FreeForums

Design and Implementation

Devraj Mukherjee

November 2001

Charles Sturt University
Wagga Wagga, Australia 2650

Index

Acknowledgements

Mr Barney Dalgarno and Mr Edward Stow who have helped all along the way with my project. I have learnt a great deal from their wide experience in the industry. I also thank them for giving me this opportunity of working with them and making a lot out of my final year project.

Mr Geoff Fellows for supporting with the networking side of things, such as access to Farrer.

My Parents Mr. Satyajit Mukhopadya and Mrs. Aloka Mukherjee who have given me this wonderful chance of studying here at Charles Sturt University.

My Gurus in computing Mr. Sanjay K Choubey, and Mr Nirmal K Tewari who has helped me since I was ten with my learning of computer science.

The people participating in the JBoss and J2EE list serves, who have been on constant help to me, when I went out looking for answers to my questions. I also owe to all who have put up tutorials, white papers and FAQs, which helped in my learning.

The people who have written the various books that I have read during the course of this research, which of course has been referenced at the end of this document.

I would also like to thank all the people at Sun Microsystems and JBoss who have created these world class technologies and provided all of us with a platform to program in. Thanks are also due to all the people at W3C, Microsoft and all other companies involved in the making of XML, SOAP, and Visual Basic.

There is ought to be someone who I left out in the above acknowledgements but I want them all to know that I owe it everone who has been a part of my life. *Thank you!*

Devraj Mukherjee
dm@devraj.org

Project Manager
The FreeForums Project

Introduction to The FreeForums Project

The FreeForums project started with my discussions with Mr Barney Dalgarno and Mr Edward Stow about learning Java 2 Enterprise Programming and XML. Mr Edward Stow had been thinking about writing a Forum System for a while which would be a replacement for the CSU forums and freely available.

FreeForums is a J2EE forum system with XML integration. Along with a web-based client it also features a traditional GUI based client for the Microsoft Windows platform.

The project clearly demonstrates a working example of a J2EE application, XML parsing techniques in Java and Visual Basic, XSLT, and the usage of the SOAP (Simple Object Access Protocol). Each technology used in the FreeForums project has been explained in detail with working examples from the project.

After all our discussions it turned out that I would be producing FreeForums as my final year Bachelor degree project. This report is an insight to the development process of the FreeForums project. It is also a document which will help programmers to learn how program in the Java 2 Enterprise Edition.

Objectives of this document

This document has been written with two main objectives:

1. To provide a detailed insight of the FreeForums development process, including highlighting technologies used, project management, detailed listing of source code.
2. To provide a tutorial for programming with Java 2 Enterprise Edition and related technologies.

The document assumes that you have basic Java programming knowledge, some of the concepts have not been covered in great depth but this document points to resources for further reading on the topic. It has also been assumed that you have access to a J2EE environment, preferably running the JBoss application server integrated with Apache Tomcat.

System Charter

FreeForums is an attempt to develop a forum system based upon function provided by Charles Sturt University Forums.

The software will incorporate the base features in the CSU Forums. These features include:

1. A Web interface to central server that allows many users to view, post and reply to messages, on many forums.
2. A separate client program that allows a user to connect to the server and access all new and changed messages. The client provides the viewing, posting and reply functions. The client will feature a traditional GUI interface.

Other technical features that will be investigated are:

1. Complete separation of the Model and Database functions, allowing the use of Relational DB, or OODBMS or other persistence mechanisms.
2. Investigation of XML as the interchange format between the client and server software.
3. Investigation of XML based protocols like SOAP

Particular technologies to be investigated are:

1. Server side Java Technologies like Enterprise JavaBeans, Java Servlets, Java Server Pages and Java 2 Enterprise Edition Server for Solaris, Linux and Windows 2000.
2. The Informix Cloudscape database server.
3. Implementation of the SOAP protocol for exchange information and making Remote Procedure calls.
4. Java APIs for XML parsing like DOM, JDOM and SAX.
5. Windows Application technologies like ActiveX components for XML parsing in Visual Basic, DLLs.

Alternative component architectures that are to be discussed:

These technologies will only be discussed in report as an alternative solution to the product being developed; no practical experimentation will be carried out using this technologies.

1. DCOM/ActiveX
2. CORBA
3. RMI

Background Research

Forums are electronic discussion boards, which allow electronic asynchronous communications. Forums have features like threading, attaching files with messages. Most companies and large organizations use such systems, to provide discussion areas for their products, and services and is traditionally developed in house.

Project Objectives:

To investigate and evaluate emerging technologies, in particular Java Servlets, J2EE, and XML

Deliverables to be produced:

The following is a list of all the deliverables that are to be produced during the course of this project:

Software products:

1. FreeForums Enterprise JavaBeans Server components written using Java 2
2. FreeForums web client written using Java Servlets/JSP technology

3. FreeForums desktop GUI client written in Visual Basic 6 for the Windows Family Operating Environment.

Documents:

1. Project Plan – this document, which defines the initial project plan and the expected time lines.
2. Requirements Definition (due 1st June 2001): will provide a detailed look into the analysis of the product and produce a document of the initial use cases, class diagrams, interaction diagrams, and relational database table structures. This document will reflect the requirements specified by the user and will also outline the progress of the project on that date.
3. System development documentation (due 13th November 2001):
    a. Information about the development process, design documentation.
    b. Class diagrams
    c. Interaction diagrams
    d. RDBMS table structures.
    e. An user manual for the developed system
    f. Reviews of alternative technologies
4. Technical documentation (due 13th November):
    a. Product Source Code
    b. Product implementation guide
    c. Discussion of technologies used
5. Project Evaluation documentation (due 13th November 2001)

Current System Summary:

The CSU Forums currently provides these following functions:

1. Multi-user and Multi-forum facilities
2. World Wide Web interface to listing, viewing, posting message.
3. Sorting and searching of message in the web based interface.
4. Allows a file to be attached with messages.

System Deficiencies:

The CSU Forums currently lacks:

A Desktop Client – Web clients provide you the flexibility of accessing the forums from anywhere in the world, but does not provide some of the functionality that a desktop application can provide. Thus for regular use of the forum system a desktop client seems useful and FreeForums will address that problem.

New System Constraints

Technical Assumptions:

1. Available access to a Java 2 Enterprise Edition compatible platform to run this product.
2. Access to hardware to support a product written using Java 2 Enterprise Edition

3. A RDBMS server already set-up and ready to run with the Java 2 Enterprise Platform.

Technical Constraints:

The Desktop client will only be available for the Microsoft Windows Family of operating systems.

Development Plan

The development plan of this product has been broken up into two Groups; tasks in Group one are the essential tasks, which must be completed. Group two tasks are the additional features that the FreeForums project intends to implement. Attached with this document is a Time Line sheet which outlines estimations of time for each task, it also outlines when each task is due to be delivered.

Group 1 Use cases:

USE CASE:	SET UP A NEW FORUM
GOAL:	MAKE A NEW FORUM READY FOR USE.
ACTORS:	FORUM ADMINISTRATORS

USE CASE:	ADD A NEW FORUM USER
GOAL:	ADDS A NEW FORUMS USER
ACTORS:	FORUM ADMINISTRATORS

USE CASE:	SUBSCRIBE OR UN-SUBSCRIBE A USER FROM A FORUM
GOAL:	SUBSCRIBES OR UN-SUBSCRIBES A USER FROM A FORUM
ACTORS:	FORUM ADMINISTRATORS

USE CASE:	POST A NEW MESSAGE
GOAL:	TO POST A NEW MESSAGE TO A FORUM
ACTORS:	FORUM USERS

The system will be developed based upon the Group 1 use cases. Each increment in the development will add another technology component to the system, so that “a big bang” implementation is avoided.

Group 1 Task List:

1. Servlets and domain classes development – using Java Servlets, & serialized files. No Enterprise JavaBeans are to be used in the development at this stage.
    a. Deliverables:
        i. Web based user interface
        ii. Servlets Code
        iii. Class Diagram
        iv. Interaction Diagrams for Important interactions.

        v. Project diary notes
2. Data Persistence – this step brings in the development of an Enterprise JavaBeans based development of the forum system.
    a. Deliverables:
        i. RDBMS – design documentation and table structures.
        ii. Entity Beans Cde
        iii. Session Beans Code
        iv. Servlets/JSP Code
        v. Architecture description
        vi. Project Diary
3. Multi-Forums – the above is then extend to produce a multi-forum system using Java 2 Enterprise Edition.
    a. Deliverables:
        i. Entity Beans Code
        ii. Session Beans Code
        iii. Servlets/JSP Code
        iv. Changes in Architecture design.
        v. Project Diary
4. Desktop GUI Client: this step concentrates in developing the prototype of the Desktop GUI client. The client will only have a fully function interface with a dummy data and will have no communication methods for communicating with the server software.
    a. Deliverables:
        i. Desktop client software – written in Visual Basic 6, without any communication with the server software.
        ii. Project Diary
5. XML on client: we then define the data that will be transferred between the server and the client, based on which we define out XML DTD(s) which is then implemented in the GUI Desktop client.
    a. Deliverables:
        i. Updated version of the GUI client demonstrating the ability to process XML data locally.
        ii. XML Document Type Definitions
        iii. Project Diary
6. XML handling into the server side system: this step involves the integration of XML onto the server side system, which allows the server to communicate with the client using XML data.
    a. Deliverables:
        i. Code for updated server side components.
        ii. Complete Architecture plan.
        iii. Modified version of the Servlets/JSP with added layer of XML processing.
        iv. Project Diary
7. XML on client with server: the above steps allow us to confirm that the XML data is being properly generated can be processed to the web-based interface. This step integrates the desktop client with the server side components to take XML data from the server and process it, instead of the dummy local XML data.
    a. Deliverables:
        i. Details on use of XML protocols like SOAP
        ii. Updated version of the desktop client code.
        iii. Project Diary

Group 2 Use cases:

USE CASE:	MODIFY FORUM DETAILS
GOAL:	MODIFY THE DESCRIPTION DETAILS OF AN EXISTING FORUM
ACTORS:	FORUM ADMINISTRATORS

USE CASE:	REMOVE A FORUM
GOAL:	REMOVE A FORUM, RELATED MESSAGES AND USER SUBSCRIPTIONS
ACTORS:	FORUM ADMINISTRATORS

USE CASE:	MODIFY FORUM USER DETAILS
GOAL:	MODIFY A FORUM USER PROFILE
ACTORS:	FORUM USERS, FORUM ADMINISTRATORS

USE CASE:	REMOVE FORUM USERS
GOAL:	REMOVE EXISTING FORUM USERS FROM THE SYSTEM
ACTORS:	FORUM ADMINISTRATORS

USE CASE:	REMOVE A MESSAGE
GOAL:	TO REMOVE A POSTED MESSAGE FROM A FORUM
ACTORS:	FORUM ADMINISTRATORS

The system will be developed based upon the Group 2 use cases. Each increment in the development will add another technology component to the system, so that "a big bang" implementation is avoided.

1. Adding extra features to the Forum system: these features are considered secondary to the Group 1 features. The features have been listed in the use cases above.

    a. Deliverables:
        i. Update RDBMS table structures.
        ii. Updated Source code for Enterprise JavaBeans
        iii. Updated Source code for Servlets/JSP
        iv. Updated GUI client source code.
        v. Updated design documents
        vi. Project Diary.

Software Used

- JBoss Java Application Server Version 2.2.x
- Tomcat Servlet and JSP container
- Sun's Java Development Kit 1.3.1
- Sun's Forte for Java (IDE)
- Visual Basic 6

The Prototype

The project plan describes the implementation of the prototype using Java 2, Java Server Pages and Serial files. This first step demonstrated a full working model of the system. This proved to be an important step, which clarified the requirements of implementation.

Benefits

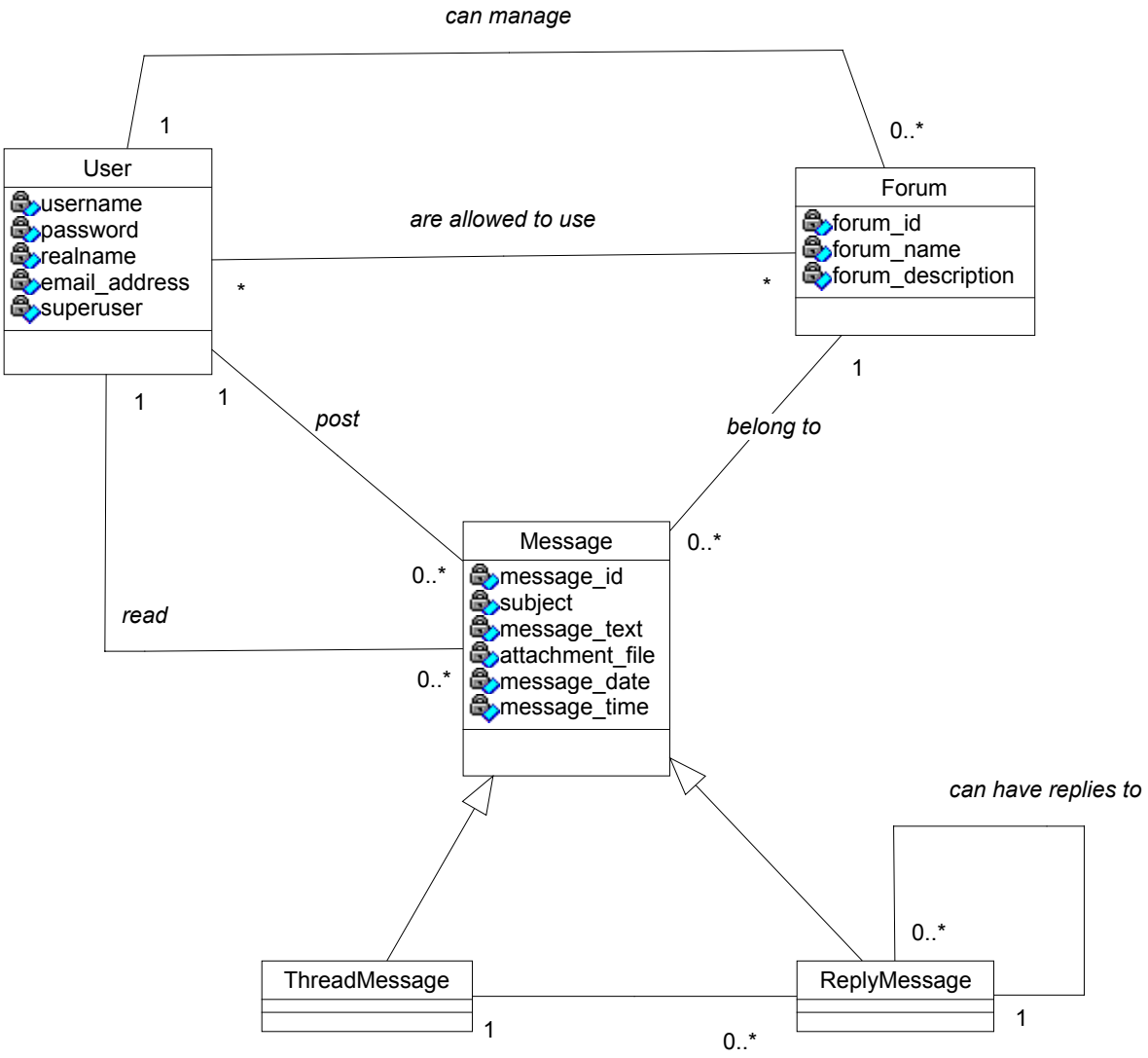
- Clarification of the system requirements
- Production of reusable code that was used later in the system.

Drawbacks in the Prototype

- The implementation of serial files: was too complicated and seemed to show a RDBMS kind of implementation. The classes could have been designed in a better fashion to do a single write and read to a binary serial file.
- Reusability: due to some design issues in the domain classes, the Java Server Pages could not be completely reused in the main implementation of the system.

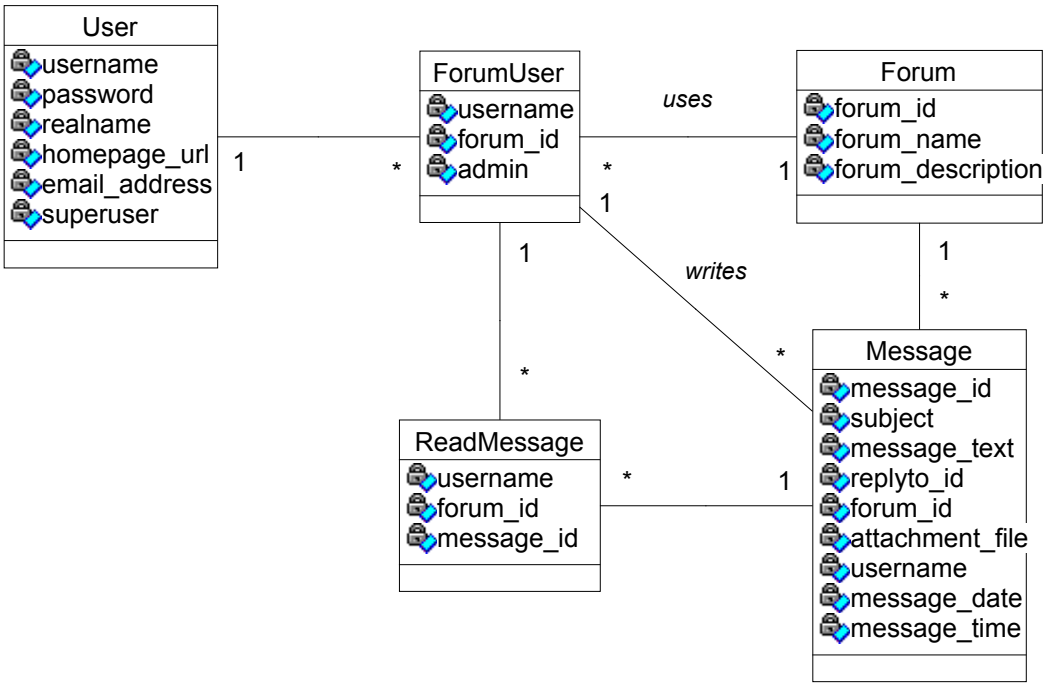
Entity Class Diagram

Original OO Model



The above diagram is the representation of entities at design time, this diagram does not consider any factors like the limitations of EJB 1.1 to purely map an Object Oriented system to a relational database model.

Original RDBMS model

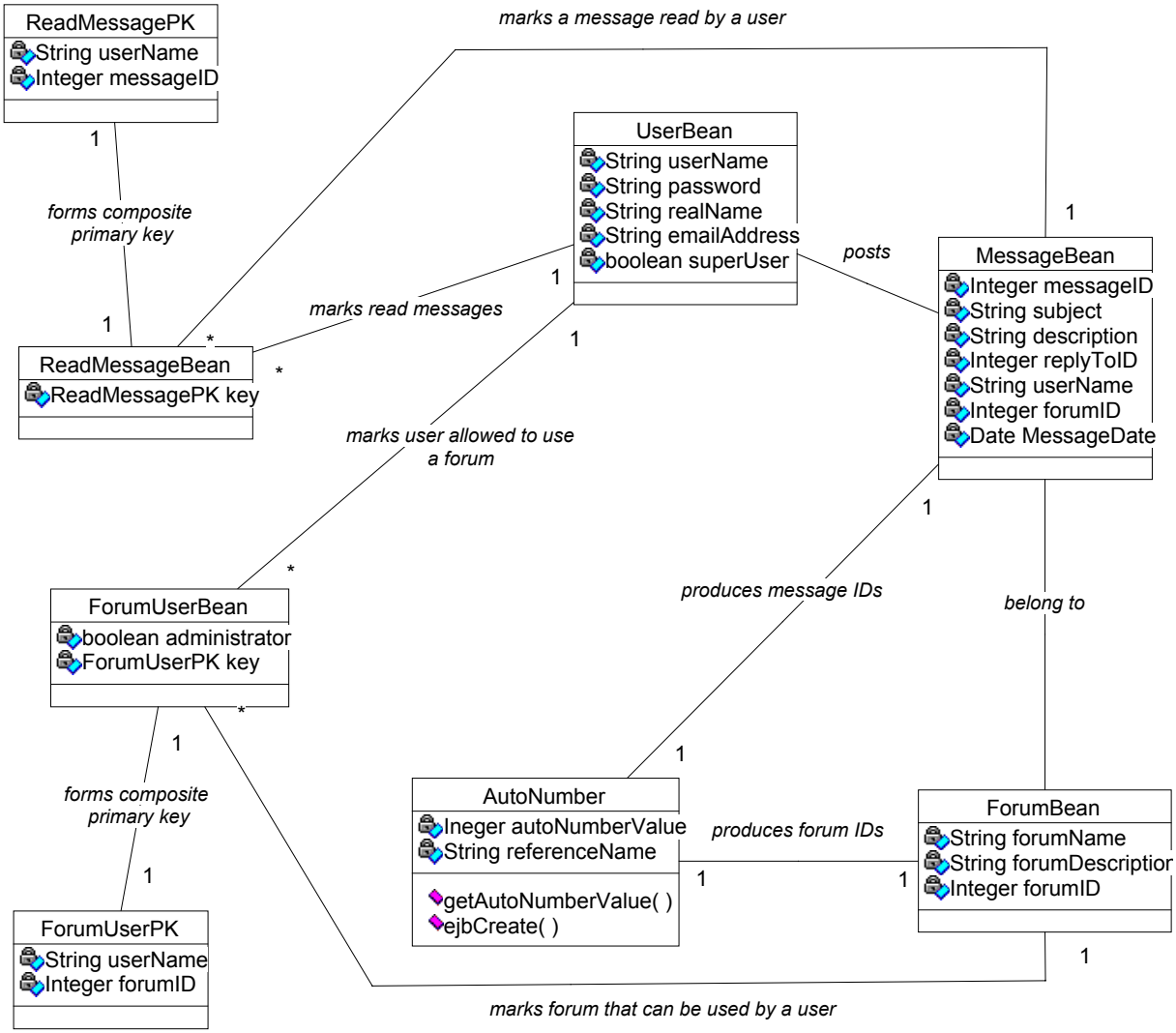


This model considers the limitations of RDBMS modeling and is thus a representation of the OO model, to RDBMS. This steps was taken because EJB maps the OO model into RDBMS to store the data.

EJB 1.1 has many liminations as mentioned, and thus the final model looked more like the RDBMS model than the OO model.

The diagram that follows represents the actual classes that exists and represent the entities used in the FreeForums system. Following that is a discussion that represents the different issues that I came across while designing and implementing the system using EJB 1.1, EJB 2.0 promises to be much more stong in mapping OO to RDBMS, a discussion about EJB 2.0's capablilities and improvements over EJB 1.1 have been outlined.

Implemented Class diagram



Discussion of design issues

The first model presented in this document, does not consider what language or technology that will be used to implement it. It follows pure OO rules. It models the entities that are required to run the FreeForums system.

The second model presented has been modelled using a RDBMS prespective; it considers the key points such as Primary keys, the rules of normalisation and also forigen keys that link the different tables.

The last of the presented Entity models is the actual implementation that FreeForums currently has. The Entities have been implemented using EJB 1.1 and are Entity Beans in the system. The Source Code can be found in Appendix E.

The EJB1.1 model does not support pure Object Oriented models to RDBMS translation. Here are some issues that have to be considered while implementing the model using Entity Beans:

- Entity Beans kinds of make the user to think in the RDBMS modelling terms, as it introduces the concepts of primary keys in an object, rather than using Object Ids.
- Writing an extra class, which encapsulates the fields that build up the primary keys, must represent composite primary keys.
- The design document available for EJB 1.1 push the fact that the data will primarily be mapped to a RDBMS model and that the design of the Entity classes should thus relfect the RDBMS model.

The Problem with AutoNumbers

AutoNumbers is the capability to generate an unique number for every record in a RDBMS table, which is then generally used as the primary key of the record.

FreeForums uses Container managed persistence; the suggested persistence mode for J2EE based applications. CMP (Container Managed Persistence) requires the programmers to declare all the attributes that will be managed by the container in an XML descriptor.

However this fails when we require numbers such as ForumID to be generated by the database management system. Most database management systems today are capable of supporting the AutoNumber feature (including Oracle, Access, MySQL) but because CMP requires the value of the fields to be set by the Session Bean that creates a new object of the Entity bean and not the database management system, this fails.

According to my discussions with the programmers on the JBoss list serve, one programmer informed me that Oracle however supports AutoNumbers with CMP with a slight configuration change in the Oracle database server configuration.

For this project I have been using InstantDB, which is packaged with JBoss as the default datastore and CMP does not support AutoNumbers with InstantDB.

To solve this problem I had to create an AutoNumber class (shown in the implemented class diagram). This class estentially keeps track of an Integer for a given key. Each key allows the programmer to relate the number to another concept or entity. Thus objects of this class can keep track of a unique number for every Entity that requires it.

The other entity beans thus expect to receive a unique id from the session beans that handle the operation of creation of objects. The Session beans thus depend on this AutoNumber class to give them the new ID provided the session bean gives them the right ID.

EJB 2.0 however takes a step futher to define a better relationship between the object and classes. For example EJB 2.0 can now handle relationships between objects as oppose to EJB 1.1, which did not have the capability to map the relationship back to RDBMS based data.

Enterprise Programming

The product of an Enterprise Programming assignment is usually more or less like a mutliteried distributed programming project. But there is something different about Enterprise Programming. In an Enteprise Programming environment like J2EE the programmers have been divided into two based on what they do best.

Business Logic programmers

Busniness logic programmers are coders who write business systems using Application Servers as their backend software. The Application server provides these developers a robust platform for producing transparent persistence models, business logic based code and even web-based applications and GUI based applications.

These programmers are mostly people who are web develoepers, business logic modellers, analysts, etc. Developers at this end do not concertrate on writing networking code, or performance tuned code. All they have to do model the business and the front end and the application server will take care of the rest.

Application Server Developers

Application Server Developers are people who manufactuer the actual development platform for J2EE, they are the people who are writing code that makes your server fast, compatible, robust, and easy to use. These people concerntrate on developing very technical code and produce a common platform so that regardless of the business that one will model this environment is going to work.

These developers must make sure that they match the standard that they follow such as J2EE. Application Servers enable a Business logic developer to develop enterprise scalable code and not be bothered by the changes that are made to the application server during the next upgrade. Because application servers follow the same standard (J2EE) – the business logic developer is totally transparent to the enterprise environment.

Benefits of Enterprise Programming environments

- Application Programmers and Business Logic programmers are totally independent of what each other are doing as each of their developments are totally transparent to the other.
- Business Logic programmers can write code once and deploy on any server that is compatible with the J2EE technology
- Business Logic implementations are very maintainable as the application server can very easily be upgraded without disturbing the application implementation
- Easy to produce robust, network based distributed applications.
- With Java based Enterprise technologies developers have the advantage of write once run anywhere.
- Persistence Transparent model available for storing data.

Java 2 Enterprise Edition

What is the Java 2 Platform, Enterprise Edition (J2EE)?

Java 2 Platform, Enterprise Edition (J2EE) is a set of coordinated specifications and practices that together enable solutions for developing, deploying, and managing multi-tier server-centric applications. Building on the Java 2 Platform, Standard Edition (J2SE), J2EE adds the capabilities necessary to provide a complete, stable, secure, and fast Java platform to the enterprise level. It provides value by significantly reducing the cost and complexity of developing and deploying multi-tier solutions, resulting in services that can be rapidly deployed and easily enhanced. [1]

What are the main benefits of J2EE? [2]

J2EE provides the following:

1. Faster solutions delivery time to market. J2EE uses "containers" to simplify development. J2EE containers provide for the separation of business logic from resource and lifecycle management, which means that developers can focus on writing business logic – their value add -- rather than writing enterprise infrastructure. For example, the Enterprise JavaBeans (EJB) container (implemented by J2EE technology vendors) handles distributed communication, threading, scaling, transaction management, etc. Similarly, Java Servlets simplify web development by providing infrastructure for component, communication, and session management in a web container that is integrated with a web server.

2. Freedom of choice. J2EE technology is a set of standards that many vendors can implement. The vendors are free to compete on implementations but not on standards or APIs. Sun supplies a comprehensive J2EE Compatibility Test Suite (CTS) to J2EE licensees. The J2EE CTS helps ensure compatibility among the application vendors,

[1] Unknown, 2001, http://java.sun.com/j2ee/faq.html
[2] Unknown, 2001, http://java.sun.com/products/ejb/faq.html

which helps ensure portability for the applications and components written for J2EE. J2EE brings Write Once, Run Anywhere (WORA) to the server.

3. Simplified connectivity. J2EE technology makes it easier to connect the applications and systems you already have and bring those capabilities to the web, to cell phones, and to devices. J2EE offers Java Message Service for integrating diverse applications in a loosely coupled, asynchronous way. J2EE also offers CORBA support for tightly linking systems through remote method calls. In addition, J2EE 1.3 adds J2EE Connectors for linking to enterprise information systems such as ERP systems, packaged financial applications, and CRM applications.

4. By offering one platform with faster solution delivery time to market, freedom of choice, and simplified connectivity, J2EE helps IT by reducing TCO and simultaneously avoiding single-source for their enterprise software needs.

What technologies are included in J2EE?

The primary technologies in J2EE are: Enterprise JavaBeans (EJBs), JavaServer Pages (JSPs), Java Servlets, the Java Naming and Directory Interface (JNDI), the Java Transaction API (JTA), CORBA, and the JDBC data access API.

JBoss

What is a Java Application Server?

A Java application server is a sophisticated software system that provide a run-time environment for executing components written in Java. The widespread adoption of the EJB standard from the J2EE specification has ensured that a well written component can be without additional development work deployed in almost all Java application servers.

There are many Java Application Servers available, which are manufactured by Oracle, IBM, BEA, Sun, Netscape and many other. Jboss seems to the primary Open Source J2EE server in use by the enterprise world.

What is Jboss?

JBoss is an Open Source, standards-compliant, Enterprise JavaBeans Application Server implemented in 100% Pure Java.The JBoss organization is working to deliver JBoss as the premier Enterprise Java application server for the Java 2 Enterprise Edition platform. JBoss will be delivered under the LGPL licence.[3]

JBoss Features

- Full EJB 1.1 support (all beans, all persistent types and all transactional tags supported)
- Partial EJB 2.0 support (Home methods and Message Driven beans are supported)
- XML compliant
- Modular design with JMX, use your own plugins, replace the Transaction engine

[3] Uknown, 2001, http://www.jboss.org/faq.jsp

- Support for JCA with JBossCX
- Support for JavaMail
- JDK1.2.2 and up support
- JNDI compliant
- JTA/JTS compliant
- JDBC compliant Container Managed Persistence
- Most database vendors work out of the box
- Easy-to-use standard configuration
- Differential metadata, easy change
- Fully modular for easy Integration
- Integrated Pool Management
- Integrated with TogetherJ
- Integrated with CastorJDO
- Integrated with CocoBase
- Integrated with JBuilder
- Integrated with Tomcat
- Integrated with JAAS for security
- Integrated with SOAP for invocation
- InVM stack optimization with Tomcat
- Optimized J2EE stack
- State of the art EAR Deployment technology
- Fast Cache technology
- Resilient and fail safe keys
- Advanced O/R mapping technology
- Automated Table creation
- Easy to use GUI
- Remote Administration
- HTTP administration
- RMI administration
- JMX compatible
- Passivating Caches

Why use Jboss?

- Dynamic "hot" deploy
- Proxy based EJBs
- Configurable container
- Small footprint
- JMX based architecture
- “Full” J2EE implementation / integration
- Large, active developer and user community
- Support for SOAP (Simple Object Access Protocol)
- Open Source
- Freeware

Integration with Apache Tomcat

The JBoss project only produces a J2EE based application server and related products, which support the application server. To handle Servlets, JSP, XSL and other web technologies, the application server depends on Apache Tomcat a Java Servlet and Java Server Pages container.

It is recommended that for commercial (serious) systems Tomcat be integrated with a commercial web server like Apache to gain performance and stability. Tomcat in the default installation runs within the same virtual machine as the application server, thus enabling the application server to have total control over the Servlet and JSP container.

Detailed documentation is available on the JBoss product web site, which describes how one can integrate JBoss with commercial web server and other Servlet and JSP containers. For this project as performance is not a major issue the default integration has been used.

Where can I get Jboss?

JBoss Servers and utilities are available for free download at www.jboss.org

Installation Requirements

- JDK 1.3
- Windows 2000, Linux or Solaris

Installation

Jboss is distributed by itself (so that it can be integrated with an existing web server like Apache, or IIS), with Apache Tomcat and Jetty. The later two versions are a ready to run solution and if one is starting from scratch one of these would be the way to go.

For this implementation Jboss with Apache Tomcat was chosen.

The installation process is as simple as downloading the archive file, which is available in ZIP, or Tarball formats. The archive has to be expanded into a directory and there are ready to run scripts that will start Jboss with or without the embedded Tomcat engine.

Deployment of applications

As mentioned above one of the major attractions for Jboss users is deployment. Assuming that you have all the XML descriptiors, Interfaces and implementations in the right place, deployment is just a matter of copying the file across to a directory under the Jboss directory tree and Jboss takes care of the download. Of course in case of errors, one would have to hunt through the log files but it is worth the trouble.

Many GUI tools are also available to help with deployment and to help with generation of the XML descriptors but writing them out manually are simple as well.

Enterprise JavaBeans

Overview

The Enterprise JavaBeans (EJB) specification defines an architecture for the development and deployment of transactional, distributed object applications-based, server-side software components. Organizations can build their own components or purchase components from third-party vendors. These server-side components, called enterprise beans, are distributed objects that are hosted in Enterprise JavaBean containers and provide remote services for clients distributed throughout the network.

Benefits of Enterprise Beans

- First, because the EJB container provides system-level services to enterprise beans, the bean developer can concentrate on solving business problems. The EJB container--not the bean developer--is responsible for system-level services such as transaction management and security authorisation.

- Second, because the beans--and not the clients--contain the application's business logic, the client developer can focus on the presentation of the client. The client developer does not have to code the routines that implement business rules or access databases. As a result, the clients are thinner, a benefit that is particularly important for clients that run on small devices.

- Third, because enterprise beans are portable components, the application assembler can build new applications from existing beans. These applications can run on any compliant J2EE server.

When To Use Enterprise Beans

- The application must be scalable. To accommodate a growing number of users, you may need to distribute an application's components across multiple machines. Not only can the enterprise beans of an application run on different machines, but their location will remain transparent to the clients.

- Transactions are required to ensure data integrity. Enterprise beans support transactions, the mechanisms that manage the concurrent access of shared objects.

- The application will have a variety of clients. With just a few lines of code, remote clients can easily locate enterprise beans. These clients can be thin, various, and numerous.

Entity Beans

An entity bean represents a business object in a persistent storage mechanism. Some examples of business objects are customers, orders, and products. In the J2EE SDK, the persistent storage mechanism is a relational database. Typically, each entity bean has an underlying table in a relational database, and each instance of the bean corresponds to a row in that table.

What makes Entity Beans different from Session Beans?

Entity beans differ from session beans in several ways. Entity beans are persistent, allow shared access, have primary keys, and may participate in relationships with other entity beans.

Persistence

Because the state of an entity bean is saved in a storage mechanism, it is persistent. persistence means that the entity bean's state exists beyond the lifetime of the application or the J2EE server process. If you've worked with databases, you're familiar with persistent data. The data in a database is persistent because it still exists even after you shut down the database server or the applications it services.

There are two types of persistence for entity beans: bean-managed and container-managed. With bean-managed persistence, the entity bean code that you write contains the calls that access the database. If your bean has container-managed persistence, the EJB container automatically generates the necessary database access calls. The code that you write for the entity bean does not include these calls. For additional information, see Container-Managed Persistence.

Shared Access

Multiple clients may share entity beans. Because the clients might want to change the same data, it's important that entity beans work within transactions. Typically, the EJB container provides transaction management. In this case, you specify the transaction attributes in the bean's deployment descriptor. You do not have to code the transaction boundaries in the bean--the container marks the boundaries for you. See Transactions for more information.

Primary Key

Each entity bean has a unique object identifier. A customer entity bean, for example, might be identified by a customer number. The unique identifier, or primary key, enables the client to locate a particular entity bean.

Container-Managed Persistence

The term container-managed persistence means that the EJB container handles all database access required by the entity bean. The bean's code contains no database access (SQL) calls. As a result, the bean's code is not tied to a specific persistent storage mechanism (database). Because of this flexibility, even if you redeploy the same entity bean on different J2EE servers that use different databases, you won't need to modify or recompile the bean's code. In short, your entity beans are portable.

In order to generate the data access calls, the container needs information that you provide in the entity bean's abstract schema.

Session Beans

A session bean represents a single client inside the J2EE server. To access an application that is deployed on the server, the client invokes the session bean's methods. The session bean performs work for its client, shielding the client from complexity by executing business tasks inside the server.

As its name suggests, a session bean is similar to an interactive session. A session bean is not shared--it may have just one client, in the same way that an interactive session may have just one user. Like an interactive session, a session bean is not persistent. (That is, its data is not saved to a database.) When the client terminates, its session bean appears to terminate and is no longer associated with the client.

State Management Modes

There are two types of session beans: stateful and stateless.

Stateful Session Beans

The state of an object consists of the values of its instance variables. In a stateful session bean, the instance variables represent the state of a unique client-bean session. Because the client interacts ("talks") with its bean, this state is often called the conversational state.

The state is retained for the duration of the client-bean session. If the client removes the bean or terminates, the session ends and the state disappears. This transient nature of the state is not a problem, however, because when the conversation between the client and the bean ends there is no need to retain the state.

Stateless Session Beans

A stateless session bean does not maintain a conversational state for a particular client. When a client invokes the method of a stateless bean, the beans's instance variables may contain a state, but only for the duration of the invocation. When the method is finished, the state is no longer retained. Except during method invocation, all instances of a stateless bean are equivalent, allowing the EJB container to assign an instance to any client.

Because stateless session beans can support multiple clients, they can offer better scalability for applications that require large numbers of clients. Typically, an application requires fewer stateless session beans than stateful session beans to support the same number of clients.

At times, the EJB container may write a stateful session bean out to secondary storage. However, stateless session beans are never written out to secondary storage. Therefore, stateless beans may offer better performance than stateful beans.

Your First EJB, JSP Example

Before we get into problem solving with EJB, I think it is a good idea to draw up a quick and simple example demonstrating the use of EJBs and JSPs.

Problem Domain:

This is more like a Hello World EJB, JSP Example. The idea here being demonstrating the use of a Session Bean (where you would store your business logic), and a Java Server Page to present the interface to the business logic. The JSP will also make use of a JavaBean, which will allow us to cut down the amount of code in the JSP file and make them more maintainable.

We want to be able to create a simple calculator, where given two numbers and an operation (we will only consider addition, multiplication, division and subtraction) the system will provide us with the result.

Some quick design notes

- We will create a Session Bean, which contain the logic of our calculator.
- We will need a JavaBean, which will be responsible for collecting information from the JSP and executing the right method in the Session Bean *(Why do we need a JavaBean? Discussed later)*
- We need a Java Server Page to be able to collect the information from the user and pass it on for processing
- A Second JSP is require to gather the data call up a Java Bean and then provide it with the information so that it can do it's job and return the result to the JSP, to be presented to the user.
- I discourage the use of primary data types, it creates a lot more work when we are trying to create Entity Beans. We can use the wrapper classes instead, for example *Integer* instead of *int.*
- Use of Packages to group our Beans – It is suggested that the Beans are grouped into packages based on the relevance. For example all the files for our calculator example would go into a package called *org.calculator*

Programming the Business Logic – Session Beans

Every Bean has at least three parts:

- The Home Interface – which describe the system level or container methods such as create to instantiate a Session bean from its home interface.
- The Remote Interface – declares the lifetime methods for a Bean, which mean that once the Bean has been instantiated these are the method that a client can use. The methods declared here have be defined in the Bean implementation.
- The Bean Implementation Class – This is Actual Implementation of the Bean, this class would implement the SessionBean or EntityBean interface based on the responsibilities of the Bean.

Unlike RMI the Remote and Home interfaces are not used by the implementation class, the EJB container is responsible for handling that side of it.

When we are programming in a true Enterprise Programming Environment, we will need to be cut off from the system level programming, so that we can concentrate on building scalable business logic code.

Naming Conventions used during EJB development

You can name your class files whatever you like because the XML descriptor defines what class, or interface is responsible for doing what. As most programming environments many similar naming conventions have been agreed upon by most J2EE programmers.
Here are some rules we will be following and a reason why:

- The Home Interface – The home interface will be called *BeanHome.java* for example for our Calculator Session Bean we will name our Home Interface *CalculatorHome.java*. This is simply because the word *Home* symbolises that this is the home interface to the Bean *Calculator*
- The Remote Interface – The remote interface for our beans will be called *Bean.java*, for example the remote interface for our Calculator bean would be called *Calculator.java*. This is because the remote interface contains the life time methods which mean if this class were to exists outside the J2EE environment it would simply be called *Calculator.java* as that would be the most logical name for a Java Class acting as a Calculator.
- The Bean Implementation Class – The Bean implementation classes will be called *BeanBean.java* for example our Calculator implementation would be called *CalculatorBean.java*

The Remote Interface

The remote interface defines the life time methods, or when the object of this Bean is instantiated these are the methods that will be available to the user. These methods will have to be declared in the Bean implementation class for it all to work.

Remote Interface extend from *EJBOject* and all the methods that are declared throw *RemoteException* and *FinderException (if a finder method is involved – finder methods are covered later)*

```
package org.calculator;

import javax.ejb.EJBObject;
import javax.ejb.*;
import java.rmi.RemoteException;

public interface Calculator extends EJBObject {

  public Integer add(Integer firstNumber, Integer secondNumber) throws RemoteException;

  public Integer subtract(Integer firstNumber, Integer secondNumber) throws RemoteException;

  public Integer multiply(Integer firstNumber, Integer secondNumber) throws RemoteException;

  public Integer divide(Integer firstNumber, Integer secondNumber) throws RemoteException;

}
```

The Home Interface

For the Home interface for our Session Bean CalculatorBean we only need one method in our home interface, which is *create()*. This method allows the Session Bean to be created after it has been looked up in the JNDI name space.

The Home interface extends from *EJBHome*. The *create()* method will need to return a proper object of the Bean, which in the J2EE environment is defined by the remote interface thus the return type of the *create()* method is set to the Remote Interface.

For our *CalculatorBean* the Home interface can be defined as follows:

```
package org.calculator;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface CalculatorHome extends EJBHome {

        public Calculator create() throws RemoteException, CreateException;

}
```

The Bean Implementation Class

The Bean implementation class implements the SessionBean Interface, which tells the EJB container that this is a Session Bean. Let's have a closer look at the methods of the Bean implementation class.

```
package org.calculator;

import java.rmi.RemoteException;
import javax.ejb.*;
import javax.naming.*;
import javax.rmi.*;
import java.rmi.ServerException;

public class CalculatorBean implements SessionBean {

/**
 Empty method body
 */
 public CalculatorBean() {}
/**
 Empty method body
 */
 public void ejbCreate() {}
/**
 Empty method body
 */
 public void ejbRemove() {}
/**
 Empty method body
*/
 public void ejbActivate() {}
/**
 Empty method body
 */
 public void ejbPassivate() {}
/**
 Empty method body
 */
 public void setSessionContext(SessionContext sc) {}

/**
 These methods are the implementations of method declarations made in
 The remote interface
 */
```



```
 public Integer add(Integer firstNumber, Integer secondNumber) throws RemoteException {
   return (new Integer(firstNumber.intValue()+secondNumber.intValue()));
 }

 public Integer subtract(Integer firstNumber, Integer secondNumber) throws RemoteException {
   return (new Integer(firstNumber.intValue()-secondNumber.intValue()));
 }


 public Integer multiply(Integer firstNumber, Integer secondNumber) throws RemoteException {
   return (new Integer(firstNumber.intValue() * secondNumber.intValue()));
 }


 public Integer divide(Integer firstNumber, Integer secondNumber) throws RemoteException {
   return (new Integer(firstNumber.intValue()/secondNumber.intValue()));
 }

} // end implementation class
```

The XML descriptor (ejb-jar.xml)

The XML descriptor for Enterprise Java Beans must be placed in a directory called *META-INF* and the file must be called *ejb-jar.xml.* This file describes to the Java Application Server what each one of these classes and interfaces do. The DTDs for these XML documents are provided by Sun and all application servers that follow the J2EE standard must follow it.

ejb-jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
"http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">

<ejb-jar>

  <description>Sample EJB Application</description>
  <display-name>Calculator</display-name>

  <enterprise-beans>

    <session>
      <description>Models a Calculator</description>
      <ejb-name>CalculatorBean</ejb-name>
      <home>CalculatorHome</home>
      <remote>Calculator</remote>
      <ejb-class>CalculatorBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>

  </enterprise-beans>

</ejb-jar>
```

If we have a closer look at this XML file it is quite simple. This section will be discussed in much greater detail when we build our model application further down the document.

Everything in this XML file is contained under the <*ejb-jar*> container tags. Under this tag we use the <*description*> tag to describe what this whole EJB Jar file represents. The <*display-name*>
Within the <*enterprise-beans*> tags we define every Enterprise Java Bean that will be used in the application. Obviously they can be anywhere on you machine because they are referred to by their packages.

To the client all that matters is the JNDI name (discussed later) and rest of the information about the beans is built from there. This XML file tells the server where each piece of these beans are and which JNDI name they are associated with.

Session beans are defined using the <*session*> tag and Entity Beans using the <*entity*> tags. We will discuss Entity beans when we are writing our sample application. Let's now have a closer look at the contents of the Session Bean definition:

<*description*> again allows us to give a descriptive name to the beans, these could be very useful while writing a distributed application over a network and where developers are working independently in distant locations.

<*ejb-name*> allows us to define the JNDI name in the JNDI namespace. In our case we are using the default namespace which is *java:comp/env/ejb/* and we wish the client software (in this case the JavaBeans) to refer to this bean with the name *CalculatorBean.*

The next three tags define where the server can find the remote interface, home interface and the ejb-class implementation for this bean. Remember that the classes and interface are located using the packages they belong to, needless to say the packages must be included in the *CLASSPATH* environment variable.

<*home*> defines where the server can find the home interface, which is in this case is *org.calculator.CalculatorHome*

<*remote*> defines where the server can find the remote interface, which in this case is *org.calculator.Calculator*

<*ejb-class*> defines where the EJB implementation can be found, in this case *org.calculator.CalculatorBean*

<*session-type*> As you now know that there are two types of Session Beans you must define this parameter for every session bean that you define in the application. Our calculator bean is a Stateless bean.

<*transaction-type*> Inside the J2EE environment you can ask the EJB Container to manage the Bean (recommended) or you can write the code to manage it, in which case it is said to be managed by the Bean. It is recommended that the Container manage the transaction wherever possible, due to many security, performance and portability issues.

This section should have given you a fair idea as to what is required to be defined in the XML descriptor, we will discuss a bit more about this when we write our sample application. When writing big applications it is recommended that you use a XML editor for the ease of editing. JBoss provides an Enterprise JavaBeans XML editor, see the resources section for a list of helper tools.

Packaging the EJB

In a J2EE application the EJBs are packaged in a jar file. Jar files are created using the jar utility distributed with the Java Development Kit. Jar files follow the ZIP algorithm to

compress the files. This is done to increase portability of the application. The whole application is packaged into one jar file, which is transported over the server. The server then looks into the Jar file for a descriptor, which then allows the server to determine the installation process for the application.

It is important that we get all the class files and ejb-jar.xml from the META-INF directory. You must preserve the directory structure for META-INF.

A sample jar command execution would look like

```
jar cfv calculator.jar org\calculator\*.class META-INF
```

The above command would take all the class files in the org\calculator directory along with the ejb-jar.xml file in the META-INF directory and package it as calculator.jar

The Client View of the Session Bean

As we are in the scenario of writing distributed applications, the client view of these classes is not quite the same as normal class files. The client application only has access to the RMI-IIOP code for the Enterprise Java Beans.

Client application to Enterprise Java Beans only need to know of the JNDI name so that they can locate the Home interface and execute methods from it. Once the Home interface has been successfully located, the client can execute the create() method to create a object of that class. Doing so the Remote method is ported over and it defines the actual methods available off that object for execution.

The Home interface of the beans can differ from Entity to Session beans. Different methods with have different effects depending on if they are an entity or a session bean. For example: the create method would create a copy of the session bean for the client calling it, but if the create method was executed for an Entity Bean then that would result in creation of a new entry in the persistence storage or in other words a new record in the backend RDBMS table.

So, What is the client view?

The client view is the way an Enterprise Java Bean appears to the Enterprise Java Bean, JSP, JavaBean or client side application using it. Session Beans are clients to Entity Beans and our JavaBeans (used to support our JSP) are clients of Session Beans.

Please read the chapter *"Naming with JNDI"* for further information on the naming conventions to be followed while writing Enterprise applications with Java.

What is JavaServer Pages technology?

JavaServer Pages (JSP) technology provides a simplified, fast way to create web pages that display dynamically-generated content. The JSP specification, developed through an industry-wide initiative led by Sun Microsystems, defines the interaction between the server and the JSP page, and describes the format and syntax of the page.

How does the JavaServer Pages technology work?

JSP pages use XML tags and scriptlets written in the Java programming language to encapsulate the logic that generates the content for the page. It passes any formatting (HTML or XML) tags directly back to the response page. In this way, JSP pages separate the page logic from its design and display.

JSP technology is part of the Java technology family; it uses a Java programming language-based scripting language, and JSP pages are compiled into servlets. JSP pages may call JavaBeansTM components (beans) or Enterprise JavaBeansTM components (enterprise beans) to perform processing on the server. As such, JSP technology is a key component in a highly scalable architecture for web-based applications.

JSP pages are not restricted to any specific platform or web server. The JSP specification represents a broad spectrum of industry input.

Why do I need JSP technology if I already have Servlets?

JSP pages are compiled into Servlets, so theoretically you could write Servlets to support your web-based applications. However, JSP technology was designed to simplify the process of creating pages by separating web presentation from web content. In many applications, the response sent to the client is a combination of template data and dynamically-generated data. In this situation, it is much easier to work with JSP pages than to do everything with Servlets.

Supporting JSP files with Java Beans

Java Beans are not Enterprise Java Beans. Java Beans are used to support applications with GUI components, easing the load of JSP and so on. A list of good articles on Java Beans can be found on http://java.sun.com/products/javabeans/

Here is a quick list of reasons why one should consider using Java Beans in conjunction with Java Beans:

- Reuse of code written in the Java Beans across the different Java Server Pages.
- Java Server Pages usually don't provide a lot of debug information when they crash, Java Beans is a good way eliminating some of these errors by pre checking the working of methods inside them.
- JSP files are less bulky, when the code is removed from the JSPs and put into the Java Beans; this also means that the compiling time for the JSPs into the intermediate Servlet for execution is less.

Java Beans can be used in many different ways. When developing Java Beans for SWING applications they also act as ActiveX components. That respect of Java Beans is out of the scope of this document. Java Beans in respect to Java Server Pages look like normal Java Classes with just a few rules. The constructors of Java Server Pages should not take in any parameters. They should contain getters and setter methods to do most of their work.

JSPs need not necessarily use Java Beans using the JSP XML styled tags. They can also be used as normal class components to part of the work for the JSP.

To makes things clearer let's have a look at a real life example. Going back to our Calculator example, we are going to develop a Java Bean to support the JSP front end of the system. Our Java Bean is going to address the following problems:

- All data that comes from a web based document is recognized as String, this class will take String input and convert it into the right data types before passing it the EJB for processing.
- Provide reusable methods for getting access to EJB, this will include JNDI calls to the Session Beans, etc.
- One single method will take the two numbers and a process instruction from the web based form and then returns a result based on the selection.

For simplicity the Java Bean will also belong to the package org.calculator of course this could go into another Java Package, which is recommended for bigger applications.

```
package org.calculator;

import javax.naming.*;
import java.util.Hashtable;
import javax.rmi.PortableRemoteObject;
import java.util.Properties;

public class CalculatorHelper extends Object {

        // Empty Constructor

        public CalculatorHelper() { }

        // Method to help add two number using the Session Bean

        public Integer processNumbers(String numberOne, String numberTwo, String process) {

        try {
                //Get the home interface for the Calculator Bean using JNDI
                CalculatorHome home = (CalculatorHome) new
                InitialContext().lookup("CalculatorBean");
                // Create the  Calculator Bean object using the Home interface
                Calculator ourCalculator = home.create();

                if(process.equals("ADD"))
                return (ourCalculator.add(new Integer(numberOne), new Integer(numberTwo)));

                if(process.equals("SUBTRACT"))
                return (ourCalculator.subtract(new Integer(numberOne), new
                Integer(numberTwo)));

                if(process.equals("DIVIDE"))
                return (ourCalculator.divide(new Integer(numberOne), new Integer(numberTwo)));

                if(process.equals("MULTIPLY"))
                return (ourCalculator.multiply(new Integer(numberOne), new
                Integer(numberTwo)));

                return null; // if nothing was selected.

        } // end Try here.

        catch(Exception e) {
                // do nothing much
                return null;
        } // end Catch here

        } // end processNumbers here
```



```
} // end Java Bean here
```

As discussed in the above documentation, the `processNumbers` method makes the call to the JNDI interface, locates the EJB, creates it and then executes the right method based on the selection

```
//Get the home interface for the Calculator Bean using JNDI
CalculatorHome home = (CalculatorHome) new InitialContext().lookup("CalculatorBean");

// Create the  Calculator Bean object using the Home interface
Calculator ourCalculator = home.create();
```

The above two lines of code, is responsible for locating the Calculator Bean in the default JNDI namespace, getting the home interface and then creating the EJB object based on the home interface that was downloaded.

The next section will demonstrate how easy it is now to write the JSP to perform a calculation using our Calculator EJB.

Writing the JSP files

Before we start writing the JSP files, we must ensure that the path where your Java Bean and EJB Interface classes have been placed must be included in the CLASSPATH environment variables of the application server.

For Jboss this can be achieved by editing the `run.bat` or `run.sh` file depending on the platform.

A simple demonstration of the system can be achieved by creating an HTML file that presents the user with a form and a JSP, which gathers the data and sends it to the Java Bean for processing.

Here is the source to a simple HTML based form for the Calculator system:

```
<html>
<head>
<title>Calculator Example</title>
</head>
<body>

<form name="calculator" action="jsp/calculator.jsp" method="POST">
<font face="Verdana" size=2>

<b>Calculator Example</b><br><br>

First Number: <input type="text" size="4" name="firstNumber"><br>
Second Number: <input type="text" size="4" name="secondNumber"><br><br>

<select name="action">
 <option value="ADD">Add</option>
 <option value="SUBTRACT">Subtract</option>
 <option value="MULTIPLY">Multiply</option>
 <option value="DIVIDE">Divide</option>
</select>

<input type="submit" name="submit" value="Calculate"><br>

</font>
</form>

</body>
</html>
```


Now we will create a JSP file to gather the data submitted by the form above and then process it using the Java Bean. The JSP has been commented for you to understand what part of the JSP does what.

It is recommended that you put all the JSP files in a sub directory called JSP

```
<! --- JSP Tag to import the helper classes -→
<%@ page import="org.calculator.CalculatorHelper" %>

<%

// This section is called a Scriplet and will make use of the Calculator Java Bean
// to process the data.

// Create the Java Bean object

CalculatorHelper helper = new CalculatorHelper();

// Get the information from the form

String firstNumber = request.getParameter("firstNumber");
String secondNumber = request.getParameter("secondNumber");
String action = request.getParameter("action");

// Calculate the result

Integer result = helper.processNumbers(firstNumber, secondNumber, action);

%>

<html>
<head>
<title>Result</title>
</head>

<body>
<font face="Verdana" size=2>

<! ---------- Print the result to the new HTML page ---→

The result is <%= result %>

</font>
</body>
</html>
```

The XML descriptor (web.xml)

The XML descriptor for web applications on a J2EE system defines things like Custom tags that are to used by the JSP or any EJB that have been referenced by a JSP web application. This is a very simple application, with all components on the same server we don't need to specify any details in a web application description file.

We will have a closer look at web.xml further in the document where we use XSLT and where we see the need of integrating Apache frameworks for XSLT into our applications.

web.xml must be placed under a directory called WEB-INF. Here is an example of a blank web.xml file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
```



```
</web-app>
```

Packaging the web components (WAR)

WAR files are JAR (ZIP algorithm based) as well and they can be packaged using the jar utility provided with JDK. Here is a sample jar command to package WARs. This assumes that the HTML file is called index.html and is placed under the root directory your web files.

```
jar cfv calculator.war *.html jsp WEB-INF
```

EAR – Enterprise Archives

Enterprise Archives are made of an EJB Jar file and a Web archive or WAR file. The contents of the file are described by application.xml (described in the next section of the document). EARs are self installable Enterprise Applications which are ready for distribution for one or more Java based application server.

The two files (WAR and JAR) must be made using the JAR utility (should follow the ZIP algorithm). application.xml is placed in a subdirectory called META-INF.

The XML descriptor (application.xml)

The application.xml is quite simple to understand. The application tag contains it all, meaning the information between a application container tag contains information about a single enterprise application.

The module tag defines a web or EJB module for a application. Let's have a quick look at the applicatio.xml file below, the first module tag defines the web module. Contained inside a web container tag it defines the WAR file to look into using the web-uri tag and the context root of the web application using the context-root application

The second module tag defines the EJBs, simple as it look just specify the name of the EJB JAR file and let ejb-jar.xml file define the rest.

```
<application>

        <display-name>Calculator Test Release</display-name>

        <module>
                <web>
                <web-uri>calculator.war</web-uri>
                <context-root>/calculator</context-root>
                </web>
        </module>

        <module>
                <ejb>calculator.jar</ejb>
        </module>

</application>
```

Packaging the Enterprise Archive Application

Again a simple JAR command will make the EAR file, similar to what we have done for creating the EJB JAR and WAR file.

```
jar cfv calculator.ear *.jar *.war META-INF
```

Deploying under JBoss

One of the major attractions for a Jboss user is it's deployment. Jboss has an auto deployment mechanism. A directory under the Jboss installation is constantly monitored by the Jboss application server, as soon as the developer copies a new version of the file across to that directory, Jboss takes charge and deploys the application.

Thus is will be a matter of copying or FTPing the file across to that directory depending if you are running the Jboss server on the same machine or a remote machine.

The messages produced during the deployment of applications are logged as well displayed on the console for debugging.

The default directory for deployment is `\Jboss\jboss\deploy`

Naming with Java Naming and Directory Interface (JNDI)

What is LDAP?

Lightweight Directory Access Protocol (LDAP) is a software protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet. LDAP is a "lightweight" (smaller amount of code) version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network. LDAP is lighter because in its initial version it did not include security features. LDAP originated at the University of Michigan and has been endorsed by at least 40 companies. Netscape includes it in its latest Communicator suite of products. Microsoft includes it as part of what it calls Active Directory in a number of products including Outlook Express. Novell's NetWare Directory Services interoperates with LDAP. Cisco also supports it in its networking products.

In a network, a directory tells you where in the network something is located. On TCP/IP networks (including the Internet), the domain name system (DNS) is the directory system used to relate the domain name to a specific network address (a unique location on the network). However, you may not know the domain name. LDAP allows you to search for an individual without knowing where they're located (although additional information will help with the search).

An LDAP directory is organized in a simple "tree" hierarchy consisting of the following levels:

- The root" directory (the starting place or the source of the tree), which branches out to
- Countries, each of which branches out to
- Organizations, which branch out to

- Organizational units (divisions, departments, and so forth), which branches out to (includes an entry for)
- Individuals (which includes people, files, and shared resources such as printers)

An LDAP directory can be distributed among many servers. Each server can have a replicated version of the total directory that is synchronized periodically. An LDAP server is called a Directory System Agent (DSA). An LDAP server that receives a request from a user takes responsibility for the request, passing it to other DSAs as necessary, but ensuring a single coordinated response for the user.

What is a namespace?

A Namespace defines valid names that can be used in a system, many Internet and network related technologies follow the concept of a Namespace to allow a systematic representation of information on the system.

LDAP and JNDI both follow namespaces to allow publishing of LDAP names for various Enterprise JavaBeans thus representing a complete Enterprise Programming.

What is JNDI?

The Java Naming and Directory Interface (JNDI) is a standard extension to the Java platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. As part of the Java Enterprise API set, JNDI enables seamless connectivity to heterogeneous enterprise naming and directory services.

Why is JNDI important in J2EE?

When writing Enterprise wide applications, directory services do the same job as our Yellow pages would do for such a widely distributed telephone network. As we have seen in our investigation of J2EE that creating and defining the enterprise components are fairly simple, but this seems to be an important concept that people tend to miss out.

JNDI lets you locate the remote and home interfaces in a JNDI name space, thus enabling you to create remote object over the network. Once the application is up and running all this will be purely transparent. An enterprise application can be very distributed, so much so that two EJBs are located in two different machines on a network. JNDI provides a naming database, which allows developers to define Human Readable names references to the EJBs. These names will allow the seeking of the remote and home interface of the classes and thus in turn allow creation of remote objects of the EJBs.

The ejb-jar.xml file contains the JNDI definitions for ever EJB that we write in the system. It is important that we know how the names of our EJBs are referred to while searching the JNDI namespace.

java:comp/env/ejb is the default namespace defined by JNDI. For a simple EJB application like ours we are better off using the default namespace to store references to our EJBs.

Naming components in the JNDI namespace

Let's take an extract from our ejb-jar.xml file for one of our EJBs.

```
<session>
  <description>Models a Calculator</description>
  <ejb-name>CalculatorBean</ejb-name>
  <home>CalculatorHome</home>
  <remote>Calculator</remote>
  <ejb-class>CalculatorBean</ejb-class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
</session>
```

The ejb-name tag in the description file determines what name the Enterprise Bean will take in the JNDI namespace. Of course the name cannot be a duplicate one. If we just provide a name as we have done in the above description file the EJB is added to the default namespace `java:comp/env/ejb`. All client programs searching for EJBs must use a fully qualified JNDI name. If you are however adding your beans to the default namespace then you may just use the bean name to search for it.

Locating and Creating EJB objects

```
//Get the home interface for the Calculator Session bean
CalculatorHome home = (CalculatorHome) new InitialContext().lookup("CalculatorBean");

// Create the Calculator Object object
Calculator enterpriseCalculator = home.create();
```

The above two lines of Java Code enables a client program locate the Calculator EJB (added to the default JNDI namespace) and create an object of it.

The first line of Java Code enables the client program to locate the EJB in the default JNDI namespace. If and when the EJB is found in the JNDI namespace the home interface is transported over the client and a new instance of it is created.

This home interface will now allow us to execute methods like create() to get a new instance of the EJB. The second line demonstrates that. The create() method from the home interface is being executed to create a new object of the Session Bean CalculatorBean.

Once this process is completed, the enterpriseCalculator object to the client will appear to be a normal Java object and you can execute methods, as you would do with a normal Java object. Remember that the methods that can be executed by the client are only those methods that have been defined in the remote interface of the EJB.

The same rules hold when a Session Bean creates an instance of an Entity Bean. The only difference holds to the results of the different methods. The later part of this document where we discuss the creation of FreeForums using EJB describes how.

XML

What is XML?

XML (Extensible Markup Language) is a universal format for structured documents and data on the World Wide Web. XML is a W3C supported standard. The initial goals of XML were to provide a flexible markup language, which could be customized for one's own data representation purpose. Today XML is used for many other purposes other than data representation such as transformation of documents (discussed later).

XML is often described as "the ASCII of the Web"[4] You can use your favourite programming language to create an arbitrary data structure and then share it with anyone using any other language on any other computing platform. XML tags name the concept you are describing and named attributed modify the tagged structures. So, you can formally describe the syntax, you have decided and share it with others.

Take the idea a bit further, XML documents can also be used to conduct what is referred to as B2B (Business 2 Business) transactions. The problem in conducting automated business between two organizations has been the implementation of their computing technologies. As every company is likely to have a different implementation and is likely to use different technologies it is very hard to agree on a common data format, or even a common data structure.

XML is the answer to all these problems. Does not matter what platform you operate on, does not matter what web server is in use, does not matter what database servers you use. As long as your front end programs can produce XML and all business participating in the B2B transaction agree on one XML data format, you are up and running with B2B solutions.

There are already many widely agreed standards of XML being used for representation of scientific data, business data, and limited configuration device synchronisation data. One of the most interesting projects is SyncML, agreed upon by many manufacturers of limited configuration devices like Mobile Phones, PDAs, etc. This standard allows XML representation of personal information data, which can be used for synchronizing two devices. More information regarding this project is available on www.syncml.org

FreeForums and XML

FreeForums uses an XML layer to represent all incoming and outgoing data to the business logic layer. The Java Server Pages front end uses XML transitions to provide the web based interface for the system. All information that is to be passed to the business logic objects to be acted on, are also encoded in XML.

This allows developers to extend FreeForums by developing customized desktop clients. The XML front end will allow clients written in any language to gather data from the FreeForums server objects, thus allowing integration of FreeForums with any software that supports the XML descriptions defined by this project.

A use of this has been demonstrated by creating a web based as well as desktop interface for the FreeForums system.

[4] Dider Martin, et al. *Professional XML*, Wrox Press, Pg 11

Why use XML?

The features of XML are what attract a developer to use XML. This section provides a detailed discussion of the features that XML provides to a developer and an application. Hopefully be the time you finish reading this section you will be able to appreciate the uses of XML and how it can make an application powerful, scaleable and expandable.

Document Type Definitions (DTDs)

DTDs define the rules that are followed by an XML document; these rules allow us to define XML documents that have verified data. Applications can thus the data with much confidence and the parsers can thus validate data and be sure that the information been passed on to the application is of the standard desired by the program.

Validation is a very important feature when writing B2B applications where you will trust other software vendors to write software that will communicate with yours. DTDs also facilitate error checking in the common data format XML.

Data Modelling

A vocabulary in the XML world is the specification of elements, their attributed and the structure of the documents that conform to a developers plan. An effective vocabulary describes the relevant objects in your problem in the way that is expressive of the problem while permitting easy access to your data.

There are no hard and fast rules of modelling data with XML, but they are usually dominated by the design factors considered your OO or RDBMS modelling.

Document Object Model (DOM)

DOM is one of the APIs available and used as a standard API for XML parsing. DOM based APIs will produce a three structured view of the XML document A DOM compliant parser reads the entire document and offers a view of the document by constructing a tree of object in memory.

Simple API for XML

SAX is the other major API that is used for XML parsing. Unlike DOM, SAX is not a product of a standards organization. It is a ad hoc product of a number of XML developers who needed an effective API early in the development of XML.

SAX remains quite popular due the way it approaches the parsing of XML documents. SAX is an event-based parser, where it fires events as is reads the different elements of the document. This allows the program not to load up the entire document into memory; this has proven to be quite effective for large amount of XML data being parsed a program.

Transforming XML

With technologies like XSL, XSLT we can transform XML into any other XML styled language like HTML. This allows us to generate XML based data and then transform the output based on the device that is requesting the output.

XML transformation has been used in FreeForums and it demonstrates how easy it becomes with the use of such a technology to extend an application to other devices like mobile phones, PDAs or even extend it to talk to desktop applications.

XML and Databases

Most RDBMS and OO DBMS now support representation of data in XML. Given a set of XML data they can add validate and add the data to the tables. This is a great push for developers to use XML. The XML output generated by the database servers can then be used directly and presented to users using transformation technologies like XSLT.

B2B (Business to Business)
XML provides the bridge for B2B. In most B2B cases the two companies have established IT infrastructure and changing the whole infrastructure just because of B2B is nearly out of the question. XML provides the answer, does not matter what platform you use and language your software was written in, XML provides validated, well-formed data which can be used at either end by either applications.

Learning by example is the best way. This document now explores how XML can be used in an Enterprise application and explore what differences it makes to it.

XML Documents

XML documents use HTML styled tags to represent data, the rules of document however can be defined by the author of the document. DTDs (Document Type Definitions) define the rules that an XML document is going to follow. Appendix B contains a listing of the DTDs used in the forum system.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserList SYSTEM "http://devraj.org/dtd/freeforums/User.dtd">

<UserList>

        <User>

                <UserName>devraj</UserName>
                <Password>1234</Password>
                <RealName>Devraj Mukherjee</RealName>
                <EmailAddress>dm@Devraj.org</EmailAddress>
                <SuperUser>Yes</SuperUser>

        </User>

        <User>

                <UserName>vinayak</UserName>
                <Password>1234</Password>
                <RealName>Vinayak Mukherjee</RealName>
                <EmailAddress>vinayak@Devraj.org</EmailAddress>
                <SuperUser>No</SuperUser>

        </User>

</UserList>
```


The above XML document has been extracted from the FreeForums systems, as it suggests it describes a list of users. Let us have a closer look at the document and get to know a few things about XML documents before we continue our discussion about XML and XML based services.

- The first line describes to the client that this is an XML document encoded using the first version of the XML specification and uses the UTF-8 encoding
- The second lines then goes on a tells the client about this XML document, it says that this XML document has a parent element called UserList and the rules of this document is described by the DTD User.dtd which can be found at http://devraj.org/dtd/freeforums/
- The Document then goes onto describing the data using the tags defined in the DTD.
- The UserList tag contains a List of Users, each user then contains information a user such as username, real name, password, etc.
- Each tag is a container tag, which essentially contains data and makes up the data model.

Document Type Definitions

As described below the Document Type Definitions (DTDs) defines the rules that a document follows, this is very important because data that is passed across from application to application or from one part of the application to the other must be valid.

XML sounds to be the answers of the B2B programming prayers. Most organizations today that are doing business electronically have some sort of computer system in place that satisfies their business needs. B2B is about bridging businesses so that their information systems can be collaborated and information can be shared.

With these systems in place, and two businesses agreeing on a common format for data exchange it is important that the data that is passed across the wire from business to business can be validated and before the data is processed and used.

Appendix B lists all the document type definitions that have been used in the FreeForums system. Listed below is the User.dtd followed by a discussion of modelling DTDs.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Devraj Mukherjee (ETNET) -->
<!ELEMENT UserName (#PCDATA)>
<!ELEMENT RealName (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!ELEMENT EmailAddress (#PCDATA)>
<!ELEMENT SuperUser (#PCDATA)>
<!ELEMENT User (UserName+, Password+, RealName+, EmailAddress+, SuperUser+)?>
<!ELEMENT UserList (User+)+>
```

DTDs are described using XML styled tags as well. That is one of the beauties of the XML technology set everything compliments each other.

Before we start discussing the XML DTD, it is important to know the thought that went into modelling this DTD. This DTD describes a list of users, which is made up of Users. Users inturn encapsulate data about the user.

This the structure of the XML document that follow the rules defined by this XML DTD would look like:

```
<UserList>
        <User>
                <!----- all the other elements here ------→
        </User>
</UserList>
```

- The first line of course says this document is described using XML version one and that this document uses the UTF-8 encoding.
- The second line a comment inserted by the XML editor I was using at the time.
- The next five lines then describe each of the sub elements of the user element. If we are going to use a DTD for a document and request a parser to validate the document, then these elements must be defined before they are used.
- The next line then defines the User element, which is made up of the five elements that we have defined above. When creating data that is based on this DTD we must keep in mind that the data should appear in the same order as define in the User Element.
- The last element is the UserList and it defines that the document may contain one UserList Element, which can in turn be made of User Element.

XML Documents necessarily don't have to follow rules defined by DTDs, you can create XML documents that don't use a DTD at all. It is however it is a good idea to do so, as it will eliminate all problems of validation of the data we are using.

The Professional XML book (included in the references list) has good chapters on modelling data in to XML DTDs. A in length discussion of the whole data modelling exercise is out of the scope of this document. However later down the document discussions about how the data was modelled for this project have been included.

Parsing XML

XML parsing using Java has been made quite simple by developers around the world. Many projects lead under the different hoods have been widely accepted by programmers as standard APIs for XML parsing using Java.

The three most prominent are DOM (Document Object Model) controlled by W3C, SAX (Simple API for XML parsing) and JDOM (Java Document Object Model) based on the DOM processor but extended to give more functionality.

DOM Parsers

The Document Object Model protocol converts an XML document into a collection of objects in your program. You can then manipulate the object model in any way that makes sense.

This mechanism is also known as the "random access" protocol, because you can visit any part of the data at any time. You can then modify the data, remove it, or insert new data.

A detailed description of the DOM parser can be found at
http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.pdf

SAX Parsers

This API was actually a product of collaboration on the XML-DEV mailing list, rather than a product of the W3C. It's included here because it has the same "final" characteristics as a W3C recommendation.

You can also think of this standard as the "serial access" protocol for XML. This is the fast-to-execute mechanism you would use to read and write XML data in a server, for example. This is also called an event-driven protocol, because the technique is to register your handler with a SAX parser, after which the parser invokes your callback methods whenever it sees a new XML tag (or encounters an error, or wants to tell you anything else).

JDOM

JDOM is, quite simply, a Java representation of an XML document. JDOM provides a way to represent that document for easy and efficient reading, manipulation, and writing. It has a straightforward API, is a lightweight and fast, and is optimized for the Java programmer. It's an alternative to DOM and SAX, although it integrates well with both DOM and SAX.[5]

JDOM documents can be built from XML files, DOM trees, SAX events, or any other source. JDOM documents can be converted to XML files, DOM trees, SAX events, or any other destination. This ability proves useful, for example, when integrating with a program that expects SAX events. JDOM can parse an XML file, let the programmer easily and efficiently manipulate the document, then fire SAX events to the second program directly - no conversion to a serialized format is necessary.

For this project we have chosen to use the JDOM API for creating and working with XML documents, the examples listed later in this document use functionality from the API to do the complex parsing work.

JDOM can be fetched from www.jdom.org, before we start cutting code using the API we must ensure that the JAR file is included in all our CLASSPATH variables.

Creating XML documents using Java

Before we find out how we parse XML documents it is important to know how we can create proper XML documents using the JDOM API. The following example gives a simple example demonstrating how one can use the API to create XML documents.

```
import org.jdom.*;
import org.jdom.output.XMLOutputter;
```

[5] Anonymous, JDOM FAQ, http://www.jdom.org/docs/faq.html


```
import java.util.*;


public class XMLTest {

        public static void main(String [] args) {

        Element root = new Element("Introduction");

        Element main = new Element("Greeting");
        main.addContent(new Element("Message").addContent("Hello"));
        main.addContent(new Element("From").addContent("Devraj"));
        main.addContent(new Element("To").addContent("World"));

        Element main2 = new Element("Greeting");
        main2.addContent(new Element("Message").addContent("How are you"));
        main2.addContent(new Element("From").addContent("World"));
        main2.addContent(new Element("To").addContent("Devraj"));

        root.addContent(main);
        root.addContent(main2);

        //root.addChild(new Element("Message"));

        Document doc = new Document(root);

        Element rootElement = doc.getRootElement();

        List childrenElements = rootElement.getChildren();

        Element [] Children = (Element[])childrenElements.toArray(new Element[0]);

        Element message = (Children[1].getChild("Message"));

        String text = message.getText();
        System.out.println(text);

        XMLOutputter output = new XMLOutputter();

        try {
         output.output(message, System.out);
        }
        catch (Exception e) {
         System.err.println(e);
        }

 } // End main method

} // End class file
```

The code published above is a Test XML class which creates an XML document using the JDOM API and then outputs the document to the screen.

```
        Element main = new Element("Greeting");
        main.addContent(new Element("Message").addContent("Hello"));
        main.addContent(new Element("From").addContent("Devraj"));
        main.addContent(new Element("To").addContent("World"));
```

The above code sinppet creates a new Element object called greeting and then adds three sub elements called Message with the content Hello, From with the content Devraj and To with the content World.

Element object may have more than one addContent method and the content gets appended to the Element. Elements may contain Elements or just text as their content. Be ware that if you specify a DTD for the Document object then the document will be verified and if the document created does not follow the rules defined in the DTD then exceptions are thrown accordingly.

We will have a closer look at creating XML document that are verified using the XML API.

```
<Greeting>
<Message>Hello</<Message>
<From>Devraj</From>
<To>World</To>
</Greeting>
```

Would then be the result of the above code. The whole program goes on to create, two of these elements and adds them to the root element called Introduction.

The root element is then used to create the document object defined in the JDOM API. The later half of the code, then extract the root element of the document, converts the sub elements into an array of Element object and outputs the second element in the array to the screen using the XMLOutputter class.

Parsing XML documents using Java

JDOM provides a comprehesive API using, which XML documents can be parsed, validated and traveserd through. The first example had an insight about creating XML documents using the JDOM API, this section concerntrates on parsing XML documents using the JDOM API.

Most examples in the FreeForums system passes in and expect to receive Document objects, which contain fully, formed (validated or unvalidated) XML documents. The XML document's root element is then extracted from the document object and then a List of Element objects is created. Each Element object can then be accessed like a normal Java Object which is then able to give back a list of elements or the values of it's attributes.

Consider the following code extract from MessageManagerBean.java, where MessageInformation is a Document object

```
Element RootElement = MessageInformation.getRootElement();

List MessageListElement = RootElement.getChildren();
Element [] MessageElements = (Element[])MessageListElement.toArray(new Element[0]);

String subject = MessageElements[0].getChild("Subject").getText();
```

The first line is used to the extract the RootElement of the Document into an Element object. The next two lines then convert all the sub elements of that RootElement into a List of element objects and then subsequently into an Array.

The fourth line of code then demonstrates how data contained in one of the sub tags of each of these Element tags is parsed and stored in a variable. The document the four lines of code is able to parse follows the Message.dtd and looks somewhat like

```
<MessageList>

        <Message>

                <Subject></Subject>

                <!------------- Sub Elements ---------->

        </Message>
```



```
</MessageList>
```

By the same token, as we can create an Array of Elements we can thus loop across all the Element objects using a simple for or while loop and perform operations on them. Considering that we have MessageElements already defined above, the loop would look like:

```
for(int counter=0; counter<MessageElements.length; counter++) {

        Element currentElement = MessageElements[counter];

}
```

The above does the simplest task of extracting an element from the Array and assigning it to the currentElement object. The section "Generating HTML documents from XML documents using JDOM" shows a much more complex example that uses recursion to produce a list of messages for display as the message list.

Integrating XML in an Enterprise Application

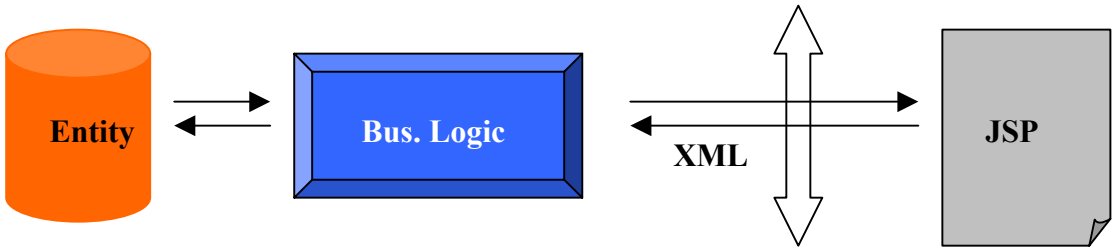
Enterprise Applications is what is referred as an *n-tier* application. Generally the value of *n* is three, the first layer being the database, or persistence layer. The second is business logic layer, which represents the working of the business. The third being the presentation layer, usually a web based interface or a desktop client.

FreeForums has been coded using the three tier model, The persistence layer is represented by the Entity Beans which take care of storing information in a RDBMS based persistence model and make it transparent to the rest of the Enterprise Application.

Session Beans in a J2EE application represents the Business Logic, all the logic behind running the forum system resides here. The Entities are treated as objects and the session beans don't have to be worried about what database is being used, counting records or executing SQL statements, the entity beans will take care of them.

The presentation layer in FreeForums is represented by the Java Server Pages and the Java Beans that help the JSP cut down complexity. The idea of having a separate presentation layer is that designers can concentrate on producing fancy pages without having to play with a lot of code. There are many editors / IDE (Integrated development environments) available in the market (such as Macromedia Ultradev) which give a perfect blend for designers and developers.

Usually the presentation layer communicates to the session beans using normal Java language parameters, sending and receiving objects of entities to gather or present data. XML does a good job here, if the data that is send between the presentation layer and the session beans is converted into XML, our application becomes very expandable.



The above diagram represents the integration of XML in the FreeForums system. The entity beans remain untouched and there is nothing special about them in terms of adding XML. The session beans are the only components of the application that is going to interact directly with the Entity Beans, so integration of XML there is not required.

The session bean layer (business logic layer) communicates with the external presentation layers, which in this case are JSP and Java Beans. The messages passed between the JSP and session beans are nothing else but XML Document objects created using the JDOM parsing API. This enables a document to be passed to the JSP and the presentation layer delivering it to the user the way it is required.

Later down this document we are going to study the way XSL has been used in FreeForums, which will demonstrate the power of data being passed using XML.

Java Server Pages and XML working together

There are many ways of using XML with JSP. The JSP can request the XML object and directly display the XML content for use by another JSP or application to the client that made the request. The other possible use of JSPs getting XML data parsing through it and producing output, and the third use would be transformation.

Java Server Pages technology supports XML very well. Considering that JSP has many built libraries for XML parsing and for XSLT based tasks, it is quite easy to use XML with JSP. The later sections explore the use of XSLT, and other parsing techniques in detail to demonstrate how XML can be used with JSP.

XSLT - Transforming XML data

One of the other wonders of XML is transformation, and that too very easily using XSLT. Transformation is very important when modelling any kind of data. For example CSV (Comma Separated values) is a standard way of distributing data when we are talking about RDBMS styled data.

It provides the common data format, which can be used by any RDBMS to convert data. Most financial software has adopted CSV and thus has made it into a standard. The point being that data format conversion is very important. And of course the quick, the easier, the better.

XML based data can thus be transformed from one XML based data format to another very easily, nearly without writing any program code. The XML data usually has to be generated by a program or a database server but the transformation process is simple and easy to use.

XSLT uses an XML based language (part of the beauty of XML related technologies) to define the XSL sheets, which can then be used by the XSLT frameworks. XSLT frameworks are available from Apache, IBM, etc, for most languages and they do everything there about XSLT. All we have to do is use a few easy to use tags in our HTML pages to instruct the frameworks to convert the XML data into HTML

Defining Style sheets using XML

XSL style sheets are based on the descriptors provided by W3C, as mentioned before that XSL sheets are defined using XML styled tags. XSL provides the programmer with a lot of power to control the flow and presentation of data. XSL processors process the XML data with respect to a XSL style sheet and insert the output where the XSLT tag was placed in the page. An application can be seen later down in this document.

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="ForumList">

<table width="80%" border="1">

 <tr>
 <td><font face="Verdana" size="2"><b>Forum Name</b></font></td>
 <td><font face="Verdana" size="2"><b>Description</b></font></td>
 </tr>

 <xsl:for-each select="Forum">
 <tr>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="ForumName"/></font>
  </td>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="ForumDescription"/></font>
  </td>

 </tr>

 </xsl:for-each>

</table>
</xsl:template>
</xsl:stylesheet>
```

The above style sheet has been extracted from the FreeForums system and it helps to produce a list of forums given an XML data set. For this XSL style sheet to perform the data set provided to it must follow the rules defined by Forum.dtd (Appendix B)

The above XSL sheet, starts by defining what this file itself it defined in and then goes on to defining what the ouput of this XSL process is going to be. XSLT can be used to convert XML styled data to any other XML styled lingo, it does not have to be XML to HTML. In a B2B implementation this would be an ideal tool if XML data generated by a business had to be converted into lingo that is used in common by it's partners.

`<xml:template match="ForumList">` instructs the XSLT processor to select the main tag in the XML data description. The next few lines are then defined in HTML, which are send to the ouput stream as is. An important point here is that any HTML tags that are embeeded into the XSL cannot contain dynamic data as the value of it's attributes. An example later in this section discusses generating dynamic values as attributes of a HTML tag.

`<xsl:for-each>` instruts the XSLT processor to go through a loop and output data in the following format for each item that starts with the `Forum` tag. `<xsl:value-of>` can then be used to extract the value encoded in a XML tag in the data and can then be used in the output being produced by this XSLT process.
Generating dynamic data as attributes of a HTML or XML tag is quite a necessary feature. XSLT does support this but it requires certain changes in the XSL style definition. Consider the following extract from the file userforumlist.xsl, which helps the FreeForums system generate a list of Forums for a given user.

```
<xsl:element name="A">

 <xsl:attribute name="HREF">
  show_messages.jsp?forum_id=<xsl:value-of select="ForumID"/>
 </xsl:attribute>

 <xsl:value-of select="ForumName"/>

</xsl:element>
```

The first tag in this extract allows us to define a tag using the XSL libraries, and using the first line of code we can generate the `<A href>` HTML tag to provide a link.

The second line then goes on to add an attribute to the A HREF tag we defined in the first line and sets the value of the attribute to `show_messages.jsp?forum_id=` and then current forum id that can be extracted from the XML data.

The next tag then inserts the data between the starting and the closing `<a href >`tag. Thus now producing a output of `<a href="show_messages.jsp?forum_id=0">EJB Forum</a>` if the forum id was 0 and the name was EJB Forum

The above demonstrates how a new XML styled tag can be produced using XSL, which contain dynamic data extracted out of an XML data set that is given to this XSL file at runtime. Important notes, the tags that are generated must be complete meaning they must be closed if started otherwise this might effect the way XSL processes the XML data.

JSP Tagextentions

Tag extentions are XML styled tags embedded in a JSP page. They however have a special meaning to a JSP page engine at translation time and enable application functionality to be invoked without the need to write Java code in JSP scriplets. Well-designed tag extention libraries can enable application functionality to be involed without the appearance of programming.

The JSP framework allows creating JSP Taglibs quite easily and incoroporating it into a web based Java application. A JSP taglib can be defined using by extending the TagSupport class include in the JSP Packages.

```
import java.io.IOException;
import javax.servlet.jsp.*;
import javax.Servlet.jsp.tagext.*;

public class SimpleTag extends TagSupport {

        public int doStartTag() throws JspTagExtention {
```



```
                // executed when the JSP processor finds the first tag

        }

        public int doEndTag() throws JspTagExtention {

                // executed when the JSP processor finds the end tag

        }

} // End Tag Lib extention
```

A set of these tags that serve a certain purpose like XSLT processing is refered to as a Tag library or a taglib. In a J2EE web application, the taglib can then be placed under the directory WEB-INF, Classes and then referenced by being declared in a Tag Library Descriptor which is placed under the tlds directory under the root directory of the web application.

The tablib then has to be declared in the web.xml file placed in the META-INF directory so that it is available to all the JSP files that are in that current web archive. The following is a code extract from web.xml to demonstrate how a tld is described:

```
<taglib>
        <taglib-uri>/simple</taglib-uri>
        <tablib-location>/WEB-INF/tlds/hello.tld</tablib-location>
</taglib>
```

For performance reasons, JSP engines will not necessarily instantiate a new tag handler instance every time a tag is encountered in a JSP. Instead they may maintain a pool of tag instances, reusing them where possible.

When a tag is encountered in a JSP, the JSP engine will try to find a Tag instance that is not being used, initialise it, use it and then release it (but not destroy it), making it available for further use. The programmer has no control over any pooling that may occur.

The repated use model is similar to a Servlet lifecycle, but one very important difference is that tag handler implementations don't need to concern themselves with thread safety. The JSP engine will not use an instance of a tag handler to handle a tag unless it is free. This is good news: as with JSP authoring in general developers need to worry about threading issues less often than when developing Servlets.

The Apache Framework for XSLT

One of the frameworks available for the Java platform is the Apache taglib for JSP. This framework uses the Apache Xerces and Xalan APIs to parse and process the XSLT instructions along with the XML data.

This taglib adds to the set of JSP tags on the system and thus allows the JSPs on the system to process XSLT instructions just by including XSLT tags. Appendix D contains the XSLT taglib documentation from the Apache web site.

To run the Apache framework for XSLT there pieces of software are required to be present in the development environment.

- A servlet container that supports the JavaServer Pages Specification, version 1.1.
- The Apache Xerces XML parser.
- The Apache Xalan XSL processor.

Follow these steps to configure your web application with this tag library:

- Copy the tag library descriptor file (xsl/xsl.tld) to the /WEB-INF subdirectory of your web application.
- Copy the tag library JAR file (xsl/xsl.jar) to the /WEB-INF/lib subdirectory of your web application.
- Copy the support libs xerces.jar and xalan.jar to the /WEB-INF/lib directory of your web application.
- Add a <taglib> element to your web application deployment descriptor in /WEB-INF/web.xml like this:

```
<taglib>
<taglib-uri>http://jakarta.apache.org/taglibs/xsl-1.0</taglib-uri>
<taglib-location>/WEB-INF/xsl.tld</taglib-location>
</taglib>
```

To use the tags from this library in your JSP pages, add the following directive at the top of each page:

```
<%@ taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="x" %>
```

where "x" is the tag name prefix you wish to use for tags from this library. You can change this value to any prefix you like.

Using XSLT with JSP

Given that we have now had a look at defining XSL style sheets using the XML styled language, and we have also had a look at JSP Taglibs and XML styled data, it is now time to put it all into practice.

The code extract below is from the file forum_list.jsp (Appendix F), which produces a list of Forums for administration purposes. To allow the JSPs to use the libraries defined the Apache tablib project we must include this line onto the top of the JSP page so that it contains a reference to the Taglibs we are about to use. A complete listing of the commands included included in the Apache taglib are contained in Appendix D.

```
<%@taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="xsl" %>
```

The code extract below uses the `<xsl:apply>` tag from the Apache Taglib for XSLT. The XMLOuputter class is used to output the XML document that is been retervied from the Session Beans, the Taglib is then capable of taking the data that is present between the `<xsl:apply>` and `</xsl:apply>` tag and process it using the XSL stylesheet present in the attribute list of the tag.

```
<xsl:apply xsl="/xsl/forumlist.xsl">

<%

 Document XMLDocument = (new ForumController()).getCompleteForumListing();
 XMLOutputter output = new XMLOutputter();
```



```
 try {
       output.output(XMLDocument, out);
 }
 catch (Exception e) {
       System.err.println(e);
 }

%>

</xsl:apply>
```

FreeForums and XML

FreeForums has been developed in three basic stages, one which was the prototype, the second being using Enteprise Java Beans and third being integration of XML. During the development, quite a lot of code has been reused and it has been ensured that the end user does not feel the difference.

The section “Integrating XML in an Enteprise Application” talks about how Session Beans and XML are integrated. FreeForums largely follows the same principles. The Entities remain the same throughout the system mostly because of the analysis that has been put into the system and because only the Session Bean interacts with it.

The Session Beans are the real face of the system to the presentation layers such as the Java Server Pages based application that runs on the web and if CORBA applications were to be written, the Session Beans would have been the client view of the system.

The other part of the project at the design stage has considered development of a desktop client written using Visual Basic 6 and SOAP. SOAP which is an XML based protocol can be generated using XSLT as the Session Beans already send back XML output for the requests send by the presentation layer.

A central Servlet or JSP can thus be the hub for receiving the requests from the desktop client and for processing the SOAP based requests. Most application servers are very capable of processing SOAP based requests, as it is a W3C accepted standard.

FreeForums uses the JDOM API to represents all XML based documents and uses the Apache XSLT framework in conjunction with Java Server Pages to do all the XML parsing and transformation. There are some performance issues with JBoss but according to the benckmark tests that have been performed on the industry standard application servers’ performance is not a concern for application servers like Websphere and Weblogic.

With commercial database servers like Oracle it is also possible to deal with XML on the entity level, thus making applications much strongly integrated with XML. J2EE will allow customisation of it’s applications quite easily to use these database servers thus managing integration of the XML at the entity level. This no doubt requires the system design and code to be changed but the fact remains that it is possible for large-scale systems.

Generating HTML documents from XML documents using JDOM

The original idea in the project was to use XSLT's recursion feature to display a list of message for a given Forum. This idea was finally dropped for two reasons:

1. The process of learning recursion was proving to be too time taking and due to time contrainsts on the project it was dropped.
2. All XML data converted into HTML in the FreeForums system was using XSLT and thus I thought it was important to include an example that used the JDOM API to convert an XML document to HTML

Recursion based algorithms usually are fairly complex, however this is a perfect example of rescursion and XML to HTML transformation using the JDOM API. Consider the following source code extract from MessageController.java

```
////////////////////////////////////////////////////////////////////////////////////////////
//
// The following part of this program is the complex code that produces the threaded list of
// messages. The code has been adopted from the previous versions of the same project.
//
//
////////////////////////////////////////////////////////////////////////////////////////////

        private int depth;

        /**
         *
         * Given a Parent Message ID this will count the number of submessages
         *
         **/

        private int countSubMessages(Element MessageElement, Document MessageList) {

                String ParentMessageID = MessageElement.getChild("MessageID").getText();

                List MessageListElement = MessageList.getRootElement().getChildren();
                Element [] MessageElements = (Element[])MessageListElement.toArray(new
                Element[0]);

                try {
                        int countMessages=0;
                        for(int cntr=0; cntr<MessageElements.length; cntr++)
                        if(MessageElements[cntr].getChild("ReplyToID").getText().equals
                        (ParentMessageID)) countMessages++;
                        return countMessages;
                }
                catch(Exception e) {
                        System.out.println("countSubMessage() failed\n" +e);
                        return 0;
                }

        }
```

The above method helps in counting the number of sub messages in the thread given an Element of Message and a Document that contains a Message List (refer to Message.dtd). The defined process is very simple whereby the method breaks down the Message Document into an array of Element objects (which contains information about Messages) and then counts the number of sub messages by running through the for loop extract the details for every Message Element.

```
/**
 *
 * Given a Message Element this method will format it as HTML and return the Element
 *
 * This is done so that no web specific output is generated in the Entity or Session Beans
 *
 * This Class is a web specific class so we can generate some HTML here.
 *
```



```
 **/

        private Element formatMessage(Element MessageElement, String UserName) {

                Element RowElement = new Element("tr");

                String MessageID   = MessageElement.getChild("MessageID").getText();
                String Subject     = MessageElement.getChild("Subject").getText();
                String Author      = MessageElement.getChild("UserName").getText();
                String MessageDate = MessageElement.getChild("MessageDate").getText();
                String ForumID     = MessageElement.getChild("ForumID").getText();

                // Check if message is read

                Element ReadMessageStatus = new Element("ReadMessage");
                ReadMessageStatus.addContent(new Element("UserName").addContent(UserName));
                ReadMessageStatus.addContent(new Element("MessageID").addContent(MessageID));

                Document ReadStatusResult = isReadMessage(new Document(ReadMessageStatus));

                // Make HTML tags

                RowElement.addContent(new Element("td").addContent(MessageDate));

                Element ReadMessage = new Element("A");
                ReadMessage.addAttribute("HREF","read_message.jsp?message_id=" + MessageID +
                "&forum_id=" + ForumID);
                ReadMessage.addContent(Subject);

                Element ImageTag = new Element("IMG");
                ImageTag.addAttribute("SRC","../images/spacer.gif");
                ImageTag.addAttribute("WIDTH",""+depth);
                ImageTag.addAttribute("HEIGHT","1");

                Element SubjectCol = new Element("td");
                SubjectCol.addContent(ImageTag);

                if(ReadStatusResult.getRootElement().getText().equals("UnRead")) {

                        Element Bold = new Element("B");
                        Bold.addContent(ReadMessage);
                        SubjectCol.addContent(Bold);

                }
                else SubjectCol.addContent(ReadMessage);

                RowElement.addContent(SubjectCol);

                RowElement.addContent(new Element("td").addContent(Author));

                Element PostMessage = new Element("A");
                PostMessage.addAttribute("HREF","post_message.jsp?reply_to=" + MessageID +
                "&forum_id=" + ForumID);
                PostMessage.addContent("Reply");

                RowElement.addContent(new Element("td").addContent(PostMessage));

                return RowElement;

        }
```

The above method formatMessage, which accepts a Message element and a user name is responsible for chaning a Message encoded in XML into HTML, what it sends back is an Element object that contains HTML code which can then be appended into the the main HTML element manage by the two methods defined and discussed after this.

These steps defined are fairly trivial for any task that builds around a similar problem. The first few lines as you can make out extracts all the details from the Element object and stores them into individual variables.

According to the specifications of the system as and when the user reads messages the system must record it and produce a display of message list according to the status of the message for the user. The next few lines then clearly demonstrate creation of the table row element using the JDOM API. The example clearly shows the use of the setAttribute method as well as adding content multiple times to the Element object.

The Element is finally send back to the parent method and as a result added to the list of messages formatted in HTML.

```
/**
 *
 * Parent method to generate the HTML
 *
 **/

public Document getMessagesForForum(Document ForumUserDetails) {

        Element MessageTableElement = new Element("table");
        MessageTableElement.addAttribute("width","100%");
        MessageTableElement.addAttribute("cellpadding","0");
        MessageTableElement.addAttribute("cellspacing","0");

        Element HeaderRow = new Element("tr");
        HeaderRow.addAttribute("bgcolor","#FFFFCC");

        Element Column1 = new Element("td");
        Column1.addAttribute("width","10%");
        Column1.addContent("Date");
        HeaderRow.addContent(Column1);

        Element Column2 = new Element("td");
        Column2.addContent("Subject");
        HeaderRow.addContent(Column2);

        Element Column3 = new Element("td");
        Column3.addAttribute("width","10%");
        Column3.addContent("Author");
        HeaderRow.addContent(Column3);

        Element Column4 = new Element("td");
        Column4.addAttribute("width","10%");
        Column4.addContent("Reply");
        HeaderRow.addContent(Column4);

        MessageTableElement.addContent(HeaderRow);

        String ForumID  =
        ForumUserDetails.getRootElement().getChild("ForumID").getText();
        String UserName =
        ForumUserDetails.getRootElement().getChild("UserName").getText();

        try {
                //Get the home interface for the user management bean
                MessageManagerHome home = (MessageManagerHome) new
                InitialContext().lookup("MessageManagerBean");
                // Create the manager object
                MessageManager manager = home.create();
                //Get the array
                Document MessageListDocument = manager.findMessagesForForum(new
                Document(new Element("ForumID").addContent(ForumID)));

                List MessageListElement =
                MessageListDocument.getRootElement().getChildren();
                Element [] MessageElements = (Element[])MessageListElement.toArray(new
                Element[0]);

                for(int cntr=0; cntr<MessageElements.length; cntr++) {
                        Element currentMessage = MessageElements[cntr];
                        if(currentMessage.getChild("ReplyToID").getText().equals("-1"))
                        {
                                depth=0;
```

```
                                MessageTableElement.addContent
                                (formatMessage(currentMessage, UserName));
                                listSubMessages(MessageTableElement,MessageListDocument,
                                currentMessage.getChild("MessageID").getText(),
                                UserName);
                        }
                }

                return (new Document(MessageTableElement));
        }
        catch(Exception e) {
                //something went wrong while making the HTML
                //quite unlikely that we would get this exception
                System.out.println("getMessageForForum failed \n" + e);
                return (new Document(new Element("Response").addContent("Failed")));
        }

} // end method
```

The above method is the parent method called by the JSP to generate the list of messages; this method has two major responsibilities. The first being connecting to the Session Bean and getting the Document object containing the list of Messages for a given Forum and the second being running through a list of parent messages and calling the listSubMessages method to add to the HTML message list.

The method starts by defining the Table Elements using the JDOM API. This is done simply by creating Element tags and adding them in order that they should be in. The second half of the method concertates on getting a Document object from the Session Bean, extracting the list of messages and processing the list of parent messages. For every message that is read and added to the HTML table element, the listSubMessages method is executed thus adding the all the other messages in the thread to the HTML element.

```
/**
 *
 * Converts the messages to HTML format
 * This method is a helper method
 *
 **/

private void listSubMessages(Element MessageTableElement, Document
MessageListDocument, String ParentMessageID, String UserName) {

        List MessageListElement = MessageListDocument.getRootElement().getChildren();
        Element [] MessageElements = (Element[])MessageListElement.toArray(new
        Element[0]);

        try {
                Element currentMessage=null;
                int countMessages=0,messageIndex=0;
                depth+=8;

                for(int cntr=0; cntr<MessageElements.length; cntr++) {
                        currentMessage = MessageElements[cntr];
                        if(currentMessage.getChild("ReplyToID").
                        getText().equals(ParentMessageID)) {
                                countMessages++;
                                messageIndex=cntr;
                        }
                }

                if (countMessages==0) return;
                if (countMessages==1) {
                        currentMessage=MessageElements[messageIndex];

                        MessageTableElement.addContent(formatMessage
                        (currentMessage,UserName));
```

```
                        if(countSubMessages(currentMessage, MessageListDocument)>0)
                        listSubMessages(MessageTableElement, MessageListDocument,
                        currentMessage.getChild("MessageID").getText(), UserName);
                        depth-=8;
                        return;
                }
                ////////////////////////////////
                // If more than one message then //
                ////////////////////////////////
                for(int cntr=0; cntr<MessageElements.length; cntr++) {
                        currentMessage=MessageElements[cntr];

                        if(currentMessage.getChild("ReplyToID").getText().equals(ParentM
                        essageID)) {

                        MessageTableElement.addContent(formatMessage(currentMessage,User
                        Name));

                        // List sub messages for this reply message
                        if(countSubMessages(currentMessage, MessageListDocument)>0)
                        listSubMessages(MessageTableElement, MessageListDocument,
                        currentMessage.getChild("MessageID").getText(), UserName);
                        }
                }
                ////////////////////////////////
                // End more than one message //
                ////////////////////////////////
                depth-=8;
                return;
        }
        catch(Exception e) {
                //Something wrong happened above, quite unlikely.
                System.out.println("listSubMessages() failed \n " + e);
                return;
        }
}
```

Recursion is refered to a method calling itself. listSubMessages as descibed above, is the method that uses recursion to add all the sub messages in the given thread to the HTML element being produced as a result of the two above methods running. This method is based on very much the sample principles of getMessagesForForum with the expection that it uses recusion and that it depends on getMessagesForForum to call it with the parameters.

Notice the code in bold that refers to the code that has handles the recusion process primarily. As the MessageTableElement object is passed on to this method this method thus uses this objet and adds the HTML code using the JDOM API to this object. As listSubMessages has a reference to this object it is modifed in turn of this method modifing this document.

The depth variable, which is a class level shared variable between getMessagesForForum and listSubMessages is used to mark the width of the spacer gif, which denotes the indentation when the HTML is presented, it could have been passed as a variable to the methods each time but has been declared at the class level just for convinence sake.

As the JSP Beans don't model entities or business logic code it seem perfectly allright to break some of the OO rules when writing code in the beans to support the JSPs.

FreeForums and J2EE

The Objective:

The objective of developing FreeForums using the J2EE technology is to achieve total separation of the persistence, business logic and presentation layer. This version of the system will look into producing a system using EJB, JSP and JavaBeans - and implement the Group 1 tasks (Refer to the project plan document for details on Group 1 tasks).

Available Information:

- Project Plan with Group 1 task list
- RDBMS design for tables.

Scenario:

The prototype of this system looked into implementing a forum system, which could handle only a single forum, using Java Server Pages and Domain classes, which much looked like Java Beans. The data was stored using serial files.

After developing the system, some design issues were pointed out:

- The way the classes were designed seemed to have represented RDBMS tables, each domain class had a collection class associated with it, which wrote to separate files for each collection of objects. As the design was faulty the implementation was fairly complicated and the three separate tiers could not be distinguished.
- The developed code could not be reused due to design issues.

The Approach:

1. The RDBMS tables design was taken into consideration and they were implemented using Container managed Entity Beans, these represent a collection of objects with default finder methods which allow finding the objects using one attribute which can be declared as the primary key.
2. The business logic of the system has been implemented using stateless session beans. The client will not access the Entity Beans directly at all. They are presented with the session beans, which provide method that do more meaningful things from the client's point of view.
3. JavaBeans and Java Server Pages make up the web presentation layer of the system. The JavaBeans are used to interact with the Session beans and the JSPs interact with the JavaBeans, this allow most of the client side code to be available in the JavaBeans and leaves very less to be done by the Java Server Pages.

Responsibility of Various Entity Beans

ForumBean

Form Bean is responsible for representing persistence data related to Forums. The Forum ID is generated using the AutoNumber Bean.

MessageBean

The MessageBean is responsible for representing persistence data related to a Message. The ID is again generated using the AutoNumber Bean.

ReadMessageBean

The ReadMessage is responsible for representing Read messages for users. This keeps track of the messages a user has read during the time they spent on FreeForums. This uses a class called ReadMessagePK as it's primary key class. The primary key is made out of the *UserName* and *MessageID*

UserBean

As discussed in depth throughout the document, the User Bean is responsible for representing persistence data for users. With UserName as its primary key, which is a String object.

ForumUserBean

This Bean represents persistence data, which keeps track of subscriptions of users to forums that exists on the server. This Bean uses ForumUserPK as its primary key class as the primary key is a composite primary key made of the *UserName* and the *ForumID*. Objects of this bean also contain information about

AutoNumberBean

This Bean represents persistence data that acts as Auto Numbers for other Entity Beans. It maintains a list of Integers associated with keys. Each key is the name of another bean, which requires a auto number as it's ID.

Responsibility of various Session Beans

UserManagerBean

- Add a new user
- Modify existing user details
- Remove an existing user
- Add existing user to a forum
- Remove existing user from forum
- Verify user for login to Forum system.
- Return all users in the database

ForumManagerBean

- Add a new Forum to the system
- Return a collection of all Forums
- Return a collection of forums for a user
- Finds a forum given a forum id

- Checks to see if a particular user can use a specified forum

MessageManagerBean

- Gets a complete message given the message id
- Add a new message
- Marks a message read for a user
- Checks to see if a message is read by a user
- Returns a collection of all messages in a particular forum.

Responsibility of various Java Beans

These JavaBeans or Support classes have been written to remove the responsibilities from the Java Server Pages. The JSP code now looks fairly simple, as all the calls to the EJBs have been encapsulated in these classes.

Some methods in these classes contain web specific code. These classes are likely to change with the integration of XML.

UserController

- Login User
- Gets user details
- Adds a new User details
- Modify user details
- Authenticate Web Session
- Add a user to forum
- Remove user from forum
- List users
- List users for use with a web combo box component.

MessageController

- Gets a message given the message id
- Adds a new message
- Marks a message read for a user
- Checks to see if the message is read by a user
- Formats Messages for display on the web page
- Counts Sub Messages to a Parent Message

ForumController

- Adds a new Forum
- Checks to see if a user is an admin to any forums
- Produces a list of Forums
- Produces a list of forums for use with web combo boxes.

- Checks to see if a user is allowed to use a particular forum.

The Desktop Client

The original plan of the of FreeForums, included the idea of writing a Desktop Client in Microsoft Visual Basic, which would demonstrate the cross platform capabilities of XML and the use of SOAP (XML Based protocol) in an Enterprise Application.

As time did not permit the complete production of this client, the prototype of the system has been included in the CD-ROM for the project. The client at it's present state does not communicate with the server, it only demostrate the way the desktop client would look and handle data on the client side.

The desktop client uses an Access 2000 database as it's backend to store the information that will be downloaded from the Enterprise Server, and it is intended to use the Microsoft Internet Explorer ActiveX component to parse XML and send XML based messges to the Server.

The document now only covers the design considerations of the Desktop Client and an overview of the SOAP protocol and the Microsoft IE component for parsing XML.

SOAP (Simple Object Access Protocol)

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

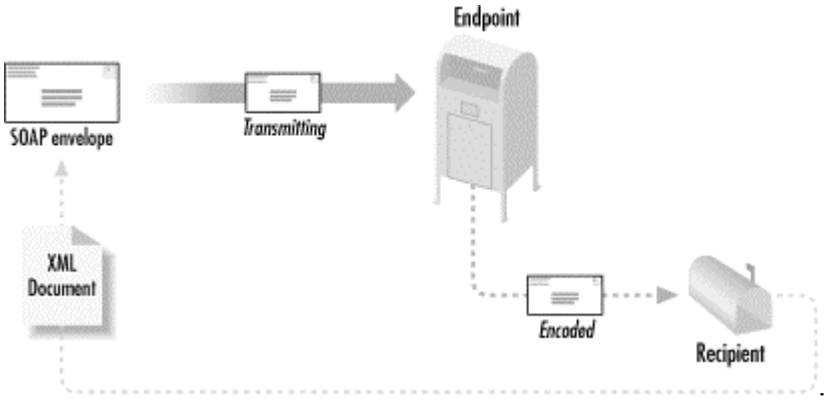
An important need for Internet application development is to allow communication between applications. Today's distributed applications communicate using remote procedure calls (RPC) between distributed objects like DCOM and CORBA. HTTP was not designed for this, so RPC calls are not easily adaptable to the Internet. Using RPC over the Internet also represents a security problem, and firewalls and proxy servers will normally block this kind of traffic. A much better way to communicate between applications would be to use HTTP, because all Internet browsers and servers support HTTP. SOAP was created to accomplish this.

More about SOAP

Adapted from Brett McLaughlin's book Java and XML

There are three basic components to the SOAP specification: the SOAP envelope, a set of encoding rules, and a means of interaction between request and response. Begin to think about a SOAP message as an actual letter; those antiquated things in envelopes with postage and an

address scrawled across the front? That analogy helps SOAP concepts like "envelope" makes a lot more sense. The figure seeks to illustrate the SOAP process in terms of this analog.



With this picture in our head, let's look at the three components of the SOAP specification. Additionally, it's these three key components that make SOAP so important and valuable. Error handling, support for a variety of encodings, serialization of custom parameters, and the fact that SOAP runs over HTTP makes it more attractive in many cases than the other choices for a distributed protocol. Additionally, SOAP provides a high degree of interoperability with other applications.

The Envelope

The SOAP envelope is analogous to the envelope of an actual letter. It supplies information about the message that is being encoded in a SOAP payload, including data relating to the recipient and sender, as well as details about the message itself. For example, the header of the SOAP envelope can specify exactly how a message must be processed. Before an application goes forward with processing a message, the application can determine information about a message, including whether it will even be able to process the message.

Distinct from the situation with standard XML-RPC calls, with SOAP actual interpretation occurs in order to determine something about the message. A typical SOAP message can also include the encoding style, which assists the recipient in interpreting the message. The example below shows the SOAP envelope, complete with the specified encoding.

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 soap:encodingStyle="http://myHost.com/encodings/secureEncoding"
>
<soap:Body>
 <article xmlns="http://www.ibm.com/developer">
  <name>Soapbox</name>
  <url>
    http://www-106.ibm.com/developerworks/library/x-soapbx1.html
   </url>
 </article>
</soap:Body>
</soap:Envelope>
```


We can see that an encoding is specified within the envelope, allowing an application to determine (using the value of the encodingStyle attribute) whether it can read the incoming message situated within the Body element. One should be sure to get the SOAP envelope namespace correct, or SOAP servers that receive messages will trigger version mismatch errors, and you won't be able to interoperate with them.

Encoding

The second major element that SOAP brings to the table is a simple means of encoding user-defined data types. In RPC (and XML-RPC), encoding can only occur for a predefined set of datatypes: those that are supported by whatever XML-RPC toolkit you download. Encoding other types requires modifying the actual RPC server and clients themselves. With SOAP, however, XML schemas can be used to easily specify new datatypes, and those new types can be easily represented in XML as part of a SOAP payload. Because of this integration with XML Schema, you can encode any datatype in a SOAP message that you can logically describe in an XML schema.

Invocation

The best way to understand how SOAP invocation works is to compare it with something we already know, such as XML-RPC. An XML-RPC call would look something like the code fragment shown:

```
// Specify the XML parser to use
XmlRpc.setDriver("org.apache.xerces.parsers.SAXParser");

// Specify the server to connect to
XmlRpcClient client =
 new XmlRpcClient("http://rpc.middleearth.com");

// Create the parameters
Vector params = new Vector( );
params.addElement(flightNumber);
params.addElement(numSeats);
params.addElement(creditCardType);
params.addElement(creditCardNum);

// Request reservation
Boolean boughtTickets =
  (Boolean)client.execute("ticketCounter.buyTickets", params);

// Deal with the response
```

The same call in SOAP would look like:

```
// Create the parameters
Vector params = new Vector( );
params.addElement(
  new Parameter("flightNumber", Integer.class, flightNumber, null));
params.addElement(
  new Parameter("numSeats", Integer.class, numSeats, null));
params.addElement(
  new Parameter("creditCardType", String.class, creditCardType, null));
params.addElement(
  new Parameter("creditCardNumber", Long.class, creditCardNum, null));

// Create the Call object
Call call = new Call( );
```



```
call.setTargetObjectURI("urn:xmltoday-airline-tickets");
call.setMethodName("buyTickets");
call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
call.setParams(params);

// Invoke
Response res = call.invoke(new URL("http://rpc.middleearth.com"), "");

// Deal with the response
```

As we can see, the actual invocation itself, represented by the Call object, is resident in memory. It allows you to set the target of the call, the method to invoke, the encoding style, the parameters, and more not shown here. It is more flexible than the XML-RPC methodology, allowing you to explicitly set the various parameters that are determined implicitly in XML-RPC.

Limited information about SOAP has been included in this document, as it has not been implemented in the project. Various resource and examples have been listed in the References, which can help enthusiasts to go about learning SOAP.

Deficiencies of FreeForums

Design issues

Session Beans design issues: There are three main session beans that the clients interact with. I would have much preferred to deal with one session bean that acted as the gateway to the whole system producing a much more robust design.

EJB 1.1 issues: EJB 1.1 does not allow pure OO to relational mapping and the application server only permitted the usage of EJB1.1 thus the implementation of the Entity classes have violated the OO design model. This issue has been discussed in detail in the design section.

Implementation issues

Issues with the application servers: The original plan of the FreeForums was to use Sun's reference implementation server. The prototype made use of it as well, however as I went into designing and implementing the EJBs it proved to be harder to use, some of it was due to lack of documentation available about the server.

JBoss came to the rescue and FreeForums has been developed using JBoss. Performance is an issue with both the products as they are written in Java. JBoss is better as it uses only one JVM where as the Sun server uses two. Commercial servers are better but much more complex to use and they are usually not available for free.

XSLT recursion issues: the processs of recursion is quite complex using XSLT, this was taking quite some time for me to learn, however I opted against learning recursion using XML and have used recursed Java method to use the JDOM API to create part of the HTML document. This demostrates the use of the JDOM API to create the HTML document as well as a complex process of recursion that deals with XML data at the same time. I deally I would have used XSLT's recursion functions.

Desktop Client Implementation: The desktop client that was to work in conjuction with SOAP could not be implemented due to lack of time. The ideas of the client however have been documented and one can read about the general ideas of implementing VB desktop clients using SOAP.

Conclusion

The FreeForums system makes use of many J2EE technologies that have been discussed in great detail over the span of the document. This report is not only meant to be an insight about creating FreeForums but also an insight to creating applications using Java 2 Enterprise Edition. Many areas that have not be covered in details in many other document available on the world wide web or the books that have been reviewed during the process of the creating FreeForums, have been covered in this document. Moreover this document look at Java 2 Enterprise Edition from a learning prespective where we are not trying to find quick and easy answers to our problems but are attempting to study every part of the technology and find it's advantages and disadvantages of the technology.

During the production of FreeForums I encountered many problems while learning J2EE related technologies, as this was a self-learning process and all my support was from the books, web based resources and the professional list servces being run on the Internet. This was a very time taking process as each new technology had it's own hurdles in terms of ease of use and implementation with respect to the other technologies in play.

Currently FreeForums does not support uploading files that can be attached with messages; this feature was excluded from the basic requirements of the system and has been shifted to the advanced development requirements. The system also lack in many management functions, as they are also part of the advanced requirements. The current version of FreeForums is not yet fit for being used in real world for a live system but it defiantly the basis for one.

The system uses reuseable enterprise components that can be reused if a fully featured forum system was being implemented using J2EE technologies. Some of the parsing, XSLT implementation could be used as is and could cut down a lot of work for developers writing Forum Systems. During my interactions with professional on the JBoss list serve one of them who was facing the same problem as me with AutoNumbers and databases downloaded the FreeForums source code and used the AutoNumberBean with my permission. Thus demonstrating me that the code written for a Forum system could be used for a J2EE based financial software without much of a rewrite.

Overall the development of FreeForums has lead to my understanding of Java 2 Enterprise Edition technologies to a great depth. It has also lead me to interact with many professional in the industry and help them solve their problems and inturn get their support to solve the problems I have encountered during the development of FreeForums. This document is also a result of my learning of J2EE and I hope it will be a starting point for many others who are starting to learn J2EE based technologies. This document fills many gaps that I found while I was reading about J2EE and XML, it also acts a quick guide to getting your first J2EE application running which is sometimes hard to find in the professional publications.

With all the ups and downs considered this project has defiantly been a very valueable and a great learning experience for me. I have managed to produce a fully functional J2EE application and support it with the document allowing others who are coming into the world of Enterprise Technologies a guide as well a reference implementation to look at.

References

Alexander Nakhimovsky and Tom Myers. 2000. *Professional Java XML programming with Servlets and JSP*. (Programmer to Programmer), Wrox Press, USA.

Brett McLaughlin. 2000. *Java and XML.* (The Java Series), O'Reilly, USA.

Bill Brogden and Chris Minnick. 2001. *Java Developer's guide to E-Commerce with XML and JSP.* Sybex, USA

Bob DuCharme. 2001. *XSLT Quickly.* Manning, Greenwich, CT.

Chelsea Valentine and Chris Minnick. 2001. *XHTML.* New Riders Press,USA.

Duane K. Fields and Mark A. Kolb. 2000. *Web Development with Java Server Pages.* Manning, Greenwich, CT

Didier Martin, Mark Birbeck, Michael Kay, Brian Loesgen, Jon Pinnock, Steven Livingstone, Peter Stark, Kevin Williams, Richard Anderson, Stephen Mohr, David Baliles, Bruce Peat, Nikola Ozu. 2000. *Professional XML.* (Programmer to Programmer), Wrox Press, USA.

Erik T. Ray. 2001. *Learning XML.* O'Reilly, USA.

Richard Monson-Haefel. 2000. *Enterprise JavaBeans.* 2nd Edn. (The Java Series), O'Reilly, USA.

Paul J. Perrone and Venkata S. R. "Krishna" R. Chaganti. 2000 *Building Java Enterprise Systems with J2EE*. Sams Publication, USA.

Paul Spencer. 2000. *Professional XML Design and Implementation.* (Programmer to Programmer), Wrox, USA.

Subrahmanyam Allamaraju, Karl Avedal, Richard Browett, Jason Diamond, John Griffin, Mac Holden, Rod Johnson, Tracie Karsjens, Andrew Longshaw, Tom Myers, Alexander Nakhimovsky, Daniel O'Conner, Sameer Tyagi, Geert Van Damme, Gordon van Huizen, Mark Wilcox, Stefan Zeiger. 2000. *Professional Java Server Programming J2EE Edition.* (Programmer to Programmer). Wrox, USA.

Java 2 Enterprise Edition FAQs (online), Retrieved via Netscape 10th July 2001. http://java.sun.com/j2ee/faq.html

Java 2 Enterprise Java Beans FAQs (online), Retrieved via Netscape 10th July 2001. http://java.sun.com/products/ejb/faq.html

Jboss FAQ section (online), Retrieved via Netscape 10th July 2001. http://www.jboss.org/faq.jsp

Gopalan Suresh Raj - *Enterprise Java Beans* (online), Retrieved via Netscape 10th July 2001 http://www.execpc.com/~gopalan/java/ejb/entity.html

http://www.execpc.com/~gopalan/java/ejb/session.html

Introduction to XSLT, Accessed via Microsoft IE
http://java.sun.com/xml/jaxp-1.1/docs/tutorial/xslt/1_intro.html

Documentation for XSLT Tag Library, Accessed via Microsoft IE
http://jakarta.apache.org/taglibs/doc/xsl-doc/index.html

Echoing an XML file with a SAX parser, Accessed via Microsoft IE
http://java.sun.com/xml/jaxp-1.0.1/docs/tutorial/sax/2a_echo.html

Possible to use recursion with XSLT Transform [ListServe Message Posting]
http://www.jxml.com/archive/java-xml-interest/0404.html

Serial Access with SAX, Accessed via Microsoft IE
http://www.jxml.com/archive/java-xml-interest/0404.html

Setting and Using variables and Parameters, Accessed via Microsoft IE
http://www.xml.com/lpt/a/2001/02/07/trxml9.html

Producing HTML tables with XSLT, Accessed via Microsoft IE
http://www.cogsci.ed.ac.uk/~dmck/xmlperl/xslt-tutorial.html

The JDOM Project, Accessed via Netscape http://www.jdom.org

XML Parsers: DOM and SAX put to Test, Accessed via Microsoft IE on 16th October 2001
http://www.xml-zone.com/articles/sf020101/sf0201-1.asp

Using the SAX Parser, Accessed via Microsoft IE on 16th October 2001
http://www.javacommerce.com/tutorial/xmldev/SaxParser1.htm

XML and Related Spec: Digesting the Alphabet Soup, Accessed via Microsoft IE on 16th October 2001, http://industry.ebi.ac.uk/~senger/xml/docs/tutorial/overview/2_specs.html

The Jakarta-Tablib project: XSL Tag Library, Accessed via Microsoft IE on 16th October 2001, http://jakarta.apache.org/taglibs/doc/xsl-doc/intro.html

Beth Stearns, April 2001, Enterprise Java Beans 2.0 Specification Changes, Accessed via Microsoft IE on 20th October 2001,
http://developer.java.sun.com/developer/technicalArticles/ebeans/ejb20/index.html

Beth Stearns, September 2001, Migrating from EJB 1.1 to EJB 2.0, Accessed via Microsoft IE on 20th October 2001,
http://developer.java.sun.com/developer/technicalArticles/ebeans/ejbmigrate/

W3C, 2001, SOAP, Access via Microsoft IE on 4th November 2001
http://www.w3.org/TR/SOAP

W3Schools, SOAP Syntax, Accessed via Microsoft IE on 4th November 2001.
http://www.w3schools.com/soap/soap_syntax.asp

Appendix A

Readings on Java 2 Enterprise Edition collected from the World Wide Web. These articles are copyright of their respective Authors.

All articles included in the Appendix are an unaltered version of the original piece.

Compiled By Devraj Mukherjee

What is an Entity Bean?

By Gopalan Suresh Raj

An Entity Bean represents persistent data that are maintained in a domain model, as well as methods that act on that data. Or to be more specific, an Entity Bean exactly maps to a record in your domain model. In a relational database context, there is one Bean for each row in a table. This is not a new concept, since this is how object databases have been modeled all along. A primary key identifies each Entity Bean. Entity Beans are created by using an object factory create() method. Entity beans are transactional and are recoverable following a system crash. Entity beans are also implicitly persistent. An EJB object can manage its own persistence, or it can delegate its persistence to its container.

Entity Beans always have states which can be persisted and stored across multiple invocations. Multiple EJB Clients may however share an Entity Bean. The lifetime of an Entity Bean is not limited by the lifetime of the virtual machine within which it executes. A crash of the virtual machine may result in a rollback of the current transaction, but will neither destroy the Entity Bean nor invalidate the references that other clients have to this Entity Bean. Moreover, a client can later connect to the same Entity Bean using its object reference since it encapsulates a unique primary key allowing the EJB Bean or its container to reload its state. Entity beans can thus survive system shutdowns.

Persistence in Entity Beans is of two types. They are:

- **Container-managed persistence**: Here, the EJB container is responsible for saving the state of the Entity Bean. Since it is container-managed, the implementation is independent of the data source. However, all container-managed fields need to be specified in the Deployment Descriptor for the persistence to be automatically handled by the container.

- **Bean-managed persistence**: Here, the Entity Bean is directly responsible for saving its own state and the container does not need to generate any database calls. Consequently, this implementation is less adaptable than the previous one as the persistence needs to be hard-coded into the Bean.

Every Entity Bean has the following characteristics:

- Entity beans can share access from multiple users.
- They can participate in transactions.
- Entity beans represent data in a domain model.
- They are persistent. They live as long as the data lives in the domain model.
- Entity beans can survive EJB server crashes. Any EJB server crash is always transparent to the client.
- Entity beans have a persistent object reference. The object reference encapsulates the Persistent key for this Bean.

What is a Session Bean?

By Gopalan Suresh Raj

A session bean is created by a client and in most cases exists only for the duration of a single session. A session bean performs operations on behalf of the client such as database access or performing calculations. Although session beans can be transactional, they are not recoverable following a system crash. Session beans can be stateless or can maintain conversational state across methods and transactions. The container manages the conversational state of a session bean if it needs to be evicted from memory. A session bean must manage its own persistent data.

Each Session Bean is usually associated with one EJB client. Each Session Bean is created and destroyed by the particular EJB client that it is associated with. It is thus transient and will not outlive the virtual machine on which it was created. A Session Bean can either have states or they can be stateless. However, Session Beans do not survive a system shutdown.

There are two types of Session Beans. They are Stateless Session Beans: These types of session EJBs have no internal state. Since they are stateless, they need not be passivated and can be pooled in to service multiple clients.

Stateful Session Beans: These types of session EJBs possess internal states. Hence they need to handle Activation and Passivation. However, there can be only one stateful Session Bean per EJB client. Since they can be persisted, they are also called Persistent Session Beans. These types of EJBs can be saved and restored across client sessions. The getHandle() method returns a Bean object's instance handle which can be used to save the Bean's state. To restore a Bean from persistent storage, the getEJBObject() method is used.

The characteristics of a Session Bean can be summarized as follows:

- Session EJBs execute on behalf of a single client.
- A Session Bean instance is an extension of the client that creates it.
- Session beans may be transaction-aware.
- They can update data in an underlying database.
- They are relatively short-lived, since their lifetime is limited to that of their client.
- They may be destroyed when the EJB server crashes. The client has to establish connection with a new Session Bean object to resume any computation.
- Session beans do not represent data that has to be stored in a database.

Every Session Bean can read and update a database on behalf of the client. Its fields may contain conversational state on behalf of the client. The state describes the conversation represented by a specific client/instance pair. Within a transaction some of this data may be cached in the Bean.

Session beans are supposed to be private resources used only by the client that created the EJB Session Bean. For this reason a Session Bean hides its identity and is anonymous, in sharp contrast to an Entity Bean that exposes its identity using its primary key.

Java 2 Enterprise Edition

Adapted from Sun Microsystems Java 2 Enterprise Edition web site

Overview:

The Platform for Enterprise Solutions

The JavaTM 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. J2EE simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming. The Java 2 platform, Enterprise Edition, takes advantage of many features of the Java 2 platform, Standard Edition, such as "Write Once, Run AnywhereTM" portability, JDBCTM API for database access, CORBA technology for interaction with existing enterprise resources, and a security model that protects data even in internet applications. Building on this base, Java 2 Enterprise Edition adds full support for Enterprise JavaBeansTM components, Java Servlets API, JavaServer PagesTM and XML technology. The J2EE standard includes complete specifications and compliance tests to ensure portability of applications across the wide range of existing enterprise systems capable of supporting J2EE.

Making Middleware Easier

Today's enterprises gain competitive advantage by quickly developing and deploying custom applications that provide unique business services. Whether they're internal applications for employee productivity, or internet applications for specialized customer or vendor services, quick development and deployment are key to success.

Portability and scalability are also important for long term viability. Enterprise applications must scale from small working prototypes and test cases to complete 24 x 7, enterprise-wide services, accessible by tens, hundreds, or even thousands of clients simultaneously.

However, multitier applications are hard to architect. They require bringing together a variety of skill-sets and resources, legacy data and legacy code. In today's heterogeneous environment, enterprise applications have to integrate services from a variety of vendors with a diverse set of application models and other standards. Industry experience shows that integrating these resources can take up to 50% of application development time.

As a single standard that can sit on top of a wide range of existing enterprise systems – database management systems, transaction monitors, naming and directory services, and more - J2EE breaks the barriers inherent between current enterprise systems. The unified J2EE standard wraps and embraces existing resources required by multitier applications with a unified, component-based application model. This enables the next generation of components, tools, systems, and applications for solving the strategic requirements of the enterprise.

With simplicity, portability, scalability and legacy integration, J2EE is the platform for enterprise solutions.

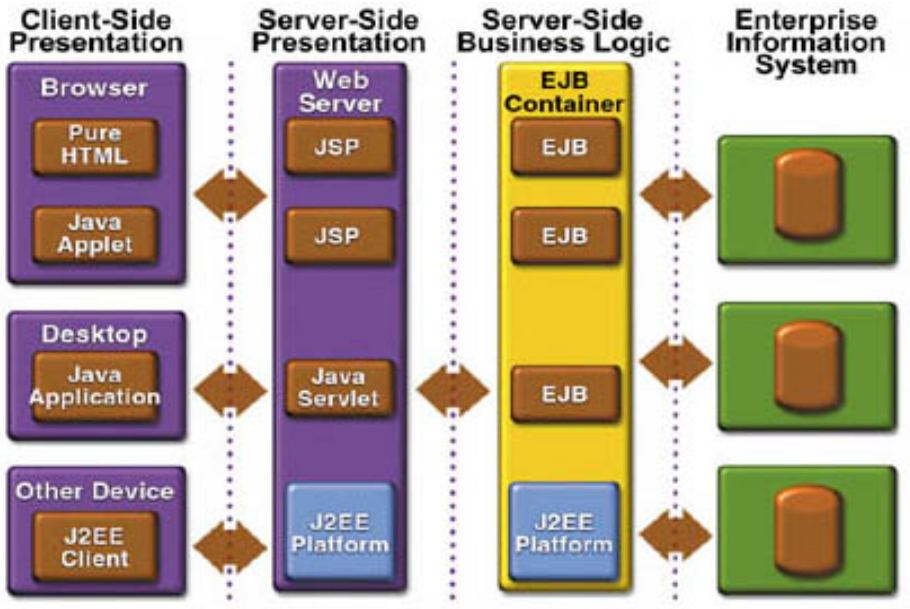
The Standard with Industry Momentum

While Sun Microsystems invented the Java programming language and pioneered its use for enterprise services, the J2EE standard represents a collaboration between leaders from throughout the enterprise software arena. Our partners include OS and database management system providers, middleware and tool vendors, and vertical market applications and component developers. Working with these partners, Sun has defined a robust, flexible platform that can be implemented on the wide variety of existing enterprise systems currently available, and that supports the range of applications IT organizations need to keep their enterprises competitive.

The J2EE Application Model

The J2EE Blueprints describe the J2EE application model and best practices for using the J2EE platform. Building on the Java 2 Platform, Enterprise Edition, the J2EE Blueprints provide a simplified approach to developing highly scalable and highly available internet or intranet based applications.

Maybe the most interesting thing about Java 2 Enterprise Edition applications is what they don't do. That is, various complexities inherent in enterprise applications -- transaction management, life-cycle management, resource pooling -- are built into the platform and provided automatically to the components it supports. Component and application developers are free to focus on specifics such as business logic and user interfaces.



Another advantage of the J2EE platform is that the application model encapsulates the layers of functionality in specific types of components. Business logic is encapsulated in Enterprise JavaBeansTM (EJB) components. And client interaction can be presented through plain HTML web pages, through web pages

powered by Java technology-based applets, Java Servlets API, or JavaServer PagesTM technology, or through stand-alone Java applications. Components communicate transparently using various standards: HTML, XML, HTTP, SSL, RMI, IIOP, and others.

J2EE platform reusable components mean competitive choices for enterprise developers and IT organizations. The J2EE platform will enable them to assemble applications from a combination of standard, commercially available components and their own custom components.

From general business application components to vertical market solutions, a range of standardized Java 2 Enterprise Edition functionality is expected to be available off the shelf.

This means that an e-commerce site could be built using a combination of off-the-shelf EJB components for shopping cart behaviors, modified EJB components for specialized customer services, and completely customized layouts using JavaServer Pages technology that bring a unique look and feel to the site.

This approach means faster development time, better quality and maintainability, and portability across a range of enterprise platforms. The bottom line benefits are increased programmer productivity, better strategic use of computing resources, and greater return on an organization's technology investments.

Flexible User Interaction

The J2EE platform provides choices for graphical user interfaces across a company's intranet or on the world wide web. Clients can run on desktops, laptops, PDA's, cell phones, and other devices. Pure client-side user interfaces can use standard HTML and Java technology-based applets. Support for simple HTML means quicker prototypes, and support for a broader range of clients.
Additionally, the J2EE platform supports automatic download of the Java PlugIn to add applet support where it's lacking. The J2EE platform also supports stand-alone Java application clients.

For server-side deployment of dynamic content, the J2EE platform supports both Java servlets API and JavaServer Pages (JSP) technology. The Java Servlets API enables developers to easily implement server-side behaviors that take full advantage of the power of the rich Java API.

JavaServer Pages technology combines the ubiquity of HTML with the power of server-side scripting in the Java programming language. The JSP 1.0 specification supports static templates, dynamic HTML generation, and custom tags.

Enterprise JavaBeans -- the Flexible Business Component Model Since it was introduced, Enterprise JavaBeans (EJB) technology has developed significant
momentum in the middleware marketplace. That's because it enables a simplified approach to multitier application development, concealing application complexity and enabling the component developer to focus on business logic. And the J2EE platform is the natural evolution of Enterprise JavaBeans technology.

EJB technology gives developers the ability to model the full range of objects useful in the enterprise by defining two distinct types of EJB components: Session Beans and Entity Beans. Session Beans represent behaviors associated with client sessions -- for example, a user purchase transaction on an e-commerce site. Entity Beans represent collections of data -- such as rows in a relational database -- and encapsulate operations on the data they represent. Entity Beans are intended to be persistent, surviving as long as the data they're associated with remains viable.

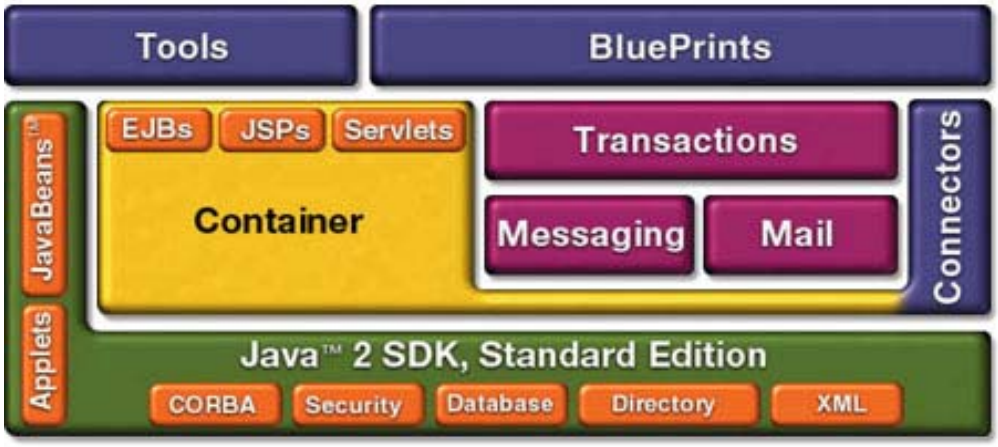
The J2EE platform extends the power and portability of EJB components by defining a complete infrastructure that includes standard clients and service APIs for their use.

A Model for Expediting Development

Based on these flexible component configurations, the J2EE application model means quicker development, easier customization and greater ability to develop powerful enterprise applications. And, because it's based on the Java programming language, this model enables all the J2EE applications achieve all the benefits of Java technology: scalability, portability, and programming ease.

Components:

In addition to providing support for Enterprise JavaBeansTM, Java Servlets and JavaServer Pages components, the Java 2 Platform, Enterprise Edition specification defines a number of standard services for use by J2EE components.



Java Naming and Directory InterfaceTM API

Designed to standardize access to a variety of naming and directory services, the Java Naming and Directory Interface (JNDI) API provides a simple mechanism for J2EE components to look up other objects they require.

JDBC API

JDBC API enables applications to manipulate existing data from relational databases and other data repositories. J2EE includes the latest implementation of JDBC API -- version 2.0.

This new technology includes handling of SQL User-Defined Types (UDTs), rowset manipulation, connection pooling, and distributed transactions support.

JavaMail API

J2EE includes JavaMail to support applications such as e-commerce websites. The JavaMail API provides the ability to send order confirmations and other user feedback.

CORBA Compliance

J2EE supports two CORBA-compliant technologies: JavaIDL and RMI-IIOP. JavaIDL enables Java applications to interact with any CORBA-compliant enterprise system. RMI-IIOP technology combines the programming ease of the Java Remote Method Invocation API (RMI) with CORBA's Internet Inter-ORB Protocol (IIOP) for easier integration of J2EE applications with legacy applications.

Java Transaction API

While J2EE provides transaction support automatically, the Java Transaction API (JTA) provides a way for J2EE components and clients to manage their own transactions and for multiple components to participate in a single transaction.

XML Deployment Descriptors

J2EE defines a set of descriptors in the universal data language, XML. With its ability to support both standard and custom data types, XML makes it easier to implement customizable components and to develop custom tools.

Java Message Service

The Java Message Service (JMS) API defines a standard mechanism for components to send and receive messages asynchronously, for fault-tolerant interaction. JMS is optional for J2EE release 1.0.

Appendix B

XML Document Type Definition

Author: Devraj Mukherjee

User.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Devraj Mukherjee (ETNET) -->
<!ELEMENT UserName (#PCDATA)>
<!ELEMENT RealName (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!ELEMENT EmailAddress (#PCDATA)>
<!ELEMENT SuperUser (#PCDATA)>
<!ELEMENT User (UserName+, Password+, RealName+, EmailAddress+, SuperUser+)?>
<!ELEMENT UserList (User+)+>
```

Response.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Devraj Mukherjee (FreeForums) -->
<!ELEMENT Response (#PCDATA)>
```

Message.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Devraj Mukherjee (FreeForums) -->
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT MessageBody (#PCDATA)>
<!ELEMENT UserName (#PCDATA)>
<!ELEMENT MessageDate (#PCDATA)>
<!ELEMENT ForumID (#PCDATA)>
<!ELEMENT MessageID (#PCDATA)>
<!ELEMENT ReplyToID (#PCDATA)>
<!ELEMENT Message (MessageID+, Subject+, MessageBody+, MessageDate+, UserName+, ForumID+,
ReplyToID+)*>
<!ELEMENT MessageList (Message+)*>
```

Forum.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Devraj Mukherjee (FreeForums) -->
<!ELEMENT ForumName (#PCDATA)>
<!ELEMENT ForumDescription (#PCDATA)>
<!ELEMENT ForumID (#PCDATA)>
<!ELEMENT Forum (ForumID+, ForumName+, ForumDescription+)*>
<!ELEMENT ForumList (Forum+)?>
```


Appendix C

XSL (XML Style Sheets)

Author: Devraj Mukherjee

userlist.xsl

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="UserList">

<table width="100%" border="1">

 <tr>
 <td><font face="Verdana" size="2"><b>User Name</b></font></td>
 <td><font face="Verdana" size="2"><b>Real Name</b></font></td>
 <td><font face="Verdana" size="2"><b>Email Address</b></font></td>
 <td><font face="Verdana" size="2"><b>Super User</b></font></td>
 </tr>

 <xsl:for-each select="User">
 <tr>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="UserName"/></font>
  </td>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="RealName"/></font>
  </td>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="EmailAddress"/></font>
  </td>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="SuperUser"/></font>
  </td>

 </tr>

 </xsl:for-each>

</table>
</xsl:template>
</xsl:stylesheet>
```

userforumlist.xsl

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="ForumList">

<table width="80%" border="1">

 <tr>
 <td><font face="Verdana" size="2"><b>Forum Name</b></font></td>
 <td><font face="Verdana" size="2"><b>Description</b></font></td>
 </tr>

 <xsl:for-each select="Forum">
 <tr>

  <td>
  <font face="Verdana" size="2">

  <xsl:element name="A">

   <xsl:attribute name="HREF">
    show_messages.jsp?forum_id=<xsl:value-of select="ForumID"/>
   </xsl:attribute>

   <xsl:value-of select="ForumName"/>

  </xsl:element>

  </font>
  </td>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="ForumDescription"/></font>
  </td>

 </tr>

 </xsl:for-each>

</table>
</xsl:template>
</xsl:stylesheet>
```


user_combo_box.xsl

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="UserList">

<select name="username">

 <xsl:for-each select="User">

  <xsl:element name="OPTION">

   <xsl:attribute name="VALUE">
    <xsl:value-of select="UserName"/>
   </xsl:attribute>

   <xsl:value-of select="RealName"/>

  </xsl:element>

 </xsl:for-each>

</select>

</xsl:template>
</xsl:stylesheet>
```


message_list.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="MessageList">

 <table border="0" width="100%">

 <tr bgcolor="#000000">
  <td><font face="Verdana" size="2" color="#FFFFFF"><b>Date</b></font></td>
  <td><font face="Verdana" size="2" color="#FFFFFF"><b>Subject</b></font></td>
  <td><font face="Verdana" size="2" color="#FFFFFF"><b>Posted By</b></font></td>
  <td><font face="Verdana" size="2" color="#FFFFFF"><b>Reply</b></font></td>
 </tr>

 <xsl:for-each select="Message">
 <tr>

  <td width="10%">
  <font face="Verdana" size="2"><xsl:value-of select="MessageDate"/></font>
  </td>

  <td>
   <font face="Verdana" size="2">

   <xsl:element name="A">

   <xsl:attribute name="HREF">
    read_message.jsp?message_id=<xsl:value-of select="MessageID"/>&amp;forum_id=
    <xsl:value-of select="ForumID"/>
   </xsl:attribute>

   <xsl:value-of select="Subject"/>

   </xsl:element>

   </font>
  </td>

  <td width="10%">
   <font face="Verdana" size="2"><xsl:value-of select="UserName"/></font>
  </td>

  <td width="10%">
   <font face="Verdana" size="2">

   <xsl:element name="A">

   <xsl:attribute name="HREF">
    post_message.jsp?reply_to=<xsl:value-of select="MessageID"/>
    &amp;forum_id=<xsl:value-of select="ForumID"/>

   </xsl:attribute>

    Reply

   </xsl:element>

   </font>
  </td>

 </tr>

 </xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

forum_list.xsl

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="ForumList">

<table width="80%" border="1">

 <tr>
 <td><font face="Verdana" size="2"><b>Forum Name</b></font></td>
 <td><font face="Verdana" size="2"><b>Description</b></font></td>
 </tr>

 <xsl:for-each select="Forum">
 <tr>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="ForumName"/></font>
  </td>

  <td>
  <font face="Verdana" size="2"><xsl:value-of select="ForumDescription"/></font>
  </td>

 </tr>

 </xsl:for-each>

</table>
</xsl:template>
</xsl:stylesheet>
```

forum_combo_box.xsl

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" omit-xml-declaration="yes"/>
<xsl:template match="/">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="ForumList">

<select name="forumID">

 <xsl:for-each select="Forum">

  <xsl:element name="OPTION">

   <xsl:attribute name="VALUE">
    <xsl:value-of select="ForumID"/>
   </xsl:attribute>

   <xsl:value-of select="ForumName"/>

  </xsl:element>

 </xsl:for-each>

</select>

</xsl:template>
</xsl:stylesheet>
```

Appendix D

Apache TagLib for XSLT Documentation

Author: The Apache Project

Documentation for the XSL Tag Library

Introduction

The *XSL* custom tag library contains an working examples of the tags described in the JavaServerPages specification, Version 1.1. As such, the tags are focused more on being examples of custom tag library code techniques, and less on being useful in production applications.

With this custom tag library you can process a XML document with a XSL stylesheet and insert it in place.

Prerequisite software

- A servlet container that supports the JavaServer Pages Specification, version 1.1.

  The binary distributions of the packages :

- The Apache Xerces XML parser.
- The Apache Xalan XSL processor.

  this can be found at http://xml.apache.org/dist, we recommend that you use the version of the Xerces library that comes with the Xalan binary distribution

Configuration Information

Follow these steps to configure your web application with this tag library:

- Copy the tag library descriptor file (xsl/xsl.tld) to the /WEB-INF subdirectory of your web application.
- Copy the tag library JAR file (xsl/xsl.jar) to the /WEB-INF/lib subdirectory of your web application.
- Copy the support libs xerces.jar and xalan.jar to the /WEB-INF/lib directory of your web application.
- Add a <taglib> element to your web application deployment descriptor in /WEB-INF/web.xml like this:

```
<taglib>
    <taglib-uri>http://jakarta.apache.org/taglibs/xsl-1.0</taglib-uri>
    <taglib-location>/WEB-INF/xsl.tld</taglib-location>
</taglib>
```

To use the tags from this library in your JSP pages, add the following directive at the top of each page:

```
<%@ taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="x" %>
```

where "x" is the tag name prefix you wish to use for tags from this library. You can change this value to any prefix you like.

Tag Information

This custom tag library includes several tags, with the following characteristics:
The "InsertWithXSL" tag

This tag inserts a XML document previously transformed by a XSL stylesheet into the output of the current page.

Attribute	Description	Required
url	The URL of XML document to be inserted in place	Yes
Xsl	The URL of XSL formatting stylesheet aplied to the XML document before inserting	Yes

The return tag

This tag flush the output buffers of the page, this is an utility tag. This not have any attributes
The clearresponse tag

This tag clears the output buffers of the page, this is an utility tag. This not have any attributes

Usage Examples

See the example application (xsl/xsl-examples.war) for examples of the usage of the tags from this custom tag library.

Appendix E

Source Code (Enterpise Java Beans and Java Beans)

Arranged in the order of the package.

Author: Devraj Mukherjee

AutoNumberHome.java

```
/*
 * AutoNumberHome.java
 *
 * Defines the home interface for the AutoNumberBean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.util;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public interface AutoNumberHome extends EJBHome {

        public AutoNumber create(String referenceName) throws RemoteException,
        CreateException;

        public AutoNumber findByPrimaryKey (String referenceName) throws RemoteException,
        FinderException;

 }
```

AutoNumber.java

```
/*
 * AutoNumber.java
 *
 * Defines the remote interface for the AutoNumberBean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.util;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public interface AutoNumber extends EJBObject {

        public Integer getAutoNumberValue() throws RemoteException;

 }
```

AutoNumberBean.java

```
/*
 * AutoNumberBean.java
 *
 * AutoNumberBean implementation
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.util;

 import java.rmi.RemoteException;
 import javax.ejb.*;
```



```
public class AutoNumberBean implements EntityBean {

        transient private EntityContext context;

        public Integer autoNumberValue;
        public String referenceName;

        public String ejbCreate(String referenceName) throws CreateException {

                this.referenceName = referenceName;
                this.autoNumberValue = new Integer(0);
                return null;

        }

        public void ejbPostCreate(String referenceName) { }

        public void setEntityContext(EntityContext context) { this.context = context; }

        public void unsetEntityContext() { context = null; }

        public void ejbActivate() { }
        public void ejbPassivate() { }
        public void ejbLoad() { }
        public void ejbStore() { }
        public void ejbRemove() { }

        /* The following are the implementation of the method from the
           AutoNumber Remote interface
         */

        public Integer getAutoNumberValue() throws RemoteException {

                Integer currentValue = autoNumberValue;
                autoNumberValue = new Integer(currentValue.intValue()+1);
                return currentValue;

        }

} // end class file
```

ForumManagerHome.java

```
/*
 * ForumManagerHome.java
 *
 * Defines a Session bean for forum based functions
 *
 * @author: Devraj Mukherjee
 *
 * Created: 04th July 2001
 *
 */

package freeforums.forum;

import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface ForumManagerHome extends EJBHome {

        public ForumManager create() throws RemoteException, CreateException;

}
```

ForumManagerBean.java

```
/*
 * ForumManagerBean.java
 *
 * Forum Manager bean represent functions for
 * managing the forums
 *
 * @author: Devraj Mukherjee
 *
 * Created: 07th July 2001
 *
 */

 package freeforums.forum;

 import java.rmi.RemoteException;
 import javax.ejb.*;
 import javax.naming.*;
 import javax.rmi.*;
 import java.rmi.ServerException;
 import java.util.Collection;
 import java.util.Vector;

 import freeforums.util.AutoNumberHome;
 import freeforums.util.AutoNumber;

 import freeforums.user.ForumUser;
 import freeforums.user.ForumUserHome;
 import freeforums.user.ForumUserPK;

 import org.jdom.*;
 import java.util.List;

 public class ForumManagerBean implements SessionBean {

        /**
        Helper function to get the home interface of a specified Bean. This is used by
        most of the methods in this class.
        */
        private Object getHome (String path) {
                try {
                        return (new InitialContext().lookup(path));
                }
               catch (NamingException e) {
                       System.err.println("An exception occured " + e);
                       return null;
               }
        }

        /**
        Empty method body
        */
        public ForumManagerBean() {}
        /**
        Empty method body
        */
        public void ejbCreate() {}
        /**
        Empty method body
        */
        public void ejbRemove() {}
        /**
        Empty method body
        */
        public void ejbActivate() {}
        /**
        Empty method body
        */
        public void ejbPassivate() {}
        /**
        Empty method body
        */
        public void setSessionContext(SessionContext sc) {}

        public Document addForum(Document forumDetails) throws RemoteException {


                Element RootElement = forumDetails.getRootElement();
```

```
        List forumListElement = RootElement.getChildren();
        Element [] forumElements = (Element[])forumListElement.toArray(new
        Element[0]);

        String forumName = forumElements[0].getChild("ForumName").getText();
        String forumDescription =
        forumElements[0].getChild("ForumDescription").getText();

        Element responseElement = new Element("Response");
        String identifier = "Forum";
        Integer id=null;

        try {

                /**
                 Process of creating AutoNumbers
                 **/
                AutoNumberHome ANHome = (AutoNumberHome)
                getHome("java:comp/env/ejb/AutoNumber");
                AutoNumber AutoNumberFactory = ANHome.findByPrimaryKey(identifier);
                id = AutoNumberFactory.getAutoNumberValue();

        }
        catch (Exception e) {
                /**
                 This Key was not found in the storage so
                 create a new one with this name
                 **/
                 try {
                        AutoNumberHome ANHome = (AutoNumberHome)
                        getHome("java:comp/env/ejb/AutoNumber");
                        AutoNumber AutoNumberFactory = ANHome.create(identifier);
                        id = AutoNumberFactory.getAutoNumberValue();
                 }
                 catch(Exception exp) {
                        /**
                         Could not create this primary key
                         **/
                  }

         }

         if (id==null) return (new Document(responseElement.addContent("Failed")));

         try {

                 /**
                 Process of creating the Forum Object
                 **/

                ForumHome home = (ForumHome) getHome("java:comp/env/ejb/Forum");
                Forum forum = home.create(id,forumName,forumDescription);
                return (new Document(responseElement.addContent("Done")));

        }
        catch(Exception e) {
                //do nothing
                System.out.println("Exception occured while creating Forum \n" + e);
                return (new Document(responseElement.addContent("Failed")));
        }
}

/*
 * Modified version returns XML document description of Forum List
 */

 public Document findAll() throws RemoteException, FinderException {

         ForumHome home = (ForumHome) getHome("java:comp/env/ejb/Forum");
         Collection all = home.findAll();
         Forum [] forumArray = (Forum[]) all.toArray(new Forum[0]);

         Element rootElement = new Element("ForumList");

         for(int counter=0; counter<forumArray.length; counter++) {
```

```
                Forum currentForum = forumArray[counter];

                Element ForumIDElement = new
                Element("ForumID").addContent
                ((currentForum.getForumID()).toString());

                Element ForumNameElement = new
                Element("ForumName").addContent(currentForum.getForumName());

                Element ForumDescriptionElement = new
                Element("ForumDescription").addContent
                (currentForum.getForumDescription());

                Element ForumElement = new Element("Forum");
                ForumElement.addContent(ForumIDElement);
                ForumElement.addContent(ForumNameElement);
                ForumElement.addContent(ForumDescriptionElement);

                rootElement.addContent(ForumElement);

        }

        return (new Document(rootElement));
}

/*
 * Find Forum method, finds a method given it's primary key
 */

 public Document findForum(Document forumID) throws RemoteException, FinderException
 {

         Integer forumid = new Integer(forumID.getRootElement().getText());

         try {
                 ForumHome home = (ForumHome) getHome("java:comp/env/ejb/Forum");
                 Forum currentForum = home.findByPrimaryKey(forumid);
                 Element ForumIDElement = new
                 Element("ForumID").addContent
                 ((currentForum.getForumID()).toString());

                 Element ForumNameElement = new
                 Element("ForumName").addContent(currentForum.getForumName());

                 Element ForumDescriptionElement = new
                 Element("ForumDescription").addContent
                 (currentForum.getForumDescription());

                 Element ForumElement = new Element("Forum");

                 ForumElement.addContent(ForumIDElement);
                 ForumElement.addContent(ForumNameElement);
                 ForumElement.addContent(ForumDescriptionElement);

                 return (new Document(ForumElement));
         }
         catch(Exception e) {
                 return (new Document(new Element("Response").addContent("Failed")));
         }
 }

/*
 * Find Forum method, finds a method given it's primary key
 */

 private Element findForumAsElement(Document forumID) throws RemoteException,
 FinderException {

         Integer forumid = new Integer(forumID.getRootElement().getText());

         try {
                 ForumHome home = (ForumHome) getHome("java:comp/env/ejb/Forum");
                 Forum currentForum = home.findByPrimaryKey(forumid);
                 Element ForumIDElement = new Element("ForumID").addContent
                 ((currentForum.getForumID()).toString());

                 Element ForumNameElement = new
```

```
                    Element("ForumName").addContent(currentForum.getForumName());

                   Element ForumDescriptionElement = new
                   Element("ForumDescription").addContent(currentForum.getForumDescription
                   ());
                      Element ForumElement = new Element("Forum");

                      ForumElement.addContent(ForumIDElement);
                      ForumElement.addContent(ForumNameElement);
                      ForumElement.addContent(ForumDescriptionElement);

                      return (ForumElement);
               }
               catch(Exception e) {
                      return (new Element("Response").addContent("Failed"));
               }
       }

       /*
        * Returns an Array of Forum object for a given username
        */

   public Document findForumsForUser(Document userDetails) throws RemoteException,
   FinderException {

              Element RootElement = userDetails.getRootElement();
              List UserListElement = RootElement.getChildren();
              Element [] UserElements = (Element[])UserListElement.toArray(new Element[0]);

              String username = (UserElements[0].getChild("UserName").getText());
              String superuser = (UserElements[0].getChild("SuperUser").getText());

              boolean adminStatus = false;

              if(superuser.equals("Yes")) adminStatus = true;

              int ForumCounter = 0;

              Element rootElement = new Element("ForumList");

              try {
                      ForumUserHome home = (ForumUserHome)
                      getHome("java:comp/env/ejb/ForumUser");
                      Collection user_forums = home.findAll();
                      ForumUser[] user_forums_array = (ForumUser[]) user_forums.toArray(new
                      ForumUser[0]);

                      for(int counter=0; counter<user_forums_array.length; counter++) {
                              if(user_forums_array[counter].getUserName().equals(username)){
                              Element tempForum = findForumAsElement(new Document(new
                              Element("ForumID").addContent(user_forums_array[counter].
                              getForumID().toString())));
                                      if(adminStatus==true) {
                                      if(user_forums_array[counter].isAdministrator()) {
                                                    rootElement.addContent(tempForum);
                                                    ForumCounter++;
                                      }
                              }
                              else {
                              rootElement.addContent(tempForum);
                              ForumCounter++;
                              }
                      }
              }

              System.out.println("Found " + ForumCounter + " forums for " + username);

              return (new Document(rootElement));
       }
       catch(Exception e) {
              //something happened on the way to heaven
              System.out.println("Exception occurred while finding forums\n" + e);
              return (new Document(new Element("Response").addContent("Failed")));
       }

    }
```

```
        /*
         * Is user allowed to use a particular forum
         */

    public Document isUserAllowed(Document userDetails) throws RemoteException,
    FinderException {

                Element rootElement = userDetails.getRootElement();

                Element responseElement = new Element("Response");

                String username = rootElement.getChild("UserName").getText();
                Integer forumID = new Integer((rootElement.getChild("ForumID")).getText());

                try {
                        ForumUserHome home = (ForumUserHome)
                        getHome("java:comp/env/ejb/ForumUser");
                        ForumUser isAllowed = home.findByPrimaryKey(new ForumUserPK(username,
                        forumID));
                        if (isAllowed == null) responseElement.addContent("Failed");
                        else responseElement.addContent("Done");
                }
                catch(Exception e) {
                        //something happened on the way to heaven
                        System.out.println("Exception occurred while finding forums\n" + e);
                        responseElement.addContent("Failed");
                }

                return (new Document(responseElement));

        }

} // end of class file
```

ForumManager.java

```
/*
 * ForumManager.java
 *
 * Defines the remote interface for the ForumManager Session Bean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 24th June 2001
 *
 */

package freeforums.forum;

import javax.ejb.EJBObject;
import javax.ejb.*;
import java.rmi.RemoteException;
import java.util.Vector;

import org.jdom.*;

public interface ForumManager extends EJBObject {

        public Document addForum(Document forumDetails) throws RemoteException;

        public Document findAll() throws RemoteException, FinderException;

        public Document findForumsForUser(Document userDetails) throws RemoteException,
        FinderException;

        public Document findForum(Document forumID) throws RemoteException, FinderException;

        public Document isUserAllowed(Document userDetails) throws RemoteException,
        FinderException;

}
```

ForumHome.java

```
/*
 * ForumHome.java
 *
 * Defines the home interface for the ForumBean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.forum;

 import java.rmi.RemoteException;
 import javax.ejb.*;
 import java.util.Collection;

 public interface ForumHome extends EJBHome {

        public Forum create (Integer forumID, String forumName, String forumDescription)
        throws RemoteException, CreateException;

        public Forum findByPrimaryKey (Integer forumID) throws RemoteException,
        FinderException;

        public Collection findAll() throws RemoteException, FinderException;

 }
```

ForumBean.java

```
/*
 * ForumBean.java
 *
 * FormBean implementation
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.forum;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public class ForumBean implements EntityBean {

        transient private EntityContext context;

        public String forumName, forumDescription;
        public Integer forumID;

        public Integer ejbCreate(Integer forumID, String forumName, String forumDescription)
        throws CreateException {

                this.forumID = forumID;
                this.forumName = forumName;
                this.forumDescription = forumDescription;
                return null;

        }

        public void ejbPostCreate(Integer forumID, String forumName, String forumDescription)
        { }

        public void setEntityContext(EntityContext context) { this.context = context; }

        public void unsetEntityContext() { context = null; }
```



```
        public void ejbActivate() { }
        public void ejbPassivate() { }
        public void ejbLoad() { }
        public void ejbStore() { }
        public void ejbRemove() { }

        /* The following are the implementation of the method from the
           Forum Remote interface
         */

        public String getForumName() throws RemoteException { return forumName; }
        public void setForumName(String forumName) throws RemoteException
        { this.forumName = forumName; }

        public String getForumDescription() throws RemoteException
        { return forumDescription; }

        public void setForumDescription() throws RemoteException
        { this.forumDescription = forumDescription; }

        public Integer getForumID() throws RemoteException { return forumID; }
        public void setForumID(Integer forumID) throws RemoteException
        { this.forumID = forumID; }

 }
```

Forum.java

```
/*
 * Forum.java
 *
 * Defines the remote interface for the ForumBean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.forum;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public interface Forum extends EJBObject {

        public String getForumName() throws RemoteException;
        public void setForumName(String forumName) throws RemoteException;

        public String getForumDescription() throws RemoteException;
        public void setForumDescription() throws RemoteException;

        public Integer getForumID() throws RemoteException;
        public void setForumID(Integer forumID) throws RemoteException;

 }
```

UserManagerHome.java

```
/*
 * UserManagerHome.java
 *
 * Desfines the Home interface for the
 * Session bean that represents the User Management
 * facilities for the FreeForums System.
 *
 * @author: Devraj Mukherjee
 *
 * Created: 24th June 2001
```



```
 *
 */

 package freeforums.user;

 import java.io.Serializable;
 import java.rmi.RemoteException;
 import javax.ejb.CreateException;
 import javax.ejb.EJBHome;

 public interface UserManagerHome extends EJBHome {

        public UserManager create() throws RemoteException, CreateException;

 }
```

UserManagerBean.java

```
/*
 * UserManagerBean.java
 *
 * Session bean that represents the User Management
 * facilities for the FreeForums System.
 *
 * @author: Devraj Mukherjee
 *
 * Created: 24th June 2001
 *
 */

 package freeforums.user;

 import java.rmi.RemoteException;
 import javax.ejb.*;
 import javax.naming.*;
 import javax.rmi.*;
 import java.rmi.ServerException;
 import java.util.Collection;
 import java.util.List;

 import org.jdom.*;

 public class UserManagerBean implements SessionBean {

        /**
        Helper function to get the home interface of a specified Bean. This is used by
        most of the methods in this class.
        */
        private Object getHome (String path) {
                try {
                        return (new InitialContext().lookup(path));
                }
               catch (NamingException e) {
                       System.err.println("An exception occured " + e);
                       return null;
               }
        }

        /**
        Empty method body
        */
        public UserManagerBean() {}
        /**
        Empty method body
        */
        public void ejbCreate() {}
        /**
        Empty method body
        */
        public void ejbRemove() {}
        /**
        Empty method body
        */
```

```
public void ejbActivate() {}
/**
Empty method body
*/
public void ejbPassivate() {}
/**
Empty method body
*/
public void setSessionContext(SessionContext sc) {}

public Document modifyUser(Document userDetails) throws RemoteException {

        /*****
         * Parse XML data
         *
         * This modifyUser method now supports multiple users
         *****/

        String userName,password,realName,emailAddress;
        boolean superUser;

        Element userListElement = userDetails.getRootElement();
        Element responseElement = new Element("Response");

        // Extract the User Elements

        List userElementList = userListElement.getChildren();
        Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

        int counter=0;

        UserHome home = (UserHome) getHome("java:comp/env/ejb/User");

        userName = (userElements[counter].getChild("UserName")).getText();
        password = (userElements[counter].getChild("Password")).getText();
        realName = (userElements[counter].getChild("RealName")).getText();
        emailAddress = (userElements[counter].getChild("EmailAddress")).getText();

        if((userElements[counter].getChild("SuperUser")).getText().equals("Yes"))
        superUser=true;
        else superUser=false;

        try {
                User existingUser = home.findByPrimaryKey(userName);
                existingUser.setPassword(password);
                existingUser.setRealName(realName);
                existingUser.setEmailAddress(emailAddress);
                existingUser.setSuperUser(superUser);
                responseElement.addContent("Done");
        }
        catch(Exception e) {
                // error occured
                responseElement.addContent("Failed");
        }

        return (new Document(responseElement));
}

/**************
 *
 * Method to add new usrs
 *
 **************/

public Document addUser(Document userDetails) throws RemoteException {

        /*****
         * Parse XML data
         *****/

        String userName,password,realName,emailAddress;
        boolean superUser;

        Element userListElement = userDetails.getRootElement();
        Element responseElement = new Element("Response");

        // Extract the User Elements
```

```
            List userElementList = userListElement.getChildren();
            Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

            UserHome home = (UserHome) getHome("java:comp/env/ejb/User");

            int counter=0;

            // Get all the details off the user element

            userName = (userElements[counter].getChild("UserName")).getText();
            password = (userElements[counter].getChild("Password")).getText();
            realName = (userElements[counter].getChild("RealName")).getText();
            emailAddress = (userElements[counter].getChild("EmailAddress")).getText();

            if((userElements[counter].getChild("SuperUser")).getText().equals("Yes"))
            superUser=true;
            else superUser=false;

            try {
                    User existingUser = home.findByPrimaryKey(userName);
                    String oldUserName = existingUser.getUserName();
                    // Force loading of data here
                    // If we get here, the user is already in the collection
                    System.out.println("Could not complete request \n User exists " +
                    oldUserName);
                    responseElement.addContent("Duplicate");
           }
           // The database does not have an User with the same username
           // so it is now safe to try and add a new user

           catch (Exception e) {
                  System.out.println("Added new user with the userName = " + userName);
                  try {
                          User user = home.create(userName);
                          user.setPassword(password);
                          user.setRealName(realName);
                          user.setEmailAddress(emailAddress);
                          user.setSuperUser(superUser);
                          responseElement.addContent("Done");
                  }
                  catch (Exception ex) {
                          responseElement.addContent("Failed");
                  }
           }

           return (new Document(responseElement));

   }

   /*
    * Return a user object
    */

public Document getUser(Document userDocument) throws RemoteException {

           Element rootElement = userDocument.getRootElement();
           String username = rootElement.getText();

           UserHome home = (UserHome) getHome("java:comp/env/ejb/User");
           try {
                  User existingUser = home.findByPrimaryKey(username);
              Element user = new Element("User");

                  user.addContent(new
                  Element("UserName").addContent(existingUser.getUserName()));
                  user.addContent(new
                  Element("Password").addContent(existingUser.getPassword()));
                  user.addContent(new
                  Element("RealName").addContent(existingUser.getRealName()));
                  user.addContent(new
                  Element("EmailAddress").addContent(existingUser.getEmailAddress()));

                  if(existingUser.isSuperUser()) user.addContent(new
                  Element("SuperUser").addContent("Yes"));
                  else user.addContent(new Element("SuperUser").addContent("No"));
```

```
                        Element rootResponseElement = new Element("UserList");
                        rootResponseElement.addContent(user);

                        return (new Document(rootResponseElement));

                }
                catch(Exception e) {
                        Element rootResponseElement = new Element("Response");
                        rootResponseElement.addContent("Not Found");
                        return (new Document(rootResponseElement));
                }
        }

        /*
         * Login process for a user
         *
         * Check for the existance of the user account and then check the password.
         *
         */

        public Document loginUser(Document loginDocument) throws RemoteException {

                UserHome home = (UserHome) getHome("java:comp/env/ejb/User");

                Element userListElement = loginDocument.getRootElement();
                List userElementList = userListElement.getChildren();
                Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

                String userName = userElements[0].getChild("UserName").getText();
                String password = userElements[0].getChild("Password").getText();

                Element responseElement = new Element("Response");

                try {
                        User existingUser = home.findByPrimaryKey(userName); // find the user
                        if(existingUser.getPassword().equals(password))
                        return getUser(new Document(new
                        Element("UserName").addContent(userName)));
                        else responseElement.addContent("Failed");
                }
                catch(Exception e) {
                        System.out.println("Could not complete request \n" + e);
                        responseElement.addContent("Failed");
                }

                return (new Document(responseElement));

        }

        /*
         * add a user to a forum
         *
         * if the user already is subscribed to that forum then
         * this method will return false, else it will return true.
         */

        public Document addUserToForum(Document userDetails) throws RemoteException {

                Element rootElement = userDetails.getRootElement();
                Element responseElement = new Element("Response");

                List userElementList = rootElement.getChildren();
                Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

                String userName = userElements[0].getChild("UserName").getText();
                Integer forumID = new Integer(userElements[0].getChild("ForumID").getText());
                String adminstatus = userElements[0].getChild("Administrator").getText();

                boolean administrator = false;

                if(adminstatus.equals("Yes")) administrator = true;

                ForumUserHome home = (ForumUserHome) getHome("java:comp/env/ejb/ForumUser");

                try {
                        ForumUser existingForumUser = home.findByPrimaryKey(new
ForumUserPK(userName, forumID));
```

```
                String existingUserName = existingForumUser.getUserName();
                responseElement.addContent("Done");
                System.out.println("User " + userName + " allowed to use " + forumID);
        }
        catch(Exception e) {
                try {
                        ForumUser tempForumUser =
                        home.create(new ForumUserPK(userName, forumID));
                        tempForumUser.setAdministrator(administrator);
                        responseElement.addContent("Done");
                        System.out.println("User " + userName + " allowed to use " +
                        forumID);
                }
                catch (Exception ex) {
                        responseElement.addContent("Failed");
                        System.out.println("Failed - User " + userName + " allowed to
                        use " + forumID);
                }
        }

        return (new Document(responseElement));
}

/***
 * remove a user from a forum
 ***/

public Document removeUserFromForum(Document userDetails) throws RemoteException {

        Element rootElement = userDetails.getRootElement();
        Element responseElement = new Element("Response");

        List userElementList = rootElement.getChildren();
        Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

        String userName = userElements[0].getChild("UserName").getText();
        Integer forumID = new Integer(userElements[0].getChild("ForumID").getText());

        try {
                ForumUserHome home = (ForumUserHome)
                getHome("java:comp/env/ejb/ForumUser");
                ForumUser existingForumUser = home.findByPrimaryKey(new
                ForumUserPK(userName, forumID));
                existingForumUser.remove();
                responseElement.addContent("Done");
        }
        catch(Exception e) {
                responseElement.addContent("Failed");
        }

        return (new Document(responseElement));
}


/*****
 *
 * findAll()
 *
 * Creates a XML representation of the User List based on User.dtd
 *
 *****/

 public Document findAll() throws RemoteException, FinderException {

         UserHome home = (UserHome) getHome("java:comp/env/ejb/User");
         Collection all = home.findAll();
         User [] userArray = (User[]) all.toArray(new User[0]);
         DocType dtd = new
         DocType("UserList","http://devraj.org/dtd/freeforums/User.dtd");

         Element root = new Element("UserList");

         for(int counter=0; counter<userArray.length; counter++) {

                 Element user = new Element("User");
                 User currentUser = userArray[counter];
```

```
                        user.addContent(new
                        Element("UserName").addContent(currentUser.getUserName()));
                        user.addContent(new
                        Element("Password").addContent(currentUser.getPassword()));
                        user.addContent(new
                        Element("RealName").addContent(currentUser.getRealName()));
                        user.addContent(new
                        Element("EmailAddress").addContent(currentUser.getEmailAddress()));

                        if(currentUser.isSuperUser()) user.addContent(new
                        Element("SuperUser").addContent("Yes"));
                        else user.addContent(new Element("SuperUser").addContent("No"));

                        root.addContent(user);

                }

                return (new Document(root,dtd));
        }

} // this bracket ends the class file.
```

UserManager.java

```
/******
 * UserManager.java
 *
 * Defines the Remote Interface for the
 * Session bean that represents the User Management
 * facilities for the FreeForums System.
 *
 * @author: Devraj Mukherjee
 *
 * Created: 24th June 2001
 *
 ******/

package freeforums.user;

import javax.ejb.EJBObject;
import javax.ejb.*;
import java.rmi.RemoteException;
import org.jdom.*;

public interface UserManager extends EJBObject {

        public Document addUser(Document userDetails) throws RemoteException;

        public Document modifyUser(Document userDetails) throws RemoteException;

        public Document getUser(Document userName) throws RemoteException;

        public Document loginUser(Document loginDocument) throws RemoteException;

        public Document addUserToForum(Document userDetails) throws RemoteException;

        public Document removeUserFromForum(Document userDetails) throws RemoteException;

        public Document findAll() throws RemoteException, FinderException;

}
```

UserHome.java

```
/*
 * UserHome.java
 *
 * Home Interface for UserBean
 *
```

```
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.user;

 import java.rmi.RemoteException;
 import javax.ejb.*;
 import java.util.Collection;

 public interface UserHome extends EJBHome {

        public User create(String username) throws RemoteException, CreateException;

        public User findByPrimaryKey(String username) throws RemoteException,
        FinderException;

        public Collection findAll() throws RemoteException, FinderException;

 }
```

UserBean.java

```
/*
 * UserBean.java
 *
 * UserBean implementation
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.user;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public class UserBean implements EntityBean {

        transient private EntityContext context;

        public String userName, password, realName, emailAddress;
        public boolean superUser;

        public String ejbCreate(String userName) {

                this.userName = userName;
                return null;

        }

        public void ejbPostCreate(String userName) { }

        public void setEntityContext(EntityContext context) { this.context = context; }

        public void unsetEntityContext() { context = null; }

        public void ejbActivate() { }
        public void ejbPassivate() { }
        public void ejbLoad() { }
        public void ejbStore() { }
        public void ejbRemove() { }

        /* The following are the implementation of the method from the
           User Remote interface
         */

        public String getUserName() throws RemoteException { return userName; }
        public void setUserName(String userName) throws RemoteException
        { this.userName = userName; }
```



```
        public String getPassword() throws RemoteException { return password; }
        public void setPassword(String password) throws RemoteException
        { this.password = password; }

        public String getRealName() throws RemoteException { return realName; }
        public void setRealName(String realName) throws RemoteException
        { this.realName = realName; }

        public String getEmailAddress() throws RemoteException { return emailAddress; }
        public void setEmailAddress(String emailAddress) throws RemoteException
        { this.emailAddress = emailAddress; }

        public boolean isSuperUser() throws RemoteException { return superUser; }
        public void setSuperUser(boolean superUser) throws RemoteException
        { this.superUser = superUser; }


 }
```

User.java

```
/*
 * User.java
 *
 * Remote Interface for UserBean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.user;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public interface User extends EJBObject {

        public String getUserName() throws RemoteException;
        public void setUserName(String userName) throws RemoteException;

        public String getPassword() throws RemoteException;
        public void setPassword(String password) throws RemoteException;

        public String getRealName() throws RemoteException;
        public void setRealName(String realName) throws RemoteException;

        public String getEmailAddress() throws RemoteException;
        public void setEmailAddress(String emailAddress) throws RemoteException;

        public boolean isSuperUser() throws RemoteException;
        public void setSuperUser(boolean superUser) throws RemoteException;

 }
```

ForumUserPK.java

```
/*
 * ForumUserPK.java
 *
 * Defines a composite Primary Key for ForumUser
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.user;
```



```
import java.io.Serializable;

public class ForumUserPK implements Serializable {

        public String userName;
        public Integer forumID;

        public ForumUserPK() { }

        public ForumUserPK(String userName, Integer forumID) {

                this.userName = userName;
                this.forumID = forumID;

        }

        public boolean equals(Object obj) {

                if (obj==null || !(obj instanceof ForumUserPK)) return false;

                if ((((ForumUserPK)obj).userName == this.userName) &&
                (((ForumUserPK)obj).forumID == this.forumID))
                 return true;

                return false; //default value that will be returned

        }

        public int hashCode() { return forumID.intValue(); }

        public String toString() { return userName + " " + forumID; }
}
```

ForumUserHome.java

```
/*
 * ForumUserHome.java
 *
 * Defines the Home interface for the ForumUser Bean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

package freeforums.user;

import java.rmi.RemoteException;
import java.util.Collection;
import javax.ejb.*;

public interface ForumUserHome extends EJBHome {

        public ForumUser create(ForumUserPK key, boolean administrator) throws
        RemoteException, CreateException;

        public ForumUser create(ForumUserPK key) throws RemoteException, CreateException;

        public ForumUser findByPrimaryKey(ForumUserPK primaryKey) throws RemoteException,
        FinderException;

        public Collection findAll() throws RemoteException, FinderException;

}
```

ForumUserBean.java

```
/*
```

```
 * ForumUserBean.java
 *
 * Form User Bean implementation
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.user;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public class ForumUserBean implements EntityBean {

        transient private EntityContext context;

        public ForumUserPK key;
        public boolean administrator;

        public ForumUserPK ejbCreate(ForumUserPK key, boolean administrator) {

                this.key = key;
                this.administrator = administrator;
                return null;

        }

        public ForumUserPK ejbCreate(ForumUserPK key) {

                this.key = key;
                this.administrator = false;
                return null;

        }

        public void ejbPostCreate(ForumUserPK key, boolean administrator) { }
        public void ejbPostCreate(ForumUserPK key) { }

        public void setEntityContext(EntityContext context) { this.context = context; }

        public void unsetEntityContext() { context = null; }

        public void ejbActivate() { }
        public void ejbPassivate() { }
        public void ejbLoad() { }
        public void ejbStore() { }
        public void ejbRemove() { }

        /* The following are the implementation of the method from the
           ForumUser Remote interface
         */

        public String getUserName() throws RemoteException { return key.userName; }
        public void setUserName(String userName) throws RemoteException
        { key.userName = userName; }

        public Integer getForumID() throws RemoteException { return key.forumID; }
        public void setForumID(Integer forumID) throws RemoteException
        { key.forumID = forumID; }

        public boolean isAdministrator() throws RemoteException { return administrator; }
        public void setAdministrator(boolean administrator) throws RemoteException
        { this.administrator = administrator; }

}
```

ForumUser.java

```
/*
 * ForumUser.java
 *
```

```
 * Defines the Remote interface for the ForumUser Bean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

package freeforums.user;

import java.rmi.RemoteException;
import javax.ejb.*;

public interface ForumUser extends EJBObject {

        public String getUserName() throws RemoteException;
        public void setUserName(String userName) throws RemoteException;

        public Integer getForumID() throws RemoteException;
        public void setForumID(Integer forumID) throws RemoteException;

        public boolean isAdministrator() throws RemoteException;
        public void setAdministrator(boolean administrator) throws RemoteException;

}
```

ReadMessageUserPK.java

```
/*
 * ReadMessageUserPK.java
 *
 * Defines a composite Primary Key for Read Messages
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

package freeforums.message;

import java.io.Serializable;

public class ReadMessagePK implements Serializable {

        public String userName;
        public Integer messageID;

        public ReadMessagePK() { }

        public ReadMessagePK(String userName, Integer messageID) {

                this.userName = userName;
                this.messageID = messageID;

        }

        public boolean equals(Object obj) {

                if (obj==null || !(obj instanceof ReadMessagePK)) return false;

                if ((((ReadMessagePK)obj).userName == this.userName) &&
                (((ReadMessagePK)obj).messageID == this.messageID))
                 return true;

                return false; //default value that will be returned

        }

        public int hashCode() { return messageID.intValue(); }

        public String toString() { return userName + " " + messageID; }
}
```

ReadMessageHome.java

```
/*
 * ReadMessageHome.java
 *
 * Defines the home interface for Read Messages
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.message;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public interface ReadMessageHome extends EJBHome {

        public ReadMessage create(ReadMessagePK key) throws RemoteException, CreateException;

        public ReadMessage findByPrimaryKey(ReadMessagePK primaryKey) throws
        RemoteException, FinderException;

 }
```

ReadMessageBean.java

```
/*
 * ReadMessageBean.java
 *
 * Defines the message bean implementation
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.message;

 import java.rmi.RemoteException;
 import javax.ejb.*;

 public class ReadMessageBean implements EntityBean {

        transient private EntityContext context;

        public ReadMessagePK key;

        public ReadMessagePK ejbCreate(ReadMessagePK key) {

                this.key = key;
                return null;

        }

        public void ejbPostCreate(ReadMessagePK key) { }

        public void setEntityContext(EntityContext context) { this.context = context; }

        public void unsetEntityContext() { context = null; }

        public void ejbActivate() { }
        public void ejbPassivate() { }
        public void ejbLoad() { }
        public void ejbStore() { }
        public void ejbRemove() { }
```



```
        /* The following are the implementation of the method from the
           ReadMessage Remote interface
         */

        public String getUserName() throws RemoteException { return key.userName; }
        public void setUserName(String userName) throws RemoteException
        { key.userName = userName; }

        public Integer getMessageID() throws RemoteException { return key.messageID; }
        public void setMessageID(Integer messageID) throws RemoteException
        { key.messageID = messageID; }

}
```

ReadMessage.java

```
/*
 * ReadMessage.java
 *
 * Defiens the remote interface for Read Messages
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

package freeforums.message;

import java.rmi.RemoteException;
import javax.ejb.*;

public interface ReadMessage extends EJBObject {

        public String getUserName() throws RemoteException;
        public void setUserName(String userName) throws RemoteException;

        public Integer getMessageID() throws RemoteException;
        public void setMessageID(Integer messageID) throws RemoteException;

}
```

MessageManagerHome.java

```
/*
 * MessageManagerHome.java
 *
 * Defines a Session bean for forum based functions
 *
 * @author: Devraj Mukherjee
 *
 * Created: 04th July 2001
 *
 */

package freeforums.message;

import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface MessageManagerHome extends EJBHome {

        public MessageManager create() throws RemoteException, CreateException;

}
```


MessageManagerBean.java

```
/*
 * MessageManagerBean.java
 *
 * Message Manager bean represent functions for
 * managing the forums
 *
 * @author: Devraj Mukherjee
 *
 * Created: 07th July 2001
 *
 */

 package freeforums.message;

 import java.rmi.RemoteException;
 import javax.ejb.*;
 import javax.naming.*;
 import javax.rmi.*;
 import java.rmi.ServerException;
 import java.util.Date;
 import java.util.Collection;
 import java.util.Vector;

 import freeforums.user.*;
 import freeforums.util.*;

 import org.jdom.*;
 import java.util.List;

 public class MessageManagerBean implements SessionBean {

        /**
        Helper function to get the home interface of a specified Bean. This is used by
        most of the methods in this class.
        */
        private Object getHome (String path) {
                try {
                        return (new InitialContext().lookup(path));
                }
               catch (NamingException e) {
                       System.err.println("An exception occured " + e);
                       return null;
               }
        }

        /**
        Empty method body
        */
        public MessageManagerBean() {}
        /**
        Empty method body
        */
        public void ejbCreate() {}
        /**
        Empty method body
        */
        public void ejbRemove() {}
        /**
        Empty method body
        */
        public void ejbActivate() {}
        /**
        Empty method body
        */
        public void ejbPassivate() {}
        /**
        Empty method body
        */
        public void setSessionContext(SessionContext sc) {}

        /**
         Gets a message given it's message id
         **/
```

```
public Document getMessage(Document message) throws RemoteException,
FinderException {

        Integer messageID = new Integer(message.getRootElement().getText());

        try {
                MessageHome home = (MessageHome) getHome("java:comp/env/ejb/Message");
                Message existingMessage = home.findByPrimaryKey(messageID);

                Date MessageDate = existingMessage.getMessageDate();
                String DisplayDate = MessageDate.getDate() + "/" +
                 MessageDate.getMonth() + "/" + (1900 + MessageDate.getYear());

                Element MessageElement = new Element("Message");

                MessageElement.addContent(new Element("MessageID").addContent
                (existingMessage.getMessageID().toString()));

                MessageElement.addContent(new Element("Subject").addContent
                (existingMessage.getSubject()));

                MessageElement.addContent(new Element("MessageBody").addContent
                (existingMessage.getDescription()));

                MessageElement.addContent(new Element("MessageDate").addContent
                (DisplayDate));

                MessageElement.addContent(new Element("UserName").addContent
                (existingMessage.getUserName()));

                MessageElement.addContent(new Element("ForumID").addContent
                (existingMessage.getForumID().toString()));

               MessageElement.addContent(new Element("ReplyToID").addContent
               (existingMessage.getReplyToID().toString()));

                Element RootElement = new Element("MessageList");
                RootElement.addContent(MessageElement);

                return (new Document(RootElement));
        }
        catch(Exception e) {
                return (new Document(new Element("Response").addContent("Failed")));
        }
}

/**
 Adds a new message to the collection
 **/

public Document addMessage(Document MessageInformation) throws
RemoteException, FinderException {

        Element RootElement = MessageInformation.getRootElement();

        List MessageListElement = RootElement.getChildren();
        Element [] MessageElements = (Element[])MessageListElement.
        toArray(new Element[0]);

        String subject = MessageElements[0].getChild("Subject").getText();
        String description = MessageElements[0].getChild("MessageBody").getText();
        Integer forumID = new Integer(MessageElements[0].
        getChild("ForumID").getText());
        String userName = MessageElements[0].getChild("UserName").getText();
        Integer replyToID = new Integer(MessageElements[0].
        getChild("ReplyToID").getText());


        String identifier = "Message";
        Integer id=null;

        try {

                /**
                 Process of creating AutoNumbers
                 **/
```

```
                AutoNumberHome ANHome = (AutoNumberHome)
                getHome("java:comp/env/ejb/AutoNumber");
                AutoNumber AutoNumberFactory = ANHome.findByPrimaryKey(identifier);
                id = AutoNumberFactory.getAutoNumberValue();

        }
        catch (Exception e) {
                /**
                 This Key was not found in the storage so
                 create a new one with this name
                 **/
                 try {
                        AutoNumberHome ANHome = (AutoNumberHome)
                        getHome("java:comp/env/ejb/AutoNumber");
                        AutoNumber AutoNumberFactory = ANHome.create(identifier);
                        id = AutoNumberFactory.getAutoNumberValue();
                 }
                 catch(Exception exp) {
                        /**
                         Could not create this primary key
                         **/
                  }

         }

         if (id==null) return (new Document(new
         Element("Response").addContent("Failed")));

         try {

                 /**
                 Process of creating the Forum Object
                 **/

                MessageHome home = (MessageHome) getHome("java:comp/env/ejb/Message");
                Message tempMessage =
                home.create(id,subject,description,forumID,userName);
                tempMessage.setReplyToID(replyToID);
                tempMessage.setMessageDate(new Date());
                return (new Document(new Element("Response").addContent("Done")));

        }
        catch(Exception e) {
                //do nothing
                System.out.println("Exception occured while creating Forum \n" + e);
                return (new Document(new Element("Response").addContent("Failed")));
        }

 } // end add message method

/**
 Marks a message to be read
 **/

public Document markReadMessage(Document MessageInfo) throws
RemoteException, FinderException {

        Element rootElement = MessageInfo.getRootElement();
        String userName = rootElement.getChild("UserName").getText();
        Integer messageID = new Integer(rootElement.getChild("MessageID").getText());

        try {
                ReadMessageHome home = (ReadMessageHome)
                getHome("java:comp/env/ejb/ReadMessage");
                ReadMessage tempReadMessage = home.create(new ReadMessagePK(userName,
                messageID));
                return (new Document(new Element("Response").addContent("Done")));
         }
         catch(Exception e) {
                 //do nothing
                 System.out.println("Exception while marking message read \n" + e);
                return (new Document(new Element("Response").addContent("Failed")));
         }
} // end method

/**
 Checks to see if a message is read by a user
```

```
 **/

public Document isReadMessage(Document MessageInfo) throws
RemoteException, FinderException {

        Element rootElement = MessageInfo.getRootElement();
        String userName = rootElement.getChild("UserName").getText();
        Integer messageID = new Integer(rootElement.getChild("MessageID").getText());

        try {
                ReadMessageHome home = (ReadMessageHome)
                getHome("java:comp/env/ejb/ReadMessage");
                ReadMessage tempReadMessage = home.findByPrimaryKey(new
                ReadMessagePK(userName, messageID));
                if(tempReadMessage==null) return (new Document(new
                Element("Response").addContent("Read")));
                else return (new Document(new
                Element("Response").addContent("UnRead")));
         }
         catch(Exception e) {
                //do nothing
                return (new Document(new Element("Response").addContent("UnRead")));
         }
} // end method


 /**
  finds messages for a forum id
  **/

 public Document findMessagesForForum(Document ForumID) throws
 RemoteException, FinderException {

          Integer forumID = new Integer(ForumID.getRootElement().getText());

          Element rootElement = new Element("MessageList");

          try {
                  MessageHome home = (MessageHome)
                  getHome("java:comp/env/ejb/Message");
                  Collection messages = home.findAll();
                  Message[] messageArray = (Message[])
                  messages.toArray(new Message[0]);

                  for(int counter=0; counter<messageArray.length; counter++) {

                          if((messageArray[counter].getForumID()).equals(forumID)) {

                                  // Convert the message to a XML document

                                 Message existingMessage = messageArray[counter];

                                 Date MessageDate = existingMessage.getMessageDate();

                                 String DisplayDate = MessageDate.getDate() + "/" +
                                 MessageDate.getMonth() + "/" + (1900 +
                                 MessageDate.getYear());

                                 Element MessageElement = new Element("Message");
                                 MessageElement.addContent(new
                                 Element("MessageID").addContent
                                 (existingMessage.getMessageID().toString()));

                                 MessageElement.addContent(new
                                 Element("Subject").addContent
                                (existingMessage.getSubject()));

                                 MessageElement.addContent(new
                                 Element("MessageBody").addContent
                                 (existingMessage.getDescription()));

                                 MessageElement.addContent(new
                                 Element("MessageDate").addContent(DisplayDate));

                                 MessageElement.addContent(new
                                 Element("UserName").addContent
                                 (existingMessage.getUserName()));
```

```
                                        MessageElement.addContent(new
                                        Element("ForumID").addContent
                                        (existingMessage.getForumID().toString()));

                                        MessageElement.addContent(new
                                        Element("ReplyToID").addContent
                                        (existingMessage.getReplyToID().toString()));

                                        rootElement.addContent(MessageElement);

                                }

                        }

                        return (new Document(rootElement));
                }
                catch(Exception e) {
                        //something happened on the way to heaven
                        System.out.println
                        ("Exception occurred while finding messages\n" + e);
                        return (new Document(new Element("Response").addContent("Done")));
                }
        } // end method

} // end class file
```

MessageManager.java

```
/*
 * MessageManager.java
 *
 * Defines the remote interface for the MessageManager Session Bean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 24th June 2001
 *
 */

package freeforums.message;

import javax.ejb.EJBObject;
import javax.ejb.*;
import java.rmi.RemoteException;

import org.jdom.*;

public interface MessageManager extends EJBObject {

        public Document getMessage(Document message) throws RemoteException, FinderException;

        public Document addMessage(Document MessageInformation) throws RemoteException,
        FinderException;

        public Document markReadMessage(Document MessageInfo) throws RemoteException,
        FinderException;

        public Document isReadMessage(Document MessageInfo) throws RemoteException,
        FinderException;

        public Document findMessagesForForum(Document ForumID) throws RemoteException,
        FinderException;

}
```

MessageHome.java

```
/*
 * MessageHome.java
 *
```

```
 * Defines the home interface for the message bean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.message;

 import java.rmi.RemoteException;
 import java.util.Collection;
 import javax.ejb.*;

 public interface MessageHome extends EJBHome {

        public Message create(Integer messageID, String subject, String description, Integer
        forumID, String userName) throws RemoteException, CreateException;

        public Message findByPrimaryKey(Integer messageID) throws RemoteException,
        FinderException;

        public Collection findAll() throws RemoteException, FinderException;

 }
```

MessageBean.java

```
/*
 * MessageBean.java
 *
 * Defines the message bean implementation
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.message;

 import java.rmi.RemoteException;
 import java.util.Date;
 import javax.ejb.*;

 public class MessageBean implements EntityBean {

        transient private EntityContext context;

        public Integer messageID;
        public String subject;
        public String description;
        public Integer replyToID;
        public Integer forumID;
        public String attachmentFile;
        public String userName;
        public Date messageDate;

        public Integer ejbCreate(Integer messageID, String subject, String description,
        Integer forumID, String userName) {

                this.messageID = messageID;
                this.subject = subject;
                this.description = description;
                this.forumID = forumID;
                this.userName = userName;
                return null;

        }

        public void ejbPostCreate(Integer messageID, String subject, String description,
        Integer forumID, String userName) { }

        public void setEntityContext(EntityContext context) { this.context = context; }
```

```
        public void unsetEntityContext() { context = null; }

        public void ejbActivate() { }
        public void ejbPassivate() { }
        public void ejbLoad() { }
        public void ejbStore() { }
        public void ejbRemove() { }

        /* The following are the implementation of the method from the
           Message Remote interface
         */

        public Integer getMessageID() throws RemoteException { return messageID; }
        public void setMessageID(Integer messageID) throws RemoteException
        { this.messageID = messageID; }

        public String getSubject() throws RemoteException { return subject; }
        public void setSubject(String subject) throws RemoteException
        { this.subject = subject; }

        public String getDescription() throws RemoteException { return description; }
        public void setDescription(String descrption) throws RemoteException
        { this.description = description; }

        public Integer getReplyToID() throws RemoteException { return replyToID; }
        public void setReplyToID(Integer replyToID) throws RemoteException
        { this.replyToID = replyToID; }

        public Integer getForumID() throws RemoteException { return forumID; }
        public void setForumID(Integer forumID) throws RemoteException
        { this.forumID = forumID; }

        public String getAttachmentFile() throws RemoteException { return attachmentFile; }
        public void setAttachmentFile(String attachmentFile) throws RemoteException
        { this.attachmentFile = attachmentFile; }

        public String getUserName() throws RemoteException { return userName; }
        public void setUserName(String userName) throws RemoteException
        { this.userName = userName; }

        public Date getMessageDate() throws RemoteException { return messageDate; }
        public void setMessageDate(Date messageDate) throws RemoteException
        { this.messageDate = messageDate; }

 }
```

Message.java

```
/*
 * Message.java
 *
 * Defines the remote interface for the message bean
 *
 * @author: Devraj Mukherjee
 *
 * Created: 23rd June 2001
 *
 */

 package freeforums.message;

 import java.rmi.RemoteException;
 import java.util.Date;
 import javax.ejb.*;

 public interface Message extends EJBObject {

        public Integer getMessageID() throws RemoteException;
        public void setMessageID(Integer messageID) throws RemoteException;

        public String getSubject() throws RemoteException;
        public void setSubject(String subject) throws RemoteException;
```



```
        public String getDescription() throws RemoteException;
        public void setDescription(String descrption) throws RemoteException;

        public Integer getReplyToID() throws RemoteException;
        public void setReplyToID(Integer replyToID) throws RemoteException;

        public Integer getForumID() throws RemoteException;
        public void setForumID(Integer forumID) throws RemoteException;

        public String getAttachmentFile() throws RemoteException;
        public void setAttachmentFile(String attachmentFile) throws RemoteException;

        public String getUserName() throws RemoteException;
        public void setUserName(String userName) throws RemoteException;

        public Date getMessageDate() throws RemoteException;
        public void setMessageDate(Date messageDate) throws RemoteException;

}
```

UserController.java

```
/****
 * UserController.java
 *
 * JavaBean to allow the JSP pages to perform the right
 * user functions.
 *
 * @author: Devraj Mukherjee
 *
 ****/

package freeforums.jspbeans;

import javax.naming.*;
import java.util.Hashtable;
import javax.rmi.PortableRemoteObject;
import java.util.Properties;
import java.io.FileInputStream;
import java.util.Properties;

import freeforums.user.UserHome;
import freeforums.user.User;
import freeforums.user.UserManagerHome;
import freeforums.user.UserManager;

import java.util.List;

import org.jdom.*;

public class UserController extends Object {

        /*
         * Default Constructor does not have to initialize any instance vars
         */

        public void UserController() {

                // Blank Constructor do nothing

        }

        public boolean isSuperUser(Document UserDocument) {

                 Element userListElement = UserDocument.getRootElement();
                 List userElementList = userListElement.getChildren();
                 Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

                 String SuperUser = userElements[0].getChild("SuperUser").getText();

                 if(SuperUser.equals("Yes")) return true; // Is a Superuser
                 return false; // Not a superuser

        }
```

```
	public Document authenticateSession(Document userDocument) {

		/*****
		 *
		 * This method will authenticate the session passed on as a userDocument
		 * this method can be modified for tighter secruity be verifying the user
		 * using the Session Beans
		 *
		 *****/

		 Element userListElement = userDocument.getRootElement();
		 List userElementList = userListElement.getChildren();
		 Element [] userElements = (Element[])userElementList.toArray(new Element[0]);

		 String userName = userElements[0].getChild("UserName").getText();
		 String password = userElements[0].getChild("Password").getText();

		 Element responseElement = new Element("Response");

		 if(userName.equals("") || password.equals(""))
		 responseElement.addContent("Not Verified");
		 else responseElement.addContent("Verified");

		 return (new Document(responseElement));

	}

	public Document getUserList() {

		try	{
			//Get the home interface for the user management bean
			UserManagerHome home = (UserManagerHome) new
			InitialContext().lookup("UserManagerBean");
			// Create the manager object
			UserManager manager = home.create();
			return manager.findAll();
		}
		catch(Exception e) {
			// do nothing much
			return null;
		}

	}

	public Document addUser(Document xmlUserList) {

		try	{
			//Get the home interface for the user management bean
			UserManagerHome home = (UserManagerHome) new
			InitialContext().lookup("UserManagerBean");
			// Create the manager object
			UserManager manager = home.create();
			return manager.addUser(xmlUserList);
		}
		catch(Exception e) {
			// do nothing much
			return (new Document(new Element("Response").addContent("Failed")));
		}

	}

	public Document loginUser(Document loginDocument) {

		try	{
			//Get the home interface for the user management bean
			UserManagerHome home = (UserManagerHome) new
			InitialContext().lookup("UserManagerBean");
			// Create the manager object
			UserManager manager = home.create();
			return manager.loginUser(loginDocument);
		}
		catch(Exception e) {
			// do nothing much
			Element responseElement = new Element("Response");
			responseElement.addContent("Failed");
			return (new Document(responseElement));
```

```
                }

        }

        public Document getUserDetails(Document loginDocument) {

                Element UserListElement = loginDocument.getRootElement();
                List UserElementList = UserListElement.getChildren();
                Element [] UserElements = (Element[])UserElementList.toArray(new Element[0]);

                String UserName = UserElements[0].getChild("UserName").getText();

                try     {
                        //Get the home interface for the user management bean
                        UserManagerHome home = (UserManagerHome) new
                        InitialContext().lookup("UserManagerBean");
                        // Create the manager object
                        UserManager manager = home.create();
                        return manager.getUser(new Document(new
                        Element("UserName").addContent(UserName)));
                }
                catch(Exception e) {
                        // do nothing much
                        Element responseElement = new Element("Response");
                        responseElement.addContent("Failed");
                        return (new Document(responseElement));
                }
        }

        // Bridge method to

        public Document modifyUserDetails(Document UserDocument) {

                try     {
                        //Get the home interface for the user management bean
                        UserManagerHome home = (UserManagerHome) new
                        InitialContext().lookup("UserManagerBean");
                        // Create the manager object
                        UserManager manager = home.create();
                        return manager.modifyUser(UserDocument);
                }
                catch(Exception e) {
                        // do nothing much
                        Element responseElement = new Element("Response");
                        responseElement.addContent("Failed");
                        return (new Document(responseElement));
                }

        }

        // Bridge method to addUserToForum

        public Document addUserToForum(Document UserDocument) {

                try     {
                        //Get the home interface for the user management bean
                        UserManagerHome home = (UserManagerHome) new
                        InitialContext().lookup("UserManagerBean");
                        // Create the manager object
                        UserManager manager = home.create();
                        return manager.addUserToForum(UserDocument);
                }
                catch(Exception e) {
                        // do nothing much
                        Element responseElement = new Element("Response");
                        responseElement.addContent("Failed");
                        return (new Document(responseElement));
                }

        }

} // end class file here
```

MessageController.java

```
/*
 * MessageController.java
 *
 * JavaBean to allow the JSP pages to perform the right
 * user functions.
 *
 * @author: Devraj Mukherjee
 *
 * Includes XML Support
 *
 */

package freeforums.jspbeans;

import javax.naming.*;
import java.util.Hashtable;
import javax.rmi.PortableRemoteObject;
import java.util.Properties;
import java.io.FileInputStream;
import java.util.Properties;
import java.util.Date;

import java.util.List;

import freeforums.message.*;
import freeforums.user.*;

import org.jdom.*;

public class MessageController extends Object {

        public MessageController() { }

        /** Bridge method to addMessage **/

        public Document addMessageToForum(Document ForumDetails) {

                try     {
                        //Get the home interface for the user management bean
                        MessageManagerHome home = (MessageManagerHome) new
                        InitialContext().lookup("MessageManagerBean");
                        // Create the manager object
                        MessageManager manager = home.create();
                        return manager.addMessage(ForumDetails);
                }
                catch(Exception e) {
                        // do nothing much
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        }

        public Document markReadMessage(Document MessageDetails) {

                try     {
                        //Get the home interface for the user management bean
                        MessageManagerHome home = (MessageManagerHome) new
                        InitialContext().lookup("MessageManagerBean");
                        // Create the manager object
                        MessageManager manager = home.create();
                        return manager.markReadMessage(MessageDetails);
                }
                catch(Exception e) {
                        // do nothing much
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        }

        public Document isReadMessage(Document MessageDetails) {

                try     {
                        //Get the home interface for the user management bean
```

```
                        MessageManagerHome home = (MessageManagerHome) new
                        InitialContext().lookup("MessageManagerBean");
                        // Create the manager object
                        MessageManager manager = home.create();
                        return manager.isReadMessage(MessageDetails);
                }
                catch(Exception e) {
                        // do nothing much
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        }

        public Document getMessage(Document MessageDetails) {

                try     {
                        //Get the home interface for the user management bean
                        MessageManagerHome home = (MessageManagerHome) new
                        InitialContext().lookup("MessageManagerBean");
                        // Create the manager object
                        MessageManager manager = home.create();
                        return manager.getMessage(MessageDetails);
                }
                catch(Exception e) {
                        // do nothing much
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        }

////////////////////////////////////////////////////////////////////////////////////////////
//
// The following part of this program is the complex code that produces the threaded list of
// messages. The code has been adopted from the previous versions of the same project.
//
//
////////////////////////////////////////////////////////////////////////////////////////////

        private int depth;

        /**
         *
         * Given a Parent Message ID this will count the number of submessages
         *
         **/

        private int countSubMessages(Element MessageElement, Document MessageList) {

                String ParentMessageID = MessageElement.getChild("MessageID").getText();

                List MessageListElement = MessageList.getRootElement().getChildren();
                Element [] MessageElements = (Element[])MessageListElement.toArray(new
                Element[0]);

                try {
                        int countMessages=0;
                        for(int cntr=0; cntr<MessageElements.length; cntr++)
                        if(MessageElements[cntr].getChild("ReplyToID").getText().equals
                        (ParentMessageID)) countMessages++;
                        return countMessages;
                }
                catch(Exception e) {
                        System.out.println("countSubMessage() failed\n" +e);
                        return 0;
                }

        }

/**
 *
 * Given a Message Element this method will format it as HTML and return the Element
 *
 * This is done so that no web specific output is generated in the Entity or Session Beans
 *
 * This Class is a web specific class so we can generate some HTML here.
 *
 **/
```



```
 private Element formatMessage(Element MessageElement, String UserName) {

        Element RowElement = new Element("tr");

        String MessageID   = MessageElement.getChild("MessageID").getText();
        String Subject     = MessageElement.getChild("Subject").getText();
        String Author      = MessageElement.getChild("UserName").getText();
        String MessageDate = MessageElement.getChild("MessageDate").getText();
        String ForumID     = MessageElement.getChild("ForumID").getText();

        // Check if message is read

        Element ReadMessageStatus = new Element("ReadMessage");
        ReadMessageStatus.addContent(new Element("UserName").addContent(UserName));
        ReadMessageStatus.addContent(new Element("MessageID").addContent(MessageID));

        Document ReadStatusResult = isReadMessage(new Document(ReadMessageStatus));

        // Make HTML tags

        RowElement.addContent(new Element("td").addContent(MessageDate));

        Element ReadMessage = new Element("A");
        ReadMessage.addAttribute("HREF","read_message.jsp?message_id=" + MessageID +
        "&forum_id=" + ForumID);
        ReadMessage.addContent(Subject);

        Element ImageTag = new Element("IMG");
        ImageTag.addAttribute("SRC","../images/spacer.gif");
        ImageTag.addAttribute("WIDTH",""+depth);
        ImageTag.addAttribute("HEIGHT","1");

        Element SubjectCol = new Element("td");
        SubjectCol.addContent(ImageTag);

        if(ReadStatusResult.getRootElement().getText().equals("UnRead")) {

               Element Bold = new Element("B");
               Bold.addContent(ReadMessage);
               SubjectCol.addContent(Bold);

        }
        else SubjectCol.addContent(ReadMessage);

        RowElement.addContent(SubjectCol);

        RowElement.addContent(new Element("td").addContent(Author));

        Element PostMessage = new Element("A");
        PostMessage.addAttribute("HREF","post_message.jsp?reply_to=" + MessageID +
        "&forum_id=" + ForumID);
        PostMessage.addContent("Reply");

        RowElement.addContent(new Element("td").addContent(PostMessage));

        return RowElement;

 }

/**
 *
 * Parent method to generate the HTML
 *
 **/

public Document getMessagesForForum(Document ForumUserDetails) {

       Element MessageTableElement = new Element("table");
       MessageTableElement.addAttribute("width","100%");
       MessageTableElement.addAttribute("cellpadding","0");
       MessageTableElement.addAttribute("cellspacing","0");

       Element HeaderRow = new Element("tr");
       HeaderRow.addAttribute("bgcolor","#FFFFCC");

       Element Column1 = new Element("td");
```



```
            Column1.addAttribute("width","10%");
            Column1.addContent("Date");
            HeaderRow.addContent(Column1);

            Element Column2 = new Element("td");
            Column2.addContent("Subject");
            HeaderRow.addContent(Column2);

            Element Column3 = new Element("td");
            Column3.addAttribute("width","10%");
            Column3.addContent("Author");
            HeaderRow.addContent(Column3);

            Element Column4 = new Element("td");
            Column4.addAttribute("width","10%");
            Column4.addContent("Reply");
            HeaderRow.addContent(Column4);

            MessageTableElement.addContent(HeaderRow);

            String ForumID  =
            ForumUserDetails.getRootElement().getChild("ForumID").getText();
            String UserName =
            ForumUserDetails.getRootElement().getChild("UserName").getText();

            try {
                    //Get the home interface for the user management bean
                    MessageManagerHome home = (MessageManagerHome) new
                    InitialContext().lookup("MessageManagerBean");
                    // Create the manager object
                    MessageManager manager = home.create();
                    //Get the array
                    Document MessageListDocument = manager.findMessagesForForum(new
                    Document(new Element("ForumID").addContent(ForumID)));

                    List MessageListElement =
                    MessageListDocument.getRootElement().getChildren();
                    Element [] MessageElements = (Element[])MessageListElement.toArray(new
                    Element[0]);

                    for(int cntr=0; cntr<MessageElements.length; cntr++) {
                            Element currentMessage = MessageElements[cntr];
                            if(currentMessage.getChild("ReplyToID").getText().equals("-1"))
                            {
                                    depth=0;

                                    MessageTableElement.addContent
                                    (formatMessage(currentMessage, UserName));
                                    listSubMessages(MessageTableElement,MessageListDocument,
                                    currentMessage.getChild("MessageID").getText(),
                                    UserName);
                            }
                    }

                    return (new Document(MessageTableElement));
            }
            catch(Exception e) {
                    //something went wrong while making the HTML
                    //quite unlikely that we would get this exception
                    System.out.println("getMessageForForum failed \n" + e);
                    return (new Document(new Element("Response").addContent("Failed")));
            }

    } // end method

    /**
     *
     * Converts the messages to HTML format
     * This method is a helper method
     *
     **/

    private void listSubMessages(Element MessageTableElement, Document
    MessageListDocument, String ParentMessageID, String UserName) {

            List MessageListElement = MessageListDocument.getRootElement().getChildren();
            Element [] MessageElements = (Element[])MessageListElement.toArray(new
```

```
                Element[0]);

                try {
                        Element currentMessage=null;
                        int countMessages=0,messageIndex=0;
                        depth+=8;

                        for(int cntr=0; cntr<MessageElements.length; cntr++) {
                                currentMessage = MessageElements[cntr];
                                if(currentMessage.getChild("ReplyToID").
                                getText().equals(ParentMessageID)) {
                                        countMessages++;
                                        messageIndex=cntr;
                                }
                        }

                        if (countMessages==0) return;
                        if (countMessages==1) {
                                currentMessage=MessageElements[messageIndex];

                                MessageTableElement.addContent(formatMessage
                                (currentMessage,UserName));

                                if(countSubMessages(currentMessage, MessageListDocument)>0)
                                listSubMessages(MessageTableElement, MessageListDocument,
                                currentMessage.getChild("MessageID").getText(), UserName);
                                depth-=8;
                                return;
                        }
                        ////////////////////////////////
                        // If more than one message then //
                        ////////////////////////////////
                        for(int cntr=0; cntr<MessageElements.length; cntr++) {
                                currentMessage=MessageElements[cntr];

                                if(currentMessage.getChild("ReplyToID").getText().equals(ParentM
                                essageID)) {

                                MessageTableElement.addContent(formatMessage(currentMessage,User
                                Name));

                                // List sub messages for this reply message
                                if(countSubMessages(currentMessage, MessageListDocument)>0)
                                listSubMessages(MessageTableElement, MessageListDocument,
                                currentMessage.getChild("MessageID").getText(), UserName);
                                }
                        }
                        ////////////////////////////////
                        // End more than one message //
                        ////////////////////////////////
                        depth-=8;
                        return;
                }
                catch(Exception e) {
                        //Something wrong happened above, quite unlikely.
                        System.out.println("listSubMessages() failed \n " + e);
                        return;
                }
        }

} // end class file
```

ForumController.java

```
/*
 * ForumController.java
 *
 * JavaBean to allow the JSP pages to perform the right
 * user functions.
 *
 * @author: Devraj Mukherjee
 *
 */
```



```
package freeforums.jspbeans;

import javax.naming.*;
import java.util.Hashtable;
import javax.rmi.PortableRemoteObject;
import java.util.Properties;
import java.io.FileInputStream;
import java.util.Properties;

import freeforums.forum.ForumManager;
import freeforums.forum.ForumManagerHome;
import freeforums.forum.Forum;
import freeforums.forum.ForumHome;

import org.jdom.*;


public class ForumController extends Object {

        /**
         Empty Constructor expected to do nothing
         **/

        public ForumController() { }

        /**
         Forum Listing for User
         **/

        public Document getForumListingForUser(Document UserDetails) {

                try     {
                        //Get the home interface for the user management bean
                        ForumManagerHome home = (ForumManagerHome) new
                        InitialContext().lookup("ForumManagerBean");
                        // Create the manager object
                        ForumManager manager = home.create();
                        return manager.findForumsForUser(UserDetails);

                }
                catch(Exception e) {
                        // do nothing much
                        System.err.println("Exception occured \n" + e);
                        return (new Document(new Element("Response").addContent("Failed")));
                }

         } // End Method getForumListingForUser

        /**
         Complete Forum Listing
         **/

         public Document getCompleteForumListing() {

                try     {
                        //Get the home interface for the user management bean
                        ForumManagerHome home = (ForumManagerHome) new
                        InitialContext().lookup("ForumManagerBean");
                        // Create the manager object
                        ForumManager manager = home.create();
                        return manager.findAll();

                }
                catch(Exception e) {
                        // do nothing much
                        System.err.println("Exception occured \n" + e);
                        return (new Document(new Element("Response").addContent("Failed")));
                }

         } // End Method getCompleteForumListing

         public Document addForum(Document ForumInformation) {

                try     {
                        //Get the home interface for the user management bean
                        ForumManagerHome home = (ForumManagerHome) new
```

```
                        InitialContext().lookup("ForumManagerBean");
                        // Create the manager object
                        ForumManager manager = home.create();
                        return manager.addForum(ForumInformation);

                }
                catch(Exception e) {
                        // do nothing much
                        System.err.println("Exception occured \n" + e);
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        } // End Method Add Forum

        public Document getForumListForUser(Document ForumInformation) {

                try     {
                        //Get the home interface for the user management bean
                        ForumManagerHome home = (ForumManagerHome) new
                        InitialContext().lookup("ForumManagerBean");
                        // Create the manager object
                        ForumManager manager = home.create();
                        return manager.findForumsForUser(ForumInformation);

                }
                catch(Exception e) {
                        // do nothing much
                        System.err.println("Exception occured \n" + e);
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        } // End Method getForumListForUser

        public Document isUserAllowed(Document ForumUserInfo) {

                try     {
                        //Get the home interface for the user management bean
                        ForumManagerHome home = (ForumManagerHome) new
                        InitialContext().lookup("ForumManagerBean");
                        // Create the manager object
                        ForumManager manager = home.create();
                        return manager.isUserAllowed(ForumUserInfo);

                }
                catch(Exception e) {
                        // do nothing much
                        System.err.println("Exception occured \n" + e);
                        return (new Document(new Element("Response").addContent("Failed")));
                }

        } // End Method

} // End Class file
```

Appendix F

Source Code (Java Server Pages, CSS, XSL, JavaScript)

Author: Devraj Mukherjee

Freeforums.css

```
P { font: 10pt Verdana, sans-serif  }
H1 { font: bold 11pt Verdana, sans-serif }
H2 { font: bold 10pt Verdana, sans-serif }
H5 { font: 8pt Verdana, sans-serif }
TD { font: 10pt Verdana, sans-serif }
```

Controls.js

```
function validateForm(the_form)
{
 for (var counter= 0; counter< the_form.elements.length; counter++)
 {
  if (the_form.elements[counter].value=="") {
   alert("Please fill all the fields");
   return false;
  }
 }
 return true;
}



function loadframe(page_name,page_title,width,height)
{
  cwin=open("","contact","toolbar=no","location=no","status=no","menubar=no")
  cwin.resizeTo(width,height)
  cwin.moveTo(170,50)
  cwin.document.open("text/html")
}
```

user_list.jsp

```
<%@taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="xsl" %>
<%@ page import="freeforums.jspbeans.UserController,org.jdom.*,org.jdom.output.XMLOutputter"
%>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/

%>
```



```
<html>

<body bgcolor="#CFD9E3">

<font face="Verdana" size=2 color="#000000">

<b>Forum User Manager</b><br><br>

<xsl:apply xsl="/xsl/userlist.xsl">

<%

 Document XMLDocument = (new UserController()).getUserList();
 XMLOutputter output = new XMLOutputter();

 try {
       output.output(XMLDocument, out);
 }
 catch (Exception e) {
       System.err.println(e);
 }

%>

</xsl:apply>

<br><br>

[<a href="add_forum_user.html">Add a new User</a> | <a href="add_user_forum.jsp">Add Users to
Forums</a>]

<br><br>

</font>
</body>
</html>
```

user_forum_list.jsp

```
<%@taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="xsl" %>
<%@ page import="freeforums.jspbeans.ForumController,org.jdom.output.XMLOutputter" %>
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*,java.util.List" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
 controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
 response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
  *****/

 Document UserDetails =
controller.getUserDetails((Document)session.getAttribute("loginDetails"));

 List UserListElement = UserDetails.getRootElement().getChildren();
```



```
 Element [] UserElements = (Element[])UserListElement.toArray(new Element[0]);


%>

<html>
<head>
  <title>FreeForums - Add a new user</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
</head>

<body bgcolor="#E0EAF4" text="#000000">

<p><b>Welcome <%= UserElements[0].getChild("RealName").getText() %></b></p>

<p><b>Below is a list for Forum you are subscribed to, click on one to read the
messages</b></p>

<xsl:apply xsl="/xsl/userforumlist.xsl">

<%

 Document XMLDocument = (new
ForumController()).getForumListingForUser((Document)session.getAttribute("loginDetails"));
 XMLOutputter output = new XMLOutputter();

 try {
       output.output(XMLDocument, out);
 }
 catch (Exception e) {
       System.err.println(e);
 }

%>

</xsl:apply>

<br>

</body>
</html>
```

show_messages.jsp

```
<%@ page import="freeforums.jspbeans.UserController, freeforums.jspbeans.MessageController,
freeforums.jspbeans.ForumController, org.jdom.*, java.util.List, org.jdom.output.XMLOutputter"
%>

<%

 String ForumID = request.getParameter("forum_id");

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 // Get user name from document

 Document UserDetailDocument =
controller.getUserDetails((Document)session.getAttribute("loginDetails"));
```



```
 List userElementList = UserDetailDocument.getRootElement().getChildren();
 Element [] UserElements = (Element[])userElementList.toArray(new Element[0]);

 Element ForumUser = new Element("ForumUser");
 ForumUser.addContent(new
Element("UserName").addContent(UserElements[0].getChild("UserName").getText()));
 ForumUser.addContent(new Element("ForumID").addContent(""+ForumID));

 // If the user is not allowed to use this forum send them back

 Document UserAllowedToForum = (new ForumController()).isUserAllowed(new Document(ForumUser));

 if(UserAllowedToForum.getRootElement().getText().equals("Failed"))
response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/

  MessageController messageBank = new MessageController();

  Element ForumMessages = new Element("ForumMessages");
  ForumMessages.addContent(new Element("ForumID").addContent(ForumID));
  ForumMessages.addContent(new
Element("UserName").addContent(UserElements[0].getChild("UserName").getText()));

  Document MessageList = messageBank.getMessagesForForum(new Document(ForumMessages));

%>

<html>
<head>
  <title>FreeForums - Add a new user</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
</head>

<body bgcolor="#CFD4D9" text="#000000">
<font face="Verdana" size=2>

<b>Message List</b><br><br>


<%

 XMLOutputter output = new XMLOutputter();

 try {

  output.output(MessageList,out);

 }
 catch(Exception e) {

 }

%>

<br><br>

[<a href="post_message.jsp?reply_to=-1&forum_id=<%= ForumID %>">Post Message</a>]<br><br>

</font>
</body>
</html>
```

setup_freeforums.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*, java.util.List,
org.jdom.output.XMLOutputter" %>
```



```
<%

 UserController controller = new UserController();

 String username  = "admin";
 String password  = "freeforums";
 String realname  = "FreeForums Default Admin Profile";
 String email     = "you@yourdomain.com";
 String superuser = "Yes";

 Element UserElement = new Element("User");
 UserElement.addContent(new Element("UserName").addContent(username));
 UserElement.addContent(new Element("Password").addContent(password));
 UserElement.addContent(new Element("RealName").addContent(realname));
 UserElement.addContent(new Element("EmailAddress").addContent(email));
 UserElement.addContent(new Element("SuperUser").addContent(superuser));

 Document AdditionDocument = new Document(new Element("UserList").addContent(UserElement));

 Document ResultDocument = controller.addUser(AdditionDocument);

 String Result = ResultDocument.getRootElement().getText();

 if(Result.equals("Done")) {
     session.setAttribute("message","Default Admin User Added");
 }
 else if(Result.equals("Duplicate")) {
  session.setAttribute("message","Changes could not be made");
 }
 else {
  session.setAttribute("message","Changes could not be made");
 }

 session.setAttribute("redirect_url","../index.html");
 response.sendRedirect("message_box.jsp");

%>
```

seperator.html

```
<html>
<body bgcolor="#000000">
<br>
</body>
</html>
```

save_user.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*, java.util.List,
org.jdom.output.XMLOutputter" %>


<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
 controller.authenticateSession((Document)session.getAttribute("loginDetails"));
```

```
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/

 String username  = request.getParameter("username");
 String password  = request.getParameter("password");
 String realname  = request.getParameter("realname");
 String email     = request.getParameter("email");
 String superuser = request.getParameter("superuser");

 Element UserElement = new Element("User");
 UserElement.addContent(new Element("UserName").addContent(username));
 UserElement.addContent(new Element("Password").addContent(password));
 UserElement.addContent(new Element("RealName").addContent(realname));
 UserElement.addContent(new Element("EmailAddress").addContent(email));
 UserElement.addContent(new Element("SuperUser").addContent(superuser));

 Document AdditionDocument = new Document(new Element("UserList").addContent(UserElement));

 Document ResultDocument = controller.addUser(AdditionDocument);

 String Result = ResultDocument.getRootElement().getText();

 if(Result.equals("Done")) {
     session.setAttribute("message","User Added");
 }
 else if(Result.equals("Duplicate")) {
  session.setAttribute("message","Changes could not be made");
 }
 else {
  session.setAttribute("message","Changes could not be made");
 }

 session.setAttribute("redirect_url","add_forum_user.html");
 response.sendRedirect("message_box.jsp");

%>
```

save_personal_details.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*, java.util.List,
org.jdom.output.XMLOutputter" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
```

```
  *****/

 String Password     = request.getParameter("password");
 String RePassword   = request.getParameter("reenter_password");
 String EmailAddress = request.getParameter("email");

 if(!Password.equals(RePassword)) {

  session.setAttribute("message","Passwords did not match");
  session.setAttribute("redirect_url","modify_personal_details.jsp");
  response.sendRedirect("message_box.jsp");

 }

 Document UserDetailDocument =
controller.getUserDetails((Document)session.getAttribute("loginDetails"));

 List userElementList = UserDetailDocument.getRootElement().getChildren();
 Element [] UserElements = (Element[])userElementList.toArray(new Element[0]);

 String UserName  = UserElements[0].getChild("UserName").getText();
 String SuperUser = UserElements[0].getChild("SuperUser").getText();
 String RealName  = UserElements[0].getChild("RealName").getText();

 Element UserElement = new Element("User");
 UserElement.addContent(new Element("UserName").addContent(UserName));
 UserElement.addContent(new Element("Password").addContent(Password));
 UserElement.addContent(new Element("RealName").addContent(RealName));
 UserElement.addContent(new Element("EmailAddress").addContent(EmailAddress));
 UserElement.addContent(new Element("SuperUser").addContent(SuperUser));

 Document ModificationDocument = new Document(new
Element("UserList").addContent(UserElement));

/*
 XMLOutputter output = new XMLOutputter();
 output.output(ModificationDocument, out);
*/

 Document ResultDocument = controller.modifyUserDetails(ModificationDocument);

 String Result = ResultDocument.getRootElement().getText();

 if(Result.equals("Done")) {
     session.setAttribute("message","Modifcations Made");
     session.setAttribute("redirect_url","modify_personal_details.jsp");
     response.sendRedirect("message_box.jsp");
  }
  else {
   session.setAttribute("message","Changes could not be made");
   session.setAttribute("redirect_url","modify_personal_details.jsp");
   response.sendRedirect("message_box.jsp");
  }

%>
```

save_message.jsp

```
<%@ page import="freeforums.jspbeans.MessageController, freeforums.jspbeans.UserController,
org.jdom.*, java.util.List, org.jdom.output.XMLOutputter" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/
```

```
 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
  *****/

 Document UserDetails = (Document)session.getAttribute("loginDetails");

 List UserListElement = UserDetails.getRootElement().getChildren();
 Element [] UserElements = (Element[])UserListElement.toArray(new Element[0]);

 String subject   = request.getParameter("subject");
 String message   = request.getParameter("message");
 String ReplyToID = request.getParameter("ReplyToID");
 String ForumID   = request.getParameter("ForumID");
 String UserName  = UserElements[0].getChild("UserName").getText();

 Element Message = new Element("Message");
 Message.addContent(new Element("Subject").addContent(subject));
 Message.addContent(new Element("MessageBody").addContent(message));
 Message.addContent(new Element("UserName").addContent(UserName));
 Message.addContent(new Element("ForumID").addContent(ForumID));
 Message.addContent(new Element("ReplyToID").addContent(ReplyToID));

 Element MessageList = new Element("MessageList");
 MessageList.addContent(Message);

 Document RequestDocument = new Document(MessageList);

 MessageController messageBank = new MessageController();

 Document ResponseDocument = messageBank.addMessageToForum(RequestDocument);

 String Answer = ResponseDocument.getRootElement().getText();

 if(Answer.equals("Done")) {

  session.setAttribute("message","Message added to Forum");
  session.setAttribute("redirect_url","show_messages.jsp?forum_id=" + ForumID);

 }

 else {

  session.setAttribute("message","Failed to add message");
  session.setAttribute("redirect_url","post_message.jsp?forum_id=" + ForumID + "&reply_to=" +
ReplyToID);

 }

 response.sendRedirect("message_box.jsp");

%>
```

save_forum.jsp

```
<%@ page import="freeforums.jspbeans.ForumController, freeforums.jspbeans.UserController,
org.jdom.*" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
```

```
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/

 String ForumName   = request.getParameter("forumname");
 String Description = request.getParameter("description");

 Element ForumElement = new Element("Forum");
 ForumElement.addContent(new Element("ForumName").addContent(ForumName));
 ForumElement.addContent(new Element("ForumDescription").addContent(Description));

 Element ForumListElement = new Element("ForumList");
 ForumListElement.addContent(ForumElement);

 Document ForumInformation = new Document(ForumListElement);

 ForumController forumController = new ForumController();

 Document responseDocument = forumController.addForum(ForumInformation);

 String SessionBeanResponse = responseDocument.getRootElement().getText();

 if(SessionBeanResponse.equals("Done")) session.setAttribute("message","New Forum Added");
 else session.setAttribute("message","Forum could not be added");

 session.setAttribute("redirect_url","add_forum.jsp");
 response.sendRedirect("message_box.jsp");


%>
```

read_message.jsp

```
<%@ page import="freeforums.jspbeans.ForumController, freeforums.jspbeans.UserController,
freeforums.jspbeans.MessageController, org.jdom.*, java.util.List,
org.jdom.output.XMLOutputter" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
```

```
  *
  *****/

 // Get user details

 Document CurrentUserDetails = (Document)session.getAttribute("loginDetails");

 List UserListElement = CurrentUserDetails.getRootElement().getChildren();
 Element [] UserElements = (Element[])UserListElement.toArray(new Element[0]);


 // Reterieve the message details

 String message_id = request.getParameter("message_id");

 Document RequestDocument = new Document((new Element("MessageID")).addContent(message_id));

 MessageController messageBank = new MessageController();

 Document ResultDocument = messageBank.getMessage(RequestDocument);

 List MessageListElement = ResultDocument.getRootElement().getChildren();
 Element [] MessageElements = (Element[])MessageListElement.toArray(new Element[0]);

 String UserName = MessageElements[0].getChild("UserName").getText();
 String Subject  = MessageElements[0].getChild("Subject").getText();
 String MessageDate = MessageElements[0].getChild("MessageDate").getText();
 String MessageBody = MessageElements[0].getChild("MessageBody").getText();
 String ForumID     = MessageElements[0].getChild("ForumID").getText();

 // See if the user is allowed in this forum

 Element ForumUser = new Element("ForumUser");
 ForumUser.addContent(new
Element("UserName").addContent(UserElements[0].getChild("UserName").getText()));
 ForumUser.addContent(new Element("ForumID").addContent(""+ForumID));

 // If the user is not allowed to use this forum send them back

 Document UserAllowedToForum = (new ForumController()).isUserAllowed(new Document(ForumUser));
 if(UserAllowedToForum.getRootElement().getText().equals("Failed"))
response.sendRedirect("../index.html");

 // Mark this message read

 Element ReadMessage = new Element("ReadMessage");

 ReadMessage.addContent(new
Element("MessageID").addContent(MessageElements[0].getChild("MessageID").getText()));
 ReadMessage.addContent(new
Element("UserName").addContent(UserElements[0].getChild("UserName").getText()));

 Document ReadMessageResult = messageBank.markReadMessage(new Document(ReadMessage));


%>


<html>
<head>
  <title>FreeForums - Add a new user</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
</head>

<body bgcolor="#CFD4D9" text="#000000">
<font face="Verdana" size=2>

<table border=0 cellpadding=4 cellspacing=4 width="100%">
<tr bgcolor="#000000"><td>
<font face="Verdana" size=2 color="#FFFFFF">
<b><%= Subject %></b>
</font>
</td></tr>
</table>

<br>
```

```
Posted By <%= UserName %> on <%= MessageDate %><br><br>

<table border=0 cellpadding=4 cellspacing=4 width="100%">
<tr bgcolor="#000000"><td>
<font face="Verdana" size=2 color="#FFFFFF">
Body
</font>
</td></tr>
</table>

<br>

<%= MessageBody %>


<br><br><br>

[<a href="show_messages.jsp?forum_id=<%= request.getParameter("forum_id") %>">Message
List</a>]<br><br>

</font>

<font face="Verdana" size=1>

<%

 if(ReadMessageResult.getRootElement().getText().equals("Done"))
 out.println("Marked Read For " + UserElements[0].getChild("UserName").getText());
 else
 out.println("Could not set flag for " + UserElements[0].getChild("UserName").getText());

%>


</font>

</body>

</html>
```

post_message.jsp

```
<%@ page import="freeforums.jspbeans.UserController, freeforums.jspbeans.MessageController,
org.jdom.*, java.util.List, org.jdom.output.XMLOutputter" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
  *****/

%>

<html>
```

```
<head>
  <title>FreeForums - Add a new user</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
</head>

<body bgcolor="#CFD4D9" text="#000000">

<form name="postmessage" action="save_message.jsp" method="POST">

<input type="hidden" name="ReplyToID" value="<%= request.getParameter("reply_to") %>">
<input type="hidden" name="ForumID" value="<%= request.getParameter("forum_id") %>">

<font face="Verdana" size=3>

<% if((new Integer(request.getParameter("reply_to"))).intValue() == -1) { %>
<b>Post a new Message</b>
<% } else { %>
<b>Reply to Message</b>
<% } %>

<br><br>
</font>

<table border=0 cellpadding=4 cellspacing=4>

<tr>
<td><font face="Verdana" size=2><b>Subject:</b></font></td>
<td><input type="text" size=40 name="subject"></td>
</tr>

<tr>
<td><font face="Verdana" size=2><b>Message:</b></font></td>
<td><textarea cols=40 rows=10 name="message"></textarea></td>
</tr>

</table>

<br><br>

<input type="submit" name="Submit" Value="Post Message">   
<input type="reset" name="Reset" Value="Clear">

</form>

</body>
</html>
```

options.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));

 /*****
```



```
  *
  * Verification code completed
  *
  *****/

%>


<html>

<body bgcolor="#CFD9E3">

<font face="Verdana" size=1 color="#000000"><b>

  <a href="http://www.devraj.org/projects/freeforums/" target="_top"><img
src="../images/freeforums_logo.jpg"  border="0"></a>
  <br><br><br>

  <a href="user_forum_list.jsp" target="main" STYLE="color:#000000;text-decoration:none;"
border="0">Forum List</a><br><br>

  <a href="modify_personal_details.jsp" target="main" STYLE="color:#000000;text-
decoration:none;" border="0">Modify Profile</a><br><br>

  <% if(SuperUser) { %>

  <a href="user_list.jsp" target="main" STYLE="color:#000000;text-decoration:none;"
border="0">Manage Users</a><br>
  <a href="forum_list.jsp" target="main" STYLE="color:#000000;text-decoration:none;"
border="0">Manage Forums</a><br><br>

  <% } %>

  <a href="logout.jsp" target="_top" STYLE="color:#000000;text-decoration:none;"
border="0">Logout</a>

</b></font>

</body>

</html>
```

modify_personal_details.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*, java.util.List,
org.jdom.output.XMLOutputter" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
  *****/
```



```
  Document UserDetailDocument =
controller.getUserDetails((Document)session.getAttribute("loginDetails"));

  List userElementList = UserDetailDocument.getRootElement().getChildren();
  Element [] UserElements = (Element[])userElementList.toArray(new Element[0]);

  String password     = UserElements[0].getChild("Password").getText();
  String emailAddress = UserElements[0].getChild("EmailAddress").getText();

%>

<html>
<head>
  <title>FreeForums - Add a new user</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
</head>
<body bgcolor="#CFD4D9" text="#000000">
<form name="changedetails" method="POST" action="save_personal_details.jsp">
<h1>Change your details</h1>

 <table border=0>
  <tr>
   <td><p>Email Address:</p></td>
   <td><input type="text=" name="email" size=30 value="<%= emailAddress %>"></td>
  </tr>
  <tr>
   <td><p>Password:</p></td>
   <td><input type="password" name="password" size=15 value="<%= password %>"></td>
  </tr>
  <tr>
   <td><p>Re-enter<br>Password:</p></td>
   <td><input type="password" name="reenter_password" size=15 value="<%= password %>"></td>
  </tr>
 </table><br><br>
 <input type="submit" value="Modify User">   
 <input type="reset" value="Clear"><br>
</form>
</body>
</html>
```

message_box.jsp

```
<%

 String message = (String)session.getAttribute("message");
 String redirecturl = (String)session.getAttribute("redirect_url");

 String task = (String)request.getParameter("task");
 if(task==null) task="showmessage";

 if(task.equals("redirect")) response.sendRedirect(redirecturl);

%>

<html>

<body bgcolor="#CFD4D9" link="#666699" vlink="#666699" alink="#CCCCFF">

<form name="login" action="message_box.jsp" method="POST">
<font face="Verdana" size=2 color="#000000">
<center>

<br><br>

<table border="0" cellspacing="0" cellpadding="1" bgcolor="#000000" align="center"
width="45%">
<tr><td>

 <table border="0" cellspacing="0" cellpadding="0" align="center" width="100%">
 <td><td bgcolor="#CFD4D9">

<center>
```

```
<br><p><font face="Verdana" size=2 color="#000000"><b><%= message %></b></font></p><br>

<form name="redirect" action="message_box.jsp" method="POST">

<input name="task" type="hidden" value="redirect">
<input type="submit" name="submit" value=" Ok ">
</form>
</center>


 </td></tr>
 </table>

</td></tr>
</table><br><br>

<font face="Verdana" size=2>

<br>&copy; 2001 The FreeForums Project<br>

</font>
</center>

</font>
</form>

</body>
</html>
```

logout.jsp

```
<%

  /**
   Logout.jsp

   This allows the user to reset all the information on the server
   about this session and then close the session.

   Once this script is executed the current session of the user
   will not be recognized by the system.

   **/

  session.setAttribute("username","");
  session.setAttribute("password","");
  session.setAttribute("login_status","false");

  response.sendRedirect("../index.html");

  /**
   It is expected that index.html will make use of JavaScript to
   load itself up in the parent frame
   **/

%>
```

login.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*" %>

<%


 String username=request.getParameter("username");
 String password=request.getParameter("password");

 // Create a new Document Type object for verifing the XML document

 DocType dtd = new DocType("UserList","http://devraj.org/dtd/freeforums/User.dtd");
```



```
 // The few lines here prepares the User Element for the object and adds the content for the
 // different elements.

 Element userElement = new Element("User");
 userElement.addContent(new Element("UserName").addContent(username));
 userElement.addContent(new Element("Password").addContent(password));

 Element userListElement = new Element("UserList");
 userListElement.addContent(userElement);

 Document userListDocument = new Document(userListElement);

 UserController loginmanager = new UserController();

 // Get the result of login, it will either be a user or a
 // Response document.

 Document result = loginmanager.loginUser(userListDocument);

 Element responseElement = result.getRootElement();
 String responseContent = responseElement.getText();

 if(responseContent.equals("Failed")) {
  session.setAttribute("message","Login Failed");
  session.setAttribute("redirect_url","../index.html");
  response.sendRedirect("message_box.jsp");
 }
 else {
  session.setAttribute("loginDetails",result); // store the xml doc for later reference.
  response.sendRedirect("freeforums_menu.jsp");
 }

%>
```

freeforums_menu.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*, org.jdom.output.XMLOutputter"
%>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
  *****/


%>

<html>
<head>
 <title>FreeForums - Web Gateway</title>
</head>

<frameset cols="120px,2px,*" border=0>
```

```
 <frame src="options.jsp" border=0 scrolling="no" name="toolbar">
 <frame src="seperator.html" border=0 scrolling="no" name="seperator">
 <frame src="user_forum_list.jsp" border=0 name="main">

</frameset>

</html>
```

forum_list.jsp

```
<%@taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="xsl" %>
<%@ page import="freeforums.jspbeans.ForumController,org.jdom.*,org.jdom.output.XMLOutputter,
freeforums.jspbeans.UserController" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/

%>

<html>

<body bgcolor="#CFD9E3">

<font face="Verdana" size=2 color="#000000">

<b>Forum Manager</b><br><br>

<xsl:apply xsl="/xsl/forumlist.xsl">

<%

 Document XMLDocument = (new ForumController()).getCompleteForumListing();
 XMLOutputter output = new XMLOutputter();

 try {
       output.output(XMLDocument, out);
 }
 catch (Exception e) {
       System.err.println(e);
 }

%>

</xsl:apply>

<br><br>

[<a href="add_forum.jsp">Add new Forums</a>]
```

```
<br><br>

</font>
</body>
</html>
```

add_user_to_forum.jsp

```
<%@ page import="freeforums.jspbeans.ForumController, freeforums.jspbeans.UserController,
org.jdom.*" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/


 String UserName = request.getParameter("username");
 Integer ForumID  = new Integer(request.getParameter("forumID"));
 String Administrator = request.getParameter("SuperUser");

 if(Administrator==null) Administrator = "No";
 if(!Administrator.equals("Yes")) Administrator = "No";

 Element RootElement = new Element("UserList");
 Element ForumUser = new Element("ForumUser");

 ForumUser.addContent(new Element("UserName").addContent(UserName));
 ForumUser.addContent(new Element("ForumID").addContent(ForumID.toString()));
 ForumUser.addContent(new Element("Administrator").addContent(Administrator));

 RootElement.addContent(ForumUser);

 Document ResultDocument = controller.addUserToForum(new Document(RootElement));

 String ResultString = ResultDocument.getRootElement().getText();

 if(ResultString.equals("Done")) session.setAttribute("message","User Added to Forum");
 else session.setAttribute("message","User Could not be Added");

 session.setAttribute("redirect_url","add_user_forum.jsp");
 response.sendRedirect("message_box.jsp");

%>
```


add_user_forum.jsp

```
<%@taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0" prefix="xsl" %>
<%@ page
import="freeforums.jspbeans.ForumController,org.jdom.*,org.jdom.output.XMLOutputter,freeforums
.jspbeans.UserController" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");


 /*****
  *
  * Verification code completed
  *
  *****/

%>

<html>

<body bgcolor="#CFD9E3">
<form name="add_to_forum" action="add_user_to_forum.jsp" method="POST">
<font face="Verdana" size=2 color="#000000">

<b>Add User to Forums</b><br><br>

Select a User and Forum to add make a user of the Forum.<br><br>

<table border=0 cellpadding=4 cellspacing=4>

<tr>
<td><font face="Verdana" size=2><b>Select Forum:</b></font></td>
<td>
<xsl:apply xsl="/xsl/forum_combo_box.xsl">

<%

 Document XMLDocument = (new ForumController()).getCompleteForumListing();
 XMLOutputter output = new XMLOutputter();

 try {
       output.output(XMLDocument, out);
 }
 catch (Exception e) {
       System.err.println(e);
 }

%>

</xsl:apply>
</td></tr>

<tr>
<td><font face="Verdana" size=2><b>Select UserName:</b></font></td>
<td>
<xsl:apply xsl="/xsl/user_combo_box.xsl">
```



```
<%

 Document XMLDocument = (new UserController()).getUserList();
 XMLOutputter output = new XMLOutputter();

 try {
       output.output(XMLDocument, out);
 }
 catch (Exception e) {
       System.err.println(e);
 }

%>

</xsl:apply>
</td></tr>

</table><br>

<input type="checkbox" value="Yes" name="SuperUser"> <b>User is Administrator to this
Forum</b>

<br><br><br>

<input type="submit" value="Add to Forum">

</form>
</font>
</body>
</html>
```

add_forum_user.html

```
<html>
<head>
  <title>FreeForums - Add a new user</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
</head>
<body bgcolor="#E0EAF4" text="#000000">
<form name="adduser" method="POST" action="save_user.jsp">
 <h1>Add a new user</h1>
 <table border=0>
  <tr>
   <td><p>Username:</p></td>
   <td><input type="text=" name="username" size=15></td>
  </tr>
  <tr>
   <td><p>Password:</p></td>
   <td><input type="text" name="password" size=15></td>
  </tr>
  <tr>
   <td><p>Real Name:</p></td>
   <td><input type="text" name="realname" size=30></td>
  </tr>
  <tr>
   <td><p>Email:</p></td>
   <td><input type="text" name="email" size=30></td>
  </tr>
  <tr>
   <td><p>Account permissions:</p></td>
   <td>
    <select name="superuser">
     <option value="Yes">Superuser</option>
     <option value="No">Normal User</option>
    </select>
   </td>
  </tr>
 </table><br><br>
 <input type="submit" value="Add User">   
 <input type="reset" value="Clear"><br>
</form>
</body>
```



```
</html>
```

add_forum.jsp

```
<%@ page import="freeforums.jspbeans.UserController, org.jdom.*" %>

<%

 /*****
  *
  * This block of code determines the security for these web components
  * it will be placed at the begining of every JSP file which will act
  * once the user has properly logged in to the system.
  *
  * If the login/session is not proper then the user will be bounced
  * back to the login HTML page
  *
  *****/

 UserController controller = new UserController();
 Document VerificationDocument =
controller.authenticateSession((Document)session.getAttribute("loginDetails"));
 if(!VerificationDocument.getRootElement().getText().equals("Verified"))
response.sendRedirect("../index.html");
 boolean SuperUser = controller.isSuperUser((Document)session.getAttribute("loginDetails"));
 if(!SuperUser) response.sendRedirect("../index.html");

 /*****
  *
  * Verification code completed
  *
  *****/

%>

<html>
<head>
 <head>
  <title>FreeForums - Add a new Forum</title>
  <LINK REL=STYLESHEET TYPE="text/css" HREF="../css/freeforums.css">
  <SCRIPT language="JavaScript" src="../javascript/controls.js"></SCRIPT>
 </head>
<body bgcolor="#CFD4D9" text="#000000">
<form name="addforum" method="POST" action="save_forum.jsp">
 <h1>Add a new Forum</h1>
 <table border=0>
  <tr>
   <td><p>Forum Name:</p></td>
   <td><input type="text=" name="forumname" size=50></td>
  </tr>
  <tr>
   <td><p>Description:</p></td>
   <td><input type="text" name="description" size=50></td>
  </tr>
 </table><br><br>
 <input type="submit" value="Add Forum">   
 <input type="reset" value="Clear"><br>
</form>
</body>
</html>
```


Appendix G

Project Time Line

ITC307 Software Development Project
Charles Sturt University
Wagga Wagga

Spring 2001

Author: Devraj Mukherjee

FreeForums - Project Time Line

FreeForums Project Development TimeLines
ITC307 Software Development Project

Task	22-Mar	12-Apr	3-May	24-May	14-Jun	5-Jul	26-Jul	16-Aug	6-Sep	27-Sep	18-Oct	8-Nov
Project Plan and Proposal												
Implementation and Configuration of J2EE server												
Servlets/JSP based prototype implementation												
Addition of data persistance and Enterprise JavaBeans												
Addition of Multiple Forums feature to the system.												
Development of Desktop GUI Client with dummy local data processing.												
Development of XML DTD(s) and implementation of XML data parsing on client side.												
Implementation of XML handling in server side, including XML processing layer for the web interface												
XML on client/server - development of XML parsing sent from the server to the client.												
	NOT IMPLEMENTED											
Group 2 - Advanced features implementation												
	NOT IMPLEMENTED											

* The dates indicate the ending date of every timeline.

Actual Time Line

Estimated Time Line