Servlets is one of the web technology developed by SUN Microsystem. The basic aim of servlets is to develop **Web Application.**

**Internet:** Collection of autonomous computer interconnected and connected to server software which is available in particular location to getting resources that is called internet.

**Browser:** A **web browser** is a [software application](http://en.wikipedia.org/wiki/Software_application) for retrieving, presenting, and traversing information resources on the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web).

**WWW:**  www full form is World Wide Web**.** It meanscollection of information in the form of web documents we access those documents by using web browser.

**Web Application:** A web application is a Collection of static web resource programs and dynamic programs.

Static/client side web resources

Dynamic/server side web resources

**Server softwares:** The basic aim of server software is to provide concurrent access. Based on the place where the programs of web applications are running by residing at Serverside, those programs are divided into two types they are

1. Static web resource programs
2. Dynamic web resource programs

**Static web resource programs:** These are those which are residing at side and running in the context of client side browser.

The basic role of static web resource application development is to accept the data from client side/browser side. Since static web resource programs are running in the context of client/browser they are also known as client side technologies.

**EX:** HTML, DHTML, Javascript, vbscript, e.t.c.,

**Dynamic web resource programs:** These are one which are residing in the context of server and running in the context of server.

The role of dynamic web resource programs in the web application development is to validate the data which is brought from client side by static web resource programs.

**EX:** Servlets, JSP(java based),ASP,ASP.NET(Microsoft), CGI(Open community).

Since dynamic web resource programs are running in the context of server, they are also known as server side technologies.

**Need of web application development:** With the concept of J2SE we are able to develop the following applications and unable to develop some other applications also.

1. JSE allows us to develop standalone applications and unable to develop distributed applications.
2. With JSE we are able to develop only intranet applications but unable to develop internet applications.
3. With JSE we are able to get one to one/half duplex/walky-talky applications but unable to develop one to many/ full-duplex applications.
4. We are able to develop only two tier applications(Client program, Serverside program) but unable to develop 3 tier applications(Client ,server side programs and Database software) hence the above limitations makes us to go for another application development called web application/distributed application by making use of Serverside technologies.
5. All JSE applications runs in the context of local disk and provides local access and they are unable to run in the context of browser/server/www and unable to provide global access.
6. JSE applications are not getting explicit support from universal protocols such as HTTP, FTP etc.,
7. JSE applications are not getting any predefined support from third party server softwares such as Tomcat(Apache Jakartha software Foundation), weblogic(BEA), Websphere(IBM) e.t.c.,
8. All JSE applications provides**language dependency.** That is a part of network programming of JSE, both client and server side programs written language Java. But we are unable to develop **language independent applications**(Client side programs can be written in any language and serverside program can be always return in Java) with JSE(including JDBC)\

**Architecture or setup required for development of web application:** According to industry standards for the development of web application and distributed applications we require to follow the following architecture which is prescribed by SUN Microsystems.

Physical(server) Machine

**url**

**Client**

4. Server software

6.1

6.2

**Firewall**

5. Web application

req

Middleware Services 10

res

Tier-1

View Layer

Database Software

Tier-2 Controller Layer

Tier-3

.. Data Layer

**Step:**

1. The internet user decides which web application to users.
2. The internet user makes a request to the web application. Whatever the request made by the internet user, over the network the request reaches to physical machine where the web application is running.
3. The physical machine receives the client request and pass it to server at which web application is running.
4. If physical machine contain multiple servers which are running mode and they can be differentiated on their port numbers.

**EX:** Tomcat🡪 8080

1. The server software receives the request through the physical machine and handover to the appropriate web application then the server software may contain many number of web applications and then can be differentiated based on their names. In servers the imposed rule is each and every web application must contain unique names.
2. 6.1, 6.2 Internet user makes a very first request to the server software always executes the static web resource programs for giving homepage or starting page from second to further subsequent results, most of the times the server softwares executes dynamic web resource programs. In the most of the server softwares servlets are always residing in server software and executing in the context of CATALENA of server software and JSP also executes in the JASPER in the server software. CATALENA and JASPER are called **Containers.** After completion of 6.1 and 6.2 the server may give response back to the client or interact with universal Databases.
3. The dynamic web resource programs(Servlets, JSP) sends a request to the Database for obtaining the data from Database software.
4. The Database software receives the request from dynamic web resource programs and executes.
5. After execution of dynamic web resource programs request by the Database software, the Database software gives it result back to dynamic web programs.
6. Dynamic web resource program receives the result from Database software and gives the dame result back to client browser
7. The browser software displays the received result form the web application on the console. This type of architecture pattern is also known as MVC(Model, View, Architecture)

**Middleware:** Servlets are those which are provided by third party server softwares for effective execution of web application in the context of server softwares.

**EX:** Connection pooling service.

Security services, Transaction services(ACID Properties) etc., known as middleware services.

**Technologies required for web application/distributed application:**

1. List of Browser softwares
2. List of server softwares
3. List of web technologies
4. List of Database softwares

1. **List of Browser softwares:** The following table gives browser software name and whose name corresponding vendor name.

|  |  |
| --- | --- |
| **Browsers** | **Vendors** |
| 1. Internet Explorer 2. Netscape Navigator 3. Firefox 4. Google Chrome 5. Opera 6. Hotjava | Microsoft  Netscape  Mozilla  Google  Opera soft  Redhat |

1. **List of server softwares:**

|  |  |
| --- | --- |
| **Webservers** | **Vendors** |
| 1. Tomcat 2. JWS(Java webserver(outdated)) 3. Weblogic(both web, application server) 4. Jetty 5. Resin 6. Personal web server(non Java) 7. IIS(internet information server) (non Java) | Apache Jakartha software Foundation  SUN Microsystem  BEA(Oracle)  Adobe  Resin soft  Microsoft  Microsoft |

|  |  |
| --- | --- |
| **Application Servers** | **Vendors** |
| 1. Weblogic 2. Web sphere(Purely application server) 3. Oracle Application Server 4. Glass Fish 5. JBOSS 6. JRUN | BEA(Oracle)  IBM  Oracle  SUN Microsystem  Apache Software Foundation  Macromedia |

**Conclusion:** The functionalities of application servers are more than the functionalities of webserver. Every application server can act as a webserver but a webserver can’t act as Application server.

1. **List of web technologies:** Based on the web application development, web technologies classified into two types those are
2. List of clientside/static web resources technologies
3. List of Serverside dynamic web resource
4. **List of clientside/static web resources technologies:**

|  |  |
| --- | --- |
| **Client side technology name** | **Client side technology vendor name.** |
| HTML  Javascript  VB Script  AJAX(Asynchronous Java script and XML processing) | W3C (World wide web consortium)  Netscape  Microsoft  Adaptive Path |

1. **List of Serverside dynamic web resources**

|  |  |
| --- | --- |
| **Server side technology name** | **Vendor** |
| 1. CGI(Common Gateway interface)   (non java)   1. Serverside Java script(non Java) 2. Cold Fusion(non Java) 3. PHP(Personal home page and hypertext pre processing)(non Java) 4. ASP, ASP.NET(non Java) 5. Servlets, JSP(java) | Open Community  Netscape  Adobe  Apache  Microsoft  SUN Microsystem |

1. **List of Database softwares**

|  |  |
| --- | --- |
| **Database** | **Vendor** |
| 1. Oracle 2. SQL Server 3. MySql 4. Sybase 5. DB2 | Oracle Corporation  Microsoft  SUN Microsystem  Redhat  IBM |

**CGI:** CGI is one of the specification released by open community for the development of web applications.

1. CGI is not a language but it is one of the technology/specification
2. We know that a specification is nothing but set of rules and guidelines developed by vendors for making the developers to develop web applications
3. In earlier days for development of web applications we have first specification called CGI and it was implemented by various developers by making use of the language like C, Perl, C++, Shell script, VB etc.,
4. In those days CGI based web applications are not become that much popular because it is having various limitations because of their implementation languages.

**Limitations:**

1. CGI based web applications possesses platform dependency because whose implementation language is C.
2. CGI is one of the **Process based server side** technology that is in the internet if any user makes a request to the CGI based web application then the server loads(swapping) the CGI application in the server memory and creates a process, process the client request and gives response back to the client. Once response is completed CGI based application swapped out from memory this process is applicable for all the internet users whoever makes a request to the CGI based web application.
3. All the CGI based web applications are treated as **heavy weight components.** Because creation of process will take more amount of time and results in poor communication.
4. In process based applications **Context switch is more** because switching the control one process to another process is taking more time.
5. All CGI based web applications are by default **unsecured.** Because whose implementation languages are not providing inbuilt security. Hence unsecured web applications leads to unauthorized modification.
6. CGI based applications possesses **low performance** because in bottle neck time in the internet many number of users are accessing the CGI based web application then the server memory contains many **number of processes** created for the same CGI based web application.
7. CGI based web applications are not scalable(Scalability=fast in communication+higher performance+ Consistent result)
8. All CGI based web applications are **non portable**because CGI based web applications to be configurable separately as and when Operating system is changing, server is changing and process architecture is change.
9. All CGI based web applications are **inefficient** compare to servlets to eliminate all the above drawbacks of CGI based web applications we need to use an another Serverside technology called Servlets.

**Servlets:** The basic aim of servlets is to develop web applications.

1. Servlets is one of the dynamic web technologies developed by SUN Microsystem.
2. Since servlets comes under technology SUN Microsystem has released specification for servlets which makes us to understand how to develop the strong web applications,
3. We know that specification is nothing but set of rules and guidelines and they will be given in the form of purely **abstract methods** which are present in **interfaces.** That is all the interfaces related servlets specification developed by **SUNMicrosystem.**
4. Servlets are meant for development of web applications and they are running in the context of server, all the server vendors are come forward and developed implementation for the interfaces and given in the form of classes are developed by server vendors.
5. Hence servlets API is a collection of packages which are classes, interfaces and sub packages. All these interfaces are developed by SUN Microsystem and whose corresponding classes are developed by server vendors.
6. In order to deal with servlet programming we must import the following two packages
7. javax.servlet.\*
8. javax.servlet.http.\*

The above packages are come along with the server software **in the form of jar file** but not coming along with JDK software. For example in Tomcat server the above packages are available in the form of servlet api.jar and it presents in C:\Tomcat5.5\common-lib folder.

1. Similarly in weblogic server software the servlet api available in the form of **weblogic.jar** and it present C:\BEA\weblogic 9.0\server\lib folder etc.,
2. Hence to compile any servlet program it is mandatory to the servlet developer to set the class path for any of the above jar files.

**EX:** set classpath=C:\tomcat5.5\common\lib\servlet-api.jar;.;

(Or)

setclasspath=C:\BEA\weblogic9.0\server\lib\weblogic.jar;.;

**Servlet:** A servlet is one of the Serverside, dynamic, platform independent multi threaded Java program which is running in the context of server by extending the functionality of either web server or application server.

(or)

Single instance multiple thread technology is known as servlet.

**Advantages of Servlets over CGI:** When we use servlets for the development of web applications instead of CGI we get following advantages.

1. Platform independent
2. Thread based server side technology
3. Light weight component because threads are use for establishing the communication between internet user and server based web application
4. Secured
5. Higher performance is achieved.
6. Portable.
7. Scalable components.

**Servlet hierarchy chart:** The following hierarchy chart gives number of approaches to develop servlet based web applications.

**javax.servlet.Servlet**

**javax.servlet.GenericServlet**

**javax.servlet.HttpServlet**

User defined class

**OurServlet**

The above hierarchy chart makes us to understand the flow of methods inherited from top most base interface to intermediate base classes and from intermediate base classes to bottom most users defined derived classes.

ways to dev servlet?

3ways we can dev servlets:

> by using Servlet interface

> by using GenericServlet class

> by using HttpServlet class

how many ways to config servlet?

we can config servlet in 2ways:

> using xml file

> using Annotation

how many ways to deploy servlet app?

there 3ways to deploy servlet app:

> man deploy

> auto deploy(IDE based)

> tool based (live)

how many ways to design servlets?

> man approch

> ide/auto approch

**javax.servlet.Servlet (1st approach)**

Servlet is one of the predefined interfaces which are available on the top of hierarchy of all the interfaces in servlet programming.

The heart of Servlet architecture is the Servlet interface, it provides the framework for all servlet.

**This interface contains the following five methods:**

1. **public abstract void init(ServletConfig);**
2. **public abstract void service(ServletRequest, ServletResponse) throws ServletException, IOE;**
3. **public abstract void destroy();**
4. public abstract ServletConfig getServletConfig();
5. public abstract String getServletInfo();

**javax.servlet.GenericServlet (2nd approach)**

GenericServlet is one of the predefined abstract classes which are used for developing protocol independent servlet. This class is one of the universal base-class for all type of servlet which are coming in the near future depends on type of protocols. GenericServlet is implementing Servlet interface and defined all the methods except **service()**. So that the current GenericServlet class made as abstract.

**javax.servlet.http.HttpServlet (3rd approach)**

Here http is one of the sub package of servlet package this package(http) is compatible with http protocol which is currently used in internet world. HttpServlet is predefined abstract class which is used in real industry for the development of protocol dependent servlet. Industry is highly recommended to write protocol dependent servlet only, but not protocol independent servlet because the current internet world is using http protocol. Programmatically **HttpServlet is extending GenericServlet.**

**OurServlet**

It is user defined class (servlet) which will extend either GenericServlet or HttpServlet, or implements Servlet interface.

**Life cycle or loop back methods**

Life cycle methods are those which are automatically executed by either web server or application server.

Since servlet are running in the context of web server or application server, the concept of servlet contains life cycle/loop back methods.

**Servlet contain three life cycle methods, those are:**

* init()
* service()/doGet()/doPost()
* destroy()

**public abstract void init(ServletConfig)**

this method defined in GenericServlet and further inherited into HttpServlet. **This method called by the server only once whenever we make very first request to the Servlet based web application**. Since this method is calling only once, as a servlet developer **we always write one time operations such as opening the file, obtaining the Database connection, initialization of parameters etc**., in general in init() we always write the block of statements which illustrate or performs resources gathering logic.

**service()/doGet()/doPost()**

* this(service()) method available in Servlet interface and it is undefined in GenericServlet further inherited into HttpServlet.
* After execution of init(-) at the time of making first request, service()/doGet/doPost will be called by the Container of servlet.
* From second request to further sub sequent requests container will call directly only serving method.
* Since this **method is calling every time, as a Servlet developer we need to write the block of statements which will perform repeated operations such as reading data from file, Database operations, modification of initiated parameters, business code etc…**
* In general service() contains the block of statements which will provide **Request processing logic** or **Response production logic**. When the container creates one Thread and it will be live until response back to the client. That is in the beginning of service() execution thread creation will be taken place and after completion of service() thread is destroyed.

**public abstract void destroy()**

* This is method of Servlet interface and defined in GenericServlet and further inherited into HttpServlet.
* **This method will be called by the container/server provided the servlet/web application related to servlet removed from context of server.** We are no longer to execute in the context of server as part of destroy().
* We write code like JDBC objects and FileStream objects termination etc., in general destroy() always contains the block of statements which are performing **Resource Release logic.**

**Phases of Servlet:** when we are executing servlet based web application in the context of server, a servlet will undergo the following phases.

1. Loading Servlet Class
2. Instantiation of Servlet
3. Initialization of Servlet
4. Servicing/Request Processing
5. Destroying of Servlet
6. Servlet Unavailable
7. **Loading Phase:** Whenever we make a first request to a servlet, the container loads our servlet class(Demo) in the server memory. This phase will be done only once at the time of making 1st request.

**Class.forName(“Demo”);**

1. **Instantiation Phase:** Instantiation phase is nothing but creating an object of our servlet class(**Demo**) which is loaded in the server memory in phase-I by using default constructor **Ex:** Demo ob=new Demo()

**Note:** It is highly recommended to the Servlet programmers not write any parameterized constructors in the servlet program.

**Conclusion:** Instantiation phase will be performed by container only once when we make first request.

**Note:** Once instantiation phase is completed, an object of our servlet(Demo) obtains **Servletness.**

1. **Initialization:** Performing initialization phase is nothing but executing init(-) this always only once by the container when we make first request.
2. **Servicing/Request processing phase:** Performing this phase is nothing but executing service(-,-) this phase will be performed by the container each and every time when client makes a request.
3. **Destruction:** This phase will be performed by the container when the servlet is removed from the context of web server or application server. Which is nothing but execution of destroy().
4. **Unavailability:** The mechanism of making a request to that servlet which was removed from the context of server is known as unavailability phase. When unavailable phase is performing we get an exception called javax.servlet.UnavailableException.

**Q) What are the different approaches of developing Servlets?**

1. Choose an appropriate undefined class and **implements interface** (not recommended) because we don’t want to develop out optimization code.

2. Choose an appropriate user defined class extends GenericServlet recommended for protocol independent servlet but not protocol dependent.

3. Choose an appropriate user defined class and it must extends **HttpServlet** class(Recommonded in the real industry for protocol dependent servlets).

**Guidelines or steps required for developing servlets:** Sun Microsystem has prescribed the following guidelines for development of servlet web applications.

1. Import javax.servlet.\*, javax.servlet.http.\* and other packages if required.
2. Choose an appropriate user defined class and ensure that whose modifier must be public.
3. Whatever the class is selected in step-2 it must extends either GenericServlet and HttpServlet.
4. Override the required life cycle methods of servlet.
5. Whenever the class we have taken in step-2 it must be given as a file name with an extendsion.java.

**P) Write a servlet which will display hello servlet world**

importjavax.servlet.\*;

public class MyServlet extends GenericServlet

{

public void service(ServletRequestreq, ServletResponse res) throws ServletException,IOException

{

PrintWriter pw=res.getWrter();

Pw.println(“<h1> hello Servlet</h1>”);

Pw.println(“<h2> Welcome to servlets </h2> “);

Pw.close();

}

In the above program

1. When the server control comes to service() automatically the objects of ServletRequest and ServletResponse will be created. We know that req contains client data is going to contain the result of servlet or result of request processing logic. In order to write the output of the servlet or Request processing logic Result will require an object of PrintWriter class object. This class object can be created by using the following method present in ServletResponse interface.

javax.servlet.ServletResponse

publicPrintWritergetWriter();

res

pw

PrintWriter pw=res.getWriter();

res

pw

Programatically here pw is used for writing the result of the servlet in the Response object the content of Response Object gives to browser/client when the service() is about to complete or when we call close() upon the PrintWriter class object. To wrote the actual data of the servlet to the Response object we use the following method present in PrintWriter.

Java.io.printWriter

Public void println(xxx)

Here xxx represents any fundamental data type, String and date. After writing the output of the servlet to the Response object, it is highly recommended to close an object of PrintWriter by using the following method.

java.io.close()

Public void close()

**Deployment Descriptor or web application configuration file:**

1. Deployment Descriptor is one of the xml document.
2. The purpose of deployment descriptor is to describe or to express the static web resource programs and dynamic web resource programs which are present in web application.
3. The need of writing deployment descriptor is to hide the technology used in web application development.
4. The name of the Deployment descriptor is always saved on a filename with an extension .xml which is one of the fixed name and whose name can’t be change in the context of servlets and JSP.
5. The aim of deployment descriptor is to provide enough security to the web applications from internet users.

**Structure of DD:**

<webapp>

<servlet>

<servlet-name> logical name of servlet</servlet-name>

<servlet-class>fully qualified name of servlet</servlet-class>

</servlet>

<servlet-name>logical name of servlet</servlet-name>

<url-pattern>user friendly name of servlet</url-pattern>

</web app>

In the above Deployment Descriptor structure

1. <servlet>------</servlet> this tag always contains logical name of the servlet(Deployerurl) and fully qualified name of servlet(Private url)
2. <servlet-mapping> ----------------</servlet-mapping> this tag contains logical name of the servlet and user friendly name of the servlet(Public url) this tag which is stand by server for containing user friendly name of the servlet.
3. <url-pattern>-------------</url-pattern> this tag always contains user friendly name of the servlet for easy accessing purpose for the internet users.
4. <servlet-name>This tag contains logical name of the servlet/alias name or alternative name to the actual servlet. This is used for mapping user friendly name of the servlet to fully qualified name of the servlet.
5. <servlet-class> this tag contains fully qualified name of the servlet.

<web-app>

<servlet>

<servlet-name>Siva</servlet-name>

<servlet-class>MyServlet</servlet-class>

<servlet-mapping>

<servlet-name>Siva</servlet-name>

<url-pattern>/india.com</url-pattern>

</servlet-mapping>

</web-app>

**Staging directory based or directory structure:** In order to execute any web application in any of the web server or application server, SUN Microsystems has prescribed mandatory structure known as staging directory or directory structure or context root.

1. Create a folder/directory in the current working machine this directory is known as staging directory or directory structure.
2. Create a folder called **SRC** in staging directory. Creating SRC is optional.
3. Write the sevlet(\*.java), write the JSP(.JSP) as dynamic web resource programs and write a html program(\* .htm), write a Java script(\*.JS) as static web resource programs and save them in SRC folder or directly staging directory.
4. Create a folder WEB-INF in staging directory. Create a folder classes under WEB-INF folder.

iv.1 Write web.xml and place in WEB-INF .

1. Compile a servlet program and ensure whose file must be generated and it should be into class folder.
2. Create folder called LIB with the table WEB-INF folder, LIB folder always contains \*.jar files in general diagrammatic format of directory structure is given below.

SRC

\*.java

\*.JSP

\*.htm

\*.JS

WEB-INF

WEB-INF

Classes

**\*.class**

lib

\*.jar

**Syntax for running a servlet:**

http://DNS(or)IPaddress:portno\directory structure name\ urlpatten of servlet

**DNS:** It is one of the physical name of the computer where server software is installed. Default DNS of every PC is localhost.

**IPaddress:** It is one of the four parts numerical address of a physical machine where server software is installed. The default IP address of every PC is 127.0.0.1 known as loop back address.

**Port number:** A port number is one of the logical numerical ID where server software is running. For example a Tomcat server is by default running at 8080 and we can change its protnumber(For example we can change to 2012 but it should lies with in 1024 to 65536 because to 1023 universal protocols are running). Directory structure represents name of the staging directory where WEB-INF folder presents. url pattern of a servlet represents user friendly name of the servlet which is specified in <url-pattern>------</url-pattern> of <servlet-mapping>

------</servlet-mapping>

**EX:**

http://localhost:2012/firstservlet/india.com

**Common steps for developing, deploying and running the web application:**

1. Develop a servlet/JSP/HTML/JS
2. Develop deployment descriptor web application configuration file<web.xml>
3. Prepare staging directory
4. Deploy the web application wither in webserver/application server(hard deployment, console based deployment, tool based deployment)
5. Run the web application by opening the browser and by passing url of web application.

The above 1,2,3,4 steps will be perfomed by server side programmer and step-5 performed by internet user.

**Tomcat Manager Opening Problem:** By default Tomcat doesn’t contain Tomcat Manager opening option. To activate it just change this XML file in following location.

C:\Program Files\Apache Software Foundation\Tomcat 6.0\conf

<?xml version=’1.0′ encoding=’utf-8′?>  
<tomcat-users>  
<role rolename=”manager”/>  
<role rolename=”admin”/>  
<user username=”admin” password=”admin” roles=”manager,admin”/>  
</tomcat-users>

**Note: After Changing don’t forget restart tomcat server.**

**P) Write a Servlet program which will display current date and time of the system using**

**For Example please refer ServletFirst document page numbers 1-2, DispalyTimeAndDate(GenericServlet) project in NetBeans projects.**

**P) Develop a web application which will display current date and time of the system by using HttpServlet.**

**For Example please refer DispalyTimeAndDate(HttpServlet) project in NetBeans projects.**

Compile the above program java –d.DateServ.java ensure that package ds must be created, DateServ java program must be compile and its DateServ class and automatically copied into ds package. Copy or cut ds folder and place it into classes folder of WEB-INF folder. Copy DateServlet web application and deploy in the web application folder of TomCatserver(hard deployment).

**Annotations:** Annotations are the Java statements which can be used to perform meta data operations and resource configurations.

1. Annotations are alternate for XML files towards performing resource configuration. Data about data is called metadata.
2. Configuring resource in XML file and passing more details about that resource to the underlying executing environment is called as Metadata operations or resource configuration.
3. Configuring servlet program in web.xml file is called as resource configuration or metadata operation. From servlet 2.5 api onwards this work can be done in .java files using annotations.

**Syn:@**<connection\_name>(param 1=val1, param2=val2….)

There are two types of annotations in Java

1. Documentation annotations
2. Programming annotations
3. **Documentation annotations:** These will be used in documentation comments/\*…….\*/. These are available in jdk1.1 to 1.4
4. **Programming annotations:** These annotations will be used for programming in Java. These will be used as alternative for XML files for resource configuration.

These are introduced from jdk 1.5

**Examples for documentation annotations:**

@author, @since, @param, @throws and e.t.c.,-----------

(refer .java files of <java\_home\src.zip file)

**Examples of Programming annotations:**

@Override, @FailSafe and e.t.c.,

**Q) What are the differences between configuring resources using XML files and using annotations?**

|  |  |
| --- | --- |
| **XML Files** | **Annotations** |
| 1. XML parsers read data from xml files. 2. XML parser is heavy weight software application so working with XML files for resources configuration 3. Gives bad performance but gives good flexibility of modification with out modifying the source code | 1. Annotations are placed in .java files directly 2. Annotations are underlying runtime environment or container. So these are recognized directly by JVM. 3. Gives good performance, flexibility is not there because each and every time we have to compile .java file. |

**The technologies which are giving support to annotations:**

Servlet 2.5, 3.X, Hibernate 3.5+, 3.X, EJB 3.X, Struts 2.X e.t.c.,

Every annotation is special @interface containing methods representing annotation parameters.

Annotations of Standalone applications will be recognized and processed by jre/JVM , Servlet alllication will be recognize by servlet container(Catalina), For EJB EJB container, For Hibernate applications 🡪 Hibernate software.

We can apply annotations at three levels of Java source code.

1. Resource level/on the top of the class/on the top of the interface
2. Method level on the top of the Java methods.
3. Field level/ on the top of the Data member.

These Annotations makes resources, methods, fields to be recognized by the container or jre as special entity.

**Important annotations of servlet 3.X programming:**

@WebFileter (to configure servlet Filter program)

@WebServlet (to configure servlet program)

@WebListener (to configure servlet Listener)

@WebInitParam(to specify init parameter value).

**Steps for developing application:**

1. Keep Tomcat7 server in running mode.
2. Add <Tomcat-home>/lib/servlet-api jar file to class pat.
3. Develop the Java application as shown below.

**P) Write above program replacing with annotations.**

**For program refer AnnotationSupport project in NetBeans projects.**

**To run above application manually it requires JDK1.6 + version, Tomcat7 version, no need to write web.xml. Otherwise Run it in NetBeans 7.1, Otherwise run in NetBeans IDE 7.1 or higher versions.**

**Class Path:** C:\Program Files\Apache Software Foundation\Tomcat 7.0\lib\annotations-api.jar;C:\Program Files\Apache Software Foundation\Tomcat 7.0\lib\servlet-api.jar

1. HTTP is one of the application level protocols comes under TCP, HTTP is one of the state less protocols http is one of the acknowledgement oriented protocol.
2. Each and every web application makes use of http protocol for exchanging the data between client and server side applications.
3. All the real world serves which are available are implementing the specification of http so that all these servers are treated as **Stateless servers.**

**EX:** Tomcat, Web logic, Web sphere, JBoss, JRun etc.,

A stateless server is one which maintains **every request** of the same internet user as a individual/ fresh request that is even though a single user maks multiple requests from the dame client machine treated by the server software as individual fresh requests. We know that a stateless protocol is one which maintains an identity of a limited span of time. We know that a protocol is the set of rules used for exchanging the data between client and server side applications while it is exchanging the data between client and server side applications, hhtp makes its use of methods known as http communication methods.

Http methods are classified into seven types. They are GET, POST, HEAD, PUT, DELTE, TRACE, OPTIONS.

1. **GET:** This is one of the default method taken by http protocol for sending the data form client side to server side applications to get the response from server side applications.

If we use GET method as a part of http then the data of the client sending to the server side applications as a part of url.

<http://localhost:2012/empServlet>

empno:100?ename=siva?esal=3.5

using GET method is not that much recommended in the web application development because of following drawbacks.

1. No security for the client data.
2. Limited secrecy of client data can be transferred to the server side application.
3. With GET method we are unable to upload files.
4. Unable to send audio and video files.
5. **POST:** This method is also used for sending the data of the client to the server side application for getting the appropriate response. If we use POST method as a http method then the data of the client can be transferred to the Serverside application as a part of **request body.** When we use POST method instead of GET method we get the following advantages.
6. Security is monitories for client data.
7. Unlimited amount of Data can be transferred.
8. One can upload files, images, audio, video files etc., the neglisible drawbacks of POST method is slow in communication between client and server side applications.
9. **HEAD:** This method is used for **additional information** regarding client request as browser name used for sending client request, whose vendor name type of protocol used, browser versions and type of data sending.
10. **PUT :**This method is used for placing or adding additional middleware services to underlying server.
11. **DELETE :**This method is used for deleting or removing the middleware services in the underlying server.

**Note:**The methods PUT and DELETE are disable in most of the servers because without the prior permissions of server vendors middleware services can’t be added and can’t be removed.

1. **TRACE:** This method is used the internal flow of servlet with underlying server.
2. **OPTIONS:** This method is used for finding which http methods are supporting and which http methods are not supporting by the underlying server. Most of the real world servers, OPTIONS method gives the supporting of GET, POST, TRACE, OPTIONS bit not PUT and DELETE.

**Conclusions:** All the above http methods specified in the Client side program.

**Http Servlet:** HttpServlet is one of the sub class of GenericServlet so that all the methods of GenericServlet are inherited into HttpServlet.

1. HttpServlet developed by server with the specification of servlet api and http protocol.
2. All the methods of http protocol are implemented in HttpServlet class and whose general notation is protected void doXxx(HttpServletRequest, HttpServletResponse) throws ServletException,IOException
3. Here Xxx represents any http method. If our class extends GenericServlet then such servlet is known as GenericServlet and they are protocol independent and in practical world are not recommended. If our class extends HttpServlet then such servlet is known as HttpServlets and they are protocol dependent and recommended to develop in particular world.

**Methods in HttpServlet:**

1. Public void service(ServletRequest, ServletResponse) throws ServletException, IOException
2. Protected void service(HttpServletRequest, HttpServletResponse) throws ServletException, IOException
3. Protected void doXxx(HttpServletRequest, HttpServletResponse) throws ServletException, IOException
4. **Public void service(ServletRequest, ServletResponse) throws ServletException, IOException:** This method is one of the predefined method in HttpServlet class by inheriting GenericServlet. This method is meant for writing request processing logic by making use of protocol independent requests and protocol independent responses.
5. **Protected void service(HttpServletRequest, HttpServletResponse) throws ServletException, IOException:** This method is used for writing request processing logic by handling protocol dependent requests and protocol dependent responses.

Method 1 is overridden in such a way that method 2 to be called automatically and **casting** ServletRequest(Protocol independent) into ServletRequest(Protocol dependent) and also casted ServletResponse into HttpServletResponse. ProgramaticallyHttpServletRequest and HttpServletResponse are the sub interfaces of ServletRequest and ServletResponse interfaces respectively. Method 2 is further overridden to call appropriate doXxx(-,-) depends on type of http method used by the client.

1. **Protected void doXxx(HttpServletRequest, HttpServletResponse) throws ServletException, IOException:** This method represents in seven forms by taking htto methods but as a servlet developer always writing request processing logic.

**NOTE:** doGet(-,-), doPost(-,-) are recommended methods.

1. Client makes a first request to the web application.
2. User request goes to the physical machine depends on type of DNSIP address.
3. Physical machine locates type of server in which web application is deployed depends on its port number and handover user request.
4. The server identifies the web application name(Staging directory)
5. The container looks for **web.xml** and scanned container works for <url-pattern> ----</url-pattern> of web.xml
6. If user requested url pattern is not matching with <url-pattern> values of web.xml then container gives 404 errror as a response to the client(Response not exists). If it is making then further steps will be continued by the container.
7. <servlet-name> of <servlet-mapping> is matching is converting <url-pattern> into actual servlet.
8. Container takes fully qualified name of the servlet and go to classes folder and loads the .class file. <servlet> in the server memory only once. After loading process instantation and initialization phases will be performed.
9. The container will call public service(-,-) it interns calls protected service(-,-) and it interns calls either doGET(-,-) doPOST(-,-) and executes request processing logic.
10. Container gives result of Request processing logic to the client(browser) all the above steps will be performed when the user maks a first request if the same user makes second and further subsequent requests container will perform step no-11.

**Note:** In the above steps 7,8 and 9,10 are same

**P) Develop a servlet which illustrate the concept of Life cycle methods of servlet**

**For program please Refer LifeCycleServletMethds project in NetBeans projects.**

**Types of Deployment:** A server side programmer can deploy the web application in three ways they are

1. Hard Deployment
2. Console based deployment
3. Tool based deployement.
4. **Hard Deployment:** In Hard deployment process we copy the web application name or staging directory from our current working machine and paste in the proper location of server(for example the staging directory name) copied from our current working machine and pasted it webapps folder of Tomcat server. Industry is not recommended to deploy the Webapplication based on hard deployment.
5. **Console based Deployment:** In Console based deployment we select **warfile**from our current working machine and we use the services of the underlying server for its automatic deployment. In console based deployment, deploying the web application is taken care by server.

**Syn: Creation of warfile:**

**EX:**

C:\servletex\LifeServlet>jar cfvsat.war WEB-INF

added manifest

adding: WEB-INF/(in = 0) (out= 0)(stored 0%)

adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)

adding: WEB-INF/classes/lf/(in = 0) (out= 0)(stored 0%)

adding: WEB-INF/classes/lf/LifeServlet.class(in = 1401) (out= 737)(deflated 47%)

adding: WEB-INF/web.xml(in = 243) (out= 105)(deflated 56%)

**Jar:** jar is a tool or application program or exe file used to create war files/jar files.

**War files to be created for web applications:**

It is used for to find background details about what files are placed in war file, what is the size of original file(in) size of the file after reducing (out) , percentage of reduction (deployed 47%).

Filename.war represents war file name must be created for servlets and JSP that is web application deployment.

WEB-INF represents a folder in the staging directory and what are all the files are available in the WEB-INF folder including its sub folders will be compressed and placed in war file.

**Steps for console based deployment in Tomcat:**

1. Start the Tomcat server.
2. Open the browser and write the following url

Ensure that homepage of the Tomcat server must be shown.

1. Choose **Tomcat Manager** enter the authentication details of the Tomcat server

Ensure Tomcat Webapplication manager must be opened.

Browse

1. Select the war file to deploy choose select the war file from our current

open

working directory and choose deploy

1. Ensure that after the complete deployment, Tomcat web application manager gives the status message as ok. Otherwise Tomcat web application manager gives fail.

Open the browser and makes request to access the servlet/Webapplication

Sat is war file name.

In order to undeploy the web application(war files) go to Tomcat web application manager, select **undeploy**of the corresponding war file.

**Web logic server:**

Type: Application server

Vendor: BEA systems(Oracle)

Type of Software: Commercial software

Port number : 7001

Download: [www.commerce.bea.com](http://www.commerce.bea.com)

Help related documents: [www.edos.bea.com](http://www.edos.bea.com)

Version: 8.1🡪jdk(out dated)

9.1->jdk 1.5,1.6

10.0 ->jdk 1.7

Jar file: weblogic.jar

**Note:** weblogic9.1 supports jdk1.5,1.6 only not for 1.7

**Weblogic server Configuration and deployment refer servlet extra page numbers 1-17.**

**Q) Difference between web servers and application servers.**

|  |  |
| --- | --- |
| **Web server** | **Application server** |
| 1. Web servers understands only http protocol but not other protocols. 2. Web servers execute or run web applications but not business applications. 3. Web server executes servlets and JSP related applications but not EJB,JTA(Java transaction API). 4. Web servers implements the specification of web components like Servlets, JSP but not the specification of business components like EJB, JTA e.t.c., 5. Web servers are not able to provide middleware services. | 1. Application servers understand http protocols and other universal protocols of both TCP and UDP. 2. Application server executes both web and business applications. 3. Applciation server executes servlets, JSP, EJB, JTA, e.t.c., 4. Application server implements the specification of both web, business components. 5. Application servers are containing effective middleware services like connection pooling, security services, transaction services. |

**MIME Types:** Whenever client makes a request to the servlet, the servlet gives its response back to the client most of the internet users are getting response from web applications in the form of Word format, pdf format, XML format, Excel format etc., These varieties of formats given by the servlet as a response to the client is known as MIME type in otherwords HTML type always represents the type of response given by the servlet to the client. The default MIME type in every web application is text/plain(no HTML tag or text/HTML(if HTML tags present)) programmatically one can also set the MIME types.

**Application/Msword:** This represents the response of the servlet to be given in the word format.

**Application/pdf:** This represents the response fo the servlet is in pdf format.

**Text/XML:** This gives XML response.

In order to set MIME type to the response object we use the following method present in http servlet response interface.

Javax.servlet.http.HttpServletResponse

Public void setContentType(String)

Here String parameter represents MIME type res.setContentType(“application/msword”);

Per servlet we must set the MIME type only once but not more than once.

**Common MIME Types:**

|  |  |
| --- | --- |
| **Type** | **Meaning** |
| 1. application/word 2. application/octect-stream 3. application/pdf 4. application/postscript 5. application/vnd.lotus-notes 6. application/vnd.ms-excel 7. application/vnd.ms-powerpoint 8. application/x-gzip 9. application/x-java-archieve 10. application/x-java-searialized-object 11. application/x-java-vm 12. application/zip 13. audio/basic 14. audio/x-aiff 15. audio/x-wav 16. image/gif 17. image/jpeg 18. image/png 19. image/tiff 20. image/x-xbitmap 21. text/css 22. text/html 23. text/plain 24. text/xml 25. video/mpeg 26. video/quicktime | Microsoft word document  Unrecognized or binary data  Acrobat(.pdf) file  PostScript file  Lotus notes file  Excel Spreadsheet  Power Point Presentation  Gzip archive  JAR file  Serialized java object  Java bytecode(.class) file  Zip archieve  Sound file in .au or .snd format  AIFF sound file  Microsoft windows sound file  GIF image  JPEG image  PNG image  TIFF image  X windows bitmap image  HTML cascading style sheet  HTML document  Plain text  XML  MPEG video clip  QuickTime video clip |

**P) Develop a web application which illustrate the concept of MIME type?**

**For program please referMIMEServlet project in NetBeans project.**

**Note:** Here XML file will read directly by Browser.

**Note:** If we want to change the url pattern of or servlet after deploying the web application in the Tomcat server then open web.xml file in the context of server and save the web.xml file(No need to reload in the Tomcat web application manager).

**Note:** If we want to change Request processing logic of the servlet after deploying in the context of server then open the servlet programming in the context of server, do the changes in Request processing logic compile those servlets in the context of server and choose Reload in the context Tomcat web application manager.

**Note:** If we deploy any kind of war file or staging directory in the webapps folder of Tomcat then that war file are staging directory automatically placed in work\catalina\loaclhost of Tomcat server. Because Catalina is one of the container for executing servlets.

**Communication with HTML:** We know that a web application is a collection of static web application of static web resource programs and dynamic web resource programs. In web application development the role of static web resource programs are to accept client data(browser) and hand over to dynamic web resource programs for developing static web resource programs we make use of the following technologies. HTML, DHTML, JS, VB Script etc., The role of dynamic web resource programs in the web application development is to validate the client data which is coming from HTML forms. In order to accept client data we require user interfaces/ GUI applications we make use of either AWT/SWINGS/Applets or HTML. Developing GUI applications with applets, SWINGS are difficult for the web application programmer so in the most of the web application development for development of GUI applications we may not use AWT, SWING, Applet but industry always recommended as a part of project based applications for development of user interface we always use HTML. IN HTML to develop user interfaces we have various GUI controls such as Text Area, Checkbox, Radio buttons etc.,

After deploying form/user interface by using HTML, that user interface data to be submitted to the dynamic web resource programs like servlet or JSP. Syntax for calling servlet from the context of HTML

**Syn:** <form action =”url pattern of dynamic web resource program” method =”GET/POST”>

<Input --------------/>

<Input -------------/>

</form>

1. <form>------------</form> is one of the HTML tag used for calling a dynamic web resource programs from the context of HTML program.
2. Action, method represents attributes of form tag.
3. The value of action attribute is url pattern of dynamic web resource program which is specified in <url pattern>-----------</url pattern> of web.xml file.
4. The value of method attribute is either get or post. If we use GET method then the client data will be submit into the servlet as a part of url.
5. If we use **POST** method then client data submitted as a part of request body.
6. The attribute method is optional to use if we are not using method attribute then by default GET method will be considered. Depends on the type of method we use GET or POST, in the servlet program we need to override the corresponding method that is either doGET or doPOST

**P) Develop a web applicaton which will contain one HTML program and one servlet program. The HTML program must accept name of the student and course name. The servlet program will accept student name and course and display the same to the client as a response.**

**For complete project please refer NetBeans(servlets)\ServletStudentDetails folder**

HTML Form

Student.html

Student.java

WEB - INF

Web.xml

sp

Classes

Student.class

Student.html

Student.java

<http://localhost:7001/st/student.html>

Enter UR name :

Enter UR Course:

UR Name :Siva

Selected Course: Servlets

Servlets

ssss

Web servlet

**NOTE:**To delete war file delete file.war

In order to make the servlet as flexible we write the request processing logic either in doGET method or doPOST method and call another method from the context of some other method. For example request processing logic is written in doPOST method and client request is coming to doGET method and doPOST method and doGET method should call doPOST method.

Let us consider the following code segment.

protecteddoPOST(-,-)

{

//Request processing logic

}

protected void doPOST(-,-)

{

doPOST(req,res);

}

}

The above approach makes us to understand irrespective of the type of HTTP method used in the HTML program, it can be processed either in doGET or doPOST method without changing the source code of the servlet.

In real world application development( in most of the popular IDEs), doGET method and doPOST methods are used for **Request handling**purpose and actual request process logic will be written in user defined methods and that user defined methods will be called as a part of either doGET method or doPOST method. Let us consider the following code segment.

protected void goGET(-,-)

{

processRequest(req,res)

}

protected void ProcessRequest(HTTPServletRequestreq, HTTPServletResponse res)

{

//request processing logic

}

}

In the above code segment container never calls directly the user defined method called process request.

**P) Develop a web application which contains HTML program and a servlet, the HTML program will accept two numbers from the client and then send to the servlet. Servlet program must receive two numbers and find biggest of those numbers.**

**For example program please referBignumber project in NetBeans Projects.**

BigServlet

Big.html

BigServ.java

WEB-INF

Web.xml

classes

DS

BigServ.class

**Note:** For above program create jar file like this

jarcfvfss.war WEB-INF \*.html

Retrieving the client data in the context of Servlet we know that whenever we send the client data through the HTML program to the servlet, the client data is available in the Request object of HTTPServlet Request in the form of (key, value).

1. Keys represents HTML form parameter names and values represents HTML form parameter values/request parameter values.
2. For the purpose of Request processing logic either as a part of doGet or doPost we need to retrieve form parameter these values we have the following approaches.
3. public String getParameter(String)
4. public String[] getParameterValues(String)
5. public Enumeration getParameterNames()
6. public Map getParameterMap()

All the above methods are present in HttpServletRequest.

1. **public String getParameter(String):** This method used for retrieving form parameter values when we have distinct form parameter names. The taking parameter String represents form parameter name and return type String represents form parameter value.

**EX:** req

Stno: 100

Sname:siva

Marks:99.99

String sno=req.getParameter(“sno”);

String sname=req.getParameter(“sname”); Request parameter gathering logic.

Pw.println(“StudentNo=”+sno);

Pw.println(“studentName=”+sname);

Pw.println(“student Marks=”+smarks);

**public String[] getParameterValues(Stringpname)**

In the some of the HTML forms, one form field name may contain multiple values and when we submit such form data to the servlet, in the context of servlet we use the above method for retrieving multiple values by passing single form field name.

**EX:** String pv[]=request.getParameterValues(“course”);

for(int i=0;i<pv.length;i++)

out.println(pv[i]+”<br>”);

**public Enumeration getParameterNames()**

When client submits the form data to the servlet, the server side programmer may not be aware of form parameter names and their values. In order to find unknown form parameter names and their values we use the above method.

**EX:Enumeration enum =** request.getParameterNames();

while(enum.hasMoreElements())

**request object**

a=10,20,30,40

b=20

c=60,90,70

{

String pn=enum.nextElement();

String pv[]=request.getParameterValues(pn);

for(int i=0;i<pv.length;i++)

{

out.println(pv[i]+”<br>”);

}

}

**public Map getParameterMap()**

The functionality of the method is similar to the above method but this method returns an Object of Map interface. Map is one of interface present in java.util package, and it allows us to retrieve concurrency whereas Enumeration interface object allows us to retrieve the data sequentially.

In Other words Enumeration interface object belongs to synchronized and Map interface object belongs to non-synchronized.

**EX:** Map m = request.getParameterMap();

Set s=m.getKeySet();

Iterator itr=s.iterator();

while(itr.hasNext())

{

String pn=(String)itr.next();

String pv[] =request.getParameterValues(pn);

out.println(“<br>” + pn);

for(int i=0;i<pv.length;i++){

out.println(pv[i]+ “, “);

}

}

**Same Program working with Net Beans 7.1 IDE:**

**Note:** Refer working with NetBeansIDE MS Word document or Pdf file.

**Servlets and Database Connectivity:** In most of the circumstances what ever the data we retrieved from the client side(HTML form) in the servlet, the servlet may insert, may update, or may delete the data from the Database. If we want to perform any Database operations to the servlet, we need to establish communication between servlet and data taken is achieve the data persistency. To the existing web application we add any another layer called Database then the web application is known as distributed application and satisfies three-tier architecture. In order to write JDBC related statements in the servlet, we write following steps.

1. Load the drivers and obtain the Connection.
2. Use the JDBC Connection and obtain either statement obj or PreparedStatementobj or Calleable Statement obj.
3. Release the JDBC stream objects.

SUN has recommended these approaches for writing the Database operations.

**Approach1:**

**Step1:**

1. Override public static init(ServletConfig) in such as way that we write a code for loading the drivers and Connection.
2. Override public void service(-,-) in such a way that it uses JDBC connection obj or obtains either select or PreparedStatement or CallableStatement
3. Override public void destroy() in such a way that we release/terminate the JDBC statement obj.

Approach 1

Personal.html

PersServ.java

WEB-INF

Web.xml

classes

ps

PreparedStatement.class

Lib

classes111.jar

ojdbc.jar

classes12.jar

**browser server**

<http://localhost:8080/>

persdetails

pno:

name:

Personal.html

PersServ.java

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

Result from Database

clear

save

|  |  |  |
| --- | --- | --- |
| Pname | Addr | Age |
| Siva | Ampt | 25 |

Web/application server

PersServ

**Advantages of Approach1:** Irrespective of the number of clients who are accessing the servlet, there exists a single Connection object which is used by all the clients it improves the performance of web application.

**P) Develop a web application by using approach1**

Please refer page absolute project in NetBeans projects or refer Page numbers 28-45 from ServletFirst

**Disadvantages:**

1. Whatever the variables/ objects created as a part of servlet as instance Datamembers, they are all accessed by multiple clients.
2. When multiple clients are accessing single instance data members then we get inconsistent results hence instance datamembers of our servlet class leads to Thread safety problems.
3. As a servlet programmer we need to write synchronization mechanisms(Synchronized methods and blocks). Hence Approach1 gives Thread safety problems.

**Approach2:**

1. Load the JDBC drivers and get the Database Connection as a part of service(-,-)
2. Use JDBC Connection object and obtain Statement/ PreparedStatement/ CallableStatement as a part of service(-,-)
3. Release or terminate the JDBC Stream objects as a part of service(-,-)

Let us consider the following diagram which gives the occurrences about approach2

**Web/ Application Server**

PersonalServ

Service(-,-)

**Advantage of approach2:**

We create all the JDBC related Stream objects as a part of service(-,-) so that consistency of the web application result is taken care by the server by creating multiple threads that is all the JDBC Stream objects are Thread safety and synchronization is taken care by server.

**Limitations:**

1. As and when a client makes a request to the servlet a separate object is created in multiple clients are making a request same servlet multiple times Connection objects are created.
2. This leads to less performance to the web application because getting the connection obj in every time is a time consuming process.

Approach2

Personal.html

DBSelect.java

WEB-INF

Web.xml

classes

db

DbSelect.class

Lib

classes111.jar

ojdbc.jar

classes12.jar

**P) Develop a web application which deals product information(Case Study).**

**Refer CaseStudy project in NetBeans Projects.**

**P) Develop a project for Login from which check login details in Database.**

**Refer ProjectLoginDatabase in NetBeans Projects.**

**browser server**

<http://localhost:8080/>

Product mgmt system

Insert entprdno:

Deleteentprnam:

Modifyprice:

ProductInsert.java

ProductUpdate.java

ProductDelete.java

res

subb

Product

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**Approach3:**

**Connection Pooling in servlets:** Connection Pooling is one of the middleware service provided by most of the application server vendors we know that middleware services are those which are providing effective execution environment for web application some of the middleware services provided by Application servers are Connection Pooling, security services, Transaction services etc.,

**Def of Connection Pooling:** A Connection Pooling is a factory which contains Connection objects in web application development we have two types of Connection Pools they are

1. Application Managed Connection Pool
2. Server Managed Connection Pool
3. **Application Managed Connection Pool:**
4. An application managed Connection Pool is one which created by a Java programmer with the Collection of JDBC Connection objects.
5. In Application Managed Connection Pool, Creation of JDBC Connections, Managing the JDBC Connections and destroying the JDBC Connections must be taken care by Java programmer. These types of Connection Pools are not that much recommended may be used I Context of Standalone applications.
6. **Server Managed Connection Pool:**
7. Server Managed Connection Pools are those which are created by server, managed by server and deployed by server.
8. Industry is highly recommended to use server managed Connection Pool as a part of distributed application development because they provide more performance and thread safety.
9. As a servlet developer we make use of JDBC Connection object from pool by browsing the Connection Object from the pool and we release the JDBC Connection object to the pool.
10. For example as a servlet developer if we create a Connection Pool for the Oracle Database by making use of Oracle driver. Then it is known as Oraclce Connection Pool similarly the same thing is applied for other Databases.

When we are implementing Connection Pooling as part of servlet, Connection Pooling Concept makes use of Type-3 driver which are developed by server vendors in Java language.

**Type3 Driver:** This is Known as net protocol driver which is not purely treated as JDBC driver for establishing the communication between JDBC application and Database but this driver always used for establishing for communication between servlet and JSP and Connection Pool. In otherwords type-3 driver borrows a readily available Connection object from Connection Pool.

**browser server**

<http://localhost:8080/>

Enter empno:

ename: Siva sal: 2500

Personal.html

DBServ.java

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

Database request

|  |  |  |
| --- | --- | --- |
| empno | ename | sal |
| 7788 | scott | 2500 |

**Steps for implementation of Connection Pool:**

1. Create a Connection Pool by choosing the application Database.
2. Each and every Connection Pool must be associated with Database.
3. Each and every Datasource of Connection Pool must binds registry software known as JNDI( Java naming directory interface)
4. As a servlet programmer we look for JNDI gives access to Datasource and access to Pool of JDBC Connections and finally available JDBC Connection object from Connection Pool.
5. As a servlet Programmer we make use of JDBC Connection object and perform multiple operations.
6. As a servlet programmer we release the JDBC Connection object and it is collected by server adds to the Connection Pool.

**Registry Software:** Registry software is one of the service available in the servers for making middleware services globally visible for any external program that is servlet, JSP, EJB, Hibernate etc.,. In weblogic server the registry software name is JNDI.

**Code for getting Connection Pool object from Connection Pool:**

1. Create an object of InitialContext. It is one of the predefined class present in javax.naming.\* package
2. An object of IntialContext is always ponting the registry software name called JNDI.

InitialContextic=new InitialContext();

1. Look up for JNDI name if is successfully located Type casting into javax.sql.DataSource. If JNDI name is unable to find then we get a predefined exception called javax.naming.NamingException. The prototype lookup() is shown below.

javax.naming.InitialContext

Public Object lookup(String) throws javax.naming.NamingException

Here String parameter represents JNDI name

Object obj=ic.lookup(“siva”);

DataSource ds=(DataSource) obj;

DataSource ds=(DataSource)io.lookup(“siva”);

Here an object of DataSource is ponting to DataSource which we created in the web logic(ds)

1. Obtain or borrow a JDBC Connection object from the pool. To do this apply the following method which is present in javax.sql.DataSource.

javax.sql.DataSource

public Connection getConnection

**EX:** Connection con=ds.getConnection();

return the Connection object which is borrowed from the Connection Pool to the servlet/JSP

return con;

**EX:** Write a user defined method for getting a Connection Pool object from Connection Pool.

Connection Pool con=ds.getConnection();

{

Connection con=null;

try

{

InitialContextic=new InitialContext();

DataSource ds=(DataSource)ic.lookup(“siva”);

Connection con=ds.getConnection();

}

catch(Exception e)

{

e.printStackTrace();

}

return con;

}

**P) Write a program on Server Managed Connection Pool .**

**For Program please refer NetBeans(servlets)\ConnectionPooling folder.**

**Note:** Please do this program through weblogic. Don’t go for Glass Fish because some internal problem is in GlassFish Connection Pooling. This application never run run in Tomcat because Tomcat is not Application server.

The following diagram gives the overall view about Connection pooling implementation steps and usage

Application server

Registry Software(JNDI) – siva

External program

Data source

JDBC Application

Servlet Application

EJB Application

Program Framework App

Etc.,

Con1

Con2

Con n

Database Software

1. Create the Connection Pool in the application server like web logic
2. Each and every Connection Pool must be associated with DataSource
3. Each and every DataSource must binds with Registry software that is JNDI name.
4. As a Java programmer we write an external program like a JDBC application or servlet/JSP application or EJB application or any Framework related applications.
5. Each and every external program look up for registry software that is JNDI name.
6. JNDI gives access to datasource.
7. DataSoruce obtains a Conenction object and gives Connection object to external program.
8. External Program receives/ borrow a Connection object from Conncetion Pool through DataSource name.
9. External program(Servlet) establish Connection with Database Connection with Database software.
10. As external program developer(servlet), over the Connection we send the Database related queries and they executed in the Database server(11) and gives result back to external program.
11. External program receives the result which is coming from the Database software.
12. Since the external programmer is **no longer interested** in maintaining Connection with the external programmer receives the Database connection. Once it is released, application server added to the Connection Pool.

**Approach3:** Approach3 makes us to understand the usage of Connection Pooling concept as a part of Servlet to communicate with the Database.

**Advantages:**

1. Thread safety is provided.
2. Higher performance is achieved. Recommended to use Approach3 to establish the Communication between servlet/jsp and Database software.

**Q) In how many ways one can obtain an object of JDBC Connection Pool for Communication with the Database?**

1. Obtaining the Connection from application managed Connection Pool

**EX:** Connection con=DriverManager.getConnection(“jdbc:oracle:thin:@localhost:1521:orcl”,”scott”,”tiger”);

2.Obtaining a Connection object from DriverManaged Connection Pool

PooledConnectionpcon=ocpds.getPooledConnection();

Connection con=pcon.getConnection();

1. Obtaining a Connection object from server managed Connection Pool.

InitialContextic=new InitialContext();

DataSource ds=(DataSource) ic.lookup(“siva”);

Connection con=ds.getConnection();

**<load-on-startup>-------</load-on-startup>:** Whenever we make a first request to the servlet, the servlet container will perform following phases.

1. Initialization Phase by loading the servlets in the main memory of the server.
2. Initialization phase.
3. Request processing phase. When we make second request to further subsequent requests the servlet container will perform only request processing phase. The points makes us to understand the response time of first request is **more** and the response time of second and further subsequent requests takes less time. Which is not recommended in the web application development.
4. In order to equalize the response time of all the requests, we must use the concept of <load-on-startup>--------</load-on-startup>
5. <load-on-startup> is one of the **declarative approach** (which is nothing but add the XML entries in the web.xml file).

**Note:** Declarative approach means add XML entries in web.xml file, programmable approach means perform operations in program, Configurable approach means perform from server).

**Syn:** <load-on-startup> priority<load-on-startup> Here priority represents a **numerical integer value** the priority value represents positive, zero or negative.

**+ve value:** If we specify highest positive priority value then container will take it as low priority to the servlet. If we specify lowest positive value then container will give highest priority to the servlet.

0 value: If priority value is zero then the container gives least priority to the servlet. In all the servers excepts Tomcat 6

**-ve value:** Using –ve value for the priority is nothing but giving no priority to the servlet which is nothing but neglecting such negative values as <load-on-startup> in another words writing –ve value is nothing but not writing and <load-on-startup>

\*If we write equal +ve value for all the servlets in a web application then priority will be given in the order which they written in the web.xml file that is in which ever order we write the servlets in the web.xml file in the same order their initialization and initialization will be perform before we make first request.

**EX:**

<web-app>

<servlet>

<servlet-name>siva</servlet-name>

<servlet-class>bg.Big</servlet-class>

<load-on-startup>10</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>siva</servlet-name>

<url-pattern>/big.com</url-pattern>

</servlet-mapping>

</web-app>

**ServletConfig and ServletContext:**

**ServletConfig:** If we write any servlet in the fixed and technical information and if the servlet is decided to change the fixed details then the servlet developer has to perform the following applications.

1. Go to the servlet, change the required fixed values and save.
2. Compile the above servlet and ensure .class file must be generated.
3. Copy Corresponding .class file into the classes folder and generate war file.
4. Deploy the war file in the server.

The above cycle of operations are repeated whenever we are changing the source code of the servlet. This type of approach is known as **Hard coded approach.** Industry is not recommended this type of approach but it is always recommended flexible approach.

**Note:**

1. In hard coded approach the values of the application participates in compilation time. Where as flexible approach is one the values participates in execution time. In order to develop flexible servlets, we use a predefined interface called ServletConfig which is present in javax.servlet.\* package.
2. Developing flexible servlet is nothing but getting the technical details /fixed details form one of the external file called web.xml(In JDBC external file is properties/resource bundle file) ServletConfig interface object is responsible for getting technical details from web.xml file. An object of ServletConfig is automatically created by container **during the initialization time.**
3. An object of subclass of ServletConfig is nothing but sub class of SerlvetConfig. ServletConfig interfaceobject obtaining the data from web.xml file which is written in <init-param> ------</init-param> of <servlet> -----------</servlet> of web.xml file.

**EX:** Write a web.xml file which supply technical information of a database to the specific servlet.

<web-app>

<servlet>

<servlet-name>siva</servlet-name>

<servlet-class>bg.Big</servlet-class>

<load-on-startup>10</load-on-startup>

<init-param>

<param-name>driver</param-name>

<param-value>oracle.jdbc.driver.OracleDriver</param-value>

<param-name>url</param-name>

<param-value>jdbc:oracle:thin:@localhost:1521:orcl</param-value>

<param-name> user</param-name>

<param-value>scott</param-name>

<param-name> password</param-name>

<param-value> tiger</param-value>

</webapp>

</servlet>

<servlet-mapping>

<servlet-name>siva</servlet-name>

<url-pattern>/big.com</url-pattern>

</servlet-mapping>

</web-app>

Hence the <init-param> data which we have written in the above file in the DBServ is automatically available in the object of ServletConfig interface at the time of initialization phase. An object of ServletConfig contains the <init-param> data in the form of (key,value) , key represents <init-param> name and value represents <inti-param> value. Hence an object of ServletConfig exists one per servlet. ServletConfig object is available with the servlet as long as the servlet is under execution. Once servlet execution is completed automatically ServletConfig object will be removed by the container.

**Q) How many ServletConfig objects are available when a web application is deployed in which 10 are under execution 5 are under stopped mode and 5 are compiled mode?**

10 config objects are available.

**Note:** When we deploy <load-on-startup> disable servlet in the container no config object is available. Where as when we deploy <load-on-startup> enabled servlet on container , we get Config object.

**Creating an Object of ServletConfig:** An object of SerlvetConfig can be achieved in three ways they are

1. Directly calling getSerlvetConfig() from initialization of the servlet onwards

Public class DBServ extends HTTPSerlvet

{

-----------

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

}

Protected void doGet()

{

---------

----------

ServletConfig config=getServletConfig();

{

-----------

}

In the above code segment getSerlvetConfig is one of the abstract method present in Servlet interface and defined in GenericServlet and further inherited into HTTPSerlvet and finally available with our Servlet.

1. **Override public void init(ServletConfig)**

**EX:** class DBServ extends HTTPSerlvet

{

public void init(SerlvetConfigconfig)

{

//we can use configobj this context

--------

}

}

Making the ServletConfig object gloabal for the entire from initialization process onwards.

public class BDServ extends HttpServlet

{

ServletConfig comfig;

public void init(ServletConfig config)

{

this.config;

}

---------

----------

Retrieving the data from object of ServletConfig will be created automatically phase InitParam data of a servlet of web.xml file in the form of (key.value). It is desirable to retrieve the (key, value) related values from SerlvetConfig interface object for developing flexible servlet to retrieve(key, value) from ServletConfig interface object we use the following two methods.

1. public String getInitParameter(String)
2. public Enumeration getIntiParameter()

The above two methods are present in ServletConfig interface.

1. **public String getInitParameter(String):** This method is used for obtaining InitParameter value by pasingInitParameter name, This method will be used by us when we know about InitParameter names.

**EX:** String dname=config.getInitParameter(“driver”);

String url=config.getIntiParameter(“uname”);

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

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

1. **public Enumeration getIntiParameter():** this method is used to obtaining InitParameter names when the Java programmer has not having an idea about InitParam names. Pass the InitParam names to method1. We get InitParam values.

**EX:** Enumeration en=config.getInitParametrNames();

While(en.hasMoreElements)

{

Object obj=en.nextElement();

String ipn=(String) obj;

String ipr=config.getInitParameter(“ipn”);

pw.println(ipn+” “+ipr);

**P) Write a Servlet which will read the data from web.xml file by making use of ServletConfig?**

**Refer ReadServletConfigproject in NetBeans Projects**

**P) Develop a web application to deal with Database which will read the data from web.xml file by using ServletConfig**

**Refer Ne tBeans(servlets)\DatabaseServletConfig folder**

As a part of IDE we need not to explicitly write <init-param> names but while we are configuring the servlet, we use to give initialization parameter names and their values. Without manually writing in web.xml file.

When we develop any flexible servlet which deals with various databases, it is mandatory to the Java programmer to add the corresponding jar file to the IDE.

The corresponding jar file to the IDE

(Choose libraries 🡪 right click add jar)

**SerlvletContext:** The basic aim of ServletContext is to develop flexible web application. SerlvetContext interface present in javax.servlet.\* package

An object of SerlvetContext will be created automatically by the container whenever the web application is develoyed. The scope of SerlvetContext interface object exists entire web application that is object common for all the servlets in the web application hence it is treated as global object for the servlets. An object of ServletContext contains the <context-param>----</context-param> data of web.xml file in the form of key, value pair. The place of writing<context-param>------------</context-param> in the web.xml file with in <webapp> and before all servlets. The following diagram gives the view about SerlvetContextand SerlvletConfig.

**Number of ways to create concept of ServletContext:** we can create an object of ServletContext in two ways they are

1. By calling method getServletContext.
2. By using ServletConfig object
3. **By calling method getServletContext:**getServletContext is one of the method present in GenericSerlvlet and further inherited into HttpServlet and finally available with our servlet class.

public class serv extends HttpSerlvet

{

-------

-----------

Protected void goGet(-,-)

{

ServletContext context=getServletContext();

}

The prototy of getServlet context method is shown below

GenericServlet

Public servletContextgetServletContext()

1. **By using ServletConfig object:** ServletConfig interface contains the following method for creating an object of ServletContext. To create an object of ServletContext, first we need to create an object of ServletConfig and later all the following

SerlvetConfig

Public ServletContext getServletContext()

**EX:** public class serv1 extends HttpServlet

{

Protected void doGet(-,-)

{

ServletConfig config=getServletConfig();

ServletContext context=config.getServletContext();

--------

--------

}

**Number of ways to retrieve the data from ServletContext:** We know that an object of ServletContext will be created during deployment time of web application whenever the ServletContext object is created, it is by default containing <context-param>-----</context-param> data of web.xml file in the form of (key, value). To retrieve context param data we use the following methods.

1. Public String getInitParameter(String)
2. Public Enumeration getInitParameterNames()

The above two methods are present in ServletContext

**Q) How do you find Context param values and configparam values?**

A) <context-param> values of web.xml file can be obtained by making use of an object of servlet context similarly configparam values can be obtained by making use of ServletConfig interface object.

**P) Develop a web application which illustrates the concept of ServletContext and also ServletConfig?**

**For complete program please refer ServletsWhole\Servlet\NetBeans(servlets)\ServletContextANDServletConfig**

context

a=100

InitServ InitServ1

config config1

a=20

a=10

**Q) Differences between ServletConfig and ServletContext?**

|  |  |
| --- | --- |
| **ServletConfig** | **ServletContext** |
| 1. ServletConfig is exists one per servlet. 2. ServletConfig object is treated as global object to particular servlet. 3. The basic aim of SerlvetConfig is to develop a flexible servlet. 4. An object of SerlvetConfig created by the container during initialization phase. 5. The scope of the ServletConfig object is only per a particular servlet that is as long as servlet in under execution. 6. ServletConfig object contains <init-param> data of the servlet of web.xml 7. Before we make first request to web application ServletConfig object is not available. 8. An object of ServletConfig can be created directly by calling the following method   public ServletConfig getServletConfig() | 1. ServletContext is exists one per web application. 2. ServletConfig object is treated as global object to entire web application. 3. The basic aim of SerlvetContext is to develop a flexible web application. 4. An object of SerlvetContext created by the container during deployment. 5. The scope of the ServletContext object is only per a entire web application that is as long as it present in Server context. 6. ServletContext object contains <context-param> data of the web.xml 7. Before we make first request to the web application ServletContext object is available. 8. An object of ServletContext can be created directly by calling the following method   public ServletContextgetServletContext() |

**Conclusion:** The similarity of the ServletContext and SerlvetConfig is to get the external technical data like Database details fixed email addresses, fixed server details etc., from web.xml file.

Context

Servlet1 serlvet2 servlet3

config1 config2 config3

**Servlet to Servlet Communication:** We know that a web application is a collection of static web resource programs and dynamic web resource programs. The role of static web resource programs in the web application development is to accept client data. In a web application we may have multiple static web resource programs and they always reside in server running in the context of client (browser).

**EX:** HTML Programs

The role of Dynamic web resource programs in the web application development is to validate client data by residing and running in the context of server.

**EX:** Servlets and JSP.

According to real industry the server side operations must be performed in a servlet or JSP by writing various types of Request processing logics those are

1. Content management logic
2. Resource gathering logic
3. Authentication logic
4. Persistency logic
5. Business logic etc.,

It is not recommended to write all the above types of logics in a single servlet because servlet will become heavy weight component but servlets must be always light weight components rather than writing all types of logics in a single servlet and getting as a heavy weight component, industry is highly recommended to write different types of logics in different types of servlets for making a servlet component as light weight component and also it makes the programmer for easy maintenance of web project.

**Servlet Chaining Def:** The mechanism of processing the single request of client request by **multipleservlets** for giving single response is known as **Servlet Chaining, servlet collaboration.**

(or)

If multiple servlets participating in a single request of a client then these multiple servlets are known as servlet chaining or **servlet collaboration.**

(or)

Single request of the Client is able to be chaining and processing between multiple servlets then the communication between these servlets is known as Servlet communication.

\*In servlet chaining concept always there exists single request and single response but not multiple requests and multiple responses.

In the above web application development we found or we come across the following communication modes.

1. Establishing the communication between multiple servlets of same web application which are running in the same server.

browser

Client

request

1. Establishing the communication between multiple servlets which are available in multiple web applications running in same server (Supported by application server not web server). Web logic server

Request

S1

Web application 1 web applciation2

Client

Request

S2

Response

1. Establishing the communication between multiple servlets which are available in multiple web applications which are running in **different servers.**

GlassFish Server Weblogic

Web applicaton

S1

Web application

S2

Process

<http://localhost:8080/is/serv.com>

Client

**Implementation of Servlet chaining:** In order to establish the communication between multiple servers which are available in same web application or different web application which are running in the same server. We use an object of Request dispatcher interface.

**RequestDispatcher:** It is one of a predefined interface present in javax.servlet.\* package and whose object is used for establishing the communication between multiple servlets which are present in same or different web applications which are running in same server. Programatically an object RequestDispather interface is nothing but creating an object of sub class of RequestDispather interface.

**Number of ways to create an object of RequestDispather interface:**  An object of RequestDispatcher interface can be created in three ways they are

1. Request based RequestDispatcher
2. Context based RequestDispatcher
3. Context based named RequestDispather
4. **Request based RequestDispatcher:** Request based RequestDispatcher allows us to establish the communication between multiple servlets of same web application but not the communication between multiple servlets of different web applications. Programmatically to create Request based RequestDispatcher we use the following method present in HttpServletRequest.

HttpSerlvetRequest

Public RequestDipatchergetRequestDipatcher(String)

Here String parameter represents url pattern of destination servlet present in same web application.

**EX:** protected void doGet(-,-)

{

----

-----

RequestDipatcherrd=req.getRequestDispatcher(“/s2url”);

Here writing ‘/’ is opetional. Here rd is known as RequestDipatcher.

1. **Context based RequestDispatcher:** Context based request dispatcher is used for establishing the communication between multiple servlets of same web application or different web application. Context based request dispatcher can be obtaining by using the following method.

ServletContext

Public RequestDispatcher.getRequestDispatcher(String)

Here String parameter represents url pattern of the destination servlet which may belongs to either same or different web application.

public class servlet extends HttpServlet

{

protected void doGet(-,-)

{

ServeletContext context=getServletContext()

RequestDispatcher rd=context.getRequestDispatcher(“/surl”);

//Here writing ‘/’ is mandatory

1. **Context based named RequestDispatcher:** This approach is also used for establishing the communication between multiple servlets of same and different web applications but NamedRequestDispatcher communicates with destination servlet on the basis of logical name of the servlet and approach2 communicates with destination servlet on the basis of url pattern of the servlet.

To get context based named dispatcher we use the following method

ServletConfig

Public RequestDispatchergetNamedDispatcher

Here String parameter represents logical name of the servlet.

**EX:** public class ser1 extends HttpServlet

{

Protected void doGet(-,-)

{

ServletContext context=getServletContext();

RequestDispatcher rd=context.getNamedDispatcher(“siva”);

Here **Siva** is logical name of the servlet in web.xml file.

**Types of models in Request Dispatcher:** While we are establishing the communication between multiple servlets of same web application or different web application through the RequestDispatcher we have two types of models they are

1. Forward model
2. Include model
3. **Forward model:** In forward model the Request of the client taken by the source servlet, performs Request processing logic and for the rest of the Request processing logic, source servlet forward its Request and Response objects to the destination servlet. Destination servlet completes the rest of the Request processing logic and gives its result back to the client directly without giving the result of source servlet. The following diagram gives the work flow of the forward model.

**browser web application**

forward

Result

Of Dispatcher servlet

**request**

**response**

**Client**

1. Client makes a request to the source servlet explicitly[http://localhost:8080/web/s1url Source servlet s1](http://localhost:8080/web/s1url%20Source%20servlet%20s1) receives client request and make it to available in the Request object of source servlet.
2. Source servlet makes use of the client Request data and performs its Request processing logic and place into Response object.
3. Source servlet forward its Request and Response object to the destination servlet s2 for further Request processing logic.
4. Destination Servlet(s2) performs its rest of the Request processing logic and making to available in Response object of destination servlet.
5. Destination servlet finally gives result back to the client destination servlet result only can be seen on the browser window but not the result of the source servlet(S1).
6. Client receives the result of destination servlet(<http://localhost:8080/web/s1url>). Programmatically to implement the forward model we use the following method which is present in RequestDispatcher interface.

RequestDispatcher

Public void forward(ServletRequest, ServletResponse)

S1.java(Source Servlet)

RequestDispatcher rs=req.getRequestDispatcher(“/s2url”);

Rd.forward(req,res)

-----

------

The above block of statements must be written Request Processing phase related method. In forward model one servlet/JSP (dynamic web resource programs) can forward to destination servlet/JSP dynamic web resource program only but not static web resource programs(HTML. JS, DHTML) because static web resource programs does not have Request processing capabilities in this model one dynamic web resource program can forward request and Response to only one dynamic web resource program but not multiple dynamic web resource programs because when we use forward method in the source servlet immediately the control of the server switch over to destination servlet so that there is no meaning of using multiple forward methods in source servlets.

1. **Include model:** In include model when the client makes a request control goes to the soruce servlet, source servlet obtains the client request and including the result of multiple dynamic web resource programs and multiple static web resource programs and client receives the entire response from source servlet. The following diagram gives work flow of include model.

**Browser web application**

S1

Include(req,res)

Include(req,res)

Include(req,res)

Result of Source servlet

S1

**Request**

S2

**Response**

HTML

1. Client makes a first request to the source servlet.<http://localhost:8080/web/s1url>
2. Source servlets receives the client request data.
3. A source servlet may include the multiple dynamic web resource programs for performing Request processing logic on the same Request and Response objects of source servlets and also include static web resource programs(HTML programs) for fixed content generates such as headers and footers of website
4. The dynamic and static web resource programs runs in the context of source servlet(4) and makes the result is ready in the context of source servlet(5).
5. The result of both source servlet and destination servlets and HTML programs given back to the client in the form of response through the context of source servlet(<http://localhost:8080/web/s1url>).
6. Client receives the result of web application which is collectively processed by source, destination servlet and HTML programs.

Hence the include model a source can include any number of dynamic web resource programs and static web resource programs. Programmatically to implement the include model we use the following method present in the RequestDispatcher interface.

RequestDispatcher

public void include(ServletRequest, ServletResponse)

ReqestDispatherrd=req.getRequestDispatcher(“s2url”);

Rd=req.getRequestDispather(“header.html”);

**P) Develop a web application which illustrate the concept of forward and include model.**

**Refer ForwardAndIncludeModels project in Netbeans Projects**

**P) Develop a web application which illustrates the concept of authentication including of headers and footers by communication between multiple servlets.**

**ReferAuthenticationIncludingHeadersAndFooters project in NetBeans Projects.**

**Browser webApplication**

<http://----login.html>

UName

Paswrd

Siva Tech

Login success

Login denied

Login.html

RecServlet.com

Forward(req,res)

Header.html footer.html

DispServlet.java

**Inserting, modifying and deleting Request object:** We know that whenever client submits HTML form data to the servlet, the HTML form data available in the servlet in the form of an object of HTTPServletRequest.

An object of HTTPServletRequest containing the HTML form data in the form of (key, value). In the (key,value) key represents form parameter name or Request parameter name and value represents form parameter value or Request parameter value.

In some of the circumstances of web application we my be performing the following operations on the Request object.

1. Inserting(key, value) in the Request object
2. Modifying value of value based on key value.
3. Removing the complete entry(key, value) from Request object. To perform the above three operations we us the following methods.
4. Public void setAttribute(String, Object)
5. Public void removeAttribute(String)
6. Public Object getAttribute(String)

The above methods are present in ServletRequest interface and further inherited into HTTPServletRequest.

1. **Public void setAttribute(String, Object) :** This method is used for performing both inserting and modifying the entries of Request object. If new form parameter name is **not already** in Request object then (key, value) will be taken as inserted **entry** if the new form parameter name **already present** in the Request object then the **value of value** will be modified based on key value. This entry is treated as modified entry.

**EX:** request.setAttribute(sal, new Float(3.5f)

request.setAttribute(“name”,”siva”)

request.setAttribute(“sal”, new Float(4.5f))

1. **Public void removeAttribute(String):** This is used for removing the entry of request object based on key.

**EX:** request.removeAttribute(“empno”)

1. **Public Object getAttribute(String):** This method is used for obtaining **value of value** by passing key

**EX:** Object eno=request.getAttribute(“empno”);

Object ename=request.setAttribute(“ename”);

Object sal=request.getAttribute(“sal”);

String ename1=(String) ename;

Float fo=(Float)sal;

float f1=fo.floatValue()

In the above methods the parameter represents Request Parameter name/key. Object parameter represents value of value/Request parameter value.

**Q) What is difference between HTML form parameters and form parameter names added to the Request object in the servlet.**

**A)** HTML form parameters are available in the ServletRequest interface object in the form of (key, value). Both the values are treated as String type. Where as the Request names whicha re specially in the servlet to the Request object in the form of (key, value) key is treated as String type and value of value is treated as java.lang.Object

**P) Develop a web application which illustrate the concept of modifying, inserting and deleting Request parameter names and values and include and forward model by dealing with the Database.**

**For complete program please referNetBeans(servlets)\InsertingDeletingModifyingRequestParamNames program.**

**Establishing the communication between multiple web applications which are running in the same server:**

1. We know that Request based RequestDispatcher can establish the communication between multiple servlets of same web application which are running in the same server
2. But Request based RequestDispatcher never establish the communication between multiple servlets of different web applications which are running in the same server in order to establish the communication between multiple servlets of different web applications which are running in the context of same server
3. we must create an object of **Context based RequestDispatcher.** The following steps are used for establishing the communication between multiple servlets of different web applications which are running in the same server.
4. Obtain an Object of ServletContext interface in the source servlet of web application that is obtaining current web application context.

**EX:** SerlvetContextcsctx=getServletContext();

1. Obtain an object of ServletContext of different web applications with respect to current web application context for communicating with destination servlet of other web application to do this we use the following method.

ServletContext

Public ServletContext(String)

Here the parameter String represents name of the other web application which is running in the same server.

**EX:** ServletContext octx=csctx.getCContext(“/wa2”);

Destination web application name.

1. Obtain an object of RequestDispatcher with respect to another web application context based RequestDispather.

**EX:** RequestDispatherrd=osctx.getRequestDispather(“/s2url”);

1. Appply either include or forward model for exchanging the Request and Response objects between multiple servlets of different web applications

rd.include(req, res) (or) rd.forward(req,res)

The above steps are expressed in the following diagram

csctx🡨----------------🡪csctx

wa1 wa2

rd.include(req, res)

(or)

rd.forward(req,res)

web.xml web.xml

**P) Write a web application which will establish the communication between multiple servlets of different web applications which are running in the same server.**

**Note: This program is not running in Glass Fish server. So we have to configure weblogic server. Otherwise open weblogic server outside to the IDE. Please refer coms (weblogic) or coms2 (weblogic) projects in NetBeans projects**

**Establishing the communication between multiple servlets of multiple web applications which are running in different servers:**

An object of RequestDispatcher can establish the communication between either they belongs to same web application or different web application which are exceeding same server but an object of RequestDispather can’t establish the communication between multiple servlets of different web applications which are running in different servers. The following diagram gives the sequence of steps for establishing the communication between multiple servlets of different web applications which are running in the different servers.

Tomcat Web logic

S2

S1

Explicit Request Explicit response Implicit Request Implicit Response

http://localhost:8080/web/s1url

Resultt of destination servelet running on different servlers

1. Client makes first request to the web application which is running in one server.
2. Whatever the request made by the client in step-1 will be treated as explicit request to the servlet of web application which is running in one of the server.
3. Servlet receives the explicit request of the client, process the RequestProcessing logic
4. Source servlet of one web application which is running in one server gives **implicit response** which is nothing but an implicit request to destination servlet of different web application which is running in the different server.
5. The request which is making to the servlet of different web application running in another server is treated as implicit Request (which is not made explicitly by client but it is made source servlet of different web application of another server).
6. Destination servlet receives the implicit Request process and the Request processing logic.
7. Destination servlet gives response to the client.
8. Result of the destination servlet received by client.

The entire above communication model be make explicit request to the source servlet of one server and we get the explicit response from destination servlet of another application server.

Programatically in order to achieve the communication between multiple servlets/ multiple web applicatios/ multiples servlets, we have to use the following method.

ServletResponse

Public void sendRequest(String)

Here String parameter represents avctualurl of destination servlet of different web applications running in the different server. This model internally gives implicit response which will become implicit request to the destination servlet which is running in the different server. Res.sendRedirect(“http://localhost:7001/wa1/s2url)

The above statement written as a part of source servlet.

**Note:** It is highly recommended to write any statements which are followed by sendRedirect(-).

**EX:** SUN Micro system is handed over by Oracle corporation. Across the globe SUN Microsystem having lot of customers they know the url of SUN Microsystem. Every SUN Microsystem customer makes a request to SUN Microsystem website . It receives the request of the the customer and refirect to the Oracle. Oracle website receives the implicit request from SUN Microsystem, process the client Request and gives explicit response back to the SUN Microsystem client.

**P) Develop a web application which illustrates the concept of servlet to servlet communication which are running in different servers.**

**Please refer communicationBetweenMultipleServers, coms2 projects in NetBeanspeojects.**

**Q) what is the difference between forward(-,-) and sendRedirect(String)?**

|  |  |
| --- | --- |
| **forward(-,-)** | **sendRedirect(Stirng)** |
| 1. These methods will establish the communication between multiple servlets which are running in the same server and they may present in same web application or different web application. 2. These models exchanges the data between multiple servlets in the form of Request and Response objects. 3. In this model same references of Request objects will be maintained between multiple servlets which are running in the same server. 4. In this model the result of destination servlet is given to the client to the source servlet url. | 1. These methods will establish the communication between multiple servlets which are present in multiple web applications and running in different servers. 2. In this model the data exchanging between servlets which are running in different servers in the form of QueryString(<http://localhost:7001/s2url?a=10@b=20>) 🡪 Query String 3. In this model different references of Request and Response objects will be maintained across the multiple servlets running in different servers. 4. In this model the results of the destination servlets which is running in different server is directly given to the client which its own url but not through source servlet. |

**Q) In How many number of ways are there to pass the data to the servlet/the Request to the servlet?**

A) We can make a request to the servlet in three ways they are

i) Form based request

ii) Query String based request

iii) Hyperlink based Request

**Q) What are the different types of url patterns of a servlet?**

**A)** In our web application we can use three types of url patterns they are

1. Exact Match

2. Directory Match

3. Extension Match

1. **Exact Match:**

**EX:** <url-pattern>/siva</url-pattern>

1. **Directory Match:**

**Syn:** a **)**<url-pattern>/siva/hall1/\* </url-pattern>

**EX:** <http://localhost:8080/hall1/x.java> (valid)

b)<http://localhost:8080/hall1/hall2/hall3/x> -valid

c) [http://localhost:8080/siva/\*.c(invalid)](http://localhost:8080/sathya/*.c(invalid))

d) [http://localhost/hyd/siva/i.sat(invalid)](http://localhost/hyd/sathya/i.sat(invalid))

**Note:** Exact Match will fix with the directory match

**Extension Match:**

**Syn:** <url-pattern>\*.do </url-pattern>

**EX:** http://localhost:8080/siva/\*.java

Valid

Htpp://localhost:8080/siva/abc

Valid

**Note:** Extension Match should not fix with Direct Match

**Form Validations:** In the web application development a form is a collection of GUI components/ controls

1. In web application development a form can developed by making use of html controls but not by using SWING/AWT/Applets components because HTML GHI Components are more easy to use and implement than SWING components.
2. Hence industry always recommends develop web application related forms with HTML controls.
3. For product based development like Tomcat, GlassFish, Weblogicservet side technologies it is recommended that to develop with SWING components.
4. HTML forms are known as static web resource programs. We know that the role of static web resource programs are to accept client side data and handover tot eh servlet side program.
5. In most of the web application development the data which is submitted to the client must be validated.
6. In web application development we have two types of validations those are
7. client side validations
8. server side validations
9. **client side validations:** The process of checking the consistency of form data either in client side or server side is known as **form validation.**
10. Client side validations are those which can be validated at client machine(browser) by making use of Java and VB Script. These comes under Client side validation technologies. There is no relation between Java and Java script.
11. **server side validations:** Server side validations are those which will check consistency of client data in the server by making use either servlet or JSP. Hence servlets and JSP are called server side scripting technologies.

Let us consider the following form

Enter Student No

Enter Student Name

Enter Student Marks

Submit

Reset

**Validations on Student No:**

1. student No should not be null.
2. Student No should not contain space.
3. Student No must be integer

**Validations on Sname:**

1. Student Name should not be null.
2. Student Name should not contain space.

**Validations on Marks:**

1. Student marks should not be null.
2. Student marks should not contain spaces.
3. Marks must be integers

To carry out all the above validations on student form we need to write the block of statements in the Servlet/JSP such validations are known as server side validations.

**For example program please refer Formvalidation project in NetBeans projects**