2003

J2EE vs. Microsoft Dot Net: A Qualitative and Quantitative Comparison for Building Enterprises Supporting XML-based Web Services

Raquel V. Clark
University of North Florida

J2EE VS. MICROSOFT DOT NET
A QUALITATIVE AND QUANTITATIVE COMPARISON FOR BUILDING ENTERPRISES SUPPORTING XML-BASED WEB SERVICES

by

Raquel V. Clark

A project submitted to the
Department of Computer and Information Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
April, 2003

The project "J2EE vs. Microsoft Dot Net A Qualitative and Quantitative Comparison for Building Enterprises Supporting XML-Based Web Services" submitted by Raquel V. Clark in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the committee: Date

Signature Deleted

Sanjay Ahuja
Project Director — 5/1/03

Signature Deleted

Judith Solano
Chairperson of the Department — 5/1/03

Signature Deleted

Charles Winton
Graduate Director — 5/1/2003

ACKNOWLEDGMENT

I wish to specially thank my husband Vincent for his admirable support and understanding during the long hours I dedicated to achieving this milestone in my life and career. His selfless commitment truly symbolizes the wind beneath my wings.

CONTENTS

FIGURES

TABLES

ABSTRACT

Increasing speed of networks and worldwide availability has made the World Wide Web the most significant medium for information exchange. Web technologies have become more and more important as large and small businesses continue to make their presence on the web. Today's businesses have more than just a "face" on the worldwide web. The use of a web browser is no longer restricted to viewing static pages. Browsers are becoming more and more a standard interface to a multifaceted reign of programs that live on the worldwide web. Two main technologies stand out for the implementation of web applications, Sun Microsystems' Java 2 Enterprise Edition (J2EE) and Microsoft' Dot Net Framework. The purpose of this study is to provide an unbiased comparison of the two technologies based on performance and other software qualities.

Chapter 1

INTRODUCTION

Sun Microsystems' J2EE technology has arguably maintained the stance as the leading platform for web based applications. The popularity of J2EE technology led to strong competition among application servers, the software to run J2EE applications, and industry giants like IBM, Oracle and BEA began to compete for the application server market. In early 2001, Sun Microsystems published a J2EE sample application, the Java Pet Store. Since its publication, the java Pet Store became the comparison tool among Application Server vendors. J2EE vendors rushed to reveal their own Pet Store benchmarks and results were used as a marketing tool. In November 2001 Microsoft re implemented the java Pet Store with similar functionality as the J2EE counterpart, and released a benchmark claiming a significant performance gain over the J2EE pet store implementation. Many Java developers, Sun and J2EE vendors like Oracle and IBM® claimed that the benchmark was invalid because the Pet Store implementation was not meant to be a benchmark, but more so a guide to illustrate the J2EE architecture and software development best practices. Many benchmarks have been conducted since then, ultimately all products benchmarked yielded the same results; all benchmarks favored the given vendor conducting the benchmark. Perhaps individual vendors spent time and effort fine tuning their own implementation without even realizing it. Nonetheless, software engineers and software managers must select a technology that represents the most optimal solution to fit their business needs. This study aims to provide software

engineers and managers with an unbiased report containing qualitative experiences of software development with both technologies, as well as a quantitative performance comparison of a distributed web application developed with both technologies. This study also compares the performance and ease of consumption of web services by both technologies.

Chapter 2

JAVA 2 ENTERPRISE EDITION (J2EE)

The J2EE is an industry standard for java technology application development. J2EE is composed of several sub standards that address individual application functionality. The collection of standards is large; among the most pertinent standards for web application development are Java Beans, Enterprise Java Beans (EJB), Servlets, Java Database Connectivity (JDBC), XML technologies and Java Server Pages (JSP). It is important to emphasize that J2EE is not a product, but rather a collection of standards for application development that defines a contract between applications and its container.

2.1 Architecture Overview

J2EE Architectures involve at least three major tiers, although some introduce an additional division in the middle tier. The Enterprise Information System (EIS) Tier consists of resources that J2EE applications must access. These include Database Management Systems (DBMS), and legacy mainframe applications. The Middle Tier contains the application business objects. Business objects are those that implement business rules and functionality. The middle tier is also in charge of mediating access to EIS tier resources. The User Interface (UI) Tier, also known as the Web Tier, exposes the middle-tier business objects to the users. In web applications the UI tier is made up of servlets, helper classes used by servlets, and view components such as Java Server Pages. It is important to emphasize that the client of a web application is a web browser.

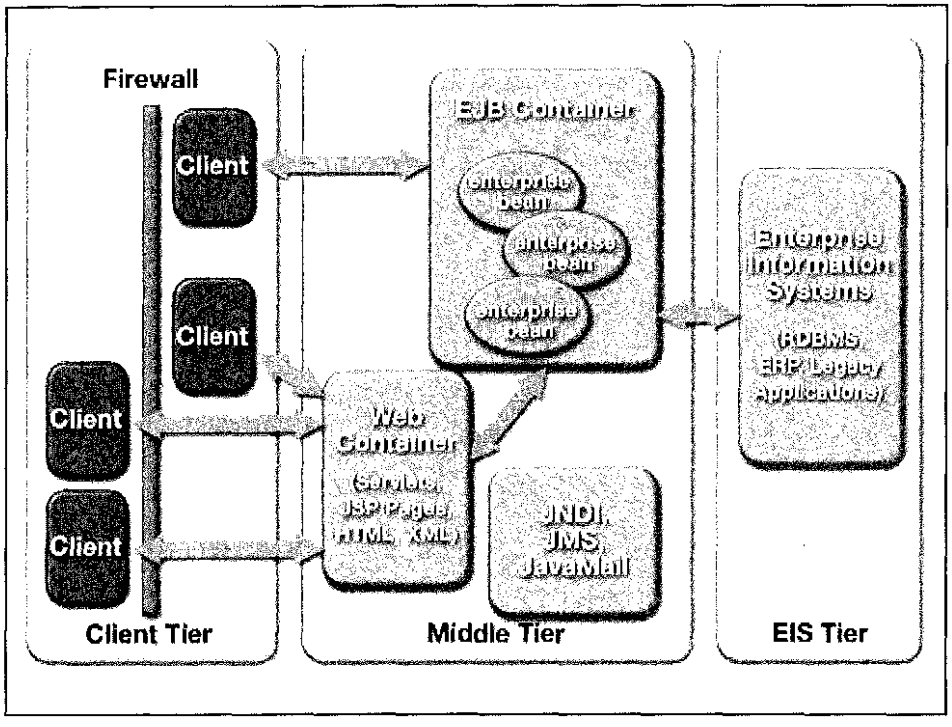


Figure 1: J2EE Environment

2.1.1 Non-Distributed Architecture

Non-distributed web applications run in a single Java Virtual Machine (JVM). Since the middle tier and the web tier run in the same JVM, they must be logically separated. J2EE best practices recommend creating an interface layer between web tier objects and the middle tier objects. Separation added by interfaces contributes to clarify object responsibility. This architecture exhibits the best performance, and it is the most widely used for simple and even complex web applications.

2.1.2 Distributed Architecture with EJBs

By definition a distributed application is one that resides in multiple machines. This architecture has the ability to split the middle tier physically and logically, into separate

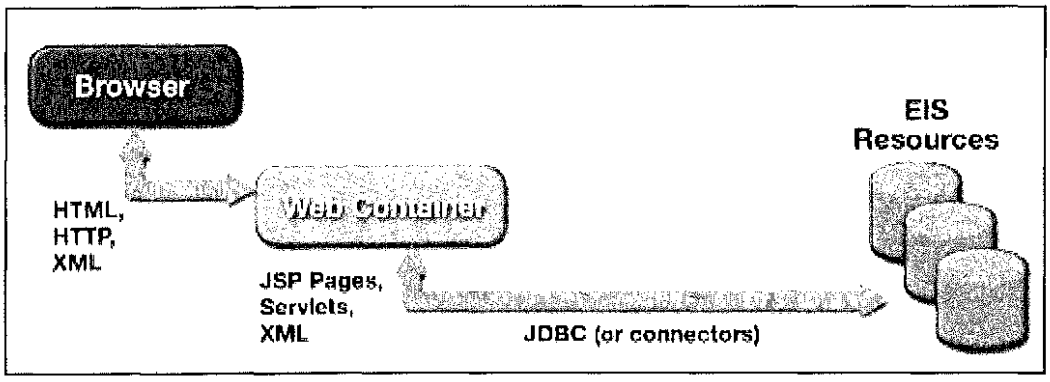


Figure 2: Non Distributed Web Application

components that are able to run in multiple JVMs, and thus in multiple machines. This architecture uses Remote Method Invocation (RMI), between web tier objects and middle tier objects, which are exposed as Enterprise Java Beans (EJBs). The details of RMI communication are concealed by the EJB container. The EJB container is one of the components of an Application Server. RMI involves object serialization and there is a significant overhead added by the RMI communications layer. Applications designed with this type of architecture must attempt to reduce the number of remote calls since remote calls add considerable overhead and reduce performance. Chatty network communication between the web tier objects and business tier objects can lead to performance bottlenecks. This architecture offers a clean separation between web and business tier objects since communication with EJBs is achieved by the use of interfaces.

2.1.3 Distributed Application with Web Services

The emergence of web services standards such as Simple Object Access Protocol (SOAP) allows J2EE applications not to be bound to strictly RMI connections in support of distributed clients. Furthermore, J2EE components are able to exchange

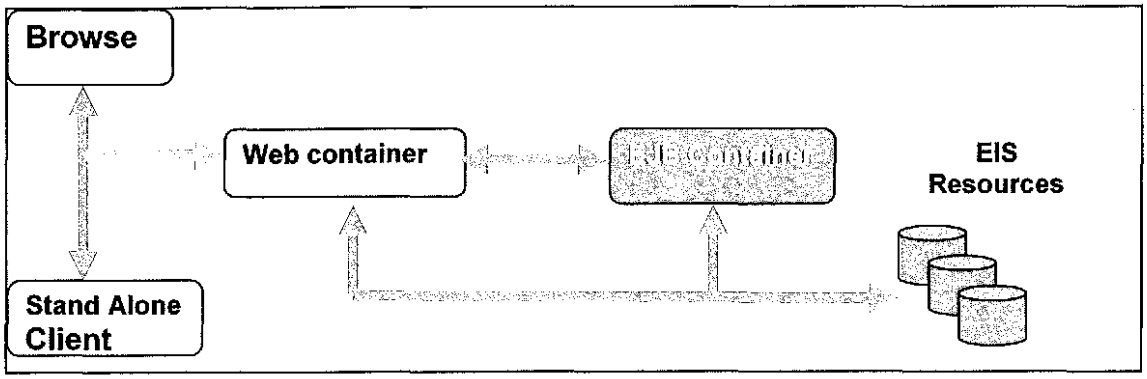


Figure 3: Distributed Architecture with EJBs

information with non-J2EE components, such as Microsoft components and applications. This architecture adds an object layer exposing the web services, and a transport layer implementing the necessary protocols. The object layer may simply be the application's business interfaces. Currently, J2EE does not have a standard for web services. Therefore, the structure of the transport layer will vary depending on the implementation. Third-party products such as Apache Axis provide easy SOAP web services support on any J2EE server. This architecture differs from the distributed EJB architecture not only in the remote protocol used, but also in that the remote interfaces are typically added onto an existing application, rather than build into the structure of the application. The overhead of passing objects over an XML-based protocol such as SOAP is likely to be higher than that of RMI, because XML documents needs to be parsed.

2.2 Model View Controller Architectural Pattern

The Model View Controller architectural pattern, or MVC, has its origins in Smalltalk user interfaces, and has become one of the most successful and widely used Object Oriented patterns. MVC divides components needing to support a user interface into

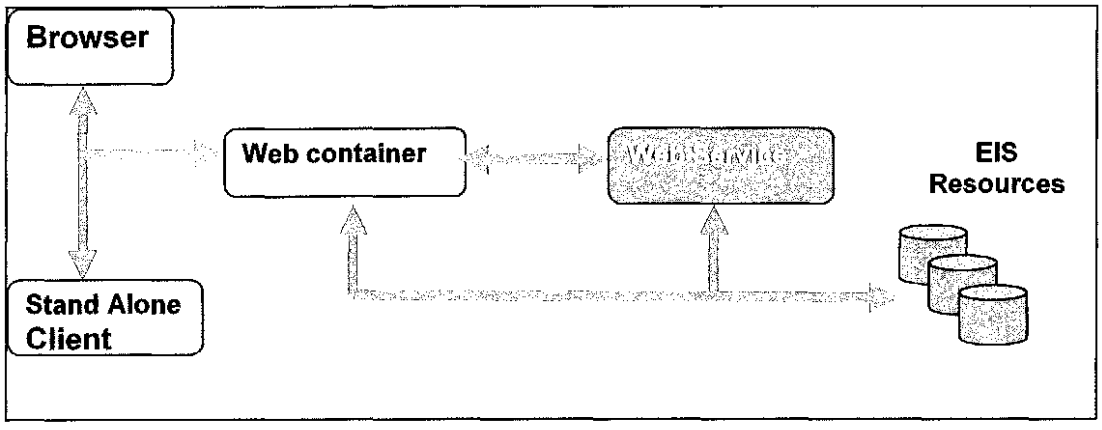


Figure 4: Distributed Architecture with Web Services

three types of objects, ensuring a clean separation of responsibility. First is the model. The model is responsible for containing data; it has no presentation specific details. Second is the view. It is responsible for displaying the model data. Third is the controller. It reacts to user inputs and updates the model accordingly.

There are currently two approaches to implementing the MVC architectural pattern. MVC model 1 exhibits a higher coupling between the view and the controller. Views may very well act as controllers and may include forwarding logic to other views. MVC model 2 is the most recommended approach to application development. In model 2, the views do not contain any controlling logic and all requests are funneled to a front controller.

A typical implementation of the MVC model 2 architectural pattern in J2EE constitutes a standard controller servlet as a single point of entry into the entire web application. The controller chooses one of multiple application specific sub controllers to handle each request. The controller servlet is in essence a controller of controllers. It produces no output itself, but processes requests, initiates business operations, manipulates session

and application state, and redirects requests to the appropriate view, where the actual data from the model is rendered to the user. The MVC model 2 pattern objects map to J2EE objects as follows: a model is a Java Bean that exposes its data through data accessor methods. Model objects should not be able to retrieve data from business objects, but rather they should expose bean properties to be modified by a controller. The benefit of this approach is that the rendering of a view should not fail because of errors in retrieving its data. This greatly simplifies error handling in the user interface tier, since the controller is aware of the success or failure of data access and can choose to display the appropriate view. XML documents may also act as model objects. XML documents have a defined structure and are capable of storing elements and attributes. A view is a component that is used to display the data in a model. A view should never perform business logic or obtain data other that what is available in a model. The most common type of view in J2EE is Java Server Pages.

A controller is a Java class that handles incoming requests, interacts with business objects, builds models, and forwards requests to the designated view. The controller is not concerned with business logic and should not implement any business rules. The controller's job is simply to coordinate the processing of a given request. The front controller is a special type of controller. Front controllers in J2EE are servlets. The front controller is responsible of the initial processing of a request. It simply determines which requests controller is the one designed to handle such request and forwards the request to the appropriate controller.

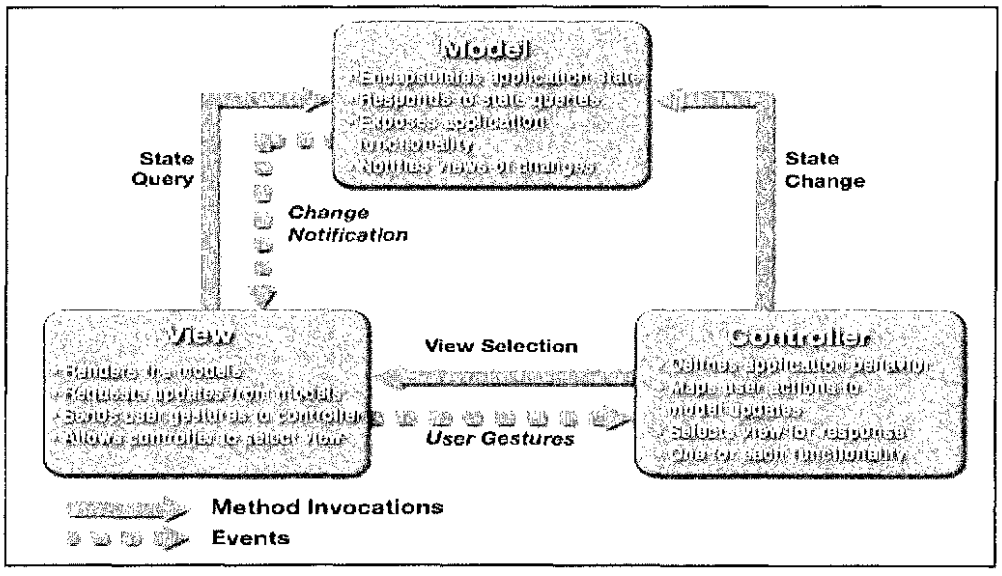


Figure 5: MVC model 2

Chapter 3

MICROSOFT .NET

The Microsoft .NET strategy was implemented in response to the decreasing cost of communications, the increase in client processing capabilities, and the increased use of the Internet as medium to run software applications. Microsoft .NET is a comprehensive platform that provides prefabricated solutions to common programming problems. At the heart of .NET is the .NET Framework. The .NET Framework consists of four core components, the Common Language Runtime or CLR, the .NET Class Libraries, Windows Forms, and Active Server Pages ASP.NET. These components are the foundations on which the .NET strategy is implemented. Unlike J2EE, .NET offers multi-language support, although it is bound to the Windows operating system. Application code written in any of the supported languages is translated into an intermediate language called IL. IL is Just In Time, (JIT) compiled before execution. IL is not interpreted, and is not byte code. IL is a language and its code is compiled and converted into machine code.

3.1 Architecture Overview

Although it is possible to build non-distributed applications with the .Net Framework the .NET general architectural approach is a distributed one. .NET offers a Service-Based Application Architecture where XML web services are categorized as distributed

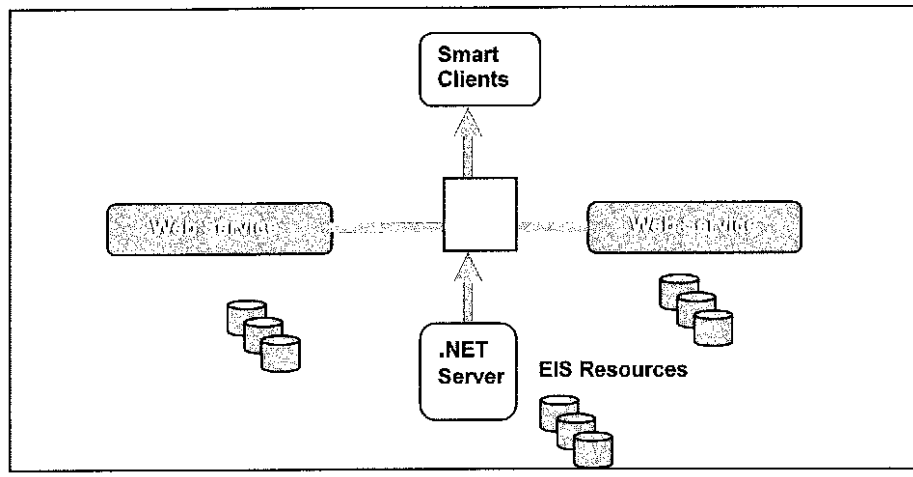


Figure 6: .NET Architecture

reusable components. Web services are remote business objects that can be invoked with the use of the Simple Access Object Protocol (SOAP). Web applications may contain references to multiple web services deployed in remote machines. This architecture is similar to the J2EE distributed application with web services previously introduced.

3.2 The .NET Framework

The Common Language Runtime or CLR is perched on top of the Windows operating system. The CLR handles language integration into IL, and it is responsible for run time conversion of the IL into native code. The .NET Class Libraries are object-oriented and hierarchically structured. The class libraries are a growing effort just as the Java class libraries. Windows Forms are s set of classes within the base classes that enable building smart clients or Windows-based desktop applications. Windows forms have XML Web services support and can access remote web services components. ASP.NET gives a programming model that provides high-level components and services aimed specifically at creating XML Web services and web applications. ASP.Net has its own HTTP

runtime, which is not the same as the CLR. This runtime is multithreaded and asynchronously processes all incoming HTTP requests.

3.3 Building Web Applications with ASP.NET

The model view controller architectural pattern in not clearly employed in building .NET web applications. ASP.NET provides the presentation and event handling of web applications. The view is represented by the aspx page. An aspx page is a mixture of HTML and a set of controls including HTML controls and Server controls. Each aspx page is bound to a class known as the codebehind, representing the model. HTML tags in the aspx page are controlled from the codebehind class. This eliminates the need for a scripting language since all event based code is handled by the codebehind class which is written in any of the languages supported by the .NET Framework. Each ASP.NET page is designed to post back to itself; therefore, the implementation of a Front Controller is not as clear. There is no analogous object to a J2EE servlets in ASP.NET. One strategy could be to write a class to provide flow control of events fired by the aspx pages. The codebehind classes would contact the controller class to request the action or the next view to forward the request to. The .NET architecture provides a clean separation between the model and the view, but the controller seems coupled to the model. ASP.NET resembles more the earlier Model View Controller Model 1.

3.4 Server controls

One of the most significant differences between building web applications with JSP technology and building web applications with ASP.NET technology is something called

server controls. Server controls are of three types, HTML controls, Web controls and Validating controls. HTML server controls in ASP.NET are HTML elements containing attributes that make them visible to methods on the server. HTML elements in a web form are not visible on the server side and require the use of scripting languages for processing. By exposing the HTML elements as HTML server controls in ASP.NET they become visible on the server side and programmers are able to write code to respond to user events without the need to use a scripting language. The object model for the HTML server controls maps to the object model of the HTML elements. HTML attributes are exposed as properties of the HTML server controls. Any HTML element can easily be converted to server controls by adding the attribute runat="server".

Web server controls are custom made HTML controls designed to simplify page processing. There is not a one to one correspondence of web controls to HTML elements. Additionally all web server controls can be bound to a given data source with minimal effort. Like the HTML controls, there is a pre-defined set of web controls available from ASP.NET. Web server controls can represent collections like radio button groups and drop down lists among others. Validation server controls provide a way to check user input in web or HTML server controls. Validation controls replace client side validation usually accomplished with a scripting language. There are six types of validation controls and they simplify the most common field validation. ASP.NET also allows for custom server controls which provide flexibility beyond the implemented code for programmers to replace some of the functionality usually obtained by the use of scripting languages.

Chapter 4

BRIEF COMPARISON OF COMPONENTS

Both technologies aim to develop applications that are container managed. The container manages functionality such as transactions, load balancing, message queuing, legacy system integration and connection pooling among others. It is important to realize that .NET is a product strategy, and J2EE is a set of standards that are implemented by a given vendor. Examples of these include IBM Websphere, BEA WebLogic, Oracle 9i AS, JBoss, JONAS and many more. The following table compares features existing in .NET and J2EE implementations.

Feature	.NET Implementation	J2EE Implementation
Middleware	COM+	RMI/CORBA
Runtime Environment	Common Language Runtime	Java Runtime Environment
Server Side Components	.NET Managed Objects	EJBs and Servlets
Data Access	ADO.NET	JDBC and SQLJ
Web Services standards	SOAP, WSDL, UDDI	SOAP, WSDL, UDDI
Web Services Implementation	.NET Framewok	JAXP, Apache Axis and other 3rd parties.
Dynamic Web content	ASP.NET	JSP, Velocity, Java Server Faces
Legacy Support	Host Integration Server 2000	J2EE Connector Architecture (JCA)
Platform Support	Microsoft Windows	Platform agnostic

Table 1: Feature Comparison of J2EE and .NET

Chapter 5

INTRODUCTION TO WEB SERVICES

Web services, in the general meaning of the term, are services offered via the Web. These services are provided by programs that can be accessed via their remote interfaces. Web services can be considered as web applications that can be used by programs instead of users. The service receives the request, processes it, and returns a response, but the parties involved in this web conversation are two programs with no human intervention. A web service can be ingested by a web enterprise, in which case it simply becomes a remote component of the enterprise.

Web services and consumers of Web services are typically businesses, making Web services predominantly business-to-business (B-to-B) transactions. An enterprise can be the provider of Web services and also the consumer of other Web services.

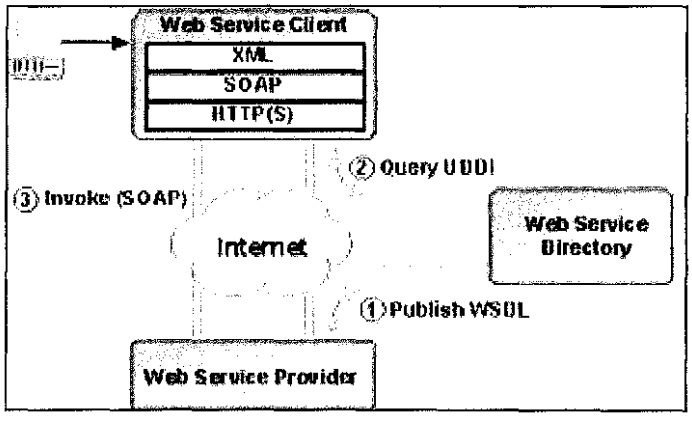


Figure 7: Web Services Process

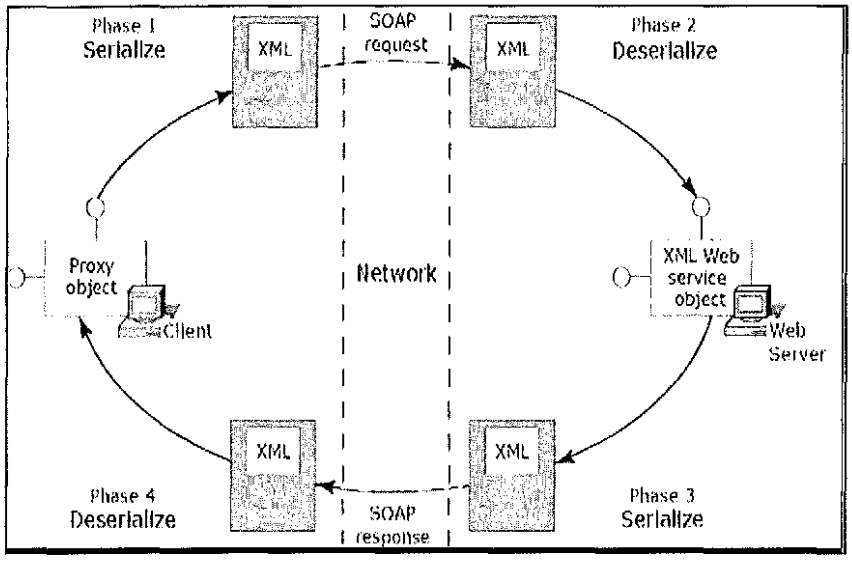


Figure 8: SOAP Message Process

5.1 Web Services Concepts

Simple Object Access Protocol (SOAP) is the protocol web services use to send and receive messages. SOAP is an XML based protocol that wraps requests and response objects into what is called a SOAP envelope. SOAP envelopes are written and read by the publisher and consumer of the web service.Web Services Description Language (WDSL) is a new specification to describe networked XML-based services. It provides a simple way for service providers to describe the basic format of requests to their systems WSDL is a key part of the effort of the Universal Description, Discovery and Integration (UDDI) initiative to provide directories and descriptions of such on-line services for electronic business.

UDDI is designed to provide registries, either public or private, for registering and discovering Web services. Publishing a web service requires registration with a UDDI.

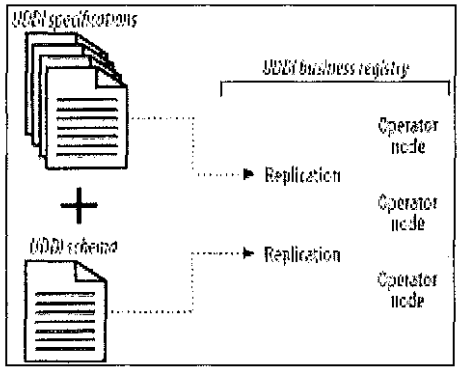


Figure 9: UDDI Registry

5.2 Building Web Services

Building Web Services is no different than building a remote component with previous technologies. First, identify the services the web service will provide. Develop the functionality with whichever technology you have chosen. Write an interface which will represent the methods the web service is going to expose. Development tools will do the rest. Use a development tool to convert the generated interface into a web service. The specific steps to generate the web service will depend on the chosen technology. All products however will at a minimum produce a remote interface and the WSDL to describe the service.

5.3 Consuming Web Services

Consuming web services has become almost automatic. Once a web service has been located the consumer party converts the published WSDL to actual code. Most web service tools in the present market, whether are Java web services tools or .NET tools, perform this conversion automatically. The tools ingest the WSDL and generate a proxy for the web service public interface. Proxies are a similar concept introduced by stubs

and skeletons produced by Common Object Request Brokers (CORBA) and those produced by the Component Object Model (COM). The web service proxy or stub classes are ready to be invoked by a web application business objects to send a receive request and responses to and from a published web service. Since .NET is a single product, there is really only one way to translate the WSDL to .NET code. This is not the case with J2EE. Sun Microsystems has not published a web services standard for the J2EE community, thus the details of proxy code and stub code are left for the implementers. There are several vendors that implement web services with java technology. Among the major vendors are IBM, Oracle, and BEA. Several open source projects have been created in support of web services. Among the most notable are Apache SOAP and Apache Axis, both from the Apache foundation.

Chapter 6

RESEARCH WEB APPLICATION

The web application of this research consists of a weather web application that provides observations and forecasts for a set of selected cities. The cities available are grouped by state. Selection of a given state discloses the selected cities available for the chosen state. In addition to providing observations and forecasts, the application also provides satellite and Doppler radar imagery for sections of the Continental United States. The weather web application is a distributed application that obtains weather data from a web service. The web service in turn obtains its data from a file server owned by the National Weather Service (NWS). The National Weather Service provides an FTP download server where observations and forecasts are available to the public. The web service incorporates an administration screen where downloads are started.

6.1 Web Service Description

The web service was designed to allow retrieval of observations, forecasts, radar images and satellite images. The implementation of the web service was done in Java using Apache Axis. Axis allows web services to be deployed on a servlet engine as a web application. The servlet engine of choice was Tomcat version 4.0. The web service provides an administration screen whereby products can be downloaded from National Weather Service and stored in a Relational Database. The Relational Database Management System (RDMS) of choice was the open source RDBMS MySQL.

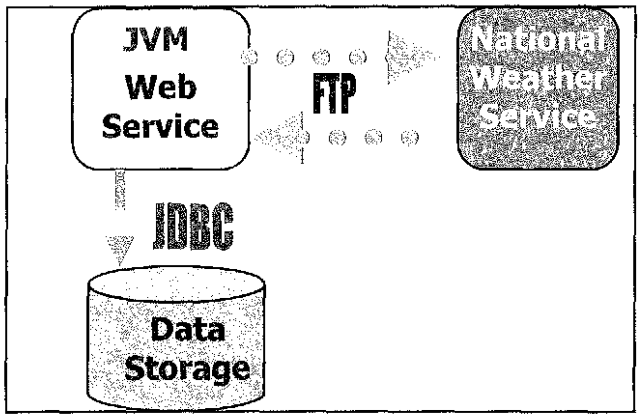


Figure 10: Anatomy of the Weather Web Service

6.2 Web Application Description

The name of the web application is Mega Vega Weather. It was designed to present a user with a welcome weather page which displayed the National Composite Radar imagery. A composite image combines Doppler radar information combined with surface isobaric analysis and weather systems depiction. Mega Vega Weather supports three distinct products, weather observations and forecast for a given city and state, radar images for a given regions, and satellite images for a given region. Mega Vega Weather was designed to support multiple events although only three events were implemented. Mega Vega Weather was written in two different technologies. Both applications posses identical screens with identical functionality. Both applications communicate and obtain data from the Axis weather web service. The Java implementation uses the Tomcat servlet engine while the .NET implementation uses the .NET Framework and ASP.NET.

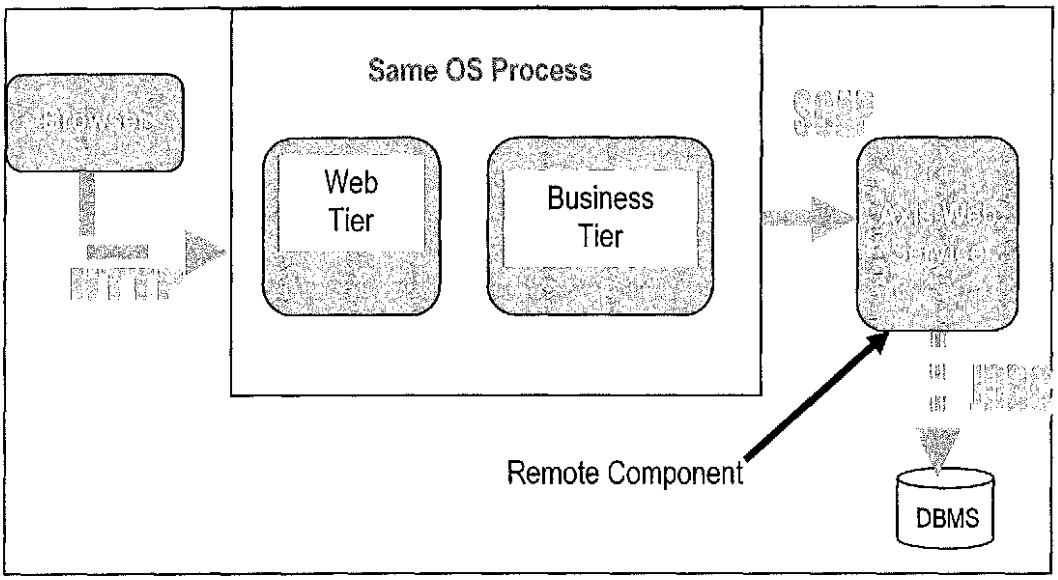


Figure 11: Anatomy of Weather Application

The user interface design was kept consistent from screen to screen, allowing for the selection of any of the three events from any of the screens displayed.

6.3 Web Service Implementation Details

The Weather web service implementation uses Apache Axis SOAP engine. Axis is deployed on Tomcat as a web application. Axis provides a servlet implementation and an Administration client that allows easy deployment of a web service. The web service is composed of four packages, the core package, the exception package, the data access package and the utilities package. See Appendix A for full Web Service Description File.

The weather web service public interface exposes the following methods:

1. getOb: returns a string data type.
2. geForecast: returns a string data type.
3. getWeatherImage: returns a byte array data type.

Upon receiving a request, the web service class implementing the remote interface forwards the request to a request handler object. This handler is the controller of the web service and in turn creates business objects to retrieve the information requested from permanent storage, the MySQL database. Data base access is handled with the use of prepared statements for performance gains. Once the data is successfully retrieved from the database is sent back as part of a SOAP envelope which is returned to the Mega Vega Weather site. The administration component of the web service is a simple command line interface that allows for the retrieval of observations and forecasts for selected cities from the National Weather Service FTP site. The administration component is capable of running in the background and retrieving information every hour on the hour. The retrieval process updates the database by replacing the older observations and forecasts with the latest download. Radar or satellites images are not freely downloadable from National Weather Service, thus no download occurs for such products. Instead, images are manually obtained from other commercial sources. Only the image paths are stored in the database. Upon receiving a request the web service reads the path from the database goes out to the file system and uploads the image into a byte array. The byte array in then sent wrapped in a SOAP envelope back to the Mega Vega Weather site where it is processed and display appropriately.

6.4 J2EE Web Application Implementation Details

The Java web application was written following the Model View Controller Model 2 architectural pattern. A single controller servlet is implemented as the Front Controller which receives the initial requests and forwards the request to the appropriate handler. The handler is obtained by the controller servlet from a helper class called the Redirector. The Redirector class is implemented as a Singleton class that is instantiated upon application startup. Additional helper classes include a Configuration Handler class, which is also implemented as a Singleton class that reads application configuration files and cooperates with the Redirector to initialize the application based on the loaded configuration. Each application event is represented by an event object, which contains the target view JSP page in which it would be displayed. When a request arrives at the servlet controller, the controller calls the Redirector to obtain the appropriate handler for the event. The Redirector returns a handler, which contains a reference to the event object, and the controller servlet forwards control to the event handler. The event handler in turn instantiates business objects which are in charge of connecting, retrieving and processing the information received from the Axis web service. Once the information has been returned by the Axis web service it is processed. The handler creates the appropriate Java Bean representing the model, and sets the information passed from the business objects. The handler then forwards the control to the JSP, the view, which in turn queries the Java Bean and displays the information on the browser.

6.5 .NET Web Application Implementation Details

The .NET web application was implemented with a similar MVC architecture. However, the absence of the Front Controller makes the flow of control less clear than the Java implementation. My solution to the missing front controller was to use the Redirector directly from the codebehind. Instead of the Redirector returned the event handler to the codebehind of the page being accessed. Each codebehind class contains a method that is always called when the respective aspx page is accessed, the PageLoad method. Each codebehind obtains the next event and respective event handler from the Redirector in a call from the PageLoad method. The codebehind then delegates the control to the handler, which behaves in a similar manner as the J2EE counterpart. It instantiates business objects, which execute requests to the web service, and updates the model. The two applications exhibit very similar design. I purposely maintained this similarity to be able to make the comparisons more clear. Allowing the same flexibility and the same number of layers in the corresponding tiers closely approaches a comparison of apples to apples and not apples to oranges. The business tier of the application, just as in the J2EE implementation is the one responsible for contacting and processing information back from the Axis web service. Once the processing is complete, the event handler sets the information received in the codebehind, the model, of the target view. The target view, or aspx page, uses the code behind data to display the results on the browser. The solution is not as clean as in J2EE, where the model, the view and the controller and completely separated.

Chapter 7

PERFORMANCE TESTING

Performance testing of a Web site is basically the process of understanding how the Web application and its operating environment respond at various user load levels. In general, we want to measure the latency, throughput, and utilization of the Web site while simulating attempts by virtual users to simultaneously access the site. One of the main objectives of performance testing is to identify a Web site's level of latency.

7.1 Performance Testing Concepts

7.1.1 Latency

Latency is often called response time. It is the delay experienced between the time a request is made and the time the server's response is received. It is usually measured in units of time, such as seconds or milliseconds. In certain testing tools, such as the Microsoft® Web Application Stress (WAS) tool, latency is best represented by the metric "time to last byte" (TTLB), which calculates the time between sending out a Web page request and receiving the last byte of the complete page content. Latency increases slowly at low levels of user load, but increases rapidly as capacity is utilized. The sudden increase in latency is often caused by the maximum utilization of one or more system resources. For example, most Web servers can be configured to start up a fixed number of threads to handle concurrent user requests. If the number of concurrent requests is

greater than the number of threads available, any incoming requests will be placed in a queue (up to a maximum number) and will wait for their turn to be processed. Any time spent in the queue naturally adds extra wait time to the overall latency.

7.1.2 Throughput

Throughput is represented as the number of client requests processed within a certain unit of time, typically a second. A throughput graph will increase with a load increase until a maximum throughput is achieved, then decline as the load continues to increase. In many ways, throughput and latency are related. In general, sites with high latency will have low throughput. Measurement of throughput without consideration of latency is misleading because latency often rises under a given load before throughput peaks. This indicates that peak throughput may occur at a latency that is unacceptable from an application usability standpoint.

7.2 Test Environment

7.2.1 Hardware Configuration

Two separate machines were used when executing testing. One of the machines played the role of the web server, hosting the Mega Vega Weather site. The other, simulated the remote component, the Axis web service and it also contained the MySQL data storage. The clients were run locally on the same serve machine pointing to the local host web server.

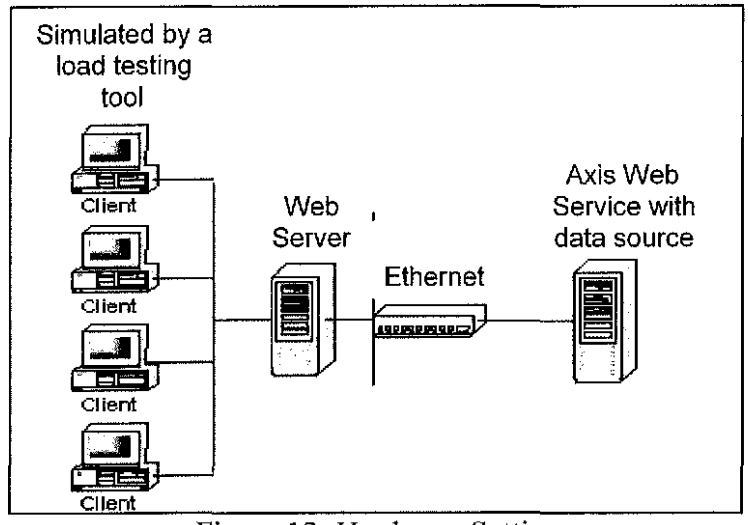


Figure 12: Hardware Settings

7.2.2 Test Environment Specifications

The following table illustrates the hardware and software specifications for the two machines involved in the testing process. The stress testing tool used to simulate client load was the Microsoft® Web Application Stress (WAS) tool.

Specification	Web Server Host Computer	Web Service Host Computer
CPU	Single Processor - 2.0 GHz	Single Processor – 1.5 GHz
RAM	256 MB	1.0 GB
Network	100 Mbps NIC	100 Mbps NIC
OS	Windows 2000 Advanced Server	Windows XP Home Edition

Table 2: Hardware/Software Specifications

7.3 Testing Methodology

The testing was divided into two distinct sessions, the J2EE implementation testing and the .NET implementation testing. The initial stage of the testing was setting up and starting all involved background processes. In testing both implementations the web service machine involved starting Tomcat and the MySQL database process. For the J2EE testing session, Tomcat was started on the web server machine. For the .Net testing session, Internet Information Service was started on the web server machine.

Since both ASP.NET and JSP required compilation the first time a given page is requested. I opened a browser session and proceeded to hit the pages that were participating in the test, to avoid delay and inconsistency of compilation.

Getting the simulated clients ready, simply involved opening the Web Application Stress tool and recording the test scripts. The first test script began with a hit to the main page, and performing a second request by hitting city/state observation and forecast. A second test script was recorded by initially hitting the main page, but the second request consisted of a satellite image. Due to the content of the data requested the scripts can be classified as a light request for the observation / forecasts test script, and a heavier request for the test script involving the satellite image request. The web application stress tool allows for multiple user configurations. Tests were run using the following user loads, 1, 5, 10, 25, 50, 75 and 100 users. The tests duration was set to 30 minutes. This means that for 30 minutes, the web application stress tool requested the recorded script incessantly from the web server, which in turn contacted the web service.

Chapter 8

RESULTS AND ANALYSIS

Initially, the web service experienced problems and the Tomcat servlet engine was throwing logging exceptions. The testing session was discontinued and I proceeded to research the problem. The Apache web site provided answers in the Tomcat server configuration section. The Tomcat server configuration specifies the Coyote HTTP ./1.1 Connector. The default settings of the connector are to create a maximum number of 75 threads with a maximum queuing of 100. I modified the default settings to create a maximum number of 200 threads with a queuing maximum of 300 threads. I re started the testing session only to find that after 75 concurrent users the Java Virtual Machine started to report out of memory exceptions. Again, I stopped all testing and researched the problem. The Java Virtual Machine allows command line inputs to override the default startup settings. The JVM argument –Xms specifies how much memory is allocated for the heap when the JVM starts. I selected the start up value of 128MB, from a recommended setting in the Apache Tomcat site. The JVM argument –Xmx specifies the maximum heap size the Java interpreter will use for dynamically allocated objects and arrays. The recommended value again was 128MB.

The memory settings must be specified in the CATALINA_OPTS environmental variable as follows: set CATALINA_OPTS = -Xms128m –Xmx128m. Resume normal startup of Tomcat. The CATALINA_OTPS will be picked up by the Tomcat startup script.

I resumed the testing session again with the new settings, this time the testing session was a success and the entire set of user loads were run against the local host. The .NET testing session involved starting up the IIS web server and recording the same scripts as the Java testing session. Since all the settings to run the web service had already been performed, this testing session was successful the first time. The web application stress tool produces a tabulated report from which I constructed the datasheets and graphs to illustrate the results.

8.1 Results

To determine the latency of the web application the following graphs were created from the Time To Last Byte (TTLB) metric. A set of five operations were chosen to provide a variety and coverage of the test. The following table depicts the hits selected in the graphs.

Operation	Type of data	Reason for selection
GET	Initial Page (main page)	Combination of image and textual data.
GET	Image (background)	Large image
GET	Image (logo)	Small image
POST	Form content (text)	Small data illustrates post back to the server
GET	Image (product)	Medium size image

Table 3: Graphed Operations

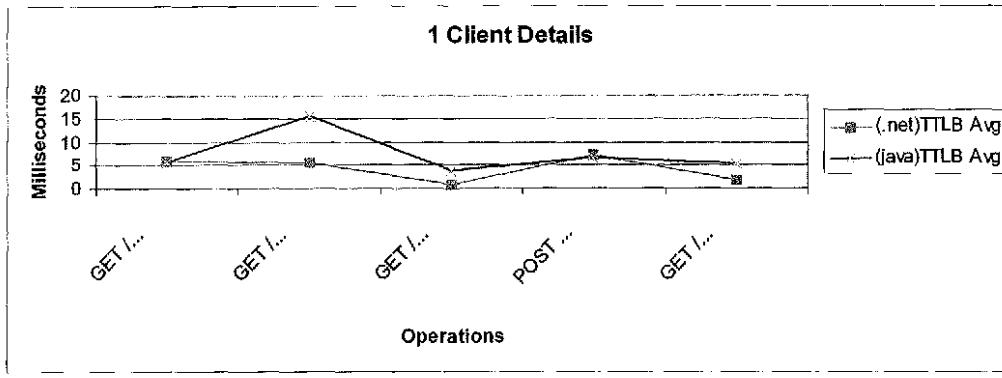


Figure 13: TTLB Averages with a Single User

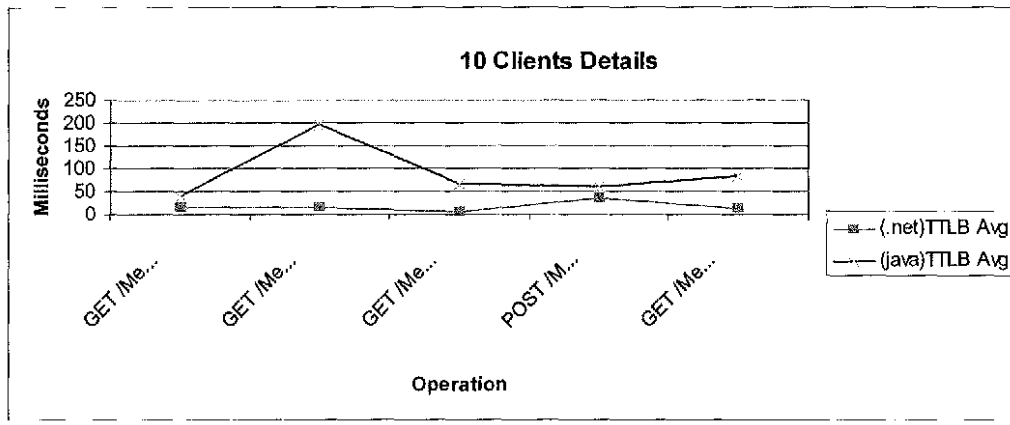


Figure 14: TTLB Averages with 10 Users

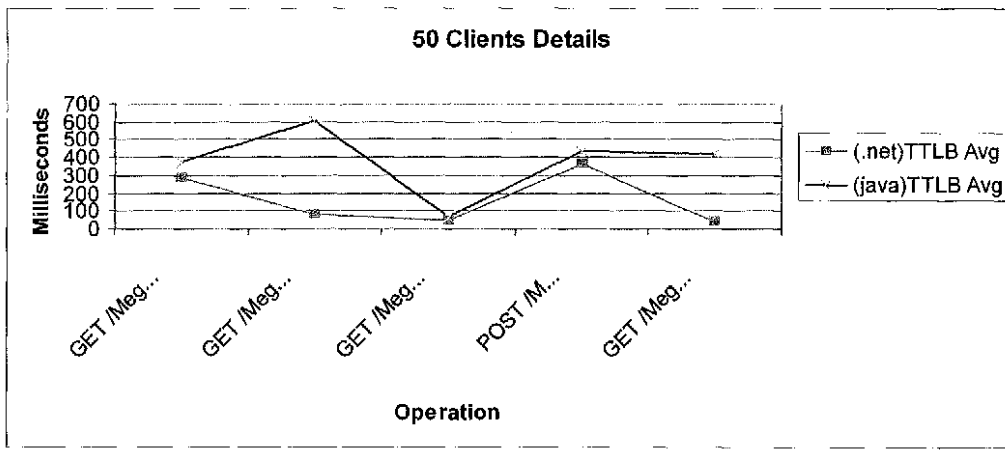


Figure 15: TTLB Averages with 50 Users

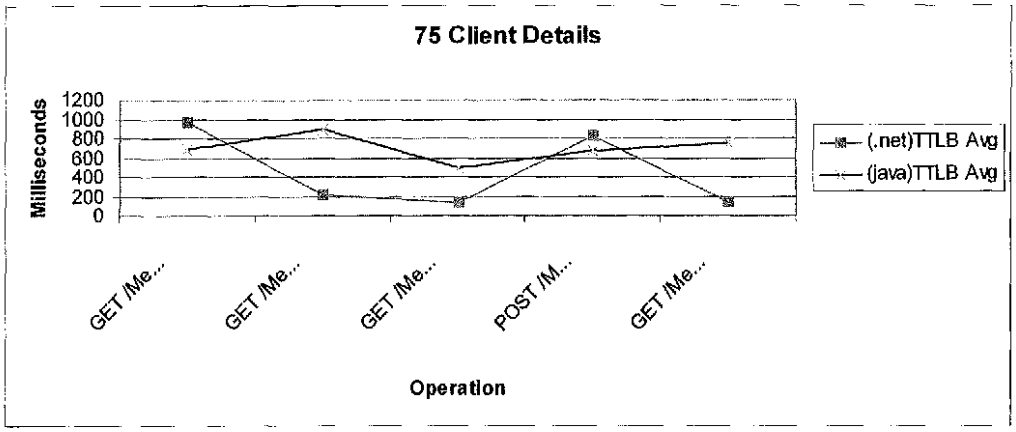


Figure 16: TTLB Averages with 75 Users

8.2 Analysis

The results confirm that the .NET implementation is significantly faster, but the inconsistency across operations left me with a serious doubt. It appears from the graphs, that the performance of .NET was significantly faster for images. The largest image shows the largest difference between the performances of the two implementations. A closer look at the graphs indicates that in small GET operations, for example, the initial hit to the main page, the small logo image and the POST operation exhibit a similar performance. I conducted additional research to attempt to analyze and understand these discrepancies.

The Apache web site has a set of best practices for using Apache products. The preferred production configuration is a combination of the Apache HTTP server, and the Tomcat servlet engine in combination. According to Apache foundation, the Apache HTTP server is tuned for static content delivery. The following table illustrates a performance comparison of Apache HTTP server to Tomcat and Jetty serving the same static data.

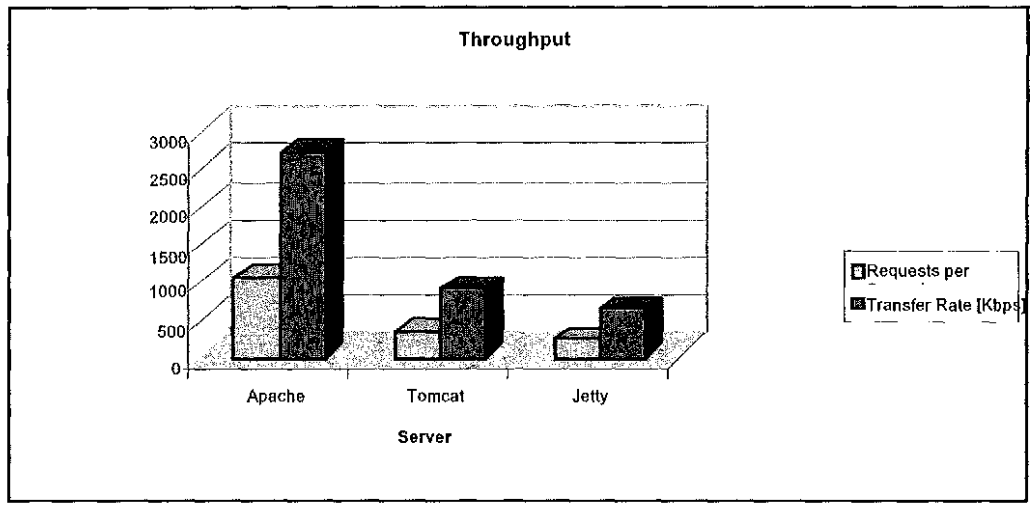


Figure 17: Throughput of Web servers

The purple bars indicate the actual number of requests per second, or the throughput of the server. Recall that throughput and latency are closely related to each other. Another measurement of throughput is the transfer rate represented by the burgundy bars. From this graph it is clear that Apache HTTP server is significantly more efficient that Tomcat when rendering static content and that Jetty comes dead last. These results explain the great differences between the .NET results and the Tomcat results when measuring the operations that involved retrieval of static content, especially when larger images were involved. Other operations, showed .NET with a slight performance gain, but not a real significant difference.

Chapter 9

QUALITATIVE COMPARISON

Qualitative factors of software applications are not as clear as the numbers produced by a performance analysis, and thus are much more subjective. Some of the areas most important to software project managers include time-to-market, availability of software, language support, maintainability, portability, scalability and cost.

9.1 Qualitative Factors

9.1.1 Time to market

When developing web application solutions in today's marketplace, a few months of time can be a significant factor. Missing a small window of opportunity may be the difference between a business that is first to market, and a business that must play catch-up for years. One way to speed time to market is to choose a Web platform that allows rapid application development. This enables developers to write, test and maintain code faster and more efficiently thus reducing time to market.

Both J2EE and .Net offer powerful Integrated Developing Environments (IDE) that greatly reduces the development time. It is important here to consider that .NET is a product ready to be used in development, where as J2EE is a set of standards with no specific implementation. Both platforms however offer rapid development with the availability of frameworks. ASP.NET offers great enhancements especially with the innovative server controls which eliminate the need to scripting languages. Several web

frameworks like Jakarta Struts, are available freely to boost J2EE web development. In my personal experience while building both applications I feel that the Microsoft .Net environment offers a simpler development environment and a significantly more simple deployment environment. The total number of lines of code is often regarded as a unit of measurement of the effort required to develop an application. The graph representing the lines of code required to develop both applications shows a similar amount of code required. I believe the similarities stem from the fact that both implementations followed a similar architecture. In my opinion this software metric can be easily manipulated, and it not really indicative of the development effort.

9.1.2 Availability of Pluggable Software

The initial set of Application Program Interfaces (API) that the Microsoft .NET framework supports is very large. However, due to language maturity the number of available APIs for the Java language is much larger. There is an abundance of third party software, commercial and open source that are readily available for programmers to grab and use. In this category I feel the J2EE environment has at least for now, the upper hand.

9.1.3 Maintainability

Ease of maintenance is closely related to developing environments, but it is also affected by software design and applying best practices of software development. In my opinion, this metric is more specific to a given software system. In my case, the only real basis for comparison will be the ease of maintenance the Mega Vega Weather web applications may have. As previously explained in the application details section, the implementation

of the J2EE web application offered a cleaner separation between the model, the view and the controller. Additionally one could speculate that since .Net supports multiple languages, programmers will write code in their preferred language. This may lead to a disorder of code practices that may potentially become a threat to maintainability.

9.1.4 Language Support

Clearly .Net is the winner in this category since .Net offers a programming environment that support multi-language development. J2EE on the other hand is a single language platform, the Java language.

9.1.5 Portability

Microsoft .Net only runs on the Windows family of operating systems. Since J2EE is implemented with the Java language, it is platform agnostic.

9.1.6 Scalability

Based on the weather web applications the scalability of both systems is similar. Both applications use the Http Session object to maintain state. Clearly neither application will be able to participate in a Web Farm; therefore their scalability is limited.

9.1.7 Cost

The cost involved in development and deployment of web applications is not only related to the cost of hardware, and software, but also development time and even more significant is the maintainability of the application. Determining the cost of a software system includes taking into account factors like maintainability, time-to-develop, availability of software plus the cost of hardware and software to deploy the server.

Servlet engine Tomcat, Apache Http sever, a large collection of freely available APIs, free web frameworks like Struts make the J2EE option the most economic. Production Microsoft .Net must run on the Windows 2000 Advanced Server or the Windows 2000 .Net Server neither of these servers are free or charge.

Chapter 10

CONCLUSIONS

According to the recorded data, the .Net implementation exhibits better performance, at least rendering static content. However, the significant improvements that Apache Http server can offer to Tomcat may bring the gap much closer. The time to development and ease of deployment of the .Net environment offer attractive gains in the time to market comparison. The maturity of J2EE however, may offer a wider range of solutions and a large code repository whereby programmers may reduce the development effort. Both technologies offer similar functionality for applications development. Both technologies aim for a container managed environment, where the container provides a number of services like transaction management, message queuing and legacy systems integration. Web services are becoming increasingly more important in the distributed programming arena and both technologies provide ample support for the development, deployment and consumption of web services.

Developing new small web applications may best be accomplished by using the benefits and rapid development of ASP.NET. Particularly attractive is the fact that in ASP.NET the need to use a scripting language is eliminated. The J2EE community has released a similar concept to the server controls Microsoft has implemented, but at the time this comparison is written this is very new technology and the writer has not had the opportunity to evaluate it. For larger web applications where the adherence to proven

design patterns and maintainability is a concern the maturity of J2EE is significantly more attractive.

Performance is certainly an important part of the success of any web enterprise. However, performance is not a single isolated quality of software systems. Most often performance is related to good object oriented design and best practices for software development. Performance is often inversely proportional to maintainability. However, it is better to design a system with maintainability in mind, profile, load test it, and improve the performance as required. The software product obtained by using such practice will be much more maintainable and scalable, the one constructed with performance as the driving effort.

J2EE and Dot Net both provide a set of solutions for today's programming challenges. In my opinion programmers in the web enterprise arena need to shift from J2EE vs. .Net mentality, to a coexistence of J2EE and .Net technologies.

Chapter 11

FUTURE DEVELOPMENTS

The proposed developments will focus on two new performance comparisons. One, consisting of measuring the performance of the J2EE implementation using the recommended production settings of Apache Http web server in combination with Tomcat, and comparing those results to the .NET results already collected. The second comparison will involve building the weather web service using .NET and measuring the performance of both applications with all components built with their own technology.

REFERENCES

[Anderson02]
Anderson, R. et al, Professional ASP.NET 1.0 Special Edition, Wrox Press Ltd., Birmingham, U.K., 2002.

[Chappell02]
Chappell, D., and Kirk, S., "Application Design Guidelines: From N-Tier to .NET", Technical Report, Chappell and Associates, San Francisco (April, 2002).

[Ching01A]
Ching, A., Silva, P., and Wagner, A., "Performance Testing with the Web Application Stress Tool", Technical Report, Microsoft Developer Network (January, 2001).

[Ching01B]
Ching, A., and Wagner, A., "Understanding Performance Testing", Technical Report, Microsoft Developer Network (February, 2001).

[Gielens02A]
Gielens, P., "Model View Controller (MVC) Using C#, Delegates, and Events in .NET", Technical Report, The Code Project (May 2002).

[Gielens02B]
Gielens, P., "MVC for ASP .NET", Technical Report, DMB Consulting LLC (September, 2002).

[Johnson02]
Johnson, R., Expert one-on-one J2EE Design and Development, Wrox Press Ltd., Birmingham, U.K., 2002.

[Liberty01]
Liberty, J., Programming C#, O'Reilly & Associates, Sebastopol, CA, 2001.

[Lunn02]
Lunn, J., et al, Professional .NET for Java Developers with C#, Wrox Press Ltd., Birmingham, U.K., 2002

[Maddox02]
Maddox, A. A., "Distributed Web Application Development: A Comparison of .Net and J2EE", Technical Report, Department of Electrical and Electronic Engineering, Manukau Institute of Technology, Auckland, New Zealand, 2002.

[Platt01]
Platt, D. S., Introducing Microsoft .NET, Microsoft Press, Redmond, WA, 2001.

[Thai01]
Thai, T. and Lam H. Q., .Net Framework, O'Reilly & Associates, Sebastopol, CA, 2001.

[Troelsen01]
Troelsen, A., C# and the .NET Platform, Apress, Berkeley, CA, 2001.

[Vawter01]
Vawter, C. and Roman, E., "J2EE vs. Microsoft .NET: A comparison of building XML-based web services", Technical Report, The Middleware Company, Austin, 2001.

Appendix A

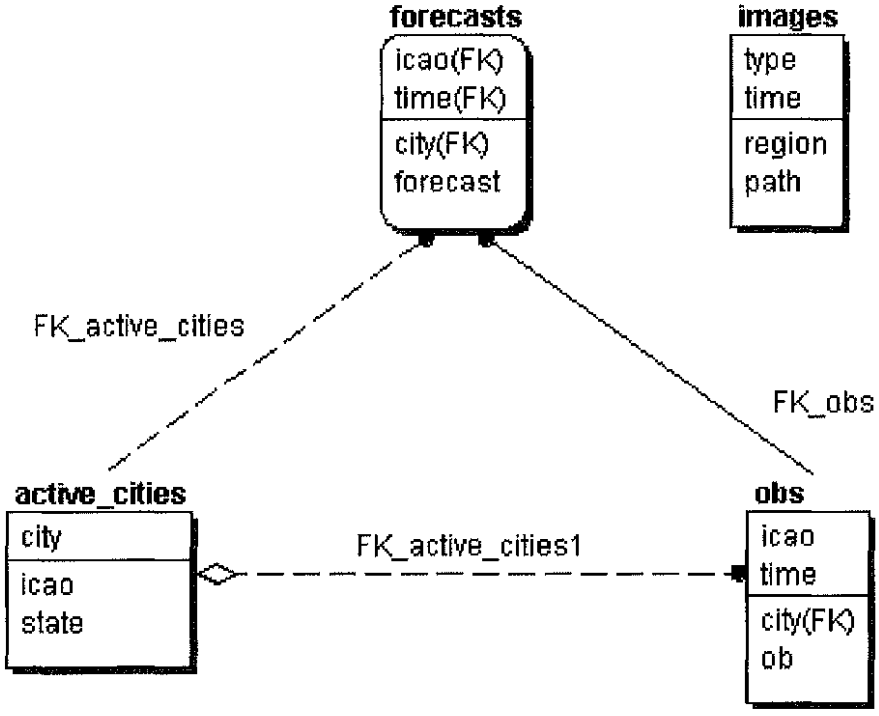
Web Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://192.168.0.164/axis/services/WeatherService"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://192.168.0.164/axis/services/WeatherService"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types />
  <message name="getWxForecastRequest">
    <part name="city" type="s:string" />
  </message>
  <message name="getObResponse">
    <part name="getObReturn" type="s:string" />
  </message>
  <message name="getWxImageRequest">
    <part name="type" type="s:string" />
    <part name="region" type="s:string" />
  </message>
  <message name="getWxImageResponse">
    <part name="getWxImageReturn" type="s:base64Binary" />
  </message>
  <message name="getObRequest">
    <part name="city" type="s:string" />
  </message>
  <message name="getWxForecastResponse">
    <part name="getWxForecastReturn" type="s:string" />
  </message>
  <portType name="MegavegaWx">
    <operation name="getOb" parameterOrder="city">
      <input name="getObRequest" message="tns:getObRequest" />
      <output name="getObResponse" message="tns:getObResponse" />
    </operation>
    <operation name="getWxForecast" parameterOrder="city">
      <input name="getWxForecastRequest"
message="tns:getWxForecastRequest" />
      <output name="getWxForecastResponse"
message="tns:getWxForecastResponse" />
    </operation>
    <operation name="getWxImage" parameterOrder="type region">
      <input name="getWxImageRequest" message="tns:getWxImageRequest"
/>
      <output name="getWxImageResponse"
message="tns:getWxImageResponse" />
    </operation>
```

```
  </portType>
  <binding name="WeatherServiceSoapBinding" type="tns:MegavegaWx">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="rpc" />
    <operation name="getOb">
      <soap:operation soapAction="" />
      <input name="getObRequest">
        <soap:body use="encoded"
namespace="http://192.168.0.164/axis/services/WeatherService"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output name="getObResponse">
        <soap:body use="encoded"
namespace="http://192.168.0.164/axis/services/WeatherService"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
    <operation name="getWxForecast">
      <soap:operation soapAction="" />
      <input name="getWxForecastRequest">
        <soap:body use="encoded"
namespace="http://192.168.0.164/axis/services/WeatherService"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output name="getWxForecastResponse">
        <soap:body use="encoded"
namespace="http://192.168.0.164/axis/services/WeatherService"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
    <operation name="getWxImage">
      <soap:operation soapAction="" />
      <input name="getWxImageRequest">
        <soap:body use="encoded"
namespace="http://192.168.0.164/axis/services/WeatherService"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output name="getWxImageResponse">
        <soap:body use="encoded"
namespace="http://192.168.0.164/axis/services/WeatherService"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  </binding>
  <service name="MegavegaWxService">
    <port name="WeatherService"
binding="tns:WeatherServiceSoapBinding">
      <soap:address
location="http://192.168.0.164/axis/services/WeatherService" />
    </port>
  </service>
</definitions>
```

Appendix B

Web Service Data Model



Appendix C

Web Service Source Code

```
package mega.ws.core;

import java.io.*;
import mega.ws.util.*;

public class WxAdmin {


	public static void main(String args[]){

		BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
		String input = null;
		String errmsg = null;
		WeatherHandler handler = new WeatherHandler();

		try{

			while((input = DisplayMenu(in, errmsg, handler))!= "7"){
				if(input.equals("bad")){
					errmsg = "INVALID COMMAND";
				}
				else{
					errmsg = null;

				}
				//clear screen
			}
		}catch(Exception e){System.out.println("ERROR: program exiting...");}
	}

	private static String DisplayMenu(BufferedReader in, String errmsg,
WeatherHandler handler)
	throws IOException {

		FtpDownloader ftp = new FtpDownloader(null);
		FileHandler h = new FileHandler("");
		ObUpdator ob = new ObUpdator();
		//ForecastUpdator fore = new ForecastUpdator();


		if(errmsg != null){
			System.out.println("ERROR reading input "+errmsg);

		}

		System.out.println("              WX Administration Client
");
```

```
        System.out.println("=============================================================");
                System.out.println("|
|");
                System.out.println("| 1 Retrieve Observation by ICAO
|");
                System.out.println("| 2 Retrieve Forecast by city name
|");
                System.out.println("| 3 Start Observation Auto Update
|");
                System.out.println("| 4 Stop Observation Auto Update
|");
                System.out.println("| 5 Start Forecast Auto Update
|");
                System.out.println("| 6 Stop Forecast Auto Update
|");
                System.out.println("| 7 Quit
|");

        System.out.println("=============================================================");

                String cmd = in.readLine();
                String icaofile = null;
                String data = null;
                String path = Constants.DATA_DIR+Constants.FILE_SEPARATOR;

                if("1".equals(cmd)){

                        System.out.println("Enter ICAO.TXT ");
                        System.out.println(">");
                    icaofile = in.readLine();
                    System.out.println("going to get the ob for: "+icaofile);
                        data = ftp.getOb(icaofile);
                    updateOb(icaofile, data, handler);
                }
                else if("2".equals(cmd)){

                        System.out.println("Enter state/city.txt state must be
abbreviated.");
                        System.out.println(">");
                    icaofile = in.readLine();
                    System.out.println("going to get the forecast for: "+icaofile);
                        data = ftp.getForecast(icaofile);
                    updateForecast(icaofile, data, handler);
                }

                else if("3".equals(cmd)){
                        ob.t.start();
                        System.out.println("Ob updator has started ");
                }
                else if("4".equals(cmd)){
                        ob.t.interrupt();
                }
                else if("5".equals(cmd)){

                //      ForecastUpdator fore = new ForecastUpdator();
                }
```

```
            else if("6".equals(cmd)){

            //      ForecastUpdator fore = new ForecastUpdator();
            }
            else if("7".equals(cmd)){
                  System.out.println("Good bye...");
                  ftp.disconnect();

                  return cmd;
            }
            else{
                  return "bad";
            }
            return icaofile;
      }

      private static void updateOb(String city, String data, WeatherHandler h){

            h.updateObs(getObCityName(city), data);

      }

      private static void updateForecast(String icao, String data, WeatherHandler h){

            h.updateForecast(getForecastCity(icao), data);
      }


      private static String getObCityName(String icao){

            //Strip off the .txt
            icao = icao.substring(0,4);
            return icao;
//ConfigurationHandler.getInstance().getCityName(city.toLowerCase());

      }

      private static String getForecastCity(String icao){

            //Strip off the .txt
            int len = icao.length();
            icao = icao.substring(3,len-4);
            char [] bytes = icao.toCharArray();
            bytes[0] = Character.toUpperCase(bytes[0]);

            return new String(bytes);
//ConfigurationHandler.getInstance().getCityName(icao);

      }

}
```

```
package mega.ws.core;

import mega.ws.exceptions.*;
import java.util.Properties;
import java.util.HashMap;
import java.io.IOException;
import java.io.FileNotFoundException;
import mega.ws.util.*;
import mega.ws.jdbc.*;
import java.sql.*;

public class ConfigurationHandler {

	private Properties configProps = null;
	private boolean initialized;
	private static ConfigurationHandler instance = new ConfigurationHandler();
	private HashMap icaos= new HashMap();


	public static ConfigurationHandler getInstance(){
		return instance;
	}

	private ConfigurationHandler()
	{

	}

	public void initialize(String directory, String file)
	throws InitializationException {


		try{
			//load default if either parameter is null
			if(directory == null || file == null){
				loadConfiguration(Constants.DEFAULT_DIR,
Constants.CONFIG_FILE);
			}
			else{
				loadConfiguration(directory, file);
			}


		}
		catch(FileNotFoundException nf){
			throw new InitializationException("Configuration File not
found",nf);
		}
		catch(IOException e){
			throw new InitializationException("Unable to read config
file",e);
		}

		if(!configProps.isEmpty()){
			initialized = true;
		}
		else{
```

```
                throw new InitializationException("Configuration Init Failed");
        }
}
/**
 * setConfiguration method is public so we can call it and
 * get additional configuration properties on the fly.
 * Always called from the Controller with the initial configuration
 */
public void loadConfiguration(String directory, String file)
throws FileNotFoundException, IOException {

        FileHandler handler = new FileHandler(directory);
        configProps = (Properties) handler.readConfiguration(file);
}

/**
 * @returns a given configuration property
 * @ null if not found.
 */
public String getConfigProperty(String key){

        return (String)configProps.get(key);
}

/**
 * This method is for administration purposes
 * will add or change a given configuration property
 * obtained from the admin page.
 */
public void setConfigProperty(String key, String value){

        configProps.put(key, value);
}


public void addToConfiguration(Properties p){

        configProps.putAll(p);
}

public void removeFromConfig(String key){

        configProps.remove(key);
}

public boolean containsProperty(String key){

        return configProps.contains(key);
}

public String getCityName(String key){
        return (String) icaos.get(key);
}


public HashMap loadIcaos(WeatherDataSource ds){
```

```
		//check if already loaded
		if(icaos.size() > 0){
			return icaos;
		}

		//WeatherDataSource ds = new WeatherDataSource(false);
		JdbcTemplate jt = new JdbcTemplate(ds);
		ResultSet result = null;
		String sql = "SELECT city, icao from Active_Cities";

		try{
		Connection con = ds.getConnection();
		Statement s = con.createStatement();
		result = s.executeQuery(sql);
		processRow(result);

	 }catch(SQLException e){
		System.out.println("ERROR getting city names "+e.getMessage());
		}
		finally{
			try{
				result.close();
			}catch(SQLException e){
				System.out.println("ERROR closing sql connection");
			}
		}
		return icaos;
	}

	private void processRow(ResultSet rs)
	throws SQLException {

		while(rs.next()){
			icaos.put(rs.getString("city"), rs.getString("icao"));
		}



	}


}
```

```
package mega.ws.core;

import mega.ws.util.*;


public class MegavegaWx {


    private WeatherHandler handler = new WeatherHandler();

    public MegavegaWx(){}

    public String getOb(String city){
       return handler.getOb(city);
    }

    public String getWxForecast(String city){

        return handler.getForecast(city);
    }

    public byte [] getWxImage(String type, String region){

//System.out.println("Got a request for image:"+ type+" - "+region);
        // StringBuffer path = new StringBuffer(Constants.DATA_DIR);
                FileHandler filehandler = new FileHandler(Constants.DATA_DIR);

                //path.append(Constants.FILE_SEPARATOR);
                //if("SAT".equals(type)){
                        //path.append(Constants.SAT_DIR);
                //}
                //else{ //is radar image
                        //path.append(Constants.RADAR_DIR);
                //}
                //path.append(Constants.FILE_SEPARATOR);
                //path.append(handler.getImageName(type, region, null));
         String path = handler.getImageName(type, region, null);
//System.out.println("about to read the file -- "+path);
          byte [] image = filehandler.readFile(path);
System.out.println("read successfull");
//System.out.println("length of array is: "+image.length);
          return image;
    }


}
```

```
package mega.ws.core;

import java.util.*;
import mega.ws.exceptions.DataAccessException;
import mega.ws.jdbc.*;
import mega.ws.util.FtpDownloader;
import mega.ws.util.Constants;

public class ObUpdator implements Runnable, Updator {

	private static final long OB_INTERVAL = 3600000;//one hour
	private static final String TXT = ".TXT";
	public Thread t = null;
	private WeatherDataSource ds=null;
	private WeatherHandler wh = null;
	private FtpDownloader ftp = null;
	private JdbcTemplate jdbc = null;

	public ObUpdator(){

		ds = new WeatherDataSource(false);
		wh = new WeatherHandler();
		ftp = new FtpDownloader(null);
		t=new Thread(this);
		//t.start();
	}

	/**
	 * @ int result is number of successful retrievals.
	 * Future use may be to compare to requested list and log discrepancies.
	 */

	public void run(){

		try{
			while(true){
				HashMap data = getData(getIcaos());
				int result = writeToDb(data);
				t.sleep(OB_INTERVAL);
			}

		}catch(InterruptedException e){
			System.out.println("Ob updator interrupted");
		}

	}

	public boolean updateSingle(String icao, String data){

		return true;
	}

	public int updateMultiple(HashMap data){

		int totalUpdated=0;

		return totalUpdated;
```

```
}

private HashSet getIcaos(){

        //make config handler load the data from db
        HashMap allCities = ConfigurationHandler.getInstance().loadIcaos(ds);
        //get the icaos, but need to add the .TXT for data retrieval from NWS
        HashSet set = new HashSet(allCities.values());

        return set;

}

/**
 * Method to add the .TXT required by NWS FTP server to retrieve ob info
 * The directory path for decoded obs is added when the FTPDownloaded calls
 * @ returns a HashSet of ready to be called icaos;
 */
private HashMap prepForCall(HashSet icaos){

        HashMap dataset = new HashMap(icaos.size());

        for(Iterator i = icaos.iterator(); i.hasNext();){
                String s= (String) i.next();
                dataset.put(s, s+TXT);
        }

        return dataset;
}
/**
 * Method to retrieve the data stream from NWS.
 * @return is HashMap of icaos (without .TXT mapped to data)
 */
private HashMap getData(HashSet icaos){

        //add the .TXT

        return ftp.getObList(prepForCall(icaos));

}

private int writeToDb(HashMap dataset){

        int count=0;

        for(Iterator i = dataset.keySet().iterator(); i.hasNext();){
                String key = (String)i.next();
                wh.updateObs(key, (String)dataset.get(key));
                count++;
        }

        return count;

}
```

```
}


package mega.ws.core;

import java.sql.ResultSet;
import java.sql.SQLException;

/**
* Callback interface used by the JdbcTemplate class.
* Implementations of this interface
* perform the actual work of extracting results,
* but don't need to worry about exception handling.
*/
public interface RowCallbackHandler {

        /**
         * Process one row of data. This method should not call
         * next() on the ResultSet, but extract the current values.
        * Exactly what the implementation chooses to do is up to it;
        * a trivial implementation might simply count rows, while another
        * implementation might build an XML document.
        * @param ResultSet
        * @throw SQLException if a SQLException is encountered getting
        * column values (that is, there's no need to catch SQLException)
        */
        String processRow(ResultSet rs) throws SQLException; // app exception


}


package mega.ws.core;

import java.util.HashMap;

public interface Updator {

        public boolean updateSingle(String icao, String data);

        public int updateMultiple(HashMap data);


}
```

```
package mega.ws.core;


import java.sql.*;
import javax.sql.DataSource;
import java.util.*;
import java.io.PrintWriter;
import mega.ws.util.Constants;


public class WeatherDataSource implements DataSource {

//      public static Logger logger = Logger.getLogger(WeatherDataSource.class);
        private String dbUrl    = null;
        private String userName = null;
        private String password = null;
        private Connection con  = null;
        private boolean connected;
        private PrintWriter writer = new PrintWriter(System.out);
        private int loginTimeout;
        ConfigurationHandler h = ConfigurationHandler.getInstance();

        /**
         * @param boolean b to connect to db on instanciation
         */
        public WeatherDataSource(boolean stayConnected){


                try{
                        h.initialize(null,null);

                }catch(mega.ws.exceptions.InitializationException e){
                        System.out.println("ERROR initing Cache Manager in Weather data
source " +e.getMessage());
                }
                dbUrl = h.getConfigProperty(Constants.dbUrl);
                userName = h.getConfigProperty(Constants.dbUser);
                password = h.getConfigProperty(Constants.dbPassword);
                loadDriver(h.getConfigProperty(Constants.jdbcDriver));
                h.loadIcaos(this);

        }

        private void loadDriver(String jdbcDriver){

                try{
                        Class.forName(jdbcDriver);
                }
                catch(ClassNotFoundException nf){
        //              logger.fatal("JDBC driver class not found");
                }

        }

        /**
         * Connecting method
         */
```

```
	public Connection getConnection(){

		return getConnection(userName, password);
	}
	public Connection getConnection(String userName, String password){

		try{

	//if(((String)h.getConfigProperty(Constants.dbType)).equals("mysql")){
					//con = getMySqlConnection();
				//}
				//else{
					con=DriverManager.getConnection(dbUrl, userName,
password);
				//}

		}catch(SQLException e){
				System.out.println(e.toString());
		}
		return con;
	}

	public Connection getMySqlConnection()
	throws SQLException {

		StringBuffer url = new StringBuffer(this.dbUrl);
		url.append("user="+ userName);
		url.append("&password="+password);

		return DriverManager.getConnection(url.toString());
	}
	/**
	 * Disconnect method
	 */
	public void disconnect(){

		try{
			if(con != null && !con.isClosed()){
				con.close();
			}
		}catch(SQLException e){
				System.out.println(e.toString());
		}

	}

	/**
	 * Sets the dbUrl
	 * @param dbUrl The dbUrl to set
	 */
	public void setDbUrl(String dbUrl) {
		this.dbUrl = dbUrl;
	}
```

```
        /**
         * Sets the userName
         * @param userName The userName to set
         */
        public void setUserName(String userName) {
                this.userName = userName;
        }



        /**
         * Sets the password
         * @param password The password to set
         */
        public void setPassword(String password) {
                this.password = password;
        }


        public PrintWriter getLogWriter(){
                return writer;
        }

        public void setLogWriter(PrintWriter writer){
                this.writer = writer;
        }


        public int getLoginTimeout(){
                return loginTimeout;
        }

        public void setLoginTimeout(int loginTimeout){
                this.loginTimeout = loginTimeout;
        }




}
```

```
package mega.ws.core;

import mega.ws.jdbc.*;
import mega.ws.exceptions.DataAccessException;
public class WeatherHandler {


        private JdbcTemplate jdbc = null;
        private static final String obQuery="SELECT ob from OBS where city = ?";
              private static final String forecastQuery =
                    "SELECT forecast from FORECASTS where city = ?";
              private static final String imageQuery =
                    "SELECT path from IMAGES where TYPE = ? AND REGION = ?";

              private static final String updObsQuery=

                    "UPDATE OBS SET time = now(), ob=? where icao=?";
              private static final String updForecastQuery =
                    "UPDATE FORECASTS SET time = now(),forecast=? where city=?";

              private WeatherDataSource ds = new WeatherDataSource(false);

        public WeatherHandler(){

            jdbc= new JdbcTemplate(ds);
        }

        public String getImageName(String type, String region, String time){

                    //String [] params = new String[2];
           // params [0] = type;
           // params [1] = region;
           // params [2] = time;
            region = region.toLowerCase();
            String temp = null;
            int pos = region.indexOf(" ");
            if(pos != -1){
              temp = region.substring(0,pos);
              if(!temp.equals("continental")){
                    temp+= region.substring(pos+1,region.length());
              }
              temp+=".jpg";
            }
            String path = null;

            //try{
            //System.out.println("recieved type: "+type+" and region:"+region);
            //path = jdbc.query(PreparedStatementCreatorFactory.
            //              newPreparedStatementCreator(imageQuery), new
RowHandler(),
            //              params);

            if("SAT".equals(type)){
              path="C:\\weatherData\\sat\\"+temp;
            }
            else{
              path="C:\\weatherData\\radar\\"+temp;
```

```
            }
              System.out.println("Image path is:"+path);

            // }catch(DataAccessException e){
            // System.out.println("ERROR getting "+type+" Image for :"+region);
            // }
             return path;
        }

        public String getOb(String city){

            String s=null;
            String [] params = new String[1];
            params [0]= city;

            //try{
            // s=jdbc.query(PreparedStatementCreatorFactory.
            //        newPreparedStatementCreator(obQuery),  new RowHandler(), params
);
            //}catch(DataAccessException e){
            // System.out.println("ERROR getting OB for :"+city);
            //}
            //hardcode to avoid going to the database for retrieving
            //this is needed for testing of performance.  Database limits to
            //75 connection unless Pools are implemented
            s="Weather for Jacksonville Florida";
            s+="Sky condition: Clear";
            s+="Wind: North east at 10 mph";
            s+="Temperature/Dewpoint: 56F/54F";
            System.out.println("**** sending ob: length: "+s.length());
            return s;
        }

        public String getForecast(String city){

            String s=null;
            /*String [] params = new String[1];
            params [0]= city;

            try{
              s=jdbc.query(PreparedStatementCreatorFactory.
                     newPreparedStatementCreator(forecastQuery),  new RowHandler(),
params );
            }catch(DataAccessException e){

              System.out.println("ERROR getting Forecast for :"+city);
            }*/
            s="Forecast for Jacksonville Florida";
            s+="Tomorrow : Clear and cold";
            s+="Wind: North east at 10 mph";
            s+="Temperature Hi/Low: 67F/50F";
          System.out.println("********* sending forecast ** length:"+s.length());
            return s;
        }

        public void updateObs(String key, String data){
```

```
            String [] params = new String [2];
            params[0] = data;
            params[1] = key;
          try{
            jdbc.update(PreparedStatementCreatorFactory.
                                newPreparedStatementCreator(updObsQuery), params);
          }catch(DataAccessException e){
            System.out.println("ERROR getting Forecast for :"+ key);
          }


        }

        public void updateForecast(String key, String data){

            String [] params = new String [2];
            params[0] = data;
            params[1] = key;
          try{
            jdbc.update(PreparedStatementCreatorFactory.
                                newPreparedStatementCreator(updForecastQuery),
params);
          }catch(DataAccessException e){

            System.out.println("ERROR getting Forecast for :"+ key);
          }


        }}
```

```
package mega.ws.exceptions;

public class InitializationException extends Exception{

        private Exception originalException = null;
        private static final String DEFAULT_MSG = "No additional message available";


        public InitializationException(String msg) {

                super(msg);
        }

    public InitializationException(String msg, Exception e) {

         super(msg);
         originalException = e;
    }

    public Exception getOriginalException(){
        return originalException == null?
        new InitializationException(""): originalException;
    }

    public String getOriginalMessage(){
        return originalException.getMessage() == null?
        DEFAULT_MSG : originalException.getMessage();
    }
}

package mega.ws.exceptions;

import java.sql.SQLException;

public class DataAccessException extends SQLException {

        /**
         * Constructor for DataAccessException.
         * @param s message
         */
        public DataAccessException(String s) {
                super(s);
        }


        public DataAccessException(String msg, String sql) {
                super(msg, sql);
        }

}
```

```
package mega.ws.jdbc;

/**
 * Object to represent a SQL parameter definition.
 * Parameters may be anonymous, in which case name is null.
 * However all parameters must define a SQL type constant
 * from java.sql.Types.
 */
public class SqlParameter {

	private String name;

	/** SQL type constant from java.sql.Types */
	private int type;


	/**
	 * Add a new anonymous parameter
	 */
	public SqlParameter(int type) {
		this(null, type);
	}

	public SqlParameter(String name, int type) {
		this.name = name;
		this.type = type;
	}

	public String getName() {
		return name;
	}

	public int getSqlType() {
		return type;
	}

}
```

```
package mega.ws.jdbc;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.SQLWarning;
import java.sql.Statement;
//import org.apache.log4j.Logger;
import mega.ws.core.WeatherDataSource;
import javax.sql.DataSource;

import mega.ws.exceptions.DataAccessException;
import mega.ws.core.RowCallbackHandler;
/**
 * Class that iterates over a ResultSet, handling any errors,
 * and calling back a helper on each line.
 * Provides a convenient way to retrieve data using JDBC.
 */
public class JdbcTemplate {


      //protected final Logger logger =
      Logger.getLogger(getClass().getName());
      private WeatherDataSource dataSource;


      /** Construct a new JDBCTemplate, given a connection factory
       * @param connectionFactory connection factory to use to obtain
connections from
       */
      public JdbcTemplate(DataSource dataSource) {
            this.dataSource =  (WeatherDataSource)dataSource;
      }

      /**
       * Return the DataSource used by this template
       * @return the DataSource used by this template
       */
      public DataSource getDataSource() {
            return dataSource;
      }

      /**
       * Execute a query given static SQL.
       * Doesn't use a prepared statement.
       * @param sql
       * @param callbackHandler
       * @throws SQLException
       * @throws NamingException
       */
      public void query(String sql , RowCallbackHandler
callbackHandler)
      throws DataAccessException {

            Connection con = null;
            PreparedStatement ps = null;
```

```
            ResultSet rs = null;
            try {
                  con = dataSource.getConnection();
                  //Statement sta = con.createStatement();
                  //rs = sta.executeQuery(sql);
                  ps = con.prepareStatement(sql);
                  rs = ps.executeQuery();
            //    logger.info("Executing static SQL query '" + sql +
"'");

                  while (rs.next()) {
                        callbackHandler.processRow(rs);
                  }

                  SQLWarning warning = ps.getWarnings();
                  rs.close();
                  ps.close();

            }
            catch (SQLException ex) {
                  throw new DataAccessException("JdbcTemplate Executing
query", sql);
            }
            finally {

                  dataSource.disconnect();
            }
      }     // query


      /**
       * Query using a prepared statement. Most other methods use
       * this.
       * @param psc
       * @param callbackHandler
       * @throws SQLException
       * @throws NamingException
       */
      public String query(PreparedStatementCreator psc,
RowCallbackHandler callbackHandler, String[] params)
      throws DataAccessException {

          JdbcTemplate jdbc;
          Connection con;
            Statement s = null;
            ResultSet rs = null;
            StringBuffer result=new StringBuffer();

            try {
                  con = dataSource.getConnection();
                  if(con != null){
                        PreparedStatement ps =
psc.createPreparedStatement(con);
                  //    logger.info("Executing SQL query using
PreparedStatement: [" + psc + "]");

                        for(int i=0; i < params.length ; i++){
```

```
                        if(params[i] != null)
                            ps.setString(i+1,params[i]);
                    }

                    rs = ps.executeQuery();

                    while (rs.next()) {

                //          logger.info("Processing row of
ResultSet");

      result.append(callbackHandler.processRow(rs));
                    }

                    SQLWarning warning = ps.getWarnings();
                    rs.close();
                    ps.close();
                }
            }
            catch (SQLException ex) {
                throw new
DataAccessException("JdbcTemplate.query(psc) with
PreparedStatementCreator [" + psc + "]", null);
            }
            finally {
                dataSource.disconnect();
            }
            return result.toString();
      }     // query


      /**
       * Method update.
       * @param sql
       * @return int
       * @throws SQLException
       * @throws NamingException
       */
      public int update(final String sql) throws DataAccessException {

            //logger.info("Running SQL update '" + sql + "'");
            return
update(PreparedStatementCreatorFactory.newPreparedStatementCreator(sql)
);
      }

      /**
       * Method update.
       * @param psc
       * @return int
       * @throws SQLException
       * @throws NamingException
       */
      public int update(PreparedStatementCreator psc) throws
DataAccessException {
            return update(new PreparedStatementCreator[] { psc })[0];
      }
```

```
	public void update(PreparedStatementCreator psc, String []
params) throws DataAccessException {

		Connection con = null;
		int index = 0;
		PreparedStatement ps = null;

		try {
			con = dataSource.getConnection();
			ps = psc.createPreparedStatement(con);

			for (index = 0; index < params.length; index++) {

				ps.setString(index+1, params[index]);
			}
			ps.executeUpdate();
			//logger.info("JDBCTemplate: update Completed");
			ps.close();
		}catch(SQLException e){}
	}
	/**
	 * Run multiple updates.
	 * @param pscs
	 * @return int[]
	 * @throws SQLException
	 * @throws NamingException
	 */
	public int[] update(PreparedStatementCreator[] pscs) throws
DataAccessException {
		Connection con = null;
		//Statement s = null;
		int index = 0;
		try {
			con = dataSource.getConnection();
			int[] retvals = new int[pscs.length];
			for (index = 0; index < retvals.length; index++) {
				PreparedStatement ps =
pscs[index].createPreparedStatement(con);
				retvals[index] = ps.executeUpdate();
				//logger.info("JDBCTemplate: update affected "
+ retvals[index] + " rows");
				ps.close();
			}
			// Don't worry about warnings, as we're more likely
to get exception on updates
			// (for example on data truncation)
			return retvals;
		}
		catch (SQLException ex) {
			throw new DataAccessException(ex.getMessage());
		}
		finally {
			dataSource.disconnect();
		}
	}	// update
} // class JdbcTemplate
```

```
package mega.ws.jdbc;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public interface PreparedStatementCreator {

      /** Create a statement in this connection. Allows
            * implementations to use PreparedStatements. Only invoked
            * if no SQL is passed into the ResultSetHandler.
            * The ResultSetHandler will close this statement.
            * @param conn Connection to use to create statement
            * @return a prepared statement
            */
      PreparedStatement createPreparedStatement(Connection conn)
      throws SQLException;

}
package mega.ws.jdbc;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Types;
import java.util.Arrays;
import java.util.Collections;
import java.util.LinkedList;
import java.util.List;

import mega.ws.exceptions.DataAccessException;

/**
 * Helper class that can efficiently create multiple
 * PreparedStatementCreator objects with different
 * parameters based on a SQL statement and a single set of parameter
declarations.
 **/
public class PreparedStatementCreatorFactory {

      /**
       * Convenient method to return a PreparedStatementCreator that
has no arguments
       */
      public static PreparedStatementCreator
newPreparedStatementCreator(final String sql) {
            return new PreparedStatementCreator() {
                  public PreparedStatement
createPreparedStatement(Connection conn) throws SQLException {
                        PreparedStatement ps =
conn.prepareStatement(sql);
                        return ps;
                  }
            };
      }
```

```
	/**
	 * Convenient method to declare variables and parameters in a
single operation.
	 * If issuing multiple statements with the same parameters,
construct an instance of
	 * PreparedStatementCreatorFactory to hold the parameters
instead.
	 */
	public static PreparedStatementCreator
newPreparedStatementCreator(String sql, int[] types, Object[] params) {
		PreparedStatementCreatorFactory pscf = new
PreparedStatementCreatorFactory(sql, types);
		return pscf.newPreparedStatementCreator(params);
	}

	/**
	 * Convert a list of JDBC types, as defined in the java.sql.Types
class,
	 * to a List of SqlParameter objects as used in this package
	 */
	public static List sqlTypesToAnonymousParameterList(int[] types)
{
		List l = new LinkedList();
		if (types != null) {
			for (int i = 0; i < types.length; i++) {
				l.add(new SqlParameter(types[i]));
			}
		}
		return l;
	}
	/**
	 * List of SqlParameter objects. May not be null.
	 */
	private List declaredParameters = new LinkedList();
	private String sql;

	/**
	 * Create a new factory. Will need to add parameters
	 * via the addParameter() method or have no parameters
	 */
	public PreparedStatementCreatorFactory(String sql) {
		this(sql, new LinkedList());
	}

	/**
	 * Create a new factory with sql and parameters with the given
JDBC types
	 * @param sql SQL to execute
	 * @param types int array of JDBC types
	 */
	public PreparedStatementCreatorFactory(String sql, int[] types) {
		this(sql, sqlTypesToAnonymousParameterList(types) );
	}

	/**
	 * Create a new factory with sql and the given parameters
	 * @param sql SQL
```

```
         * @param declaredParameters list of SqlParameter objects
         */
        public PreparedStatementCreatorFactory(String sql, List
declaredParameters) {
                this.sql = sql;
                this.declaredParameters = declaredParameters;
        }
        /**
         * Add a new declared parameter
         * Order of parameter addition is significant
         */
        public void addParameter(SqlParameter p) {
                declaredParameters.add(p);
        }
        /**
         * Return a new PreparedStatementCreator given these parameters
         * @param params parameter array. May be null.
         */
        public PreparedStatementCreator
newPreparedStatementCreator(Object[] params) {
                return new PreparedStatementCreatorImpl((params != null) ?
Arrays.asList(params) : new LinkedList());
        }

        /**
         * Return a new PreparedStatementCreator instance given this
parameters.
         * @param params List of parameters. May be null.
         */
        public PreparedStatementCreator newPreparedStatementCreator(List
params) {
                return new PreparedStatementCreatorImpl(params != null ?
params : new LinkedList());
        }
        /**
         * PreparedStatementCreator implementation returned by this class
         */
        private class PreparedStatementCreatorImpl implements
PreparedStatementCreator {
                private List parameters;

                /**
                 * @param params list of SqlParameter objects. May not be
null
                 */
                private PreparedStatementCreatorImpl(List params) {
                        this.parameters = params;

                }

                public PreparedStatement createPreparedStatement(Connection
conn)
                throws SQLException {
                        PreparedStatement ps = conn.prepareStatement(sql);

                        // Set arguments: does nothing if there are no
parameters
```

```
                for (int i = 0; i < parameters.size(); i++) {
                        SqlParameter declaredParameter = (SqlParameter)
PreparedStatementCreatorFactory.this.declaredParameters.get(i);
                        // We need SQL type to be able to set null
                        if (parameters.get(i) == null) {
                                ps.setNull(i + 1,
declaredParameter.getSqlType());
                        }
                        else {
                                switch (declaredParameter.getSqlType()) {
                                        case Types.VARCHAR :
                                                ps.setString(i + 1, (String)
parameters.get(i));
                                                break;
                                        //case Types. :
                                        //    ps.setString(i + 1, (String)
parameters.get(i));
                                        //    break;
                                        default :
                                                ps.setObject(i + 1,
parameters.get(i), declaredParameter.getSqlType());
                                                break;
                                }
                        }
                }
                return ps;
        }

        public String toString() {
                StringBuffer sbuf = new
StringBuffer("PreparedStatementCreatorFactory.PreparedStatementCreatorI
mpl: sql={" + sql + "}: params={");
                for (int i = 0; i < parameters.size(); i++) {
                        if (i > 0)
                                sbuf.append(",");
                        sbuf.append(parameters.get(i));
                }
                return sbuf.toString() + "}";
        }
    }

}
```

```
package mega.ws.jdbc;


import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import mega.ws.core.RowCallbackHandler;
/**
 * Convenient superclass for callback handlers.
*/
public class RowHandler implements RowCallbackHandler {

	private int rowCount;
	private int columnCount;

	/** Indexed from 0. Type (as in java.sql.Types) for the columns
	 * as returned by ResultSetMetaData object
	 */
	private int[] columnTypes;

	private String[] columnNames;

	/** Implementation of ResultSetCallbackHandler.
	 * Work out column size if this is the first row,
	 * otherwise just count rows.
	 * <br/>Subclasses can perform custom extraction or processing
	 * by overriding the processRow(ResultSet, int) method.
	 */
	public String processRow(ResultSet rs) throws SQLException {
		if (rowCount == 0) {
			ResultSetMetaData rsmd = rs.getMetaData();
			columnCount = rsmd.getColumnCount();
			columnTypes = new int[columnCount];
			columnNames = new String[columnCount];
			for (int i = 0; i < columnCount; i++) {
				columnTypes[i] = rsmd.getColumnType(i + 1);
				columnNames[i] = rsmd.getColumnName(i + 1);
			}
			// Could also get column names
		}
		return processRow(rs, rowCount++);
	}


	/** Subclasses may override this to perform custom extraction
	 * or processing. This class's implementation does nothing.
	 * @param rs ResultSet to extract data from. This method is
	 * invoked for each row
	 * @param rowNum number of the current row (starting from 0)
	 */
	protected String processRow(ResultSet rs, int rowNum) throws
SQLException {
		System.out.println("in processrow...");
		//System.out.println("default process row: value at 0 is "
+ rs.getObject(1));
		return rs.getString(1);
	}
```

```
        /** Return the types of the columns as java.sql.Types constants
         * Valid after processRow is invoked the first time.
         * @return the types of the columns as java.sql.Types constants.
         * <b>Indexed from 0 to n-1.</b>
         */
        public final int[] getColumnTypes() {
              return columnTypes;
        }

        /** Return the types of the columns as java.sql.Types constants
         * Valid after processRow is invoked the first time.
         * @return the types of the columns as java.sql.Types constants.
         * <b>Indexed from 0 to n-1.</b>
         */
        public final String[] getColumnNames() {
              return columnNames;
        }

        /** Return the row count of this ResultSet
         * Only valid after processing is complete
         * @return the number of rows in this ResultSet
         */
        public final int getRowCount() {
              return rowCount;
        }


        /** Return the number of columns in this result set.
         * Valid once we've seen the first row,
         * so subclasses can use it during processing
         * @return the number of columns in this result set
         */
        public final int getColumnCount() {
              return columnCount;
        }

}       // class RowCountCallbackHandler
```

```
package mega.ws.util;


public class Constants {

      public static final String dbUrl = "dbUrl";
      public static final String dbUser = "dbUser";
      public static final String dbPassword = "dbPassword";
      public static final String dbName = "dbName";
      public static final String DEFAULT_DIR =
"C:\\Weather\\weather\\webApplication\\admin";
      public static final String CONFIG_FILE = "weather.config";
      public static final String jdbcDriver = "jdbcDriver";
      public static final String dbType = "dbType";
      public static final String MAIN_EVENT_DESC = "Initial entry point
of application.";
      public static final String GENERIC_EVENT_DESC = "Generic Event";
    public static final String
DEFAULT_MAP="weatherData/currentWxMap.jpg";
    public static final String DEFAULT_CONTENT
="weatherData/currentWxDesc.html";
      public static final String FILE_SEPARATOR =
System.getProperty("file.separator");
      public static final String END_POINT_URL =

      "http://192.168.0.164:8080/axis/Mega/MegavegaWx.jws";

      public static final String XSD_STRING = "XMLType.XSD_STRING";

      public static final String IN = "ParameterMode.IN";
      public static final String OUT = "ParameterMode.OUT";
      public static final String DATA_DIR ="C:\\weatherData";
      public static final String OB_DIR = "obs";
    public static final String FORECAST_DIR = "forecasts";
      public static final String SAT_DIR = "sat";
      public static final String RADAR_DIR = "radar";
}
```

```
package mega.ws.util;

import java.io.*;
import java.util.Properties;
import java.util.Map;
import java.util.StringTokenizer;
import mega.ws.jdbc.*;


public class FileHandler {

	private static final int PAIR = 2;
	private String directory = null;
	private StringBuffer fullFileName = null;
	private String fileSeparator =
System.getProperty("file.separator");

	public FileHandler(String dir){
		directory = dir;

	}

	public Properties readAppConfig(String filename)
	throws java.io.IOException, FileNotFoundException {

		Properties props = new Properties();
		//hardcode(props);

		return props;
	}

	public Properties readConfiguration(String filename)
	throws java.io.IOException, FileNotFoundException {

		Properties p = new Properties();

		BufferedReader	in =
			new BufferedReader(new
FileReader(buildPath(filename)));
		StringTokenizer st;
		String line = null;


		while( (line = in.readLine()) != null){
			try{
				st = new StringTokenizer(line, "=");
				if(st.countTokens() == PAIR){ //ok to read
ignore what is not set.
					p.put(st.nextToken(), st.nextToken());
				}
			}catch(NullPointerException e){
				throw new IOException("Error during processing
of config file");
			}
		}

		return p;
```

```
	}

	private String buildPath(String filename){

		StringBuffer buff = new StringBuffer(directory);
		buff.append(fileSeparator);
		buff.append(filename);

		return buff.toString();
	}

    public byte[] readFile(String filename){

System.out.println("reading file "+filename);
		byte [] buffer = null;
		try{

		File file = new File(filename);
		BufferedInputStream bis = new BufferedInputStream(new

FileInputStream(file));
		int bytes = (int) file.length();
		buffer = new byte[bytes];
		int readBytes = bis.read(buffer);
		bis.close();
System.out.println("Read " + readBytes + " and expected to read " +
bytes);
		}catch(FileNotFoundException e){
			System.out.println("ERROR reading image
file"+e.getMessage());
		}catch(IOException e){
			System.out.println("ERROR reading image file"+
e.getMessage());
		}

		return buffer;
	}

	public int writeImage(String imagepath, byte [] buffer){


		try{

		FileOutputStream os = new
FileOutputStream("C:\\dataTest\\img.jpg");
		os.write(buffer);
		os.close();

		System.out.println("Wrote " + buffer.length);
		}catch(FileNotFoundException e){
			System.out.println("ERROR reading image
file"+e.getMessage());
		}catch(IOException e){
			System.out.println("ERROR reading image file"+
e.getMessage());
		}
```

```
            return buffer.length;
      }

      public void writeFile(String filename, String data){

                  try{

                        BufferedWriter out = new BufferedWriter(new
FileWriter(filename));
                        out.write(data);
                        out.close();

                  }catch(FileNotFoundException e){
                        System.out.println("ERROR reading image
file"+e.getMessage());
                  }catch(IOException e){
                        System.out.println("ERROR reading image file"+
e.getMessage());
                  }

      }


}
```

```
package mega.ws.util;

import java.util.Properties;
import java.util.ArrayList;
import java.util.*;
import com.enterprisedt.net.ftp.*;

public class FtpDownloader {

	public static String host="tgftp.nws.noaa.gov";
	public static String OB_DIR =
"/data/observations/metar/decoded/";
	public static String FORECAST_DIR = "/data/forecasts/city/";
	public static String CHART_DIR = "/data/forecasts/city/";

	public static int PORT = 21;
	public static String username = "Anonymous";
	public static String password = "hola";
	public com.enterprisedt.net.ftp.FTPClient ftp;
	public boolean connected;


	public FtpDownloader(Properties p){

		if(p != null){
			host = (String) p.get("host");
			username = (String) p.get("username");
			password = (String) p.get("password");
		}

	}

	public void connect(){
		if(connected){
			return;
		}
		try{
			ftp = new FTPClient(host,PORT);
			ftp.login("Anonymous", "");
			connected = true;
		}catch(FTPException e){
			System.out.println("got an FTP exception");
			connected =false;
		}catch(java.io.IOException e){
			System.out.println("got an io exception");
			connected =false;
		}
	}

	public void disconnect(){

			try{
				ftp.quit();
			}catch(FTPException e){
				System.out.println("got an FTP exception");
				connected =false;
			}catch(java.io.IOException e){
```

```
                        System.out.println("got an io exception");
                        connected =false;
                }

        connected=false;

    }

    public String getOb(String obfile){


        return getFile(OB_DIR + obfile.toUpperCase());
    }

    private String getFile(String fullpath){

        String data = null;
        try{
            connect();
            data = new String(ftp.get(fullpath));
        }catch(FTPException e){
                    System.out.println("got an FTP exception");
                    connected =false;
        }catch(java.io.IOException e){
                    System.out.println("got an IO exception");
                    connected =false;
        }

        return data;
    }
    /**
     * forecast path must include the state
/data/forecasts/city/fl/jacksonville.txt
     */
    public String getForecast(String forecast){

        return getFile(FORECAST_DIR + forecast.toLowerCase());
    }

    public String getCharts(String chart){

        return getFile(CHART_DIR + chart.toUpperCase());
    }
    /**
     * Must receive a list of clean icaos with the respective NWS
call .TXT
     */
    public HashMap getObList(HashMap list){


        for(Iterator i = list.keySet().iterator(); i.hasNext();){
            String t = (String) i.next(); //this is the real icao
            list.put(t, getOb((String)list.get(t)));
        }

        this.disconnect();
        return list;
```

```
	}

	public HashMap getForecasts(HashMap list){

		for(Iterator i = list.keySet().iterator(); i.hasNext();){
			String t = (String) i.next();
			list.put(t, getForecast((String)list.get(t)));
		}
		this.disconnect();
		return list;
	}

	public Map getCharts(HashSet list){

		HashMap map = new HashMap();

		for(Iterator i = list.iterator(); i.hasNext();){
			String t = (String) i.next();
			map.put(t, getCharts(t));

		}

		return map;
	}
}
```

Appendix D

Java Web Application Source Code

```
package vega.weather.beans;


import java.util.*;
import vega.weather.core.CacheManager;
import vega.weather.interfaces.DataBean;



public abstract class WeatherBean implements DataBean {

	private ArrayList allStates = null;
	private String  state  = null;

	private String  city   = null;
	private ArrayList stateCities = null;

	private ArrayList regions = null;
	private String region = null;

	private ArrayList products = null;
	private String product = null;

	private Date date = null;



	public WeatherBean(){
		fillCacheData();
	}

	public abstract String getClassName();

	public abstract String getContent();

	public abstract String getTopLink();


	private void fillCacheData(){

		CacheManager cm = CacheManager.getInstance();
		allStates = cm.getStates();
		regions = cm.getRegions();
		products = cm.getProducts();

	}

	/**
```

```
 * Gets the allStates
 * @return Returns a ArrayList
 */
public ArrayList getAllStates() {
	return allStates;
}
/**
 * Sets the allStates
 * @param allStates The allStates to set
 */
public void setAllStates(ArrayList allStates) {
	this.allStates = allStates;
}


/**
 * Gets the selectedState
 * @return Returns a String
 */
public String getState() {
	return state;
}
/**
 * Sets the selectedState
 * @param selectedState The selectedState to set
 */
public void setState(String selectedState) {
	state = selectedState;
}


/**
 * Gets the selectedCity
 * @return Returns a String
 */
public String getCity() {
	return city;
}
/**
 * Sets the selectedCity
 * @param selectedCity The selectedCity to set
 */
public void setCity(String selectedCity) {
	city = selectedCity;
}


/**
 * Gets the stateCities
 * @return Returns a ArrayList
 */
public ArrayList getStateCities() {
	return stateCities;
}
/**
 * Sets the stateCities
 * @param stateCities The stateCities to set
```

```
 */
public void setStateCities(ArrayList stateCities) {
	this.stateCities = stateCities;
}


/**
 * Gets the regions
 * @return Returns a ArrayList
 */
public ArrayList getRegions() {
	return regions;
}
/**
 * Sets the regions
 * @param regions The regions to set
 */
public void setRegions(ArrayList regions) {
	this.regions = regions;
}


/**
 * Gets the selectedRegion
 * @return Returns a String
 */
public String getRegion() {
	return region;
}
/**
 * Sets the selectedRegion
 * @param selectedRegion The selectedRegion to set
 */
public void setRegion(String selectedRegion) {
	region = selectedRegion;
}


/**
 * Gets the products
 * @return Returns a ArrayList
 */
public ArrayList getProducts() {
	return products;
}
/**
 * Sets the products
 * @param products The products to set
 */
public void setProducts(ArrayList products) {
	this.products = products;
}


/**
 * Gets the selectedProduct
 * @return Returns a String
```

```
	 */
	public String getProduct() {
		return product;
	}
	/**
	 * Sets the selectedProduct
	 * @param selectedProduct The selectedProduct to set
	 */
	public void setProduct(String selectedProduct) {
		product = selectedProduct;
	}


	/**
	 * Gets the date
	 * @return Returns a Date
	 */
	public Date getDate() {
		return date;
	}
	/**
	 * Sets the date
	 * @param date The date to set
	 */
	public void setDate(Date date) {
		this.date = date;
	}


}
```

```
package vega.weather.beans;

import vega.weather.interfaces.DataBean;
import java.util.Date;


public class AdminBean implements DataBean {

	private String summary = null;
	private String imagePath = null;
	private String content = null;
	private Date date = null;

	public AdminBean(){}
	public String getClassName(){
		return this.getClass().getName();
	}
	public String getContent(){
		return content;
	}
	/**
	 * Convenient method to get text to update the daily
	 * National Summary.
	 */
	public void setSummary(String summary){
		this.summary = summary;
	}
	public String getSummary(){
		return summary;
	}
	public String getTopLink(){
		return "MegaVega Weather Administration";
	}
	/**
	 * Method to update the file path for the national image
	 */
	public void setImagePath(String path){
		imagePath = path;
	}
	public String getImagePath(){
		return imagePath;
	}
	/**
	 * Gets the date
	 * @return Returns a Date
	 */
	public Date getDate() {
		return date;
	}
	/**
	 * Sets the date
	 * @param date The date to set
	 */
	public void setDate(Date date) {
		this.date = date;
	}
```

```
}
package vega.weather.beans;

import java.util.*;
import vega.weather.core.CacheManager;
public class CitywxBean extends WeatherBean {
    private String content = null;
    private String forecast = null;
    private String ob = null;
       // new StringBuffer("Today partly cloudy, winds north at 15
miles per hour, no chance for showers.");

      public CitywxBean(){
            super();
            //forecast.append("Highs in the mid 70's, lows tonight in
the low 50's.");
      }

      public String getClassName(){
            return getClass().getName();
      }

      public String getContent(){
            return content;
      }

    public String getForecast(){
          return forecast.toString();
    }
      public void setForecast(String forecast){
            this.forecast = forecast;
      }
      /**
       * Sets the content
       * @param content The content to set
       */
      public void setContent(String content) {
            this.content = content;
      }
      public String getTopLink(){
            return "Current Weather and Forecast for " + getCity();
      }
      /**
       * Gets the ob
       * @return Returns a String
       */
      public String getOb() {
            return ob;
      }
      /**
       * Sets the ob
       * @param ob The ob to set
       */
      public void setOb(String ob) {
            this.ob = ob;
      }
```

```
package vega.weather.beans;

import java.util.*;
import vega.weather.core.CacheManager;
import vega.weather.util.Constants;
public class MainPageBean extends WeatherBean {
	private String content = Constants.DEFAULT_CONTENT;
	private String natlMap = Constants.DEFAULT_MAP;
	public MainPageBean(){
		super();
	  }
	public void setState(String state){
		super.setState(state);

	super.setStateCities(CacheManager.getInstance().getCities(state))
	}
	public String getClassName(){
		return getClass().getName();
	}
	public String getContent(){
		return content;
	}
	/**
	 * Sets the content
	 * @param content The content to set
	 */
	public void setContent(String content) {
		this.content = content;
	}
	public String getTopLink(){
		return "Current US Conditions";
	}

	/**
	 * Gets the natlMap
	 * @return Returns a String
	 */
	public String getNatlMap() {
		return natlMap;
	}
	/**
	 * Sets the natlMap
	 * @param natlMap The natlMap to set
	 */
	public void setNatlMap(String natlMap) {
		this.natlMap = natlMap;
	}


}
```

```
package vega.weather.beans;

import java.util.*;
public class RadarBean extends WeatherBean {

	private String content = null;
	private String imagePath =null;


	public RadarBean(){
		super();
	}

	public String getClassName(){
		return getClass().getName();
	}

	public String getContent(){
		return content;
	}

	/**
	 * Sets the content
	 * @param content The content to set
	 */
	public void setContent(String content) {
		this.content = content;
	}

	public String getTopLink(){
		return "Doppler Radar ";
	}
	/**
	 * Gets the imagePath
	 * @return Returns a String
	 */
	public String getImagePath() {
		return imagePath;
	}
	/**
	 * Sets the imagePath
	 * @param imagePath The imagePath to set
	 */
	public void setImagePath(String imagePath) {
		this.imagePath = imagePath;
	}


}
```

```
package vega.weather.beans;

import java.util.*;

public class SatelliteBean extends WeatherBean {

	private String content = null;
	private String imagePath =null;

	public SatelliteBean(){
		super();
	}

	public String getClassName(){
		return getClass().getName();
	}

	public String getContent(){
		return content;
	}

	/**
	 * Sets the content
	 * @param content The content to set
	 */
	public void setContent(String content) {
		this.content = content;
	}

	public String getTopLink(){
		return "Satellite Products";
	}
	/**
	 * Gets the imagePath
	 * @return Returns a String
	 */
	public String getImagePath() {
		return imagePath;
	}
	/**
	 * Sets the imagePath
	 * @param imagePath The imagePath to set
	 */

	public void setImagePath(String imagePath) {
		this.imagePath = imagePath;
	}


}
```

```
 package vega.weather.core;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import java.rmi.RemoteException;
import javax.xml.rpc.ParameterMode;
import javax.xml.rpc.ServiceException;
import java.net.MalformedURLException;
import vega.weather.util.*;
import vega.weather.exceptions.InitializationException;
import vega.weather.ws.*;

public class WxConnection {


     private static final String GET_WX_FORECAST = "getWxForecast";
     private static final String GET_WX_OB ="getOb";

     private static final String GET_WX_IMAGE = "getWxImage";
     //private Logger logger = Logger.getLogger(WxConnection.class);
     private String wsURL = null;

     private MegavegaWxService service = null;
     private MegavegaWx wx = null;

      public WxConnection(){

            try{
                  //wsURL = (String)ConfigurationHandler.getInstance().
                   //          getConfigProperty("webserviceURL");

                  service = new MegavegaWxServiceLocator();
                  wx = service.getWeatherService();

            }catch(ServiceException e){
                  System.out.println("Fatal ERROR: Service Exception");
            }

     }

     public String getForecast(String city, String time)
     throws RemoteException {


            /*Call call = getCall(wsURL);
            call.setOperationName(this.GET_WX_FORECAST);
            call.addParameter( "icao", XMLType.XSD_STRING,
ParameterMode.IN);
            call.setReturnType( XMLType.XSD_STRING );
            return (String) call.invoke( new Object [] { city });*/
            return wx.getWxForecast(city);
     }

     public String getWx(String city, String time)
     throws RemoteException {
```

```
            /* Call call = getCall(wsURL);
                  call.setOperationName(this.GET_WX_OB);
            call.addParameter( "icao", XMLType.XSD_STRING,
ParameterMode.IN);
            call.setReturnType( XMLType.XSD_STRING );
            return (String) call.invoke( new Object [] { city }); */
            return wx.getOb(city);
      }
      private Call getCall(String endpoint){

            Call call = null;
            try{
                  Service service = new Service();
                  call = (Call) service.createCall();
                  call.setTargetEndpointAddress( new
java.net.URL(endpoint) );
            }catch(MalformedURLException e){
            //    logger.error("ERROR getting Call Object:", e );
            }
            catch(ServiceException e){
            //    logger.error("ERROR getting Call Object:", e );
            }
            return call;
      }

      /**
       * Method to retrieve the image and write it to a file where it
       * can be retrieved for display.
       * @ param type: is the type of image currently supporting
Satellite
       *                and doppler.
       * @ param region is the region the image covers.
       * @ param time: may be used in future to retrieve other images.
       * @ byte [] result holds the image data
       * @return flags if image was captured successfully.
       * */

      public String getImage(String type, String region, String name)
      throws java.rmi.RemoteException{

            byte [] image = wx.getWxImage(type,region);
            //no specific directory
            FileHandler h = new FileHandler(null);
            System.out.println("****** image received
size:"+image.length);
            return h.writeFile(type, region, name, image);
      }

}
```

```
package vega.weather.core;

import java.util.*;
import vega.weather.util.FileHandler;


public class CacheManager {

	private HashMap cities;
	private ArrayList states  ;
	private ArrayList regions ;
	private ArrayList products ;
	private boolean isInitialized;
	private static CacheManager cm = new CacheManager();


	private CacheManager(){

		if(!isInitialized){
			init();
		}
	}
	//private void CacheManager(){}
	private void init(){

		initCities();
		initStates();
		initProducts();
		initRegions();

		isInitialized = true;
	}

	public static CacheManager getInstance(){
		return cm;
	}

	private void initCities(){

		cities = new HashMap();
		//Florida
		cities.put("Jacksonville","Florida");
		cities.put("Miami","Florida");
		cities.put("Orlando","Florida");
		cities.put("Tampa","Florida");
		cities.put("Key West","Florida");
		cities.put("Talahassee","Florida");
		cities.put("Melbourne","Florida");
		cities.put("Pensacola","Florida");
		cities.put("Ft. Lauderdale","Florida");
		//Georgia
		cities.put("Atlanta","Georgia");
		cities.put("Macon","Georgia");
		cities.put("Valdosta","Georgia");
		cities.put("Savanah","Georgia");
		//Alabama
		cities.put("Birmingham","Alabama");
```

```
        cities.put("Mobile","Alabama");
        cities.put("Guien","Alabama");
        cities.put("More","Alabama");

        //Maine
        cities.put("Bangor","Maine");
        cities.put("Portland","Maine");
        cities.put("Booth Bay Harbor","Maine");
        cities.put("Brunswick","Maine");
}

private void initStates(){

        states = new ArrayList();

        states.add("Florida");
        states.add("Georgia");
        states.add("Alabama");
        states.add("Maine");
}

private void initProducts(){

        products = new ArrayList();

        products.add("Satellite");
        products.add("Radar");

}

private void initRegions(){

        regions = new ArrayList();

        regions.add("Continental US");
        regions.add("South East");
        regions.add("North East");
        regions.add("Great Plains");
        regions.add("North West");
        regions.add("South West");
}


/**
 * Gets the cities
 * @return Returns a HashMap
 */
public ArrayList getCities(String state) {

        ArrayList list = new ArrayList();

        if(state ==null){
                return list;
        }

        for(Iterator i = cities.keySet().iterator(); i.hasNext();){
                String key = (String) i.next();
```

```
                String value = (String) cities.get(key);

                if(state.equals(value)){
                      list.add(key);
                }
          }
          return list;
    }

    /**
     * Gets the states
     * @return Returns a HashMap
     */
    public ArrayList getStates() {
          return states;
    }

    /**
     * Gets the regions
     * @return Returns a HashSet
     */
    public ArrayList getRegions() {
          return regions;
    }

    /**
     * Gets the products
     * @return Returns a HashSet
     */
    public ArrayList getProducts() {
          return products;
    }



    public boolean isInitialized(){
          return isInitialized;
    }

    public String getStateCode(String city){
          return (String) cities.get(city);
    }


}
```

```
package vega.weather.core;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;
import vega.weather.exceptions.InitializationException;
import vega.weather.util.FileHandler;
import vega.weather.events.*;
import vega.weather.interfaces.*;
import vega.weather.beans.*;

public class ConfigurationHandler
{
      private HashMap configProps;
      private HashMap targets;
    private boolean initialized;
    private static ConfigurationHandler instance = new
ConfigurationHandler();

    public static ConfigurationHandler getInstance()
    {
        return instance;
    }

    private ConfigurationHandler()
    {
        configProps = null;
            targets = new HashMap();
            initialize(null,null);
    }

    public void initialize(String directory, String file)
    {
        try
        {
            if(directory == null || file == null)

loadConfiguration("C:\\Weather\\weather\\webApplication\\admin",
"weather.config");
            else
                loadConfiguration(directory, file);
        }
        catch(FileNotFoundException nf)
        {
           // throw new InitializationException("Configuration File not
found", nf);
        }
        catch(IOException e)
        {
          //  throw new InitializationException("Unable to read config
file", e);
        }
        if(!configProps.isEmpty())
            initialized = true;
        //else
```

```
            // throw new InitializationException("Configuration Init
Failed");
    }

    public void loadConfiguration(String directory, String file)
        throws FileNotFoundException, IOException
    {
        FileHandler handler = new FileHandler(directory);
        configProps = handler.readConfiguration(file);
        hardcode(configProps);
    }

    public String getConfigProperty(String key)
    throws InitializationException{
        return (String)configProps.get(key);
    }

    public EventHandler getEventHandler(String event){
      return (EventHandler)configProps.get(event);
    }

    public void setConfigProperty(String key, String value)
    {
        configProps.put(key, value);
    }

    public void addToConfiguration(HashMap p)
    {
        configProps.putAll(p);
    }

    public void removeFromConfig(String key)
    {
        configProps.remove(key);
    }

    public boolean containsProperty(String key)
    {
        return configProps.containsKey(key);
    }

    private void hardcode(HashMap p)
    {
       /* try
      {

      for(Iterator i=p.keySet().iterator(); i.hasNext();){

            String key = i.next().toString();
            String fullClassName = null;
            String eventName = null;
            String eventView = null;
            String beanClass = null;

            int index;
```

```
            if((index = key.indexOf("_event")) != -1){ //event
parameter found
                  fullClassName = (String) p.get(key);
                  if(fullClassName == null){//load the event class

                        System.out.println("FATAL ERROR Events are not
initialized");
                        return;
                  }
                  else{
                        Event event =
(Event)getClass().getClassLoader().

      loadClass(fullClassName).newInstance();
                        eventName = key.substring(0,index);
                        event.setName(eventName);

      event.setViewName((String)configProps.get((eventName+"_target")))
;
                        beanClass =
(String)configProps.get((eventName+"_bean"));
                        DataBean bean = (DataBean)
getClass().getClassLoader().

      loadClass(beanClass).newInstance();
                        event.setBean(bean);

                        loadEventHandler(event);

                  }
            }


      }
      }catch(ClassNotFoundException e){

      }
      catch(IllegalAccessException e){

      }
      catch(InstantiationException e){
            e.printStackTrace();
      }*/
        vega.weather.events.MainEvent me =
EventFactory.createMainEvent("main", "jsp/MainPage.jsp", (DataBean)new
MainPageBean());
        vega.weather.events.CitywxEvent ce =
EventFactory.createCityWxEvent("citywx", "jsp/CityWx.jsp",
(DataBean)new CitywxBean());
        vega.weather.events.SatelliteEvent se =
EventFactory.createSatelliteEvent("satellite", "jsp/Satellite.jsp",
(DataBean)new SatelliteBean());
        vega.weather.events.RadarEvent re =
EventFactory.createRadarEvent("radar", "jsp/Radar.jsp", (DataBean)new
RadarBean());
```

```
        vega.weather.events.AdminEvent ae =
EventFactory.createAdminEvent("admin", "jsp/Admin.jsp", (DataBean)new
AdminBean());
        targets.put("main", new GenericEventHandler(me));
        targets.put("citywx", new GenericEventHandler(ce));
        targets.put("satellite", new GenericEventHandler(se));
        targets.put("radar", new GenericEventHandler(re));
        targets.put("admin", new GenericEventHandler(ae));
    }

    private void loadEventHandler(Event event)
    throws ClassNotFoundException, IllegalAccessException,
    InstantiationException{


      //try to get the event specific handler
      String handlerName =
(String)configProps.get(event.getName()+"_handler");
      //no event handler given in config file use default
      if(handlerName == null){
            handlerName = (String)configProps.get("default_handler");
      }
      if(handlerName != null){

            EventHandler handler =
(EventHandler)getClass().getClassLoader().

      loadClass(handlerName).newInstance();

            handler.setEvent(event);
            //add this event to the config properties
            targets.put(event.getName(), handler);
      }
      // this is a severe application error no default handler found.
      // should throw application error
      else{
            System.out.println("Fatal ERROR no default handler
specified");
      }

    }

      public HashMap getEventMaps(){

              return targets;
      }
}
```

```
package vega.weather.core;

import java.io.PrintStream;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletRequest;
import javax.servlet.http.*;
import vega.weather.beans.*;
import vega.weather.interfaces.Event;
import vega.weather.interfaces.EventHandler;

// Referenced classes of package vega.weather.core:
//            Product, CacheManager

public class GenericEventHandler
implements EventHandler
{

      private Event event;
      private static final String IMG_NAME = "current.jpg";
        Product product = new Product();


    public GenericEventHandler(Event event)
    {
        this.event = event;
    }
    //must implement a no args constructor for dynamic class loading
    public GenericEventHandler(){

    }

    public Event getEvent()
    {
        return event;
    }

    public void setEvent(Event e)
    {
        event = e;
    }

    public void getWxProduct()
    {
    }

    public void setBeanData()
    {
    }

    public void forward(HttpServletRequest req, HttpServletResponse
res)
    {
        System.out.println("Forwarding to " + event.getViewName());
        try
        {
           String view = event.getViewName();
           req.getRequestDispatcher(view).forward(req, res);
```

```
        }
        catch(Exception e)
        {
            System.out.println("GenericEventHandler ERROR:" +
e.getMessage());
        }
    }

    public void doProcess(HttpServletRequest req, HttpServletResponse
res)

    {
        vega.weather.interfaces.DataBean bean = event.getBean();
        String eventname = event.getName();
        StringBuffer content = new StringBuffer();
        if("citywx".equals(eventname))
        {
            CitywxBean citybean = (CitywxBean)bean;
            citybean.setCity(req.getParameter("city"));
            citybean.setState(req.getParameter("state"));

citybean.setStateCities(CacheManager.getInstance().getCities(citybean.g
etState()));
            citybean.setOb(product.getCityWx(citybean.getCity(),
IMG_NAME));
           citybean.setForecast(product.getForecast(citybean.getCity(),
"current"));
            req.getSession().setAttribute("citybean", citybean);
        } else
        if("satellite".equals(eventname))
        {
            SatelliteBean satBean = (SatelliteBean)bean;
            satBean.setProduct(req.getParameter("product"));
            satBean.setRegion(req.getParameter("region"));
            satBean.setCity(req.getParameter("city"));

         satBean.setImagePath(product.getSatellite(satBean.getRegion(),
"current"));
            req.getSession().setAttribute("satbean", satBean);
        } else
        if("radar".equals(eventname))
        {
            RadarBean radarBean = (RadarBean)bean;
            radarBean.setProduct(req.getParameter("product"));
            radarBean.setRegion(req.getParameter("region"));
            radarBean.setCity(req.getParameter("city"));
         radarBean.setImagePath(product.getRadar(radarBean.getRegion(),
"current"));
            req.getSession().setAttribute("radarbean", radarBean);
        } else
        {
            MainPageBean mainBean = (MainPageBean)bean;
            mainBean.setContent(product.getSummary("current"));
            mainBean.setNatlMap(product.getNatlMap("current"));
        }
        forward(req, res);
    }
```

```
}
package vega.weather.core;

import java.io.PrintStream;
import java.rmi.RemoteException;


// Referenced classes of package vega.weather.core:
//            WxConnection

public class Product
{
      private WxConnection wscon;
    //private Logger logger;

    public Product()
    {
        //logger = Logger.getLogger(vega.weather.core.Product.class);
        wscon = new WxConnection();
    }

    public String getCityWx(String city, String time)
    {

        String result = null;
        try
        {
            result = wscon.getWx(city, time);
            System.out.println("Got an ob for:"+city);
        }
        catch(RemoteException re)
        {
           // logger.error("Error getting Web service update");
        }
        return result;
    }

    public String getForecast(String city, String time)
    {
        String result = null;
        try
        {
            result = wscon.getForecast(city, time);
            System.out.println("Got a forecast for:"+city);
        }
        catch(RemoteException re)
        {
            //logger.error("Error getting Web service update");
        }

        return result;
    }

      /**
       * returns the virtual path of the image
       * The image_base parameter specified in the weather.config
```

```
     * Images need to be rendered by the web server.
     */
    public String getSatellite(String region, String name)
    {
        String path = null;
        try
        {
             path = wscon.getImage("SAT", region, name);

        }
        catch(RemoteException re)
        {
//           logger.error("Error getting Web service image update");
        }
        //return s;
       return path;
    }

    public String getRadar(String region, String name)
    {
        String path = null;
        try
        {
            path = wscon.getImage("RAD", region, name);
        }
        catch(RemoteException re)
        {
 //         logger.error("Error getting Web service image update");
        }
        //return s;
       return path;
    }

    public String getSummary(String time)
    {
        return "On the Weather map a cold front is developing...";
    }

    public String getNatlMap(String time)
    {
        return "c:\\natl12z.gif";
    }


}
```

```
package vega.weather.core;

import java.io.IOException;
import java.io.PrintStream;
import java.util.Hashtable;
import java.util.HashMap;
import vega.weather.exceptions.InitializationException;
import vega.weather.interfaces.EventHandler;

import vega.weather.util.FileHandler;

public class Redirector
{
    private HashMap targets;
    private static Redirector instance = new Redirector();

    private Redirector()
    {
      try{
            initialize();
      }catch(InitializationException e){
            System.out.println("Redirector failed to intialize");
      }
    }
    public static Redirector getInstance(){

        return instance;
    }

    public void initialize()
    throws InitializationException
    {
            targets =
ConfigurationHandler.getInstance().getEventMaps();
    }
    private void initialize(String directory, String configFile)
    throws InitializationException
    {
           // FileHandler handler = new FileHandler(directory);
            //targets = handler.readAppConfig(configFile);
            targets =
ConfigurationHandler.getInstance().getEventMaps();

    }
    public EventHandler getTarget(String key)
    {
        key = key.toLowerCase();

       // System.out.println("target is: " +
targets.get(key).toString());
        return (EventHandler)targets.get(key);
    }
    public void setTarget(String key, Object property)
    {
        targets.put(key, property);
    }
}
```

```
package vega.weather.events;

import vega.weather.interfaces.*;

public class SatelliteEvent extends GenericEvent {


	public SatelliteEvent(String name, String view, DataBean bean){
		super(name, view, bean);

	}

	public SatelliteEvent(){

	}



}
package vega.weather.events;

import vega.weather.interfaces.*;

public class AdminEvent extends GenericEvent {


	public AdminEvent(String name, String view, DataBean bean){
		super(name, view, bean);

	}

	public AdminEvent(){

	}



}
package vega.weather.events;

import vega.weather.interfaces.*;

public class CitywxEvent extends GenericEvent {


	public CitywxEvent(String name, String view, DataBean bean){
		super(name, view, bean);

	}

	public CitywxEvent(){
	}

}
```

```
package vega.weather.events;

import vega.weather.interfaces.*;
import vega.weather.events.*;

public class EventFactory {


      public static MainEvent createMainEvent(String n, String v,
DataBean b) {
            return new MainEvent(n,v,b) ;

      }

      public static CitywxEvent createCityWxEvent(String n, String v,
DataBean b) {
            return new CitywxEvent(n,v,b) ;

      }

      public static SatelliteEvent createSatelliteEvent(String n,
String v, DataBean b) {
            return new SatelliteEvent(n,v,b) ;

      }

      public static RadarEvent createRadarEvent(String n, String v,
DataBean b) {
            return new RadarEvent(n,v,b) ;

      }

      public static AdminEvent createAdminEvent(String n, String v,
DataBean b) {
            return new AdminEvent(n,v,b) ;

      }




}

package vega.weather.events;

import vega.weather.interfaces.*;
import vega.weather.util.Constants;

import vega.weather.beans.MainPageBean;

public class GenericEvent implements Event {

      private String name = null;
      private String viewName = null;
      //private EventHandler handler;
      private String description = Constants.GENERIC_EVENT_DESC;
      private DataBean bean;
```

```
	public GenericEvent(String name, String view, DataBean b){
		this.name = name;
		viewName = view;
	//	handler = h;
		bean = b;
	}

	public GenericEvent(){

	}

	public void setName(String name){
		this.name = name;
	}

	public void setViewName(String view){
		viewName = view;
	}

	public void setBean(DataBean bean){
		this.bean = bean;
	}

	public String getName(){
		return name;
	}

	public DataBean getBean(){
		return bean;
	}

	public String getViewName(){
		return viewName;
	}

	/*public EventHandler getHandler(){
		return handler;
	}*/

	public Class getBeanClass(){
		return bean.getClass();
	}

	public  String getDescription(){
		return description;
	}



}
```

```
package vega.weather.events;

import vega.weather.interfaces.*;

public class MainEvent extends GenericEvent {


	public MainEvent(String name, String view, DataBean bean){
		super(name, view, bean);

	}

	public MainEvent(){
	}



}

package vega.weather.events;

import vega.weather.interfaces.*;

public class RadarEvent extends GenericEvent {


	public RadarEvent(String name, String view, DataBean bean){
		super(name, view, bean);

	}

	public RadarEvent(){}

}
```

```
package vega.weather.exceptions;

public class InitializationException extends Exception{

      private Exception originalException = null;
      private static final String DEFAULT_MSG = "No additional message
available";


      public InitializationException(String msg) {

            super(msg);
      }

    public InitializationException(String msg, Exception e) {

        super(msg);
        originalException = e;
    }

    public Exception getOriginalException(){
      return originalException == null?
      new InitializationException(""): originalException;
    }

    public String getOriginalMessage(){
      return originalException.getMessage() == null?
      DEFAULT_MSG : originalException.getMessage();
    }
}

package vega.weather.exceptions;

import java.sql.SQLException;

public class DataAccessException extends SQLException {

      /**
       * Constructor for DataAccessException.
       * @param s message
       */
      public DataAccessException(String s) {
            super(s);
      }


      public DataAccessException(String msg, String sql) {
            super(msg, sql);
      }

}
```

```
package vega.weather.interfaces;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletContext;
public interface EventHandler {

	public Event getEvent();
	public void setEvent(Event e);
	public void getWxProduct();
	public void setBeanData();
	public void forward(HttpServletRequest req, HttpServletResponse
res);
	public void doProcess(HttpServletRequest req, HttpServletResponse
res);
	//public void doGet(HttpServletRequest req, HttpServletResponse
res);

}
```

```
package vega.weather.interfaces;

import java.util.ArrayList;
import java.util.Date;

public interface DataBean {

	public String getClassName();
	public String getContent();
	public String getTopLink();
	public Date getDate();
	public void setDate(Date date);
}
```

```
package vega.weather.interfaces;

public interface Event {

	public String getName();
	public DataBean getBean();
	public String getViewName();
	public Class getBeanClass();
	public String getDescription();
	public void setName(String name);
	public void setViewName(String view);
	public void setBean(DataBean bean);
}
```

```
package vega.weather.servlets;

import java.io.IOException;
import java.io.PrintStream;
import javax.servlet.*;
import javax.servlet.http.*;
import vega.weather.core.ConfigurationHandler;
import vega.weather.core.Redirector;
import vega.weather.exceptions.InitializationException;
import vega.weather.interfaces.EventHandler;

public class ControllerServlet extends HttpServlet
{
    public void init(ServletConfig config)
        throws ServletException
    {
        System.out.println("Initializing Controller Servlet");
        javax.servlet.ServletContext context =
config.getServletContext();
        //this.config = config;
        try
        {

ConfigurationHandler.getInstance().initialize(config.getInitParameter("
config_dir"), config.getInitParameter("config_file"));
            Redirector.getInstance().initialize();
config.getInitParameter("targets"));
        }
        catch(InitializationException ie)
        {
            System.out.println(ie.getMessage());
        }
        super.init(config);
        System.out.println("Controller Servlet Ready.");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException
    {
        String eventname = request.getParameter("event");
        response.setContentType("text/html");
        if(eventname == null)
            eventname = "main";
        EventHandler handler =
Redirector.getInstance().getTarget(eventname);

        handler.doProcess(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException
    {
        String event = null;
        try
        {
```

```
            event = request.getParameter("event");
            event = event.toLowerCase();
            if(event != null){
                process(request, response, event);
            }
        }
        catch(Exception e)
        {

request.getRequestDispatcher("jsp/ErrorPage").forward(request,
response);
        }
    }

    public void process(HttpServletRequest request, HttpServletResponse
response, String event)
        throws ServletException, IOException
    {
        javax.servlet.http.HttpSession session =
request.getSession(true);
        EventHandler handler =
Redirector.getInstance().getTarget(event);
        handler.doProcess(request, response);
    }
}
```

```
package vega.weather.util;

import java.io.*;
import java.util.*;
import vega.weather.beans.*;
import vega.weather.core.GenericEventHandler;
import vega.weather.core.ConfigurationHandler;
import vega.weather.events.EventFactory;
import vega.weather.interfaces.DataBean;
import vega.weather.exceptions.InitializationException;

public class FileHandler
{
      private static final int PAIR = 2;
    private String directory;
    private StringBuffer fullFileName;
    private String fileSeparator;
      private String SAT_DIR = "C:\\weatherData\\sat";
    private String RADAR_DIR = "C:\\weatherData\\radar";

    public FileHandler(String dir)
    {
        directory = null;
        fullFileName = null;
        fileSeparator = System.getProperty("file.separator");
        directory = dir;
    }


    public HashMap readConfiguration(String filename)
        throws IOException, FileNotFoundException
    {
        HashMap p = new HashMap();
        BufferedReader in = new BufferedReader(new
FileReader(buildPath(filename)));
        for(String line = null; (line = in.readLine()) != null;){

            //indicates a comment in the config file
            if(line.startsWith("#")){
                  continue;
            }
            try
            {
                StringTokenizer st = new StringTokenizer(line, "=");
                if(st.countTokens() == 2)
                    p.put(st.nextToken(), st.nextToken());
            }
            catch(NullPointerException e)
            {
                throw new IOException("Error during processing of
config file");
            }
        }
        return p;
    }

    private String buildPath(String filename)
```

```
    {
        StringBuffer buff = new StringBuffer(directory);
        buff.append(fileSeparator);
        buff.append(filename);
        return buff.toString();
    }

    public byte[] readFile(String filename)
    {
        byte buffer[] = null;
        try
        {
            File file = new File(filename);
            BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(file));
            int bytes = (int)file.length();
            buffer = new byte[bytes];
            int readBytes = bis.read(buffer);
            bis.close();
            System.out.println("Read " + readBytes + " and expected to
read " + bytes);
        }
        catch(FileNotFoundException e)
        {
            System.out.println("ERROR reading image file" +
e.getMessage());
        }
        catch(IOException e)
        {
            System.out.println("ERROR reading image file" +
e.getMessage());
        }
        return buffer;
    }

      /**
       * @config param: image_base  -- must be an entry in the
weather.config file.
       *                            will be used to build the virtual path
of images to be rendered
       *                            by the web server.
       * @config param: write_flag -- indicator whether images need to
be written to file
       *                            convenience flag to test performance of
just the reads, so I don't
       *                            have to worry about synchronizing the
writes.
       */
    public String writeFile(String type, String region, String name,
byte [] data)
    {
      StringBuffer virtual = null;
      StringBuffer sb = new StringBuffer();
      String write_flag = null;

      try{
```

```
            virtual= new
StringBuffer((String)ConfigurationHandler.getInstance().

      getConfigProperty("image_base"));
            write_flag = (String)ConfigurationHandler.getInstance().

      getConfigProperty("write_flag");
      }catch(InitializationException e){
            System.out.println("Fatal Error: Configuration not
initialized");
      }


      if(virtual==null){
            virtual.append("weatherData");
      }
            //write the image to file
            if("SAT".equals(type)){
                  sb.append(SAT_DIR);
                  virtual.append("/sat/");
            }
            else if("RAD".equals(type)){
                  sb.append(RADAR_DIR);
                  virtual.append("/radar/");
            }
            sb.append(fileSeparator+region+".jpg");
            virtual.append(region+".jpg");
        try
        {
            if("true".equals(write_flag)){
                  FileOutputStream fos = new
FileOutputStream(sb.toString());
                  fos.write(data);

                  fos.close();
                  System.out.println("*** Writing enabled: wrote
"+data.length+
                                          " to file "+sb.toString());
            }
        }
        catch(FileNotFoundException e)
        {
            System.out.println("ERROR reading image file" +
e.getMessage());
        }
        catch(IOException e)
        {
            System.out.println("ERROR reading image file" +
e.getMessage());
        }

        return virtual.toString();

    }


}
```

```
package vega.weather.util;


public class Constants {

      public static final String DBURL = "DBURL";
      public static final String dbUser = "dbUser";
      public static final String dbPassword = "dbPassword";
      public static final String dbName = "dbName";
      public static final String DEFAULT_DIR =
"C:\\Weather\\weather\\webApplication\\admin";
      public static final String CONFIG_FILE = "weather.config";
      public static final String jdbcDriver = "jdbcDriver";
      public static final String dbType = "dbType";

      public static final String MAIN_EVENT_DESC = "Initial entry point
of application.";
      public static final String GENERIC_EVENT_DESC = "Generic Event";
    public static final String
DEFAULT_MAP="weatherData/currentWxMap.jpg";
    public static final String DEFAULT_CONTENT =
    "On the weather map, a cold front extends across the eastern
seaboard of the United states.";

      public static final String XSD_STRING = "XMLType.XSD_STRING";
      public static final String FILE_SEP =
System.getProperty("file.separator");
      public static final String IN = "ParameterMode.IN";
      public static final String OUT = "ParameterMode.OUT";

}
```

```
/**
 * WeatherServiceSoapBindingStub.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package vega.weather.ws;

public class WeatherServiceSoapBindingStub
extends org.apache.axis.client.Stub implements MegavegaWx {
    private java.util.Vector cachedSerClasses = new java.util.Vector();
    private java.util.Vector cachedSerQNames = new java.util.Vector();
    private java.util.Vector cachedSerFactories = new
java.util.Vector();
    private java.util.Vector cachedDeserFactories = new
java.util.Vector();
    static org.apache.axis.description.OperationDesc [] _operations;
    static {
        _operations = new org.apache.axis.description.OperationDesc[3];
        org.apache.axis.description.OperationDesc oper;
        oper = new org.apache.axis.description.OperationDesc();
        oper.setName("getOb");
        oper.addParameter(new javax.xml.namespace.QName("", "city"),
new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
"string"), java.lang.String.class,
org.apache.axis.description.ParameterDesc.IN, false, false);
        oper.setReturnType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
"string"));
        oper.setReturnClass(java.lang.String.class);
        oper.setReturnQName(new javax.xml.namespace.QName("",
"getObReturn"));
        oper.setStyle(org.apache.axis.enum.Style.RPC);
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
        _operations[0] = oper;

        oper = new org.apache.axis.description.OperationDesc();
        oper.setName("getWxForecast");
        oper.addParameter(new javax.xml.namespace.QName("", "city"),
new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
"string"), java.lang.String.class,
org.apache.axis.description.ParameterDesc.IN, false, false);
        oper.setReturnType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
"string"));
        oper.setReturnClass(java.lang.String.class);
        oper.setReturnQName(new javax.xml.namespace.QName("",
"getWxForecastReturn"));
        oper.setStyle(org.apache.axis.enum.Style.RPC);
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
        _operations[1] = oper;

        oper = new org.apache.axis.description.OperationDesc();
        oper.setName("getWxImage");
        oper.addParameter(new javax.xml.namespace.QName("", "type"),
new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
```

```
"string"), java.lang.String.class,
org.apache.axis.description.ParameterDesc.IN, false, false);
        oper.addParameter(new javax.xml.namespace.QName("", "region"),
new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
"string"), java.lang.String.class,
org.apache.axis.description.ParameterDesc.IN, false, false);
        oper.setReturnType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",
"base64Binary"));
        oper.setReturnClass(byte[].class);
        oper.setReturnQName(new javax.xml.namespace.QName("",
"getWxImageReturn"));
        oper.setStyle(org.apache.axis.enum.Style.RPC);
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
        _operations[2] = oper;

    }
    public WeatherServiceSoapBindingStub() throws
org.apache.axis.AxisFault {
         this(null);
    }
    public WeatherServiceSoapBindingStub(java.net.URL endpointURL,
javax.xml.rpc.Service service) throws org.apache.axis.AxisFault {
         this(service);
         super.cachedEndpoint = endpointURL;
    }
    public WeatherServiceSoapBindingStub(javax.xml.rpc.Service service)
throws org.apache.axis.AxisFault {
        if (service == null) {
            super.service = new org.apache.axis.client.Service();
        } else {
            super.service = service;
        }
    }
    private org.apache.axis.client.Call createCall() throws
java.rmi.RemoteException {
        try {
            org.apache.axis.client.Call _call =
                    (org.apache.axis.client.Call)
super.service.createCall();
            if (super.maintainSessionSet) {
                _call.setMaintainSession(super.maintainSession);
            }
            if (super.cachedUsername != null) {
                _call.setUsername(super.cachedUsername);
            }
            if (super.cachedPassword != null) {
                _call.setPassword(super.cachedPassword);
            }
            if (super.cachedEndpoint != null) {
                _call.setTargetEndpointAddress(super.cachedEndpoint);
            }
            if (super.cachedTimeout != null) {
                _call.setTimeout(super.cachedTimeout);
            }
            if (super.cachedPortName != null) {
                _call.setPortName(super.cachedPortName);
```

```
            }
            java.util.Enumeration keys = super.cachedProperties.keys();
            while (keys.hasMoreElements()) {
                java.lang.String key = (java.lang.String)
keys.nextElement();
                _call.setProperty(key,
super.cachedProperties.get(key));
            }
            return _call;
        }
        catch (java.lang.Throwable t) {
            throw new org.apache.axis.AxisFault("Failure trying to get
the Call object", t);
        }
    }

    public java.lang.String getOb(java.lang.String city) throws
java.rmi.RemoteException {
        if (super.cachedEndpoint == null) {
            throw new org.apache.axis.NoEndPointException();
        }
        org.apache.axis.client.Call _call = createCall();
        _call.setOperation(_operations[0]);
        _call.setUseSOAPAction(true);
        _call.setSOAPActionURI("");

_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANT
S);
        _call.setOperationName(new
javax.xml.namespace.QName("http://localhost:8080/axis/services/WeatherS
ervice", "getOb"));

        setRequestHeaders(_call);
        setAttachments(_call);
        java.lang.Object _resp = _call.invoke(new java.lang.Object[]
{city});

        if (_resp instanceof java.rmi.RemoteException) {
            throw (java.rmi.RemoteException)_resp;
        }
        else {
            getResponseHeaders(_call);
            extractAttachments(_call);
            try {
                return (java.lang.String) _resp;
            } catch (java.lang.Exception _exception) {
                return (java.lang.String)
org.apache.axis.utils.JavaUtils.convert(_resp, java.lang.String.class);
            }
        }
    }

    public java.lang.String getWxForecast(java.lang.String city) throws
java.rmi.RemoteException {
        if (super.cachedEndpoint == null) {
            throw new org.apache.axis.NoEndPointException();
        }
```

```
        org.apache.axis.client.Call _call = createCall();
        _call.setOperation(_operations[1]);
        _call.setUseSOAPAction(true);
        _call.setSOAPActionURI("");

_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANT
S);
        _call.setOperationName(new
javax.xml.namespace.QName("http://localhost:8080/axis/services/WeatherS
ervice", "getWxForecast"));

        setRequestHeaders(_call);
        setAttachments(_call);
        java.lang.Object _resp = _call.invoke(new java.lang.Object[]
{city});

        if (_resp instanceof java.rmi.RemoteException) {
            throw (java.rmi.RemoteException)_resp;
        }
        else {
            getResponseHeaders(_call);
            extractAttachments(_call);
            try {
                return (java.lang.String) _resp;
            } catch (java.lang.Exception _exception) {
                return (java.lang.String)
org.apache.axis.utils.JavaUtils.convert(_resp, java.lang.String.class);
            }
        }
    }

    public byte[] getWxImage(java.lang.String type, java.lang.String
region) throws java.rmi.RemoteException {
        if (super.cachedEndpoint == null) {
            throw new org.apache.axis.NoEndPointException();
        }
        org.apache.axis.client.Call _call = createCall();
        _call.setOperation(_operations[2]);
        _call.setUseSOAPAction(true);
        _call.setSOAPActionURI("");

_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANT
S);
        _call.setOperationName(new
javax.xml.namespace.QName("http://localhost:8080/axis/services/WeatherS
ervice", "getWxImage"));

        setRequestHeaders(_call);
        setAttachments(_call);
        java.lang.Object _resp = _call.invoke(new java.lang.Object[]
{type, region});

        if (_resp instanceof java.rmi.RemoteException) {
            throw (java.rmi.RemoteException)_resp;
        }
        else {
            getResponseHeaders(_call);
```

```
            extractAttachments(_call);
            try {
                return (byte[]) _resp;
            } catch (java.lang.Exception _exception) {
                return (byte[])
org.apache.axis.utils.JavaUtils.convert(_resp, byte[].class);
            }
        }
    }

}
/**
 * MegavegaWx.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package vega.weather.ws;

public interface MegavegaWx extends java.rmi.Remote {
    public java.lang.String getOb(java.lang.String city) throws
java.rmi.RemoteException;
    public java.lang.String getWxForecast(java.lang.String city) throws
java.rmi.RemoteException;
    public byte[] getWxImage(java.lang.String type, java.lang.String
region) throws java.rmi.RemoteException;
}
/**
 * MegavegaWxService.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package vega.weather.ws;

public interface MegavegaWxService extends javax.xml.rpc.Service {
    public java.lang.String getWeatherServiceAddress();

    public MegavegaWx getWeatherService()
    throws javax.xml.rpc.ServiceException;

    public MegavegaWx getWeatherService(java.net.URL portAddress)
    throws javax.xml.rpc.ServiceException;
}
/**
 * MegavegaWxServiceLocator.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package vega.weather.ws;

public class MegavegaWxServiceLocator
extends org.apache.axis.client.Service implements MegavegaWxService {
```

```
    // Use to get a proxy class for WeatherService
    private String WeatherService_address =
"http://localhost:8080/axis/services/WeatherService";

    public java.lang.String getWeatherServiceAddress() {
        return WeatherService_address;
    }

    // The WSDD service name defaults to the port name.
    private java.lang.String WeatherServiceWSDDServiceName =
"WeatherService";

    public java.lang.String getWeatherServiceWSDDServiceName() {
        return WeatherServiceWSDDServiceName;
    }

    public void setWeatherServiceWSDDServiceName(java.lang.String name)
{
        WeatherServiceWSDDServiceName = name;
    }

    public MegavegaWx getWeatherService() throws
javax.xml.rpc.ServiceException {
       java.net.URL endpoint;
        try {
            endpoint = new java.net.URL(WeatherService_address);
        }
        catch (java.net.MalformedURLException e) {
            throw new javax.xml.rpc.ServiceException(e);
        }
        return getWeatherService(endpoint);
    }

    public MegavegaWx getWeatherService(java.net.URL portAddress)
    throws javax.xml.rpc.ServiceException {
        try {
            WeatherServiceSoapBindingStub _stub = new
WeatherServiceSoapBindingStub(portAddress, this);
            _stub.setPortName(getWeatherServiceWSDDServiceName());
            return _stub;
        }
        catch (org.apache.axis.AxisFault e) {
            return null;
        }
    }

    /**
     * For the given interface, get the stub implementation.
     * If this service has no port for the given interface,
     * then ServiceException is thrown.
     */
    public java.rmi.Remote getPort(Class serviceEndpointInterface)
throws javax.xml.rpc.ServiceException {
        try {
            if
(MegavegaWx.class.isAssignableFrom(serviceEndpointInterface)) {
```

```
                WeatherServiceSoapBindingStub _stub = new
WeatherServiceSoapBindingStub(new java.net.URL(WeatherService_address),
this);
                _stub.setPortName(getWeatherServiceWSDDServiceName());
                return _stub;
            }
        }
        catch (java.lang.Throwable t) {
            throw new javax.xml.rpc.ServiceException(t);
        }
        throw new javax.xml.rpc.ServiceException("There is no stub
implementation for the interface:  " + (serviceEndpointInterface ==
null ? "null" : serviceEndpointInterface.getName()));
    }

    /**
     * For the given interface, get the stub implementation.
     * If this service has no port for the given interface,
     * then ServiceException is thrown.
     */
    public java.rmi.Remote getPort(javax.xml.namespace.QName portName,
Class serviceEndpointInterface) throws javax.xml.rpc.ServiceException {
        if (portName == null) {
            return getPort(serviceEndpointInterface);
        }
        String inputPortName = portName.getLocalPart();
        if ("WeatherService".equals(inputPortName)) {
            return getWeatherService();
        }
        else  {
            java.rmi.Remote _stub = getPort(serviceEndpointInterface);
            ((org.apache.axis.client.Stub)
_stub).setPortName(portName);
            return _stub;
        }
    }

    public javax.xml.namespace.QName getServiceName() {
        return new
javax.xml.namespace.QName("http://localhost:8080/axis/services/WeatherS
ervice", "MegavegaWxService");
    }

    private java.util.HashSet ports = null;

    public java.util.Iterator getPorts() {
        if (ports == null) {
            ports = new java.util.HashSet();
            ports.add(new javax.xml.namespace.QName("WeatherService"));
        }
        return ports.iterator();
    }

}
```

Appendix E

.NET Web Application Source Code

```
using System;
using MegaVega.WebReference1;
using MegaVega.utils;

namespace MegaVega.core
{
	/// <summary>
	/// Summary description for WxConnection.
	/// </summary>
	public class WxConnection
	{
		MegavegaWxService ws_ref;
		public WxConnection()
		{
			ws_ref = new MegavegaWxService();

		}

		public string getWx(string city, string time)
		{

			string result = null;
			result = ws_ref.getOb(city);
			return result;
		}

		public string getImage(string type, string region, string name){

			byte [] image =	ws_ref.getWxImage(type, region);
			FileHandler h = new FileHandler(null);
			return h.writeFile(type, region, name, image);
		}

		public String getForecast(String city, String time)
		{
			return ws_ref.getWxForecast(city);
		}
	}
}
```

```
using System;
using System.Collections;

namespace MegaVega.core
{
	/// <summary>
	/// Summary description for CacheManager.
	/// </summary>
	public class CacheManager
	{
		private Hashtable cities;
		private ArrayList states  ;
		private ArrayList regions ;
		private ArrayList products ;
		private bool initialized;
		private static CacheManager cm = new CacheManager();

		private CacheManager()
		{
			if(!initialized)
			{
				init();
			}
		}
		//private void CacheManager(){}
		private void init()
		{
			initCities();
			initStates();
			initProducts();
			initRegions();
			initialized = true;
		}

		public static CacheManager getInstance()
		{
			return cm;
		}

		private void initCities()
		{
			cities = new Hashtable();
			//Florida
			cities.Add("Jacksonville","Florida");
			cities.Add("Miami","Florida");
			cities.Add("Orlando","Florida");
			cities.Add("Tampa","Florida");
			cities.Add("Key West","Florida");
			cities.Add("Talahassee","Florida");
			cities.Add("Melbourne","Florida");
			cities.Add("Pensacola","Florida");
			cities.Add("Ft. Lauderdale","Florida");
			//Georgia
			cities.Add("Atlanta","Georgia");
			cities.Add("Macon","Georgia");
			cities.Add("Valdosta","Georgia");
			cities.Add("Savanah","Georgia");
```

```
        //Alabama
        cities.Add("Birmingham","Alabama");
        cities.Add("Mobile","Alabama");
        cities.Add("Guien","Alabama");
        cities.Add("More","Alabama");

        //Maine
        cities.Add("Bangor","Maine");
        cities.Add("Portland","Maine");
        cities.Add("Booth Bay","Maine");
        cities.Add("Brunswick","Maine");

}

private void initStates()
{

        states = new ArrayList();

        states.Add("Select a state");
        states.Add("Florida");
        states.Add("Georgia");
        states.Add("Alabama");
        states.Add("Maine");
}

private void initProducts()
{

        products = new ArrayList();

        products.Add("Select a product");
        products.Add("Satellite");
        products.Add("Radar");

}

private void initRegions()
{

        regions = new ArrayList();

        regions.Add("Continental US");
        regions.Add("South East");
        regions.Add("North East");
        regions.Add("Great Plains");
        regions.Add("North West");
        regions.Add("South West");
}


/**
 * Gets the cities
 * @return Returns a HashMap
 */
public ArrayList getCities(String state)
{
```

```
                ArrayList list = new ArrayList();

                if(state ==null)
                {
                        return list;
                }

                foreach(string cityKey in cities.Keys)
                {
                        string stInCollection = (string) cities[cityKey];
                        if(stInCollection.Equals(state))
                        {
                                list.Add(cityKey);
                        }
                }
                return list;
        }
        /**
         * Gets the states
         * @return Returns a HashMap
         */
        public ArrayList getStates()
        {
                return states;
        }

        /**
         * Gets the regions
         * @return Returns a HashSet
         */
        public ArrayList getRegions()
        {
                return regions;
        }

        /**
         * Gets the products
         * @return Returns a HashSet
         */
        public ArrayList getProducts()
        {
                return products;
        }
        public bool isInitialized()
        {
                return initialized;
        }

        public String getStateCode(String city)
        {
                return (String) cities[city];
        }

    }
}
using System;
```

```
using System.Collections;
using MegaVega.exceptions;
using MegaVega.events;
using MegaVega.interfaces;
using MegaVega.utils;


namespace MegaVega.core
{
	/// <summary>
	/// Summary description for ConfigHandler.
	/// </summary>
	public class ConfigHandler
	{
		private Hashtable configProps;
		private Hashtable targets;
		private static bool initialized;
		private static ConfigHandler instance;

		public static ConfigHandler getInstance()
		{
			instance = new ConfigHandler();
			if(!initialized)
			{
				instance.initialize(null,null);
			}
			return instance;
		}

		private ConfigHandler()
		{
			configProps = null;
			targets = new Hashtable();
			initialize(null,null);
		}

		public void initialize(string directory, string file)
		{
			try
			{
				if(directory == null || file == null)

	loadConfiguration("C:\\Inetpub\\wwwroot\\MegaVega\\resources",
"weather.config");
				else
				loadConfiguration(directory, file);
			}
			//catch(FileNotFoundException nf){
			//	throw new InitializationException(this.GetType().Name);
			//}
			catch(System.Exception e)
			{
				throw new InitializationException("Unable to read config
file");
			}
			if(configProps.Count != 0 )
			{
```

```
                        initialized = true;
                }
                else
                {
                        throw new InitializationException("Configuration Init
Failed");
                }
        }

        public void loadConfiguration(string directory, string file)
        {
                FileHandler handler = new FileHandler(directory);
                configProps = handler.readConfiguration(file);
                //hardcode(configProps);
        }

        public string getConfigProperty(string key)
        {
                return (string)configProps[key];
        }

        public MegaVega.interfaces.EventHandler getEventHandler(string
eventName)
        {
                return (MegaVega.interfaces.EventHandler)configProps[eventName];
        }

        public void setConfigProperty(string key, string property)
        {
                configProps.Add(key, property);
        }

        /*public void addToConfiguration(Hashtable p)
        {
                configProps.(p);
        }*/

        public void removeFromConfig(string key)
        {
                configProps.Remove(key);
        }

        public bool containsProperty(string key)
        {
                return configProps.ContainsKey(key);
        }

        /*private void hardcode(Hashtable p)
        {
        try
        {

        for(Iterator i=p.keySet().iterator(); i.hasNext();)
        {

        string key = i.next().tostring();
        string fullClassName = null;
```

```
                string eventName = null;
                string eventView = null;
                string beanClass = null;

                int index;

                if((index = key.IndexOf("_event")) != -1)
                { //event parameter found
                fullClassName = (string) p[key];
                if(fullClassName == null)
                {//load the event class
                        Console.WriteLine("FATAL ERROR Events are not initialized");
                return;
                }
                                else
                                {
                        //revisit
                                //Event eventArg =
(Event)this.GetType().getClassLoader().
                                //loadClass(fullClassName).newInstance();
                                eventName = key.Substring(0,index);
                                eventArg.setName(eventName);

        eventArg.setViewName((string)configProps[eventName+"_target"]);
                                beanClass = (string)configProps[eventName+"_bean"];
        //revisit
                        //      DataBean bean = (DataBean) getClass().getClassLoader().
                        //      loadClass(beanClass).newInstance();
                        //      eventArg.setBean(bean);

                                loadEventHandler(eventArg);

                }
                }


                }
                }
                catch(ClassNotFoundException e)
                {

                }
                catch(IllegalAccessException e)
                {

                }
                catch(InstantiationException e)
                {
                e.printStackTrace();
                }

                        vega.weather.events.MainEvent me =
EventFactory.createMainEvent("main", "jsp/MainPage.jsp", (DataBean)new
MainPageBean());
                                vega.weather.events.CityWxEvent ce =
EventFactory.createCityWxEvent("cityWx", "jsp/CityWx.jsp", (DataBean)new
CityWxBean());
```

```
                        vega.weather.events.SatelliteEvent se =
EventFactory.createSatelliteEvent("satellite", "jsp/Satellite.jsp", (DataBean)new
SatelliteBean());
                        vega.weather.events.RadarEvent re =
EventFactory.createRadarEvent("radar", "jsp/Radar.jsp", (DataBean)new RadarBean());
                        vega.weather.events.AdminEvent ae =
EventFactory.createAdminEvent("radar", "jsp/Admin.jsp", (DataBean)new AdminBean());
                        p.Add("main", new GenericEventHandler(me));
                        p.Add("citywx", new GenericEventHandler(ce));
                        p.Add("satellite", new GenericEventHandler(se));
                        p.Add("radar", new GenericEventHandler(re));
                        p.Add("admin", new GenericEventHandler(ae));
            }*/

            /*private void loadEventHandler(Event eventObj)
            {

            //try to get the event specific handler
                  string handlerName =
(string)configProps[eventObj.getName()+"_handler"];
                  //no event handler given in config file use default
                  if(handlerName == null)
                  {
                  handlerName = (string)configProps["default_handler"];
                  }
                  if(handlerName != null)
                  {

                  MegaVega.interfaces.EventHandler handler =
(MegaVega.interfaces.EventHandler)getClass().getClassLoader().
                  loadClass(handlerName).newInstance();

                  handler.setEvent(eventObj);
                              //add this event to the config properties
                  targets.Add(eventObj.getName(), handler);
                  }
                        // this is a severe application error no default handler
found.
                        // should throw application error
                  else
                  {
                        Console.WriteLine("Fatal ERROR no default handler
specified");
                  }

      }*/
            /**
             * Method to return targets collection
             * Targets maps an event name to an event handler.
             * Event handlers contain a referece to the event object.
             * */
            public Hashtable getEventMaps()
            {
                  if(!initialized){
                        throw new InitializationException(this.GetType().Name);
                  }
```

```
                    //revisit to provide dynamic class loading from configuration

                    targets.Add("main",new GenericEventHandler(new MainEvent()));
                    targets.Add("citywx",new GenericEventHandler(new
CitywxEvent()));
                    targets.Add("satellite",new GenericEventHandler(new
SatelliteEvent()));
                    targets.Add("radar",new GenericEventHandler(new RadarEvent()));
                    targets.Add("admin",new GenericEventHandler(new AdminEvent()));

                    return targets;
                }
        }
}
```

```
using System;
using System.Collections;
using MegaVega.interfaces;
using System.Web;
using System.Web.SessionState;
using System.Text;

namespace MegaVega.core
{
	/// <summary>
	/// Summary description for GenericEventHandler.
	/// </summary>
	public class GenericEventHandler:MegaVega.interfaces.EventHandler
	{
		private Event eventObj;
		Product product = new Product();

		public GenericEventHandler(Event eventObj)
		{
			this.eventObj = eventObj;
		}
		//must implement a no args constructor for dynamic class loading
		public GenericEventHandler()
		{
		}

		public Event getEvent()
		{
			return eventObj;
		}

		public void setEvent(Event e)
		{
			eventObj = e;
		}

		public void getWxProduct()
		{
		}


		public void forward(HttpRequest req, HttpResponse res)
		{
			Console.WriteLine("Forwarding to " + eventObj.getViewName());
			string view = eventObj.getViewName();
			res.Redirect(view);
		}

		public void doProcess(HttpRequest req, HttpResponse res,
HttpSessionState session)
		{

			string eventname = eventObj.getName();
			Hashtable parameters = new Hashtable();

			//StringBuilder content = new StringBuilder();
			if("citywx".Equals(eventname))
```

```
                        {
                                string city = req.Params["cities"];

                                parameters.Add("ob",product.getCityWx(city,null));

        parameters.Add("forecast",product.getForecast(city,null));
                                parameters.Add("city",city);
                                parameters.Add("state",req.Params["states"]);
                                eventObj.setReqParams(parameters);
                                session.Add("weatherEvent",eventObj);

                        }
                        else if("satellite".Equals(eventname))
                        {
                                string region = req.Params["regions"];
                                parameters.Add("imagepath",product.getSatellite(region
,null));
                                parameters.Add("region",region);
                                parameters.Add("productName",req.Params["products"]);
                                eventObj.setReqParams(parameters);
                                session.Add("weatherEvent",eventObj);
                        }
                        else if("radar".Equals(eventname))
                        {
                                string region = req.Params["regions"];
                                parameters.Add("imagepath",product.getRadar(region
,null));
                                parameters.Add("region",region);
                                parameters.Add("productName",req.Params["products"]);
                                eventObj.setReqParams(parameters);
                                session.Add("weatherEvent",eventObj);
                        }
                        else
                        {
                                session.Add("weatherEvent",eventObj);
                        }
                        forward(req, res);
                }
        }
}
```

```
using System;
using MegaVega.utils;

namespace MegaVega.core
{
	/// <summary>
	/// Summary description for Product.
	/// </summary>
	public class Product
		{
			private WxConnection wscon;

			public Product()
			{
				wscon = new WxConnection();
			}
			/**
		 * Gets the output
		 * @return Returns a Object
		 */
			public string getCityWx(string city, string time)
			{
				//maybe get the icao from the CacheManager.
				string result = null;
				try
				{
					result = wscon.getWx(city, time);
				}
				catch(Exception re)
				{
					string m = re.Message;
					string t = re.StackTrace.ToString();
					Console.WriteLine(m);
				}
				return result;
			}

			/**
			 * Gets the output
			 * @return Returns a Object

			 */
			public String getForecast(String city, String time)
			{

				String result = null;
				try
				{
					result = wscon.getForecast(city, time);
				}
				catch(Exception re)
				{
					//logger.error("Error getting Web service
update");
				}
				return result;
			}
```

```
                        /**
                         * Gets the output
                         * @return Returns a Object
                         */

                        public string getSatellite(String region, String name)
                                {
                                        string path=null;
                                        try
                                        {
                                                path = wscon.getImage("SAT", region, name);
                                        }
                                        catch(Exception re)
                                        {
                                                //logger.error("Error getting Web service
image update");
                                        }
                                        return path;
                        }

                        /**
                         * Returns the path to the downloaded image.
                         * @return Returns a Object
                         */
                        public String getRadar(String region, String name)
                        {
                                string path=null;
                                try
                                {
                                        path = wscon.getImage("RAD", region, name);
                                }
                                catch(Exception re)
                                {
                                        //logger.error("Error getting Web service image
update");
                                }
                                return path;
                        }


                        public String getSummary(String time)
                        {
                                return "On the Weather map a cold front is
developing..."+
                                        "<BR> the southeast is going to get drenched";
                        }

                        public String getNatlMap(String time)
                        {
                                return "c:\\natl12z.gif";
                        }
        }
}
```

```
using System;
using System.Collections;

namespace MegaVega.core
{
	/// <summary>
	/// Singleton class
	/// </summary>
	public class Redirector
	{
		private Hashtable targets;
		private static Redirector instance = new Redirector();

		private Redirector()
		{
			initialize();
		}

		private void initialize(){
			targets = ConfigHandler.getInstance().getEventMaps();
		}

		public static Redirector getInstance(){
			return instance;
		}
		//returns the event.
		//Event must contain required information.
		public MegaVega.interfaces.EventHandler getTarget(string key){
			return (MegaVega.interfaces.EventHandler)targets[key];
		}

	}
}
```

```
using System;
using System.Collections;
using MegaVega.interfaces;
using MegaVega.utils;

namespace MegaVega.events
{
	/// <summary>
	/// Summary description for WeatherEvent.
	/// </summary>
	public abstract class WeatherEvent :Event
	{
		private String name = null;
		private String viewName = null;
		protected MegaVega.interfaces.EventHandler handler;
		private String description = Constants.GENERIC_EVENT_DESC;
		protected Hashtable reqParams;

		public WeatherEvent(string name, string view)
		{
			this.name = name;
			viewName = view;
		}
		public WeatherEvent(){
		}

		public String getName(){
			return name;
		}

		public String getViewName(){
			return viewName;
		}

		public String getDescription(){
			return description;
		}

		public MegaVega.interfaces.EventHandler getHandler()
		{
			return handler;
		}

		public abstract Hashtable getReqParams();


		public void setDescription(String desc)
		{
			description = desc;
		}
		public void setName(String name){
			this.name = name;
		}

		public void setViewName(String view){
			viewName = view;
		}
```

```
		//left to subclasses to implement the required params
		public abstract void setReqParams(Hashtable reqP);

		public void setHandler(MegaVega.interfaces.EventHandler h)
		{
			handler = h;
		}
	}
}
```

```
using System;
using System.Collections;
using MegaVega.core;

namespace MegaVega.events
{
	/// <summary>
	/// Summary description for AdminEvent.
	/// </summary>
	public class AdminEvent : WeatherEvent
	{
		private MegaVega.interfaces.EventHandler eventhandler;
		private Hashtable parameters = new Hashtable();

		public AdminEvent(string name, string view)
			:base (name,view){
			handler = new GenericEventHandler();
		}

		public AdminEvent()
		{
			base.setViewName(ConfigHandler.getInstance().
				getConfigProperty("admin_target"));
			base.setName("admin");
			base.reqParams = parameters;
		}

		public override void  setReqParams(Hashtable req)
		{
			base.reqParams = req;
		}

		public override Hashtable getReqParams()
		{
			return base.reqParams;
		}
	}
}
```

```
using System;
using System.Collections;
using MegaVega.core;

namespace MegaVega.events
{
	/// <summary>
	/// Summary description for CitywxEvent.
	/// </summary>
	public class CitywxEvent : WeatherEvent
	{
		private MegaVega.interfaces.EventHandler eventhandler;
		private Hashtable parameters = new Hashtable();

		public CitywxEvent(string name, string view)
			:base (name,view){
			handler = new GenericEventHandler();
		}

		public CitywxEvent()
		{
			base.setViewName(ConfigHandler.getInstance().
				getConfigProperty("citywx_target"));
			base.setName("citywx");
			base.reqParams = parameters;
		}

		public override void  setReqParams(Hashtable req){
			base.reqParams = req;
		}

		public override Hashtable getReqParams()
		{
			return base.reqParams;
		}

	}
}
```

```
using System;
using System.Collections;
using MegaVega.core;

namespace MegaVega.events
{
	/// <summary>
	/// Summary description for MainEvent.
	/// </summary>
	public class MainEvent : WeatherEvent
	{
		//private MegaVega.interfaces.EventHandler eventhandler;
		private Hashtable parameters = new Hashtable();

		public MainEvent(string name, string view)
			:base (name,view){
			base.handler = new GenericEventHandler();
			base.reqParams = parameters;
		}

		public MainEvent()
		{
			base.setViewName(ConfigHandler.getInstance().
					getConfigProperty("main_target"));
			base.setName("main");
			base.reqParams = parameters;
		}
		public override void  setReqParams(Hashtable req)
		{
			base.reqParams = req;
		}

		public override Hashtable getReqParams()
		{
			return base.reqParams;
		}

	}
}
```

```
using System;
using System.Collections;
using MegaVega.core;

namespace MegaVega.events
{
	/// <summary>
	/// Summary description for RadarEvent.
	/// </summary>
	public class RadarEvent : WeatherEvent
	{
		private Hashtable parameters = new Hashtable();

		public RadarEvent(string name, string view)
			:base (name,view){
			handler = new GenericEventHandler();
		}

		public RadarEvent()
		{
			base.setViewName(ConfigHandler.getInstance().
				getConfigProperty("radar_target"));
			base.setName("radar");
			base.reqParams = parameters;
		}
		public override void  setReqParams(Hashtable req)
		{
			base.reqParams = req;
		}

		public override Hashtable getReqParams()
		{
			return base.reqParams;
		}

	}
}
```

```
using System;
using System.Collections;
using MegaVega.core;

namespace MegaVega.events
{
	/// <summary>
	/// Summary description for SatelliteEvent.
	/// </summary>
	public class SatelliteEvent : WeatherEvent
	{
		private Hashtable parameters = new Hashtable();

		public SatelliteEvent(string name, string view)
			:base (name,view){
			handler = new GenericEventHandler();
		}

		public SatelliteEvent()
		{
			base.setViewName(ConfigHandler.getInstance().
				getConfigProperty("satellite_target"));
			base.setName("satellite");
			base.reqParams = parameters;
		}
		public override void  setReqParams(Hashtable req)
		{
			base.reqParams = req;
		}

		public override Hashtable getReqParams()
		{
			return base.reqParams;
		}

	}
}
```

```
using System;

namespace MegaVega.exceptions
{
	/// <summary>
	/// Summary description for InitializationException.
	/// </summary>
	public class InitializationException: System.Exception
	{
		public InitializationException(string classname): base(
					(classname +" fail to Initialize.")){}
	}
}
using System;

namespace MegaVega.exceptions
{
	/// <summary>
	/// Summary description for DataAccessException.
	/// </summary>

		public class DataAccessException: System.Exception
		{
			public DataAccessException(string classname): base(
				(classname +" fail to Initialize.")){}
		}

}
```

```
using System;
using System.Web;
using System.Web.SessionState;

namespace MegaVega.interfaces
{
	/// <summary>
	/// Summary description for EventHandler.
	/// </summary>
	public interface EventHandler
	{
		Event getEvent();

		void setEvent(Event e);

		void getWxProduct();

		void forward(HttpRequest req, HttpResponse res);

		void doProcess(HttpRequest req, HttpResponse res, HttpSessionState s);
	}
}
```

```
using System;
using System.Collections;

namespace MegaVega.interfaces
{
	/// <summary>
	/// Summary description for Event.
	/// </summary>
	public interface Event
	{
		string getName();

		string getViewName();

		string getDescription();

		MegaVega.interfaces.EventHandler getHandler();

		Hashtable getReqParams();

		void setName(string name);

		void setViewName(string view);

		void setDescription(string desc);

		void setReqParams(Hashtable reqParams);

		void setHandler (MegaVega.interfaces.EventHandler handler);
	}
}
```

```
using System;
using System.Collections;
using System.IO;
using System.Text;
using MegaVega.core;
using MegaVega.exceptions;

namespace MegaVega.utils
{
	/// <summary>
	/// Summary description for FileHandler.
	/// </summary>
	public class FileHandler
	{
		private const int PAIR = 2;
		private string directory;
		private StringBuilder fullFileName;
		private string fileSeparator="\\";
		private string SAT_DIR = "C:\\weatherData\\sat";
		private string RADAR_DIR = "C:\\weatherData\\radar";


    public FileHandler(string dir)
    {
        directory = null;
        fullFileName = null;
     //   fileSeparator = Environment.
            directory = dir;
    }
        /**
         * Tokenizes the config file based on the '=' character
         * takes the first param as the config key and the second as
         * the config value.
         * '#' is considered a comment and therefore ignored.
         * @return Hashtable with configuration properties.
         * */

    public Hashtable readConfiguration(string filename)
        {
        Hashtable p = new Hashtable();
            FileInfo file = new FileInfo(buildPath(filename));
        StreamReader reader = file.OpenText();
            string [] elements;
            char [] delimiters={'='};

        for(string line = null; (line = reader.ReadLine()) != null;){

            //indicates a comment in the config file
            if(line.StartsWith("#")){
                  continue;
            }
           try
           {
               elements = line.Split(delimiters);
                        if(elements.Length == 2)
                        {
                              p.Add(elements[0], elements[1]);
```

```
                        }
            }
            catch(NullReferenceException e)
            {
                        Console.WriteLine(e.Message);
                throw new IOException("Error during processing of config file");
            }

        }
              reader.Close();
        return p;
    }

    private string buildPath(string filename)
    {
        StringBuilder buff = new StringBuilder(directory);
        buff.Append("\\");
        buff.Append(filename);
        return buff.ToString();
    }

    public byte[] readFile(string filename)
    {
        byte [] buffer = null;
        try
        {
            FileInfo file = new FileInfo(filename);
            StreamReader bis = file.OpenText();
            int bytes = (int)file.Length;
            buffer = new byte[bytes];
            int readBytes = bis.ReadToEnd().Length;
            bis.Close();
            Console.WriteLine("Read " + readBytes + " and expected to read " + bytes);
        }
        catch(FileNotFoundException e)
        {
            Console.WriteLine("ERROR reading image file" + e.Message);
        }
        catch(IOException e)
        {
            Console.WriteLine("ERROR reading image file" + e.Message);
        }
        return buffer;
    }

       /**
        * @config param: image_base  -- must be an entry in the weather.config file.
        *                               will be used to build the virtual path of images
to be rendered
        *                               by the web server.
        * @config param: write_flag -- indicator whether images need to be written to
file
        *                               convenience flag to test performance of just the
reads, so I don't
        *                               have to worry about synchronizing the writes.
        * Not implemented in C#
        */
```

```
    public string writeFile(string type, string region, string name, byte [] data)
    {
        StringBuilder path = null;
        StringBuilder sb = new StringBuilder();
        string write_flag = null;

        try{
                path= new StringBuilder((string)ConfigHandler.getInstance().
                                                        getConfigProperty("image_base"));
                write_flag = (string)ConfigHandler.getInstance().
                                                        getConfigProperty("write_flag");
        }catch(InitializationException e){
                Console.WriteLine("Fatal Error: Configuration not
initialized"+e.Message);
        }


        if(path==null){
                path.Append("weatherData");
        }
                //write the image to file
                if("SAT".Equals(type)){
                        sb.Append(SAT_DIR);
                        path.Append("/sat/");
                }
                else if("RAD".Equals(type)){
                        sb.Append(RADAR_DIR);
                        path.Append("/radar/");
                }
                sb.Append(fileSeparator+region+".jpg");
                path.Append(region+".jpg");
         try
         {  //write_flag will no be set to true for testing
                if("true".Equals(write_flag)){
                /*StreamWriter fos = new FileOutputStream(sb.ToString());
                fos.write(data);

                fos.close();
                Console.WriteLine("*** Writing enabled: wrote "+data.Length+
                                        " to file "+sb.ToString());*/
                }
         }
         catch(FileNotFoundException e)
         {
             Console.WriteLine("ERROR reading image file" + e.Message);
         }
         catch(IOException e)
         {
             Console.WriteLine("ERROR reading image file" + e.Message);
         }

         return path.ToString();

    }
       }
}
```

```
using System;

namespace MegaVega.utils
{
	/// <summary>
	/// Summary description for Constants.
	/// </summary>
	public class Constants
	{
		public  const string DBURL = "DBURL";
		public const string dbUser = "dbUser";
		public  const string dbPassword = "dbPassword";
		public  const string dbName = "dbName";
		public  const string DEFAULT_DIR =
"C:\\Weather\\weather\\webApplication\\admin";
		public  const string CONFIG_FILE = "weather.config";
		public  const string jdbcDriver = "jdbcDriver";
		public  const string dbType = "dbType";
		public  const string MAIN_EVENT_DESC = "Initial entry point of
application.";
		public  const string GENERIC_EVENT_DESC = "Generic Event";
		public  const string DEFAULT_MAP="weatherData/currentWxMap.jpg";
		public  const string DEFAULT_CONTENT =
		"On the weather map, a cold front extends across the eastern seaboard
of the United states.";

		public  const string XSD_string = "XMLType.XSD_string";
		//public  const string FILE_SEP = System.getProperty("file.separator");
		public  const string IN = "ParameterMode.IN";
		public  const string OUT = "ParameterMode.OUT";

	}
}
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using MegaVega.core;
using MegaVega.interfaces;
using MegaVega.events;

namespace MegaVega
{
	/// <summary>
	/// Summary description for MainPage.
	/// </summary>
	public class MainPage : System.Web.UI.Page
	{
		//web form controls
		protected DropDownList states;
		protected DropDownList cities;
		protected DropDownList products;
		protected DropDownList regions;
		protected Label today;
		protected Button prod_bttn;
		protected Button goCity;
		protected System.Web.UI.WebControls.Image weatherMap;
		//arrays to hold the data
		protected ArrayList ctyArray = new ArrayList();//just the first time is
empty
		protected ArrayList stArray;
		protected ArrayList productsArray;
		protected System.Web.UI.HtmlControls.HtmlGenericControl summaryHeader;
		protected System.Web.UI.WebControls.Label summary;
		protected ArrayList regionsArray;
		protected HtmlInputHidden action;
		private MegaVega.interfaces.EventHandler handler;
		private MegaVega.interfaces.Event weatherEvent;


		private void Page_Load(object sender, System.EventArgs e)
		{

			if(!IsPostBack)
			{
				weatherEvent =
(MegaVega.interfaces.Event)Session["currentEvent"];
				if(weatherEvent == null)
				{//first time hitting the application
					weatherEvent = new
MainEvent("main","MainPage.aspx");
					action.Value="main";
				}
```

```
                today.Text = DateTime.Now.ToLongDateString();
                //get data from the cache manager to populate drop downs
                stArray = CacheManager.getInstance().getStates();
                productsArray=CacheManager.getInstance().getProducts();
                regionsArray = CacheManager.getInstance().getRegions();
                //binding data to server controls
                states.DataSource = stArray;
                states.DataBind();
                products.DataSource = productsArray;
                products.DataBind();
                regions.DataSource = regionsArray;
                regions.DataBind();

        }
        if(states.SelectedIndex == 0)
        {
                ctyArray.Add("Select a city");
                cities.DataSource = ctyArray;
                cities.DataBind();
        }

}
/**
 * Method to get the cities for the selected state.
 * Cities are retrieved from the CacheManager
 * */
public void getCities(object sender, System.EventArgs e)
{
        ctyArray = CacheManager.getInstance().
                                getCities(states.SelectedItem.Text);
        cities.DataSource = ctyArray;
        cities.DataBind();
}
/**
 * Method to retrieve the summary description
 * */
public void getSummary(object sender, EventArgs e)
{
//      summary.Text = handler.getSummary();
        weatherMap.Visible = true;
        summaryHeader.Visible = false;
}
public void getCity(object sender, EventArgs e){
        //set next action
        action.Value = "citywx";
        handler = Redirector.getInstance().getTarget("citywx");
        handler.doProcess(Request, Response, Session);
}
public void getProduct(object sender, EventArgs e)
{
        string prod = products.SelectedItem.Value.ToLower();
        handler = Redirector.getInstance().getTarget(prod);
        handler.doProcess(Request, Response, Session);
}

        #region Web Form Designer generated code
override protected void OnInit(EventArgs e)
```

```
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.action.ServerChange += new
System.EventHandler(this.action_ServerChange);

            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void action_ServerChange(object sender, System.EventArgs e)
        {

        }
    }
}
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using MegaVega.core;

namespace MegaVega
{
	/// <summary>
	/// Summary description for Radar.
	/// </summary>
	public class Radar : System.Web.UI.Page
	{
		protected DropDownList states;
		protected DropDownList cities;
		protected DropDownList products;
		protected DropDownList regions;
		protected Button prod_bttn;
		protected Button goCity;
		protected System.Web.UI.WebControls.Label radarHeader;
		protected System.Web.UI.WebControls.Image radarImage;
		//arrays to hold the data
		protected ArrayList ctyArray = new ArrayList();//just the first time is
empty
		protected ArrayList stArray;
		protected ArrayList productsArray;
		protected ArrayList regionsArray;
		protected HtmlInputHidden action;
		private MegaVega.interfaces.EventHandler handler;
		private MegaVega.interfaces.Event weatherEvent;
		private void Page_Load(object sender, System.EventArgs e)
		{
			if(!IsPostBack)
			{
				weatherEvent =
(MegaVega.interfaces.Event)Session["weatherEvent"];
				Hashtable data = weatherEvent.getReqParams();
				string region = (string)data["region"];
				radarHeader.Text = "Satellite Image for the "+region+"
region.";
				radarImage.ImageUrl =(string)data["imagepath"];
				//get data from the cache manager to populate drop downs
				stArray = CacheManager.getInstance().getStates();
				productsArray=CacheManager.getInstance().getProducts();
				regionsArray = CacheManager.getInstance().getRegions();
				//binding data to server controls
				states.DataSource = stArray;
				states.DataBind();
				products.DataSource = productsArray;
				products.DataBind();
```

```
                regions.DataSource = regionsArray;
                regions.DataBind();
        }
        if(states.SelectedIndex == 0)
        {
                ctyArray.Add("Select a city");
                cities.DataSource = ctyArray;
                cities.DataBind();
        }
}
/**
 * Method to get the cities for the selected state.
 * Cities are retrieved from the CacheManager
 * */
public void getCities(object sender, System.EventArgs e)
{
        ctyArray = CacheManager.getInstance().
                getCities(states.SelectedItem.Text);
        cities.DataSource = ctyArray;
        cities.DataBind();
}
public void getCity(object sender, EventArgs e)
{
        //set next action
        action.Value = "citywx";
        handler = Redirector.getInstance().getTarget("citywx");
        handler.doProcess(Request, Response, Session);
}
public void getProduct(object sender, EventArgs e)
{
        string prod = products.SelectedItem.Value.ToLower();
        handler = Redirector.getInstance().getTarget(prod);
        handler.doProcess(Request, Response, Session);
}
#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
        InitializeComponent();
        base.OnInit(e);
}
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        this.Load += new System.EventHandler(this.Page_Load);

}#endregion }}
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using MegaVega.core;
using MegaVega.interfaces;
using MegaVega.events;

namespace MegaVega
{
	/// <summary>
	/// Summary description for Citywx.
	/// </summary>
	public class Citywx : System.Web.UI.Page
	{
		protected DropDownList states;
		protected DropDownList cities;
		protected DropDownList products;
		protected DropDownList regions;
		protected Button prod_bttn;
		protected Button goCity;
		//arrays to hold the data
		protected ArrayList ctyArray = new ArrayList();//just the first time is
empty
		protected ArrayList stArray;
		protected ArrayList productsArray;
		protected System.Web.UI.WebControls.Label ob;
		protected System.Web.UI.WebControls.Label forecast;
		protected System.Web.UI.WebControls.Label foreHeader;
		protected System.Web.UI.WebControls.Label obHeader;
		protected ArrayList regionsArray;
		protected HtmlInputHidden action;
		private MegaVega.interfaces.EventHandler handler;
		protected System.Web.UI.WebControls.Label today;
		private MegaVega.interfaces.Event weatherEvent;

		private void Page_Load(object sender, System.EventArgs e)
		{
			if(!IsPostBack)
			{
				weatherEvent =
(MegaVega.interfaces.Event)Session["weatherEvent"];
				Hashtable data = weatherEvent.getReqParams();
				today.Text = DateTime.Now.ToLongDateString();
				string city = (string)data["city"];
				string state = (string)data["state"];
				obHeader.Text = "Weather Observation for "+city+"
,"+state;
				ob.Text = (string)data["ob"];
				forecast.Text = (string)data["forecast"];
```

```
                        foreHeader.Text = "Weather Forecast for "+city+"
,"+state;
                        //get data from the cache manager to populate drop downs
                        stArray = CacheManager.getInstance().getStates();
                        productsArray=CacheManager.getInstance().getProducts();
                        regionsArray = CacheManager.getInstance().getRegions();
                        //binding data to server controls
                        states.DataSource = stArray;
                        states.DataBind();
                        products.DataSource = productsArray;
                        products.DataBind();
                        regions.DataSource = regionsArray;
                        regions.DataBind();
                }
                if(states.SelectedIndex == 0)
                {
                        ctyArray.Add("Select a city");
                        cities.DataSource = ctyArray;
                        cities.DataBind();
                }

        }

        /**
         * Method to get the cities for the selected state.
         * Cities are retrieved from the CacheManager
         * */
        public void getCities(object sender, System.EventArgs e)
        {
                ctyArray = CacheManager.getInstance().
                        getCities(states.SelectedItem.Text);
                cities.DataSource = ctyArray;
                cities.DataBind();
        }
        /**
         * Method to retrieve the summary description
         * */

        public void getCity(object sender, EventArgs e)
        {
                //set next action
                action.Value = "citywx";
                handler = Redirector.getInstance().getTarget("citywx");
                handler.doProcess(Request, Response, Session);
        }
        public void getProduct(object sender, EventArgs e)
        {
                string prod = products.SelectedItem.Value.ToLower();
                handler = Redirector.getInstance().getTarget(prod);
                handler.doProcess(Request, Response, Session);
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
                //
```

```
                    // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
                    //
                    InitializeComponent();
                    base.OnInit(e);
                }

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                                    private void InitializeComponent()
                                                                    {
                                                this.Load += new
System.EventHandler(this.Page_Load);

                                    }
                #endregion
        }
}
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using MegaVega.core;

namespace MegaVega
{
	/// <summary>
	/// Summary description for Satellite.
	/// </summary>
	public class Satellite : System.Web.UI.Page
	{
		protected DropDownList states;
		protected DropDownList cities;
		protected DropDownList products;
		protected DropDownList regions;
		protected Button prod_bttn;
		protected Button goCity;
		protected System.Web.UI.WebControls.Label satHeader;
		protected System.Web.UI.WebControls.Image satImage;
		//arrays to hold the data
		protected ArrayList ctyArray = new ArrayList();//just the first time is
empty
		protected ArrayList stArray;
		protected ArrayList productsArray;
		protected ArrayList regionsArray;
		protected HtmlInputHidden action;
		private MegaVega.interfaces.EventHandler handler;
		private MegaVega.interfaces.Event weatherEvent;
		private void Page_Load(object sender, System.EventArgs e)
		{
			if(!IsPostBack)
			{
				weatherEvent =
(MegaVega.interfaces.Event)Session["weatherEvent"];
				Hashtable data = weatherEvent.getReqParams();
				string region = (string)data["region"];
				satHeader.Text = "Satellite Image for the "+region+"
region.";
				satImage.ImageUrl =(string)data["imagepath"];
				//get data from the cache manager to populate drop downs
				stArray = CacheManager.getInstance().getStates();
		productsArray=CacheManager.getInstance().getProducts();
				regionsArray = CacheManager.getInstance().getRegions();
				//binding data to server controls
				states.DataSource = stArray;
				states.DataBind();
				products.DataSource = productsArray;
				products.DataBind();
```

```
                regions.DataSource = regionsArray;
                regions.DataBind();
            }
            if(states.SelectedIndex == 0)
            {
                ctyArray.Add("Select a city");
                cities.DataSource = ctyArray;
                cities.DataBind();
            }
        }
        /**
         * Method to get the cities for the selected state.
         * Cities are retrieved from the CacheManager
         * */
        public void getCities(object sender, System.EventArgs e)
        {
            ctyArray = CacheManager.getInstance().
                getCities(states.SelectedItem.Text);
            cities.DataSource = ctyArray;
            cities.DataBind();
        }
        public void getCity(object sender, EventArgs e)
        {
            //set next action
            action.Value = "citywx";
            handler = Redirector.getInstance().getTarget("citywx");
            handler.doProcess(Request, Response, Session);
        }
        public void getProduct(object sender, EventArgs e)
        {
            string prod = products.SelectedItem.Value.ToLower();
            handler = Redirector.getInstance().getTarget(prod);
            handler.doProcess(Request, Response, Session);
        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        } #endregion } }
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;


namespace MegaVega
{
	/// <summary>
	/// Summary description for Admin.
	/// </summary>
	public class Admin : System.Web.UI.Page
	{
		private void Page_Load(object sender, System.EventArgs e)
		{
			// Put user code to initialize the page here

		}

		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP.NET Web Form
Designer.
			//
			InitializeComponent();
			base.OnInit(e);
		}

		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.Load += new System.EventHandler(this.Page_Load);
		}
		#endregion
	}
}
```

Appendix F

Instructions

Database component

1. Create database tables for web service: execute the file ws.ddl to create the required database tables.
2. Run the ws.sql file to populate the database with some default values.
3. Start the database process.

Install Tomcat. Obtain download and installation instructions from apache.org web site.

Axis Web Service

1. Copy the folder axis into the webapps folder in tomcat.
2. Add the following jars to the classpath.
3. Start an instance of Tomcat
4. Open a browser and direct it to http://localhost:8080/axis
5. Verify the WeatherService is available as one of the services deployed to axis.
6. Take note of the IP address of the machine the web service will run.

Setting up and deploying the Java application

1. Copy the weather.war file into the webapps folder of Tomcat.
2. Start an instance of Tomcat
3. Verify that s folder called weather was created by Tomcat in the webaspps folder.
4. Shut down Tomcat
5. Modify the file webapps\weather\webApplication\WEB-INF\web.xml as follows.
6. Change the entry to the application path to the machine's absolute path to the folder weather\webApplication.
7. Save the web.xml
8. Modify the variable Web Service address in the java class vega.weather.util.Constants to match the ip address of the web service.
9. Compile the java class from the same directory.
10. Restart Tomcat.
11. Ensure the web service is already running.
12. Point a web browser to http://localhost:8080/weather/megavega

Setting up the .NET application

Requires Windows 2000, or Windows XP Professional

1. Obtain and install the .NET Framework. Visit www.microsoft.com for instructions and downloads.
2. Verify IIS is installed
3. Copy the MegaVega folder to the InetPub/wwwroot folder.

4. Ensure the axis web service is running.
5. Point a web browser to http://localhost/MegaVega/Mainpage.aspx

VITA

Raquel Clark has a Bachelor of Science from Mississippi State University in Geo Sciences, 2000 and expects to receive a Master of Science in Computer and Information Sciences from the University of North Florida, May 2003. Dr. Sanjay Ahuja of the University of North Florida is serving as Raquel's project director. Raquel is currently employed as a systems programmer analyst at CSX Technology in Jacksonville, Florida and has been with the company for 11 months. Prior to that, Raquel worked 1 year as a Geographic Information Systems (GIS) programmer analyst with Engineering Methods and Applications in Jacksonville, Florida. Raquel spent eight serving in the United States Navy as a weather forecaster.

Raquel has on-going interests in object oriented software design and testing, particularly in distributed architectures. Raquel has experience in C, Java, and C# programming languages. Additionally, Raquel has strong knowledge of Java 2 Enterprise Edition (J2EE), Model View Controller (MVC), and Web Services application development. Raquel's academic work has included C, Java, SQL, XML, SOAP, CORBA, and UML. Raquel is fluent in Spanish and enjoys tennis and golf. Raquel has been married for the past 4 years.