A</td>
<td>Stage one</td>
</tr>
<tr>
<td>Section B</td>
<td>Stage two</td>
</tr>
<tr>
<td>Section C</td>
<td>Stage two</td>
</tr>
<tr>
<td>Section D</td>
<td>Stage three</td>
</tr>
</tbody>
</table>

4.7.1 Stage zero

As described in Chapter 2, “A four-stage approach to Patterns for e-business” on page 9, stage zero or pre-stage only implements a Web application with static presentation. To access the sample e-business application in stage zero, open a new window with your Web browser. Then, type:

http://as20:6662/pdk400/pre-stage/index.html

Here, as20:6662 is your AS/400e server name and HTTP port number. You should see a page like the example in Figure 143.
Figure 143. Initial application page

This page is common in all other stages. Click the Continue image (in the lower right-hand corner) to access the main menu as shown in Figure 144 on page 158.
If you explore the menu with your mouse, you will see that the Topology 1 and Topology 2 links on the left frame don’t start any action. This is because they guide you to the JSP pages that make part of the stage one concept.

In stage zero, there aren’t other elements to interact with the application. Let’s move to stage one.

4.7.2 Stage one

To run the sample e-business application that implements the technology described on stage one, open a Web browser window. Enter the following URL:

http://as20:6662/pdk400/stage1/index.html

This brings up the same pages that are shown in Figure 143 on page 157 and Figure 144. On the main menu page, click the Topology 1 image to open the page shown in Figure 145.
On this page, choose one of the planets and the date range you need to collect weather records. Click the **Submit** button. Then, one servlet invokes a communications request to a JDBC method and retrieves the information from the database. This results in the page shown in Figure 146 on page 160.
4.7.3 Stage two

To show an implementation of the concepts described in stage two, open a Web browser window. Then, enter the following URL:

http://as20:6662/pdk400/stage2/index.html

This bring up the same pages that are in Figure 143 on page 157 and Figure 144 on page 158. On the main menu screen, click the Topology 2 image to open the page shown in Figure 147.
When you enter a user name on the logon screen, the application searches the user name on the LDAP database. Depending on which name, it shows a different menu. We provide the LDAP database with the names listed in Table 15.

Table 15. LDAP database

<table>
<thead>
<tr>
<th>User ID</th>
<th>Password</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>jadams</td>
<td>password</td>
<td>All menus</td>
</tr>
<tr>
<td>hsmith</td>
<td>password</td>
<td>Menu 1 and 3</td>
</tr>
<tr>
<td>agriffin</td>
<td>password</td>
<td>Menu 2 and 3</td>
</tr>
</tbody>
</table>

In this example, when you enter the user name “jadams”, the page shown in Figure 148 on page 162 appears.
4.7.4 Stage three

To show an implementation of the concepts for stage three, open a Web browser window. Then, enter the URL:

http://as20:6662/pdk400/stage2/index.html

This results in the same pages that are shown in Figure 143 on page 157 and Figure 144 on page 158. On the main menu page, click the Topology 2 image to open the page shown in Figure 147 on page 161. After logging on, depending on the user, the next page that appears is shown in Figure 148.

Clicking the Get Data using EJB link invokes an EJB command that performs the same tasks as described for stage one, but using EJB instead of servlets. Clicking the Get Data using RPG option invokes an RPG program on the AS/400e server that retrieves the same data as the EJB command and stage one JDBC communication. The start page for both options is shown in Figure 149.
Figure 149. EJB and RPG routines to retrieve historical data

The resulting page has the exact same layout as shown in Figure 150 on page 164.
If you choose Insert Data using RPG from the menu, a data entry window appears. This function invokes an RPG program to store a new record on the database. You can add records to the database and query this data later.
Click **Submit**. Then you can see the window that contains the reading_id that you added as shown in Figure 152 on page 166.
Figure 152. Resulting window from inserting a record
Appendix A. Using the additional material

You can download the Java client programs and the AS/400e programs and libraries that are used in this redbook. The examples were developed using VisualAge for Java Enterprise, Version 3 edition and WebSphere Standard Edition 3.02. OS/400 V4R4 or later is required. These components are available:

- AS/400 RPG code
- AS/400 databases
- AS/400 Java code
- Client Java code

Important

These example programs have not been subjected to any formal testing. They are provided “as is” and are for reference only.

A.1 Locating the additional material on the Internet

The Web material associated with this redbook is available in soft copy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG245999

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the Additional materials and open the directory that corresponds with the redbook form number.

A.2 Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sg245999.zip</td>
<td>Contains all files for the sample e-business application</td>
</tr>
<tr>
<td>sg245999.savf</td>
<td>AS/400 Library Save File for HSTDATA</td>
</tr>
<tr>
<td>readme.txt</td>
<td>Demo setup instructions</td>
</tr>
<tr>
<td>pdk400.dat</td>
<td>The VisualAge for Java repository file</td>
</tr>
</tbody>
</table>

Important note: These example programs have not been subjected to any formal testing. They are provided “as is” and are for reference only.
| pdk400.war | WebSphere Studio project archive file |
| sectiona.ldif | LDAP export file |

To use the files, create a subdirectory (folder) on your workstation and copy the contents of the Web material into this folder.
Appendix B. Special notices

This publication is intended to help developing the e-business application on the AS/400e server. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM WebSphere Application Server, IBM WebSphere Studio, and IBM VisualAge for Java.

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, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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.
Any pointers in this publication to external Web sites are provided for
convenience only and do not in any manner serve as an endorsement of
these Web sites.

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

e (logo)® Redbooks
IBM ® Redbooks Logo
AIX AS/400
AS/400e CICS
CT DB2
Netfinity OS/400
RPG/400 SecureWay
SP SP1
VisualAge WebSphere
Wizard 400
Lotus Lotus Notes
Domino Notes
Tivoli TME
NetView Cross-Site
Tivoli Ready Tivoli Certified

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything.
Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet
Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli
Systems Inc., an IBM company, in the United States, other countries, or both.
In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli
A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other
countries.

Java and all Java-based trademarks and logos are trademarks or registered
trademarks of Sun Microsystems, Inc. in the United States and/or other
countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of
Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United
States and/or other countries and is used by IBM Corporation under license.
ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.
Appendix C. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 175.

- *AS/400 HTTP Server Performance and Capacity Planning*, SG24-5645
- *Design and Implement Servlets, JSPs, and EJBs for IBM WebSphere Application Server*, SG24-5754
- *WebSphere Studio and VisualAge for Java Servlet and JSP Programming*, SG24-5755

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at [ibm.com/redbooks](http://ibm.com/redbooks) for information about all the CD-ROMs offered, updates and formats.

<table>
<thead>
<tr>
<th>CD-ROM Title</th>
<th>Collection Kit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM System/390 Redbooks Collection</td>
<td>SK2T-2177</td>
</tr>
<tr>
<td>IBM Networking Redbooks Collection</td>
<td>SK2T-6022</td>
</tr>
<tr>
<td>IBM Transaction Processing and Data Management Redbooks Collection</td>
<td>SK2T-8038</td>
</tr>
<tr>
<td>IBM Lotus Redbooks Collection</td>
<td>SK2T-8039</td>
</tr>
<tr>
<td>Tivoli Redbooks Collection</td>
<td>SK2T-8044</td>
</tr>
<tr>
<td>IBM AS/400 Redbooks Collection</td>
<td>SK2T-2849</td>
</tr>
<tr>
<td>IBM Netfinity Hardware and Software Redbooks Collection</td>
<td>SK2T-8046</td>
</tr>
<tr>
<td>IBM RS/6000 Redbooks Collection</td>
<td>SK2T-8043</td>
</tr>
<tr>
<td>IBM Application Development Redbooks Collection</td>
<td>SK2T-8037</td>
</tr>
<tr>
<td>IBM Enterprise Storage and Systems Management Solutions</td>
<td>SK3T-3694</td>
</tr>
</tbody>
</table>

C.3 Other resources

These publications are also relevant as further information sources:

C.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- developWorks XML zone: http://www.software.ibm.com/xml
- JavaBeans component architecture documentation: http://www.javasoft.com/beans/docs
- IBM Toolbox for Java and JTOpen: http://www.iseries.ibm.com/toolbox
- WebSphere software platform page: http://www.ibm.com/websphere
- Java Servlet API specification page: http://java.sun.com/products/servlet/2.1
- IBM WebSphere Application Server: http://www.as400.ibm.com/products/websphere
- Prerequisites for installing and running WebSphere Application Server: http://www.as400.ibm.com/products/websphere/docs/as400v302/docs/instprrq.html
How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- Redbooks Web Site [ibm.com/redbooks](http://ibm.com/redbooks)
  
  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- E-mail Orders
  
  Send orders by e-mail including information from the IBM Redbooks fax order form to:

  In United States or Canada  
  pubscan@us.ibm.com

  Outside North America  
  Contact information is in the “How to Order” section at this site:  

- Telephone Orders
  
  United States (toll free)  
  1-800-879-2755

  Canada (toll free)  
  1-800-IBM-4YOU

  Outside North America  
  Contact information is in the “How to Order” section at this site:  

- Fax Orders
  
  United States (toll free)  
  1-800-445-9269

  Canada  
  1-403-267-4455

  Outside North America  
  Fax phone number is in the “How to Order” section at this site:  

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

---

IBM Intranet for Employees

IBM Redbooks fax 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>

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, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.
# Index

## Symbols
- .war file extension 116

## Numerics
- 5250 HTML Gateway 81

## A
- Access Control List (ACL) 26
- ACL (Access Control List) 26
- AddHstData controller development 72
- adding data using the RPG program 62
- administration interface 41
- application design 16
- Application Integration pattern 4
- application scenario 16, 114
- application server 17, 80
- application topology 2
- AS/400 Directory Services 25, 26, 86
- AS/400e server 79
  - setting up 79
- authentication 25
- automatic configuration 130, 153

## B
- B2Bi 4
- B2M2B 4
- BeanInfo class 19
- bean-managed persistence (BMP) 41
- BMP (bean-managed persistence) 41
- business logic 17
- business model design 22
- business patterns 2
- Business-to-Business pattern 3

## C
- capacity plan 15
- CCITT (Consultative Committee on International Telephony and Telegraphy) 85
- CGI 81
- changing the view 55
- CMP 42
- CMP (container-managed persistence) 41
- command beans 22
- CommonVar.java 121
- communicating with RPG programs 35
- components
  - application topology 2
  - business patterns 2
  - guidelines 2
  - product mappings 2
  - runtime topology 2
- configuration testing 109
- connection pools 24
- connection properties 119
- Consultative Committee on International Telephony and Telegraphy (CCITT) 85
- container-managed persistence (CMP) 41
- Controller layer 31

## D
- DAP (directory access protocol) 85
- data repository 85
- data source 41
- database
  - mapping 41
  - restoring 113
- DataSource 134
- descriptor class 19
- design patterns 2
- design tips for EJB 46
- directories 25
  - information 28
- directory 85
- directory access protocol (DAP) 85
- directory services 85

## E
- EJB 31, 130
- EJB (Enterprise JavaBeans) 20, 39
- EJB clients 41
- EJB design tips 46
- EJB environment 146
- EJB server 40
- EJB technology 39
- EJB unit test environment 43
- EJB with WebSphere Application Server 40
eMarketPlace 4
- enterprise application design 30
- enterprise approach 30
- Enterprise JavaBeans (EJB) 17, 20, 39
Enterprise Solution Structure (ESS) 1
entity beans 41
ESS (Enterprise Solution Structure) 1
execution environment 83
exporting the environment 154
eXtensible Markup Language (XML) 20

F
flexibility 29
four stages of Patterns for e-business 9
four-stage methodology advantages 11

G
group information 29
guidelines 2

H
histdataRPG Controller development 53
HTTP administrative server 96
HTTP server 17
HTTP server configuration 96
HTTP server instance 88, 98
HTTP-based clients 41

I
IBM Framework for e-business 8
importing the environment 155
information on directories 28
Interplanetary Weather Data application 114

J
Java applications 41
Java Database Connectivity (JDBC) 24, 81
Java Naming and Directory (JNDI) 87
JavaBean Wizard 48
JavaBeans 19
JavaServer Pages (JSP) 17, 19, 81
JDBC (Java Database Connectivity) 24, 81
JDBC driver 130
JDK 1.1.7 108
JNDI (Java Naming and Directory) 87
JSP (JavaServer Pages) 17, 19, 81
JSWDK 83

L
LDAP 86

LDAP and Operations Navigator 103
LDAP client 86
LDAP concepts 85
LDAP database 151
LDAP directory 86
LDAP directory server 26
legacy 30
legacy resources 33

M
manual configuration 130
middleware 17
Model layer 32
Model-View-Controller 21
multiple administrative servers support 83
MVC 31, 35

N
Net.Data 81
n-tier 22

O
Open System Interconnection (OSI) protocol 85
Operations Navigator and LDAP 103
OSI (Open System Interconnection) protocol 85

P
Pagelist with RPG 47
PageListServlet 47
Pattern Development Kit (PDK) 1, 33
in VisualAge for Java 34
WebSphere Studio 47
Patterns for e-business 1, 5
components 2
defined patterns 3
four-stage approach 9
how to use 5
Web site 6
PC workstation 80
PCML (Program Call Markup Language) 36
PDK (Pattern Development Kit) 1, 33
PDK sections 33
personalization 25, 27
personalization techniques 27
perspective 12
phased approach 12
product mappings 2
productivity 12
Program Call Markup Language (PCML) 36
project
  publishing on the server 122
  restoring 116
  publishing the project on the server 122

Q
QEJBSBS 95
QHTTPSVR 98
Qshell Interpreter 95
QSYSWRK 95

R
restoring the database, sample application 113
restoring the project on WebSphere Studio 116
RetrieveHistoricalDataCommandRPG 35
RPG 38
RPG program component test 38
RPG program to add data 62
RPG programs 35
running the sample application 79
runtime topology 2

S
sample application 79
  application server 80
  AS/400e server 79
  configuring the environment 130
  directory services 85
  execution environment 83
  HTTP server configuration 96
  PC workstation 80
  restoring 113
  running on a Web browser 155
  system environment 79
  system setup tasks 88
  technology overview 80
  testing the configuration 109
  WebSphere Application Server 81
sample e-business application 114
scalability 24, 29
schema 41
schema map browser 42
Section A 33
Section B 33
Section C 33
Section D 33
Section E 33
Section F 33
SecureWay Directory for OS/400 87
security policy 14
servlets 18, 81, 141
session beans 41
session object 23
source code 12
stage life cycle 11
stage one 10, 17, 158
stage three 10, 162
stage two 10
stage zero 10, 13, 156
supply/chain applications 4
system environment 79
system setup tasks 88

T
technology overview 80
testing the configuration 109
the life cycle 11
three-tier 22
Toolbox for Java 32

U
UML 16
unit test environment for EJB 43
user security approach 25
User-to-Business (U2B) applications 9
User-to-Business pattern 3, 6
User-to-Data pattern 4
User-to-Online Buying pattern 3
User-to-User pattern 4

V
view component 66
View layer 31
VisualAge for Java 34, 42

W
war file extension 116
WAS Administrator Console 88, 108
Web application 136
WebSphere Application Server 81
  configuration file 88
  multiple administrative servers support 83
publishing and setting up  76
publishing and setting up for  58
WebSphere Application Server with EJB  40
WebSphere Studio  47, 116
WebSphere Studio JavaBean Wizard  48

X
X.500  85
XML (eXtensible Markup Language)  20
IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.

- Use the online Contact us review redbook form found at ibm.com/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

<table>
<thead>
<tr>
<th>Document Number</th>
<th>SG24-5999-00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redbook Title</td>
<td>User-to-Business Patterns for e-business: Developing AS/400e e-business Applications</td>
</tr>
</tbody>
</table>

Review

What other subjects would you like to see IBM Redbooks address?

Please rate your overall satisfaction: O Very Good O Good O Average O Poor

Please identify yourself as belonging to one of the following groups:
O Customer O Business Partner O Solution Developer
O IBM, Lotus or Tivoli Employee O None of the above

Your email address:
The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.

O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.

Questions about IBM’s privacy policy?
The following link explains how we protect your personal information. ibm.com/privacy/yourprivacy/
Patterns for e-business are a group of proven, reusable models that can help speed the process of developing applications. The pattern discussed in this book, the User-to-Business pattern, is the general case for users that interact with enterprise transactions and data. In particular, it is relevant to enterprises that deal with goods and services that cannot be listed and sold from a catalog.

This redbook discusses two application topologies of the User-to-Business pattern. Application topology 1 describes a situation where you build an application that has no need to connect to back-end or legacy data. Topology 2 extends topology 1 to describe a situation where you need to access existing data on legacy or third-party systems.

For practical purposes, this redbook walks you step-by-step through the implementation of an e-business application using application topology 1 and the OS/400 system. The sample e-business application and associated files are available for download from the IBM Redbooks Web site.