EAI Project

Web Design Showcase

Project Members:

Anshuman Mehta

Maximilian Speicher

Tim Church

1 Contents

2 Introduction

2.1 Concept

We identified a need in the social networking market and hence proposed to build a social networking site to enable content designers to showcase their designs and receive valuable comments and feedback from the community.

2.2 Technology Platform and Environment

We have chosen the J2EE stack as the technology platform on which to develop the project. Specifically the technologies identified were as follows :

- Servlets : http requests control flow
- JSP : rendering the view
- JNDI : naming and directory services
- JMS : for asynchronous communication
- MDB : enabling messaging
- EJB : application server user session context, transactions and control flow
- JPA : transactions within user and application contexts
- CSS : standardisation of rendering view
- JDBC : database connectivity
- XML : configuration files

We worked with the following technology environment

- Glassfish : J2ee application server
- MySQL : Database
- Netbeans : Development environment
- SVN : Control repository

3 Web Design Showcase – Design

3.1 Application Architecture

The Architecture chosen for the application is described below in two broad categories.

3.1.1 MVC

We chose the popular Model View Control (MVC) architecture for our application architecture.
We identify below the design standards introduced inorder to maintain the architecture..

- Model
  The Model is represented by the Object Relational Mapping between the entity beans (of the EJB J2ee stack) .
- View
  The View is maintained in the JSP's and content rendering is standardized using CSS. The View is populated with standard Plain Old Java Objects (POJO's) to ensure consistency of data across layers.
- Control
  The Control is implemented via Servlets to handle Http requests and the business logic is maintained with the Session Beans (of the EJB J2ee stack).

  This can be visualised as follows

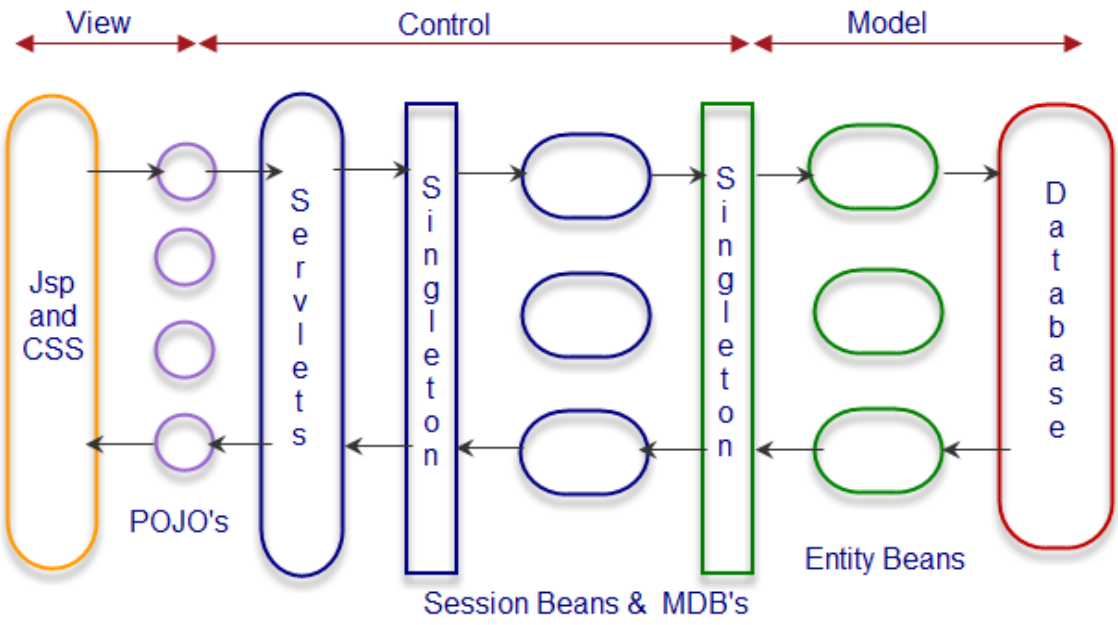


figure 1 : Application Architecture

3.1.2 Design Decisions

We discuss briefly below our design decisions for the layers as shown above in figure 1 for the Application Architecture.

- View
  The Jsp's work only with POJO's. They send and receives POJO's from their respective Servlet handlers.

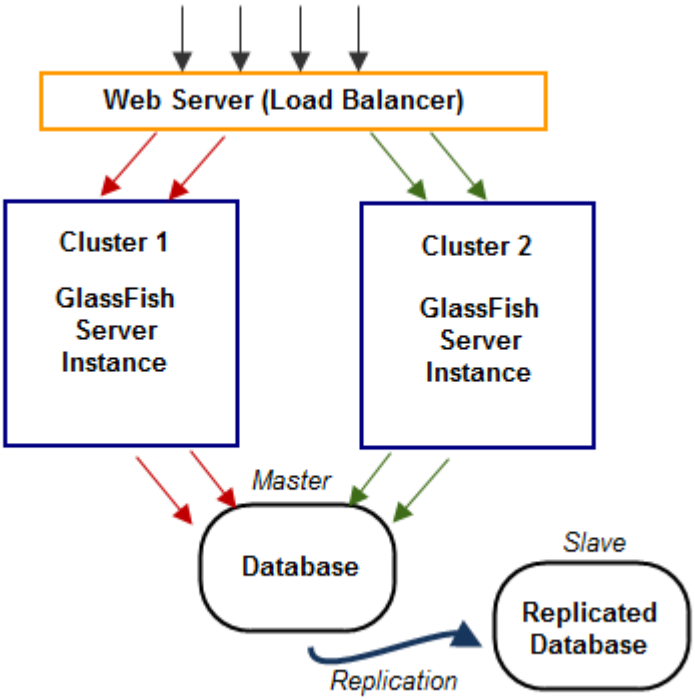
- Control
  The Singleton Session Facade provides the interface to the Servlets for issuing business requests. The Singleton Session Facade acts as the manager for delegation of business requests across different Session Beans and for maintaining transactions of business requests across Session Beans.
  The Session Beans handle all the necessary business operations in their respective business methods and delegate the task of persistence via the Singleton Session Facade over the Model Layer.
  The Singleton Session Facade over the Model Layer acts as the manager for issuing requests for CRUD persistence over the Entity Beans. It also issues EntityManagers in a round robin fashion to the requesting Session Beans.

- Model
  The Entity Beans contain the necessary mapping and CRUD operations over the Database.

3.2 System Architecture

3.2.1 Proposed System Architecture

We initially had in mind to develop the following System Architecture



At the front end we would place a Web Server (Apache Web Server) and serve the requests in a round robin fashion to two GlassFish Server Instances deployed in their respective cluster. This would help in load-balancing the requests.
At the backend both the clusters would communicate with a single Database whose content would be replicated into a failover Database. The main Database would act as the Master and the replicated Database would act as the Slave.

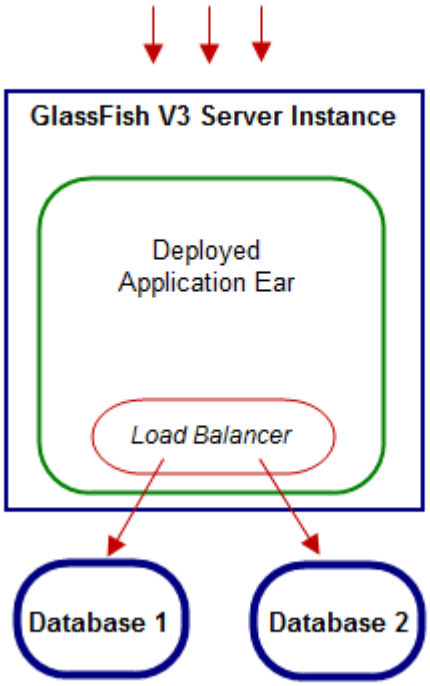
3.2.2 Problems faced in the proposed System Architecture

- GlassFish V3 does not support Apache Modjk Load Balancing.
- GlassFish V3 does not have relevant documentation to show support for Sun WebServer
- GlassFish V3 does not support clustering.
- Application developed on GlassFish V3 is not backward compatible for deployment on GlassFish V2 which supports clustering.

For the above mentioned reasons we had to give up the proposed architecture.

3.2.3 Implemented System Architecture

The system architecture implemented supports replication of Databases as shown below



3.3 Deployment

3.3.1 Requirements

The application requires the following

- GlassFish Version 3
- MySql Database

3.3.2 DB Setup

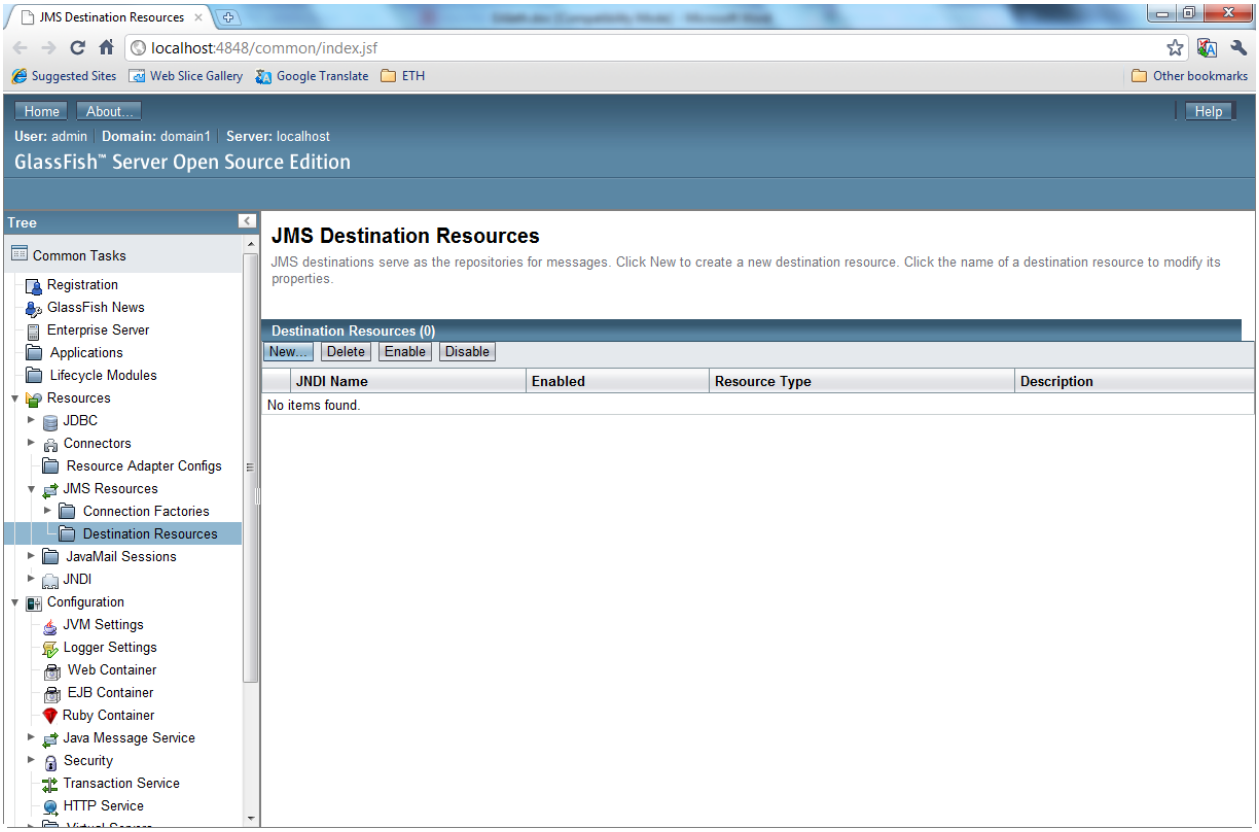
- Create two Databases named eai and eai2
- Run the attached create_db.sql in both databases.

create_db.sql

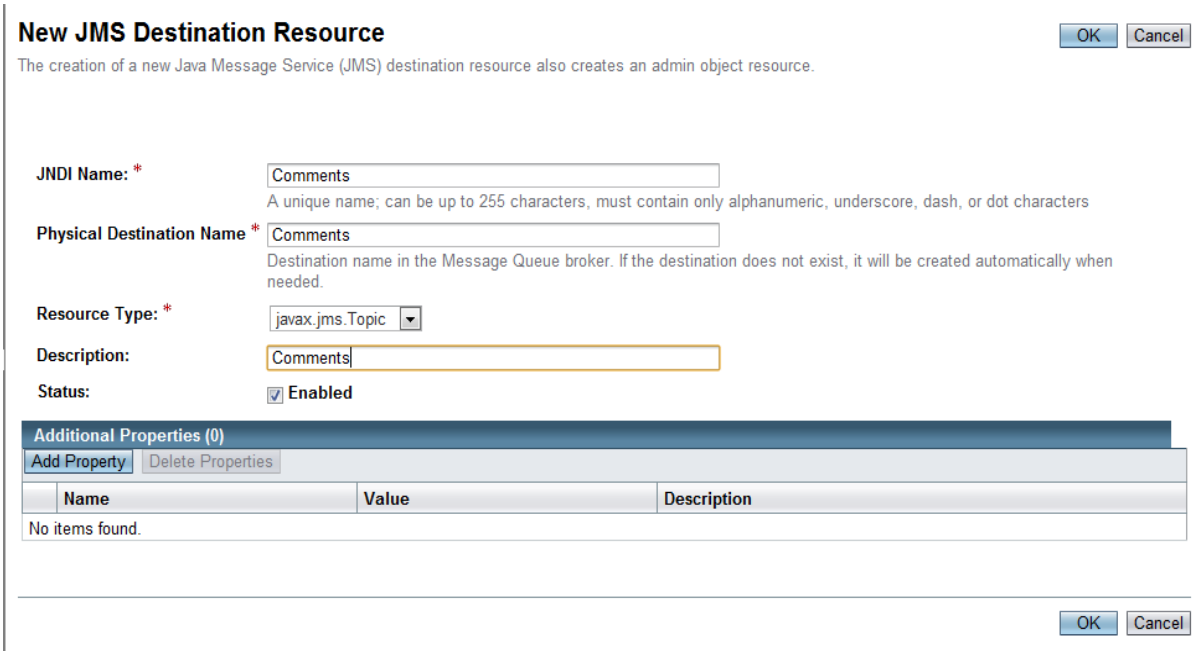
3.3.3 Glassfish Deployment

3.3.3.1 JMS Setup

- Start the GlassFish Server
- Log on into the Administration Console
- Under Resources, Navigate to JMS Resources and Choose Destination Resources

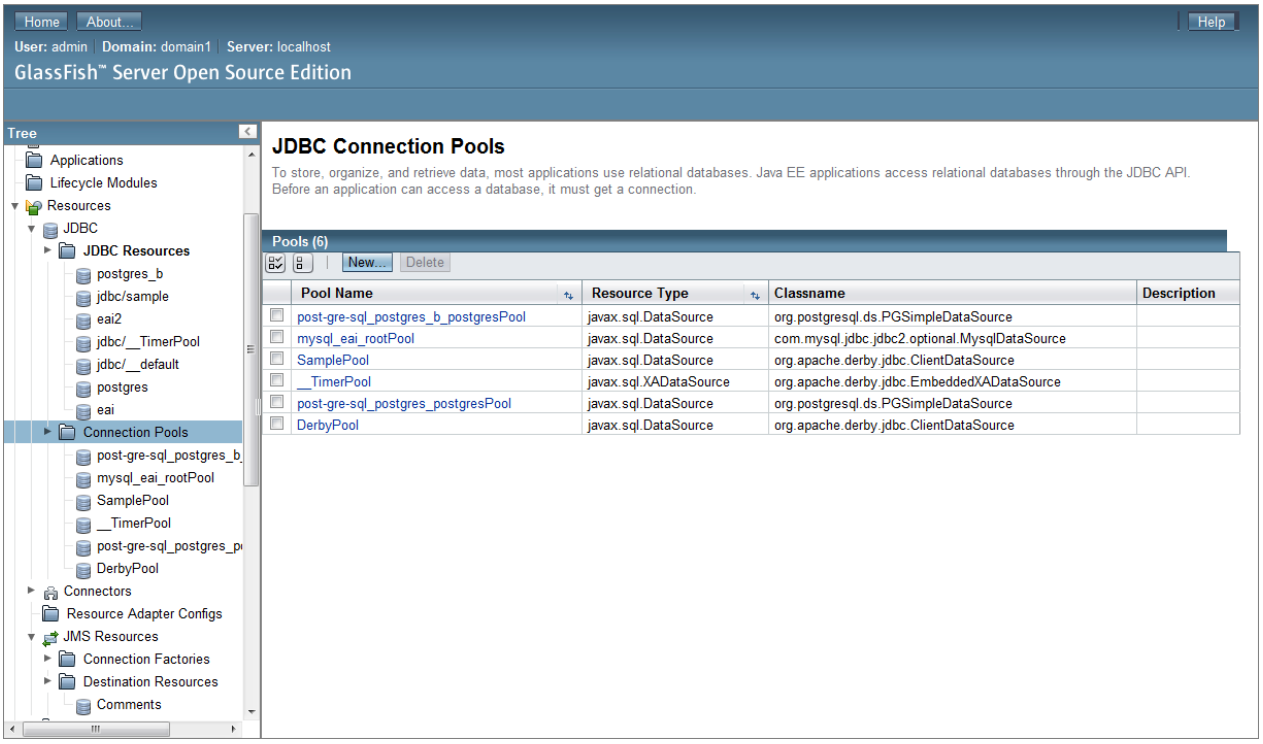


- Click on New and type in JNDI name and Physical Destination as ‘Comments’ and press OK.

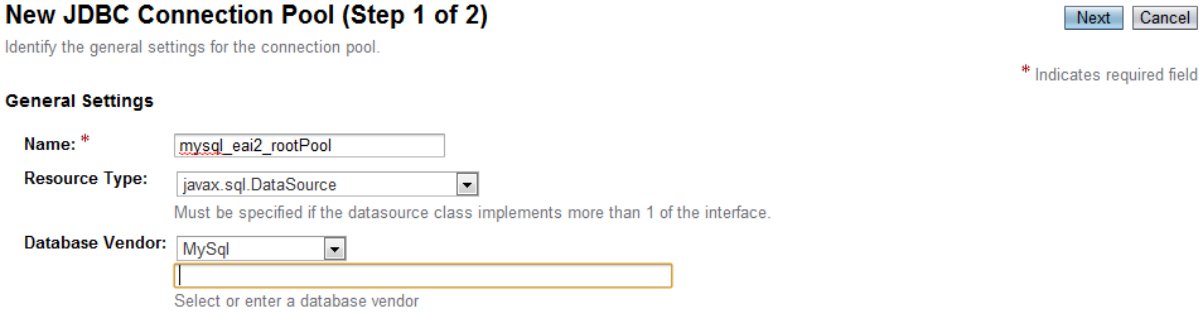


3.3.3.2 Database Setup

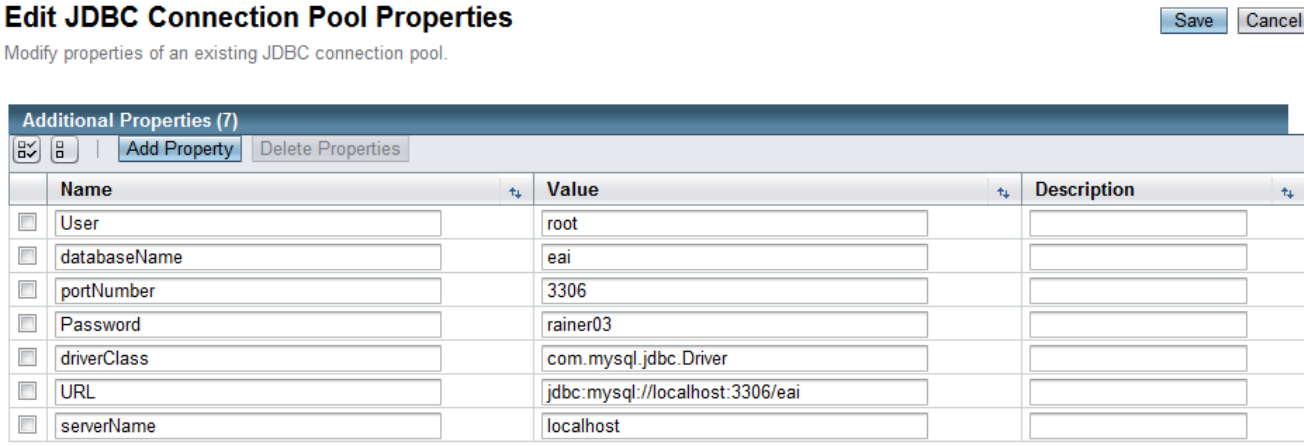
- Navigate to JDBC Resource and then Connection Pools. Click New



- Enter the Connection Pool Name and press next

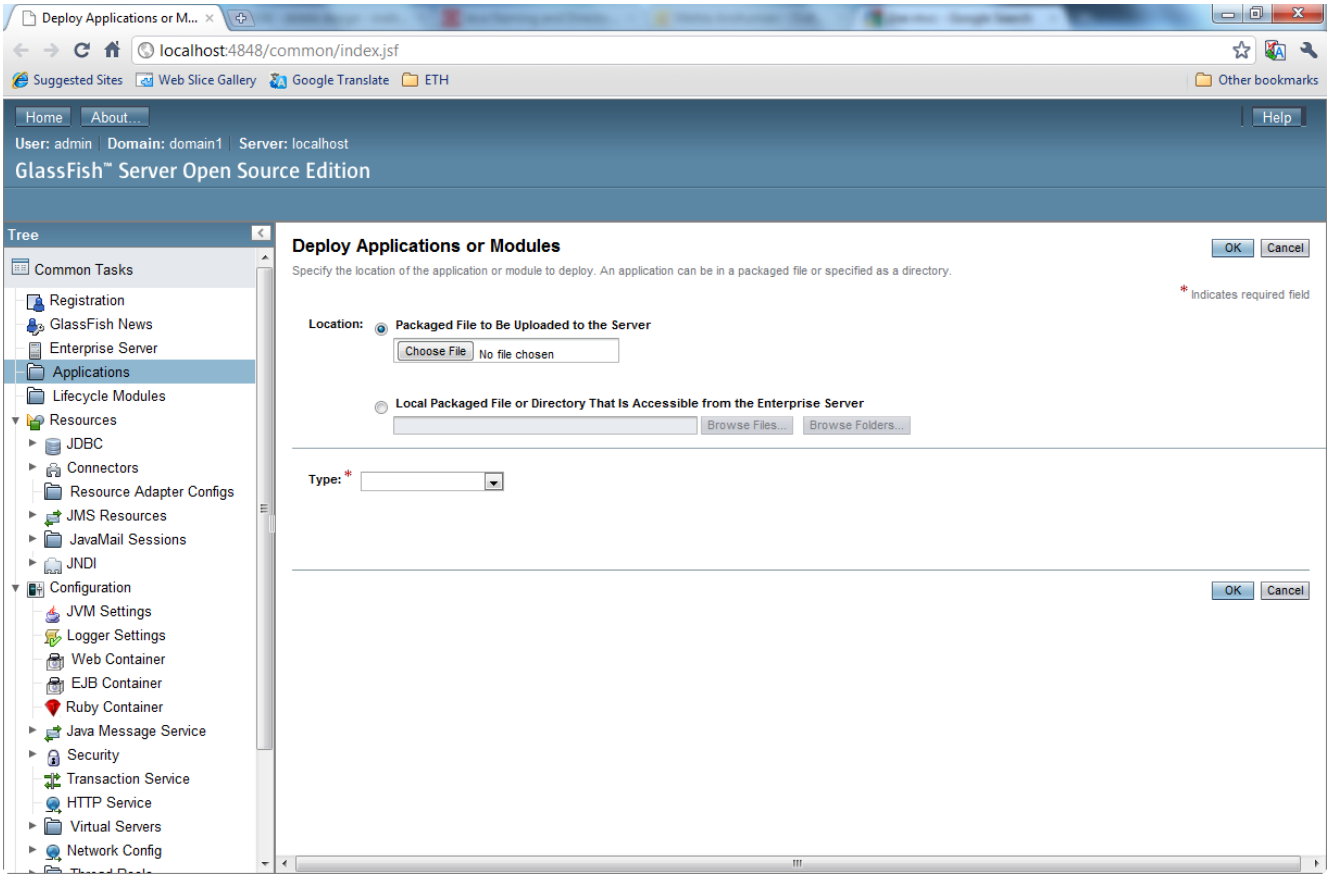


- Enter the apporiate properties according to your environment

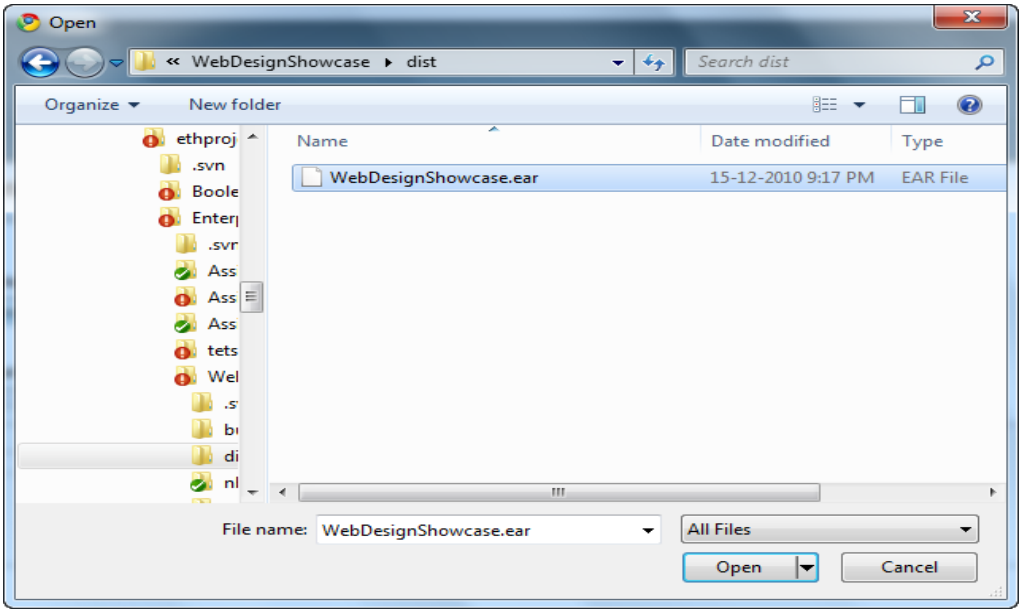


3.3.3.3 Deployment in Glassfish

- Navigate to Applications



- Click on Choose File



- Navigate to the packaged ear location in the tarball and click open.
- Choose deploy

3.3.3.4 Access Application

- Type http://localhost:8080/WebDesignShowcase-war/ to access application.

3.4 Work Packages

Artifact	Anshuman	Maximilian	Timothy
JSP			
CSS			
POJOs			
Servlets			
Session Facade Manager			
Session Beans			
Entity Beans			
Database Setup			
Web Load Balancer			
Clustering Setup			
Singleton Load Balancer			
Database Replication			
Transaction Management			
Email Notifications			
JMS / MDB			
Testing			
J2ee Design Standards			