Porting a Web Application to Hibernate Environment

By
Shiva Somishetty

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Approved by
Dr.James D. Mooney, Committee Chairperson

Dr. Y.V.Ramana Reddy

Dr.Sumitra Reddy

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2007

ABSTRACT

Porting a Web Application to Hibernate Environment

Shiva Somishetty

Currently most of the enterprise applications are developed with Object oriented technologies such as Java and the store the data in relational databases. In an object model, navigation between the objects is possible with the use of object references. In relational database the relationships are implemented using primary key, foreign key as references. Since these two technologies are different, working with these two technologies is complex. To overcome the mismatch between the technologies, I have used Hibernate persistence framework. I have created a simple web application with two-tier architecture using the Servlet, JSP and Java technologies. The same application has been improved to reflect the three tier architecture and the functionality is also extended. This is the application I have ported to the hibernate environment. I have described the process of porting the web application to hibernate environment using same java technologies mentioned in this document.

ACKNOWLEDGEMENTS

I wish to thank my supervisor Dr. James D. Mooney for his constant support and guidance for my work and on guidance prior to this problem report. I would like to extend my thanks to Dr. Yenumula V. Reddy and Dr. Sumitra Reddy and for serving on my committee and also for their encouragement and support.

I would like to dedicate this work to my family and friends without whose unconditional support and love, I would not have completed the program.

TABLE OF CONTENTS

1. INTRODUCTION

Porting is the work involved in making some application program work in new environment. Portability is an attribute of the software that refers to the easiness to port the software from an existing environment to new environment. A software unit is portable across a class of environments or platforms to the degree that the cost of porting and adapting the application to new environment is less than effort and cost involved in redevelopment of the entire software unit.

Portability is an important issue that the current developers are challenged with. Initially every application or software unit is developed according to the requirements and specification of hardware and software platforms. But when an implementation of the existing software has to be ported to new environment the developer is left with two options: porting and redevelopment. There are different things that must be considered for porting are a component, a program, a subsystem, tools, libraries or a complete system. And the porting of an application is possible at various levels such as binary, source and intermediate levels.

Achieving the portability has both advantages and disadvantages. Advantages may be the reduced development and maintenance costs for future implementations and the other advantage would to the greater reliability that is obtained by reusing a common base. The disadvantages may be increased initial costs of development effort, possible loss of quality for the ported implementations in future. Examples for disadvantages could be the reduced performance, increased resource consumption, failure to exploit the special capabilities and failure to redesign the software components for future changes according to the advances in the development methodology.

Depending on the category of workers, there exists a different goals related to the portability. Application installers' goal is to port applications to specific new environments, the task is to analyze the environment, configuration and installation and the resources could be source or executable files, application documentation and system

documentation. Application designer's goal is to design applications which can be ported among different environments or platforms, task is to define the environment classes, developing the portable design and document for portability and the resources would be requirements specification, language specification and other development standards.

2. PROBLEM DEFINITON

Generally, many software applications are developed for a particular environment using a particular programming language and many of them have problems porting to the new environments. It usually costs more for the redevelopment of the application from scratch for different environments or platforms. Keeping this in mind the developer or the programmer has to design and develop the application such that it only needs minimal efforts or changes to port the application to new environment or platform. And also the application is designed in such a way that it should be able to accommodate new design techniques with a minimal effort. Application can be defines as follows according to the architectural standpoint: An application can be viewed as a set of components so that any component can be changed or replaced with no or minimum changes to the other parts of the application.

A web application is developed using the two-tier architecture with Java Server Pages as the required web technology for front-end development and also for handling the database requests/response handling. Developed two-tier application has been ported to three-tier using the web technologies Java Server Pages, Servlet, XML mapping. Then this application is ported to the Hibernate which performs the Object Relational Mapping (ORM) and also persists the database objects.

3. WEB APPLICATION TECHNOLOGY

Java Server Pages (JSP) Technology

Stands for “Java Server Pages”, the JSP technology was developed by Sun Microsystems as an alternative to Microsoft’s Active Server Pages (ASP) technology. It is a normal HTML with Java code embedded in them. JSP technology provides a simplified, fast way to create dynamic content. It enables the rapid development of web applications that are server and platform independent. Strength of JSP technology is the display.

Definition of JSP:

JSP is an extensible web technology that uses template data, custom elements, string languages, and server-side Java objects to return dynamic contents to a client.

Java Server Pages is the server side scripting language, runs on the server before the page reaches the viewer’s computer instead of running on the viewer’s computer. It is an extension to Java Servlet technology, allows the Java code to be embedded in the HTML pages. JSP uses XML tags and scriptlets written in Java programming language to encapsulate the logic in order to generate the dynamic web content. It passes any formatting tags such as HTML and XML to the response page. This way, it separates the logic from its design and display.

JSP pages are compiled into Servlets and may call JavaBeans components (beans) or Enterprise JavaBeans components (enterprise beans) to perform processing on the server. JSP is a key component for developing the dynamic web based applications with highly scalable architecture. JSP pages are restricted to any web server or platform. It represents the wide spectrum of industry input. JSP technology was designed to simplify the process of creating the web pages by separating web presentation from web content. JSP pages syntax is simple such as html and XML tags. JSP pages perform the form processing using the bean components. JSP pages are very easy to author.

The following section describes how the JSP page is invoked and compiled: JSP pages are implemented using the translation phase that is performed once, the first time the page is called. JSP page is first compiled into servlet class and stored in the memory, so the subsequent calls to the JSP page have fast response times. JSP specification supports the creation of XML documents through the bean components or custom tags that generate the dynamic XML output. JSP pages are capable of accessing the java bean components using some standard tags included in JSP specification.

The benefits of the JSP technology are:

1. Content and display logic are separated.
2. Simplify the web application development with JavaBeans and custom tags.
3. Supports software reuse through the use of JavaBeans and custom tags.
4. Automatic Deployment: recompiles automatically when the changes are made to JSP pages.
5. Easier to author and platform-independent.

JSP Architecture:

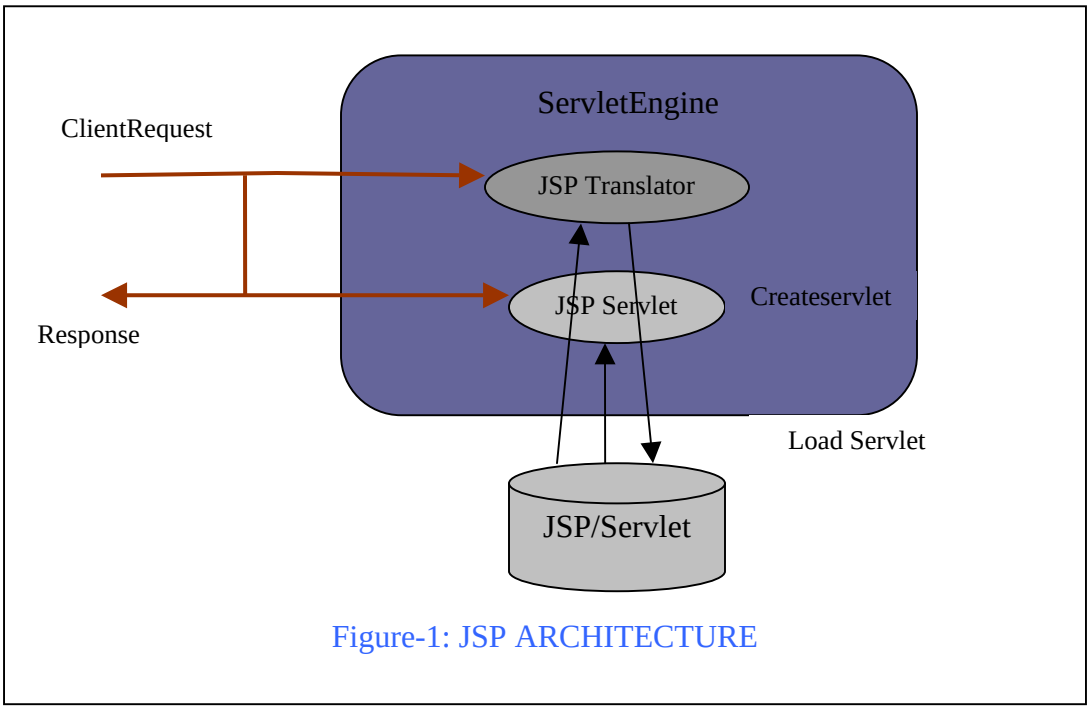


Figure-1: JSP ARCHITECTURE

Java Servlet Technology:

Java Servlet technology provides web developers with a simple and consistent mechanism for extending the functionality if the web server and for accessing existing business systems. Servlet is a program written in Java programming language that runs on the server, as opposed to the browser (applets). Servlets provide a component based platform independent method for building web applications, without the performance limitations of the CGI programs. Servlets are server and platform independent.

Servlets have access to all Java API's including JDBC API to access enterprise databases. Also they access a library of HTTP specific calls and receive the benefits of the mature java language, including portability, performance, and re-usability. Servlets are a popular choice for building interactive web applications. Servlet containers are usually a combination of web and application servers, such as, IBM WebShpere, BEA Weblogic Application Server, Sun Java System Web Server and Sun Java System Application Server.

The lifecycle of the Servlet is controlled by the container in which the Servlet is deployed. A request is mapped to the Servlet and if the Servlet is not loaded, then the container loads the Servlet class, creates an instance of the Servlet, and initializes the Servlet instance by calling init() method. Then invokes service method passing a request and response objects. Services provided a by a Servlet are implemented in service() method of *GenericServlet* class, doGet() and doPost() methods of *HttpServlet* for handling the service requests from http request. When a container needs to remove the Servlet, it calls the Servlets destroy() method. The duty of the service method is to extract the information from the client request, access the external resources and send the response back to the client. Data is shared between the web components like most of the java objects and is persistent between the invocations of J2EE and is maintained by a Database. Web components use the JDBC API 3.0 to access relational databases.

Servlets are normally multi-threaded, meaning that the Servlet container creates thread for each new request for single Servlet without any special programming. The thread is

given an object reference to the requested Servlet which provides the response back to the same thread. Servlets are mapped to URL's and the mapping details are mentioned in deployment descriptor (web.xml).

The downside of the Servlet is the presentation, typically HTML pages. HTML tags are hard coded in Servlet Java program using the print statements. So whenever the changes need do be done to the presentation layer, the java code has to be changed and then recompiled, redeployed. This results in maintenance problem for the Servlet applications.

ServletRequest and ServletResponse: A request contains the information that is sent from the client to a web server while response contains a information that is sent from server to the client. Request contains the information such as who made the request, user entered data and HTTP header entries. Response contains some static text such as HTML text or binary data such as images and also it contains HTTP response header entries and cookies, which are used to maintain session state.

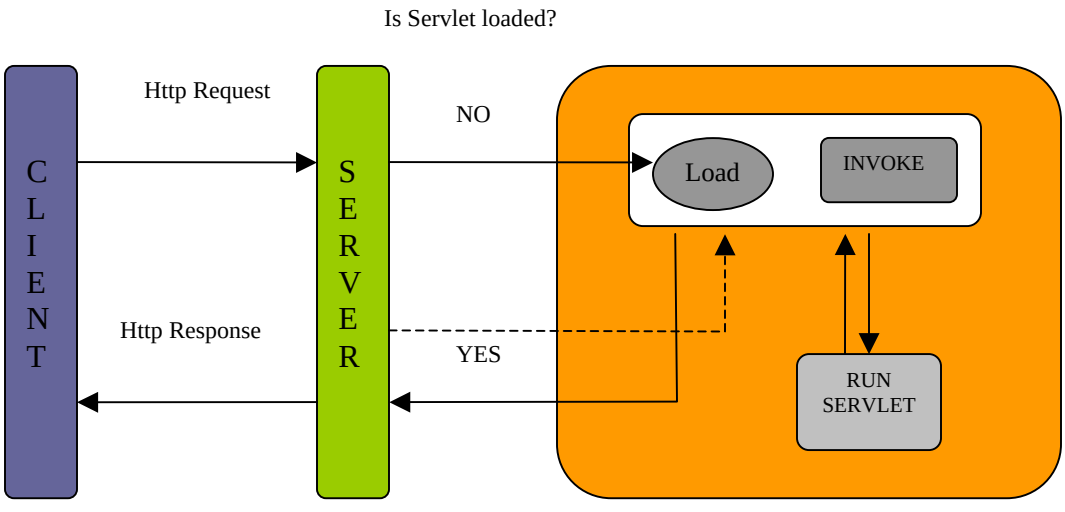


Figure-2: SERVLET LIFE CYCLE

Apache Tomcat Web Server:

Java web applications can be deployed on a variety of application and web servers that have built in support for Servlet API and JSP technology. Some widely used servers are Apache Tomcat, Bea Weblogic and IBM WebSphere. Apache Tomcat is deploys web applications very easily and has support for Servlet and JSP pages. Apache Tomcat is a open source software and is available for many different platforms. Deployment is the process of installing the web application into the web server.

Web application deployment can be done in many ways within tomcat server: statically, the web application is set up before the tomcat is started and dynamically in conjunction with tomcat manager web application or manipulating already deployed web applications. Tomcat Manager allows the URL based web application deployment features. Another tool exists in Tomcat is Client Deployer, which is a command shell based script that interacts with Tomcat Manager but provides the functionality such as compiling and validating web applications and also packages the web application into Web Application Resource (WAR) files.

Tomcat 6 installs many class loaders allowing different portions of the container and web applications running on the container, to have access to different repositories of the java classed and resources. By default, Jasper JSP Engine is configured for deploying the web applications is Tomcat. Tomcat 6.0 implement the JSP 2.0 specification by using Jasper2 JSP engine and Jasper is implemented by using the class org.apache.jasper.servlet.JspServlet

Hibernate

Hibernate is a professional open source software that is distributed under the GNU Lesser General Public License. It is a critical component of JBOSS Enterprise Middleware System suite of products. Hibernate is a framework that performs Object to Relation Mapping (ORM). That is, it maps an Object oriented domain model to a traditional relational database. It relieves the developer from a significant amount of relational data

persistence related programming tasks. Hibernate adapts to the development process, whether starting from scratch or from an existing database.

Features:

Hibernate is a framework that synchronizes the state of objects within the database.

a. Maps objects to relational model: Hibernate maps Java classes to Database tables and from Java data types to SQL data types, also provides data query and retrieval facilities. It relieves the developer from manual result set handling and object conversion by generating its own SQL calls, keeping an application portable to all SQL databases, with database portability delivered at very little performance overhead.
b. Transparent persistence: It provides transparent persistence for "Plain old java objects", the only requirement here is that the persistent class has no argument constructor (not necessarily public).
c. EJB persistent implementation: Hibernate can be used both in standalone java applications and Enterprise java applications using Servlets or EJB session beans.

Working with Hibernate:

Hibernate is capable of mapping all Object Oriented relationships (such as inheritance, interfaces, associations, composition, ternary associations) between objects. Hibernate follows both Bottom-up (existing database to OO model) and Top-down (OOModel to design the database schema) approaches:

a. ***Bottom-up***: From an existing database to a pure OO model.

The problem with OO in relation to databases is that one could take an object model and use a straightforward (and unrealistic) class-to-table mapping approach. With Hibernate, the mapping between the database and object model can be managed in such a way that neither the OO model nor the database model need to influence the other model. Even an existing relational database can be mapped to an OO model allowing flexibility in OO model. They are some tools to

build an Object Model from an existing database but there is a risk that they might give us a poor OO model which is not recommended.

b. ***Top-down***: let the Object model dictate the database design

Hibernate can create the database schema by allowing the Object Model to dictate the database design. It can even update the database.

Oracle 10g Database:

Oracle is made up of a set of processes running in your operating system. These processes manage how data is stored and how it is accessed. The Oracle RDBMS stores data logically in the form of tablespaces and physically in the form of data files. Oracle10g has a browser based user interface which is very convenient for accessing the database info rather than executing the commands using SQL command prompt. It provides an organized way of storing, managing and retrieving the database information. Browser based user interface allows the administrator with add/manage users, add/delete/update tables functionality with its convenient GUI.

JDBC 3.0 API:

JDBC API is used for accessing the relational databases, hides database specific details from the application. JDBC API keeps the everyday tasks simple and easy for implementation too. It defines a set of java interfaces that are implemented by vendor specific JDBC drivers. Applications use these operations to perform the database operations. Most of the JDBC API is located in java.sql package. For example, DriverManager, ResultSet, Connection, PreparedStatement, CallableStatement are available interfaces are defined in “java.sql” package. Advances functionality (DataSource) is included in the “javax.sql” package. Every database server has its corresponding JDBC drivers.

Eclipse IDE:

It is an IDE (Integrated Development Environment). Eclipse is an open source community, whose projects are focused on building an open development platform

comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. Eclipse includes a number of unique features such as code refactoring, automatic code updates/installs (via the Update Manager), and integration with the Jakarta Ant build tool.

4. APPLICATION HIGH LEVEL DESIGN

High Level Architecture of the Hibernate Application:

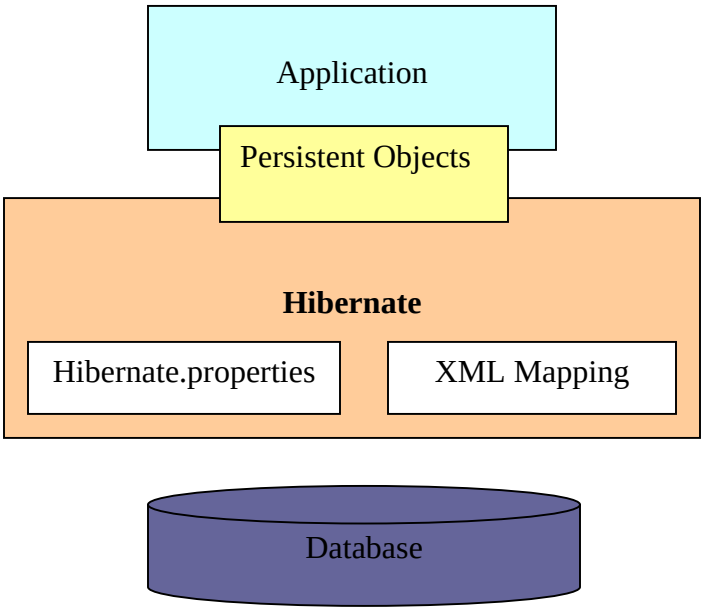


Figure-3: High Level Architecture of Hibernate

From Figure-3, it is clear that Hibernate uses Database (Oracle 10g) and configuration data (mapping files and configuration files) to provide persistent objects to the application. Hibernate architecture is divided into three main components:

a) Connection Management: It provides efficient management of the database connections. Database connection, the most expensive part of interacting with the database, requires a lot of resources to open and close the database connection.
b) Transaction Management: It provides the most important feature concurrent execution. That is, it provides an ability to the user to execute more than one database statements or operations at a time.
c) Object Relational Mapping (ORM): ORM maps the data representation from an object model to relational data model. It is used to perform all kinds of database operation on the underlying tables.

Hibernate creates its own SQL to perform database operations corresponding to the operations that the applications perform on the java objects. Hibernate provides a lot of flexibility to the user. The architecture that I used for out application is called lite architecture, defined as the architecture which uses only Object Relational Mapping.

5. PORTING FROM JDBC TO HIBERNATE ENVIRONMENT

First of all, an application has been designed with the two tier architecture with JSP pages handling all the presentation and logic to handle the database related tasks. Eclipse IDE is used for the development of java Servlets, jsp pages and for writing the configuration files and deployment descriptor file. JSP page includes the code to connect to the database, reading the user entered values, inserts the values into the database, and then finally retrieves all the employee or department related information in the database.

The deployment descriptor “web.xml” is a must to understand the URL patterns of the Servlets that are called from the JSP pages and there is a tag to mention the welcome page (home page) of the project. The deployment descriptor is same for both the JDBC project and hibernate project. Department number is the Primary Key (PK) for the Department table and Employee Number is the Primary Key (PK) for the Employee table. Department number in the Employee table is the Foreign Key (FK) for Employee table. The next section describes the building of a web application with three tier architecture from the existing two tier architecture application.

Three-tier architecture application is developed using the JSP as the front end technology. The home jsp page forwards the requests such as employee listing, department listing, add department and add employee to the corresponding Servlets. Servlets read the requests from the JSP page as a java objects, called request objects. Request object contains the parameters such as the input field values that the user submitted from the request page. Based on the request values, Servlet performs its processing then forwards the response to the JSP page for display purpose. For example, “list departments” action is performed using the following flow between the Servlet and JSP pages. The home JSP page sends a “list departments” request to the “listdepartments” Servlet, then the Servlet access the “department” table from the database and reads the values into the JDBC result-set object and the object is mapped to java object (department object which has department number and department name as the instance variables) then redirects the response to the “listdepartments” JSP page for displaying all the departments. The list

departments page provides the features to edit the database records. Edit requests will be redirected to a JSP page where the user can modify the existing fields values and update them in the database using the Servlet and JDBC connection. Servlets and jsp pages are used same as above for implementing all the functionality in the web application development. The next section deals with porting the web application to hibernate environment.

Hibernate API is available for free download as a zip archive file that contains a bunch of jar files. Archive is downloaded from www.hibernate.org website and extracted to a folder of our choice to store all the java archives files. All required hibernate java archives are added to the development environment in Eclipse IDE. These jars provide all the capabilities of hibernate to the web application as a library of java classes to access them when necessary for implementing the database persistency. The requirement of hibernate is to have POJO (Plain Old Java Object) file per database table. The POJO bean class contains instance variables corresponding to the columns in the corresponding database table. For example, the POJO class “Department.java” is created for representing the database table “department” and contains the instance variables department number and department name to represent the columns in the department table. Getters and setters (get and set methods) are defined for all the instance variables and the no argument constructor is another requirement of Hibernate POJO class.

Hibernate mapping file (projectname.hbm.xml) is required to understand the database tables and structure of the tables and corresponding java classes. Details such as primary key, foreign key constraints and index keys are also mentioned in the mapping document. Hibernate mapping files can be one or many. Because one can write a single file that stores all the database tables and java class mappings or a mapping file database table. It doesn’t matter whether the project has one or more mapping files. A configuration file (hibernate.cfg.xml) stores the information about the connection parameters (URL, connection string, driver name, username, password) for establishing the database connection. It also lists all the mapping files that are used in the project. These two files are important in understanding the database structure, connection parameters and the

whole project. An advantage of using hibernate is that hibernate doesn't care with database that the project accessing. If it is required to access another database with the same structure, no changes have been made to the project except few changes to the configuration file (hibernate.cfg.xml). All the screens in this application are listed in an Appendix at the end of this document.

Hibernate Application with other Databases:

Hibernate applications are easily migrated to other databases without making any changes to code except a small change to the XML configuration file (hibernate.cfg.xml) which contains the details of the backend database an application accesses. XML configuration file has a property called "dialect" which represents the target database. Making change to the dialect property to some other database such as SQLServer, Derby, and MySQL will make the application to use the specified target database. The dialect property that is used for Oracle database is the following:

<property name="dialect">org.hibernate.dialect.OracleDialect</property>

MEASURING PORTABILITY

Some of the standard cost estimation techniques may be used to estimate the increase in development cost due to portable development. Analysis of cost of porting involves analyzing the match between the interfaces of the software unit and those of the target. Degree of portability can be computed for a specific software unit and target as:

DP = 1 - (cost to port / cost to redevelop)

If DP value is greater than zero, then porting is more cost-effective than redevelopment. In general, porting costs will be inversely proportional to the DP. A DP value of one represents "perfect portability."

As per the definition,

Total No. of hours to port the application to the new framework = No. of hours to port the controller layer + No. of hrs to port the web tier + No. of hrs to develop the business logic

= 30hrs + 10hrs + 10hrs

= 50 hrs

Total No of hrs to redevelop the module using Hibernate framework = No of hours to redevelop the controller layer + No. of hrs to redevelop the web tier + No. of hrs to develop the business logic

= 60hrs + 20hrs + 20hrs

= 100hrs

DP = 1 - 50/100 = 1 – 0.5 = 0.5

DP value is greater than zero. Therefore it is much economical to port the whole application using the new Hibernate framework.

6. CONCLUSIONS

By analyzing the web application, it was observed that the web application developed using JDBC can be easily migrated to the new hibernate framework. Number of files that deal with business logic needs to be modified and additional code has to be written according to the database and object models. And few mapping files and configuration file needs to be added to the environment. If the cost or porting the application is measured in time, the cost to port the application would be little high, thus degree of portability of this application is good.

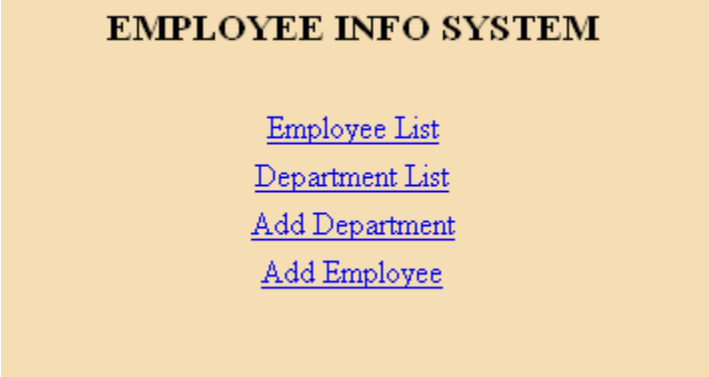
From these results, it is concluded that it is more economical to port the JDBC based web application to Hibernate framework instead of redeveloping the whole application. Hibernate applications are easily migrated to or compatible with all other databases without making any changes to the code.

7. REFERNCES

1. “CS 533: Developing Portable Software” http://www.csee.wvu.edu/~jdm/classes/cs533/
2. Servlets http://java.sun.com/products/servlet/index.jsp
3. JSP http://java.sun.com/products/jsp/index.jsp
4. Hibernate http://www.hibernate.org/hib_docs/reference/en/html/architecture.html
5. Apache Tomcat 6.0 http://apache.tomcat.org
6. Oracle-10g http://www.oracle.com
7. “Pro Hibernate 3” By Dave Minter, Jeff Linwood

8. APPENDIX

Homepage of the Application



Employee List Page

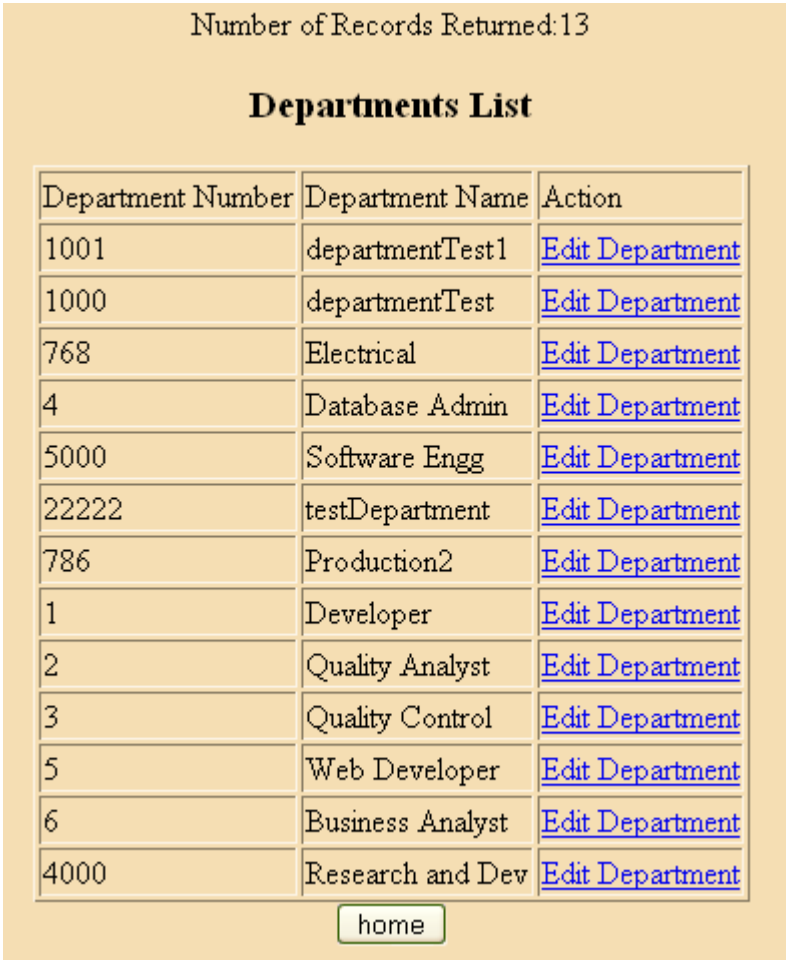
Number of Records Returned:9

Employee List

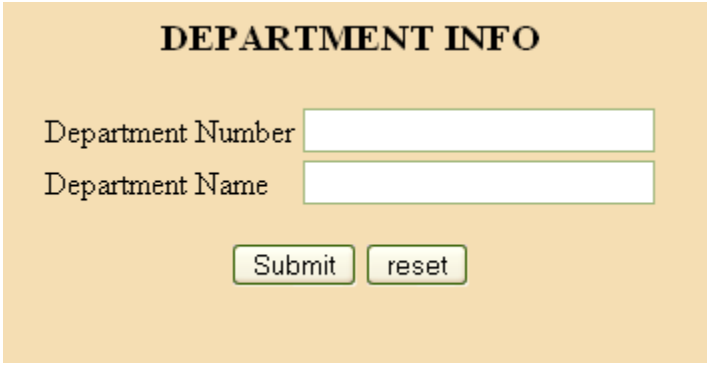
Employee Number	Employee Name	Employee Salary	Department Number	Department Name	Action
2000	testEmployee	54545	Software Engg	5000	Edit Employee
888	Shiva Somishetty	8888	testDepartment	22222	Edit Employee
3333	employeedate	3333	departmentTest1	1001	Edit Employee
78	77	888	Quality Control	3	Edit Employee
6574	John Smith	67878	Quality Analyst	2	Edit Employee
81	81	881	Quality Control	3	Edit Employee
32	Shiv	777	departmentTest	1000	Edit Employee
6	ygu	666	Developer	1	Edit Employee
1001	employee1	100100	Quality Control	3	Edit Employee

home

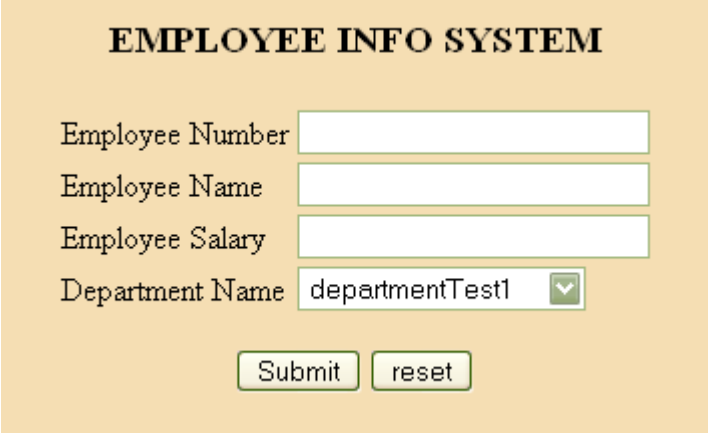
Departments List Page



Add Department Screen



Add Employee Screen



The following section contains the source code listings for few jsp pages in the application.

1. Home Page (home1.jsp)

```
<HTML>
<BODY bgcolor="wheat">
  <CENTER>
 <TABLE>
        <TR><TD> <h3>EMPLOYEE INFO SYSTEM</TD></TR>
        <TR><TD>
        <CENTER>
<a href="/ProblemReportHibernate/EmployeeList.do">Employee List</a></CENTER>
        <TR><TD align = center>
 <a href="/ProblemReportHibernate/DepartmentList.do">Department List</a>
        <TR><TD align = center>
 <a href="/ProblemReportHibernate/addDepartmentForm.jsp">Add Department</a>
        <TR><TD align = center>
 <a href="/ProblemReportHibernate/DeptNames2Employee.do">Add Employee</a>
 </TABLE>
</BODY>
</HTML>
```

2. Add Employee Page (addEmployeeForm.jsp)

```
<%@ page import="java.util.*" %>
<%@ page import="edu.wvu.report.entity.Department" %>
<HTML>
<BODY bgcolor="wheat">
<%
ArrayList<Department> departments = (ArrayList)request.getAttribute("departments");
        if(departments != null){%>
        <%}
        else{    %>null<%
        } %>
        <form name="homeForm" method="post" action="AddEmployee.do">
        <CENTER>
        <TABLE>
        <TR> <h3>EMPLOYEE INFO SYSTEM</TR>
        <TR>  <TD>Employee Number</TD>
                <TD><input type="text" name="EmpNo" size="25"></TD></TR>
        <TR>  <TD>Employee Name</TD>
                <TD><input type="text" name="EmpName" size="25"></TD></TR>
        <TR>  <TD>Employee Salary</TD>
                <TD><input type="text" name="EmpSal" size="25"></TD></TR>
        <TR>  <TD>Department Name</TD>
                <TD><select name="deptNo">
                <% for(int i=0; i<departments.size();i++){%>
                        <option value="<%= departments.get(i).getDeptNo()%>">
                                <%= departments.get(i).getDeptName()%>
                        </option>
                <% } %>
                </TD>
        </TR>
        </TABLE>
```

```
   <input type="submit" value="Submit">
   <input type="reset" value="reset">
   </CENTER>
   </form>
</BODY>
</HTML>
```

3. Add Department Page (addDepartmentForm.jsp)

```
<HTML>
<BODY bgcolor="wheat">
        <form name="deptForm" method="post" action="AddDepartment.do">
        <CENTER>
<TABLE>
        <TR> <h3>DEPARTMENT INFO</TR>
        <TR>  <TD>Department Number</TD>
                <TD><input type="text" name="deptNo" size="25"></TD> </TR>
        <TR>
                <TD>  Department Name</TD>
                <TD>  <input type="text" name="deptName" size="25"> </TD>
        </TR>
</TABLE>
      <input type="submit" value="Submit">
      <input type="reset" value="reset">
      </CENTER>
      </form>
</BODY>
</HTML>
```

4. Employee Listing Page (employeeList.jsp)

```
<%@ page import="java.util.*" %>
<%@ page import="edu.wvu.report.entity.EmployeeORM" %>

<HTML>
<BODY bgcolor="wheat">
	<%try{
	%>
<CENTER>
<%
ArrayList<EmployeeORM> employees = (ArrayList)request.getAttribute("employees");
if(employees != null){%>
Number of Records Returned:<%=employees.size()%>
<%}
else{
%>null<%
}
%>
<CENTER><H3>Employee List</H3></CENTER>
<TABLE border=1>
	<TR>
		<TD>Employee Number</TD>
		<TD>Employee Name</TD>
		<TD>Employee Salary</TD>
		<TD>Department Number</TD>
		<TD>Department Name</TD>
		<TD>Action</TD>
	</TR>
	<%
		for(int i=0;i<employees.size();i++){
```

```
        %>
        <TR>
                <TD><%= employees.get(i).getEmpNo() %></TD>
                <TD><%= employees.get(i).getEmpName() %></TD>
                <TD><%= employees.get(i).getEmpSal() %></TD>
                <TD><%= employees.get(i).getDepartment().getDeptName()%></TD>
                <TD><%= employees.get(i).getDepartment().getDeptNo()%></TD>
        <TD><a href = "PreEditEmployee.do?id=<%= employees.get(i).getEmpNo()%>"
> Edit Employee</a></TD>
        </TR>
        <% } %>
        </TABLE>
<form action="home1.jsp">
        <input type="submit" value="home" />
</form>
        <%      }
        catch(Exception e){
        out.println(e.getMessage());
        }%>
</CENTER>
</BODY>
</HTML>
```