**Creating the Home Page**

Home page is pretty straightforward. We have a menu and 8 images displayed in a table format with two rows; each row containing 4 images. On the home page we also make a check, whether the user is logged in or not. If the user is logged in we also display the username and provide a logout link.

**Creating Menu for Home Page**

< div id='cssmenu'>

< ul>

< li class=''>< a href='${pageContext.request.contextPath}'>< span>Home< /span>< /a>< /li>

< li>< a href='${pageContext.request.contextPath}/login'>< span>Login< /span>< /a>< /li>

< li>< a href='${pageContext.request.contextPath}/register'>< span>Register< /span>< /a>< /li>

< li class='#'>< a href='#'>< span>Submit a Question< /span>< /a>< /li>

< li class='#'>< a href='#'>< span>Feedback< /span>< /a>< /li>

< li>< a href='#'>< span>Contribute< /span>< /a>< /li>

< li>< a href='#'>< span>Contact us< /span>< /a>< /li>

< /ul>

< /div>

**Checking whether the user is logged in or not**

< c:if test='${not empty sessionScope.user}'>

< div style="position:absolute;top:70px;left:1100px">

Logged as < a href="#" class="button username">${sessionScope.user}< /a>

< /div>

< div style="position:absolute;top:70px;left:1300px">

< a href='${pageContext.request.contextPath}/logout'>Logout< /a>

< /div>

< /c:if>

**Showing the Tabs on home page**

< div style="position:absolute;left:120px;top:60px">

< table cellpadding="0" cellspacing="50">

< tr>

< td>< a href="takeExam?test=java">< img height="200" width="200" src="${pageContext.request.contextPath}/images/java.png"/>< /a>< /td>

< td>< a href="takeExam?test=javascript">< img height="200" width="200" src="${pageContext.request.contextPath}/images/javascript.png"/>< /a>< /td>

< td>< a href="takeExam?test=sql">< img height="200" width="200" src="${pageContext.request.contextPath}/images/sql-logo.png"/>< /a>< /td>

< td>< a href="takeExam?test=python">< img height="200" width="200" src="${pageContext.request.contextPath}/images/python.jpg"/>< /a>< /td>

< /tr>

< tr>

< td>< a href="takeExam?test=css">< img height="200" width="200" src="${pageContext.request.contextPath}/images/css.jpg"/>< /a>< /td>

< td>< a href="takeExam?test=php">< img height="200" width="200" src="${pageContext.request.contextPath}/images/php-logo.jpg"/>< /a>< /td>

< td>< a href="takeExam?test=linux">< img height="200" width="200" src="${pageContext.request.contextPath}/images/logo-linux.png"/>< /a>< /td>

< td>< a href="takeExam?test=mongodb">< img height="200" width="200" src="${pageContext.request.contextPath}/images/mongodb\_logo.png"/>< /a>< /td>

< /tr>

< /table>

< /div>

**Creating the User Registration Page**

There is nothing fancy in the registration page; just an HTML form awaiting the user to provide his name, email and password. Once we get that, we pass this to RegistrationController servlet to create an account.

Note: We are not doing any validation like password should contain 8 characters with at least one uppercase character, one number and special symbol. We will do that in upcoming posts, when we extend this application.

**Registration Code**

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

String username=request.getParameter("username");

String email=request.getParameter("email");

String password=request.getParameter("password");

Connection con=DatabaseConnectionFactory.createConnection();

try

{

Statement st=con.createStatement();

String sql = "INSERT INTO users values ('"+username+"','"+password+"','"+email+"')";

System.out.println(sql);

st.executeUpdate(sql);

}catch(SQLException sqe){System.out.println("Error : While Inserting record in database");}

try

{

con.close();

}catch(SQLException se){System.out.println("Error : While Closing Connection");}

request.setAttribute("newUser",username);

RequestDispatcher dispatcher=request.getRequestDispatcher("/WEB-INF/jsps/regSuccess.jsp");

dispatcher.forward(request, response);

}

**Getting Database Connection**

In this application we have used MySQL database to store user credentials. To get a connection to database we have defined a static method createConnection in DatabaseConnectionFactory class, where all database specific information is stored.

We have just users’ table under quiz database.

**Users’ table**

create table users(username varchar(50),email varchar(50),password varchar(50))

If you are working with some other database like Oracle you have to change the properties of the DatabaseConnectionFactory class accordingly.

**DatabaseConnectionFactory.java**

public class DatabaseConnectionFactory {

private static String dbURL="jdbc:mysql://localhost/mailsystem";

private static String dbUser="root";

private static String dbPassword="";

public static Connection createConnection()

{

Connection con=null;

try{

try {

Class.forName("com.mysql.jdbc.Driver");

}

catch(ClassNotFoundException ex) {

System.out.println("Error: unable to load driver class!");

System.exit(1);

}

con = DriverManager.getConnection(dbURL,dbUser,dbPassword);

}

catch(SQLException sqe){ System.out.println("Error: While Creating connection to database");sqe.printStackTrace();}

return con;

}

}

**Creating the Login Page**

Login page is very much similar to registration page where we are providing two input fields asking user to provide a username and password. Once we get the username and password entered by the user we pass it to LoginController to authenticate user.

**Login Validation Code**

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

String username=request.getParameter("username");

String password=request.getParameter("password");

Connection con=DatabaseConnectionFactory.createConnection();

ResultSet set=null;

int i=0;

try

{

Statement st=con.createStatement();

String sql = "Select \* from users where username='"+username+"' and password='"+password+"' ";

System.out.println(sql);

set=st.executeQuery(sql);

while(set.next())

{

i=1;

}

if(i!=0)

{ HttpSession session=request.getSession();

session.setAttribute("user",username);

RequestDispatcher rd=request.getRequestDispatcher("/WEB-INF/jsps/home.jsp");

rd.forward(request, response);

}

else

{ request.setAttribute("errorMessage","Invalid username or password");

RequestDispatcher rd=request.getRequestDispatcher("/WEB-INF/jsps/login.jsp");

rd.forward(request, response);

}

}catch(SQLException sqe){System.out.println("Error : While Fetching records from database");}

try

{

con.close();

}catch(SQLException se){System.out.println("Error : While Closing Connection");}

}

**MainController for the Application**

It is the MainController where we have written the code to redirect the user to appropriate page according to the incoming request url.

@WebServlet(urlPatterns = { "/login", "/register", "/takeExam", "/logout" })

public class MainController extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

String applicationContextPath = request.getContextPath();

if (request.getRequestURI().equals(applicationContextPath + "/")) {

RequestDispatcher dispatcher = request

.getRequestDispatcher("/WEB-INF/jsps/home.jsp");

dispatcher.forward(request, response);

} else if (request.getRequestURI().equals(

applicationContextPath + "/login")) {

RequestDispatcher dispatcher = request

.getRequestDispatcher("/WEB-INF/jsps/login.jsp");

dispatcher.forward(request, response);

} else if (request.getRequestURI().equals(

applicationContextPath + "/register")) {

RequestDispatcher dispatcher = request

.getRequestDispatcher("/WEB-INF/jsps/register.jsp");

dispatcher.forward(request, response);

} else if (request.getRequestURI().equals(

applicationContextPath + "/takeExam")) {

request.getSession().setAttribute("currentExam", null);

String exam = request.getParameter("test");

request.getSession().setAttribute("exam", exam);

System.out.println(request.getSession().getAttribute("user"));

if (request.getSession().getAttribute("user") == null) {

request.getRequestDispatcher("/login").forward(request,

response);

} else {

RequestDispatcher dispatcher = request

.getRequestDispatcher("/WEB-INF/jsps/quizDetails.jsp");

dispatcher.forward(request, response);

}

} else if (request.getRequestURI().equals(

applicationContextPath + "/logout")) {

request.getSession().invalidate();

RequestDispatcher dispatcher = request

.getRequestDispatcher("/WEB-INF/jsps/home.jsp");

dispatcher.forward(request, response);

}

}

}

**Implementing the Logout Functionality**

Once the user clicks on logout, link session is invalidated and all the objects bind in the session are removed.

request.getSession().invalidate();

**Storing the Quiz questions**

Note that we have stored the questions in separate XML files, not in the database.

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>EmailDashboard</display-name>

<welcome-file-list>

<welcome-file>login.html</welcome-file>

</welcome-file-list>

<servlet>

<description></description>

<display-name>ComposeServlet</display-name>

<servlet-name>ComposeServlet</servlet-name>

<servlet-class>com.mail.ComposeServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ComposeServlet</servlet-name>

<url-pattern>/ComposeServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>ComposeServletProcess</display-name>

<servlet-name>ComposeServletProcess</servlet-name>

<servlet-class>com.mail.ComposeServletProcess</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ComposeServletProcess</servlet-name>

<url-pattern>/ComposeServletProcess</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>DeleteMailServlet</display-name>

<servlet-name>DeleteMailServlet</servlet-name>

<servlet-class>com.mail.DeleteMailServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>DeleteMailServlet</servlet-name>

<url-pattern>/DeleteMailServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>InboxServlet</display-name>

<servlet-name>InboxServlet</servlet-name>

<servlet-class>com.mail.InboxServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>InboxServlet</servlet-name>

<url-pattern>/InboxServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>LoginServlet</display-name>

<servlet-name>LoginServlet</servlet-name>

<servlet-class>com.mail.LoginServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>LoginServlet</servlet-name>

<url-pattern>/LoginServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>LogoutServlet</display-name>

<servlet-name>LogoutServlet</servlet-name>

<servlet-class>com.mail.LogoutServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>LogoutServlet</servlet-name>

<url-pattern>/LogoutServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>MessageAction</display-name>

<servlet-name>MessageAction</servlet-name>

<servlet-class>com.mail.MessageAction</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MessageAction</servlet-name>

<url-pattern>/MessageAction</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>PermanentDeleteMailServlet</display-name>

<servlet-name>PermanentDeleteMailServlet</servlet-name>

<servlet-class>com.mail.PermanentDeleteMailServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>PermanentDeleteMailServlet</servlet-name>

<url-pattern>/PermanentDeleteMailServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>RegisterServlet</display-name>

<servlet-name>RegisterServlet</servlet-name>

<servlet-class>com.mail.RegisterServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>RegisterServlet</servlet-name>

<url-pattern>/RegisterServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>SentServlet</display-name>

<servlet-name>SentServlet</servlet-name>

<servlet-class>com.mail.SentServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>SentServlet</servlet-name>

<url-pattern>/SentServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>TrashServlet</display-name>

<servlet-name>TrashServlet</servlet-name>

<servlet-class>com.mail.TrashServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>TrashServlet</servlet-name>

<url-pattern>/TrashServlet</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>UpdateMailStatus</display-name>

<servlet-name>UpdateMailStatus</servlet-name>

<servlet-class>com.mail.UpdateMailStatus</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>UpdateMailStatus</servlet-name>

<url-pattern>/UpdateMailStatus</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>ViewMailServlet</display-name>

<servlet-name>ViewMailServlet</servlet-name>

<servlet-class>com.mail.ViewMailServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ViewMailServlet</servlet-name>

<url-pattern>/ViewMailServlet</url-pattern>

</servlet-mapping>

</web-app

**How to Read the message Stored inDatabase File**

To read the questions from the XML file we create a document that represents the sql file containing quiz questions. Whenever the user clicks on the next or previous button we call the setSender(int i) method, giving the index of question that we want to read and at the same time that question is saved in an ArrayList of SendEmail.

**How to Represent a Question?**

SendEmail is the class that represents a single message; each message will have a number, options and one correct option index.

ComposeServletProcess**.java**

**public** **class** ComposeServletProcess **extends** HttpServlet {

@Override

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

PrintWriter out=response.getWriter();

HttpSession session=request.getSession();

String from=(String)session.getAttribute("from");

String[] to=(String[])session.getAttribute("to");

String subject=(String)session.getAttribute("subject");

String message1=(String)session.getAttribute("message");

System.***out***.println(subject);

System.***out***.println(to);

System.***out***.println(from);

System.***out***.println(message1);

message1=message1.replaceAll("\n","<br/>");

String msg="";

**try**{

com.app.Encryption enc=**new** com.app.Encryption();

msg=enc.encrypt(message1);

System.***out***.println(msg);

}**catch**(Exception e){e.printStackTrace();}

**int** i=0;//ComposeDao.save(from, to, subject, msg);

**if**(i>0)

{

response.sendRedirect("Inbox.jsp");

}

request.getRequestDispatcher("footer.html").include(request, response);

out.close();

}

}

Note that since this is a web application, multiple users will be taking exams simultaneously. We have to make sure that one user’s exam does not get into another user’s exam. For example, one user might have just started Java exam and another user is on question 5 of SQL exam; we have to treat them as two separate exams. To do that we will maintain the state of each exam using session.

When the user clicks on start exam button to start the exam, we will create a new instance of email passing the test type for eg. Gmail, yahoo, etc. So each user will have a different instance of Exam class (that represents an individual exam).

Let’s see what is there in the exam class

public class SentMail {

Document dom;

public int currentuser=0;

public Map selections=new LinkedHashMap();

public ArrayList questionList = new ArrayList(10);

public Exam(String test) throws SAXException,ParserConfigurationException,IOException, URISyntaxException{

dom=CreateDOM.getDOM(test);

}

// code

}

Note that to track the current question in the exam we have currentQuestion property in exam class.

**Handling the Entire Email**

admin is the main control from where we control the email server. Here we save user selections (what user have answered for the question) in a Map. Controller also lets user move through questions by clicking next and previous button, at the back end it is the Controller which makes the function calls to retrieve questions and store user responses.

**Receive Email Code**

<%

**if**(rs.getString(9)!=**null**){

**if**(rs.getString(9).equals("Home")){

%>

<span class=*"label label-danger"*><%=rs.getString(9) %></span>

<%}**if**(rs.getString(9).equals("Work")){

%>

<span class=*"label label-success"*><%=rs.getString(9) %></span>

<%} **if**(rs.getString(9).equals("Personal")){

%>

<span class=*"label label-warning"*><%=rs.getString(9) %></span>

<%} **if**(rs.getString(9).equals("Documents")){

%>

<span class=*"label label-info"*><%=rs.getString(9) %></span>

<%} **if**(rs.getString(9).equals("Important")){

%>

<span class=*"label label-primary"*><%=rs.getString(9) %></span>

<%} }%>

<span class=*"subject"*><%= rs.getString(4) %>

</span> -

<span class=*"body"*>

<%

String msg=rs.getString(5);

Decryption dec=**new** Decryption();

msg1=dec.decrypt(msg);

%>

<%=msg1 %>

Note that until now each of our question has 4 options and there is no timer for the quiz. In the upcoming posts we are going to extend this online quiz application and will include the following functionalities :

1. Each email can have different number of user

2. Each email can have different number of options like outbox, inbox, spam etc.

3. A question can have multiple correct options

4. Implementing a timer for the email

5. Maintaining a history of the user; like how many messages a user have taken in the past and his status

6. Randomizing the order of labels presented to the user

7. Giving the user option to review his answers before submitting the test for message.

8. A dropdown box to jump to any email in between the test rather then clicking next button multiple times.

Click on the Download button to download the code.