|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 08** | **DESIGNING ENTERPRISE APPLICATION USING JMS – ONLINE LIBRARY** | | |
| Date of Exercise |  | Date of Output Verification |  |

**Aim**

To develop and deploy the enterprise Application for Online Library using J2EE programming and JMS.

**Description**

Messaging is a method of communication between software components or applications.

A messaging client can send messages to, and receive messages from, any other client.

Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages.

Clients require a means to communicate with each other to request services and to deliver services.

JMS Provider

* It is the messaging agent that is responsible for managing the messaging service

JMS clients

* Clients are apps and components written in Java that use the provider for communication

Messages

* are the objects that communicate information between JMS clients.

Administered objects

* JMS objects used for communication with JMS Provider.
* Are preconfigured JMS objects created by an administrator for the use of clients.
* The two kinds of administered objects are destinations and connection factories

**Procedure**

* Create a Java EE Project in netbeans
* Create database tables with primary keys
* Create entity and session beans for those tables
* Design the front end with jsp
* Create a java class for business logic
* And use servlets to pass the data across jsp’s, jms bean and java classes
* For JMS, right click on the server
* Go to admin console
* Add a connection factory and connection queue
* Create a JMS class in beans
* Write the business logic
* Call that JMS service in servlet

**Program**

Web.xml

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

<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd">

<servlet>

<servlet-name>JCAction</servlet-name>

<servlet-class>JCAction</servlet-class>

</servlet>

<servlet>

<servlet-name>JMSAction</servlet-name>

<servlet-class>JMSAction</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>JCAction</servlet-name>

<url-pattern>/JCAction</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>JMSAction</servlet-name>

<url-pattern>/JMSAction</url-pattern>

</servlet-mapping>

<session-config>

<session-timeout>

30

</session-timeout>

</session-config>

</web-app>

Header.jsp

<nav class="navbar navbar-default ">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">WS | Online Library System</a>

</div>

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li class="active"><a href="index.jsp">Home <span class="sr-only">(current)</span></a></li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false"> Extras <span class="caret"></span></a>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li role="separator" class="divider"></li>

<li><a href="#">Separated link</a></li>

<li role="separator" class="divider"></li>

<li><a href="#">One more separated link</a></li>

</ul>

</li>

</ul>

<form class="navbar-form navbar-left" role="search">

<div class="form-group">

<input type="text" ng-model="query" class="form-control" placeholder="WS Dynamic Search...">

</div>

<button type="button" class="btn btn-default">Don't Click</button>

</form>

<ul class="nav navbar-nav navbar-right">

<li><a href="#">Options<span class=""></span></a></li>

<%

if (session.getAttribute("name") == null) {

%>

<li><a href="login.jsp">Login <span class=""></span></a></li>

<%

} else {

%>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false"> <%=session.getAttribute("name")%> <span class="caret"></span></a>

<ul class="dropdown-menu">

<li><a href="#">Account Settings</a></li>

<li><a href="#">Privacy Settings</a></li>

<li role="separator" class="divider"></li>

<li><a href="logout.jsp">Logout</a></li>

</ul>

</li>

<%

}

%>

</ul>

</div>

</div>

</nav>

Footer.jsp

<style>

#footer{

position: fixed;

bottom: 0px;

text-align: center;

line-height: 1.2em;

background: #357ebd;

color: whitesmoke;

font-size: 1.2em;

right: 0px;

left: 0px;

letter-spacing: 1.5px;

padding-top: 12px;

}

</style>

<div id="footer">

<p class="footer-block">Developed by WS</p>

</div>

Index.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>WS | Banking System</title>

<script src="js/jquery-1.11.3.min.js"></script>

<script src="js/angularjs.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/script.js"></script>

<link rel="stylesheet" href="css/bootstrap.min.css" />

<link rel="stylesheet" href="css/styles.css" />

</head>

<body ng-app="myApp" ng-controller="booksCtrl">

<jsp:include flush="true" page="header.jsp"></jsp:include>

<jsp:include flush="true" page="index\_body.jsp"></jsp:include>

<div style="margin-bottom: 50px;"></div>

<jsp:include flush="true" page="footer.jsp"></jsp:include>

</body>

</html>

Index\_body.jsp

<%

if (session.getAttribute("name") != null) {

if (session.getAttribute("name").toString().equals("William Scott")) {

%>

<a href="admin\_modify.jsp" class="btn btn-success col-sm-offset-10"> Modify Database </a>

<%

}

}

%>

<div class='container alert alert-danger' id="errormessage">Error, Try After Login!....</div>

<div class="panel col-sm-6 col-sm-offset-3 panel-default">

<div class="panel-heading panel-success">Search Results</div>

<div class="panel-body">

<table class="table table-striped">

<thead>

<tr>

<th>Book Name</th>

<th>Book ID</th>

<th>Count of Books</th>

<th>Rent</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="x in books| filter:query | orderBy:orderByField:reverseSort">

<td>{{ x.name}}</td>

<td>{{ x.id}}</td>

<td>{{ x.count}}</td>

<td><input type="radio" name="select" value="{{x.id}}"/></td>

</tr>

<tr>

<td></td>

<td></td>

<td></td>

<td><a onclick="rent()" class="btn btn-success">Rent</a></td>

</tr>

</tbody>

</table>

</div>

</div>

<script>

$(document).ready(function () {

$("#errormessage").hide();

});

function rent() {

$("#errormessage").slideUp();

$.ajax({

type: "post",

url: "JCAction?Rent=" + $('input[name="select"]:checked').val(),

success: function (data) {

if (data == "success")

window.location.href = 'index.jsp';

else

$("#errormessage").slideDown();

}

});

}

var app = angular.module('myApp', []);

app.controller('booksCtrl', function ($scope, $http) {

$http.get("JCAction?GetAll=true")

.success(function (response) {

$scope.books = response;

});

});

</script>

Login.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>WS | Login</title>

<script src="js/jquery-1.11.3.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/script.js"></script>

<link rel="stylesheet" href="css/bootstrap.min.css" />

<link rel="stylesheet" href="css/styles.css" />

</head>

<body>

<jsp:include flush="true" page="header.jsp"></jsp:include>

<div id="body">

<jsp:include flush="true" page="login\_body.jsp"></jsp:include>

</div>

<div style="margin-bottom: 30px;"></div>

<jsp:include flush="true" page="footer.jsp"></jsp:include>

</body>

</html>

Login\_body.jsp

<script>

function signin() {

$("#errormessage").slideUp();

$.ajax({

type: "post",

url: "JCAction?Signin=true",

data: $("#signinform").serialize(),

success: function (data) {

if (data == "failed")

$("#errormessage").slideDown();

else

window.location.href = 'index.jsp';

}

});

}

function signup() {

$("#errormessage").slideUp();

$.ajax({

type: "post",

url: "JCAction?Signup=true",

data: $("#signupform").serialize(),

success: function (data) {

if (data == "failed")

$("#errormessage").slideDown();

else

window.location.href = 'index.jsp';

}

});

}

$(document).ready(function () {

$("#signupform").hide();

$("#errormessage").hide();

$("#signinclick").click(function () {

$("#signupform").slideUp();

$("#signinform").slideDown();

});

$("#signupclick").click(function () {

$("#signinform").slideUp();

$("#signupform").slideDown();

});

});

</script>

<style>

body{

background-image: url('img/login-background.jpg');

overflow-x: hidden;

}

#body form, #body{

width: 450px;

margin: 0px auto;

padding: 0px;

}

#body form{

background: rgba(0,0,0,0.4);

padding: 7px;

margin-top: 40px;

border-radius: 2px;

color: white;

border:1px white solid;

box-shadow: 0 0 5px black;

transition: box-shadow 1s ease;

}

#body form:hover{

box-shadow: 0 0 15px black;

}

#body{

margin-top: 50px;

}

#body .heading{

font-size: 2em;

margin:0 auto;

}

#body ul{

margin-left: 70px

}

#body ul li{

display: inline;

padding: 10px 15px;

cursor: pointer;

background: rgba(0,0,0,0.3);

transition: background 0.3s ease;

margin-left: 20px;

border-radius: 2px;

color: white;

}

#body ul li:hover{

background: rgba(0,0,255,0.2);

color: black;

}

</style>

<div class='container alert alert-danger col-sm-12' id="errormessage">Error, Try Again!....</div>

<div id="body">

<ul id="tabs">

<li id="signinclick">SignIn</li>

<li id="signupclick">SignUp</li>

</ul>

<form id="signinform" class="form-horizontal">

<p class="heading">Login</p>

<div class="form-group">

<label for="signin\_username" class="control-label col-sm-3">Username</label>

<div class="col-sm-8">

<input type="text" name="signin\_username" id="signin\_username" placeholder="Enter Username" class="form-control" required>

</div>

</div>

<div class="form-group">

<label for="signin\_password" class="control-label col-sm-3">Password</label>

<div class="col-sm-8">

<input type="password" name="signin\_password" id="signin\_password" placeholder="Enter Password" class="form-control" required>

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-8">

<input type="button" onclick="signin()" class="btn btn-default" value="SignIn" />

</div>

</div>

</form>

<form id="signupform" class="form-horizontal">

<p class="heading">Sign Up</p>

<div class="form-group">

<label for="signup\_fullname" class="control-label col-sm-3">Full Name</label>

<div class="col-sm-8">

<input type="text" name="signup\_fullname" id="signup\_fullname" placeholder="Enter Full Name" class="form-control" required>

</div>

</div>

<div class="form-group">

<label for="signup\_username" class="control-label col-sm-3">Username</label>

<div class="col-sm-8">

<input type="text" name="signup\_username" id="signup\_username" placeholder="Enter Username" class="form-control" required>

</div>

</div>

<div class="form-group">

<label for="signup\_email" class="control-label col-sm-3">Email</label>

<div class="col-sm-8">

<input type="email" name="signup\_email" id="signup\_email" placeholder="Enter Email" class="form-control" required>

</div>

</div>

<div class="form-group">

<label for="signup\_dob" class="control-label col-sm-3">Date of Birth</label>

<div class="col-sm-8">

<input type="date" name="signup\_dob" id="signup\_dob" class="form-control">

</div>

</div>

<div class="form-group">

<label for="signup\_Password" class="control-label col-sm-3">Password</label>

<div class="col-sm-8">

<input type="password" name="signup\_Password" id="signup\_Password" placeholder="Enter Password" class="form-control" required>

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-8">

<input type="button" onclick="signup()" class="btn btn-default" value="SignUp" />

</div>

</div>

</form>

</div>

Admin\_modify.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>WS | Admin Modify</title>

<script src="js/jquery-1.11.3.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/script.js"></script>

<link rel="stylesheet" href="css/bootstrap.min.css" />

<link rel="stylesheet" href="css/styles.css" />

</head>

<body ng-app="myApp" ng-controller="booksCtrl">

<jsp:include flush="true" page="header.jsp"></jsp:include>

<jsp:include flush="true" page="admin\_modify\_body.jsp"></jsp:include>

<div style="margin-bottom: 30px;"></div>

<jsp:include flush="true" page="footer.jsp"></jsp:include>

</body>

</html>

Admin\_modify\_body.jsp

<%@page import="java.sql.DriverManager"%>

<%@page import="java.sql.Statement"%>

<%@page import="java.sql.Connection"%>

<%

if (session.getAttribute("name") == null) {

%>

<script>

window.location.href = "index.jsp";

</script>

<%

} else if (!session.getAttribute("name").toString().equals("William Scott")) {

%>

<script>

window.location.href = "index.jsp";

</script>

<%

}

%>

<div class='container alert alert-success' id="success">Successfully Updated / Inserted</div>

<div class='container alert alert-danger' id="failed">Error, Try Again!....</div>

<div class="panel col-sm-6 col-sm-offset-3 panel-default">

<div class="panel-heading panel-success">Add / Update Books</div>

<div class="panel-body">

<form class="form-horizontal" role="form" style="overflow: hidden;" action="Action" method="post" id="actionForm">

<div class="form-group">

<label class="control-label col-sm-3" for="id">Book ID</label>

<div class="col-sm-7">

<input type="text" class="form-control" id="id" name="id" placeholder="New Record Will be Created for New Books">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-3" for="name">Book Name</label>

<div class="col-sm-7">

<input type="text" class="form-control" id="name" name="name" placeholder="Old Books Will be Recreated With This Name">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-3" for="count">Book Count</label>

<div class="col-sm-7">

<input type="text" class="form-control" id="count" name="count" placeholder="This Will be Added to previous Count">

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-8 col-sm-10">

<a onclick="action()" class="btn btn-primary">Add / Update Details</a>

</div>

</div>

</form>

</div>

</div>

<script>

$(document).ready(function () {

$("#success").hide();

$("#failed").hide();

});

function action() {

$("#success").slideUp();

$("#failed").slideUp();

$.ajax({

type: "post",

url: "JMSAction",

data: $("#actionForm").serialize(),

success: function (data) {

if (data == "success")

$("#success").slideDown();

else

$("#failed").slideDown();

}

});

}

</script>

Logout.jsp

<%

session.invalidate();

response.sendRedirect("index.jsp");

%>

JCAction.java

import LibraryAction.JavaClass;

import LibraryAll.JavaClassGetAll;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class JCAction extends HttpServlet {

JavaClass jc;

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException, SQLException, Exception {

jc = new JavaClass();

response.setCharacterEncoding("utf-8");

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session = request.getSession();

if (request.getParameter("Signin") != null) {

String data;

String username = request.getParameter("signin\_username");

String password = request.getParameter("signin\_password");

data = jc.signin(username, password);

if (!"failed".equals(data)) {

session.setAttribute("name", data);

session.setAttribute("username", username);

} else {

out.write(data);

}

} else if (request.getParameter("Signup") != null) {

String name = request.getParameter("signup\_fullname");

String username = request.getParameter("signup\_username");

String email = request.getParameter("signup\_email");

String dob = request.getParameter("signup\_dob");

String pwd = request.getParameter("signup\_Password");

String data = jc.signup(name, username, email, dob, pwd);

if (!"failed".equals(data)) {

session.setAttribute("name", data);

session.setAttribute("username", username);

} else {

out.write(data);

}

} else if (request.getParameter("GetAll") != null) {

response.setContentType("application/json");

response.setCharacterEncoding("utf-8");

String data = JavaClassGetAll.GetAllData();

out.print(data);

System.out.println("a: " + data);

} else if (request.getParameter("Rent") != null) {

if (session.getAttribute("name") != null) {

if (request.getParameter("Rent") != null) {

jc.Rent(Integer.parseInt(request.getParameter("Rent")));

out.write("success");

}

}

}

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

try {

processRequest(request, response);

} catch (Exception ex) {

Logger.getLogger(JCAction.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

try {

processRequest(request, response);

} catch (Exception ex) {

Logger.getLogger(JCAction.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

JMSAction.java

import java.io.IOException;

import java.io.Serializable;

import java.util.HashMap;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.annotation.Resource;

import javax.jms.JMSContext;

import javax.jms.JMSException;

import javax.jms.ObjectMessage;

import javax.jms.Queue;

import javax.jms.QueueConnectionFactory;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "JMSAction", urlPatterns = {"/JMSAction"})

public class JMSAction extends HttpServlet {

@Resource(mappedName = "booksQueue")

private Queue booksQueue;

@Resource(mappedName = "booksConnectionFactory")

private QueueConnectionFactory context;

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

try {

if (!request.getParameter("name").equals("") && !request.getParameter("id").equals("") && !request.getParameter("count").equals("")) {

HashMap<String, String> hm = new HashMap();

Integer.parseInt(request.getParameter("id"));

Integer.parseInt(request.getParameter("count"));

hm.put("name", request.getParameter("name"));

hm.put("id", request.getParameter("id"));

hm.put("count", request.getParameter("count"));

sendJMSMessageToBooksQueue(hm);

response.getWriter().write("success");

} else {

response.getWriter().write("failed");

}

} catch (Exception e) {

response.getWriter().write("failed");

}

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

processRequest(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

processRequest(request, response);

}

private void sendJMSMessageToBooksQueue(HashMap<String, String> d) {

try {

System.out.println(d);

JMSContext ct = context.createContext();

ObjectMessage om = ct.createObjectMessage();

om.setObject((Serializable) d);

ct.createProducer().send(booksQueue, om);

} catch (JMSException ex) {

Logger.getLogger(JMSAction.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

AbstractFacade.java

package LibraryAction;

import java.util.List;

import javax.persistence.EntityManager;

public abstract class AbstractFacade<T> {

private Class<T> entityClass;

public AbstractFacade(Class<T> entityClass) {

this.entityClass = entityClass;

}

protected abstract EntityManager getEntityManager();

public void create(T entity) {

getEntityManager().persist(entity);

}

public void edit(T entity) {

getEntityManager().merge(entity);

}

public void remove(T entity) {

getEntityManager().remove(getEntityManager().merge(entity));

}

public T find(Object id) {

return getEntityManager().find(entityClass, id);

}

public List<T> findAll() {

javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();

cq.select(cq.from(entityClass));

return getEntityManager().createQuery(cq).getResultList();

}

public List<T> findRange(int[] range) {

javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();

cq.select(cq.from(entityClass));

javax.persistence.Query q = getEntityManager().createQuery(cq);

q.setMaxResults(range[1] - range[0] + 1);

q.setFirstResult(range[0]);

return q.getResultList();

}

public int count() {

javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();

javax.persistence.criteria.Root<T> rt = cq.from(entityClass);

cq.select(getEntityManager().getCriteriaBuilder().count(rt));

javax.persistence.Query q = getEntityManager().createQuery(cq);

return ((Long) q.getSingleResult()).intValue();

}

}

Books.java

package LibraryAction;

import java.io.Serializable;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import javax.xml.bind.annotation.XmlRootElement;

@Entity

@Table(name = "BOOKS")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Books.findAll", query = "SELECT b FROM Books b"),

@NamedQuery(name = "Books.findById", query = "SELECT b FROM Books b WHERE b.id = :id"),

@NamedQuery(name = "Books.findByName", query = "SELECT b FROM Books b WHERE b.name = :name"),

@NamedQuery(name = "Books.findByCount", query = "SELECT b FROM Books b WHERE b.count = :count")})

public class Books implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@Basic(optional = false)

@NotNull

@Column(name = "ID")

private Integer id;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 220)

@Column(name = "NAME")

private String name;

@Basic(optional = false)

@NotNull

@Column(name = "COUNT")

private int count;

public Books() {

}

public Books(Integer id) {

this.id = id;

}

public Books(Integer id, String name, int count) {

this.id = id;

this.name = name;

this.count = count;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

@Override

public int hashCode() {

int hash = 0;

hash += (id != null ? id.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof Books)) {

return false;

}

Books other = (Books) object;

if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {

return false;

}

return true;

}

@Override

public String toString() {

return "LibraryAction.Books[ id=" + id + " ]";

}

}

BooksFacade.java

package LibraryAction;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class BooksFacade extends AbstractFacade<Books> implements BooksFacadeLocal {

@PersistenceContext(unitName = "Exp\_8\_-\_Library\_Management\_using\_JMS-ejbPU")

private EntityManager em;

@Override

protected EntityManager getEntityManager() {

return em;

}

public BooksFacade() {

super(Books.class);

}

}

BooksFacadeLocal.java

package LibraryAction;

import java.util.List;

import javax.ejb.Local;

@Local

public interface BooksFacadeLocal {

void create(Books books);

void edit(Books books);

void remove(Books books);

Books find(Object id);

List<Books> findAll();

List<Books> findRange(int[] range);

int count();

}

RegisteredUsers.java

package LibraryAction;

import java.io.Serializable;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import javax.xml.bind.annotation.XmlRootElement;

@Entity

@Table(name = "REGISTERED\_USERS")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "RegisteredUsers.findAll", query = "SELECT r FROM RegisteredUsers r"),

@NamedQuery(name = "RegisteredUsers.findByName", query = "SELECT r FROM RegisteredUsers r WHERE r.name = :name"),

@NamedQuery(name = "RegisteredUsers.findByUsername", query = "SELECT r FROM RegisteredUsers r WHERE r.username = :username"),

@NamedQuery(name = "RegisteredUsers.findByEmail", query = "SELECT r FROM RegisteredUsers r WHERE r.email = :email"),

@NamedQuery(name = "RegisteredUsers.findByDob", query = "SELECT r FROM RegisteredUsers r WHERE r.dob = :dob"),

@NamedQuery(name = "RegisteredUsers.findByPwd", query = "SELECT r FROM RegisteredUsers r WHERE r.pwd = :pwd")})

public class RegisteredUsers implements Serializable {

private static final long serialVersionUID = 1L;

@Size(max = 70)

@Column(name = "NAME")

private String name;

@Id

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 70)

@Column(name = "USERNAME")

private String username;

// @Pattern(regexp="[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\\.[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+)\*@(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?", message="Invalid email")//if the field contains email address consider using this annotation to enforce field validation

@Size(max = 200)

@Column(name = "EMAIL")

private String email;

@Size(max = 30)

@Column(name = "DOB")

private String dob;

@Size(max = 300)

@Column(name = "PWD")

private String pwd;

public RegisteredUsers() {

}

public RegisteredUsers(String username) {

this.username = username;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getDob() {

return dob;

}

public void setDob(String dob) {

this.dob = dob;

}

public String getPwd() {

return pwd;

}

public void setPwd(String pwd) {

this.pwd = pwd;

}

@Override

public int hashCode() {

int hash = 0;

hash += (username != null ? username.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof RegisteredUsers)) {

return false;

}

RegisteredUsers other = (RegisteredUsers) object;

if ((this.username == null && other.username != null) || (this.username != null && !this.username.equals(other.username))) {

return false;

}

return true;

}

@Override

public String toString() {

return "LibraryAction.RegisteredUsers[ username=" + username + " ]";

}

}

RegisteredUsersFacade.java

package LibraryAction;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class RegisteredUsersFacade extends AbstractFacade<RegisteredUsers> implements RegisteredUsersFacadeLocal {

@PersistenceContext(unitName = "Exp\_8\_-\_Library\_Management\_using\_JMS-ejbPU")

private EntityManager em;

@Override

protected EntityManager getEntityManager() {

return em;

}

public RegisteredUsersFacade() {

super(RegisteredUsers.class);

}

}

RegisteredUsersFacadeLocal.java

package LibraryAction;

import java.util.List;

import javax.ejb.Local;

@Local

public interface RegisteredUsersFacadeLocal {

void create(RegisteredUsers registeredUsers);

void edit(RegisteredUsers registeredUsers);

void remove(RegisteredUsers registeredUsers);

RegisteredUsers find(Object id);

List<RegisteredUsers> findAll();

List<RegisteredUsers> findRange(int[] range);

int count();

}

JavaClass.java

package LibraryAction;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class JavaClass {

BooksFacadeLocal booksFacade = lookupBooksFacadeLocal();

RegisteredUsersFacadeLocal registeredUsersFacade = lookupRegisteredUsersFacadeLocal();

public String signin(String username, String password) {

if (registeredUsersFacade.find(username) == null) {

return "failed";

} else {

if (registeredUsersFacade.find(username).getPwd().equals(password)) {

return registeredUsersFacade.find(username).getName();

} else {

return "failed";

}

}

}

public String signup(String name, String username, String email, String dob, String password) {

RegisteredUsers a = new RegisteredUsers();

a.setName(name);

a.setUsername(username);

a.setEmail(email);

a.setDob(dob);

a.setPwd(password);

try {

registeredUsersFacade.create(a);

} catch (Exception e) {

return "failed";

}

return name;

}

public String Search(int bookid) {

List a = booksFacade.findAll();

System.out.println("here: " + a.get(0));

if (booksFacade.find(bookid) == null) {

return "failed";

} else {

String name = booksFacade.find(bookid).getName();

int id = booksFacade.find(bookid).getId();

int count = booksFacade.find(bookid).getCount();

return "[{\"name\":\"" + name + "\", \"id\":\"" + id + "\", \"count\":\"" + count + "\"}]";

}

}

public void Rent(int bookid) {

Books a = booksFacade.find(bookid);

System.out.println(a.getCount());

a.setCount(a.getCount() - 1);

booksFacade.edit(a);

}

private RegisteredUsersFacadeLocal lookupRegisteredUsersFacadeLocal() {

try {

Context c = new InitialContext();

return (RegisteredUsersFacadeLocal) c.lookup("java:global/Exp\_8\_-\_Library\_Management\_using\_JMS/Exp\_8\_-\_Library\_Management\_using\_JMS-ejb/RegisteredUsersFacade!LibraryAction.RegisteredUsersFacadeLocal");

} catch (NamingException ne) {

Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);

throw new RuntimeException(ne);

}

}

private BooksFacadeLocal lookupBooksFacadeLocal() {

try {

Context c = new InitialContext();

return (BooksFacadeLocal) c.lookup("java:global/Exp\_8\_-\_Library\_Management\_using\_JMS/Exp\_8\_-\_Library\_Management\_using\_JMS-ejb/BooksFacade!LibraryAction.BooksFacadeLocal");

} catch (NamingException ne) {

Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);

throw new RuntimeException(ne);

}

}

}

MessageBean.java

package LibraryAction;

import java.util.HashMap;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.ejb.ActivationConfigProperty;

import javax.ejb.EJB;

import javax.ejb.MessageDriven;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.ObjectMessage;

@MessageDriven(activationConfig = {

@ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "booksQueue"),

@ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue")

})

public class MessageBean implements MessageListener {

@EJB

private BooksFacadeLocal booksFacade;

public MessageBean() {

}

@Override

public void onMessage(Message message) {

if (message instanceof ObjectMessage) {

try {

ObjectMessage tm = (ObjectMessage) message;

HashMap<String, String> a = (HashMap<String, String>) tm.getObject();

System.out.println("hm: " + a);

String name = a.get("name");

int id = Integer.parseInt(a.get("id"));

int count = Integer.parseInt(a.get("count"));

if (booksFacade.find(id) != null) {

Books b = booksFacade.find(id);

b.setName(name);

b.setCount(count + b.getCount());

booksFacade.edit(b);

} else {

Books b = new Books();

b.setName(name);

b.setId(id);

b.setCount(count);

booksFacade.create(b);

}

} catch (JMSException ex) {

Logger.getLogger(MessageBean.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

JavaClassGetAll.java

package LibraryAll;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class JavaClassGetAll {

public static String GetAllData() throws ClassNotFoundException, SQLException, Exception {

Class.forName("org.apache.derby.jdbc.ClientDriver");

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/j2eeexp", "ws", "6477");

String query = "select \* from BOOKS where count>0";

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(query);

String out = "[";

while (rs.next()) {

out += "{\"id\":\"" + rs.getString(1) + "\", \"name\":\"" + rs.getString(2) + "\", \"count\":\"" + rs.getString(3) + "\"}, ";

}

out = out.substring(0, out.length() - 2);

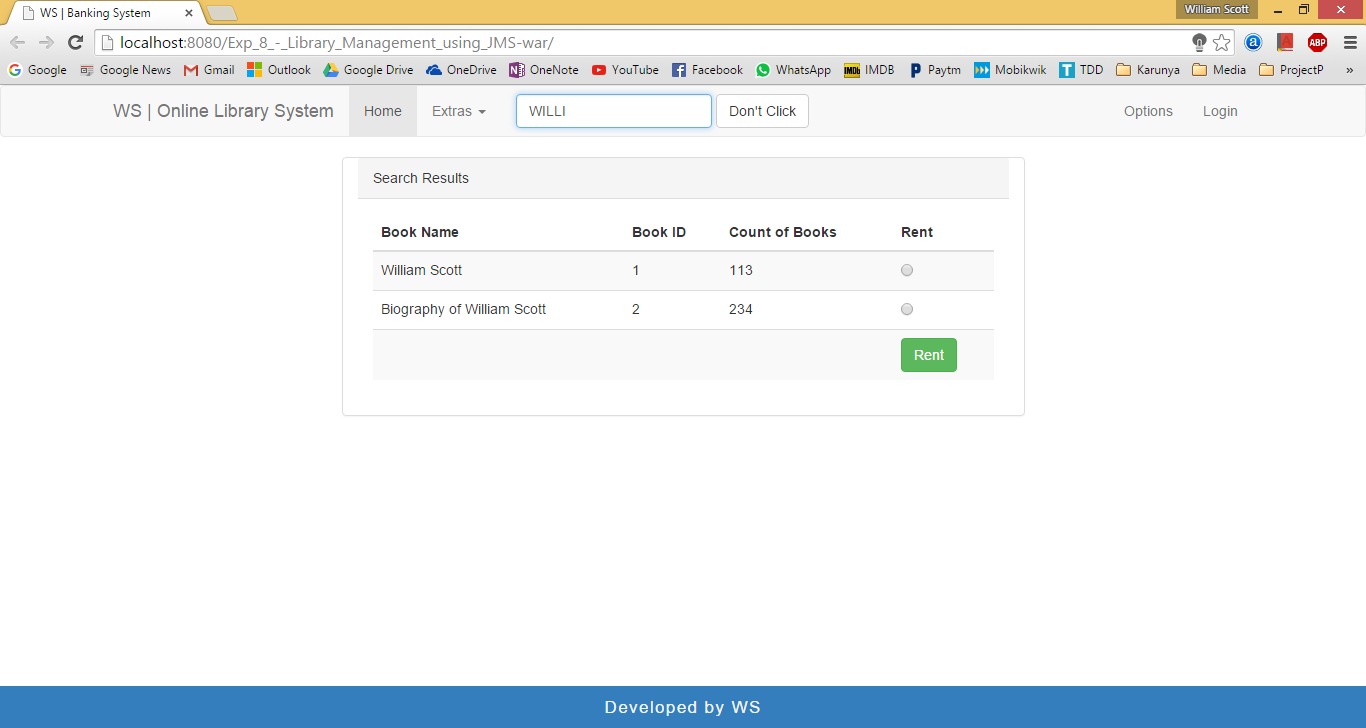
out += "]";

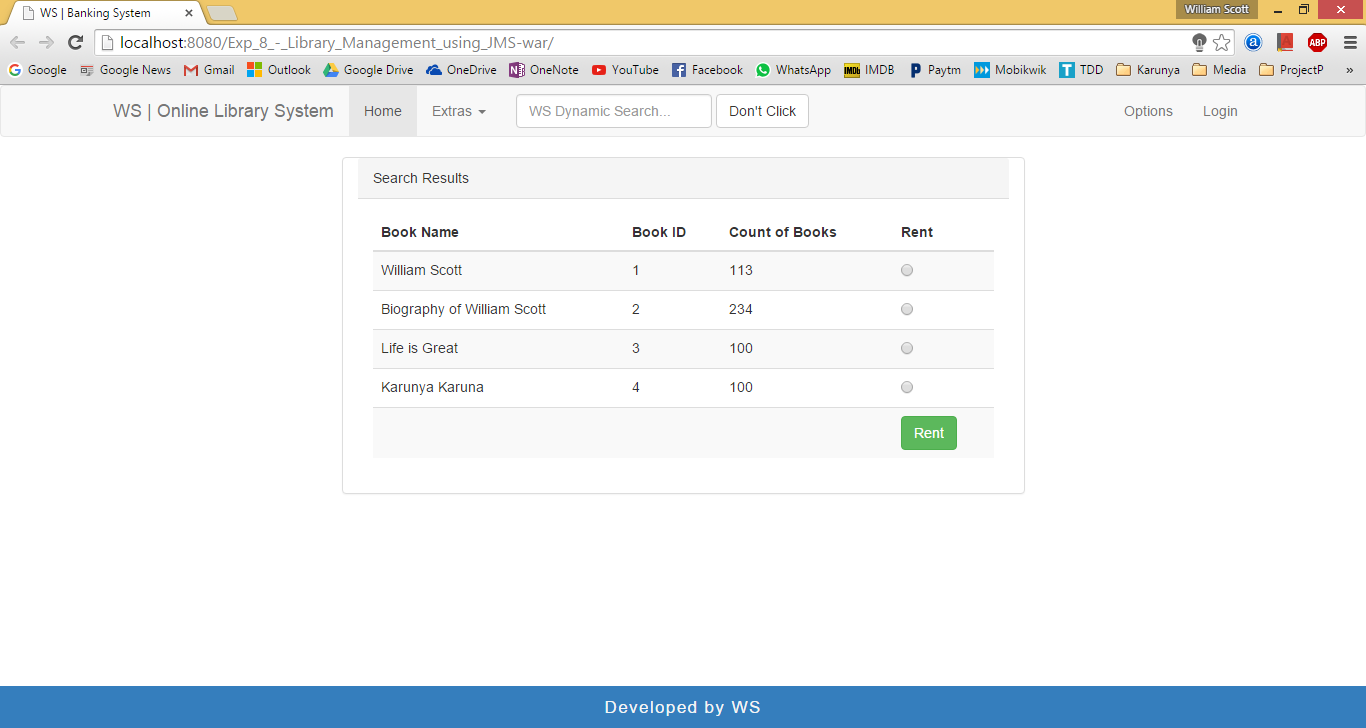
return out;

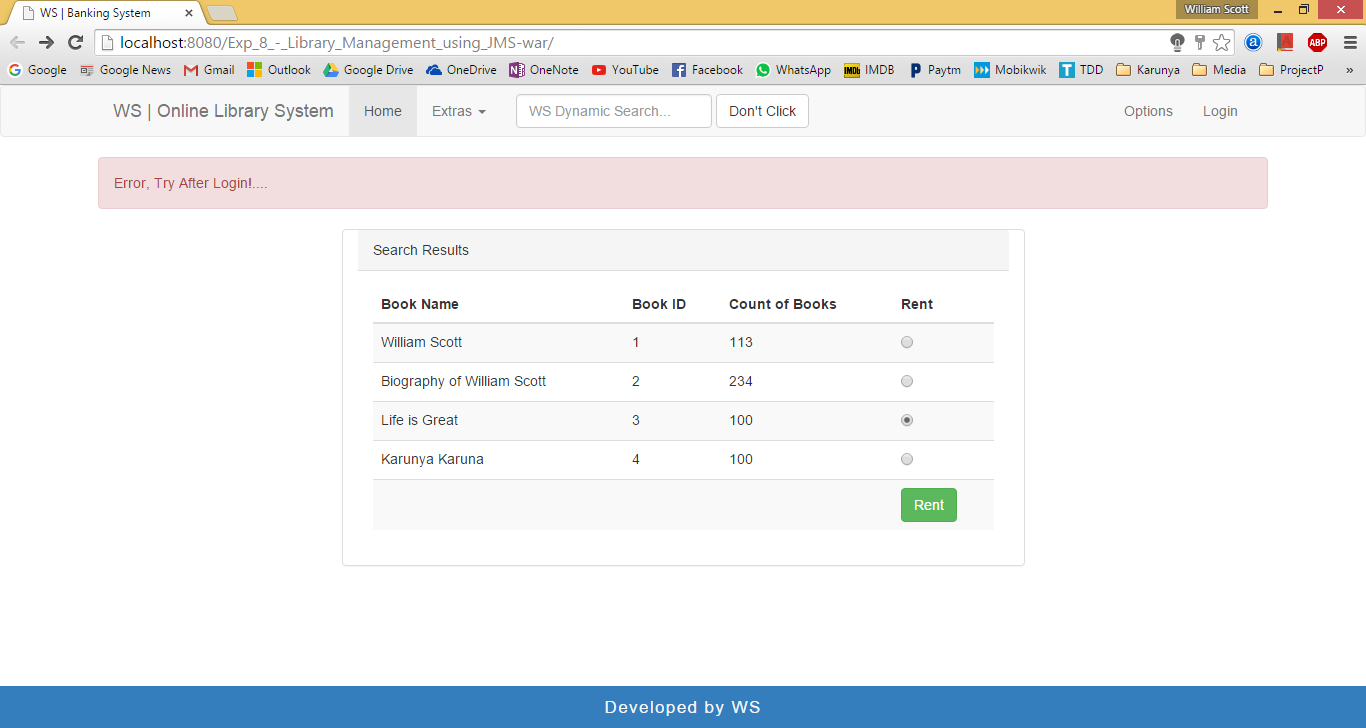
}

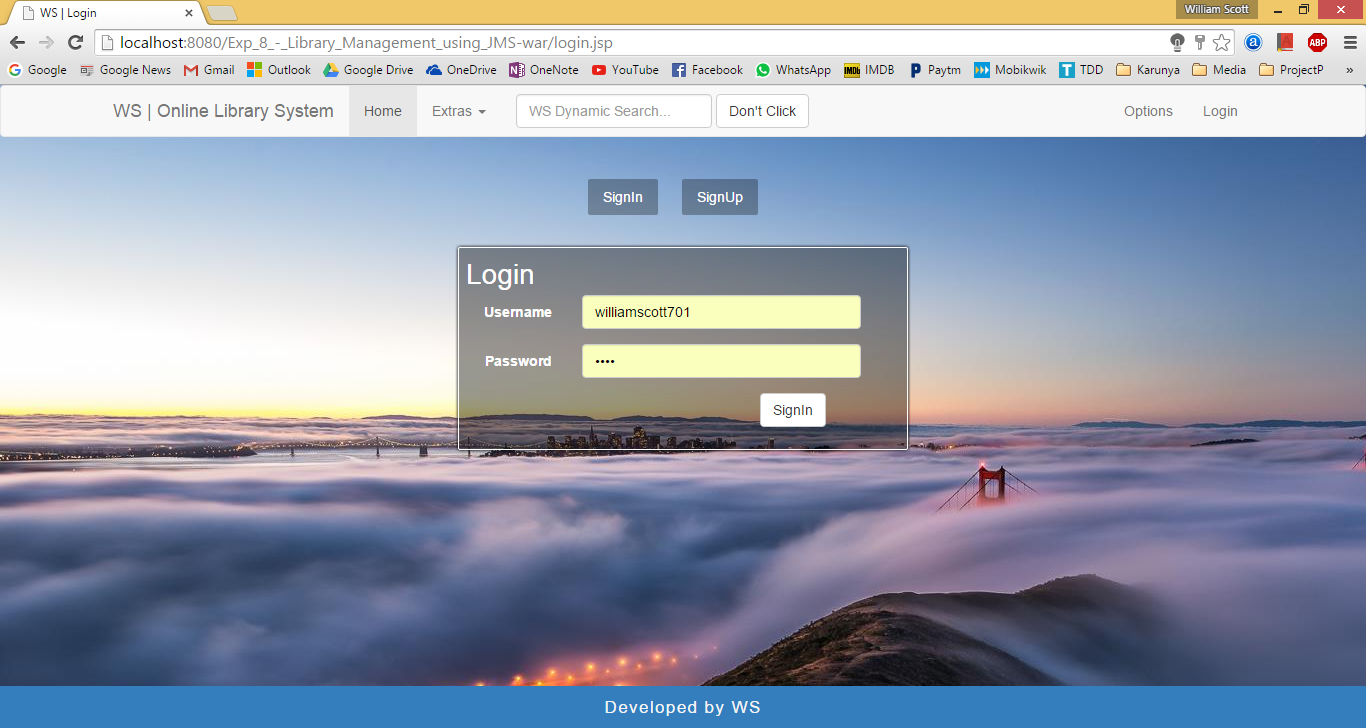
}

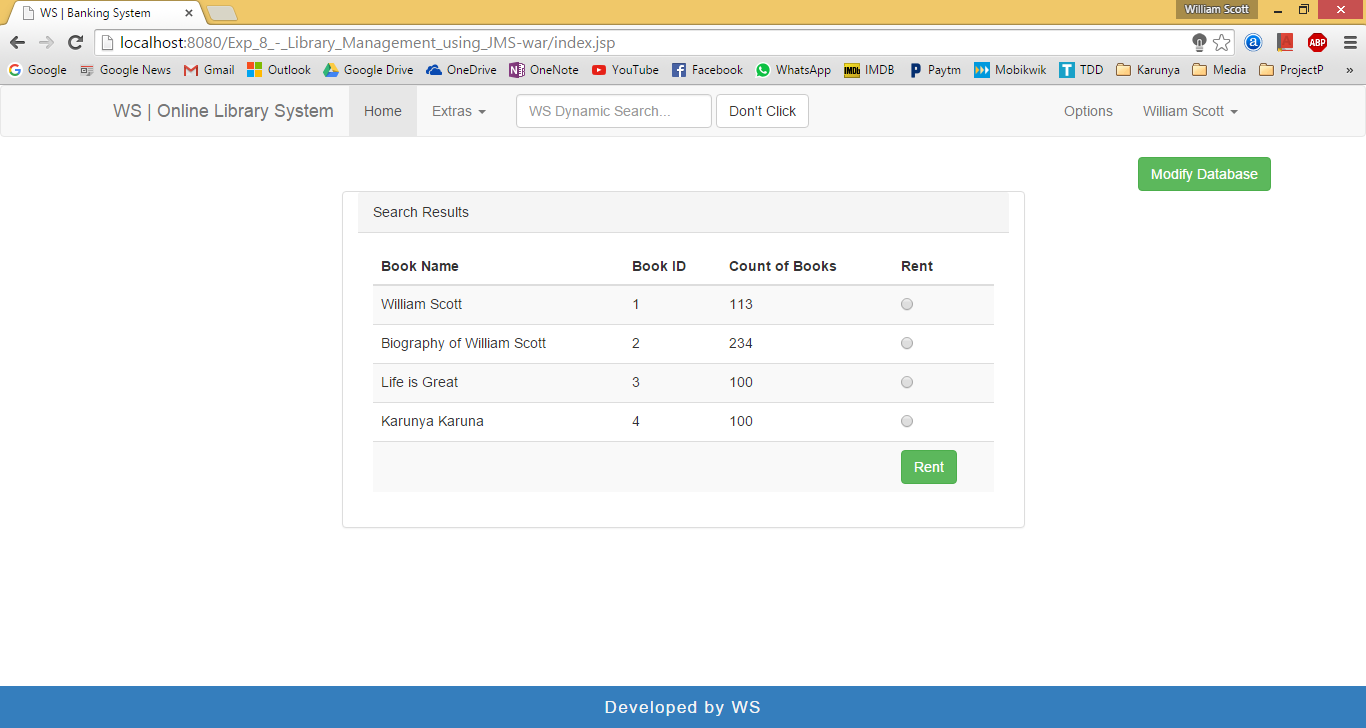
**Output**

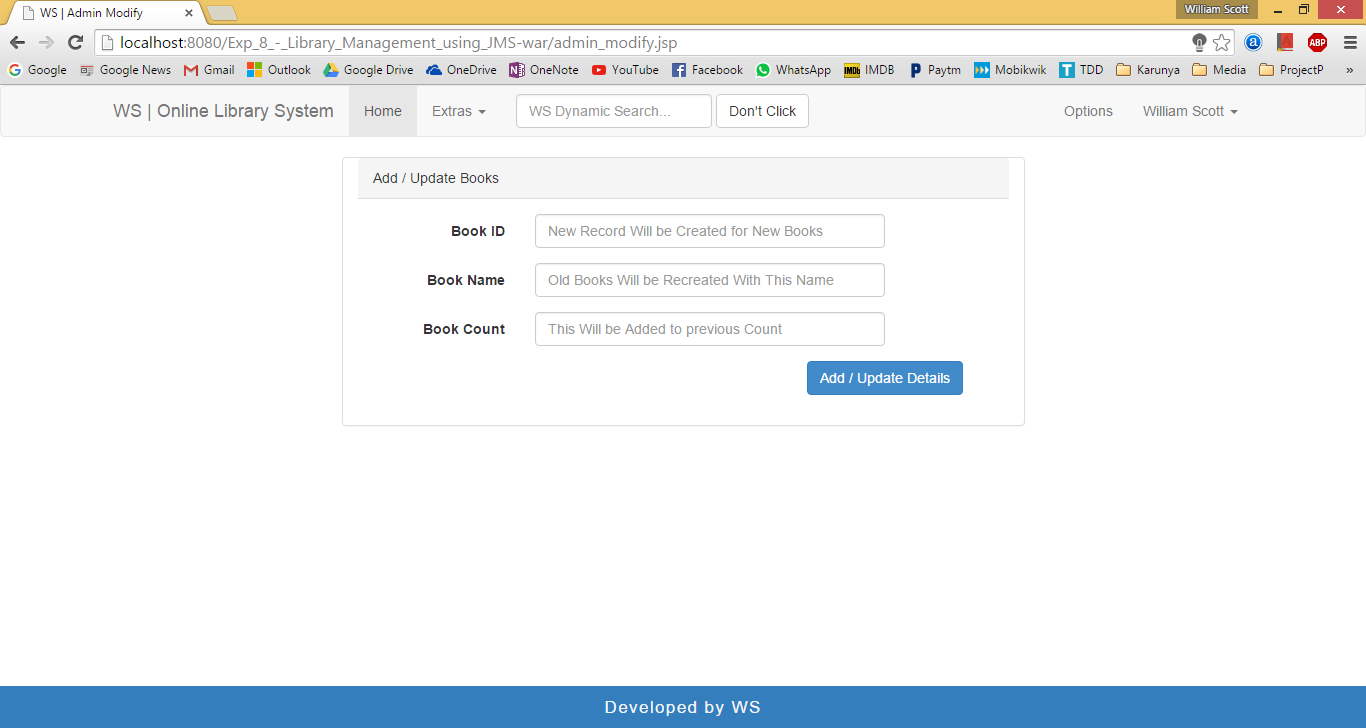


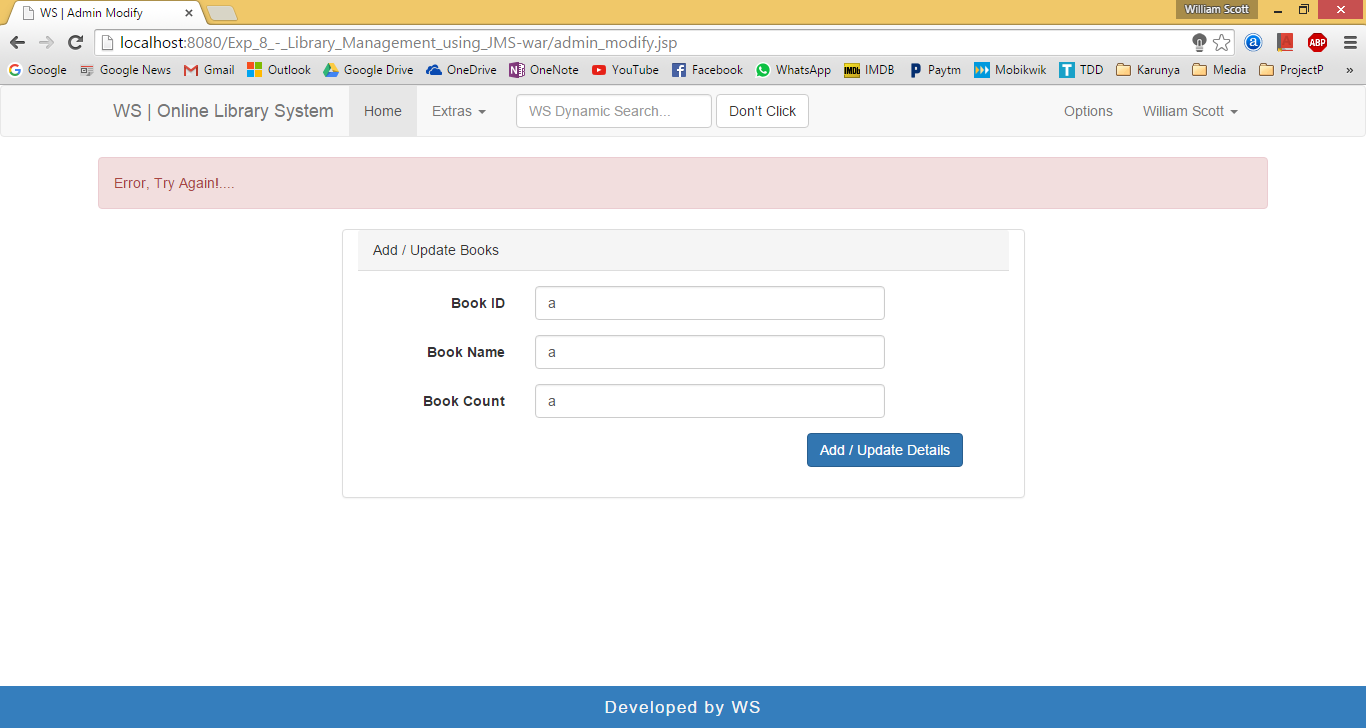


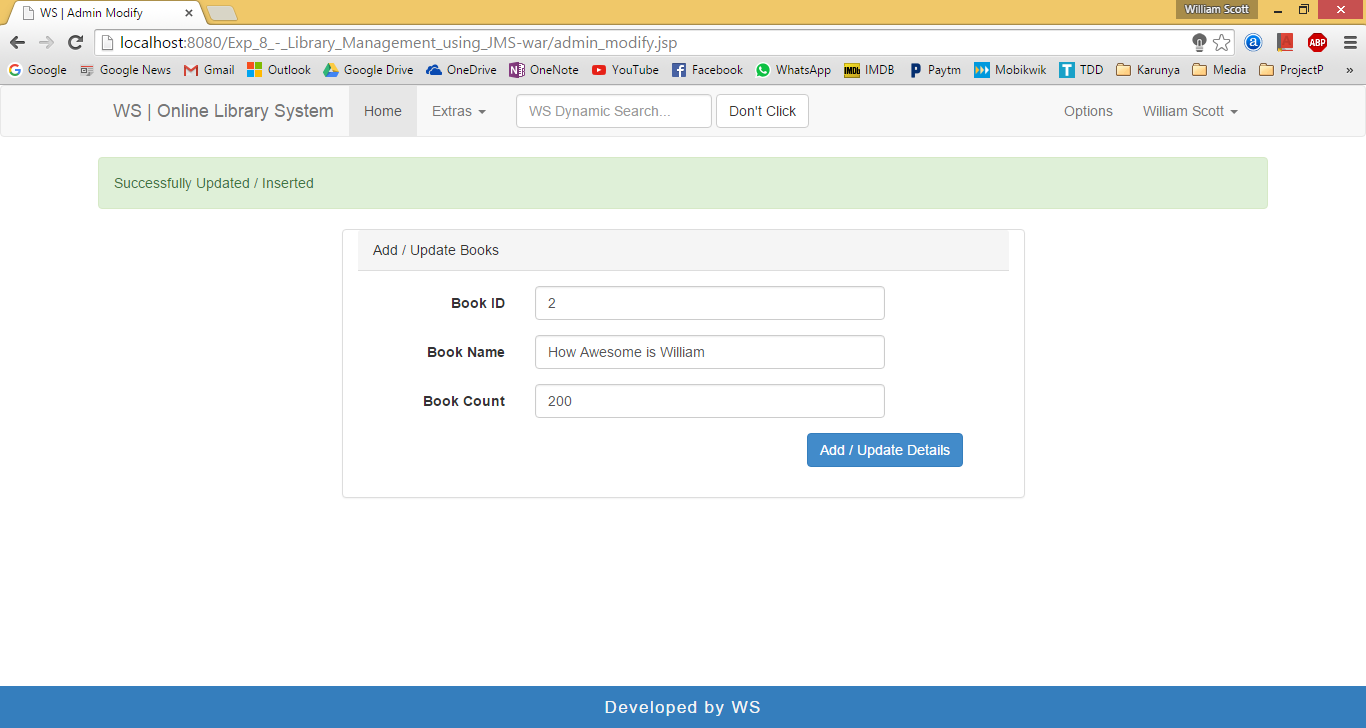


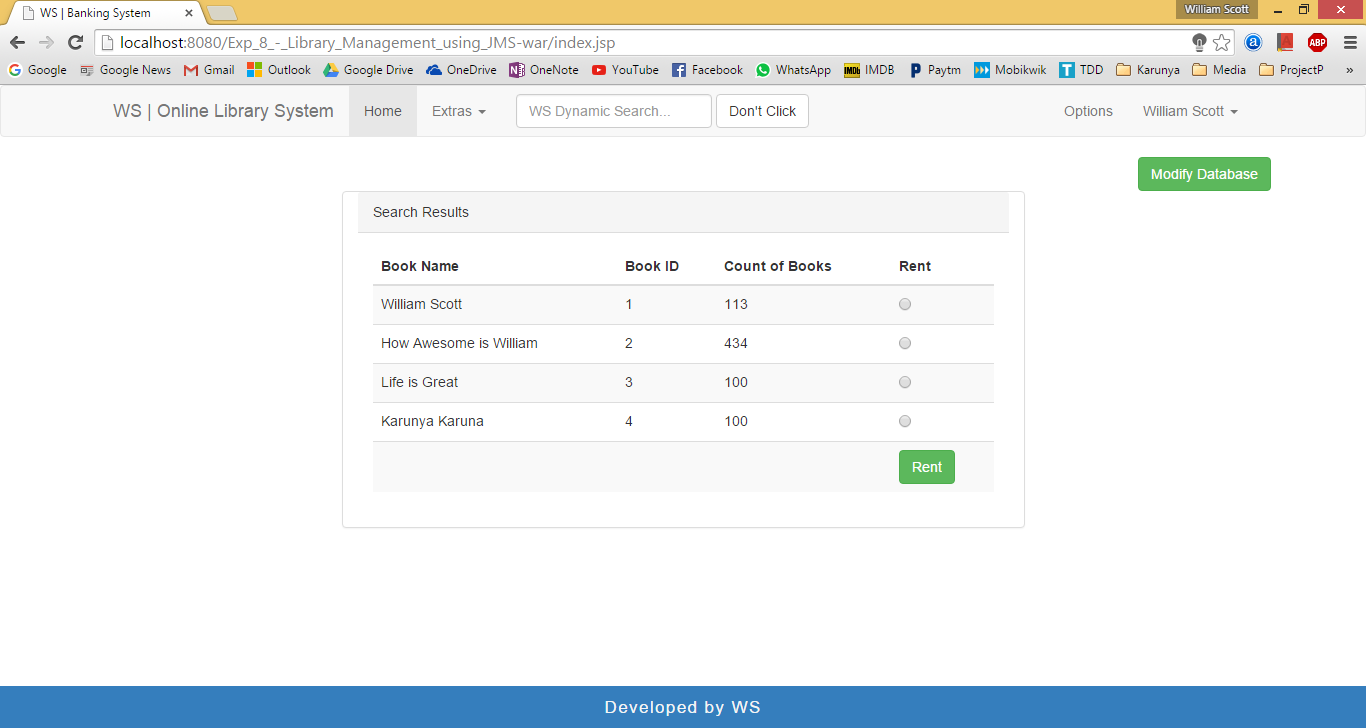


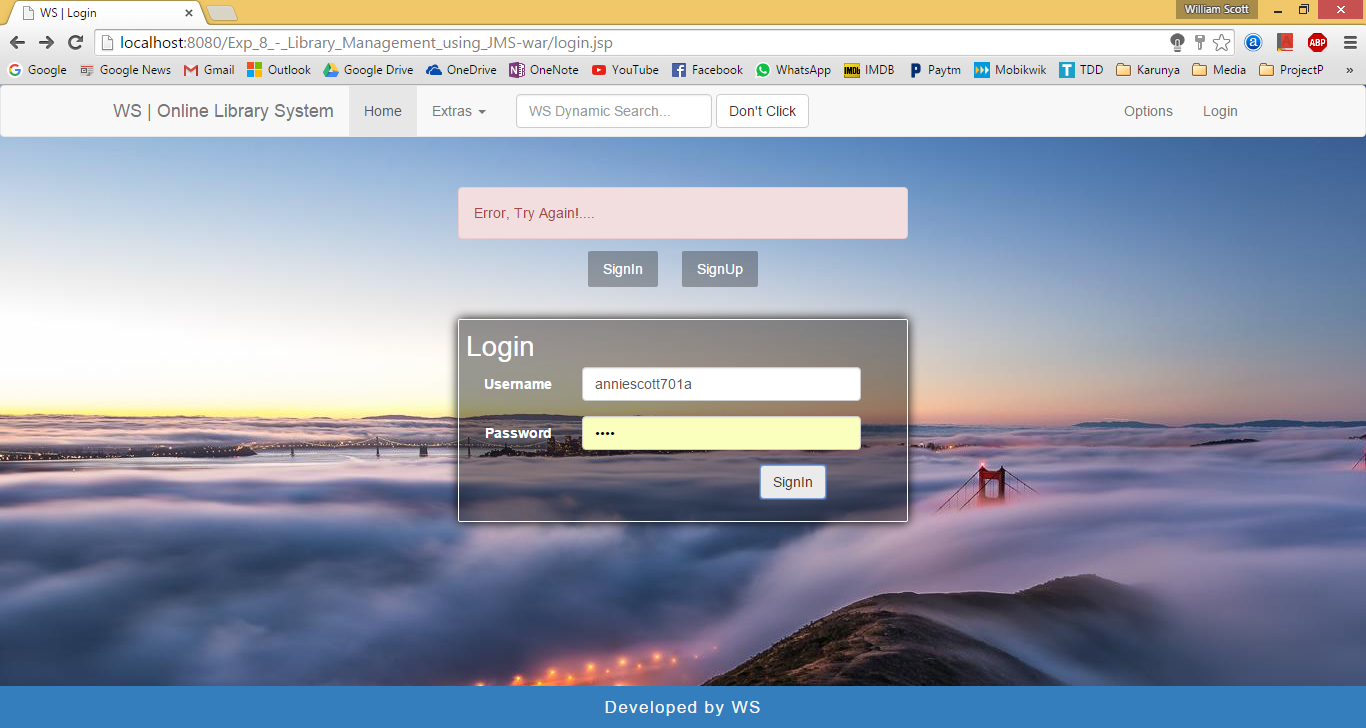


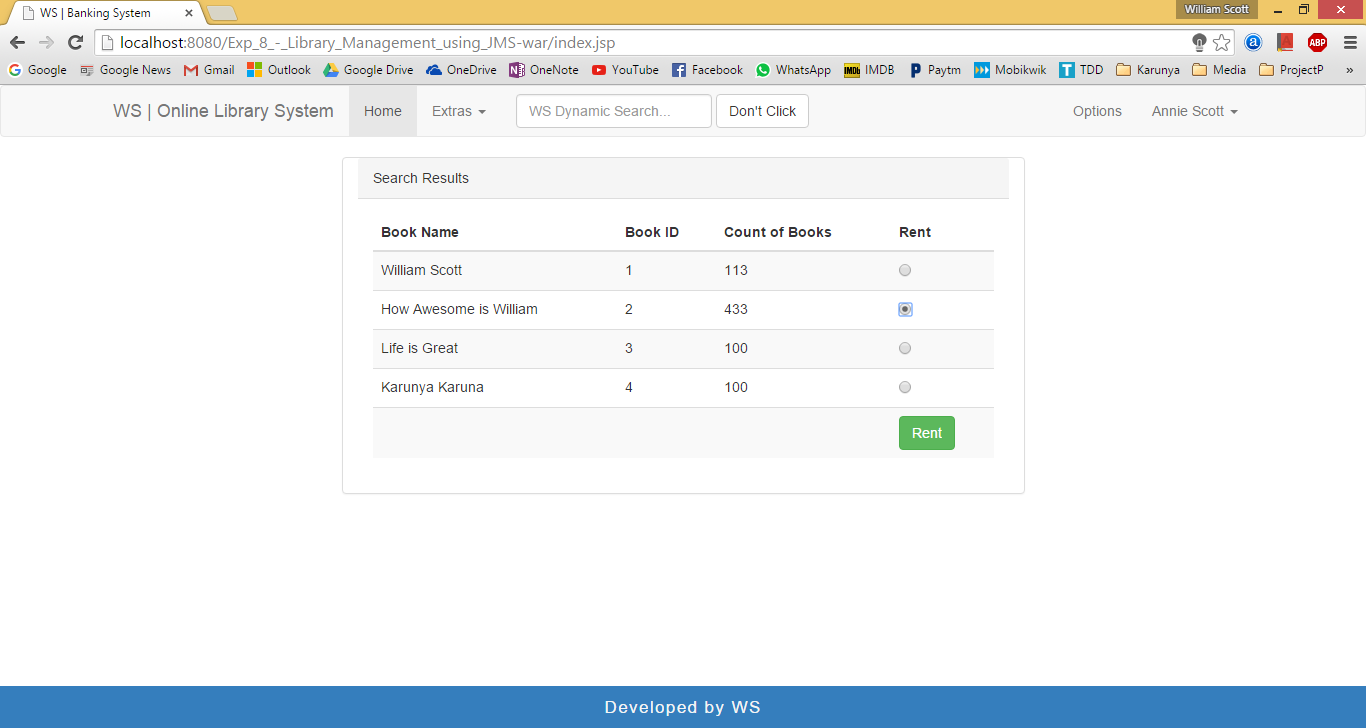












**Result**

Thus, the enterprise Application for Online Library using JMS was successfully developed and deployed

[Signature of the Staff In-charge]

[Name of the Staff In – charge]