**Source Code**

**Model**

```
package in.co.college.att.mgt.model;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import org.apache.log4j.Logger;
import in.co.college.att.mgt.bean.UserBean;
import in.co.college.att.mgt.exception.ApplicationException;
import in.co.college.att.mgt.exception.DatabaseException;
import in.co.college.att.mgt.exception.DuplicateRecordException;
import in.co.college.att.mgt.exception.RecordNotFoundException;
import in.co.college.att.mgt.util.EmailBuilder;
import in.co.college.att.mgt.util.EmailMessage;
import in.co.college.att.mgt.util.EmailUtility;
import in.co.college.att.mgt.util.JDBCDataSource;
public class UserModel {
private static Logger log = Logger.getLogger(UserModel.class);
public Integer nextPK() throws DatabaseException {
log.debug("Model nextPK Started");
Connection conn = null;
int pk = 0;
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM A_USER");
ResultSet rs = pstmt.executeQuery();
while (rs.next()) {
```

```
pk = rs.getInt(1);
}
rs.close();
} catch (Exception e) {
log.error("Database Exception..", e);
throw new DatabaseException("Exception : Exception in getting PK");
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model nextPK End");
return pk + 1; 36

}
/**
* Add a User
*
* @param bean
* @throws DatabaseException
*
*/
public long add(UserBean bean) throws ApplicationException, DuplicateRecordException {
Connection conn = null;
int pk = 0;
UserBean existbean = findByLogin(bean.getLogin());
if (existbean != null) {
throw new DuplicateRecordException("Login Id already exists");
}
try {
conn = JDBCDataSource.getConnection();
pk = nextPK();
// Get auto-generated next primary key
```

```
System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO A_USER
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setString(2, bean.getFirstName());

pstmt.setString(3, bean.getLastName());

pstmt.setString(4, bean.getLogin());

pstmt.setString(5, bean.getPassword());

pstmt.setDate(6, new java.sql.Date(bean.getDob().getTime()));

pstmt.setString(7, bean.getMobileNo());

pstmt.setLong(8, bean.getRoleId());

pstmt.setString(9, bean.getGender());

pstmt.setString(10, bean.getProfilePic());

pstmt.setString(11, bean.getCreatedBy());

pstmt.setString(12, bean.getModifiedBy());

pstmt.setTimestamp(13, bean.getCreatedDatetime());

pstmt.setTimestamp(14, bean.getModifiedDatetime());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

ex.printStackTrace();

throw new ApplicationException("Exception : 37


add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add User");
```

```
} finally {
JDBCDataSource.closeConnection(conn);
}
return pk;
}
public void delete(UserBean bean) throws ApplicationException {
Connection conn = null;
try {
conn = JDBCDataSource.getConnection();
conn.setAutoCommit(false); // Begin transaction
PreparedStatement pstmt = conn.prepareStatement("DELETE FROM A_USER WHERE ID=?");
pstmt.setLong(1, bean.getId());
pstmt.executeUpdate();
conn.commit(); // End transaction
pstmt.close();
} catch (Exception e) {
try {
conn.rollback();
} catch (Exception ex) {
throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());
}
throw new ApplicationException("Exception : Exception in delete User");
} finally {
JDBCDataSource.closeConnection(conn);
}
}
public UserBean findByLogin(String login) throws ApplicationException {
log.debug("Model findByLogin Started");
StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE LOGIN=?");
UserBean bean = null;
Connection conn = null;
```

```
System.out.println("sql" + sql);
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql.toString());
pstmt.setString(1, login);
ResultSet rs = pstmt.executeQuery(); 38

while (rs.next()) {
bean = new UserBean();
bean.setId(rs.getLong(1));
bean.setFirstName(rs.getString(2));
bean.setLastName(rs.getString(3));
bean.setLogin(rs.getString(4));
bean.setPassword(rs.getString(5));
bean.setDob(rs.getDate(6));
bean.setMobileNo(rs.getString(7));
bean.setRoleId(rs.getLong(8));
bean.setGender(rs.getString(9));
bean.setProfilePic(rs.getString(10));
bean.setCreatedBy(rs.getString(11));
bean.setModifiedBy(rs.getString(12));
bean.setCreatedDatetime(rs.getTimestamp(13));
bean.setModifiedDatetime(rs.getTimestamp(14));
}
rs.close();
} catch (Exception e) {
e.printStackTrace();
log.error("Database Exception..", e);
throw new ApplicationException("Exception : Exception in getting User by login");
} finally {
JDBCDataSource.closeConnection(conn);
```

```
}
log.debug("Model findByLogin End");
return bean;
}
public UserBean findByPK(long pk) throws ApplicationException {
log.debug("Model findByPK Started");
StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE ID=?");
UserBean bean = null;
Connection conn = null;
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql.toString());
pstmt.setLong(1, pk);
ResultSet rs = pstmt.executeQuery();
while (rs.next()) {
bean = new UserBean();
bean.setId(rs.getLong(1));
bean.setFirstName(rs.getString(2));
bean.setLastName(rs.getString(3));
bean.setLogin(rs.getString(4));
bean.setPassword(rs.getString(5));
bean.setDob(rs.getDate(6));
bean.setMobileNo(rs.getString(7));
bean.setRoleId(rs.getLong(8)); 39

bean.setGender(rs.getString(9));
bean.setProfilePic(rs.getString(10));
bean.setCreatedBy(rs.getString(11));
bean.setModifiedBy(rs.getString(12));
bean.setCreatedDatetime(rs.getTimestamp(13));
bean.setModifiedDatetime(rs.getTimestamp(14));
```

```
}
rs.close();
} catch (Exception e) {
e.printStackTrace();
log.error("Database Exception..", e);
throw new ApplicationException("Exception : Exception in getting User by pk");
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model findByPK End");
return bean;
}
public void update(UserBean bean) throws ApplicationException, DuplicateRecordException {
log.debug("Model update Started");
Connection conn = null;
UserBean beanExist = findByLogin(bean.getLogin());
// Check if updated LoginId already exist
if (beanExist != null && !(beanExist.getId() == bean.getId())) {
throw new DuplicateRecordException("LoginId is already exist");
}
try {
conn = JDBCDataSource.getConnection();
conn.setAutoCommit(false); // Begin transaction
PreparedStatement pstmt = conn.prepareStatement(
"UPDATE A_USER SET
FIRSTNAME=?,LASTNAME=?,LOGIN=?,PASSWORD=?,DOB=?,MOBILENO=?,ROLEID=?,"
+ "GENDER=?,ProfilePic=?,"
+ "CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");
pstmt.setString(1, bean.getFirstName());
pstmt.setString(2, bean.getLastName());
pstmt.setString(3, bean.getLogin());
```

```
pstmt.setString(4, bean.getPassword());
pstmt.setDate(5, new java.sql.Date(bean.getDob().getTime()) );
pstmt.setString(6, bean.getMobileNo());
pstmt.setLong(7, bean.getRoleId());
pstmt.setString(8, bean.getGender()); 40

pstmt.setString(9,bean.getProfilePic());
pstmt.setString(10, bean.getCreatedBy());
pstmt.setString(11, bean.getModifiedBy());
pstmt.setTimestamp(12, bean.getCreatedDatetime());
pstmt.setTimestamp(13, bean.getModifiedDatetime());
pstmt.setLong(14, bean.getId());
pstmt.executeUpdate();
conn.commit(); // End transaction
pstmt.close();
} catch (Exception e) {
e.printStackTrace();
log.error("Database Exception..", e);
try {
conn.rollback();
} catch (Exception ex) {
throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());
}
throw new ApplicationException("Exception in updating User ");
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model update End");
}
public List search(UserBean bean) throws ApplicationException {
return search(bean, 0, 0);
```

```
}
public List search(UserBean bean, int pageNo, int pageSize) throws ApplicationException {
log.debug("Model search Started");
StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE 1=1");
if (bean != null) {
if (bean.getId() > 0) {
sql.append(" AND id = " + bean.getId());
}
if (bean.getFirstName() != null && bean.getFirstName().length() > 0) {
sql.append(" AND FIRSTNAME like '" + bean.getFirstName() + "%'");
}
if (bean.getLastName() != null && bean.getLastName().length() > 0) {
sql.append(" AND LASTNAME like '" + bean.getLastName() + "%'");
}
if (bean.getLogin() != null && bean.getLogin().length() > 0) { 41

sql.append(" AND LOGIN like '" + bean.getLogin() + "%'");
}
if (bean.getPassword() != null && bean.getPassword().length() > 0) {
sql.append(" AND PASSWORD like '" + bean.getPassword() + "%'");
}
if (bean.getDob() != null && bean.getDob().getDate() > 0) {
sql.append(" AND DOB = " + bean.getGender());
}
if (bean.getMobileNo() != null && bean.getMobileNo().length() > 0) {
sql.append(" AND MOBILENO = " + bean.getMobileNo());
}
if (bean.getRoleId() > 0) {
sql.append(" AND ROLEID = " + bean.getRoleId());
}
}
```

```
// if page size is greater than zero then apply pagination
if (pageSize > 0) {
// Calculate start record index
pageNo = (pageNo - 1) * pageSize;
sql.append(" Limit " + pageNo + ", " + pageSize);
// sql.append(" limit " + pageNo + "," + pageSize);
}
System.out.println("user model search :"+sql);
ArrayList list = new ArrayList();
Connection conn = null;
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql.toString());
ResultSet rs = pstmt.executeQuery();
while (rs.next()) {
bean = new UserBean();
bean.setId(rs.getLong(1));
bean.setFirstName(rs.getString(2));
bean.setLastName(rs.getString(3));
bean.setLogin(rs.getString(4));
bean.setPassword(rs.getString(5));
bean.setDob(rs.getDate(6));
bean.setMobileNo(rs.getString(7));
bean.setRoleId(rs.getLong(8));
bean.setGender(rs.getString(9));
bean.setProfilePic(rs.getString(10));
bean.setCreatedBy(rs.getString(11)); 42

bean.setModifiedBy(rs.getString(12));
bean.setCreatedDatetime(rs.getTimestamp(13));
bean.setModifiedDatetime(rs.getTimestamp(14));
```

```
list.add(bean);
}
rs.close();
} catch (Exception e) {
log.error("Database Exception..", e);
throw new ApplicationException("Exception : Exception in search user");
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model search End");
return list;
}
public List list() throws ApplicationException {
return list(0, 0);
}
public List list(int pageNo, int pageSize) throws ApplicationException {
log.debug("Model list Started");
ArrayList list = new ArrayList();
StringBuffer sql = new StringBuffer("select * from A_USER");
// if page size is greater than zero then apply pagination
if (pageSize > 0) {
// Calculate start record index
pageNo = (pageNo - 1) * pageSize;
sql.append(" limit " + pageNo + "," + pageSize);
}
System.out.println("sql in list user :"+sql);
Connection conn = null;
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql.toString());
ResultSet rs = pstmt.executeQuery();
```

```
while (rs.next()) {
UserBean bean = new UserBean();
bean.setId(rs.getLong(1));
bean.setFirstName(rs.getString(2));
bean.setLastName(rs.getString(3));
bean.setLogin(rs.getString(4));
bean.setPassword(rs.getString(5)); 43

bean.setDob(rs.getDate(6));
bean.setMobileNo(rs.getString(7));
bean.setRoleId(rs.getLong(8));
bean.setGender(rs.getString(9));
bean.setProfilePic(rs.getString(10));
bean.setCreatedBy(rs.getString(11));
bean.setModifiedBy(rs.getString(12));
bean.setCreatedDatetime(rs.getTimestamp(13));
bean.setModifiedDatetime(rs.getTimestamp(14));
list.add(bean);
}
rs.close();
} catch (Exception e) {
log.error("Database Exception..", e);
throw new ApplicationException("Exception : Exception in getting list of users");
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model list End");
return list;
}
public UserBean authenticate(String login, String password) throws ApplicationException {
log.debug("Model authenticate Started");
```

```
StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE LOGIN = ? AND PASSWORD =
?");
UserBean bean = null;
Connection conn = null;
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql.toString());
pstmt.setString(1, login);
pstmt.setString(2, password);
ResultSet rs = pstmt.executeQuery();
while (rs.next()) {
bean = new UserBean();
bean.setId(rs.getLong(1));
bean.setFirstName(rs.getString(2));
bean.setLastName(rs.getString(3));
bean.setLogin(rs.getString(4));
bean.setPassword(rs.getString(5));
bean.setDob(rs.getDate(6));
bean.setMobileNo(rs.getString(7));
bean.setRoleId(rs.getLong(8));
bean.setGender(rs.getString(9));
bean.setProfilePic(rs.getString(10));
bean.setCreatedBy(rs.getString(11)); 44

bean.setModifiedBy(rs.getString(12));
bean.setCreatedDatetime(rs.getTimestamp(13));
bean.setModifiedDatetime(rs.getTimestamp(14));
}
} catch (Exception e) {
log.error("Database Exception..", e);
throw new ApplicationException("Exception : Exception in get roles");
```

```
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model authenticate End");
return bean;
}
public List getRoles(UserBean bean) throws ApplicationException {
log.debug("Model get roles Started");
StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE role_Id=?");
Connection conn = null;
List list = new ArrayList();
try {
conn = JDBCDataSource.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql.toString());
pstmt.setLong(1, bean.getRoleId());
ResultSet rs = pstmt.executeQuery();
while (rs.next()) {
bean = new UserBean();
bean.setId(rs.getLong(1));
bean.setFirstName(rs.getString(2));
bean.setLastName(rs.getString(3));
bean.setLogin(rs.getString(4));
bean.setPassword(rs.getString(5));
bean.setDob(rs.getDate(6));
bean.setMobileNo(rs.getString(7));
bean.setRoleId(rs.getLong(8));
bean.setGender(rs.getString(9));
bean.setProfilePic(rs.getString(10));
bean.setCreatedBy(rs.getString(11));
bean.setModifiedBy(rs.getString(12));
bean.setCreatedDatetime(rs.getTimestamp(13));
```

```
bean.setModifiedDatetime(rs.getTimestamp(14));
list.add(bean);
}
rs.close(); 45

} catch (Exception e) {
log.error("Database Exception..", e);
throw new ApplicationException("Exception : Exception in get roles");
} finally {
JDBCDataSource.closeConnection(conn);
}
log.debug("Model get roles End");
return list;
}
public boolean changePassword(Long id, String oldPassword, String newPassword)
throws RecordNotFoundException, ApplicationException {
log.debug("model changePassword Started");
boolean flag = false;
UserBean beanExist = null;
beanExist = findByPK(id);
if (beanExist != null && beanExist.getPassword().equals(oldPassword)) {
beanExist.setPassword(newPassword);
try {
update(beanExist);
} catch (DuplicateRecordException e) {
log.error(e);
throw new ApplicationException("LoginId is already exist");
}
flag = true;
} else {
throw new RecordNotFoundException("Old password is Invalid");
```

```
}
HashMap<String, String> map = new HashMap<String, String>();
map.put("login", beanExist.getLogin());
map.put("password", beanExist.getPassword());
map.put("firstName", beanExist.getFirstName());
map.put("lastName", beanExist.getLastName());
String message = EmailBuilder.getChangePasswordMessage(map);
EmailMessage msg = new EmailMessage(); 46

msg.setTo(beanExist.getLogin());
msg.setSubject("SUNARYS ORS Password has been changed Successfully.");
msg.setMessage(message);
msg.setMessageType(EmailMessage.HTML_MSG);
try {
EmailUtility.sendMail(msg);
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
log.debug("Model changePassword End");
return flag;
}
public UserBean updateAccess(UserBean bean) throws ApplicationException {
return null;
}
public long registerUser(UserBean bean)
throws ApplicationException, DuplicateRecordException {
log.debug("Model add Started");
long pk = add(bean);
HashMap<String, String> map = new HashMap<String, String>();
map.put("login", bean.getLogin());
```

```
map.put("password", bean.getPassword());
String message = EmailBuilder.getUserRegistrationMessage(map);
EmailMessage msg = new EmailMessage();
msg.setTo(bean.getLogin());
msg.setSubject("Registration is successful for ORS Project SunilOS");
msg.setMessage(message);
msg.setMessageType(EmailMessage.HTML_MSG);
try {
EmailUtility.sendMail(msg);
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return pk;
} 47

public boolean forgetPassword(String login)
throws ApplicationException, RecordNotFoundException, ApplicationException {
UserBean userData = findByLogin(login);
boolean flag = false;
if (userData == null) {
throw new RecordNotFoundException("Email ID does not exists !");
}
HashMap<String, String> map = new HashMap<String, String>();
map.put("login", userData.getLogin());
map.put("password", userData.getPassword());
map.put("firstName", userData.getFirstName());
map.put("lastName", userData.getLastName());
String message = EmailBuilder.getForgetPasswordMessage(map);
EmailMessage msg = new EmailMessage();
msg.setTo(login);
```

```
msg.setSubject("SUNARYS ORS Password reset");
msg.setMessage(message);
msg.setMessageType(EmailMessage.HTML_MSG);
EmailUtility.sendMail(msg);
flag = true;
return flag;
}
} 48
```

Controller

```
package in.co.college.att.mgt.controller;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.log4j.Logger;
import in.co.college.att.mgt.bean.BaseBean;
import in.co.college.att.mgt.bean.RoleBean;
import in.co.college.att.mgt.bean.UserBean;
import in.co.college.att.mgt.exception.ApplicationException;
import in.co.college.att.mgt.model.RoleModel;
import in.co.college.att.mgt.model.UserModel;
import in.co.college.att.mgt.util.DataUtility;
import in.co.college.att.mgt.util.DataValidator;
import in.co.college.att.mgt.util.PropertyReader;
import in.co.college.att.mgt.util.ServletUtility;
@WebServlet(name = "LoginCtl", urlPatterns = { "/login" })
public class LoginCtl extends BaseCtl {
```

```
private static final long serialVersionUID = 1L;
public static final String OP_REGISTER = "Register";
public static final String OP_SIGN_IN = "SignIn";
public static final String OP_SIGN_UP = "SignUp";
public static final String OP_LOG_OUT = "logout";
public static String HIT_URI = null;
private static Logger log = Logger.getLogger(LoginCtl.class);
public LoginCtl() {
super();
// TODO Auto-generated constructor stub
}
@Override
protected boolean validate(HttpServletRequest request) {
log.debug("LoginCtl Method validate Started");
boolean pass = true;
String op = request.getParameter("operation"); 49

if (OP_SIGN_UP.equals(op) || OP_LOG_OUT.equals(op)) {
return pass;
}
String login = request.getParameter("login");
if (DataValidator.isNull(login)) {
request.setAttribute("login", PropertyReader.getValue("error.require", "Login Id"));
pass = false;
} else if (!DataValidator.isEmail(login)) {
request.setAttribute("login", PropertyReader.getValue("error.email", "Login Id "));
pass = false;
}
if (DataValidator.isNull(request.getParameter("password"))) {
request.setAttribute("password", PropertyReader.getValue("error.require", "Password"));
pass = false;
```

```
}
log.debug("LoginCtl Method validate Ended");
return pass;
}
/**
* Populates bean object from request parameters
*
* @param request
* @return
*/
@Override
protected BaseBean populateBean(HttpServletRequest request) {
log.debug("LoginCtl Method populateBean Started");
UserBean bean = new UserBean();
bean.setId(DataUtility.getLong(request.getParameter("id")));
bean.setLogin(DataUtility.getString(request.getParameter("login")));
bean.setPassword(DataUtility.getString(request.getParameter("password")));
log.debug("LOginCtl Method PopulatedBean End");
return bean;
} 50

/**
* Display Login form
*
*/
/**
* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
* response)
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
```

```
log.debug("Method doGet Started");
HttpSession session = request.getSession(true);
String op = DataUtility.getString(request.getParameter("operation"));
UserModel model = new UserModel();
RoleModel role = new RoleModel();
long id = DataUtility.getLong(request.getParameter("id"));
if (id > 0) {
UserBean userBean;
try {
userBean = model.findByPK(id);
ServletUtility.setBean(userBean, request);
} catch (Exception e) {
log.error(e);
ServletUtility.handleException(e, request, response);
return;
}
} else if (OP_LOG_OUT.equals(op)) {
session = request.getSession(false);
session.invalidate();
ServletUtility.setSuccessMessage("You have been logged out successfully", request);
ServletUtility.forward(CASView.LOGIN_VIEW, request, response);
return;
}
if (session.getAttribute("user") != null) {
ServletUtility.redirect(CASView.WELCOME_CTL, request, response);
return;
}
ServletUtility.forward(getView(), request, response);
log.debug("Method doGet end");
} 51
```

```
/**
* Submit Logic
*/
/**
* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
* response)
*/
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
log.debug(" LoginCtl Method doPost Started");
HttpSession session = request.getSession(true);
String op = DataUtility.getString(request.getParameter("operation"));
// get Model
UserModel model = new UserModel();
RoleModel role = new RoleModel();
long id = DataUtility.getLong(request.getParameter("id"));
if (OP_SIGN_IN.equalsIgnoreCase(op)) {
UserBean bean = (UserBean) populateBean(request);
try {
bean = model.authenticate(bean.getLogin(), bean.getPassword());
if (bean != null) {
session.setAttribute("user", bean);
session.setMaxInactiveInterval(10 * 6000);
long rollId = bean.getRoleId();
RoleBean roleBean = role.findByPK(rollId);
if (roleBean != null) {
session.setAttribute("role", roleBean.getName());
}
// save state of session remember URL
String uri = request.getParameter("uri");
if (uri == null || "null".equalsIgnoreCase(uri)) {
```

```
ServletUtility.redirect(CASView.WELCOME_CTL, request, 52


response);
return;
} else {
ServletUtility.redirect(uri, request, response);
}
return;
} else {
bean = (UserBean) populateBean(request);
ServletUtility.setBean(bean, request);
ServletUtility.setErrorMessage("Invalid LoginId And Password", request);
}
} catch (ApplicationException e) {
log.error(e);
ServletUtility.handleException(e, request, response);
return;
}
} else if (OP_SIGN_UP.equalsIgnoreCase(op)) {
ServletUtility.redirect(CASView.USER_REGISTRATION_CTL, request, response);
return;
}
ServletUtility.forward(getView(), request, response);
log.debug("UserCtl Method doPost Ended");
}
/**
* Returns the VIEW page of this Controller
*
* @return
*/
@Override
```

```
protected String getView() {
return CASView.LOGIN_VIEW;
}
}
```

Beans

```
package in.co.college.att.mgt.bean;
import java.sql.Timestamp;
import java.util.Date;
public class UserBean extends BaseBean { 53

private String firstName;
private String lastName;
private String login;
private String password;
private String confirmPassword;
private Date dob;
private String mobileNo;
private long roleId;
private String gender;
private String profilePic;
public String getProfilePic() {
return profilePic;
}
public void setProfilePic(String profilePic) {
this.profilePic = profilePic;
}
public String getFirstName() {
return firstName;
}
/**
* @param FirstName
```

```
* To set User FirstName
*/
public void setFirstName(String firstName) {
this.firstName = firstName;
}
/**
* @return LastName Of User
*/
public String getLastName() {
return lastName;
}
/**
* @param LastName
* To set User LastName
*/
public void setLastName(String lastName) { 54

this.lastName = lastName;
}
/**
* @return Login id Of User
*/
public String getLogin() {
return login;
}
/**
* @param Login
* Id To set User Login ID
*/
public void setLogin(String login) {
this.login = login;
```

```
}
/**
* @return Password Of User
*/
public String getPassword() {
return password;
}
/**
* @param Password
* To set User Password
*/
public void setPassword(String password) {
this.password = password;
}
/**
* @return Confirm Password Of User
*/
public String getConfirmPassword() {
return confirmPassword;
}
/**
* @param Confirm
* PAssword To set User Confirm Password
*/
public void setConfirmPassword(String confirmPassword) {
this.confirmPassword = confirmPassword;
}
/**
* @return Date Of Birth Of User
*/
public Date getDob() {
```

```
return dob;
}
/** 55

* @param Date
* Of Birth To set User Date Of Birth
*/
public void setDob(Date dob) {
this.dob = dob;
}
/**
* @return Mobile No Of User
*/
public String getMobileNo() {
return mobileNo;
}
/**
* @param Mobile
* No To set User Mobile No
*/
public void setMobileNo(String mobileNo) {
this.mobileNo = mobileNo;
}
/**
* @return ROle Id Of User
*/
public long getRoleId() {
return roleId;
}
/**
* @param Role
```

```
* Id To set User ROle Id
*/
public void setRoleId(long roleId) {
this.roleId = roleId;
}
/**
* @return unSuccessfulLogin Of User
*/
public String getGender() {
return gender;
} 56

/**
* @param Gender
* To set User Gender
*/
public void setGender(String gender) {
this.gender = gender;
}
public String getKey() {
return id + "";
}
public String getValue() {
return firstName + " " + lastName;
}
}
```