Servlet:-

Servlet we are creating by servlet API.

How to configure Servlet:-

@WebServlet(“/”)

**public** **class** control **extends** HttpServlet {

@Override

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

System.***out***.println("Hello"); //this will print in console

resp.setContentType("text/html");

PrintWriter out=resp.getWriter();

out.println("hello"); //this will print in webpage

}

}

Servlet Life-Cycle :-

-> Below are the steps of Servlet Life-Cycle :-

1. Loading & Instantiation :-

= When the servlet is first requested or when the web application starts, the servlet class is loaded in the servlet container and new instance of that servlet class is created

2. Initialization :-

= After instantiation, the init() method is called by servlet container to initialize the servlet object

= We can override the init() method to perform any one time setup operations for eg. loading configuration data, establish database connection, initializing resources etc

3. Request Handling :-

= Once the servlet is initialized, it is ready to handle the client requests

= The service() method is called by the container for each incoming HTTP request

= The service() method examins the request, determines the appropriate HTTP method (eg GET, POST etc) and deligates the request to the corresponding doXXX() method (eg doGet(), doPost() etc)

4. Response Generation :-

= During the request handling process, the servlet generates a response that may include HTML, XML, JSON etc

= The response will be written to the HttpServletResponse object associated with the request

5. Termination / Destruction :-

= When the servlet container decides to shut down the web application or unload the servlet, it will call the destroy() method

= The destroy() method allows us to perform cleanup operations for eg. closing the database connections, releasing resources etc

6. Servlet Deinstantiation :-

= After calling the destroy() method, the servlet container removes the servlet instance from memory

WEB-XML:-

<web-app>

<servlet>

<servlet-name>ms</servlet-name>

<servlet-class>com.controller.control</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ms</servlet-name>

<url-pattern>/home</url-pattern>

</servlet-mapping>

</web-app>

=> Annotations in Java :

-> Annotations are the metadata (information) which are added to the programming elements i.e. class, interface, method, constructor, variable etc

-> Annotations are used for configuration, documentation and to convey addition information at compile-time or runtime

-> Annotations starts with '@'

-> Categories of annotations :-

1. Marker Annotations

= It is used to mark a declaration. It does not contain any members or data

= For eg. @Override, @Depricated, @FunctionalInterface

2. Single Value Annotations

= In this annotations, there is only one member

= For eg. @SuppressWarnings("unchecked")

3. Full Annotations

= These annotations consists of multiple data members (name-value pair)

= For eg. @Bean(name="----", initMethod="-----", destroyMethod="----")

======================================================================================================

=> Annotations in Servlet :

-> In java servlets, annotations are used to simplify the configuration of servlets and other components in web application i.e. Filters, Listeners etc

-> Annotations provides a way to declare various settings and behaviour in the source code directly, rather than configuring them in web.xml file. This makes our code more concise and easier to maintain

-> Some commonly used annotations in servlet are :-

1. @WebServlet

2. @WebFilter

3. @WebListener

4. @WebInitParam

5. @MultipartConfig

etc

---------------------------------------------------------

=> @WebServlet :-

-> This is an annotation used to define a servlet. We can specify the servlet's name, URL pattern and other configuration settings within this annotation

-> Syntax :-

@WebServlet(name="----", urlPatterns={----})

public class MyServlet extends HttpServlet

{

//override the methods

}

HTTPREQUEST CODE:-

<form action="aaa" method="get"> //When we press loging it will hit on /aaa

            Enter Email :  <input type="text" name="email1" /> <br/> <br/>

            Enter Password : <input type="password" name="pass1" /> <br/> <br/>

            <input type="submit" value="Login" />

        </form>

@WebServlet("/aaa") //when url comes as localhost:8080/aaa this page will open

public class Login extends HttpServlet

{

    @Override

    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException

    {

        PrintWriter out = resp.getWriter();

        String myemail = req.getParameter("email1");

        String mypass = req.getParameter("pass1");

        System.out.println("Email : "+myemail);

        System.out.println("Password : "+mypass);

        out.println("Email : "+myemail+"\n");

        out.println("Password : "+mypass);

    }

}