(1). Java server-side programming ?

**Servlet interface**

🡪 javax.servlet.Servlet; // it is API for server-side programming

Methods:

* Public Abstract Void init(ServletConfig config) throws ServletException;
  + Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
* Public Void service(ServletRequest req, ServletResponse res) throws ServletException, java.io.IOException;
  + Called by the servlet container to allow the servlet to respond to a request.
  + This method is only called after the servlet's init() method has completed successfully.
  + Method of ServletResponse and how to response:
    - req.setContentType(“text/html”);
    - PrintWriter out = req.getWriter();
    - out.println(“<h1>hello world</h1>”);
    - Object “out” is used to give dynamic response to webpage.
* Public Abstract Void destroy()
  + Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
* Public ServletConfig getServletConfig();
* Public abstract java.lang.String getServletInfo();

Notes:- upper three methods are life-cycle methods and other are none life-cycle methods

**Netbeans directory structure**

1. Web pages
   * All static pages are here like html, css, JavaScript etc.…...
2. Source packages
   * All java classes here
3. Libraries
   * All external libraries here like jdbc drive
4. Configuration files
   * It’s made by NetBeans …

**GenericServlet abstract class**

* The GenericServlet implement Servlet, ServletConfig and Serializable interface. It provide implementation all methods of above interface except service method. That’s why GenericServlet called as abstract class.
* GenericServlet class can handle any type of request so it is protocol-independence
* You may create generic class by inheriting GenericServlet and provide implementation of service method

**HttpServlet class**

* HttpServlet class reside in javax.servlet.http;
* It used to create http protocol specific servlet

Methods:-

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

**Life cycle of Servlet**

* load and instantiate
  + Notes:- it’s happen when request first time arrive
* call init() method for initialization servlet
  + Notes:- init() method run at once for one object
* service() method get called for processing request
  + Notes:- service() method called many time for one object
* Before destroy servlet object , destroy() method get called.
  + Notes:- destroy() method called at once for object
* Service() method is responsible for processing logic & This is most important method for Servlet

**Web.xml**

* It is configuration file which is used by server to identify servlet
* Declaration of servlet class
  + One parent tag and two child tag is used for declaration of servlet class
    - <servlet>…….</servlet> //parent tag
    - <servlet-name>……</servlet-name> //child tag
      * In this tag , we have to give name to servlet
* <servlet-class>……</servlet-class>
  + In this tag, we have to give class name with package…..
  + Example:
    - <servlet>

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

<servlet-class>com.RegisterServlet</servlet-class>

</servlet>

* Mapping the servlet class
  + One parent tag and two child class
    - <servlet-mapping>…….<servlet-mapping> //parent tag
    - <servlet-name>………</servlet-name> //child tag
      * The Servlet class can be identified by naming in this tag
    - <url-pattern>……..</url-pattern> //child tag
      * In this tag you can give URL to the class whose name is given in the tag above.
  + Example:
    - <servlet-mapping>

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

<url-pattern>/firsthello</url-pattern>

</servlet-mapping>

* It is deployment descriptor file , It must be named “web.xml”
* It must be in “web-INF” folder.
* A file that contain configuration about our java web application
* What is in the web-inf folder
* <web-app>
* Servlet declaration
* Servlet mapping
* Initialization parameter
* Welcome file
* Filter
* Listener
* Session config
* Etc……..
* </web-app>

**Submit form to Servlet :**

1. Just write “html:5” to create html basic structure
2. <form action=”Servlet-url-pattern-without-slash” method=”post OR get”></form>
   1. Post :- Data arrives in larger quantities and more securely
   2. Get :- Data arrives in limited quantities and less secure
3. Object of HttpServletRequest can used to fetch value from website while client submit their request.
   1. This class has one more method “getParameter(String s)” , which help to fetch data from field which name is same as “s” .
4. Object of HttpServletResponse can used to dynamic response to web page
5. Watch webapplication4
6. **how to fetch image in servlet**

->Part part = request.getPart(string name)

->part.getSubmittedFileName(); return string (name of submitted file)

->part.getInputStream(); return object of input stream

->input.read(bytetypeObject); fill data in bytetype object

->Note:- put @MultipartConfig before user servlet class

Welcome file list(web.xml)

* When your server has first time request then by default one page responced, that’s name is index.html and index.jsp
* In other word, when you called your website directly then one page invoked that’s name is index.html and index.jsp
* By default , index.html is mapped and show as home page .
* Also we can change home page by xml file .
* How it’s work

<web-app>

<welcome-file-list>

<welcome-file>home.html(any html page)</welcome-file>

<welcome-file>home2.html(back-up page )</welcome-file>

</welcome-file-list>

</web-app>

Request Dispatcher

* It is responsible for dispatching the request to another resource it may be html ,servlet or jsp
* Two use ---(1) forward request (2) include response of other servlet
* Example

1. RequestDispatcher rd = request.getRequestDispatcher(“url-pattern”);
2. rd.forward(request,response) ---for forward
3. rd.include(request,response)--- for include

Internal working of Servlet

* we deploy userServlet and web.xml file on server
* After , when any client request to server , server make object of userServlet class based on request.
* Just after creating object of servlet class , server make object of ServletConfig and put information about server specification initialization
* After that, server call init() method and pass object of ServletConfig as argument
* After , server call service() method of userServlet object and pass two object (ServletRequest’s object,ServletResponce’s object) as parameter
* Object of ServletRequest , we can fetch information
* Object of ServletResponce , we can generate dynamic response

**What is parameter and attribute**

Parameter:

* This are those value which is provided by user to any servlet to process request during request operation
* Servlet only can read value for request processing , it can not change value or delete value
* Parameter mostly data send using form , initialization parameter etc.
* How to fetch value from parameter:

String name = request.gerParameter(“name\_of\_parameter”)

(request is object of Request class or HttpRequest)

Attribute:

* These are the object that attached by one servlet to object (session , request,config,context etc) and other servlet can fetch the object to process the logic
* It can be modify , change and remove
* Operation of attribute:
* object.setAttribute(String name, AnyObject value);
* Object value =(Object) object.getAttribute(String name);
* objectremoveAttribute(String name);

Note: object can be session , request, config ,context

Session tracking in Servlet:-

* Session tracking is way to maintain state(data) of user
* It’s also called state management
* http protocol is stateless protocol therefor servlet doesn’t know about previous request and it treat every request as new request (state management problem)
* Session tracking techniques

1. Cookies
2. Hidden form fields
3. URL rewriting
4. HttpSession

Cookies in servlet:-

* Cookies are textual information which is stored in key value pair format to the client’s browser during multiple requests.
* When server process request and give response to client browser , it’s store cookies in client browser.
* Therefor client browser send request to server , with request also cookies will be send , and that cookies server can fetch it and use it.
* How to use cookies in java
* In order to use cookies in java, there is a Cookie class in java present in javax.servlet.http package
* To make cookies just create a object of Cookie class and pass name and its value.
* Cookie c = new Cookie(String name,Object ob)
* How to add cookie to response
* resp.addCookie(Cookie c);
* How to fetch cookie
* Cookie c[] = req.getCookie(); return array of cookie

If there are no cookie found then “c” store null value

* Implementation of cookies in java

1. First create object of cookie class , Cookie c = new Cookie(String name,String value);
2. Add cookie to response , response.addCookie(Cookie c);
3. Fetch cookie from request, request.getCookies(); return Cookie[] array
4. cookieObject.getName(); , return string
5. cookieObject.getValue(); return string