

EAI Project

Web Design Showcase

Project Members:

Anshuman Mehta

Maximilian Speicher

Tim Church

1 Contents

1	Contents	ii
2	Introduction.....	1
	2.1 Concept.....	1
	2.2 Technology Platform and Environment	1
3	Web Design Showcase – Design	2
	3.1 Application Architecture	2
	3.1.1 MVC	2
	3.1.2 Design Decisions	3
	3.2 System Architecture	4
	3.2.1 Proposed System Architecture	4
	3.2.2 Problems faced in the proposed System Architecture	4
	3.2.3 Implemented System Architecture	5
	3.3 Deployment	5
	3.3.1 Requirements	5
	3.3.2 DB Setup.....	5
	3.3.3 Glassfish Deployment	6
	3.4 Work Packages	10

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 in order 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

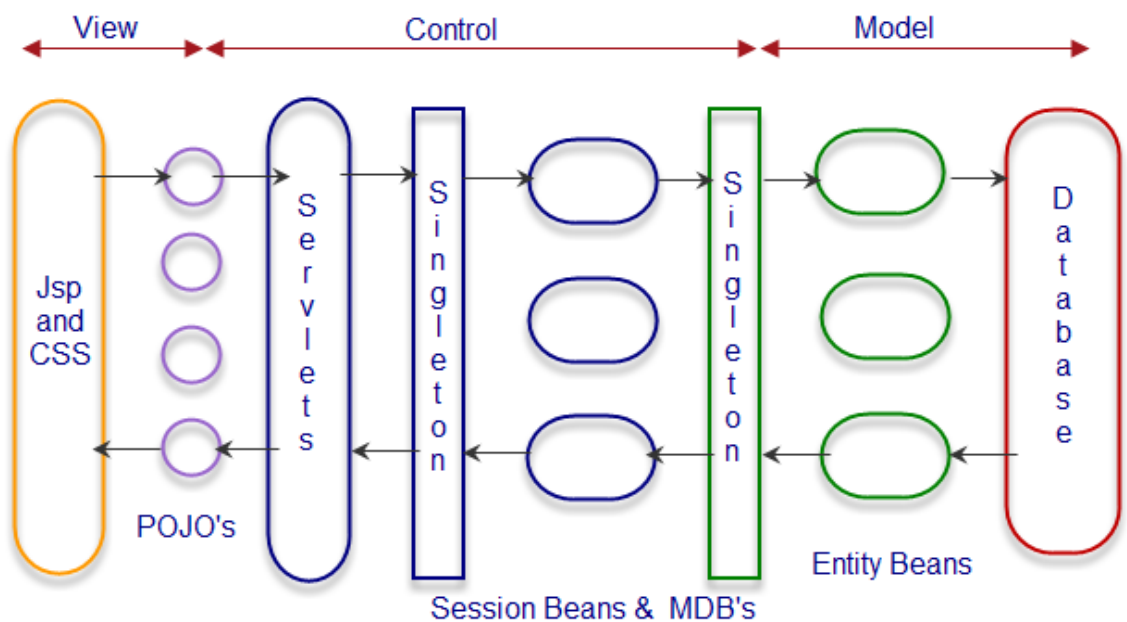


figure 1 : Application Architecture

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.

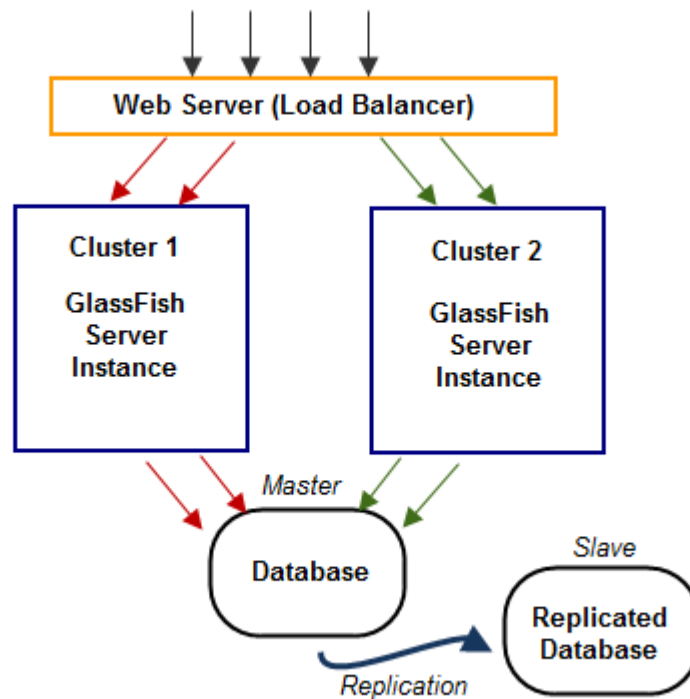
The Singleton Session Facade over the Model Layer acts as the manager for issuing requests for CRUD persistence over the Entity Beans. It also issues EntityManagers in a round robin fashion to the requesting Session Beans.
- Model

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.

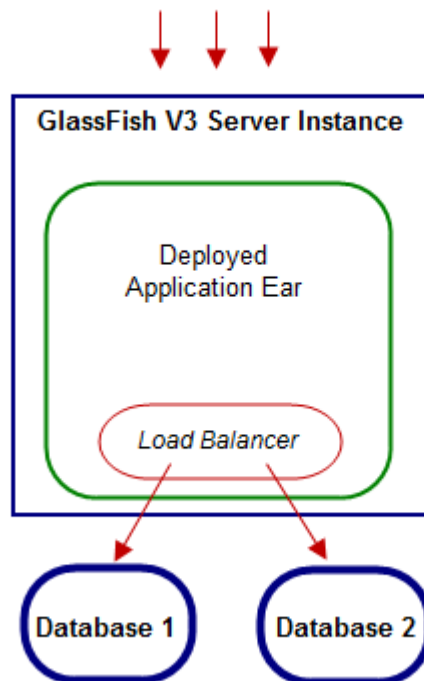
3.2.2 Problems faced in the proposed System Architecture

- 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.

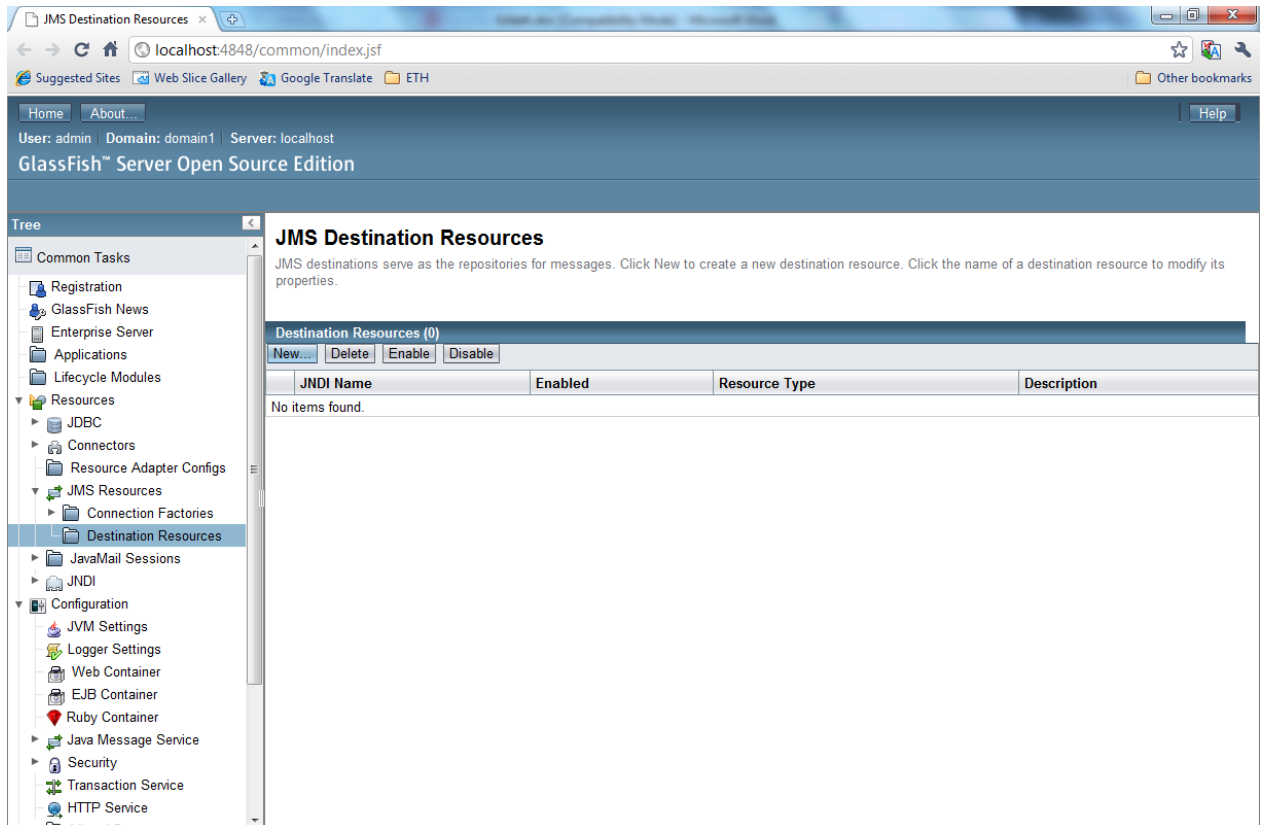


create_db.sql

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.

New JMS Destination Resource

OK Cancel

The creation of a new Java Message Service (JMS) destination resource also creates an admin object resource.

JNDI Name: *

Comments

A unique name; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

Physical Destination Name *

Comments

Destination name in the Message Queue broker. If the destination does not exist, it will be created automatically when needed.

Resource Type: *

javax.jms.Topic

Description:

Comments

Status:

☒ Enabled

Additional Properties (0)

Add Property

Delete Properties

Name	Value	Description
No items found.		

OK Cancel

3.3.3.2 Database Setup

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

The screenshot shows the GlassFish Server Open Source Edition web console. The top navigation bar includes 'Home', 'About...', and 'Help'. Below the navigation bar, the user information is displayed: 'User: admin | Domain: domain1 | Server: localhost'. The main title is 'GlassFish™ Server Open Source Edition'. On the left, a 'Tree' view shows the hierarchy: 'Applications' > 'Lifecycle Modules' > 'Resources' > 'JDBC' > 'JDBC Resources' > 'Connection Pools'. The 'Connection Pools' folder is selected, and the main content area displays the 'JDBC Connection Pools' configuration page. This page includes a description: 'To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.' Below the description is a table titled 'Pools (6)' with columns: 'Pool Name', 'Resource Type', 'Classname', and 'Description'. The table lists six pools: 'post-gre-sql_postgres_b_postgresPool', 'mysql_eai_rootPool', 'SamplePool', 'TimerPool', 'post-gre-sql_postgres_postgresPool', and 'DerbyPool'. Each pool has a checkbox in the first column. The 'Resource Type' and 'Classname' columns contain specific JDBC driver information.

Pool Name	Resource Type	Classname	Description
<input type="checkbox"/> post-gre-sql_postgres_b_postgresPool	javax.sql.DataSource	org.postgresql.ds.PGSimpleDataSource	
<input type="checkbox"/> mysql_eai_rootPool	javax.sql.DataSource	com.mysql.jdbc.jdbc2.optional.MysqlDataSource	
<input type="checkbox"/> SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/> TimerPool	javax.sql.XADataSource	org.apache.derby.jdbc.EmbeddedXADataSource	
<input type="checkbox"/> post-gre-sql_postgres_postgresPool	javax.sql.DataSource	org.postgresql.ds.PGSimpleDataSource	
<input type="checkbox"/> DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	

- Enter the Connection Pool Name and press next

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

Next Cancel

* Indicates required field

General Settings

Name: *

Resource Type:

Must be specified if the datasource class implements more than 1 of the interface.

Database Vendor:

Select or enter a database vendor

- Enter the appropriate properties according to your environment

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Save Cancel

Additional Properties (7)

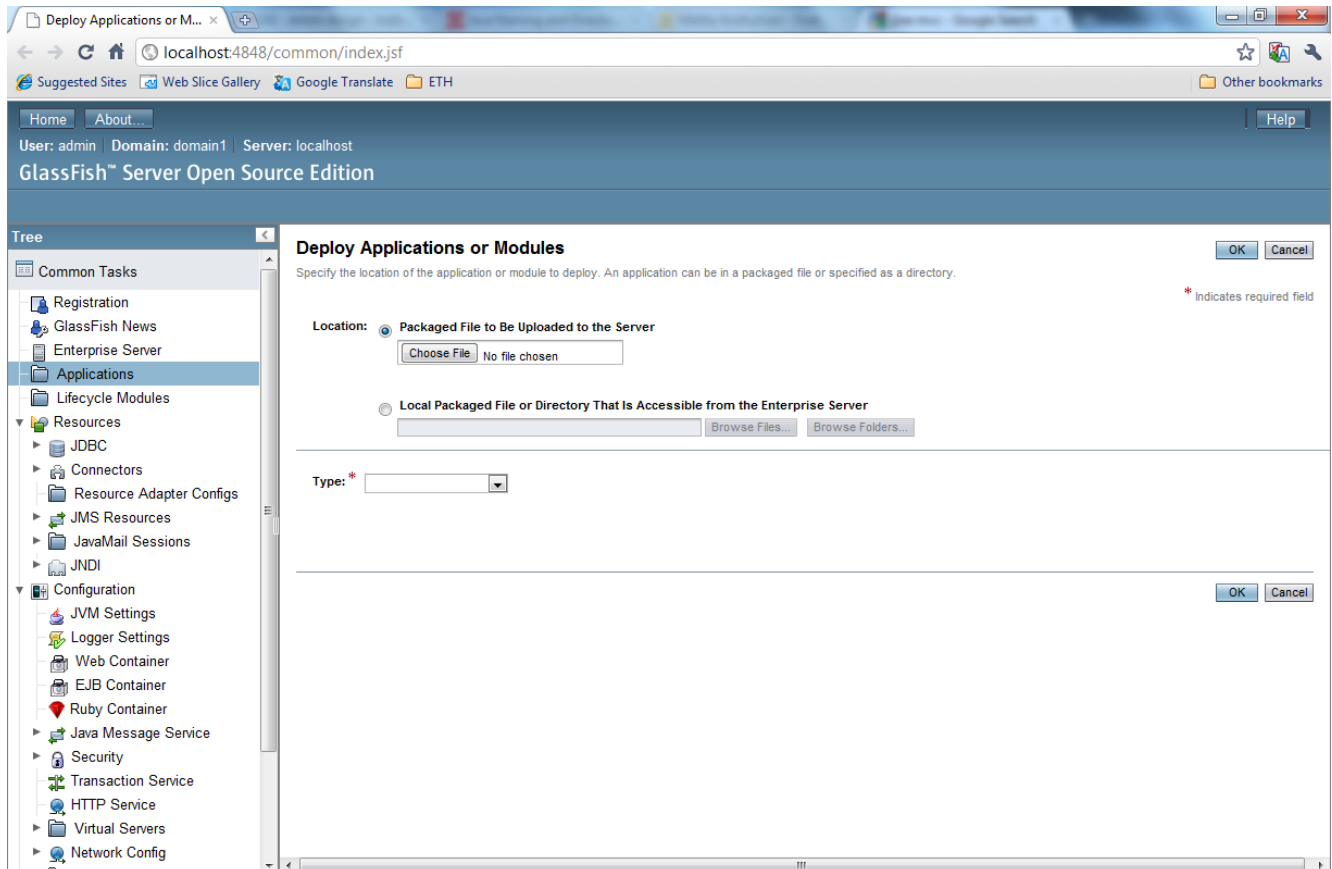
Add Property

Delete Properties

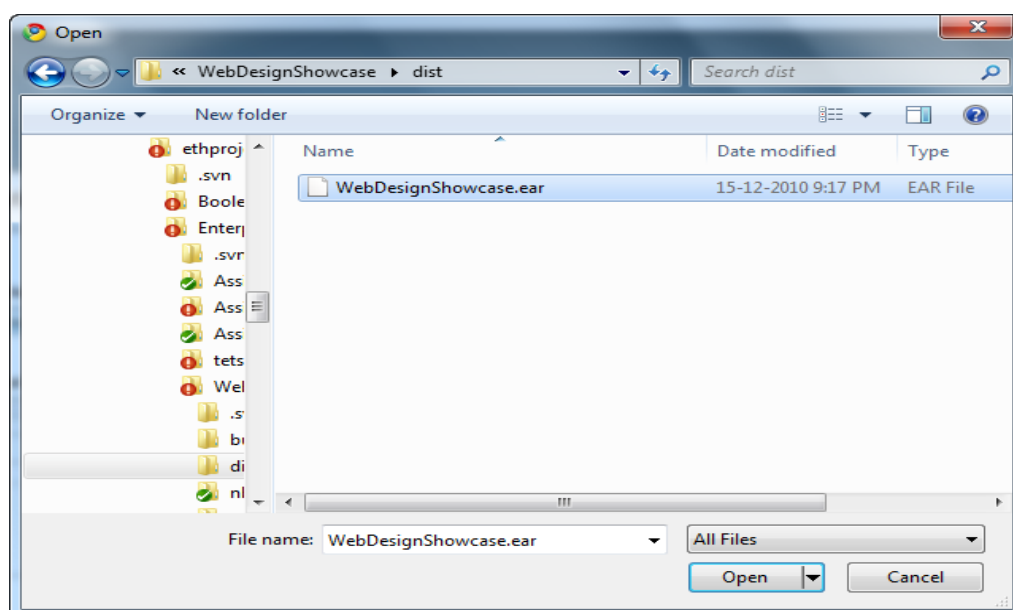
	Name	Value	Description
<input type="checkbox"/>	User	root	
<input type="checkbox"/>	databaseName	eai	
<input type="checkbox"/>	portNumber	3306	
<input type="checkbox"/>	Password	rainer03	
<input type="checkbox"/>	driverClass	com.mysql.jdbc.Driver	
<input type="checkbox"/>	URL	jdbc:mysql://localhost:3306/eai	
<input type="checkbox"/>	serverName	localhost	

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			