Redpaper

Stuart Bedoll
Rob Culp
Mike Ebbers
Bill Innis

GIS Based Enterprise Solutions with WebSphere Server and ArcGIS Server

Geographic information systems (GIS) play a role in the process of solving critical business problems in industries such as government, utilities, retail, banking, insurance, oil and gas, transportation, and health. Gathering and summarizing information about the geographic location and the movement of core business elements is essential to the operation of today's enterprises. These elements could include products, assets, or services within a commercial or government entity. Examples such as supply chain management, land management, field service delivery, and emergency response all demonstrate the importance of location information.

This IBM® Redpaper describes key components of an enterprise GIS. We focus on the interface between ESRI ArcGIS and IBM WebSphere® Application Server, where WebSphere Portal Server is deployed. We explore the integration of ArcGIS Server 9.1 and 9.2 within the WebSphere 6.0 framework. The major subsystems are described and programming examples are provided, using the Java™ 2 Enterprise Edition (J2EE™) application programming language.

Introduction

This IBM Redpaper describes key components of an Enterprise GIS with emphasis on the interface between ArcGIS Server and WebSphere Application Server where WebSphere Portal Server is deployed. Integration of ArcGIS Server within the WebSphere framework is explored. The major subsystems, including ArcGIS Server 9.2, WebSphere Application Server 6.0, and WebSphere Portal Server 6.0 are described and patterns for successful integrations are discussed, using the Java 2 Enterprise Edition (J2EE) application programming language.

Organizations are able to enhance the value of their information systems investment by leveraging the spatial information in GIS. Historically, Enterprise IT and departmental GIS often developed separately, with independent management organizations and technical infrastructures. New tools and programming styles allow organizations to bring together Enterprise IT and GIS to extend the value of both disciplines. Organizations today are realizing value through the creation of “Enterprise GIS”—the integration of Enterprise IT and GIS.

Enterprise GIS solutions provide new options. For example, IBM's WebSphere may be combined with ESRI's ArcGIS Server to build a sound JAVA platform for Enterprise GIS applications. This redpaper explores how the technologies may be integrated. We provide sample code and some lessons learned from our integration experience.

Technical overview

ESRI has developed ArcGIS Server, a solution that will enable Web-based enterprise solutions to more effectively integrate GIS capabilities. The architecture is based on a layered approach which separates and encapsulates the geoprocessing services on an independently managed application server. Access to these services is available through a rich Java object model that executes in the WebSphere Server context. The Java object model provides proxy classes that marshal the method calls to the ArcGIS Server for execution.

ArcGIS Desktop software complements ArcGIS Server by acting as a means of authoring, configuring, and maintaining data, models, and applications. This authored content can be published using ArcGIS Server, which provides the technology foundation for organizations to build and implement GIS-based Web services.

What is a spatial application?

Spatial applications incorporate the value of location information in their business processes. For example, a spatially enabled enterprise asset management application permits a manager to know a resource exists, and with spatial information, the manager will know where the resource is, relative to where it needs to be to bring more value to the enterprise.

Spatial applications are in demand by many organizational levels, from state, local and Federal governments to enterprise commercial customers. Organizations use these systems for a broad scope of solutions. A local government might want to manage building permits, land ownership, or zoning restrictions for certain areas within the city. Commercial companies can plan a marketing campaign tailored to the demographics of a particular location, or use spatial information to manage exposure to unacceptable risks.

Traditionally, GIS has been embraced by personal users and small workgroups to solve specific project or departmental problems. The benefit to the overall organization may not be

fully realized by this approach. As businesses feel increased pressure to grow revenues, provide better customer service, compete, and transform, they find that sharing the GIS analytical functions, services, and data with the whole organization provides solutions and value. An integrated approach based on a services oriented architecture (SOA) when writing a GIS application can help solve their problems.

Why run GIS on an enterprise server?

Though GIS can be implemented on a departmental server, this paper described our recommendations for running GIS in an enterprise environment. We performed our testing using a framework with a portal server and WebSphere Application Server. Enterprise-level GIS contains all the reliability, manageability, and performance requirements expected from information technology solutions. Furthermore, customer demand encouraged IBM and ESRI to expand GIS enterprise solutions to use mainframe database repositories for spatial information and analysis, because of their reliability and scalability for mission-critical applications. In addition, there are cost benefits when installing a new application on an existing system that has unused capacity. Also when application data is stored on a mainframe complex, it becomes available to the entire company.

The ESRI-IBM alliance

IBM and ESRI have formed a global alliance to serve the growing market for enterprise Geographic Information System (GIS) solutions. ESRI provides GIS software technology and IBM provides infrastructure products and services critical to make spatial applications enterprise-ready. Together, ESRI and IBM can deliver hardware, software, installation, and training, as well as outsourcing if required.

IBM has industry-leading information technology project management and systems integration capabilities necessary to deliver complex end-to-end solutions. Enterprise GIS solutions are now often seen as an information technology deployment, and IBM is well positioned to help organizations address their business problems through Enterprise GIS.

ESRI is an industry leader in Geographic Information Systems and has earned a market share leadership position. ESRI technology is used today to solve critical business problems for customers in industries such as government, utilities, retail, banking, insurance, oil and gas, transportation, and health.

ESRI based enterprise GIS solutions have frequently been deployed with IBM hardware, software, and services to address the business requirements of governments and other industry clients. IBM hardware and software technologies deployed include System p™, System x™, System z™, TotalStorage®, WebSphere, DB2® and Informix® products, and AIX®. ESRI is a Premier member of IBM PartnerWorld® for Developers and an IBM Strategic Alliance Partner.

GIS enterprise solution overview

There are a number of components that make up an enterprise GIS solution from IBM and ESRI. These include Java 2 Enterprise Edition, WebSphere Application Server, WebSphere Portal Server, ArcGIS Server, Rational® Application Developer, and RDBMS (such as DB2 and Informix) subsystems. The next section of this paper provides an overview of these components. Figure 1 shows how the components fit together.

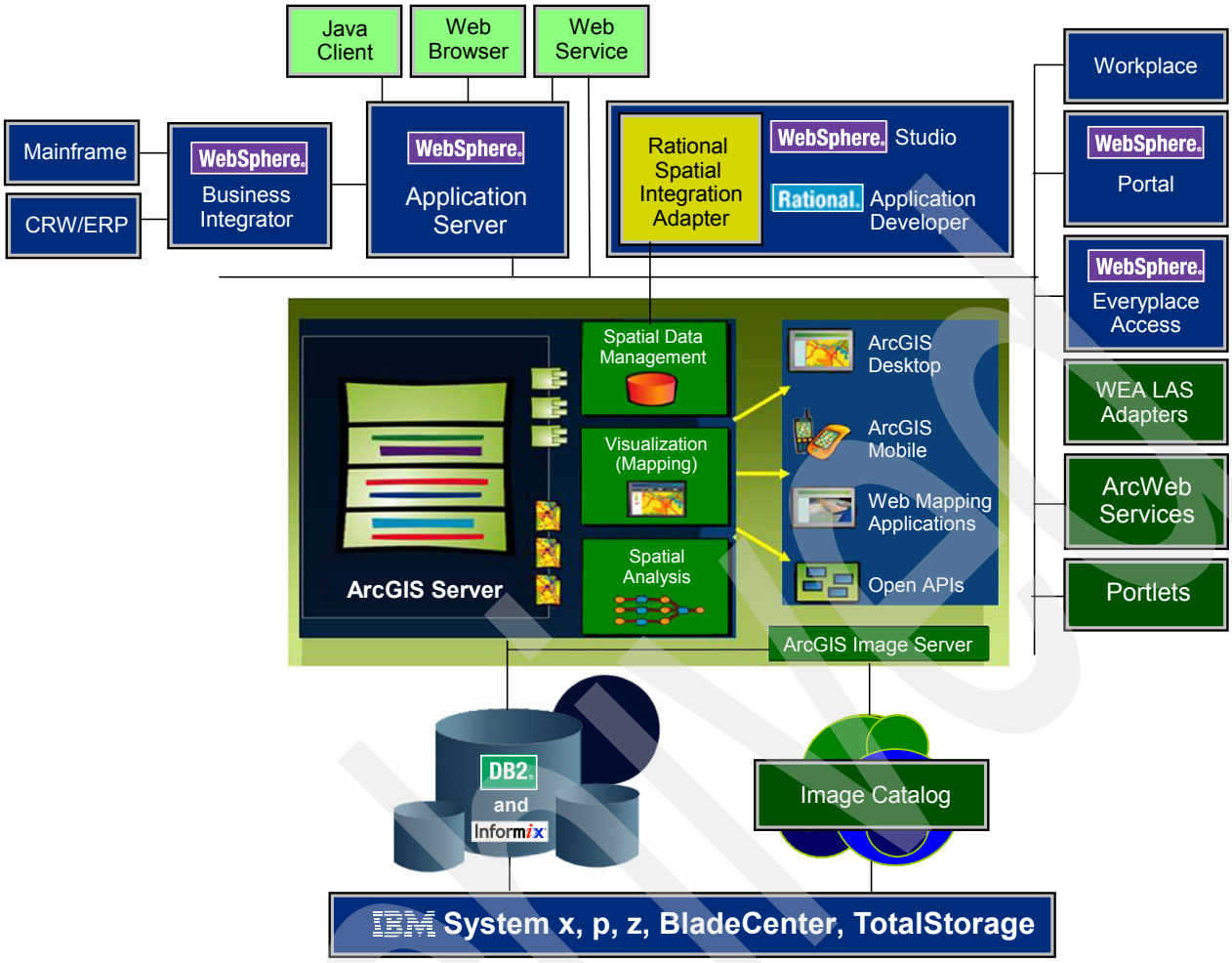

Figure 1 Enterprise GIS solution - components

Java 2 Enterprise Edition

Java 2 Enterprise Edition (J2EE) is a specification made up of many component specifications related to developing distributed applications in the Java programming language. J2EE components are used when writing Web-based applications and traditional client-server applications, and to connect to legacy resources like relational databases using a standard API. To those from a .NET development background, the Java Servlets and JavaServer™ Pages™ (JSP™) technologies are the components of most interest.

Java Servlets are Java classes that run as an extension of a Web server like IIS or the Apache Web server. A Java Servlet is analogous to an ISAPI filter, the .NET HttpHandler class, or a cgi-bin program/script. A Java Servlet runs when a client browser invokes a specifically configured URL directly or indirectly. A servlet has access to all the information in an HTTP request and can handle the request directly by providing content to be returned to the client. Alternatively, a servlet can redirect the client browser to another resource. Most J2EE Web applications use servlets primarily as targets of HTML forms, to handle user input and then process it accordingly. Generation of the response page is typically delegated to a JSP page.

JSP pages are analogous to .NET pages. That is, they are HTML pages that also contain scripting elements that execute on the server when a user requests the page. A key difference between .NET pages and JSP pages is that .NET pages use a .NET language

(such as C# and VB.NET) as their scripting language, whereas JSP pages use the Java language. A typical JSP page contains snippets of Java code and some special HTML-like tags defined in the JSP specification, interleaved with standard HTML to provide a combination of static and dynamic content. The difference between Java Servlets and JSP pages is conceptually similar to the difference between the .NET HttpHandler class and an .NET page. In both cases the former is a piece of code that can be used to emit HTML directly or redirect to other resources, and the latter is an HTML document that can contain embedded code.

Why choose J2EE?

J2EE is a well-defined set of standards, so there is a choice of J2EE implementations when deploying code. If the standard APIs are used, without vendor-specific extensions, the application will run in a variety of environments without coding changes. J2EE implementations are available on a number of platforms, from mainframes to Windows®, UNIX®, and Linux®. The application is written once, then deployed on a variety of platforms.

J2EE includes a standard API for accessing many traditional EIS systems such as CICS®, IMS™, ERP, and CRM. It also includes support for ArcGIS Server Web Services, allowing integration with .NET systems and other systems supporting industry Web services standards. J2EE includes support for a standard messaging API (Java Message Service; JMS) and an API for access to relational databases (Java Database Connectivity; JDBC™). In summary, this wide array of choices permits integration with a variety of existing systems without losing the investment in them.

WebSphere Application Server

WebSphere is the IBM brand of software products designed to work together to help deliver dynamic e-business quickly and a key building block for Service Oriented Architecture (SOA). It provides solutions for connecting people, systems, and applications with internal and external resources. WebSphere is based on infrastructure software, or middleware, designed for dynamic e-business. It delivers a proven, secure, and reliable software portfolio that can provide an excellent return on investment.

The technology that powers WebSphere products is Java. Over the years, many software vendors have collaborated on a set of server-side application programming technologies that help build Web-accessible, distributed, and platform-neutral applications. These technologies are collectively branded as the Java 2 Platform, Enterprise Edition (J2EE) platform. This contrasts with the Java 2 Standard Edition (J2SE™) platform, with which most clients are familiar. J2SE supports the development of client-side applications with rich graphical user interfaces (GUIs). The J2EE platform is built on top of the J2SE platform. J2EE consists of application technologies for defining business logic and accessing enterprise resources, such as databases, Enterprise Resource Planning (ERP) systems, messaging systems, e-mail servers, and so forth.

The potential value of J2EE to clients is tremendous. Among the benefits of J2EE are:

- An architecture-driven approach to application development helps reduce maintenance costs and allows for construction of an information technology (IT) infrastructure that can grow to accommodate new services.
- Application development is focused on unique business requirements and rules, such as security and transaction support. This improves productivity and shortens development cycles.
- Industry standard technologies allow clients to choose among platforms, development tools, and middleware to power their applications.

- Embedded support for Internet and Web technologies allows for a new breed of applications that can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Another exciting opportunity for IT is ArcGIS Server Web Services. Web services allow for the definition of functions or services within an enterprise that can be accessed using industry standard protocols that most businesses already use today, such as HTTP and XML. This allows for easy integration of both intra- and inter-business applications that can lead to increased productivity, expense reduction, and quicker time to market.

WebSphere Portal Server

The main purpose of a portal is to provide a user with a single point of access to multiple types of information. A portal aggregates information in a way preferred by the user, regardless of the location or format of that information. In addition, a portal can be made accessible from multiple types of devices, such as a Web browser on a personal computer, or a microbrowser on a mobile phone.

The IBM WebSphere Portal allows companies to build their own custom portal Web site to serve the needs of employees, business partners and customers. Users can sign on to the portal and receive personalized Web pages providing access to the information, people and applications they need. The demands for a consistent, seamless, device-independent access to relevant applications and information are increasing and the WebSphere Portal Server, with its open framework, allows other technologies to plug in to this framework. To that end, the WebSphere Portal Server provides the following:

- A flexible framework and infrastructure
- Support for multiple data formats and different devices
- User enrollment, authentication, and authorization
- Portal page customization
- Personalization
- Support for portlet installation
- Samples, plug-ins, and common portal features such as a search feature

ArcGIS Server

ESRI ArcGIS Server is a highly flexible and scalable technology that runs on WebSphere infrastructure and supports geospatial SOA initiatives. ArcGIS Server provides the technology foundation to build and implement a set of GIS-based Web services. Some common services include map (2D) and globe (3D) services (transportation, demographics, physical environment,), locator services (geocoders and gazetteers), geoprocessing services (site selection models, network analytics, and raster analytics), and data management services (replication; data check-in/checkout; transformation, and loading; and catalog services). Shared GIS services such as these can be used to add value to core business systems (e.g., CIS, ERP, CRM, etc) and enterprise-wide initiatives for collaborative computing.

ArcGIS Server supports a series of open APIs and standards that allow virtually any other client (for example, desktop, Web, and SQL-based applications) to interact with and use the mapping, spatial analysis, and data management services of ArcGIS Server. These services can also be called on, and integrated with, other Web services using standard Web service protocols such as SOAP and XML.

IBM Rational Spatial Integration Adapter (RSA)

The WebSphere Studio product family, recently rebranded to become part of the IBM Rational Software Development Platform, offers development tools that are based on the Eclipse open-source tooling framework.

The Eclipse runtime basically serves as a container for tools, offering some common functionality. Each tool is embedded into the runtime as a so-called plug-in, literally plugging in to some basic Eclipse functionality. Tools can also require other tools, or extend other tools' functionality, and each tool is extensible in itself, offering a number of well-defined extension points.

WebSphere Studio, most notably the WebSphere Studio Application Developer (Application Developer) product, consists of a large number of Eclipse plug-ins that offer tooling for building Java and J2EE applications. One of these plug-ins is the Java ServerPages (JSP) Page Designer tool. It offers a WYSIWYG visual editor to build a JSP. UI components can be inserted into a page simply by dragging and dropping from a palette. JavaScript™ and other logic can be added manually or by using wizards.

A JSP is the delivery mechanism for a JSF-based application. JSF UI components are inserted into a JSP using specific, standardized tag libraries. Application Developer allows a programmer to build a JSF-based application by adding a number of JSF UI components to the palette in the JSP Page Designer. The fact that there is a well defined, MVC-based programming model for JSF means that new applications can be rapidly developed, using just the visual page editor.

It is exactly this functionality that the Spatial Integration Adapter is built on top of. The product is developed as an Eclipse plug-in that extends the JSP Page Designer. It adds geospatial components, such as maps, to the palette of UI components, thus allowing for a completely new type of application to be built. Geospatial functionality, encapsulated in JSF, becomes available as another piece of UI. A map can be dragged and dropped into a Web page, just as a text field or a table would be inserted.

For more information, visit this Web site:

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0502_tost/0502_tost.html

Rational Application Developer

Rational (formerly WebSphere Studio) Application Developer provides a best-in-class integrated development environment for building, testing, integrating, and deploying J2EE applications, Web services, and business processes. Features include the following:

- Eclipse-based user interface
- BPEL4WS development environment
- J2EE development environment
- Java development environment
- Web services development environment
- XML development environment
- Relational database tools
- Web development environment
- Team development

- Server tools for testing and deployment
- Tracing, monitoring, and performance analysis tools
- Debugger

See “Appendix G. IBM spatial support subsystems” for an overview of IBM spatial support subsystems.

Architecture and design models

Now we describe the task of creating a GIS application. We present three views to help visualize our sample application:

- Application component view
- Application deployment view
- ArcGIS Server process view

In “How to program a GIS application” on page 16, we show two additional views that will help in create a working GIS program. These are:

- Enterprise GIS Web Application logical view
- Use case view – our sample application

First, let us review the products involved. IBM supplies these:

- Java 2 Enterprise Edition
- WebSphere Application Server
- WebSphere Portal Server
- Rational Spatial Integration Adapter (RSA) for ArcGIS Server 9.1 ONLY
- Rational Software Architect/Rational Application Developer Version 7 (Version 6 is not supported due to ArcGIS Server Java ADF 9.2 Eclipse 3.1 dependency). See “Release 9.2 installation dependencies” on page 43.
- DB2 or Informix.
- Extender products, such as DB2 Spatial/Geodetic Extender or Informix Spatial/Geodetic DataBlade™
- An operating system, such as Windows or AIX. Linux is also supported.

ESRI supplies these:

- ArcGIS Server (includes ArcSDE and ArcIMS)
- Eclipse Java ADF Plug-ins (Eclipse Version 3.1 or later): Core, Engine, and Server

Enterprise GIS Application Component View

This component model provides a view of the relationships between the various ArcGIS desktop components, ArcGIS Server components, and WebSphere Application Server components. See Figure 2 on page 10.

An important distinction in the ArcGIS architecture is the author versus consumer context. The GIS server hosts GIS services and the publishing model involves authoring, publishing, and consuming services. ArcGIS Server makes it easy for you to publish services to your GIS server, build Web applications that access your services without writing any code, and administer your server over the Web, using a tool called ArcGIS Server Manager. The GIS functionality provided by the server has been extended. In addition to publishing maps, you

can publish globe documents, you can run sophisticated geoprocessing models on the server, and you can publish geodata services that enable users to perform database replication and synchronization. In addition to these new service types, map services now have capabilities to support network analysis and the WMS and KML specifications. WMS is an OGC specification for 2D maps and KML is an industry specification for 3D maps.

ArcMap and ArcGlobe, two core components of ESRI ArcGIS desktop software, can be used to author map and globe documents respectively which can then be associated to a MapServer or GlobeServer object on the ArcGIS Server. In addition ArcMap and ArcGlobe can be configured to consume the map document that has been deployed on a MapServer or GlobeServer respectively. ArcGIS Explorer, a new ESRI ArcGIS desktop solution in 9.2, can be used to consume the globe document that has been deployed on a GlobeServer. This enables a GIS analyst to deploy their work for other users to consume from a centralized server that has access to the enterprise geodatabase. Note: communication between the desktop clients and ArcGIS Server can be accomplished through either HTTP or TCP/IP. This enables an enterprise to configure HTTP access to an ArcGIS Server through a firewall.

The enterprise geodatabase is made available using ArcSDE. Once you've published services to your GIS server, there are several different ways you can allow people to access them. You can use a new client to ArcGIS Server, ArcGIS Explorer, to view 2D and 3D map layers and to get details about your data. Additionally, you can continue to use ArcGIS Desktop and ArcReader to access your GIS services. You may also want to display your GIS services in a Web application. By deploying GIS services through a Web application, people do not need any application installed on their computer other than a Web browser.

The ArcGIS Server component contains new Web integration available in 9.2. You can use ArcGIS Server Manager to create Web applications, publish ArcGIS Explorer maps to the GIS Server, enable WMS services, and create KML network links. It provides the following capabilities:

- Server Manager Web application
  - Management of the ArcGIS Server components using a thin client Web browser interface as opposed to the ArcCatalog thick client component.
  - Authoring and deployment of simple GIS Web applications using a form based User
  - Interface. For each application you create you can control the geographic content (map services) being displayed, the operations or tasks that the application performs, and which map surrounds to display with your map, such as the header image, the table of content, an overview map, and so on. This is a really quick way to create and deploy a Web based GIS application without requiring an independent Web Server like WebSphere or Apache.
- ArcGIS Server Web Services Handler enables access to ArcGIS Server processing using Web Service protocols from an enterprise Web application deployed to a Web or Portal server. In addition, the Web Service Handler can be configured with various authentication types (HTTP, Form-based, Certificate-based).

To facilitate the authoring of enterprise GIS applications, ESRI provides an ArcGIS Server Eclipse 3.1 plug-in. This plug-in is compatible with the Rational Software Architect V7 and Rational Application Developer. This artifact is depicted in Figure 2 as an RSA plug-in. An enterprise GIS application is then authored and deployed to a Web server which is configured to access the ArcGIS Server as well as the ArcGIS Server Web components. ArcGIS Server for the Java platform provides tight integration with plug-ins for Eclipse IDEs. These plug-ins give developers a very rich programming experience with integrated help, wizards, code snippets, importable samples, and templates for building ArcGIS-based Web and enterprise applications.

Note: IBM developed the Spatial Integration Adapter for RSA to facilitate development of ArcGIS Server application using ArcGIS Server 9.1. As of this writing, this is not yet compatible with ArcGIS Server 9.2.

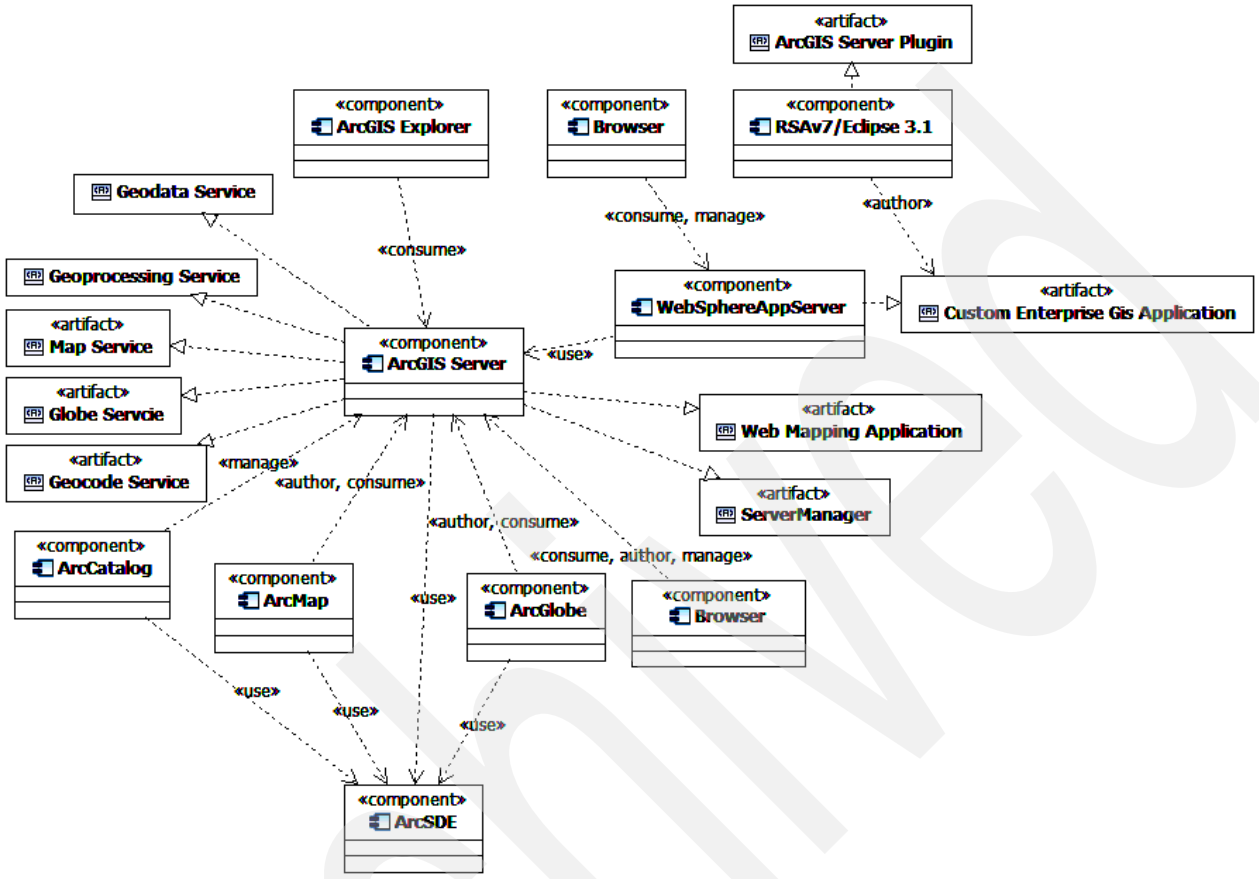

Figure 2 Enterprise ArcGIS - component model

Enterprise GIS Application Deployment View

The following sections depict high level component model of an Enterprise GIS application based on the ArcGIS Java architecture. The nodes consist of a WebSphere Application Server or WebSphere Portal Server (Note: needs to be tested on 9.2) with communication links to the ArcGIS Server. The enterprise application is accessed with a Web Browser. In Release 9.2, ESRI added ArcGIS Server Manager. It allows you to administer your server over the Web and makes it easy to build Web applications that access your services without writing any code.

WebSphere Application Server

The Portal Server or Web Server should contain the following packages:

- Custom Enterprise Web Application. You build this package to contain the JSF MVC framework that provides standard Web based behaviors associated with the enterprise application including access to an enterprise database server such DB2. It also would contain “spatial” components which provide GIS application behaviors integrated in the enterprise application. This includes access to the enterprise Geodatabase. Note: Global Transaction management is not available between the Geodatabase and the Enterprise databases.

- ArcGIS Server Web Application Development Framework (ADF). This package is installed through the ArcGIS Server Eclipse plug-in. It contains the subsystems provided by the ArcGIS Server Application Development Framework.
  a. It includes the Web controls subsystem, which contains JSF and Java Bean Web components for integrating GIS capability in a Web page.
  b. It also includes the ArcObjects API subsystem, which provides a large loosely coupled set of Java objects that provide access to a robust set of GIS capabilities. These capabilities are executed on the ArcGIS Server which is accessed using the MS DCOM protocol.

  A new feature in Release 9.2 extends the ArcGIS Server API to be based on Web Services. See Figure 5 on page 14.

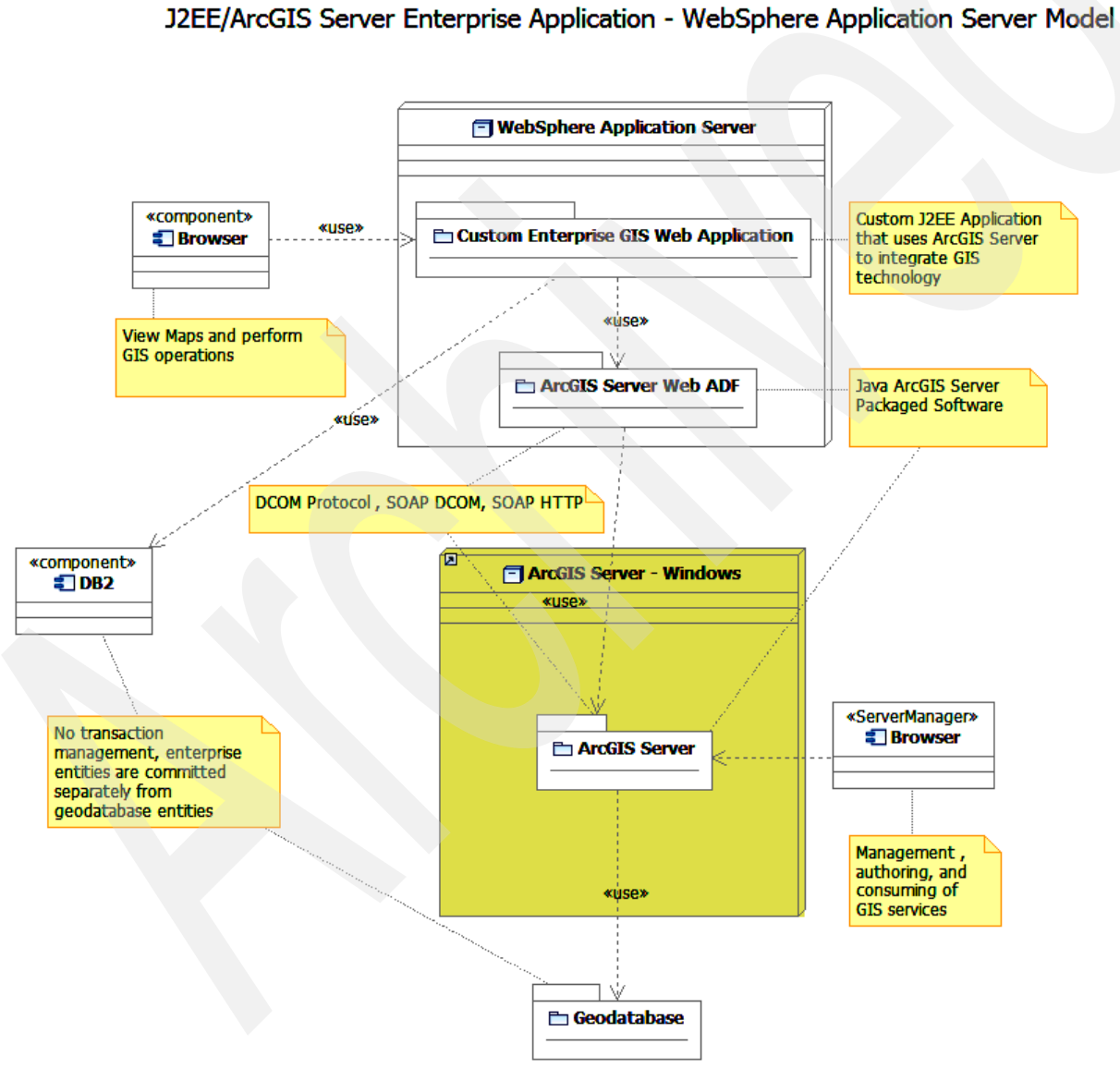

Figure 3 ArcGIS Server and WebSphere Application Server - component model

Portal Web Server With ArcGIS Server

The Portal Server or Web Server contains the following packages:

- Custom Enterprise Portal Application. You build this package to contain portlets that provide standard Web based behaviors associated with the enterprise application including access to an enterprise database server such DB2. It also would contain “spatial” portlets that provide GIS application behaviors integrated in the enterprise application. This includes access to the enterprise Geodatabase. Note: Global Transaction management is not available between the Geodatabase and the Enterprise databases.
- ArcGIS Server Web ADF. ArcGIS Server offers developers a rich, standards-based set of components for building and deploying geospatial applications. For the Java developer, the ArcGIS Server’s Software Development Kit (SDK) includes:

  a. ArcGIS Server Web ADF components for building and deploying Java Web applications and services

  b. ArcGIS Server Enterprise ADF components for J2EE 1.4 Enterprise JavaBean (EJB™) development

  The ADF components are tightly integrated into common development environments such as Eclipse to enhance the developer experience and boost productivity.

  This package is the same as defined above. However, the 9.1 implementation of the APIs includes tested support for JSR 168 portlets. Release 9.2 for portlets is being investigated.

ArcGIS Server Web ADF

For building and deploying applications and services to the Web-tier, developers can leverage developer components with the Java ADF framework. ArcGIS Server ADF for the Java platform supports Java 5.0 and runs in application servers that support Java 5.0. With this set of components you can rapidly create a Web based mapping and analysis application. Not only do the components provide a broad array of functionality out of the box, but also they have been designed to allow for easy extension working with multiple data sources, such as ArcGIS Server and ArcIMS. The Java ADF (Web) controls and framework are JavaServer Faces (JSF) components for working with geographic data sources. By using the JSF framework, you can leverage an ecosystem of JSF components from ESRI and other vendors to quickly build highly interactive Web-based GIS applications.

For server-based applications, the DCOM protocol is supported using a 100% Java implementation to talk to COM components on the server-side. No native code is required. Web ADF applications use a 100% Java implementation of the DCOM protocol to connect to server tier. Here is a runtime scenario.

ArcGIS Server Enterprise ADF

ArcGIS Server's Enterprise ADF supports development of J2EE applications powered by Enterprise JavaBeans™ (EJB) technology by providing out-of-the-box EJBs ready to use, ready to deploy across a wide variety of J2EE 1.4 certified application servers and completely integrated with ArcGIS Server Manager for point click ease. Benefits include:

- Support for development of J2EE applications powered by EJB technology with point-click ease
- Out-of-the-box Geospatial EJBs ready to use, and ready to deploy across a wide variety of J2EE 1.4 certified application servers
- Ease of integration with other application developer framework components such as ArcGIS Server Web ADF, and Eclipse IDE

Enterprise ADF applications take advantage of the Java Connector Architecture to enable connectivity between J2EE applications and the ArcGIS Server.

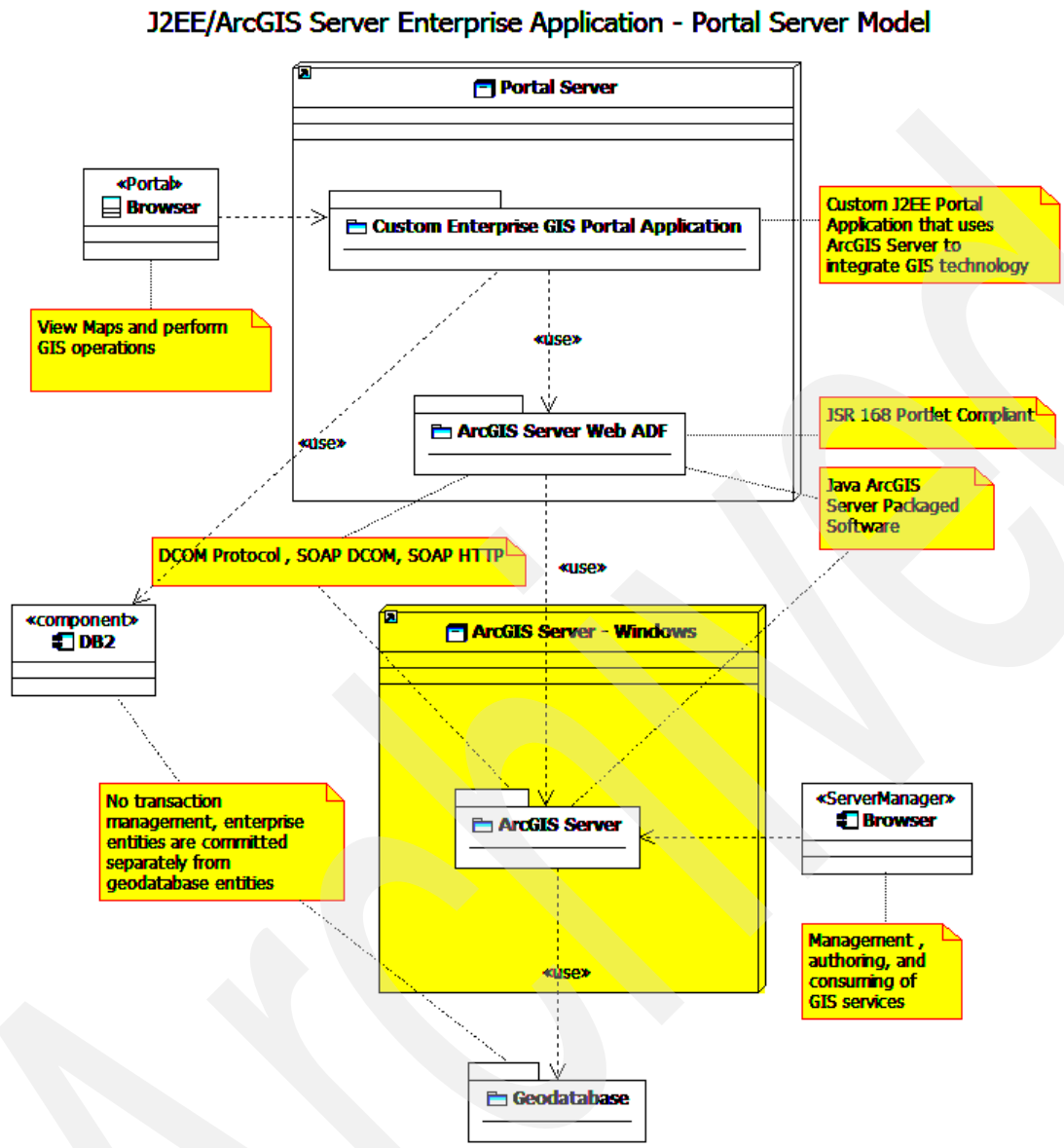

Figure 4 ArcGIS Server/WebSphere Portal Server Component Model

ArcGIS Server Web ADF Model

The Web ADF architecture consists of three tiers. The first tier is the view, or client tier, which is composed of the Web controls. The bottom tier, or model tier 2, is the GIS business objects tier which provides access to GIS data sources and exposes functionalities on those data sources. In between these are the model tier 1 objects that mediate between the view tier and the GIS business objects.

The controls are the first tier, and they cover the view/controller part of the MVC architecture. Controls are in both the view and controller because they not only interact with and render output to the client but can also affect application flow during the phases of the request lifecycle.

The Javadoc™ for the controls can be found in the package com.esri.adf.web.faces.component, which is located at:

http://edndoc.esri.com/arcobjects/9.2/Java/api/adfwebcontrols/com/esri/adf/web/faces/component/package-summary.html/

The tier below the controls is the first tier of the model in the MVC architecture; these are the data objects that work directly with the Web controls. These objects usually connect to business objects in other tiers but are not required to.The Javadoc for these objects can be found in the com.esri.adf.web.data package, which is located at:

http://edndoc.esri.com/arcobjects/9.2/Java/api/adfwebcontrols/com/esri/adf/web/data/package-summary.html/

Also within this package are the interfaces and base classes for model tier 2. These objects, such as GISResource and MapFunctionality, are not tied to working with any of the presentation tier; instead they are closely tied to GIS data and analysis.

In summary:

- In model tier 1, the controlling object is the context, and all the other controls have to be attributes of the context to work properly.
- In model tier 2, there is a concrete implementation of GISResource for each data resource.
- For a resource to work with the controls it has to be registered with the context.
- In model tier 2, there is a concrete implementation of GISFunctionality for each functionality for each data source that has that type of functionality.
- Each of the model tiers exposes functionality that you can use out of the box without having to communicate directly with the corresponding servers.

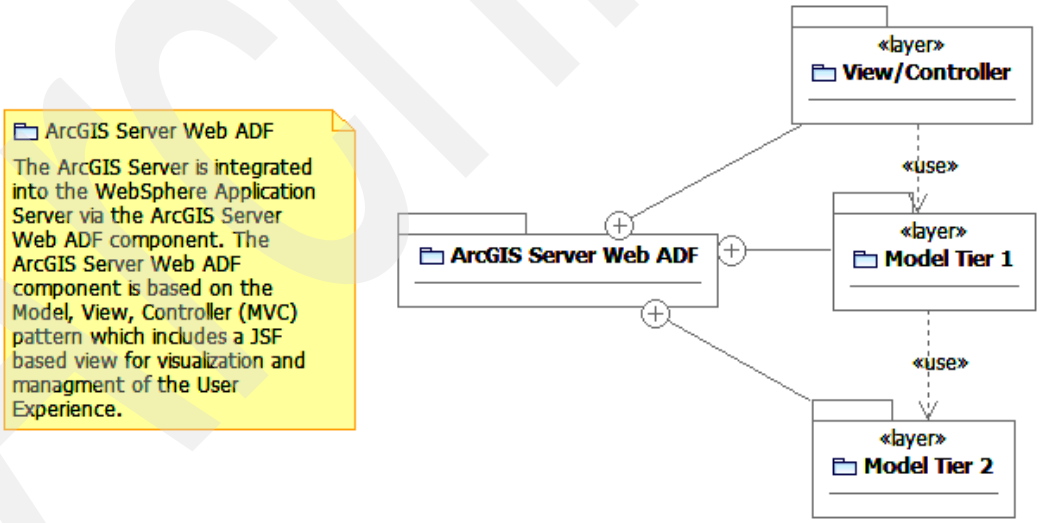

Figure 5 ArcGIS Server Java Integration Subsystems

ArcGIS Server Process View

The ArcGIS Server package defined in Figures 33 and 4 is composed of two main processes. These processes can run on Windows, Linux (Intel®, Red Hat, and Suse), IBM-AIX[1] and HP-UX[1], and Sun™ Solaris™ servers. The main processes are:

- Server Object Manager (SOM):
  - Acts a broker between the client (ArcObject proxy from ADF toolkit) and the Server Object Containers (SOC) and then steps out of the communication path.
  - The SOM can manage SOCs across multiple physical nodes.
  - Communication protocol: DCOM
- Server Object Container
  - Manages the life cycle of the ArcObjects
  - Configured as a MapServer, GoecodeServer, GlobeServer, GPServer, and GeodataServer. New in 9.2 are the GlobeServer, GPServer, and GeodataServer.
  - Can manage one or more ArcObjects depending on the isolation configuration.
  - Memory footprint: Small map = 30 MB process
  - Performance: Has not been tested as of this writing
  - Deployment: Can be deployed across multiple physical nodes associated with a single SOM.
  - Communication protocol: DCOM

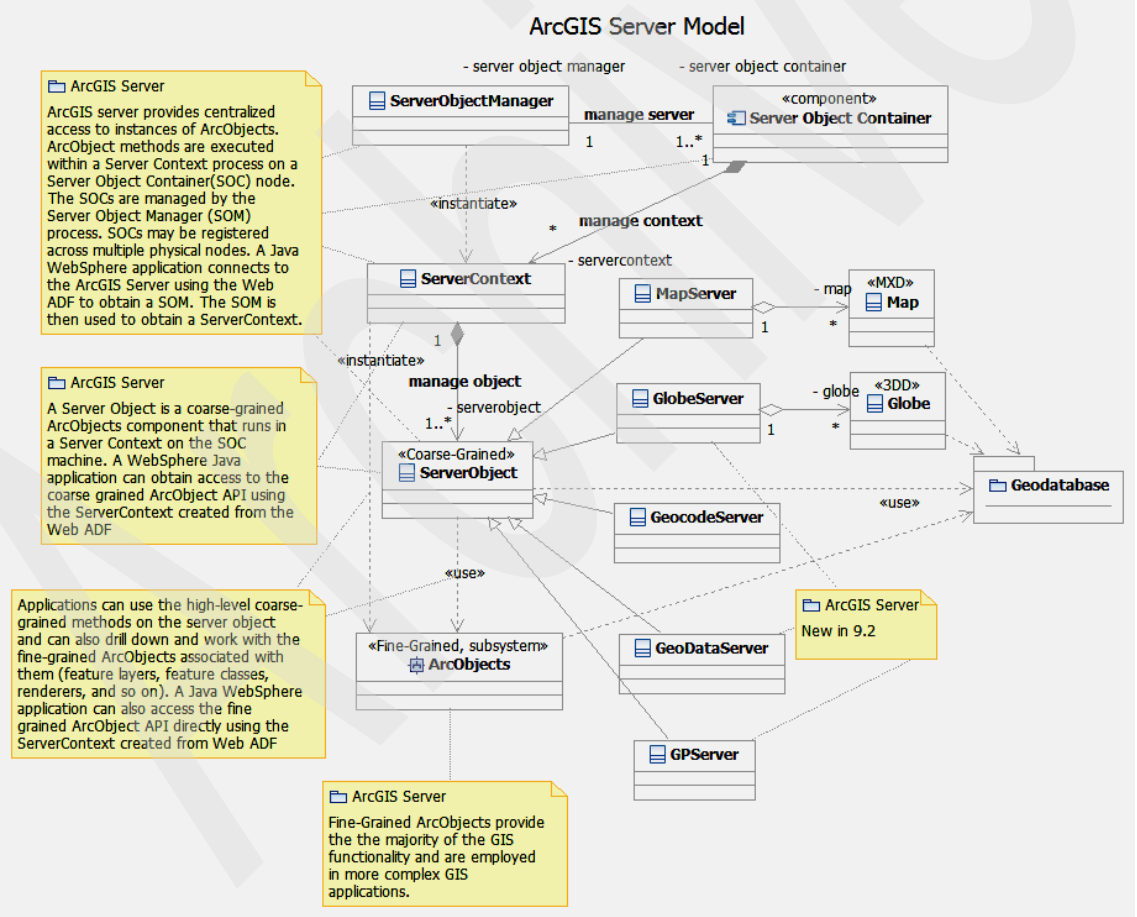

Figure 6 ArcGIS Server - process model

[1] Only ADF runtime components are supported. SOM® and SOC are not supported on this platform.

How to program a GIS application

Now that we have introduced the GIS concepts and software, this section continues with a Logical View of a J2EE application that integrates GIS capabilities using ArcGIS Server and ArcGIS Server Java Application Development Framework (ADF). This is followed by a use case view which articulates some simple enterprise GIS applications using ArcGIS Server. We take you through the necessary steps to program a GIS application using ArcGIS and J2EE.

This section was written from our own programming experience. Therefore, some of our examples are taken from ArcGIS Release 9.1 and some from the latest release, 9.2. You will see these indicated in the section headings. As we try out more examples on the new release, we will update this document. We had developed a number of use cases based on ArcGIS Server 9.1 and WebSphere Portal Server. We included some design views in the appendixes. We had not refactored this solution to reflect ArcGIS Server 9.2 at this time.

Enterprise GIS Web Application Logical View

The following sections show the logical view of an enterprise GIS Web application using WebSphere Application Server framework. As of this writing the WebSphere Portal Server framework was tested for ArcGIS Server 9.1 only and is included in the Appendixes.

Enterprise Web Application Software Model (9.2)

The following class diagram defines the objects supplied by the ArcGIS Server Java ADF and how they relate to a JSF/JSP Web page that integrates GIS capability. The Web page creates an ArcGIS Server context with an association to a Map, Overview Map, and Map Table of Contents. The object model maintains application state using the WebApplication Java bean, session state using the WebSession Java bean, and page state using the WebContext Java bean. The WebContext bean anchors the Java beans that manage the state of the Maps (main and overview) and Table of Contents on the page. It also maintains the specific attributes that identify the specific ArcGIS Server, Server Object, and authentication credentials in the AgsLocalMapResource and AgsUser Java beans. This object model is pulled together using the faces-config.xml file and the context-attributes.xml file. The details of these files are defined later in this document.

Custom ArcGIS Server solutions can be integrated into a JSF architecture through the Faces event model. A simple example of this is shown in this diagram by the references to the ArcGIS Server API and packaged custom GIS services (see “ArcGIS Server API Facade example” on page 34) from the FacesMapBackerBean object. In this case the FacesMapBackerBean object has been defined to JSF to handle User events from the Map jsp. The 9.2 Java ADF introduces a Task framework which enables custom GIS solutions to be easily integrated with this model. An example of this is shown in “Java ADF Task Framework Overview” on page 26.

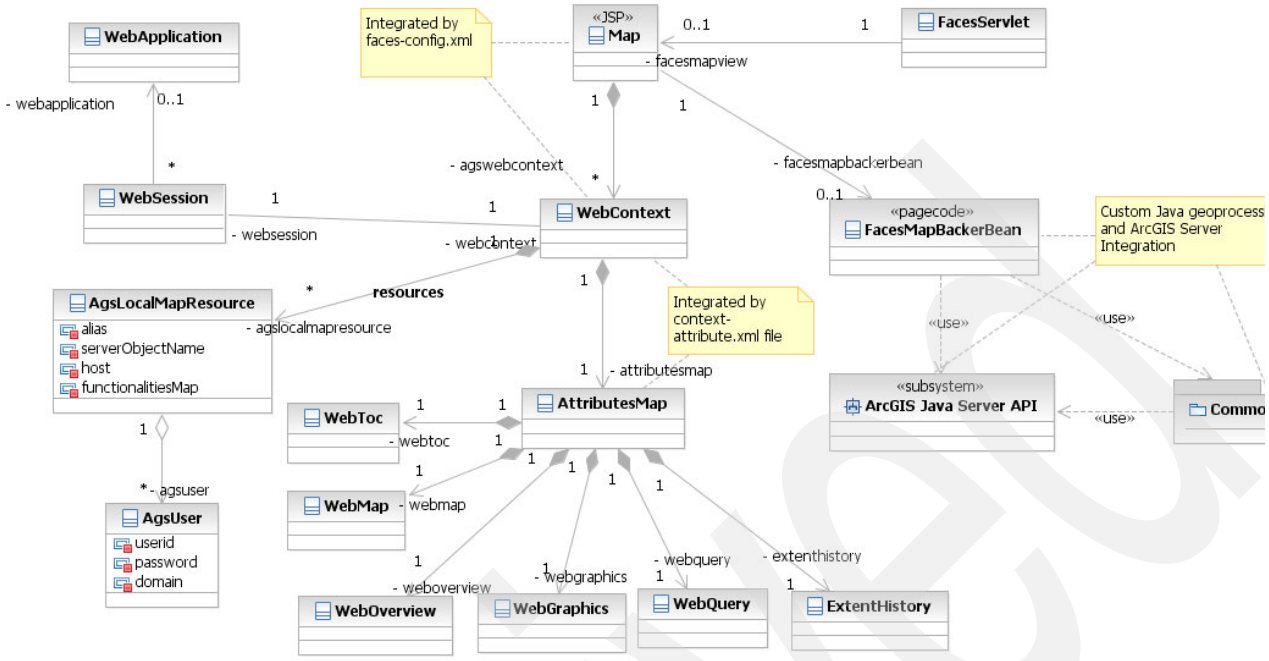

Figure 7 Web Map Application - class model

Use case view

The following use case model shows two use cases that are specified in the following sections in detail.

Overview of use cases and features

These use cases are simple but architecturally significant to an enterprise Java application that includes GIS function. The first use case (base use case) enables a GIS User to view a map of a geographical area. The location (latitude/longitude), size (spatial extent), and types of information (spatial layers) included in the map depend on the application and do not affect the mechanics of implementation. This is the base use case which lays the foundation for other GIS applications that visualize spatial content.

The second use case is an extension of the base use case. It adds the capability to subscribe to a GeoRSS feed. RSS (Really Simple Syndication) has become a more prevalent way to publish and share information using XML. GeoRSS is an Application Profile of the GML (Geography Markup Language is an XML schema) which enables one to request, aggregate, share, and map geographically tagged feeds. This use case demonstrates how one can customize the ArcGIS Java framework while applying current Web technologies to a GIS application.

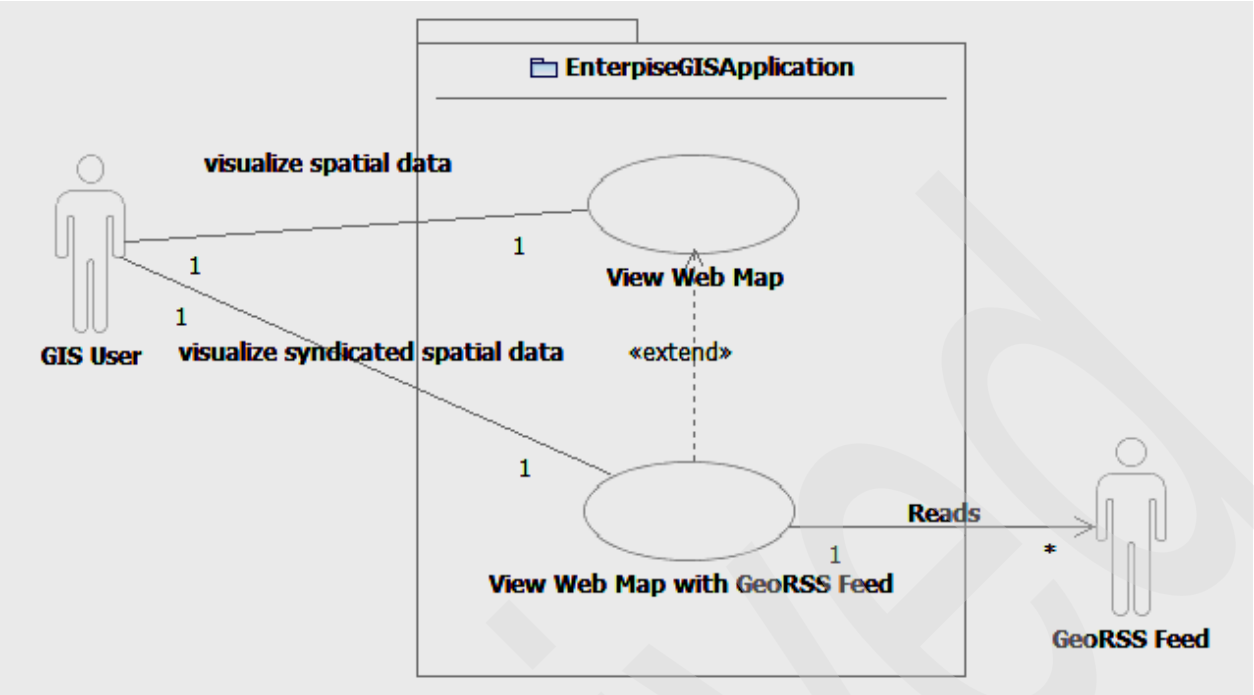

Figure 8 Enterprise GIS use case Model

The following sections provide many of the details behind developing a solution for each of these use cases using the ArcGIS Server Java ADF.

View Web map (9.2) use case

The objective of this use case is to demonstrate a simple application that will render *spatial content* in a Web browser. Spatial content can take on many forms, such as a set of polygons in the shape of the states in the United States or a set of points representing cities in the United States. In general, the states can be viewed as one layer and the cities as another. Both layers can be packaged together in a *map* and viewed together or individually. A *Table of Contents* will list the layers on the Web page and allow the user to control which layers are displayed.

The map below shows a partial view of United States with many layers. Each layer (feature class or raster data set) contains a set of polygons representing various administrative boundaries. The layer shown in the map is the Fire Planning Unit (FPU) layer which defines the geographic boundaries used for wildland fire management by the U.S. Government. The light blue area is a *raster* layer that represents thematic information about a specific FPU. You can see that the raster layer maps directly over the administrative boundary of the FPU. The administrative layers are considered *reference* layers that can be obtained from the U.S. government. The raster layer was generated by a custom application and added to the map in this view. Thus, the map can consists of both reference layers and application specific layers. The *overview* map provides a contextual view of the spatial extent shown in the main map.

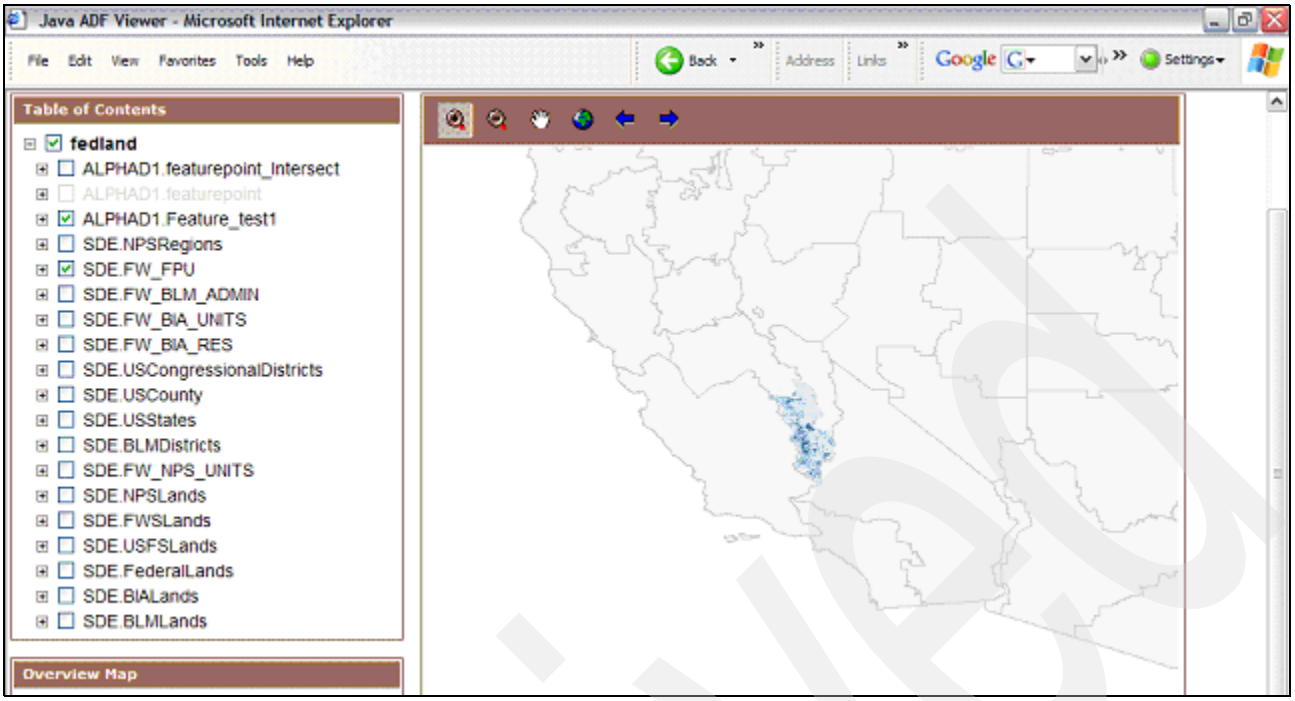

Figure 9 Federal Lands map with raster layer

The following components are required to build this application:

- Rational Software Architect Version 7
- ArcGIS Java ADF plug-ins for RSA 7: core and server.

  Note: RSA Spatial Integration Adapter (ArcGIS Server Java ADF) only supports ArcGIS Server Release 9.1.
- WebSphere Application Server (6.0)
- ArcMap
- ArcGIS Server Map Server Object
- Spatial content options:
  - Enterprise GeoDatabase:
    - ArcGIS Spatial Database Engine (ArcSDE)
    - Relational Database Server such as DB2
  - Personal or File Geodatabase
  - Shape Files (an ESRI file format that contains spatial content)

Creating a map

Prior to building the Web map application, one needs to create a map. A map is basically a container document that references the layers the map is made of. These layers can exist in various formats and in various locations. The map document is built using ArcMap. In this use case a map is created with layers of Federal Administrative boundaries (for example, states and congressional districts) and a Raster Data set which represents an area of interest. These entities were initially added or created in an enterprise SDE database and subsequently added to the map document using the ArcMap desktop tool.

Once the map is created, follow the steps for creating a map service with ArcGIS Server using this map.

The following model is a logical view of a map document with references to Feature Class (polygon, lines, points, data sets, and so on) and Raster Data sets as an example. These objects may be accessed in a Web Application using the ArcGIS Server API. The Raster and Feature Class entity types are typical; however there are other entity types that are possible.

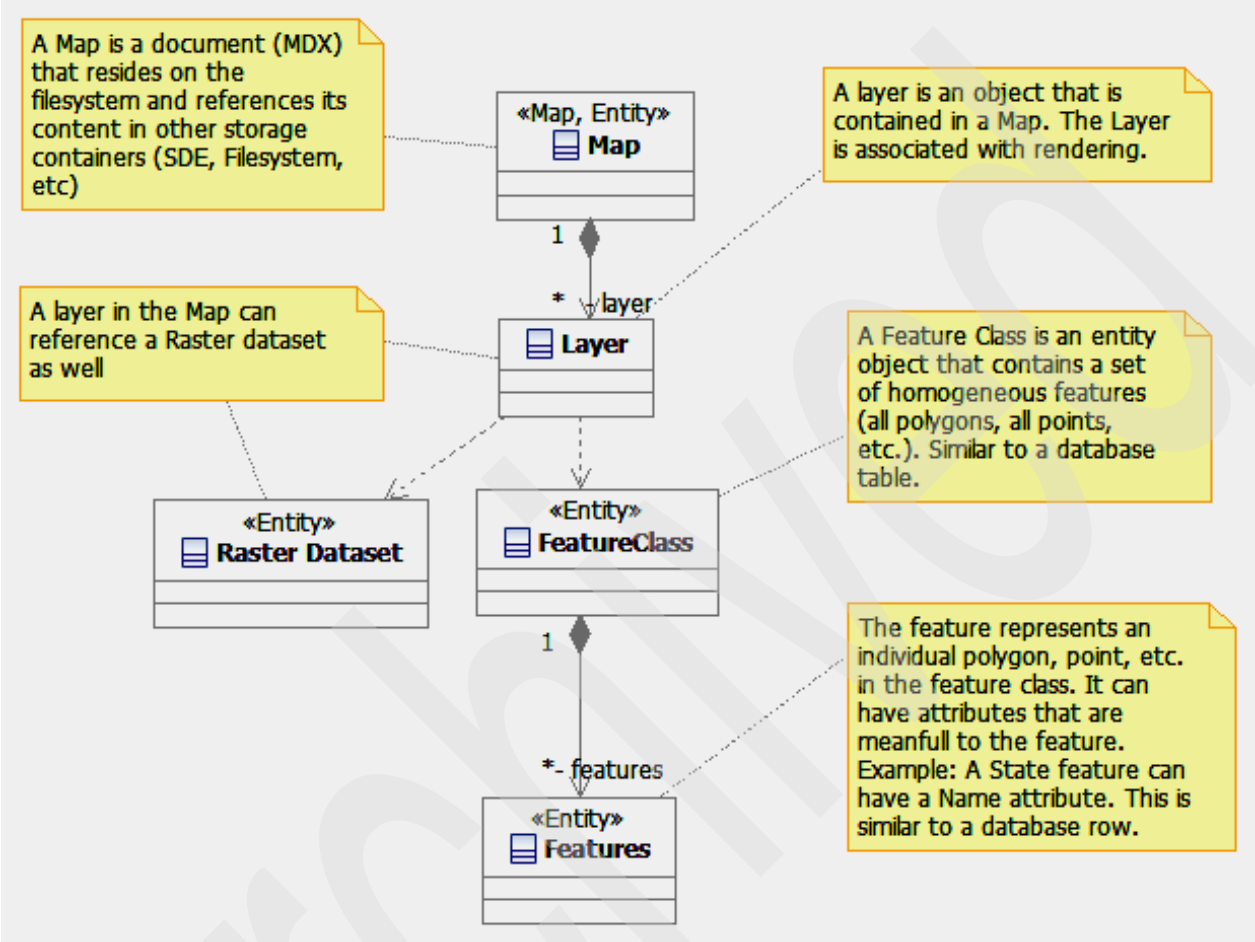

Figure 10 Logical model of a map document

Creating a spatial Web application based on servlets

Here are the steps for creating a GIS Servlet:

1. New project → Project Wizard →ESRI Templates →Server →ArcGIS Web Sample
2. Next →
3. Assign project name
4. Next →
5. Select ArcGIS Server Viewer as the template
6. ArgGIS Web Project form is displayed with multiple tabs:
   - ArcGIS Server Local tab, enter the following and test the connection
     - GIS Server
     - Username
     - Password
     - Domain
   - ArcGIS Server Internet tab

    - N/A
  - ArcIMS Local tab
    - N/A
  - ArcIMS Internet tab
    - N/A
  - ArcWeb Services tab
    - N/A
  - WMS tab
    - N/A
7. At this point you can connect to the ArcGIS Server entered above and select the map service that will be referenced by this application.
8. Press Finish and the Web project will be created seeded with sample JSPs, ESRI Javascript libraries, and ESRI jar files. The faces-config.xml and context-attributes.xml file will be populated with the information entered above.

The following are the sample JSPs that are generated. We focus on the map.jsp.

- download.jsp
- error.jsp
- footer.jsp
- index.html
- login.jsp
- map.jsp
- timeout.html
- uload.jsp
- uploadsuccess.jsp

At this point you can run the Web map view application by deploying the Enterprise Application (EAR) to a WebSphere Application Server and browsing to the following URL:

```
http://<hostname>:<port>//<project name>
```

This will start the index.html page and bring up the map.jsp with a view of the map created above.

map.jsp

The map.jsp is a significant artifact which contains the ArcGIS Server JSF tags that will render the map on the HTML page. The objective of this section is to identify the significant Java ADF tags that one must be familiar with.

Any JSP tag in the page that begins with an “a” indicates that it is referencing part of the Java Web ADF. In addition, it defines the tools that provide general map functions such as pan and zoom for this page. These tools are mapped to javascript libraries that were added to the ArcGIS Server Web project when it was created.

ArcGIS Server Web ADF tags: There are several Web ADF tags in map.jsp starting with the <f:view> tag which is the parent tag to all JSF specific tags. Because the view tag establishes the root node for the JSF view tree, all output managed by JSF must be inside the view tag.

The next JSF tag is <h: form> which is required to put controls inside of an HTML form element. This tag allows for the application to submit information back to the server. Any tags that post information to the server, such a tool, need to be within a form tag. Following this tag is the <a:context> tag, which links the context control declared in the faces-config to this JSP page. The linkage between the view tier and model tier 1 happens through the value binding of the context tag to the managed bean named mapContext. Because of the value binding all the resources and model tier 1 objects associated with the context are now available on this page. It is possible to have multiple contexts on the same page, but they would all need to be declared in the faces-config.xml or, as with any JSF application, in multiple configuration files.

Commands and Tools: The <a:toolbar> tag provides a container for the placement of tools and commands to use in the application. A toolbar is specified in only the JSP page, there are no declarations for a toolbar in the faces-config.xml. You need to associate a toolbar with a map through the controlID attribute and you can optionally set the activeTool.

A command is an element on a JSP page that triggers a server side action without any further interaction on the client. The example of a command in the sample application is the “zoom to full extent” button. Once the user clicks the button, a method is called on the server. A tool has further client side interaction before calling a method on the server. An example of a tool in this application is “zoom to rectangle”. Once the user clicks the button, they then drag a rectangle over the map, indicating the area they want to zoom to, and then a method is called on the server. This interaction between map and tool is the reason a toolbar must specify its associated map.

The first two elements on the toolbar are the tools Zoom In and Zoom Out. There are two important attributes on these tags, clientAction and serverAction. The clientAction attribute specifies which javascript function is associated with this tool, and the serverAction associates which class is called on the server when the client action is complete. For your first tool, when the user clicks the zoom in icon, the browser executes a javascript function called MapDragRectangle. After the user drags a retangle over Map0, the form submits to the server where the class *ZoomInToolAction* is called. The execute method will be called on this class and the output from the javascript will based on the methods as a MapEvent argument.

To see which tools are associated with which client side actions, consult the reference information for the tools at this link:

`http://edndoc.esri.com/arcobjects/9.2/Java/api/adfwebcontrols/index.html`

The next tag is an ArcGIS Server Web ADF command tag, which is used for pure server side processing. This example uses a ZoomFullExtentListener, which zooms the map to its full extent. The command tag is an extension of a JSF command tag that provides a richer UI within the Web ADF. An actionListener is used because clicking the button calls into a Listener class provided within the Web ADF. When the user clicks on the button for this command the server side action *processAction* is called on the class *ZoomFullExtentListener*.

The final ArcGIS Server Web ADF tag on the page is the map tag. This tag renders a map to the page and its ID is used to identify it throughout the page, such as with the toolbar. You can control certain properties of the map through the tag attributes, such as size, drag box color for tools, border, and which XSL file to use. This map control is bound to the webMap, which is a property of the context name mapContext using the value binding expression #{mapContext.webMap} for the value attribute. Therefore, this map will render any resources with a property for a map functionality and is also associated with mapContext.

Significant JSF/JSP Tags: The following are snippets of the significant JSF tags in the map.jsp template:

- This is the taglib associated with the ESRI JSF tags:

```
<%@taglib uri="http://www.esri.com/adf/web" prefix="a"%>
```

- The JSF context tag anchors the Java Bean object model associated with this Web page (see Figure 7, “Web Map Application - class model”)

```
<a:context value="#{mapContext}" />
```

- The JSF toc tag renders the Map Table-of-Context on the Web page.

```
<a:toc id="toc" mapId="MapO" value="#{mapContext.webToc}"
style="height:250px;width:300px;" styleClass="tocClass" clientPostBack="true"/>
```

- The JSF overview tag renders the Overview Map on the Web page.

```
<a:overview id="overview" mapId="MapO" value="#{mapContext.webOverview}"
width="300" height="300" clientPostBack="true"/>
```

- Example 1 shows a set of tags to define the tool bar shown above the map on the Web page (see Figure 9). The tool bar contains the tools: zoom-in, zoom-out, and panning the map. Tools are defined as those commands that require User interaction with the map supported by various javascript functions. The tool bar also contains commands: Zoom To Full Extent, Previous Extent, and Next Extent. Commands do not require User interaction with the map and are routed directly to the Web server for execution.

Example 1 Defining a tool bar

```
<a:toolbar id="Toolbar" mapId="MapO" activeTool="ZoomIn">

<a:tool id="ZoomIn" defaultImage="images/zoomin.gif" hoverImage="images/zoominU.gif"
   selectedImage="images/zoominD.gif" clientAction="EsriMapRectangle"
   serverAction="com.esri.adf.web.faces.event.ZoomInToolAction" clientPostBack="true"/>

<a:tool id="ZoomOut" defaultImage="images/zoomout.gif" hoverImage="images/zoomoutU.gif"
   selectedImage="images/zoomoutD.gif" clientAction="EsriMapRectangle"
   serverAction="com.esri.adf.web.faces.event.ZoomOutToolAction" clientPostBack="true"/>

<a:tool id="pan" defaultImage="images/pan.gif" hoverImage="images/panU.gif"
   selectedImage="images/panD.gif" clientAction="EsriMapPan"
   serverAction="com.esri.adf.web.faces.event.PanToolAction" clientPostBack="true"/>

<a:command id="fullext" defaultImage="images/fullext.gif" hoverImage="images/fullextU.gif"
selectedImage="images/fullextD.gif" clientPostBack="true">
   <f:actionListener type="com.esri.adf.web.faces.event.ZoomFullExtentListener"/>
</a:command>

<a:command id="PrevExt" defaultImage="images/back.gif" hoverImage="images/backU.gif"
selectedImage="images/backD.gif" disabledImage="images/back.gif"
action="#{mapContext.attributes['history'].doPrevious}"
disabled="#{!mapContext.attributes['history'].canUndo}" clientPostBack="true"/>

<a:command id="NextExt" defaultImage="images/forward.gif" hoverImage="images/forwardU.gif"
selectedImage="images/forwardD.gif" disabledImage="images/forward.gif"
action="#{mapContext.attributes['history'].doNext}"
disabled="#{!mapContext.attributes['history'].canRedo}" clientPostBack="true"/>

</a:toolbar>
```

- The JSF map tag renders the main map on the Web page. The ID attribute enables other tags to associate themselves with this map. The value refers to the Webmap attribute of the Web Context which defines the map to render on the page.

```
<a:map id="Map0" value="#{mapContext.webMap}" width="600" height="600"  />
```

Javascript

The following Javascript libraries are added to the project and provide the client side action associated with the tools described above. In this example the Javascript functions referenced are part of the esri_map.js library such as EsriMapRectangle.

- esri_colorchooser.js
- esri_core.js
- esri.graphics_ns.js
- esri_graphics.vml.js
- esri_map.js
- esri_navigator.js
- esri_overview.js
- esri_scalebar.js
- esri_slider.js
- esri_task_editing.js
- esri_task_gp.js
- esri_task.js
- esr_toc.js
- esri_toolbar.js
- esri_upload.js
- esri_window_mgr.js
- esri_window.js
- Mapview.js
- Taskbox.js

Faces-Config.XML description

The definition of the Faces-Config.xml file that is populated during project creation is listed in “Appendix D. Faces-Config.XML” on page 36. A number of the options selected during project creation are stored in this XML file. This XML file also defines the significant Java Bean classes in the Java ADF (defined in Figure 7, “Web Map Application - class model”) and how they relate to one another.

Context-attributes.xml description

The text in “Appendix E. Context-attributes.xml” on page 41 defines the context-attributes.xml file that extends the faces-config.xml file. This file contains the managed beans and their associate properties that are related to the Web Context (see Figure 7, “Web Map Application - class model”),

View Web map with GeoRSS Feed (9.2) use case

The objective of this use case is to demonstrate a simple application that will render a base set of spatial content in a Web browser with the extended capability to view dynamic GeoRSS feeds in context with the base content. In the image below the base map is the same map

used in the previous use case. The “blue stars” or marker symbols are created from the GeoRSS feed that identifies recent earthquakes obtained from:

`http://earthquake.usgs.gov/eqcenter/recenteqsww/catalogs/eqs1day-M2.5.xml`

Each earthquake event has a latitude and longitude attribute which is used to identify the point where each marker symbol should be placed. This custom capability is integrated into the Java ADF model defined in Figure 7, “Web Map Application - class model” on page 17 using the Java ADF Task framework.

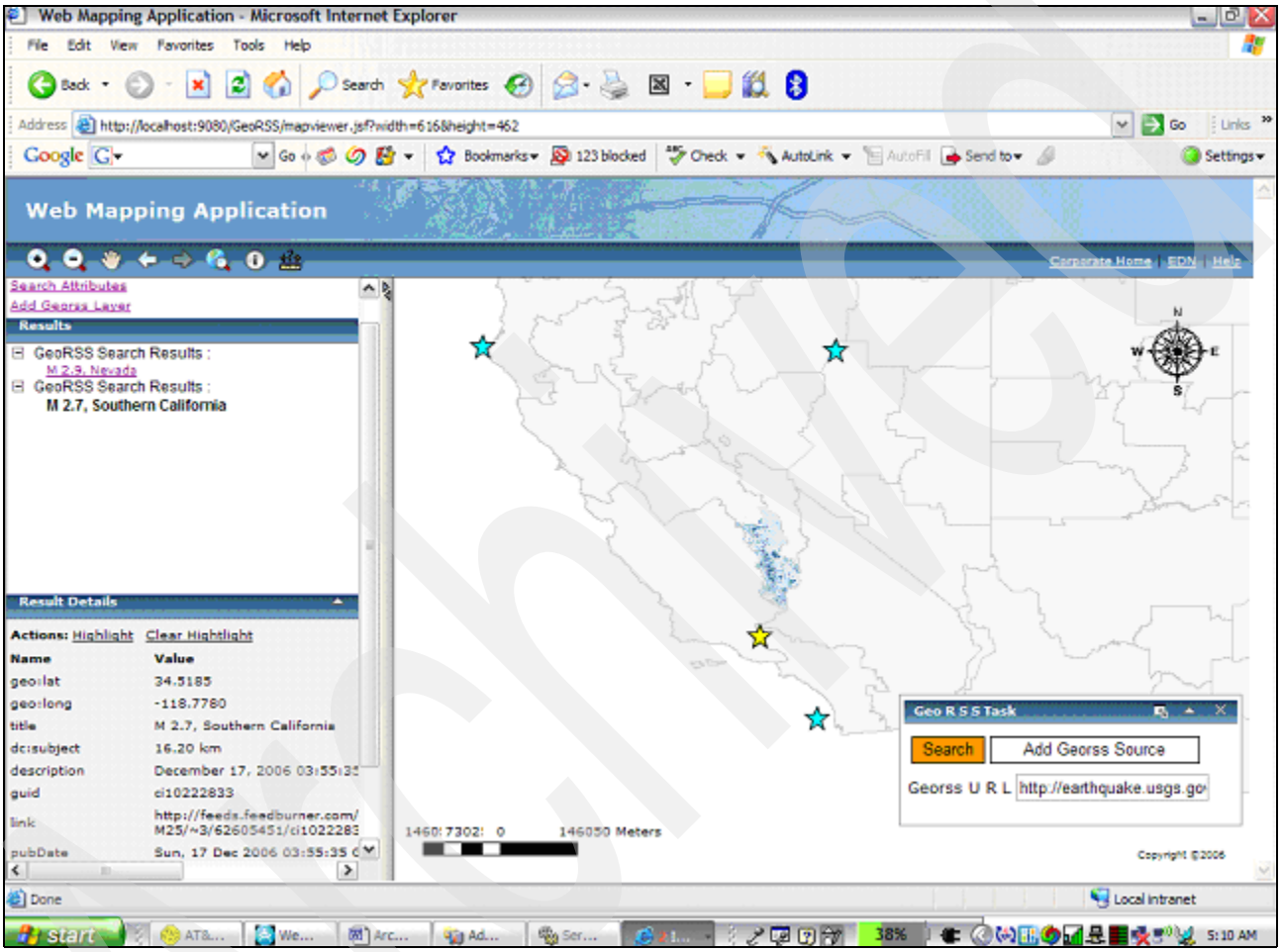

Figure 11 Web map view with GeoRSS Feed

Creating a Project

Steps for creating a GIS Servlet are the same as the previous use case, except select the ArcGIS Web Project instead of ArcGIS Web Samples:

1. New project →Project Wizard →ESRI Templates →Server →ArcGIS Web Project
2. Next →
3. Assign project name.
4. Next →
5. ArgGIS Web Project form is displayed with multiple tabs:
   - ArcGIS Sever Local tab, enter the following and test the connection
     - GIS Server
     - Username

    - Password
    - Domain
  - ArcGIS Server Internet tab
    - N/A
  - ArcIMS Local tab
    - N/A
  - ArcIMS Internet tab
    - N/A
  - ArcWeb Services tab
    - N/A
  - WMS tab
    - N/A
6. At this point you can connect to the ArcGIS Server entered above and select the map service that will be referenced by this application.
7. Press Finish and the Web project will be created as in the last use case. This map (mapviewer.jsp) page differs from the last use case in that it includes the Task framework.

Java ADF Task Framework Overview

Developing a customized solution using the Java ADF Task framework provides the following advantages:

- Integration with the Java ADF
- Event handling
- Automatic User Interface Generation
- Follows standard Java Bean technology
- AJAX enabled

A Task is implemented by a set of customized JSF managed Java Beans that are integrated with the Web page using JSF tags. There are three types of Tasks which are categorized by the input they take:

Parameters Tasks that are represented by HTML input boxes or select boxes and take textual input. Example: selecting a layer

Commands Tasks that are represented by HTML buttons and take a TaskEvent input object. Example: Zoom to Full Extent.

Tools Tasks that require interaction with a map and take a MapEvent input object. These tasks combine an HTML button and interacting with the map like dragging a rectangle with the mouse. Example: Zoom In to the rectangular extent drawn by the mouse.

Here are the steps for creating a custom task:

1. Define the objective of the Task
2. Define the Task methods based on the criteria defined above (that is, Parameter based, Command, or Tool) that will be invoked by a User event such as the press of a button.
3. Define the Task properties that will be modified by the User, such as input text, and submitted with the event.
4. Create a Java Bean with these methods and properties.
5. Define the Java Bean in the faces-config.xml file.

6. Define the Task on the JSF page using the “task” JSF tag provided with the Java ADF.
7. Define a hyperlink on the JSF page that will popup the Task window.

The solution defined for this use case uses both Commands and Tools as defined above. The following sections describe this solution at the Java class level and how to integrate it into a Web map solution. The Java code details behind building this solution can be found on the ESRI Web site:

http://edndoc.esri.com/arcobjects/9.2/Java/java/server/web_adf/tasks/writing_custom_tasks.html

GeoRSS Task

The objective of the GeoRSS task is two-fold.

Task Objectives:

1. Allow the User to enter the URL of a GeoRSS feed and integrate that content with the map on the Web page. In this case the integration is in the form of marker symbols added to the page to spatially visualize the location of the information in the feed. The XML provided below is a snippet from the GeoRSS feed which identifies recent earthquakes. You can observe the location information in the geo:lat and geo:long tags. This will yield a simple mashup of two data sources: the reference layers in the geodatabase and the GeoRSS feed.

```
<item>
  <pubDate>Sat, 16 Dec 2006 15:44:20 GMT</pubDate>
  <title>M 3.3, Unimak Island region, Alaska</title>
  <description>December 16, 2006 15:44:20 GMT</description>

<link>http://earthquake.usgs.gov/eqcenter/recenteqsww/Quakes/ak00073492.php</link>
  <geo:lat>53.6255</geo:lat>
  <geo:long>-163.3450</geo:long>
  <dc:subject>3</dc:subject>
  <dc:subject>pastday</dc:subject>
  <dc:subject>1.00 km</dc:subject>
  <guid isPermaLink="false">ak00073492</guid>
 </item>
```

2. Allow the user to search an area (rectangular region) of the map for existing events and view the details of those events found in that area.

Java Bean Methods and Properties: Refer to Figure 12, “Java Bean class diagram of GeoRSS task solution” on page 30 during this discussion. The GeoRSSTask is a Java Bean which consists of two main methods:

- addGeorssSource(TaskEvent event)
- search(MapEvent event)

The addGeorssSource method behaves as a command and takes the URL property value entered by the User, connects to it, reads the XML stream (shown above) and parses it into an org.w3c.dom.Document object. The Document is then parsed further into internal data structures that can be easily used for this task. This information is stored in the Java Bean which has session scope and thus is available for the rest of the User’s browser session.

The TaskEvent that is provided to this method is used to obtain the WebContext object which is used to obtain the WebGraphics object (Java ADF Library; See Web Map Application - class model). A GraphicElement object (Java ADF library) is created for each point on the map representing information from the GeoRSS feed. The Graphic Element is next

associated with a WebPoint (Java ADF Library; spatial location translated to the Web page) and WebSimpleMarkerSymbol (Java ADF Library; defines the symbol visualization). Each Graphic Element object is added to the WebGraphics object. When this method returns the page is updated with the GraphicElements realized as the “blue star” marker symbols on the map.

The “search” method behaves as a Tool since it accepts the MapEvent object as a parameter. This implies that additional information from the map is required to execute this request. When the Search button is pressed a Javascript function is invoked which enables the User to drag open a rectangle on the page. When the user completes dragging the rectangle the “search” Method is invoked with the MapEvent. The coordinates of the rectangle are obtained from the MapEvent object and used to populate a WebExtent (Java ADF Library) object. The content retrieved by the addGeorssRource method is searched for those points that fall inside the WebExtent. A GeorssTaskResults Java Bean is created for each selected point and added to a result list. The WebResults (Java ADF Library) object is obtained from the WebContext and populated with the results list.

The “Search” button is associated to the Javascript function that draws the rectangle via the GeorssTaskInfo Java Bean which extends the SimpleTaskInfo Class defined in the Java ADF. This class contains a TaskToolDescriptor object (Java ADF library) that associates the “search” tool with the EsriMapRectangle Javascript function in esri_map.js. The GeorssTaskInfo Java Bean is defined to the Task using the “taskInfo” attribute on the JSF “task” tag shown below.

The GeorssTaskResults Java Bean and the WebResults object require special attention. The concept behind the GeorssTaskResults object is to hold the details of a search result and provide action methods a User can perform against that result. In this example the action methods are to “highlight” and “clear highlight” the GraphicElement on the map corresponding to that result. The method “addResultsWithActionMap” is called to populate the WebResults object with the search results and to define the customized User actions.

This method takes five parameters:

- Search Results Header
- Result list
- Method name (in GeorssTaskResults Java Bean) to retrieve the result Name
- Method name (in GeorssTaskResults Java Bean) to retrieve the result Details
- Map of actions that can be performed by the User against this search result

  This is simply a Java map consisting of the display name of the action as the key and the method name of the action as the value.

The image in Figure 11, “Web map view with GeoRSS Feed” on page 25 shows the search result list on the left with the “highlight” and “clear highlight” actions followed by the result details. You can also observe the yellow star representing the results of the highlight action.

JSF Configuration: The JSF managed bean definition for GeoRSSTask is defined as follows:

```
<managed-bean>
   <managed-bean-name>addGeorssTask</managed-bean-name>
   <managed-bean-class>demo.GeoRSSTask</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
   <managed-property>
      <property-name>georssURL</property-name>

<value>http://earthquake.usgs.gov/eqcenter/recenteqsww/catalogs/eqs1day-M2.5.xm
l</value>
```

```
    </managed-property>
</managed-bean>
```

This shows that the “addGeorssTask” is implemented by the GeoRSSTask class, should be loaded into session scope, and contains one property. The property is georssURL which is initialized to the USGS earthquake GeoRSS feed URL. This managed bean is referenced by the JSF “task” tag defined below.

Java ADF JSF Task Tag: This task is added to the Web page using the following JSF tag. The “value” attribute identifies the managed bean “addGeorssTask” defined in the faces-config.xml. The taskInfo attribute refers indirectly to GeorssTaskInfo Java Bean that defines the Tool attributes for this task.

```
<a:task mapId="map1" id="addGeorssTask" value="#{addGeorssTask}"
taskInfo="#{addGeorssTask.taskInfo}" style="width:350px;height:100px;"/>
```

The rendering of this tag uses introspection of the GeoRSSTask bean to build the HTML for the User Interface. The addGeorssSource and search methods will each be assigned to a button and the georssURL property will be assigned to an input text field. When the User selects this Task on the Web page the HTML for this Task will be rendered in a small pop-up window. The User will then be able to interact with this Task through this popup window.

Note: The Task framework also enables the developer to modify the visualization of the Task interface. However, this was not part of this use case.

Task Hyperlink: The following link is added to the Web page to enable the user to select this Task and cause a pop-up window to appear so the User may interact with this Task.

```
<a href="javascript:void(0);"
onclick="toggleWindow('win_EsriTaskCell_addGeorssTask');">Add Georss Layer</a>
```

Figure 12 provides some visual context of the Java Beans described in this section with respect to the Task Framework.

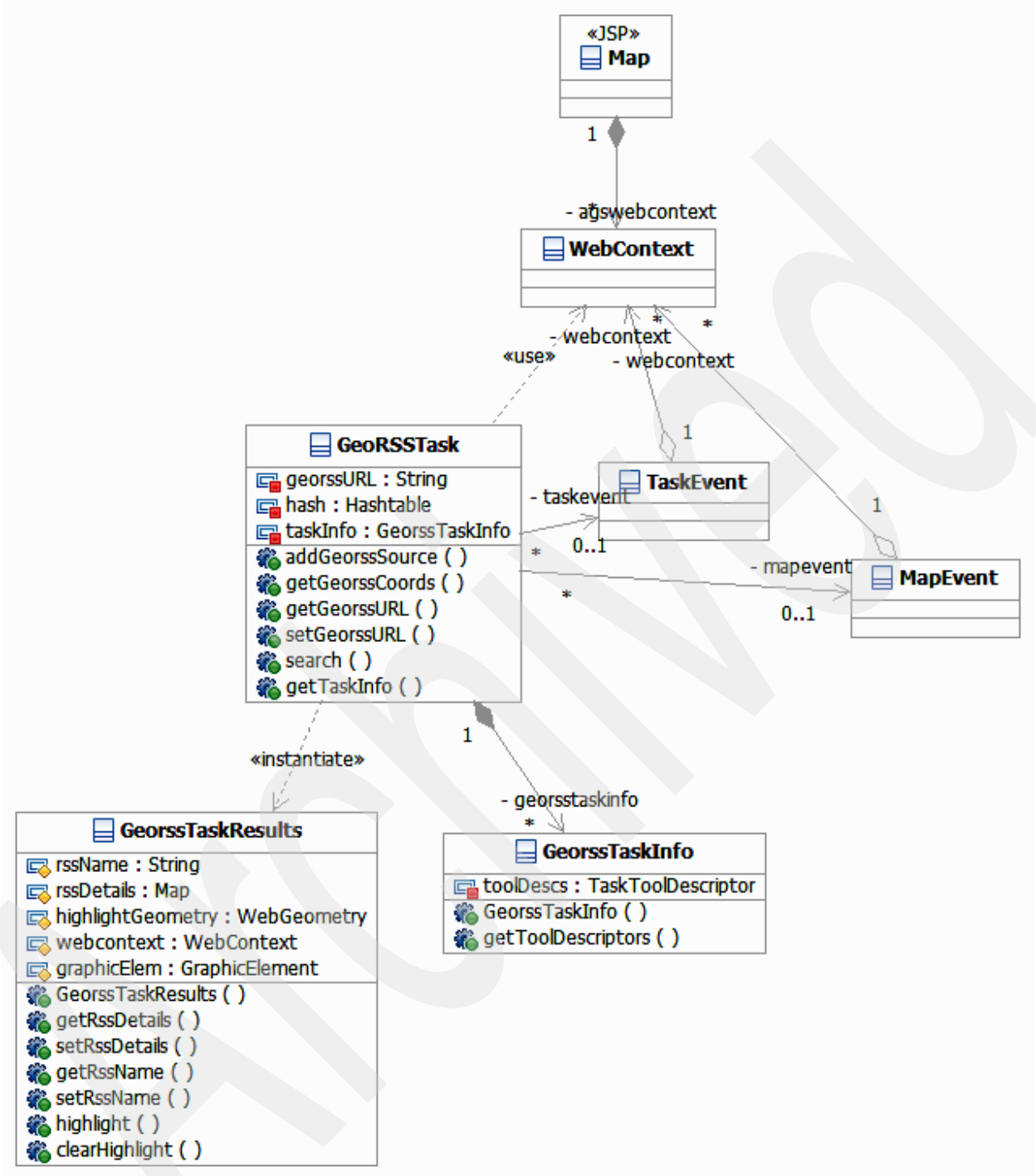

Figure 12 Java Bean class diagram of GeoRSS task solution

Summary

After developing a number of use cases using ArcGIS Server 9.1 and the associated Java ADF, we have been able to obtain a perspective of the ArcGIS Server 9.2 development framework. In general ESRI has made major improvements in the framework which enable

the integration of GIS solutions in a J2EE architecture. Three of these explored by this paper include:

- The adoption of AJAX technology which provides an enhanced user experience where map updates from User interactions update only map image.
- The introduction of the Task framework provides a common mechanism to customize extensions to the Java ADF. This enables quicker turnaround of GIS based solutions.
- Integration of the GeoRSS explored in this paper provides a simple example of how information from various sources can be integrated for value add solutions based on GIS technology.

Most lessons learned using Release 9.1 of ArcGIS Server are from leveraging the ArcGIS Server API. Using this interface requires an in-depth knowledge of ArcObjects, which will be necessary for building solutions that require geoprocessing. However, there have been enhancements in ArcGIS Server Java ADF Release 9.2 that make using this interface in Java easier. We have not explored all of these yet, but we plan to update this paper in the near future with our findings.

Resources

ESRI sites:

- ESRI: esri.com
- ESRI support: support.esri.com
- ESRI / IBM alliance: esri.com/ibm
- ESRI products overview: esri.com/products
- ESRI ArcGIS family: esri.com/software/arcgis

IBM sites:

- IBM/ESRI GIS site: ibm.com/government/esri
- Spatial Offerings: ibm.com/software/data/spatial
  Link to DB2, Rational Spatial Integration Adapter, and others
- WebSphere: ibm.com/software/websphere
- Systems sizing guides:
  www.developer.ibm.com/welcome/eserver/e3/CSFServlet?mvcid=main&packageid=3000
- Systems: ibm.com/servers

The authors of this Redpaper

This Redpaper was produced by a group of experts working with the International Technical Support Organization (ITSO), Poughkeepsie Center.

- Stuart Bedoll is an IBM certified IT Architect leading the Fire Planning Analysis (FPA) project software architecture, design, and technical direction for the AMS Team.
- Rob Culp is an IBM Alliance Manager for the ESRI Strategic Alliance.
- Mike Ebbers is an IBM certified IT Specialist and technical writer in the IBM ITSO.
- Bill Innis is an IBM Technology Manager responsible for technical relations with ESRI.

Thanks to Julio Olimpio, Sudhakar Ramakrishnan, and Eric Bader of ESRI for their technical review comments.

Appendix A. Enterprise Portal Application Software Model (V9.1)

This model represents the ArcGISEnterprisePortalApplication package shown in Figure 4 on page 13. We are using 9.1 in this model because 9.2 was not yet tested in a portal environment.

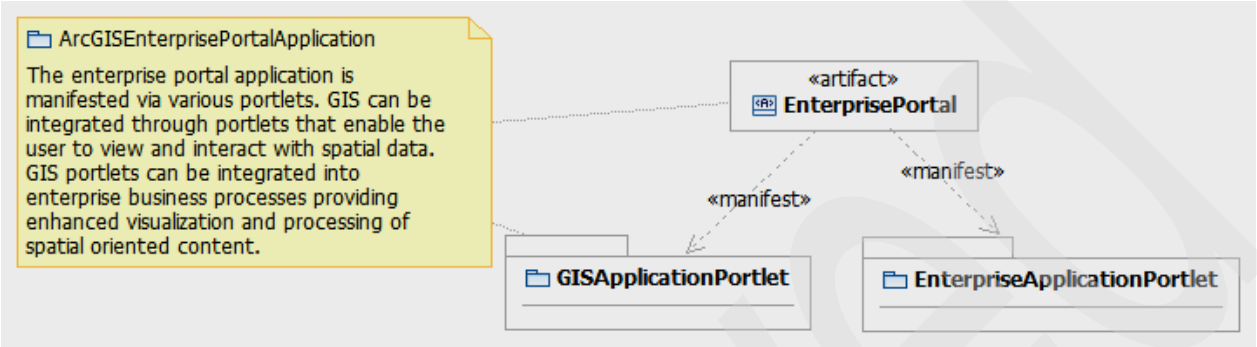

Figure 13 Portal Software Design Model

Enterprise GIS Application Portlet

The following model represents the GISApplicationPortlet package defined in Figure 13, “Portal Software Design Model” on page 32. The objects that start with AGS are objects from the WebControls Java ADF subsystem. See Figure 14.

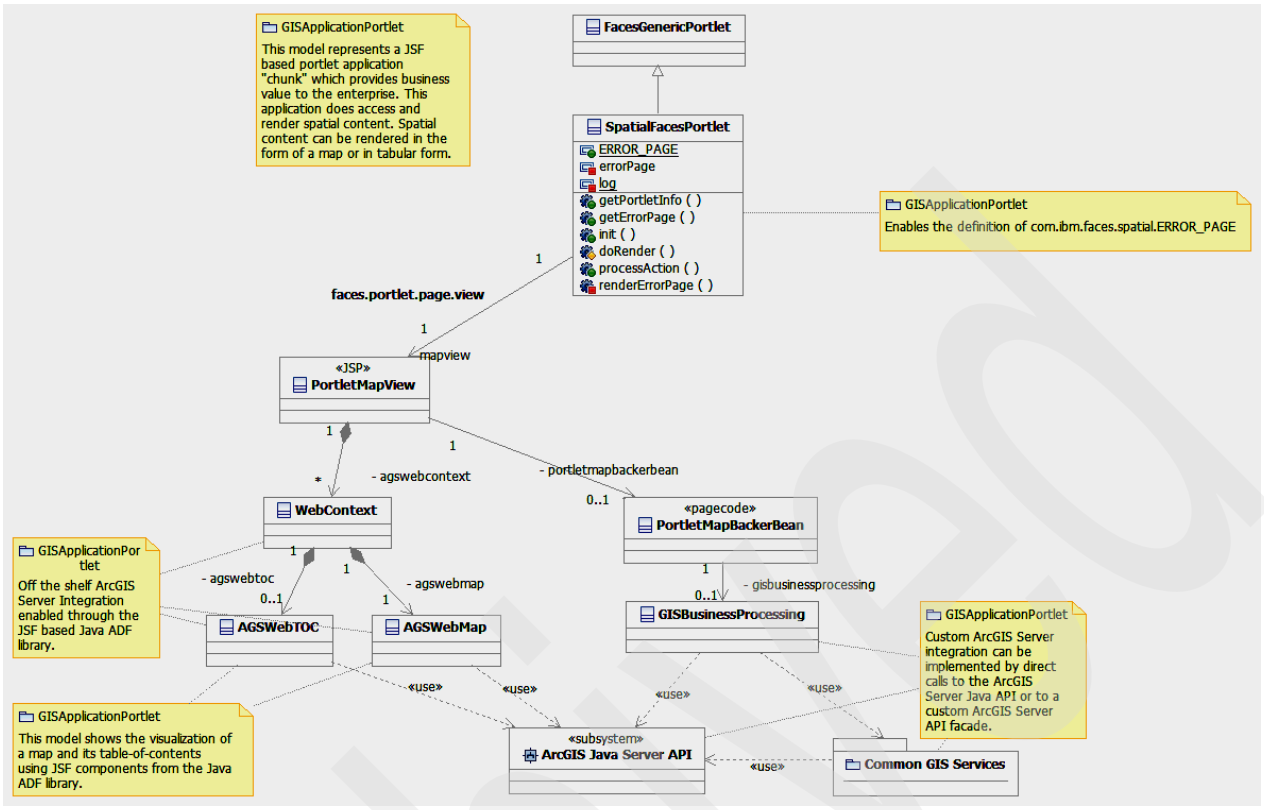

Figure 14 Enterprise GIS Portlet Application

ArcGIS Server API Facade (9.1)

This model is an example of an object model that wraps common ArcGIS Server API methods. Building a Facade of common GIS services will enhance reuse and speed up development. This model is represented by the Common GIS Services package in Figure 7, “Web Map Application - class model” on page 17. See Figure 15.

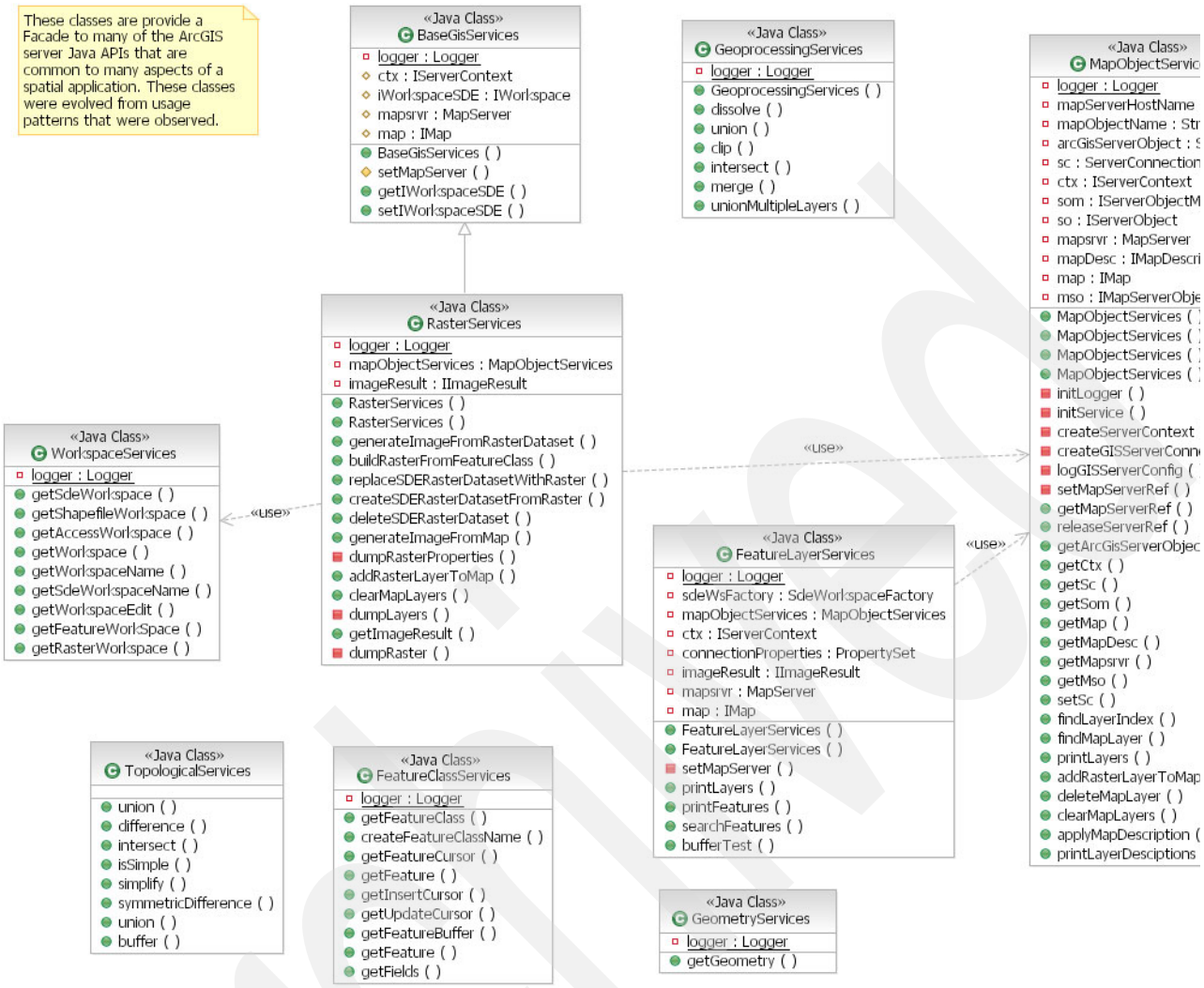

Figure 15 ArcGIS Server API Facade example

Appendix B. Map view on a portlet platform

The objective of the Map View portlet is to integrate the following components to yield an Enterprise GIS Portal Application. This list from Release 9.1 demonstrates a simple GIS based applications which includes a map and table of contents.

- RSA Spatial Integration Adapter (ArcGIS Server Java ADF)
- WebSphere Portal Server and Portlet API (JSR 168)
- ArcGIS Server API
- ArcGIS Map Server Object and Geocode Server Object
- ArcSDE
- Oracle® database server

Creating spatial portlets

1. Steps for creating a GIS Portlet:
2. Create Dynamic Web project
3. Select: Portlet project JSR 168

4. Select Portlet Type: ArcGIS portlet (JSR 168)
5. Select Portlet Modes: View (required), Edit, Help, and Configure (optional)
6. Spatial Integration Feature Options:
   - Default GIS Server
   - Domain
   - User ID
   - Password
   - Logging level
   - Cache Directory
   - XML record set (check box)
   - Make Default Connection Properties button

Note: Context root should start with a ".".

Appendix C. ArcGIS Java ADF Properties (Release 9.1)

ArcGIS Server 9.2 has major architectural changes. Therefore, 9.1 application code cannot be ported directly in 9.2 environments. Much of the 9.1 ADF code must be rewritten, since there are no migration tools. ArcGIS Server 9.1 ADF components cannot co-exist with 9.2. However, this does not mean you can't run applications you've already built. Much of the business logic you used in 9.1 can be converted to tasks and can take advantage of the multi-source ADF in 9.2.

There will soon be information about how you can take advantage of the new and improved functionalities in the core server and ADF, as well as how you can take advantage of the metadata and configure your application. For more information on migrating from 9.1 to 9.2, see:

http://edndoc.esri.com/arcobjects/9.2/Java/java/server/migrating.htm

Overview

Property files provide the capability to tailor each Web application deployment to reference a different ArcGIS server. In general you should be aware of the following:

- The ArcGIS Server Java ADF integration provides the arcgis_webapps and Res property files that are referenced by the JSF tags.
- Res XML file
- ResTemplates properties
- Each JSP that contains the ags:context tag should reference a custom resource bundle with the ArcGIS server location properties rather than hardcoding that information in the JSP.
- For customized ArcGIS Server integration, we recommend that an application-specific property file be used that contains ArcSDE connection properties as well as ArcGIS Server location and credential properties.

arcgis_webapps

This property file is provided by the ArcGIS Java ADF integration. It contains the credentials to access the Windows server where ArcGIS Server is installed. Note: this property file is sensitive to the ArcGIS server that is the target of the Web application. Thus, these properties may have to be tailored with each deployment.

It contains the following attributes:

Domain	Domain name of windows server
User Name	User ID in the associated windows domain
Password	Password associated with this user ID
Logging Level	Level of logging associated with the ArcGIS server connection. This seems to have little effect in 9.1.
Cache Directory	Location of cache directory

Custom Properties

Developing an enterprise application typically requires properties that are tailored for the deployment environment. The following is a snippet of an overlay properties file that can be tailored at the deployment site. It specifies the connection information for the ArcGIS Server and Spatial Data Base engine referenced by the deployed application. The ArcGIS server properties are redundant to arcgis_webapps but were structured as part of the property overlay design.

```
#************* Authentication information *************#
#Arcgis Server Connection Properties
arcgis.auth_domain = <windows domain name>
arcgis.auth_username = <user id>
arcgis.auth_password = <password>
arcgis.encrypted = false
arcgis.server=<host name>
#SDE Connection Properties
sde.user=<database user id>
sde.password=<database password>
sde.database=<database name>
sde.server=<SDE host name>
sde.instance=<SDE tcpip port>
sde.version=SDE.DEFAULT
```

Appendix D. Faces-Config.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
  <managed-bean>
    <managed-bean-name>mapContext</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebContext</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>webSession</property-name>
      <value>#{esriWebSession}</value>
    </managed-property>
    <managed-property>
      <property-name>attributes</property-name>
      <map-entries>
        <map-entry>
          <key>map</key>
          <value>#{map}</value>
        </map-entry>
        <map-entry>
          <key>overview</key>
          <value>#{overview}</value>
        </map-entry>
```

```
        <map-entry>
          <key>toc</key>
          <value>#{toc}</value>
        </map-entry>
        <map-entry>
          <key>graphics</key>
          <value>#{graphics}</value>
        </map-entry>
        <map-entry>
          <key>query</key>
          <value>#{query}</value>
        </map-entry>
        <map-entry>
          <key>history</key>
          <value>#{extentHistory}</value>
        </map-entry>
        <map-entry>
          <key>geocode</key>
          <value>#{geocode}</value>
        </map-entry>
        <map-entry>
          <key>results</key>
          <value>#{results}</value>
        </map-entry>
        <map-entry>
          <key>scaleBar</key>
          <value>#{scaleBar}</value>
        </map-entry>
      </map-entries>
    </managed-property>
    <managed-property>
      <property-name>resources</property-name>
      <map-entries>
        <!-- Resources [START] -->
        <!-- Resources [END] -->
      <map-entry>
<key>ags0</key>
<value>#{ags0}</value>
</map-entry>
</map-entries>
    </managed-property>
  </managed-bean>
  <!-- MapViewer Phase Listener -->
  <lifecycle>

<phase-listener>com.esri.adf.web.templates.MapViewerPhaseListener</phase-listener>
  </lifecycle>
  <!-- Application Beans -->
  <managed-bean>
    <managed-bean-name>mapViewer</managed-bean-name>

<managed-bean-class>com.esri.adf.web.templates.MapViewerBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>context</property-name>
```

```
      <value>#{mapContext}</value>
    </managed-property>
    <managed-property>
      <property-name>resultsToc</property-name>
      <property-class>com.esri.adf.web.data.results.WebResultsToc</property-class>
      <value>#{resultsToc}</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>resultsToc</managed-bean-name>

<managed-bean-class>com.esri.adf.web.data.results.WebResultsToc</managed-bean-clas
s>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>results</property-name>
      <value>#{mapContext.webResults}</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>mapToolsTask</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.tasks.MapToolsTask</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>webContext</property-name>
      <value>#{mapContext}</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>searchAttributesTask</managed-bean-name>

<managed-bean-class>com.esri.adf.web.tasks.SearchAttributesTask</managed-bean-clas
s>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>webContext</property-name>
      <value>#{mapContext}</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>northArrow</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebNorthArrow</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>webContext</property-name>
      <value>#{mapContext}</value>
    </managed-property>
  </managed-bean>
  <!-- Managed bean for ArcGIS Server resource [START] -->
  <!-- Managed bean for ArcGIS Server resource [END] -->
  <!-- Managed bean for ArcGIS Server user [START] -->
  <!-- Managed bean for ArcGIS Server user [END] -->
  <!-- Managed bean for ArcGIS Server Web Service resource [START] -->
  <!-- Managed bean for ArcGIS Server Web Service resource [END] -->
  <!-- Managed bean for ArcGIS Server EJB resource [START] -->
```

```
  <!-- Managed bean for ArcGIS Server EJB resource [END] -->
  <!-- Managed bean for ArcIMS resource [START] -->
  <!-- Managed bean for ArcIMS resource [END] -->
  <!-- Managed bean for WMS resource [START] -->
  <!-- Managed bean for WMS resource [END] -->
  <!-- Managed bean for ArcWeb service resource [START] -->
  <!-- Managed bean for ArcWeb service resource [END] -->
  <referenced-bean>
    <description>
      The ADF webapp always puts this object in application scope when
      the app first starts up.
    </description>
    <referenced-bean-name>esriWebApplication</referenced-bean-name>

<referenced-bean-class>com.esri.adf.web.data.WebApplication</referenced-bean-class
>
  </referenced-bean>
  <managed-bean>
    <managed-bean-name>esriWebSession</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebSession</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>webApplication</property-name>
      <value>#{esriWebApplication}</value>
    </managed-property>
  </managed-bean>
<managed-bean>
<managed-bean-name>ags0</managed-bean-name>
<managed-bean-class>com.esri.adf.web.ags.data.AGSLocalMapResource</managed-bean-cl
ass>
<managed-bean-scope>none</managed-bean-scope>
<managed-property>
<property-name>user</property-name>
<value>#{agsUser1}</value>
</managed-property>
<managed-property>
<property-name>alias</property-name>
<value>FedLand</value>
</managed-property>
<managed-property>
<property-name>serverObjectName</property-name>
<value>FedLand</value>
</managed-property>
<managed-property>
<property-name>hosts</property-name>
<list-entries>
<value>bedoll-t60</value>
</list-entries>
</managed-property>
<managed-property>
<property-name>functionalities</property-name>
<map-entries>
<map-entry>
<key>map</key>
<value>#{agsMap}</value>
```

```
</map-entry>
<map-entry>
<key>toc</key>
<value>#{agsToc}</value>
</map-entry>
<map-entry>
<key>overview</key>
<value>#{agsOverview}</value>
</map-entry>
<map-entry>
<key>query</key>
<value>#{agsQuery}</value>
</map-entry>
<map-entry>
<key>tile</key>
<value>#{agsTile}</value>
</map-entry>
<map-entry>
<key>scalebar</key>
<value>#{agsScaleBar}</value>
</map-entry>
</map-entries>
</managed-property>
</managed-bean>
<managed-bean>
<managed-bean-name>agsUser1</managed-bean-name>
<managed-bean-class>com.esri.adf.web.ags.data.AGSUser</managed-bean-class>
<managed-bean-scope>none</managed-bean-scope>
<managed-property>
<property-name>domain</property-name>
<value>bedoll-t60</value>
</managed-property>
<managed-property>
<property-name>userName</property-name>
<value>arcgismanager</value>
</managed-property>
<managed-property>
<property-name>passwordEncrypted</property-name>
<value>true</value>
</managed-property>
<managed-property>
<property-name>password</property-name>
<value>jHbuAZJz1Jx2zCMarcamow==</value>
</managed-property>
</managed-bean>
<managed-bean>
   <managed-bean-name>addGeorssTask</managed-bean-name>
   <managed-bean-class>demo.GeoRSSTask</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
   <managed-property>
      <property-name>georssURL</property-name>

<value>http://earthquake.usgs.gov/eqcenter/recenteqsww/catalogs/eqs1day-M2.5.xml</
value>
   </managed-property>
```

```
</managed-bean>
  <!-- Tasks [START] -->
  <!-- Tasks [END] -->
  </faces-config>
```

Appendix E. Context-attributes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces
Config 1.0//EN" "http://java.sun.com/dtd/web-facesconfig_1_0.dtd">
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
  <managed-bean>
    <managed-bean-name>map</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebMap</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
    <managed-property>
      <property-name>imageFormat</property-name>
      <value>PNG</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>toc</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebToc</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
    <managed-property>
      <property-name>expandLevel</property-name>
      <value>1</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>overview</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebOverview</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>geocode</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebGeocode</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>query</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.query.WebQuery</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>graphics</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebGraphics</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>results</managed-bean-name>

<managed-bean-class>com.esri.adf.web.data.results.WebResults</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
```

```
  </managed-bean>
  <managed-bean>
    <managed-bean-name>extentHistory</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.ExtentHistory</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>scaleBar</managed-bean-name>
    <managed-bean-class>com.esri.adf.web.data.WebScaleBar</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
<!--
  <managed-bean>
    <managed-bean-name>myAttribute</managed-bean-name>
    <managed-bean-class>com.yourcompany.MyAttribute</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
    <managed-property>
      <property-name>myProperty</property-name>
      <value>myValue</value>
    </managed-property>
  </managed-bean>
-->
</faces-config>
```

Appendix F. ESRI spatial support offerings

ArcGIS framework

The ArcGIS product line by ESRI provides a scalable, comprehensive GIS platform to meet spatial requirements, as illustrated in Figure 16.

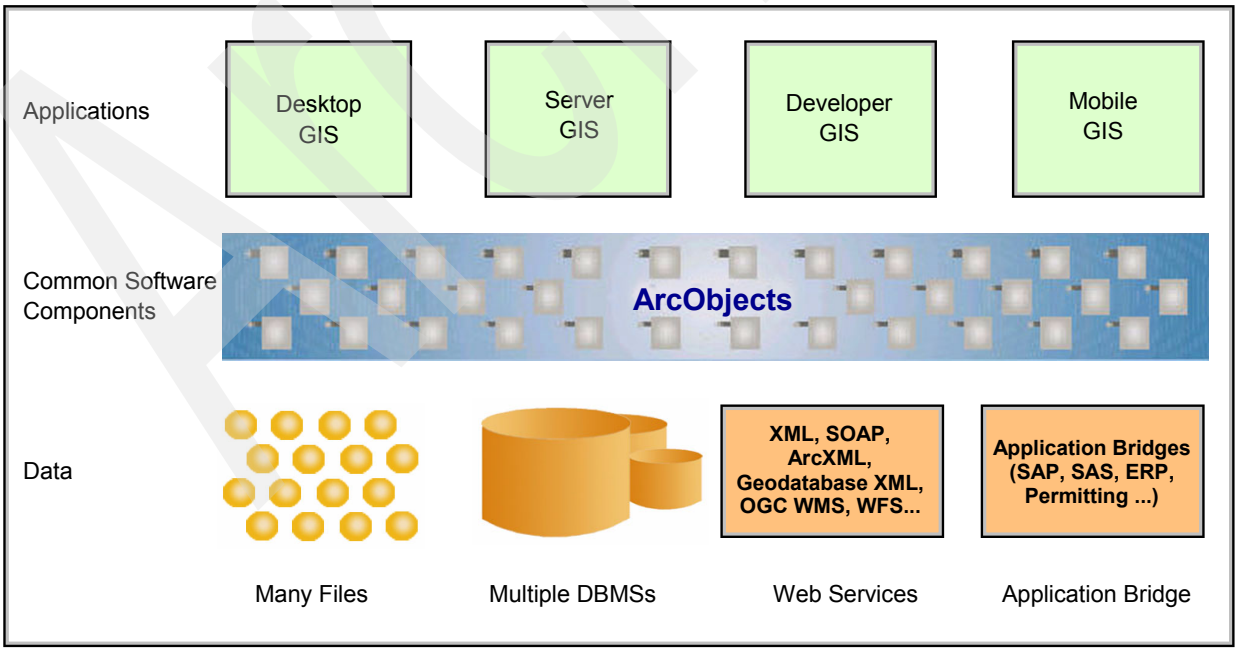

Figure 16 ArcGIS framework

ArcGIS provides a scalable framework for implementing GIS for a single user or many users on desktops, in servers, over the Web, and in the field. ArcGIS is an integrated family of GIS software products for building a complete GIS. It consists of four primary frameworks for deploying GIS:

ArcGIS Desktop	An integrated suite of professional GIS applications comprised of three main software products: ArcView, ArcExplore, ArcEditor, and ArcInfo.
Server GIS	ArcGIS Server, and ArcGIS Image Server
Mobile GIS	ArcPad and ArcGIS Mobile for field computing
Developer GIS	Embeddable software components for developers to extend GIS desktops, build custom GIS applications, add custom GIS services and Web applications, and create mobile solutions.

For more information, see http://www.esri.com/products.html

Appendix G. IBM spatial support subsystems

IBM spatial offerings provide the basis for building robust spatial applications using international and industry standard interfaces. The similar spatial capabilities in IBM's flagship databases (DB2 and Informix) allow users to exploit their database of choice. See:

http://www-306.ibm.com/software/data/spatial/

DB2 Spatial Extender and Informix Spatial DataBlade

The DB2 Spatial Extender and the Informix Spatial DataBlade allows the respective database to manage spatial information using international standard SQL interfaces.

DB2 Geodetic Extender and Informix Geodetic DataBlade

The DB2 Geodetic Extender and the Informix Geodetic DataBlade allows the respective database to manage geospatial information referenced by latitude and longitude coordinates and is well-suited to global data sets and applications.

Release 9.2 installation dependencies

This section explains installation dependencies for Rational IDE and other installation issues as well as for the Special Integration Adapter.

Rational IDE

The ArcGIS Server Java ADF Release 9.2 comes with three Eclipse plug-ins. The ArcGIS Server plug-in has dependencies on Eclipse 3.1. Thus, the Rational Software Architect (RSA) V7 or Rational Application Developer (RAD) V7 are required to install the Eclipse plug-ins. RSA and RAD 6.0.1 are based on Eclipse 3.0.2.2 which does not meet the dependencies of the Release 9.2 software.

The plug-ins are installed at a local eclipse site in the following directory:

```
C:\Program Files\ArcGIS\java\tools\eclipse_plugin\arcgis_update_site
```

The plug-ins are located in the following three subdirectories, each offering a site.xml file;

- core
- server
- engine

Installation issues

When creating a project from the ArcGIS Server templates or samples there will be errors in the XSL files located in WEB-INF/classes. The solution is to move the “taskId” variable definition from the task.xsl to the core.xsl file. This should resolve missing dependencies and avoid circular references.

```
<xsl:variable name="taskId"><xsl:value-of select="//id"/></xsl:variable>
```

Spatial Integration Adapter

The RSA Spatial Integration Adapter plug-in should be uninstalled prior to installing the 9.2 plug-ins.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document created or updated on February 21, 2007.

Send us your comments in one of the following ways:
- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

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

AIX®	IBM®	System x™
CICS®	IMS™	System z™
DataBlade™	PartnerWorld®	SOM®
DB2®	Rational®	TotalStorage®
Enterprise Asset Management®	Redbooks (logo) ™	WebSphere®
Informix®	System p™	

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Enterprise JavaBeans, EJB, Java, Javadoc, JavaBeans, JavaScript, JavaServer, JavaServer Pages, JDBC, JSP, J2EE, J2SE, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.