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

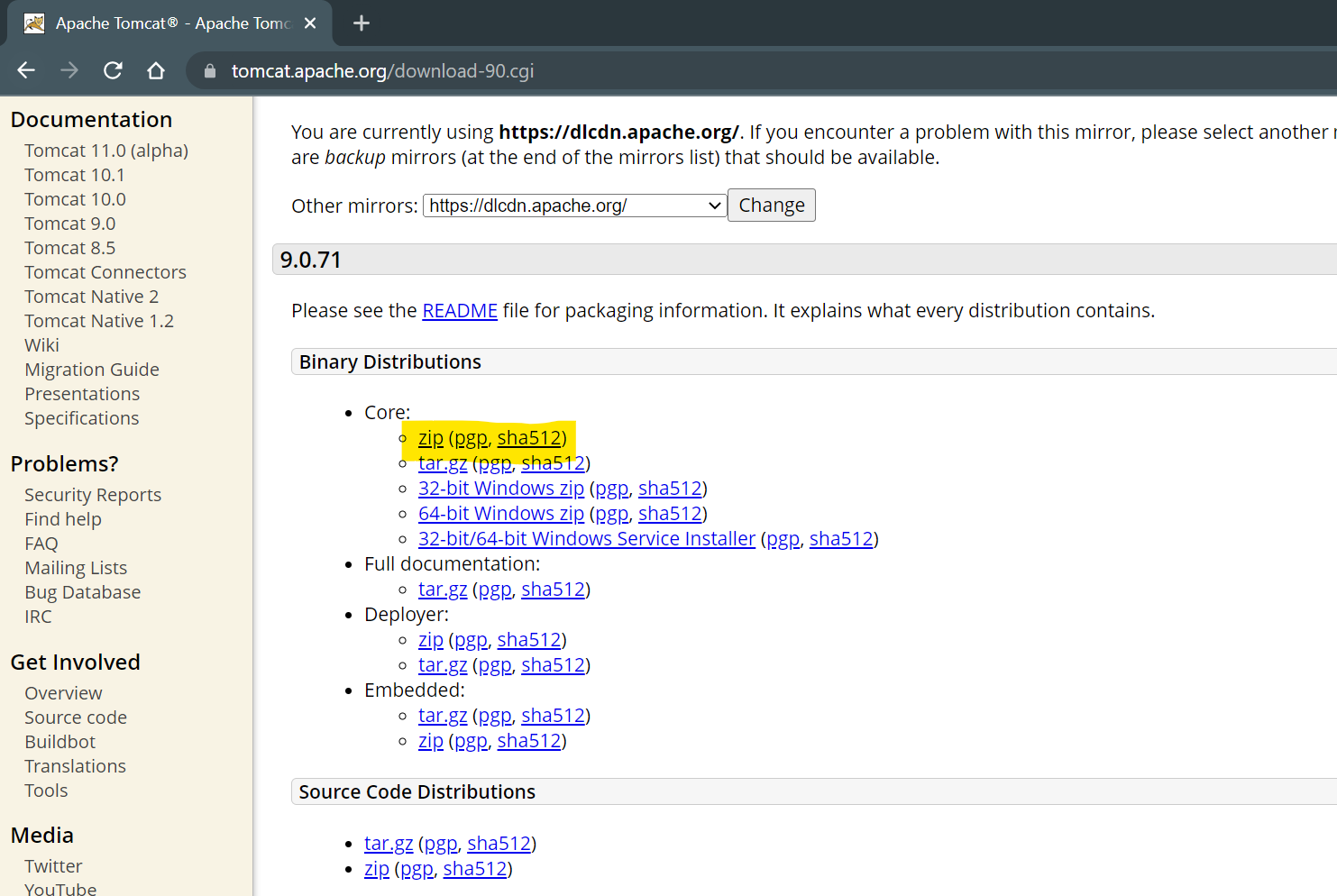
Tech Stack: HTML, CSS, JS, Jquery (client side)

Jsp and servlet (server side)

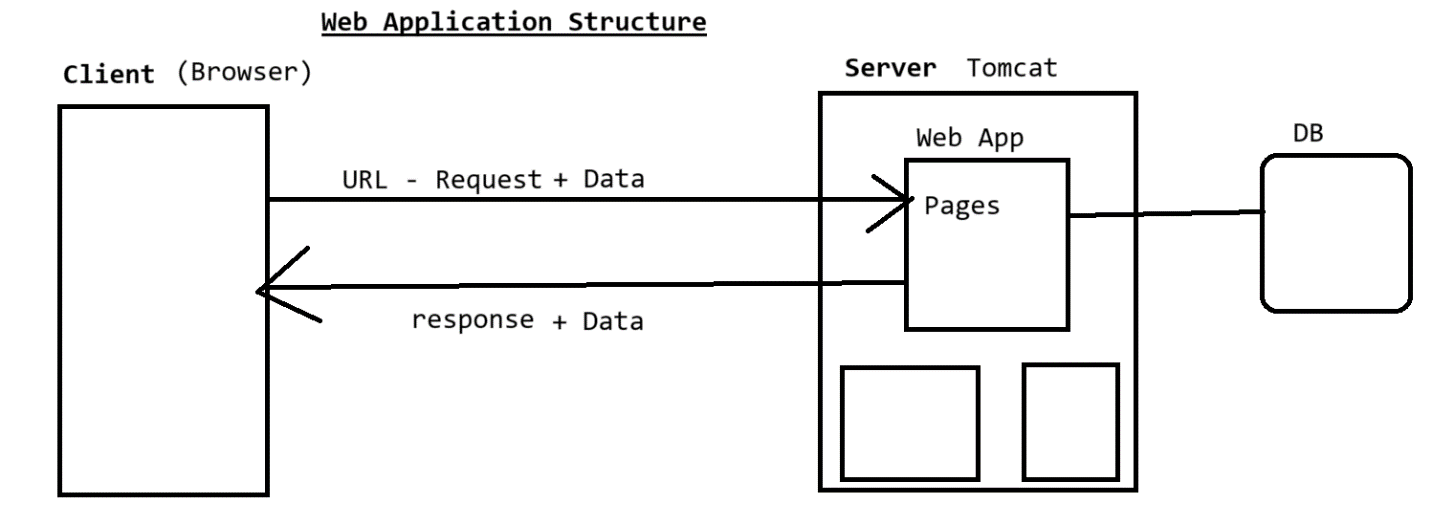
Tomcat Server Setup

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.

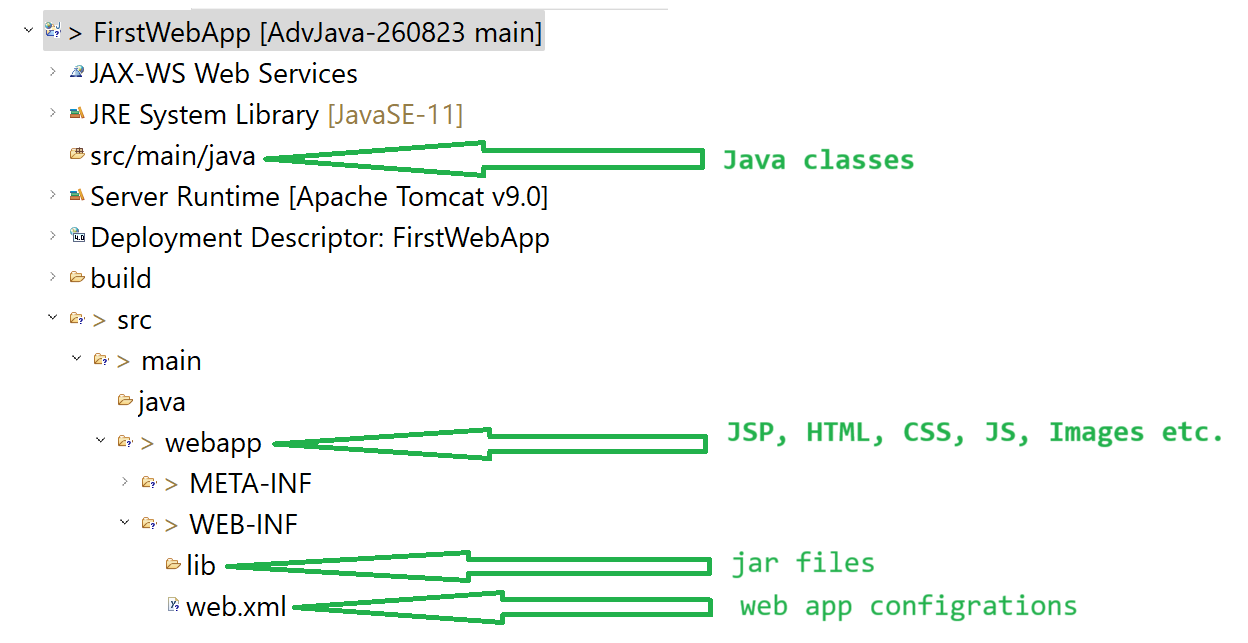
**Parameter**

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>”)**