JDBC

**Externalization of Properties**

1. Create a resources folder inside project
   1. Right Click on Project -> “New” option -> “Source Folder” option
   2. Provide name “resources”
   3. Click on Finish
2. Create a properties file inside resources folder
   1. Right click on the resources folder
   2. “New” Option -> “File” option
   3. File Name with extension “.properties”
3. Create a properties inside the file
   1. Every property must has a key and value
   2. Key and value can be separated by “=” or “:”
   3. Every property must be on new line.

**Java Editions**

JSE

Java Standard Edition (JSE) also known as Core Java

Can develop Console based and Desktop application.

Example: Eclipse, NetBeans, Oracle

JEE

Java Enterprise Edition (JEE) also known as Advance Java

JSP, Servlet, JMS, JSR, WebServices etc.

Can Develop Web Application.

Example: IRCTC, SBI, ICICI etc..

JME

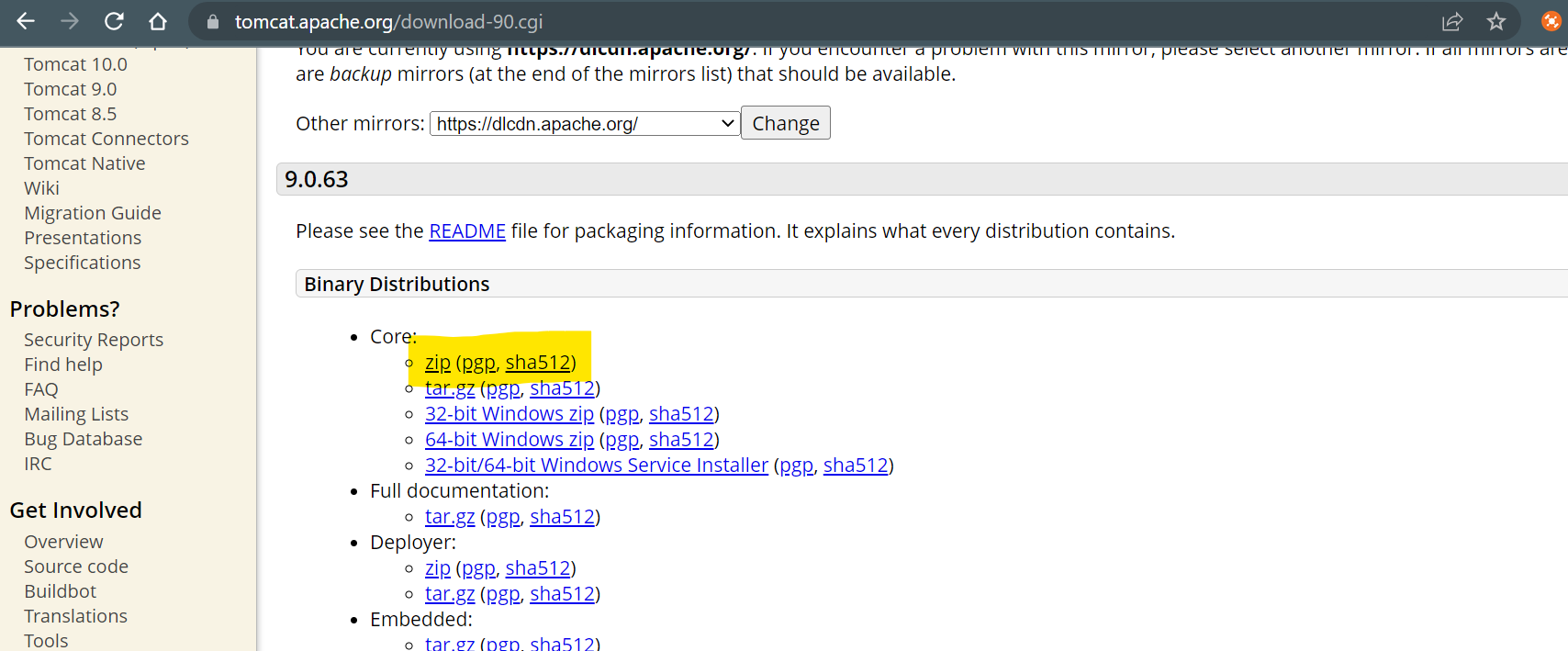
Java Micro Edition (JME).

Use to develop Embedded application, Mobile Application.

**Server Setup**

1. Download Server
   1. Tomcat Server Download

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



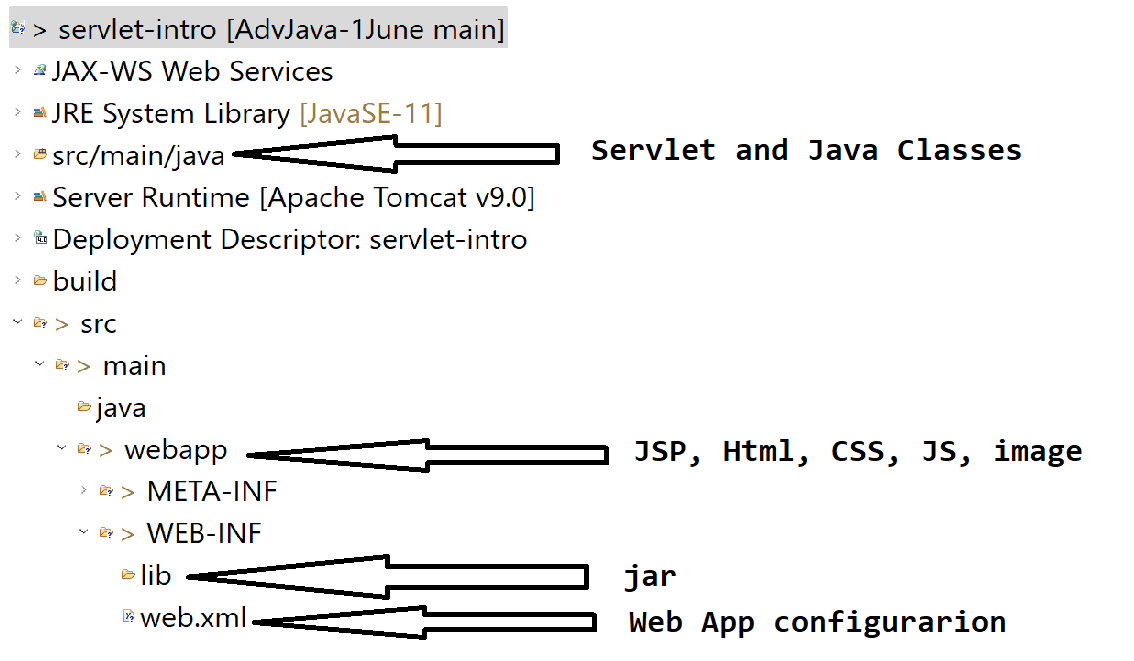
* 1. Extract the ZIP file
  2. To Start server Go to “bin” -> double click on “startup.bat”

1. Configure Server into Eclipse
   1. Open eclipse workspace
   2. Set a Java EE perspective.
   3. Add Server into Servers tab at the bottom of the window
   4. Click On the “Create New Server” link
   5. Expand The “Apache” Option from the new window and Select The downloaded tomcat version
   6. Click on “Next”
   7. Browser the director of the tomcat folder (Select a parent folder of the bin, lib, config etc.)
   8. “Next” and “Finish”
2. Start The Tomcat Server
   1. Right click on Server and click on the start.

**Dynamic Web Application**

1. File -> New –> “Dynamic Web Project”
2. Set a Project name and Make sure than Target runtime must not be “<None>”
3. Click Next - > Next
4. Make sure that select check box for “web.ml deployment descriptor”

**Dynamic Project Structure**



**JSP and Servlet**

1. Used to develop a dynamic web application
2. These technologies are executes at server side.
3. In JSP and Servlet you can get request and generate response

**Servlet**

1. Servlet is a java class.
2. Used to create dynamic Web Page
3. Servlet do not have main method.
4. Servlet are executes by the Servlet container which is a part of server.
5. Servlet can read a request, process a request and generate response.
6. Every Servlet must have a unique URL and it has to register.
7. Servlet Object creation and maintenance will be done by Servlet Container.
8. There is only one object of servlet gets created in a servlet container.
9. Servlet can contains CSS and HTML code along java code.
10. Servlet is also known as HTML in Java.

**How To Create Servlet**

1. There are 3 options to create servlet
   1. By Implementing **Servlet** interface
   2. Extending **GenericServlet** abstract class
   3. Extending **HttpServlet** abstract class
2. Can Override a service method to process request and generate response.

**Register Servlet**

1. By Using XML
   1. This configuration has to do inside web.xml (src/main/webapp/WEB-INF)
   2. XML Tags to configure Servlet

<servlet>

<servlet-name>first</servlet-name>

<servlet-class>FirstServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>first</servlet-name>

<url-pattern>/first-servlet</url-pattern>

</servlet-mapping>

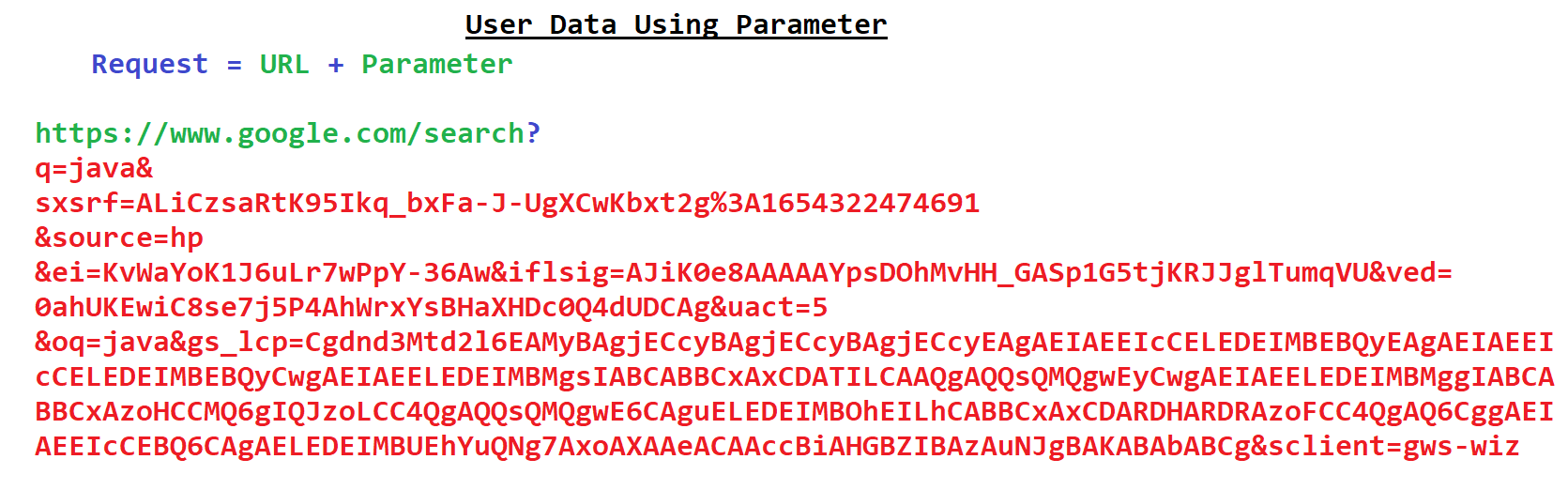
1. Using Annotation
   1. Annotation Configuration can be achieved by using @WebServlet Annotation on the servlet class.
   2. Syntax:

@WebServlet("/<URL>")

**Request and Response**

**Request**

1. Request always goes from Client to Server
2. Request can have a user data.
3. This Data can be pass in form of parameter or request body.
4. Parameter :



**Response**

1. Flows from server to Client
2. Types of response also known as MIME type

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

**Servlet Life Cycle**

1. Manage by Servlet container.
2. There are three stages of the servlet life cycle
   1. **Init stage**
      1. This stage executes when container creates an object of the servlet.
      2. As a part of this stage **init(ServletConfig)** gets invoked.
      3. Every servlet gets instantiated (Object) only once in a life cycle and hence this methos gets called only once.
      4. In this stage you can perform the initialization activity.
      5. This method gets called after constructor.
   2. **Service Stage**
      1. This stage gets executed for every request of the user.
      2. This stage gets executes multiple time in life cycle.
      3. As a part of this stage **service(HttpServletRequest, HttpServletResponse)** method gets invoked.
      4. This stage is use to accept the request, process the request data and generate the response.
   3. **Destroy Stage** 
      1. This stage gets executed only once in a life cycle.
      2. As a part of this stage **destroy()** method gets invoked.
      3. In this stage all the cleanup activities can be perform.

**Http Protocol Methods**

**GET**: to retrieve the information. The User Data Will be pass from the URL in the form of Parameter. All the user Data will be visible into the URL. Default methos is get.

**POST**: this metho is recommended to use for creating new record. The data will be passed in the form request body. The data will not visible inside URL.

**DELETE**: This method is uses to remove the records.

**PUT**: This method is use to update the existing record

**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 Life Cycle**

1. Translation Stage
   1. In this stage jsp gets converted into servlet
   2. .jsp file gets converted into .java
   3. You can get the translated file into following path

<Workspace>\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\<Project-Name>\org\apache\jsp

1. Compilation Stage
   1. In this step the translated page gets compiled
   2. .java file compiles to get .class
2. Init stage
   1. This stage gets executes only once in a life cycle.
   2. This gets executed at the time of object creation of the servlet
   3. As part of this stage **\_jspInit()** gets invoked.
   4. Here, you can perform the initialization activity.
3. Service Stage
   1. This stage gets executes multiple times for every request.
   2. To get request, process the request and generate response.
   3. As a part of this stage **\_jspService(HttpServletRequest, HttpServletResponse)** method gets executed
4. Destroy Stage
   1. This method get executed only once in a life cycle.
   2. As a part of this method **\_jspDestory()** methods gets invoked
   3. In this stage you can perform the clean up activity.

**Scripting Element.**

1. Is use to write a java code on JSP page.
2. Nesting of the tags are allowed.
3. To Achieve the nesting of the tags JSP provided a Break and continue rule.

<% <----- Start of Scriptlet Tag

**int** i = 20;

System.out.println(20\*20);

**if**(i>10) {

%> <----- Breaking the Scriptlet Tag

<%= 20\*20 %> 🡨----- Expression Tag

<% <----- Continue the Scriptlet Tag

}

%> <----- End of Scriptlet Tag

Comment tags

Html Tag Comment

<!-- -->

Scripting Tag Comment

<%-- --%>

**Implicit Object**

1. The objects which are provided on every JSP page are called implicit Object.
2. There are total 9 implicit object

|  |  |
| --- | --- |
| **Object Name** | **Class/Interface** |
| request | HttpServletRequest |
| response | HttpServletResponse |
| session | HttpSession |
| out | PrintWirter/JSPWiter |
| exception | Throwable |
| config | ServletConfig |
| pageContext | PageContext |
| application | ServletContext |
| page | this (current class Object) |

**Note : All these objects are only accessible inside a Scriptlet (<% %>) and Expression (<%= %>) tag.**

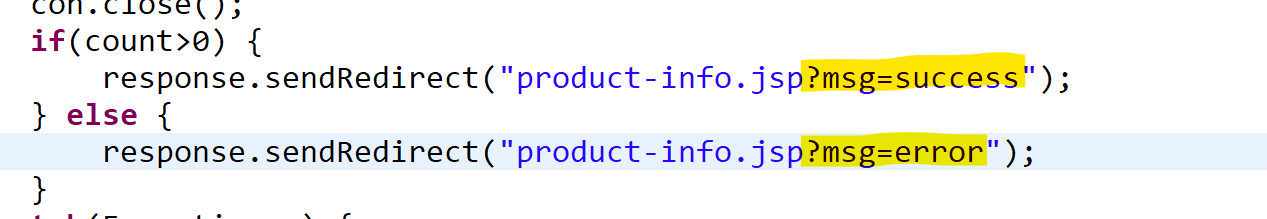
**Session Tracking Techniques**

1. These techniques are used to hold or manager user information (data) into multiple request.
2. There are 4 session tracking techniques
   1. Hidden Form Field
      1. This technique is use to pass the old request data into new request which is generated by form tag and submit button.
   2. URL rewriting
      1. This technique is use to pass the old request data into new request which generated by anchor(<a>) tag and sendredirect.
      2. Example:

Example-1



Example-2

****

* 1. Cookies
     1. Is use to maintain the user details at client side.
     2. Cookies created at the server side and sends at the client side.
     3. The cookies which store at client side will be send to server with every request.
     4. There can be multiple cookies created for a single application. There can be 40-41 cookies store at client side.
     5. Create a cookie and send to Client side

Cookie ck = **new** Cookie(“Name”, “Content”); // To create a Cookie

response.addCookie(ck); // add Cookie into a response

* + 1. To Get the cookies from the request

Cookie[] cks = request.getCookies();

* + 1. Disadvantage of Cookies
       1. Cookies can be disabled at client side.
       2. There is a limit to store the cookies at client side.
       3. Can store only String type of values inside cookies.
  1. HttpSession
     1. Is use to maintain the user details at server side.
     2. In This techniques internally the cookie concept is used.
     3. Create Session

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

The above code is use to create a session. It will create a new session only if it is not already creates else it will return a existing session.

* + 1. To set the values inside session

**session.setAttribute(String key, Object value);**

* + 1. To get the values inside session

**session.getAttribute(String key) : Value**

* + 1. To destroy session

**session.invalidate();**

* + 1. You can set the timeout configuration inside web.xml

**Error Pages in JSP**

1. Can create your custom error Page by following Steps
   1. Create and Design a JSP page
   2. Mark that page as an error page using Directive tag.

<%@ page isErrorPage=*"true"* %>

* 1. Configure the error page in side web.xml

<error-page>

<error-code>404</error-code>

<location>/not-found.jsp</location>

</error-page>