Server side Programming

**Azam Amir Reza (**[**azam20104u@gmail.com**](mailto:azam20104u@gmail.com)**) +917278039915**

**29-Nov-15:**

**Web server**

**Web client :** j

**Web Application**

Resources

Client

Client

Client

Ex, of web server

1. Apache tomcat
2. JBoss
3. Web

**Web Application:**

* Web application is one which can be access over the web

Example Gmail, FB, OLX etc.

* Web application runs on web or app server.
* Client and the server interact using http protocol.
* A server can run multiple applications at the same time.
* Web applications have any web resources. Like music, audio, video, jar file etc.
* Browser is an web client which is widely used and generic and access most of the web application
* A web server listen to http request and processed and response to client

**Types of Resources:**

The resources has to be deployed to the server, there are two types of deployment

1. Manual
2. Automated (Some software like Ant, Maven etc.)

The resources are categories different type that is called **MIME** (Multiple purposes Internet Mail Extension) type

Example of MIME is images/jpg, image/gif, application /jar, application PDF, video/QuickTime, text/html, text /xml

Note:

* An web application can be deploy in the form of jar file
* Multiple applications can be deploy into single server. But every application have unique name

**JEE web application directory Structure:**

App name

UI resource (html,jsp,css,images) public area

WEB-INF

web.xml (deployment descriptor)

lib

.jar files private area

classes

.class file

* Server represent a runtime environment which can only understand .class

**web.xmL/deployment descriptor**:

* It is aconfiguration file which has information like servlet, JSP, security information, welcome file, filters, listeners, session time out etc.
* an application can have maximum one web.xml
* in 2.x all the application configuration used to be done using web.xml when jee 3 version we can use @notation

**Container:**

* Container is an engine or smartness which manages all the resources or JEE components. If any server has JEE container than only we can run JEE application. \*\*\*\*\*
* We can integrate JEE components with many servers.

**Server**

Client 1

JEE container

Servlet cont

Client 2

Client 3

**Responsible for container:**

1. **Life cycle management**
2. **Session management**
3. **Security management**
4. **Etc.**

**Note:**

**Web.xml** **is passed by the container at the time of application loading and if web.xml has any errors then server throws ParseException.**

**All the application is sequentially loaded by the container**

28-Nov-15:

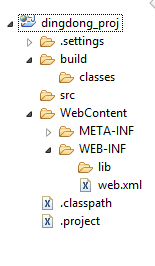
**Installation Process:**

1. Copy apache tomcat file to any drive and then create the environment
2. Fist create a path under the user for apache tomcat like, CATALINA\_HOME is the path name and value should be path location of your apache tomcat, then then create JAVA\_HOME is path name and value should be where is located your java JDK file,
3. And lastly copy the path of your apache tomcat bin and past it in the system path DONE.

**Apache Tomcat Folder and Others:**

* The port no of apache tomcat is 8080
* **Bin:** basically contents startup and shutdown batch file used to start and stop the server
* **Conf:** conf contains the details with respect to server, context and tomcat users. Where the port no can be change
* **Lib:** contains a set of jar files out of which Servlet-API and JSP-API to import few of the properties.
* **Logs:** basically there are some log messages which will always be display on the server consol.
* **Webapps:** basically needed to deploy any of the web application on to the apache tomcat server
* **Work:** basically used to store the translated servlet

01-Dec-15:



**Package + .class files**

**Package + .java files**

**UI resources like html, css, js etc**

**.jar  
creating j2ee project in eclipse:**

1. Open eclipse Java EE prospective
2. Create a new dynamic web project
3. Create web.xml (right click on project java EE tools, generate deployment Descriptor Stub
4. Create appropriate package structure under src.
5. UI resources like html, CSS, etc. must be present inside web content.

**Integrating server with eclipse:**

* Open server tab (window show view servers)
* Click on the link select appropriate tomcat version and click on next
* Brows the tomcat directory and click on finish

**Welcome file:**

* It’s a file or a page which is automatically displayed when client access the application
* By default server takes index.html as the welcome file or page
* We can configure welcome file in web.xml

Change to home page:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app >

<welcome-file-list>

<welcome-file>home.html</welcome-file>

</welcome-file-list>

</web-app>

* When we start the server, all the application are loaded by the container sequentially and at the time of loading container pass web.xml
* If web.xml has any error then container throws ParseException.

**Type of Logic:**

1. Persistence Logic
2. Presentation Logic
3. Business Logic
4. Controller Logic
5. **Persistence Logic**

The logic which is involves in persisting the data into any persistence system is called as persistence Logic, technology involves java IO, hibernate etc.

1. **Presentation Logic:**

Logic which is involves in presenting the contents to the end user is called as presentation logic. Technology involves html, css, jquery, jsp, android UI components, java swing etc.

1. **Business Logic:**

The logic which is involves in developing the core business functionality. Java or any other programming language along with frameworks

Note: generally an application is developed by using all this type of logic.

**Servlet**

* A servlet is a server side java program which performs different types of logic, processes the client request and gives the response to clients.
* Servlet are mange by container(servlet container)
* An application have any no of servlets

Developing server side program: **there are two different way to creating servlet:**

1. **Generic Servlet:**
2. **Http servlet:**
3. **Generic Servlet:**

* javax.servlet.GenericServlet is an abstract class which has only one abstract method by name service().
* Writing a servlet very simply that is write a class which extends GenericServlet and override abstract method called service ().
* Inside the service write the implementation the logic
* Every servlet must have unique URL and servlet can be configuring in two different ways.

1. In web.xml
2. Using @notation.

Note: while creating a servlet program the library has to imported from jar file (CATALINA\_HOME\lib\servlet-api.jar

Ex,

package com.azam.serv\_app;

import java.io.IOException;

import javax.servlet.GenericServlet;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

public class DisplayServ extends GenericServlet {

@Override

public void service(ServletRequest req, ServletResponse resp) throws ServletException, IOException {

//impl

}

}

Ex,

**package** com.azam.serv\_app;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.GenericServlet;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebServlet;

@WebServlet("ds")

**public** **class** DisplayServ **extends** GenericServlet {

@Override

**public** **void** service(ServletRequest req, ServletResponse resp) **throws** ServletException, IOException {

//impl

String htmlOutPut="<html><body bgcolor='yellow'>"+

"<font color='blue' size='6'>"+

"Welcome Guldu"+

"</font>"+

"</body></html>";

System.***out***.println("---------------Hello----------");

PrintWriter out=resp.getWriter();

out.println(htmlOutPut);

out.close();

}

}

**Servlet Mapping:**

Ex,

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head> url pattern of servlet

<body bgcolor=*"cyan"*>

<h1>

<a href=*"ds"*>click</a>

</h1>

<p></p>

<form action=*"ds"*>

<input type=*"submit"* value=*"click"*>

</form>

</body>

</html>

**Important Method of Servlet:**

**public** **void** service(ServletRequest req, ServletResponse resp)

public PrintWriter getWriter()

init(ServletConfig);

destroy();

getServletConfig();

getServletcontext();

getParameter(String Key);

getInitParameter(String key);

**UI OR FORM DATA:**

* Is the data which is entered by the user, it is always in the form of key value pair
* Multiple form data can be present in a request
* Form data is in the form of associated data.

Name :

Age :

Sex :

* UI of form data is associated with request object
* UI of form data can only we access with in a service() method or in doxxx() method
* In service() method we can fetch UI form data using below method:

+String getParameter(String key)

* If the given key is not present then method returns null.

Ex,

String id=req.getParameter(“id”);//25

Int actId=Integer.parseInt(id);

String name=req.getParameter(“nm”);//shishira

04-Dec-15:

Ex of ui name

--

GenericServlet is an abstract class which has core abstract as well as concrete method

**Life cycle method of servlet:**

1. + init(ServletConfig)//concrete method
2. + void service(ServletRequest, ServletResponse)//abstract
3. + void destroy()//concrete method

Container invoke🡪

1. Create an object of servlet(default constructor)
2. Init(config)[initialization]//cm
3. Service//abstract method it call multiple
4. Destroy//cm

Note;

* init and destroy is called only once and service is called multiple time
* All the life cycle methods are control or invoke by container.

07-Dec-15:

**Servlet Life Cycle:**

* Event of the phases that servlet come-cross from its object creation till the object destruction
* Below are the life cycle phases:

1. Instantiation Phase//object creation
2. Initialization phase
3. Service
4. Destruction

**1. Instantiation Phase:**

* Default constructor is responsible for this phase
* This phase is with instantiation a servlet or creating an object servlet.
* Out of multiple request , the fisrt client req comes for servlet and containers loads the servlet and create only **one object servlet and** this happen by calling the **default constructor.** If container failed to instantiate and throws ServletException. And the root cause is InstantiationException.

**Note:**  we must not write any custom constructor servlet.

**2. Initialization phase:**

* This phase is responsible for initializing the servlet.
* init() method responsible for this.
* Container creates only one implementation object of type servlet config and pass this as an argument to init () method and then initialize the servlet.
* Init() method is invoke only once.
* If this phase fails then container throws ServletException.

Ex,

+void init(ServletConfig config){

//initialization

}

InboxServlet ibs=new InboxServlet();

Ibs object have its own config object.

Note: every servlet object is associated with config object.

**3. Service Phase:**

* This phase is responsible for giving the services to client
* Service() method is used to give the response
* Container executes service () for every client request. That is multiple client are making request, service method is executed multiple times
* Service() method is **multi-threaded**
* If this phase fail then container throws ServletException

Note: service can be made single thread or synchronized in two different ways:

1. Make the service () method as Synchronized
2. By the Servlet class implements Marker Interface by name **SingleThreadModel.**

**4. Destruction phase:**

* Destruction phase is responsible for closing the costly resources and destroying the servlet object
* Before stopping the server container invoke destroy method and closes all the costly resources, and then the servlet object is destroy by the servlet container hence the life cycle ends
* Destroy () is invoke once

**Conclusion;**

The entire life cycle of servlet

1. Only one object of servlet and config is created
2. Init() and destroy() are invoke only once
3. Service () is invoke multiple times for every client request.

**Load and StartUp:**

Generally a servlet is instantiated and initialize when the first request comes. From the 2nd and subsequent request onwards directly service page is executed hence there is a delay time for the first client request.

The delay time can be avoided by configuring load and start up

Load and start up is an integer value which indicates the container to instantiate and initialize the servlet onStartUp of the server without waiting for the first client request

Priority is given based on the lowest integer value.

@WebServlet(urlPatterns=”/life.php”,loadOnStartup=1)

Public class LifeServ extends GenericServlet{

}

08-Dec-15:

Every servlet can be configure in web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app >

<servlet>

<servlet-name>life</servlet-name>

<servlet-class>com.azam.lifeapp</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>life</servlet-name>

<url-pattern>life.php</url-pattern>

</servlet-mapping>

</web-app>

While configuring servlet in web.xml servletname, class, and url pattern mendatory. But load and startup is optional.

**Initparameter:**

* Is a data in the form of key and value pair used to initialize a servlet through init() method
* Key must be unique and value can be any string data.
* Init parameter can be configured in web.xml and also annotation.

Example, in web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app >

<servlet>

<servlet-name>life</servlet-name>

<servlet-class>com.azam.lifeapp</servlet-class>

<init-param>

<param-name>dbd</param-name>

<param-value>com.mysql.jdbc.Driver</param-value>

</init-param>

<init-param>

<param-name>un</param-name>

<param-value>root</param-value>

</init-param>

<init-param>

<param-name>pwd</param-name>

<param-value>dinga</param-value>

</init-param>

<load-on-startup>2</load-on-startup>

</servlet>

</web-app>

In @notation:

@WebServlet(urlPatterns=”/life.php, loadOnStartup=1,initParams={

@WebInitParam(name=”dbd”, value=” com.mysql.jdbc.Driver”}

@WebInitParam(name=”un”, value=”root”}

@WebInitParam(name=”pwd”, value=”dinga”}

* The init parameter is associated with the config object can be fetched using bellow method:

String driver=config.getInitParameter(“dbd”);

String user=config.getInitparameter(“un”);

Note; if key is not present then we get null value.

**Servlet Config:**

* is a inter face present in javax.servlet package
* An implementation object (config) object is created by container immediately after servlet object creation.
* For every servlet object, a config object is created that is config object is associated with servlet object.
* The config object can store any no of initParameters.

q. what is diff bet config and context

q. what is web.xml

q. diff bet initparameter and contextParameter

q. can we access config other then init(). Ans is yes

note: it is possible to access config even out side the init() method that is in service and destroy() method.

+ServiceConfig getServletConfig();

Ex,

ServletConfig config=getServletConfig();

2nd Batch:

**Servlet context:**

* Servlet context is an interface and implementation object is created by container at the time of **application loading** only one context object is created for an application. Present in javax.servlet package
* The scope of context object is throughout the application
* Context is also referring as environment which is created for an application.
* Context object can store context **parameter** as well as **attributes**
* We can get the reference of context object by using below method

+ ServletContext getServletContext();

Ex, ServletContext ctx=getServletContext()

* getinitParameter() is the method used to fetch context parameters.

Context parameter

Attribute

ServletContext ctx=getServletContxt();

String name=ctx.getInitParameter(“nm”);

String age=ctx.getInitParameter(“ag”);

Int age=Integer.parseInt(age);

**Context parameter:**

* Context parameter is a key and value pair which is associated with context objects.
* Key must be a unique string and value can be any string data
* Context parameter are declared in web.xml and they can’t be declare using @notation
* Context parameter must be declare web.xml outside the servlet tag

<web-app>

<context-param>

<param-name>nm</param-name>

<param-value>azam</param-value>

</context-param>

<context-param>

<param-name>pls</param-name>

<param-value>bengalore</param-value>

</context-param>

**Attribute:**

1. **public void setAttribute(String name,Object object):**sets the given object in the application scope.
2. **public Object getAttribute(String name):**Returns the attribute for the specified name.
3. **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters as an Enumeration of String objects.
4. **public void removeAttribute(String name):**Removes the attribute with the given name from the servlet context.

* Attribute is the data in the form of key and value pair.
* Key must be a unique String and value can be any object.

Data

K=v

String=object

1. String
2. Pen
3. Student
4. Array
5. Col obj(List Set)

bk

Color

Price

lps

Pink

closy

* An attribute can be set request, context, and session object.
* Data can be added in the attribute by using below method;

+void setAttribute(String key, Object value)

+ Object getAttribute(String key)

Ex,

Product pd=new Product();

pd.prc=240;

pd.clr=”yello”;

pd.name=”bucket”;

//set the attribute

ServletContext ctx=getServletContext();

ctx.setAttribute(“bkt”, pd);

//get attribute

Product p=(Product)ctx.getAtrribute(“bkt”);

1. Config object is associated with servlet and context objet is associated with application
2. Config object is only one for one servlet and context is only for an application
3. Config object is created after servlet object creation but context object is created at the time of loading an application (before servlet object creation)
4. Config can store only initParameter(can’t store attributes) but context can store contextParameter and attributes.
5. Scope of config is servlet scope and context is for application(the highest Scope)
6. Modification made to the data of config as a impact only on the servlet but modification made to the data of context has an impact on the entire application

08-Dec-15:

Interview Question:

Q. what are the types of http request?

Ans. **GET**, **POST**, PUT, DELETE, OPTOIN, HEAD, CONNECT, TRACE

Q. Different between GET and POST?

Ans. ------

q. write servlet hierarchy?

q. why http servlet is abstract when all its methods are concrete?

* In web application client and server are using http protocol
* Below are the types of http request.

**GET**, **POST**, PUT, DELETE, OPTOIN, HEAD, CONNECT, TRACE

**GET** :

* Get request is used to get the content from the server
* It is read only request
* Get request is deals with limited data. There is 1024 characters
* In case of get request data appear in the browser URL hence it is not secure
* Get request is Idempotent.
* Get request can be bookmark

Example postal card

**POST:**

* The post request is used post some content from client to server. Example uploading video, resume, login, registration etc.
* Post is not Idempotent
* In case of post any no of data can be handled
* Post can’t be bookmark
* In case of post the data is not visible in the browsers URL, hence it is secure.

**Servlet hierarchy:**

<<interface>>

ServletConfig

getinitParameter(String)

getServletContext()

<<interface>>

Servlet

Init(ServletConfig)

Service(ServletRequest, ServletResponse)

destroy()

getServletConfig()

Implements

<<abstract>>

GenericServlet

Only one abstract method

Not specific to any protocol service()

specific to http

HttpServlet

# void doXXX(HttpServletRequest

HttpServletResponse)

* Http servlet has different doXXX() method for all the types of Http that is get has doGet() and post has doPost() but connect doesn’t have doConnect().

Ex,

# void doGet(HttpServletRequest, HttpServletResponse)

# void doPost(HttpServletRequest, HttpServletResponse)

# void doPut(HttpServletRequest, HttpServletResponse)

**2. Http Servlet:**

* httpServlet is an abstract class present in javax.servlet.http package.
* HttpServlet is very specific to http protocol
* All the methods are concrete. But still the class is abstract.
* It extends generic servlet and service method is overridern.
* For all the type of http request we have corresponding doXXX mehod

**Writing HttpServlet:**

1. Write a servlet class which extends HttpServlet
2. Override appropriate doXXX() method

Note; if you are not sure type of request then we can override doGet() as well as doPost() method

Click on link that always get request by default, if the form type is not mentioned

**Servlet Chaining:**

* Generally on the request comes from servlet, the servlet is executed and give the response to the client. But instead of responding the servlet can communicate with other resources like html, jsp, or servlet. This way of servlet to resource communication is called servlet chaining.

Html

Servlet

Client

Jsp

Servlet

**Servlet chaining can be done by using**

1. RequestDispaatcher
2. sendRedirect

**sendRedirect:**

* sendRedirect is a method which is declared in httpServletResponse interface
* **This works at client side**, that is redirection happen to an external application in the browser.
* sendRedirect is used we change from servlet to an external application.
* While redirect to external application, the type of request is **get**.

**RequestDispatch:**

* RequestDispather is an Interface present in javax.servlet Package
* RequestDispatcher is used for servlet chaining by dispatching the request from one servlet to other resource.
* We can get an implementation object of type RequestDispather by using below method;

+RequestDispatcher getRequestDispatcher(String resource)

Example,

+ RequestDispatcher rd=req .getRequestDispatcher(“resource”);//res are like name of html,jsp or servlet

+ RequestDispatcher rd=ctx .getRequestDispatcher(“resource”);

* Request.getRequestDispatcher helps to communicate with resource of the same application
* Context.getRequestDispatcher helps to communate with the resource of the other application

**There are two important method of RequestDispatcher**

1. +void include(ServletRequest, ServletResponse)
2. +void forward(ServletRequest, ServletResponse)

12-Dec-15:

Note:

RequestlDispatcher works a server side

**Include ():**

* Include() method include the responses of the other resources into a servlet
* Include() method works like response merger
* Any no of inclusion can be done that is multiple inclusions are possible.
* We can include resource like html or servlet or JSP.

Forword():

Server

req

Post

Nm=guldu

Ag=20

Client

Name:

Age:

Gender:

/s1

Servlet

* Forward () is used to forward the existing request from one servlet to other resource.
* From one servlet logically only one forward is possible that is once.
* Once the request to forward s other resource we not came back to servlet again
* In case of forward the **data** and the type of request which is available for the first servlet is also available for the 2nd resource.
* If the type of req is get for the first servlet then for the 2nd servlet also type of req is get
* We can pass the data from one servlet to other servlet by using request.set attribute and request.get attribute.