
# Uttara InfoSolutions

www.uttarainfo.com

## MVC Flavour 2 Practicals 1

**Basic JSP Questions(do this first if you are not familiar with JSPs):**

1> Create a simple jsp to return the current date and time.

2> Take an input from the user (a number) and return whether the number is even / odd (do proper validations).

3> Create a Register.html. Take name, email, date of birth as inputs. Submit to a JSP. Validate the user inputs (in a scriplet) and show a welcome response (Depending on age of person).

**Implementation of MVC for Register Usecase:**

**Things to know before implementing MVC (verify if you know this first mentally):**

a) You should have familiarity of Servlets lifecycle
b) You should have knowledge of how to use RequestDispatcher to forward control from Servlet to another resource (another Servlet / JSP / HTML)
c) You should know how to access request data (parameters, requestURI, etc) in Servlet`s doXXX()
d) You should know session management
e) You should know about the lifecycle of JSPs
f) You should know how to use the scripting elements of JSPs
g) You should know how to use EL in JSPs
h) You should know how to use directives and JSP Action elements
i) You should know how to implement incoming data usecase from input view via intermediary JSP.
j) You should know JDBC - how to connect to DB, how to do exception handling and release, how to execute insert/update/delete and select statements and retrieve data.

Copy the MVCApp.war file given to you, import it in Eclipse, create a table called register with the following sql in HSQLDB (start server and UI manager first):

create table register(sl_no int primary key generated by default as identity, name varchar(200), email varchar(200), pass varchar(200))

1) Eclipse -> Create Dynamic web project -> MVCApp -> Dynamic web module version 2.5

2) Drag and drop servlet-api.jar from Tomcats lib folder to the lib folder in WebContent\WEB-INF of MVCApp

3) Right click on project, close unrelated projects except servers

4) Right click on project-> new -> HTML/JSP -> Register.jsp

5) Provide 4 text boxes, one for name, email, password, repeat pass in a form tag with action = "register.do" and method="post" and one submit button

6) Right click on HTML/JSP -> run on server -> link server to Tomcat 6/7 if not done. Verify working.

7) Right click on project -> new -> Servlet -> ControllerServlet (put in package com.uttarainfo.mvc). Put SOPs in constructor, doGet,doPost. After saving, open web.xml-> change URL pattern to *.do (not /*.do)

8) Run the view, click on submit and test whether the SOPs are getting outputted to the console. Do you understand the control flow? Why did we add .do to the URL pattern? Why are we mapping *.do in web.xml?

9) Right click on project -> new -> java class -> RegBean. Put in package (must do) com.uttarainfo.mvc. Create 4 instance variables with same name as that of the input form element names in the view (all lowercase and name should exactly match). Right click in the source code, select source -> generate setters/getters -> choose all Put SOPs in public no-arg constructor of RegBean and setters/getters

10) Right click on project -> create new -> JSP -> RegInt.jsp with following code (do not copy paste but type it out yourself and use auto complete in Eclipse):

```
<jsp:useBean id = "reg" class="com.uttarainfo.mvc.RegBean" scope="request">
        <jsp:setProperty name="reg" property="*"/>
</jsp:useBean>
<jsp:forward page="register.do"/>
```

11> Modify action element in the Register view to submit to RegInt.jsp. Right click run on server and verify whether in the console SOPs of creating bean, invoking setters, creating CS, invoking doXXX are being fired.

12> Call process(request,response) from both doXXX() methods in CS

13> Invoke String uri = request.getRequestURI(); in process(). SOP it out. Create if block.

```
if(uri.contains("/register")
{
  // write controlling logic for registration usecase
}
```

14> Controlling logic for registration:
a) Obtain RegBean from the request scope by
RegBean rb = (RegBean) request.getAttribute("reg");
b) Invoke Models register() method and pass the RegBean as parameter. Accept String as return value.
c) If string returned is "SUCCESS" the request dispatch to Success.html else, store the returned string in the request scope as an attribute("errorMsg") and then request dispatch

to Register.jsp (promote Register.html to REgister.jsp here)

15> Create a validate() method in the RegBean and validate user inputs for mandatory checks and whether pwd=rpwd check. Return a string from validate()..."SUCCESS" if user input checks succeed.

16> In Models register(), first call regbean.validate() and verify if the returned string is "SUCCESS". If yes, continue else return the string to CS.

17> Make sure the table is present in database

18> In Models register(), after succeeding user input validations, get connection to DB. Copy code from example to know which driver class to load and how to ask DriverManager
to get connection. SOP the conn out and verify working of connection.

19> Create PreparedStatement for doing business validation check (duplicate email). (select * from register where email=?)

20> Create PreparedStatement for doing business logic (insert into register...)

21> return "SUCCESS" once the insertion is done. Run the app, verify in DB.

22> Create new interface Constants (in package com.uttarainfo.mvc) and create string variable SUCCESS with value "success". Replace use of "SUCCESS" with Constants.SUCCESS
in Model / CS.

23> Create class JDBCHelper. Create static method getConnection(). Put the code to establish connection here and invoke this method from model whenever you need to access
connection to db.

24> Put try..catch in CS and forward to Error.jsp on exceptions and display the message.

**Implementation of MVC for View Registered Users Usecase:**

1> Create a HomePage.html with 2 hyperlinks
1..Click here to Register
2..Click here to view registered users

On click of 2nd link, submit to "viewRegisteredUsers.do".

2> Add an if block in CS

```
if(uri.contains("/viewRegisteredUsers")
{
        // put SOP. verify whether control is coming here.
}
```

3> In the if block, invoke Models getUsers() method. It should return List<RegBean>.
4> Create getUsers() method in Model.
a) Invoke JDBCHelper.getConnection() to get connection to db.
b) Create PreparedStatement to "select * from reg_users".

c) Invoke ps.getResultSet(). Loop over it.

```
List al = new ArrayList();
RegBean rb = null;
while(rs.next())
{
        rb = new RegBean();
        rb.setUname(rs.getString("name");
        rb.setEmail(rs.getString("email");
        rb.setPwd(rs.getString("pwd");

        al.add(rb);
}
```

d) return al to CS.

5> Once the list is returned, store in the request scope as an attribute ("listUsers") and then request dispatch to ViewUsers.jsp.

6> ViewUsers.jsp:

```
<%
        List al = (List) request.getAttribute("listUsers");
        for (Object o:al)
        {
                RegBean rb = (RegBean) o;

                out.println("Name : "+rb.getUname() + " Email : "+rb.getEmail());
        }
%>
```

7> Test!!

**Bonus!!**

a) Build Login usecase similarly!

b) Add the load-on-startup tag to CS in web.xml

c) Add the session-config tag in web.xml and configure session timeout as 10 mins