Web Application

Static Web Application

Content of the page is same for all user

Web developer

Tech stack: HTML, CSS, JS, Jquery (client side)

**Dynamic Web Application**

Content of the page is differ for every user

Java Developer

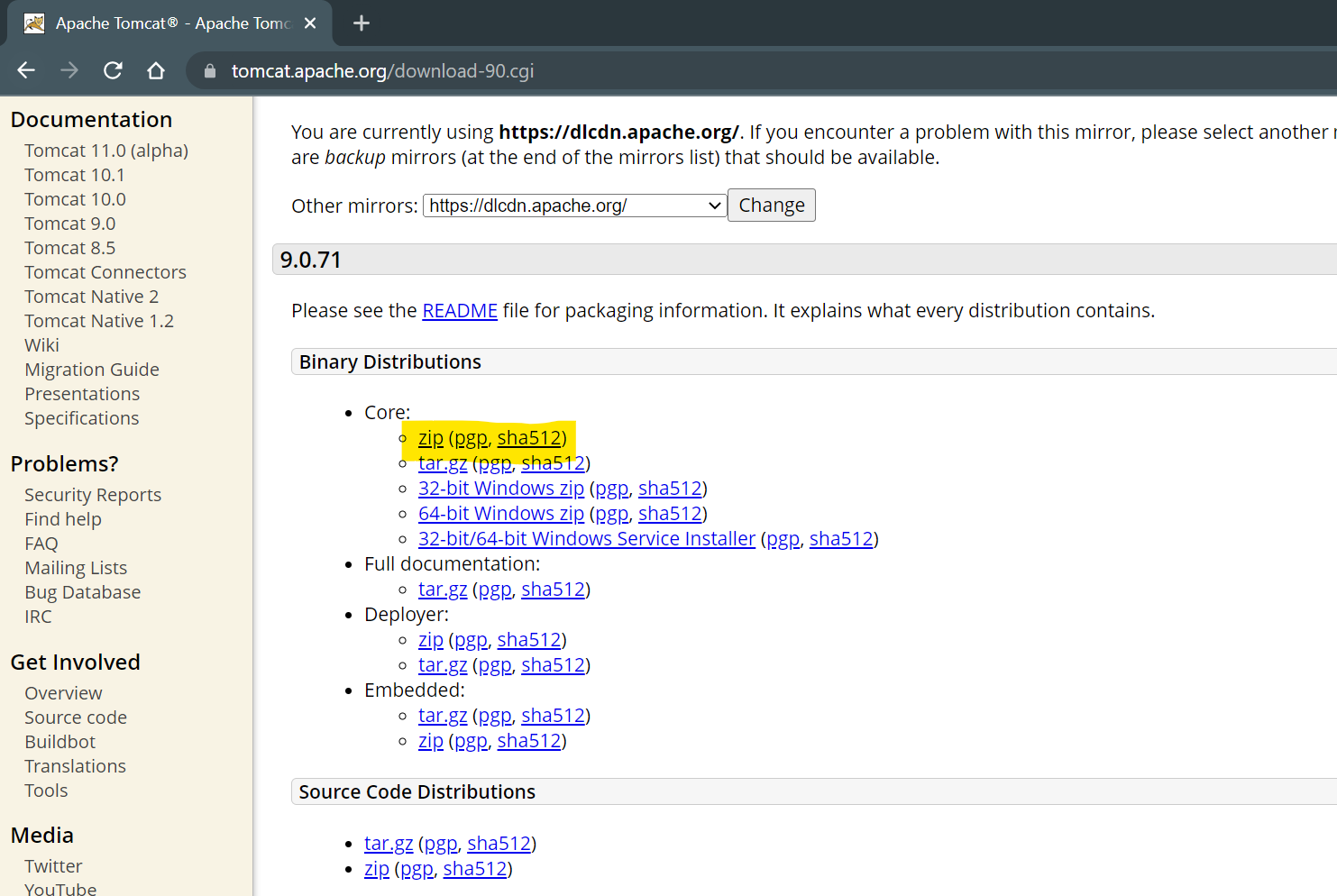
Tech Stack: HTML, CSS, JS, Jquery (client side)

Jsp and servlet (server side)

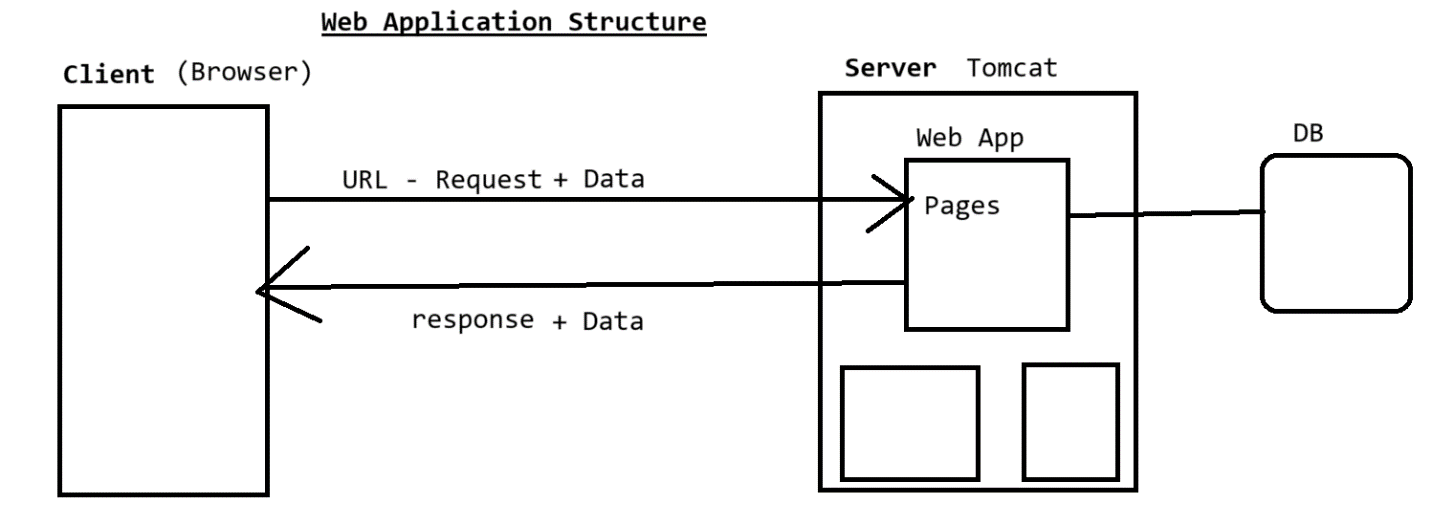
Tomcat Server Setup

1. Download tomcat

<https://tomcat.apache.org/download-90.cgi>



1. Copy and paste the zip file into specific location.
2. Extract the Zip File
3. Server Setup in Eclipse
   1. Set the “**JavaEE”** perspective
   2. Select “Servers” tab from the bottom for the window
   3. Click on the link to add new server.
   4. Expand “Apache” option from the list in the new window.
   5. Select a version which you downloaded
   6. Click on “Next”
   7. Browse and set the path of the location where you extracted the server zip file.
      1. Path of the parent folder of bin,lib, config etc.
   8. Click on Next and Finish



**Steps to create Web application**

1. Go “File” Menu -> “New” -> select “Dynamic Web Project”
2. Provide Name of the application and **make sure that Target Runtime** is selected and not <none>
3. Click on “Next” -> click On “Next”
4. **Make sure that “generate Web.xml Deployment Descriptor” option is checked/selected**

**Request (URL)**



**Servlet**

1. Servlet is a java class.
2. In servlet main method is not used.
3. Servlet will be execute by the server (Servlet Container).
4. Servlet are execute and can be access by using URL.
5. Servlet are use to create dynamic web pages.
6. In the servlet you can use java code along with HTML (HTML in JAVA)
7. Servlet has to create inside src/main/java
8. Servlet are use to get request, process a request and generate the response

**Ways To Create Servlet**

1. Create java class and use Java EE APIs to create servlet
2. Implements Class with **Servlet interface**
3. Extends class with **GenericServlet abstract class**
4. Extends class with **HttpServlet abstract class**
5. Override the service method to work with request and repones.
6. Provide the URL for the servlet using **@WebServlet** annotation on class.
   1. Use @WebServlet Annotation on class
   2. Provide the URL of the servlet which must be starts with “/”

**Steps to return Response**

1. To Return a response you have to use HttpServletResponse Object which is a part of service method.
2. You have to set the response type. The response type is also known as MIME type

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types>

1. To set the response type you can use following method

**response.setContentType(“<MIMEType>”);**

1. You will request the Writer (char type of response) or OutputStream (binary type of response) object to write a response.
2. To create a Writer you can use following method

**response.getWriter();**

1. To create a OutputStream you can use following method

**response.getOutputStream();**

1. To return a response you can use **print()** method

**Generate New Request**

1. After click onsubmit button.
2. After click on hyperlink (anchor tag)
3. After refresh the Page
4. From the java code using send redirect.

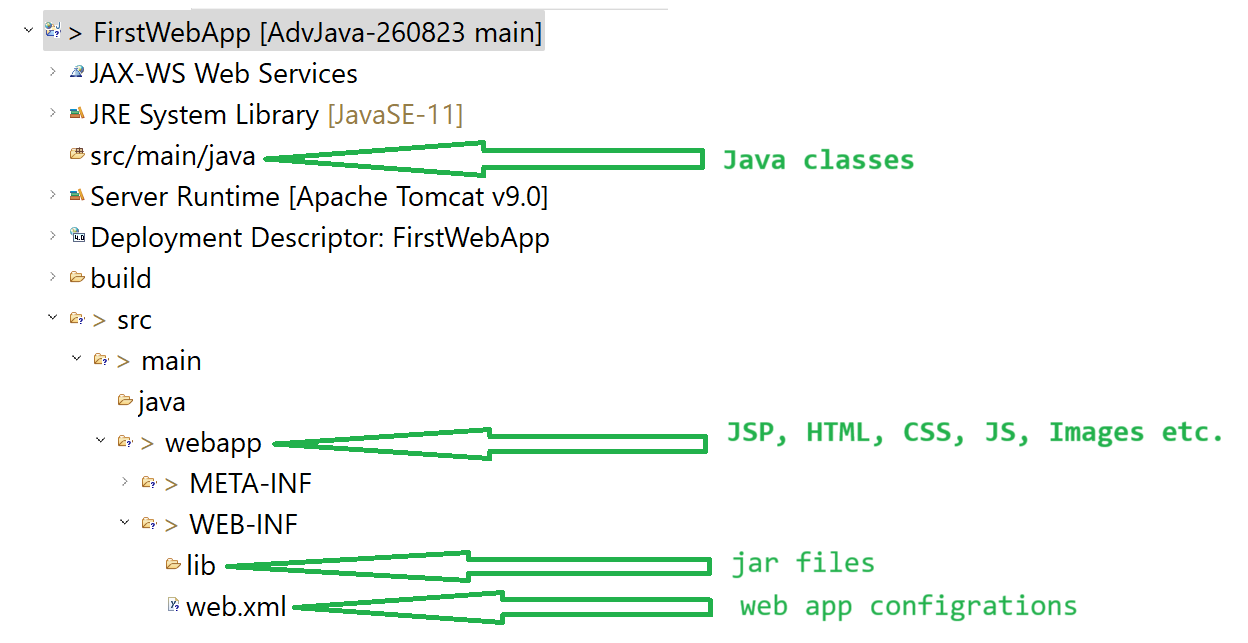
**Parameter**

1. Parameter is the use information/Data
2. This data will be always pass from the URL
3. Parameters can be added inside the request after ‘?’ or it can be pass from the request body.
4. Parameters are always in string format.
5. This parameter can be received in the Jsp/Servlet using request Object

**request.getParameter(“<name/key>”);**

1. Every parameter has key and value pair.
2. There can be a multiple parameter in the URL which will be separated by ‘&’

Project Structure



**Redirection Techniques**

1. These techniques are use to redirect user from one page to another with any user action.
2. There are two ways to achieve this
   1. **Request Dispatcher**
      1. It is an interface
      2. Is use to redirect from one page to another without user action.
      3. Using this you can redirect using an existing request.
      4. Here, no new request will be generated to go from one page to another.
      5. The existing request data will be available on the new page.
      6. Syntax:

**RequestDisptacher dis = request.getRequestDisptacher(“URL”);**

**dis.forward(request, response);**

**dis.include(request, response);**

* 1. **Send Redirect** 
     1. It is a method
     2. Is use to redirect from one page to another without user action.
     3. The new request will be generated for redirection.
     4. The data from the old request will be delete and it will not be available in the next page.
     5. Syntax:

**response.sendRedirect(“<URL>”)**

**JSP**

1. JSP is a Java Server Pages.
2. The extension of JSP page is .jsp.
3. JSP is used to design the dynamic web pages.
4. Inside JSP you can use HTML language as a primary language.
5. Other than HTML you can also use CSS, JavaScript, Java language on JSP page.
6. In JSP java will be added inside HTML (JAVA in HTML)
7. Every JSP page gets converted into Java class which is servlet before execution.
8. It is not necessary to provide URL for the JSP. By default every jsp page has a default url as /pageName.jsp
9. JSP pages are mainly use to design the pages.
10. JSP pages will be created inside src/main/webapp folder.

**Writing Java Code on JSP page**

1. Using Java code on the JSP page you can make your page dynamic.
2. To Write a java code JSP provided tags which is known as Scripting tags/element.
3. Nested of scripting is not allowed to achieve this you have to use break and continue rule.
4. There are 3 types of scripting tags/elements
   1. **Scriptlet Tag**
      1. Is use to write the java code on JSP page. This code gets added into converted page inside service method.
      2. The java code you added inside this tag will the local code.
      3. Using this tag you cannot create Instance and static variables. Also you cannot create a custom method using this tag.
      4. Syntax:

**<% Java Code %>**

* 1. **Expression Tag**
     1. Using this tag you can display the result on the Browser as an response.
     2. This is use to execute the java expressions and print the results on the browser.
     3. The java code written inside tag will also gets added inside a service methods. This code gets added inside an out.print() method.
     4. Syntax:

**<%= Java Expression %>**

* 1. **Declaration Tag**
     1. Using this tag you can write a java code which will gets added inside the class and outside the service method.
     2. Using this tag you can add the instance level code inside jsp page.
     3. You can create static variable, instance variable and methods
     4. Syntax:

**<%! Java Code %>**

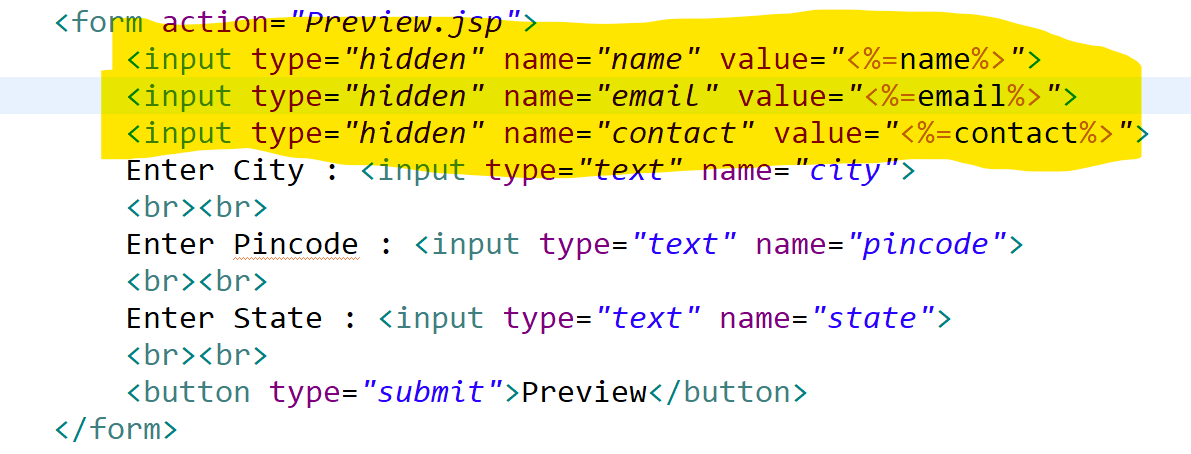
**Implicit Object**

1. These are the object which are provided by server on every JSP page.
2. These objects are internally created and provided to use in JSP page.
3. All the implicit objects are available only inside service method.
4. **These objects can be access on JSP page using scriptlet and expression tags only.**
5. There are total 9 implicit objects available on every JSP page.

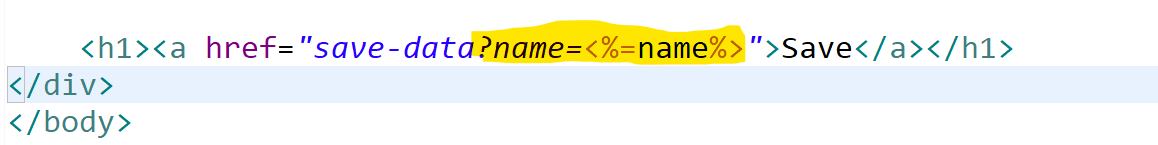
|  |  |
| --- | --- |
| **Object Name** | **Class/Interface** |
| request | HttpServletRequest |
| response | HttpServletResponse |
| out | JspWriter |
| session | HttpSession |
| application | ServletContext |
| config | ServletConfig |
| pageContext | PageContext |
| exception | Throwable |
| page | Same as this keyword |

**Session Tracking techniques**

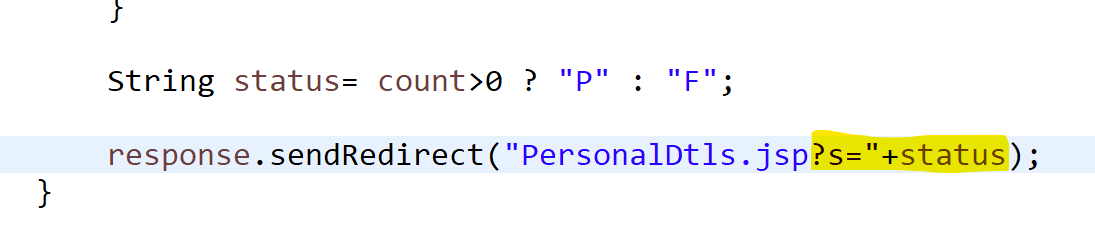
1. It is a way to retain the old request user data into new request.
2. Using this techniques you can pass the old request data into next request.
3. There are different session tracking techniques.
   1. Hidden Form field
      1. This is use to retain the old request data into new request which is generated through a form submission (Form and Submit button).
      2. This can be achieve by holding the old request data into hidden fields in the form tag.

****

* 1. URL rewriting
     1. This is use to retain the old request data into new request which is generated through the anchor tag or send redirect.
     2. Using this you can add the parameters manually inside URL.
     3. Example:
        1. Anchor Tag

****

* + - 1. Send Redirect

****

* 1. Cookies
     1. This is use to maintain the user data at client side.
     2. Disadvantage
        1. Can store only String type of user data using cookies.
        2. There is limit to store user data using cookies which is of 40 cookies.
        3. Cookies can be decline or disable by Client.
  2. HttpSession
     1. This is use to maintain the user data at server side
     2. **Create/get Session**

**HttpSession session = request.getSession();**

Above syntax to get the new session or if session is already exist then it will return the same/existing session

* + 1. **Set Value inside session**

**session.setAttribute(“Key”, value);**

In the session you can set the value using above syntax, in this Key will always be in string format and value will be in object format

* + 1. **Get the value form session**

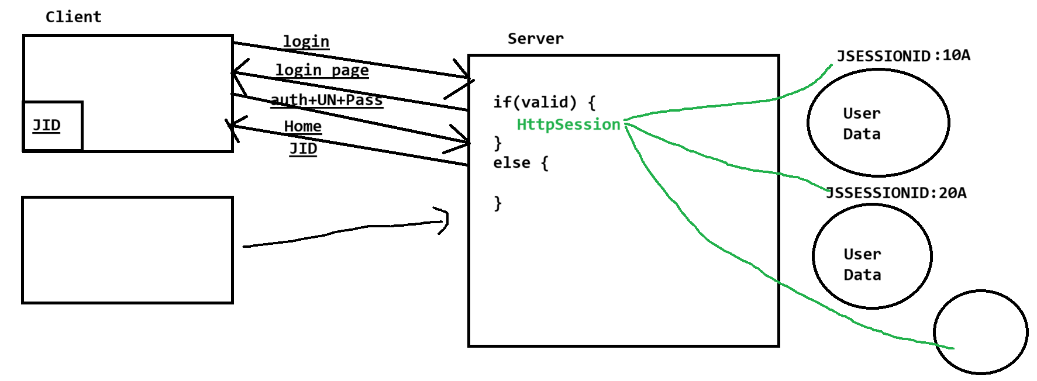
**session.getAttribute(“Key): Object**

From the session you can get the value using above syntax, you have to provide key and it will return you the value in object

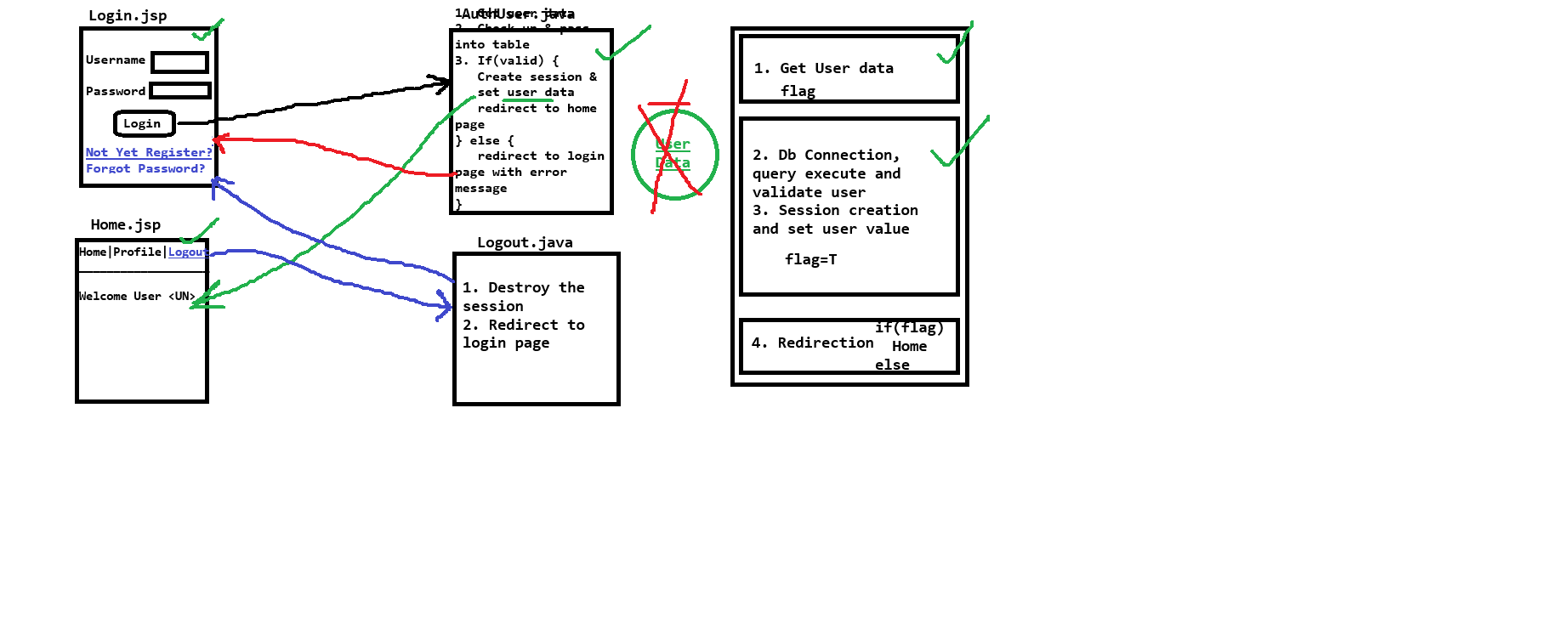
* + 1. **Destroy Session**

**session.invalidate()**

It deleted the session from the server side.

****

**Session Tracking Example:**

****