102010505
Advanced Java

Unit-3 (Part-I) Servlet API and Overview

What is Servlet?

" Servlet is java class
which
extends the functionality of web server
by
dynamically generating web pages."

Servlet: Basic Terms

- Before looking into Servlet, we will see some important keywords about web application.

Web Client: We will call **browsers** (IE, Chrome, Mozilla etc.) as a **Client**, which helps in communicating with the server

Http Request

Client

Server and Client (browser) will communicate with each other with the help of **HTTP** protocol.

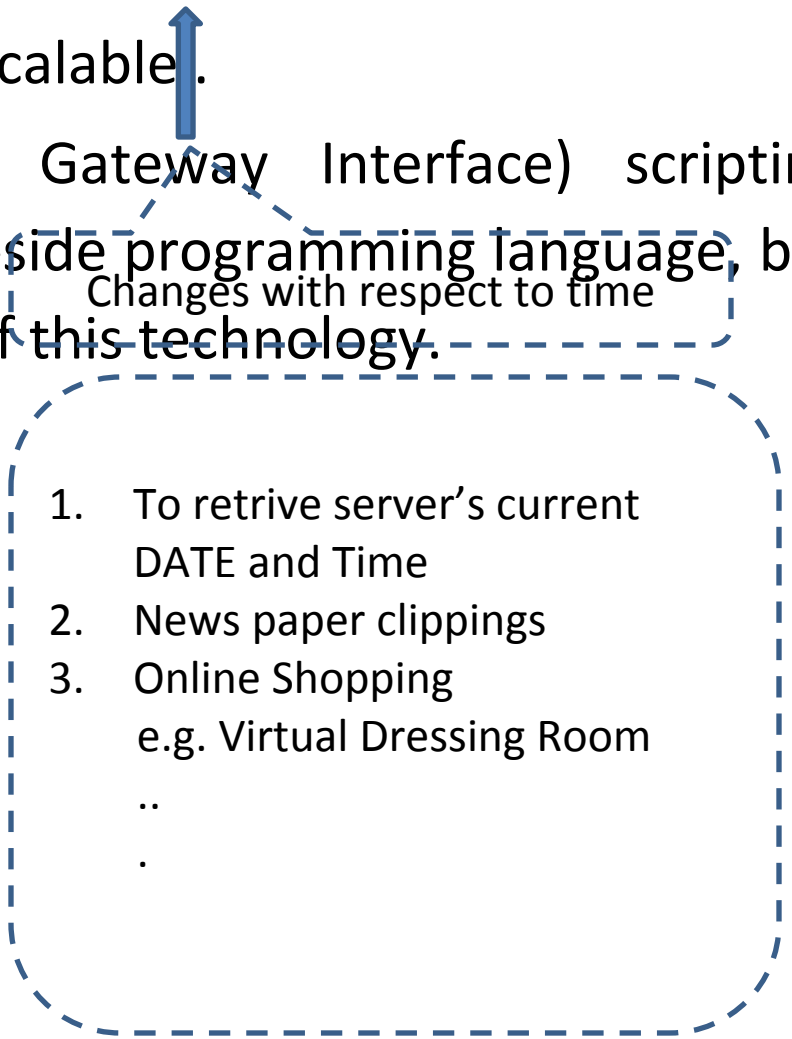
Web Server is the one which takes the client request, process the request and sends back the **response**.

Server

Http Response

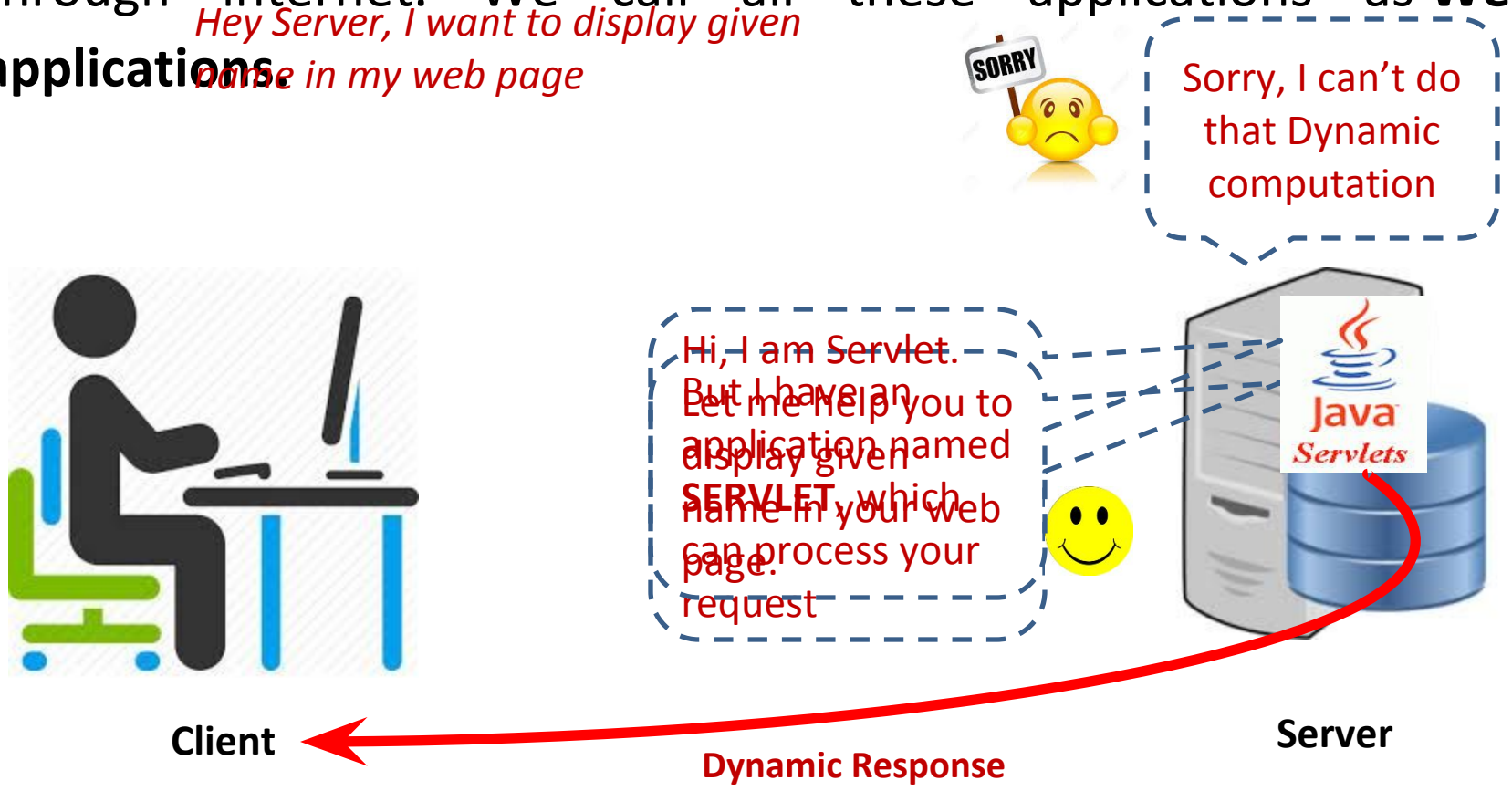
Introduction

- **Servlet** technology is used to create **Dynamic** web application
- **Servlet** technology is robust and scalable.
- Before Servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language, but there were many disadvantages of this technology.



Why we need Servlet?

- Nowadays everything is available on Internet.
- Starting from e-banking, e-commerce everything is available through internet. We call all these applications as **Web applications**.



Scripting Language

Scripting Language

Server-Side Scripting Language

PHP
ASP.NET
(C# OR Visual Basic)
C++
Java and JSP
Python
Ruby on Rails etc.

Server-side scripting is often used to provide a customized interface for the user.

Client-Side Scripting Language

JavaScript
VBScript
HTML (Structure)
CSS (Designing)
AJAX
jQuery etc.

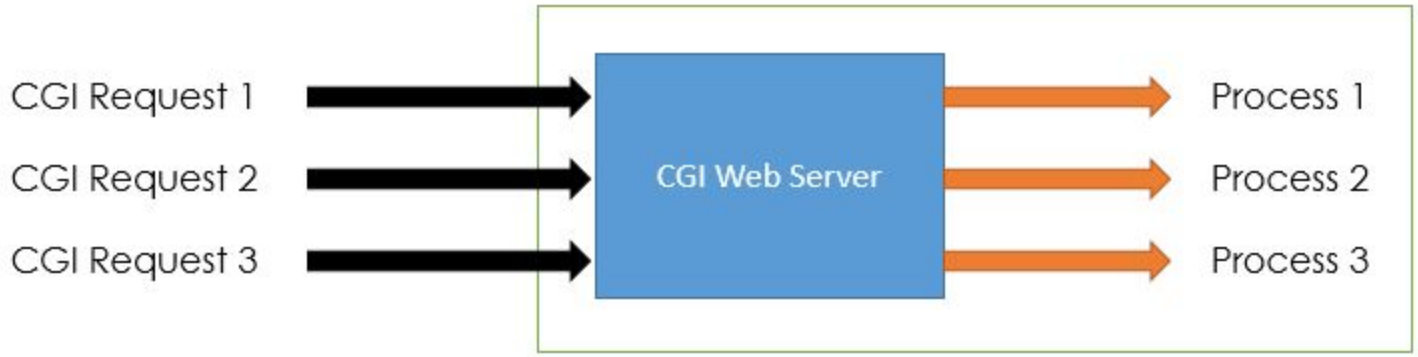
Client-side scripting is an important part of the Dynamic HTML. Usually run on client's browser.

CGI (Common Gateway Interface)

- CGI was the 1st server-side scripting technique for creating dynamic content.
- CGI is used to execute a program that resides in the server to process data or access databases to produce the relevant dynamic content.

CGI (Common Gateway Interface)

- For each request CGI Server receives, It creates new Operating System Process.



- If the number of requests from the client increases then more time it will take to respond to the request.
- As programs executed by CGI Script are written in the native languages such as C, C++, perl which are not portable.

Image Reference:http://www.c4learn.com/java/servlet/servlet-vs-cgi/

Comparing Servlet with CGI

- CGI programs are used to execute programs written inside the native language.
- While in Servlet, all the programs are compiled into the Java bytecode, which is then run in the Java virtual machine.
- In Servlet, all the requests coming from the Client are processed with the threads instead of the OS process.

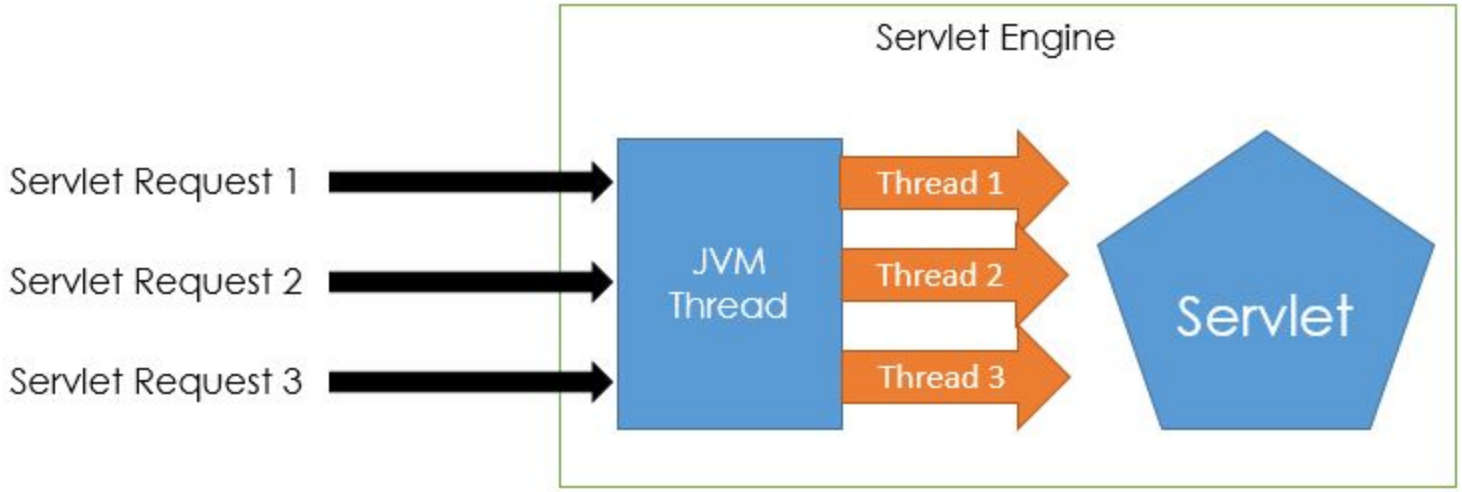


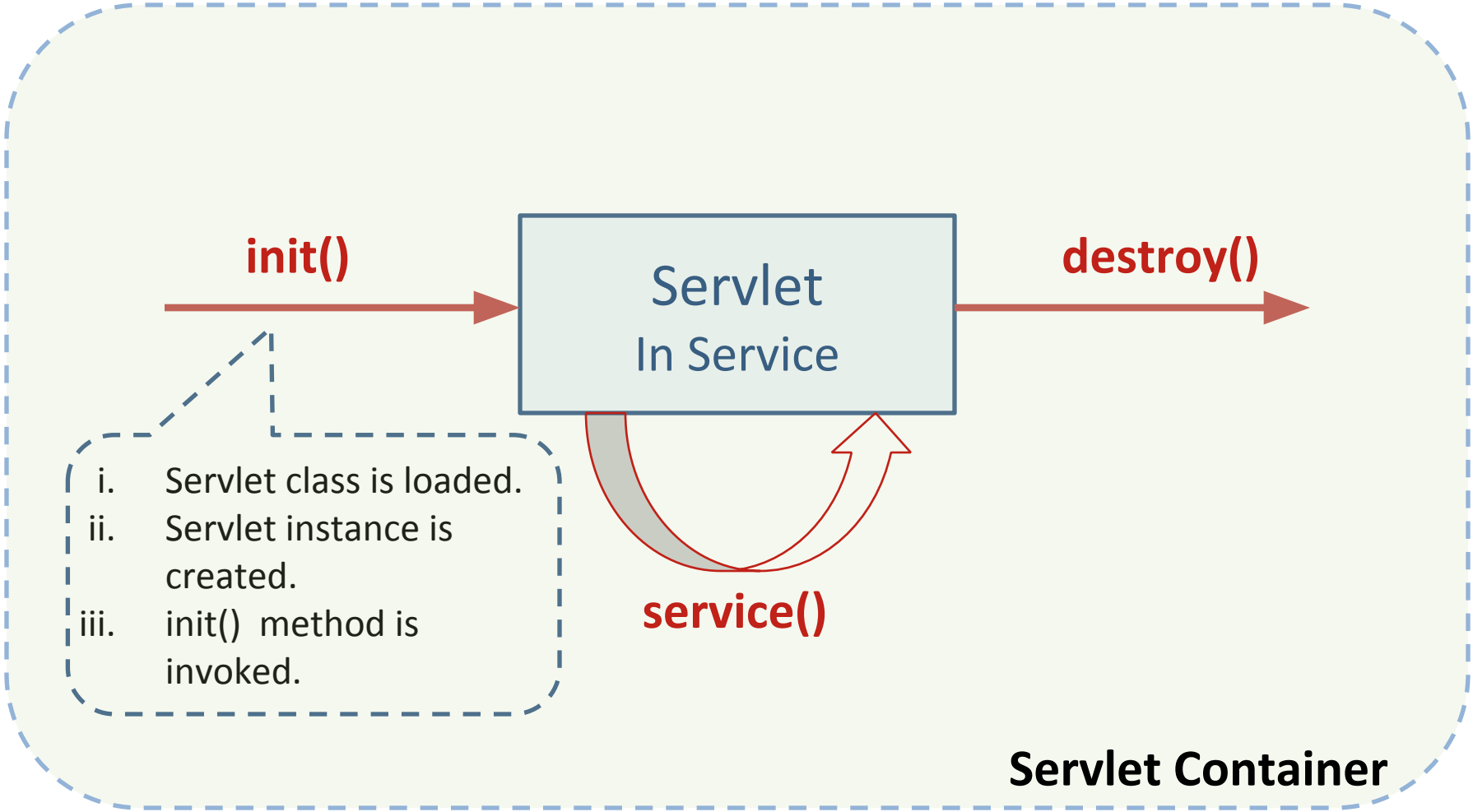
Image Reference:http://www.c4learn.com/java/servlet/servlet-vs-cgi/

Summary: CGI vs Servlet

CGI	Servlet
CGI was not portable.	Servlets are portable.
In CGI each request is handled by heavy weight OS process.	In Servlets each request is handled by lightweight Java Thread.
Session tracking and caching of previous computations cannot be performed.	Session tracking and caching of previous computations can be performed
CGI cannot handle cookies.	Servlets can handle cookies.
CGI does not provide sharing property.	Servlets can share data among each other.
CGI is more expensive than Servlets	Servlets is inexpensive than CGI.

Servlet Life Cycle

Servlet Life Cycle



Servlet Life Cycle: init()

i. Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the **web container.**

ii. Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

A Web application runs within a **Web container** of a Web server. Web container provides runtime environment.

iii. Init() method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet.

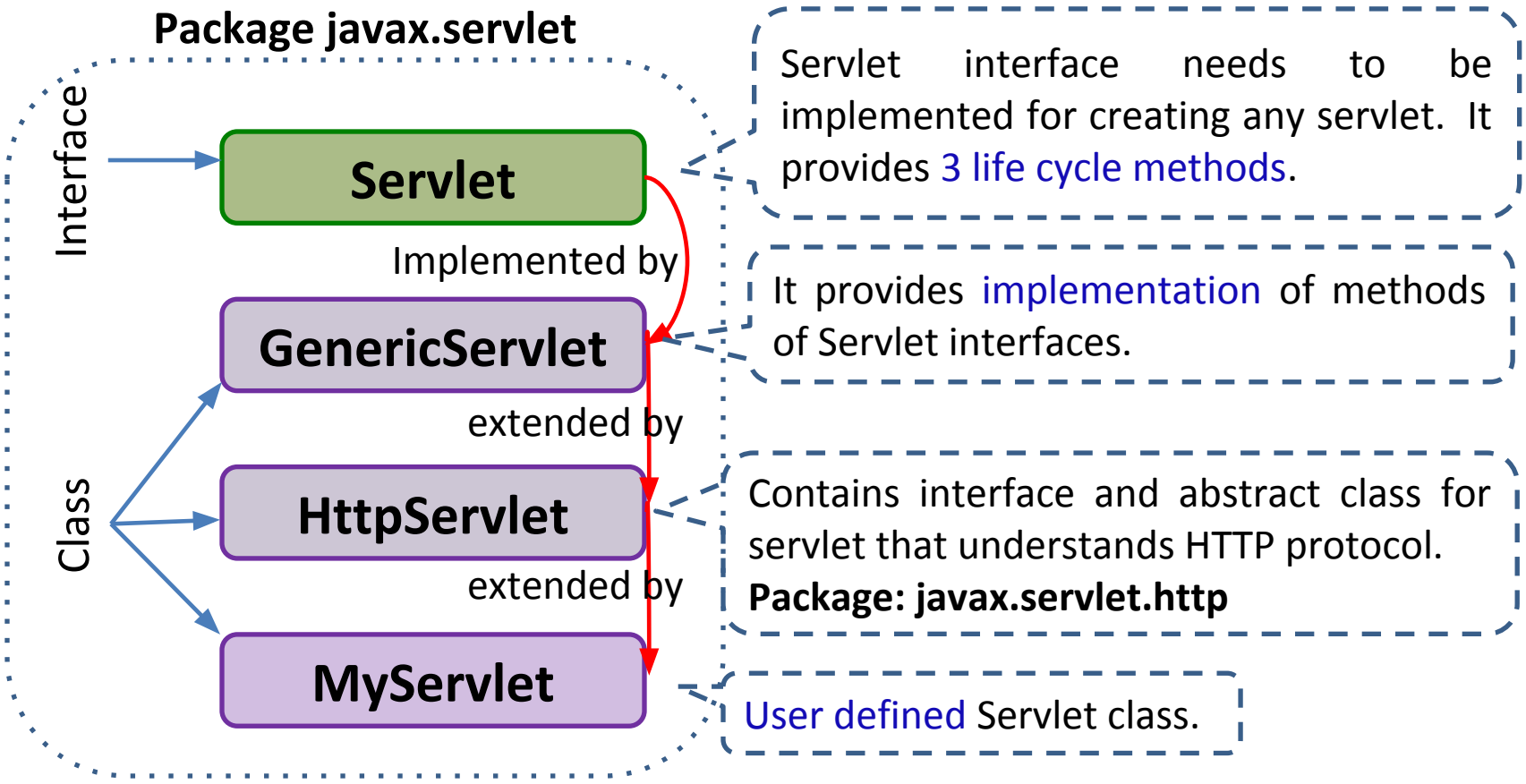
Servlet Life Cycle: init()

Syntax:

```
public void init(ServletConfig config)
                    throws ServletException
{
   //initialization…
}
```

A servlet configuration object used by a servlet container to pass information to a servlet during initialization process.

Servlet Packages



Servlet Life Cycle: Service()

- The service() method is the main method to perform the actual task.
- The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the response back to the client.
- Each time the server receives a request for a servlet, the server spawns a new thread and calls service.

Servlet Life Cycle: Service()

Syntax:

```
public void service(ServletRequest request,
                    ServletResponse response)
            throws ServletException, IOException
{
    …
    …
}
```

Servlet Life Cycle: Service()

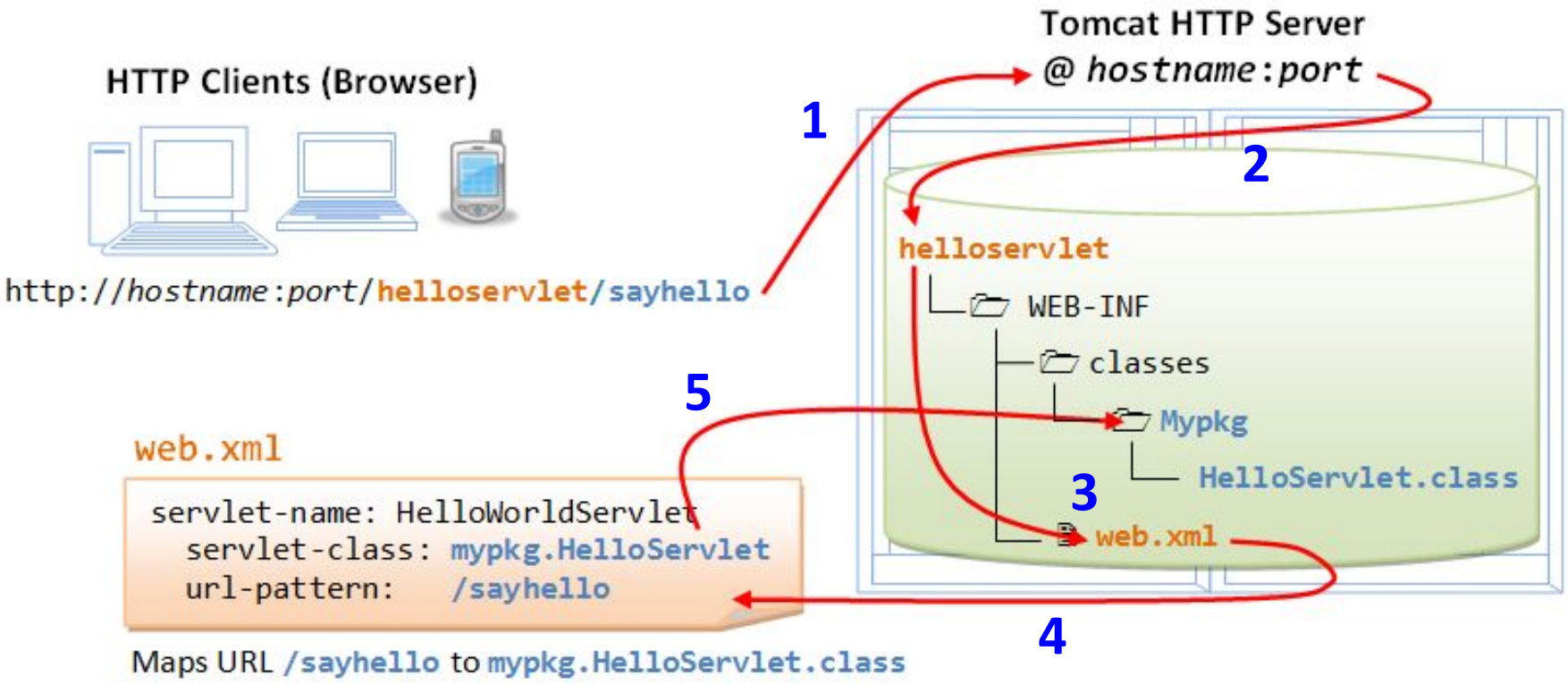
- The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.
- The doGet() and doPost() are most frequently used methods with in each service request.

Servlet Life Cycle: Destroy()

- The destroy() method is called only once at the end of the life cycle of a servlet.
- This method gives your servlet a chance to close
  i. database connections,
  ii. halt background threads,
  iii. write cookie lists or hit counts to disk, and
  iv. perform other such cleanup activities.
- After the destroy() method is called, the servlet object is marked for garbage collection.

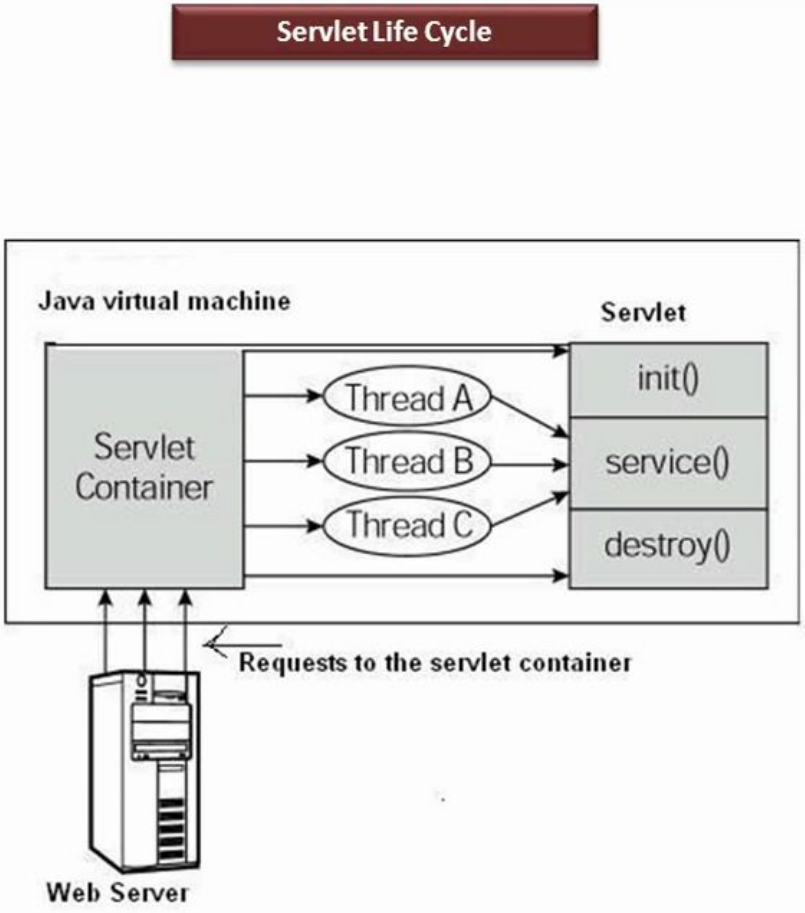
Servlet Life Cycle: Destroy()

```
public void destroy()
{
   // Finalization code...
}
```



Ref: https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html

Servlet Life Cycle



doGet() v/s **doPost()**

doGet()

A GET request results from request for a URL or from an HTML form, should be handled by doGet() method.

Syntax:

```
public void doGet
(HttpServletRequest request,HttpServletResponse response)
 throws ServletException, IOException
{
    // Servlet code …
}
```

doPost()

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

Syntax:

```
public void doPost
(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    // Servlet code …
}
```

doGet() vs doPost()

- doGet() and doPost() are HTTP requests handled by servlet classes.
- In **doGet**(), the parameters are appended to the URL and sent along with the header information.
- In **doPost**(), the parameters are sent separately.

doGet() vs doPost()

Application

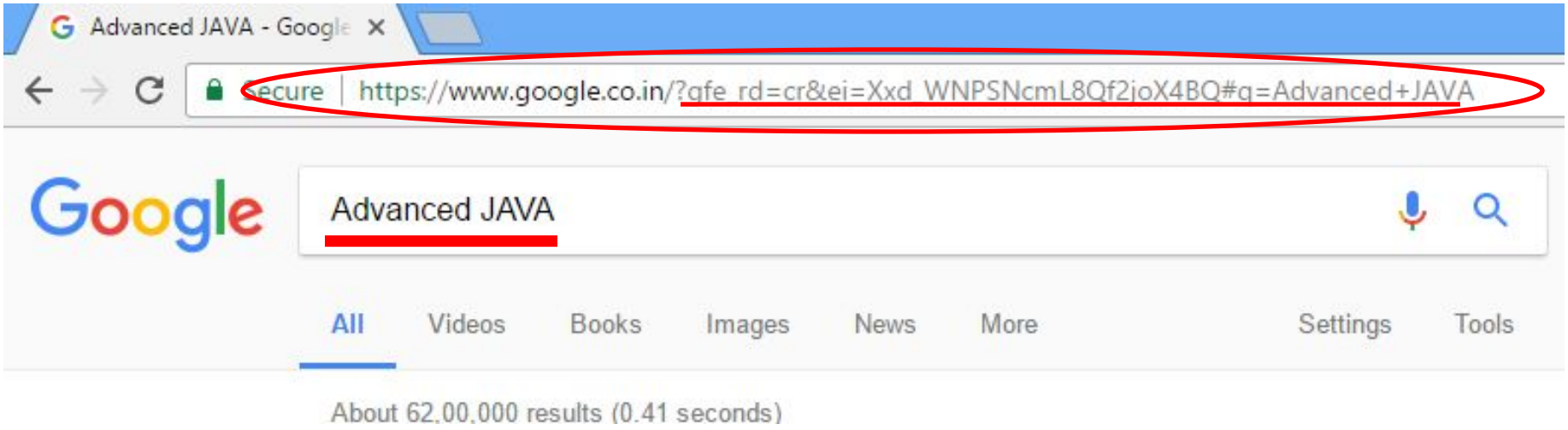
- doGet() shall be used when small amount of data and insensitive data like a query has to be sent as a request.
- doPost() shall be used when comparatively large amount of sensitive data has to be sent.

E.g.

Sending data after filling up a form or sending login & password.

doGet() vs doPost()

- Example: doGet()



doGet() vs doPost()

doGet()	doPost()
In this method, parameters are appended to the URL and sent along with header information	In doPost(), parameters are sent in separate line in the body
Maximum size of data that can be sent using doGet() is 240 bytes	There is no maximum size for data
Parameters are not encrypted	Parameters are encrypted here
Application: Used when small amount of insensitive data like a query has to be sent as a request. *It is default method.*	***Application***: Used when comparatively large amount of sensitive data has to be sent. E.g. submitting sign_in or login form.
doGet() is faster comparatively	doPost() is slower compared to doGet() since doPost() does not write the content length

Servlet Life Cycle: Servlet Code

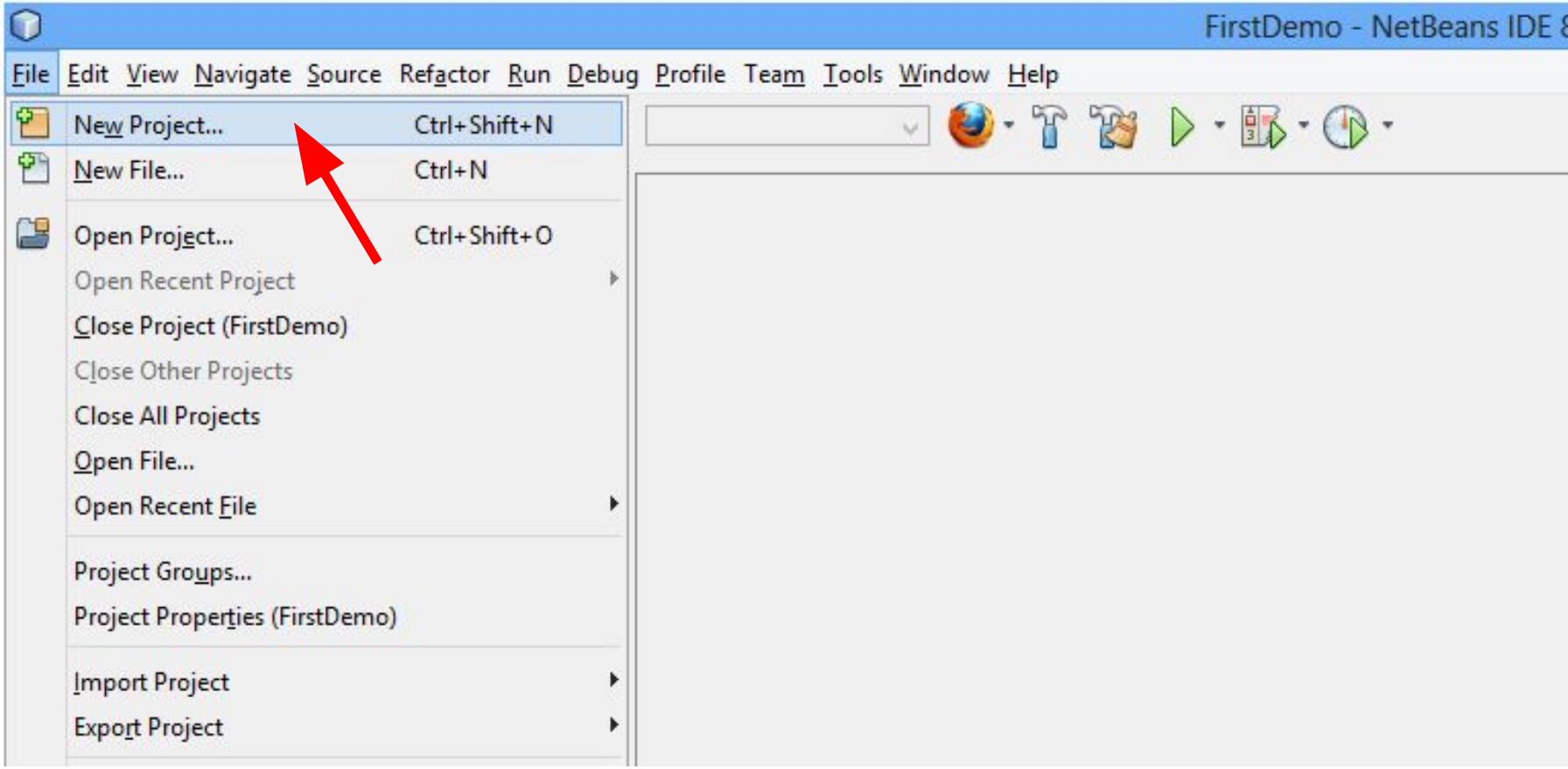
```
import java.io.*;
import javax.servlet.*;
public class MyServlet1 extends GenericServlet
{
    public void init() throws ServletException
    {//Initailization Code
    }
    public void service(ServletRequest request,
            ServletResponse response) throws
ServletException,IOException
    {//Servlet code
    }
    public void destroy()
    {//Finalization Code
    }
}
```

Steps to run Servlet Program in

Using Netbeans IDE

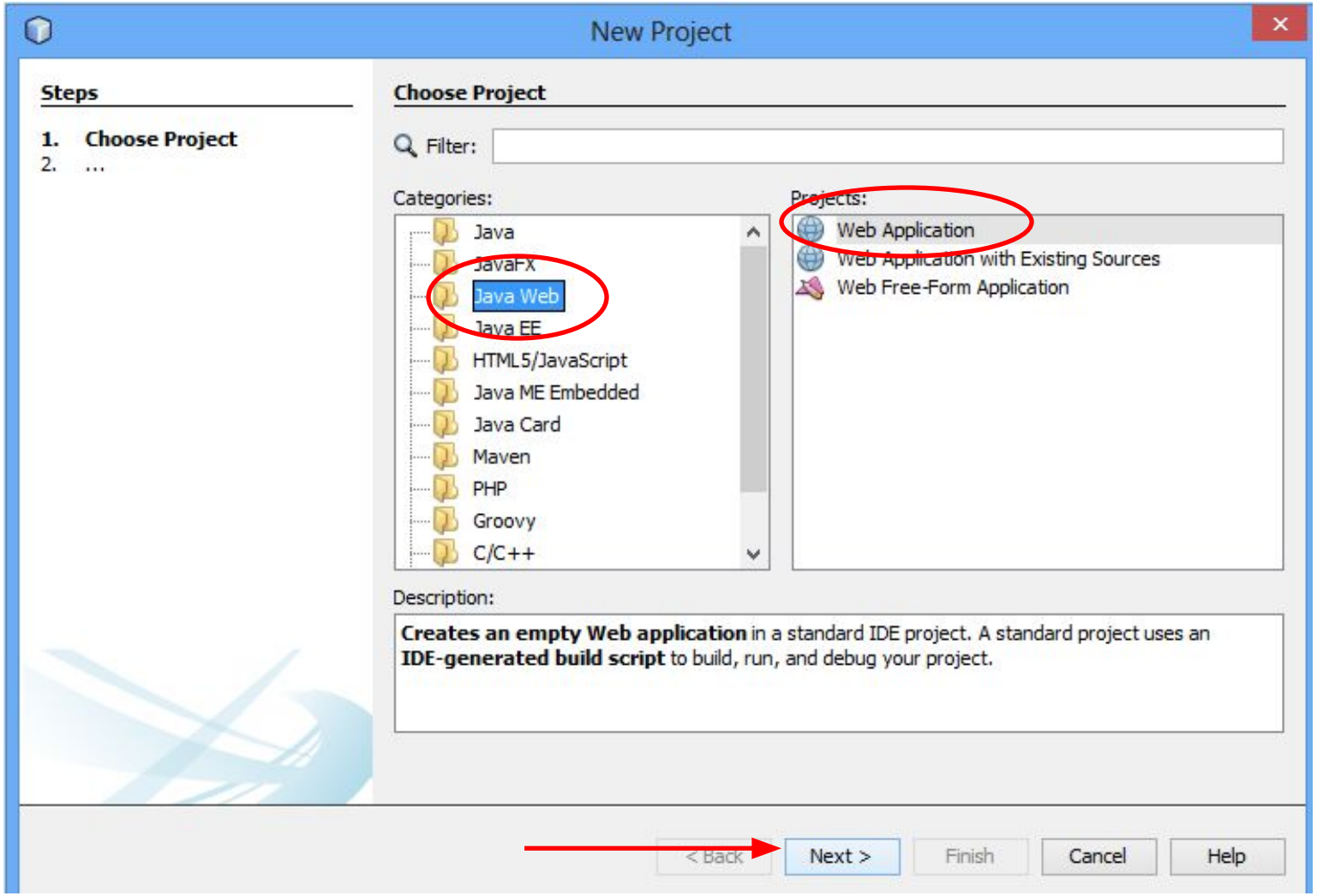
Steps to run Servlet Program

- Step 1: Open Netbeans IDE, Select **File** -> **New Project**



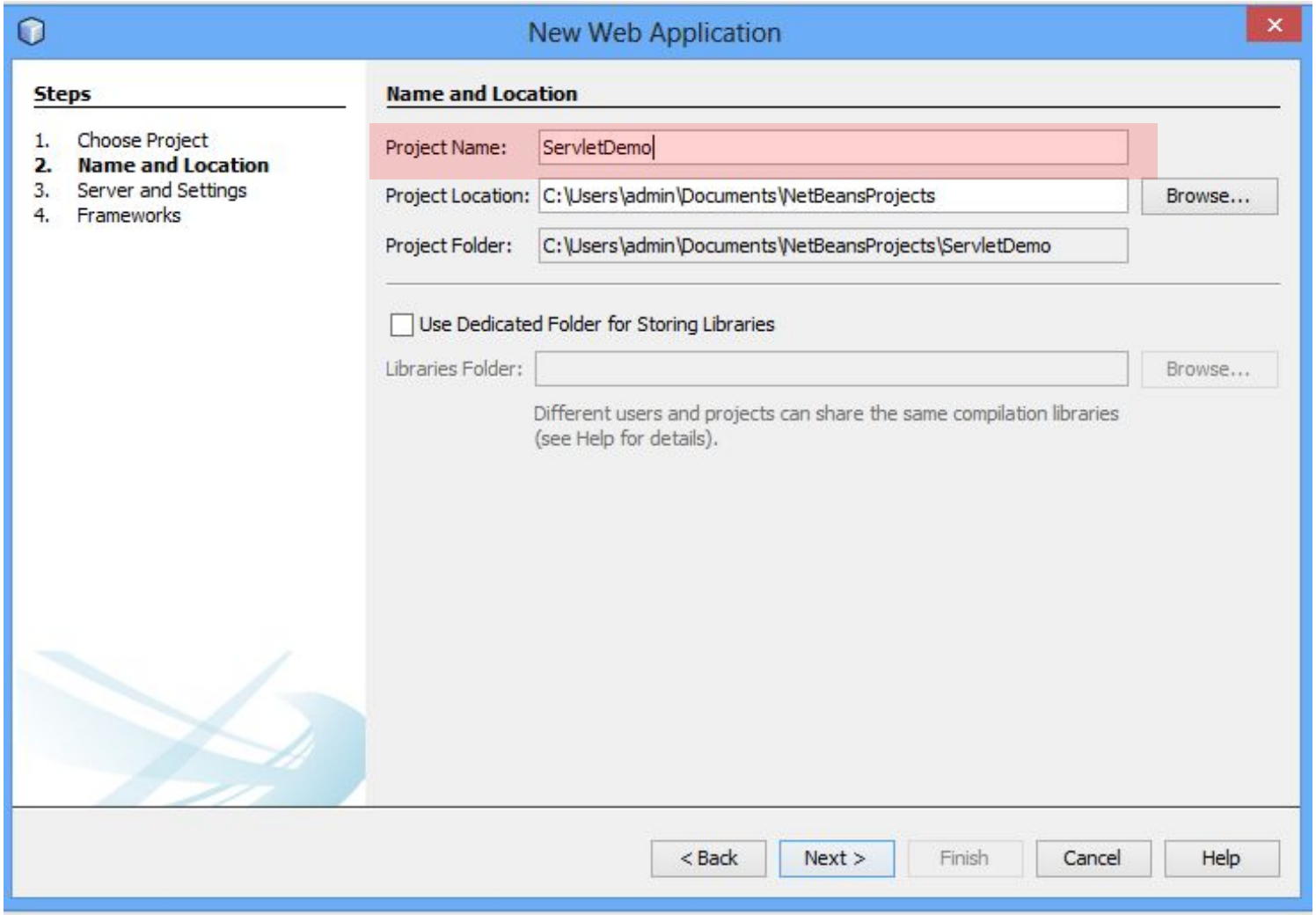
Steps for Servlet Program

Step 2: Select **Java Web** -> **Web Application**, then click on Next



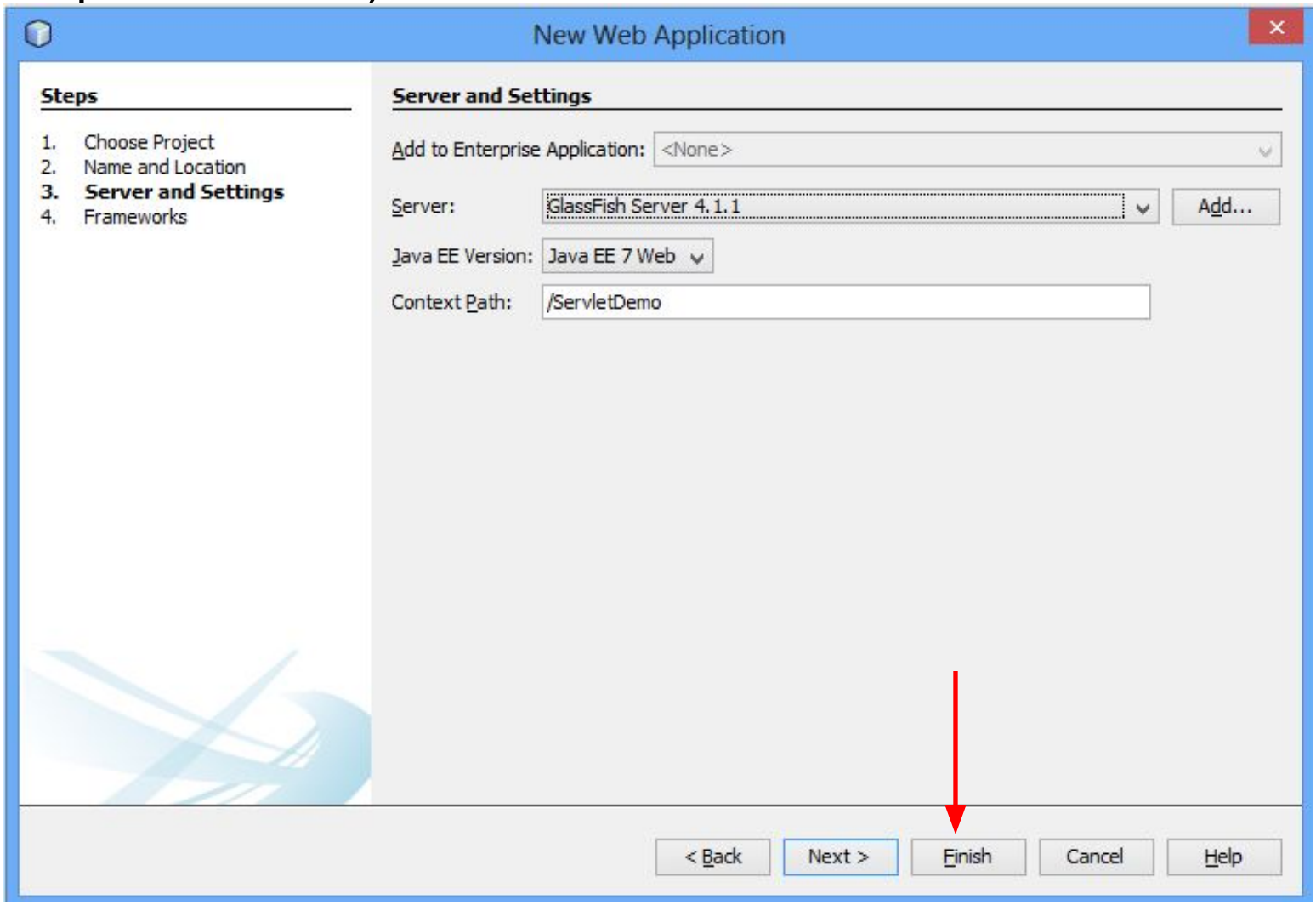
Steps for Servlet Program

- Step 3: Give a name to your project and click on Next,



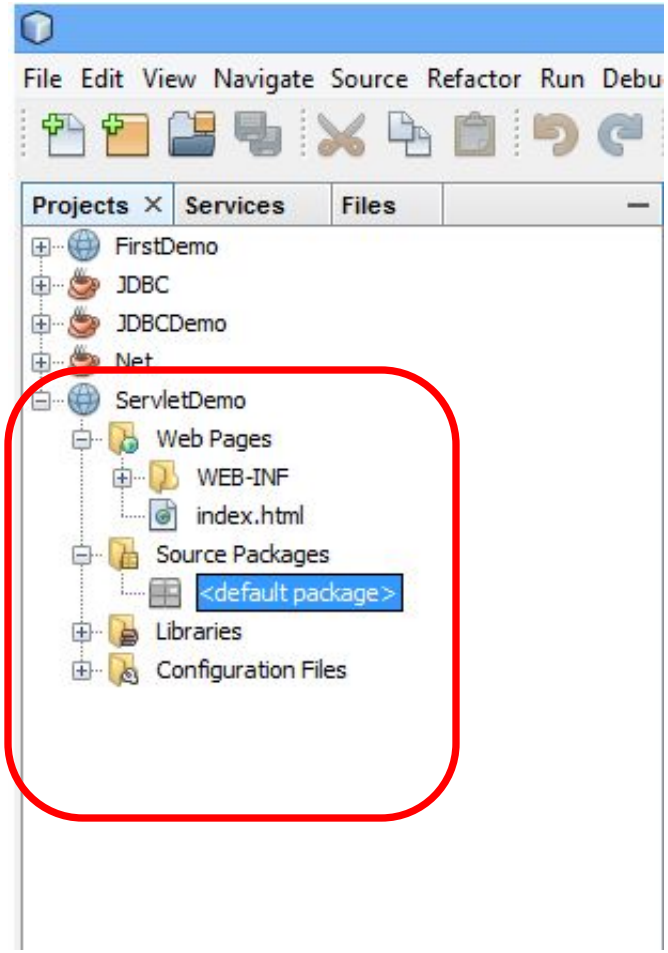
Steps for Servlet Program

- Step 4: and then, Click **Finish**



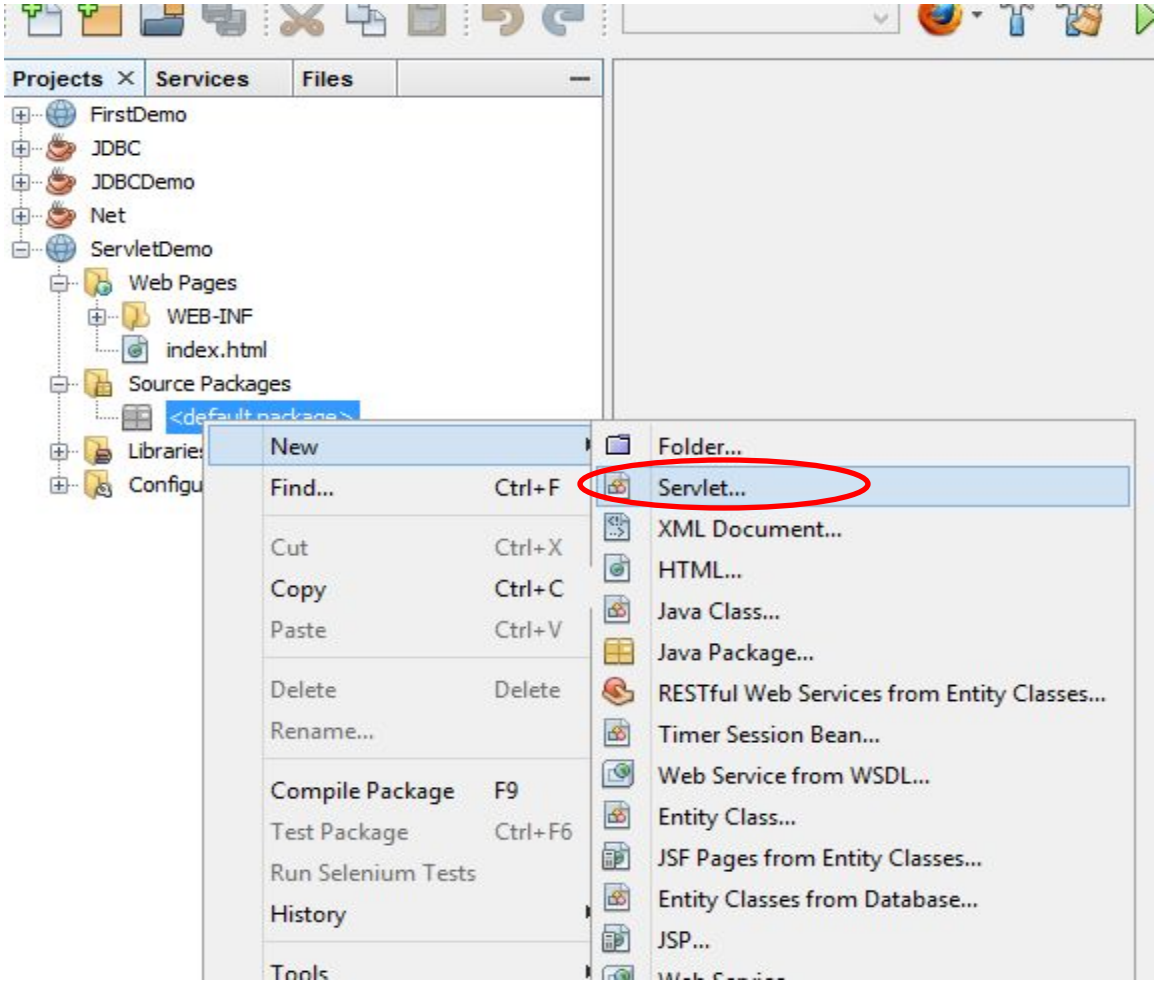
Steps for Servlet Program

- Step 5: The complete directory structure required for the Servlet Application will be created automatically by the IDE.



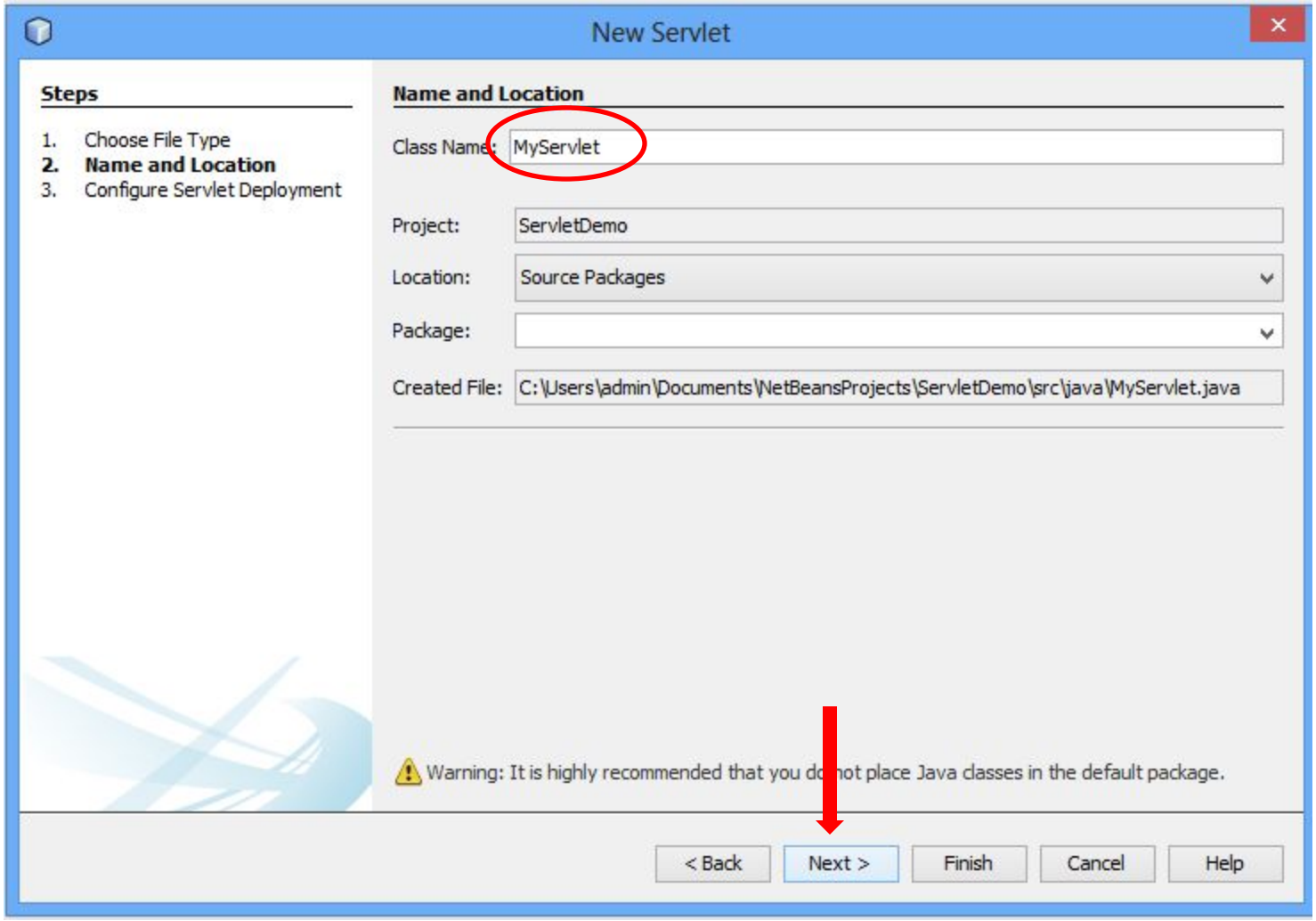
Steps for Servlet Program

Step 6: To create a Servlet, open **Source Package**, right click on **default packages** -> **New** -> **Servlet**.

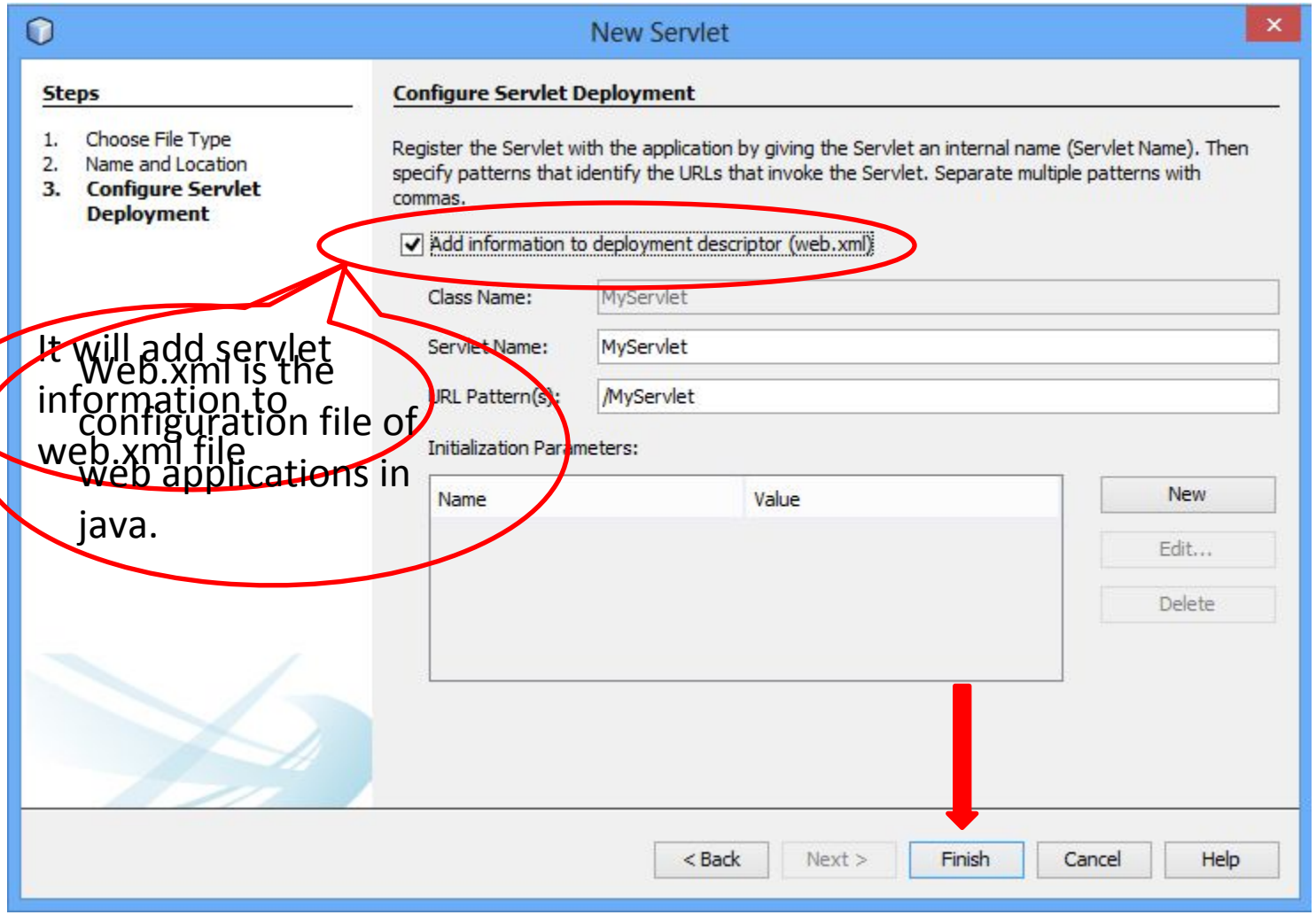


Steps for Servlet Program

- Step 7: Give a Name to your Servlet class file



Steps for Servlet Program



Step 8: Write servlet code: MyServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet1 extends HttpServlet
{    String msg="";
    PrintWriter out;
    public void init() throws ServletException
    {          msg="hello world: my first servlet program";    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
                          throws ServletException,IOException
    {
        response.setContentType("text/html");
        out=response.getWriter();
        out.println(msg);
    }
    public void destroy()
    {         out.close();
    }
}
```

MIME: Multipurpose Internet Mail Extensions

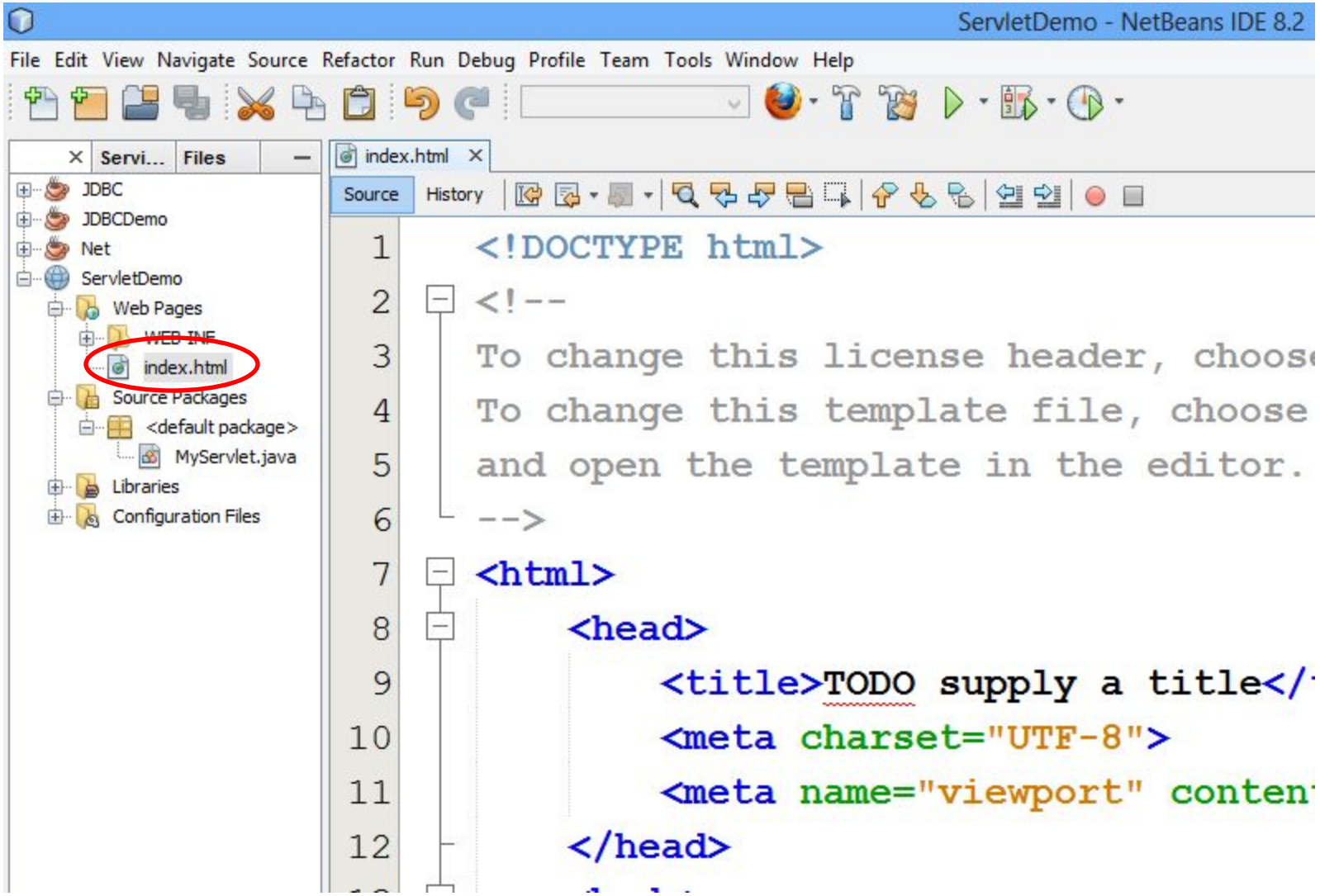
- A **MIME** type nomenclature includes a **type** and **subtype** separated by a forward slash.
- It is a **HTTP header** that provides the description about what are you sending to the browser.

1. text/html
2. **text/plain**
3. **text/css**
4. **text/richtext**
5. application/msword
6. application/jar
7. application/pdf
8. images/jpeg images/png images/gif
9. audio/mp3
10. video/mp4

MIME is a standard set to Internet to notify the format of the file contents.

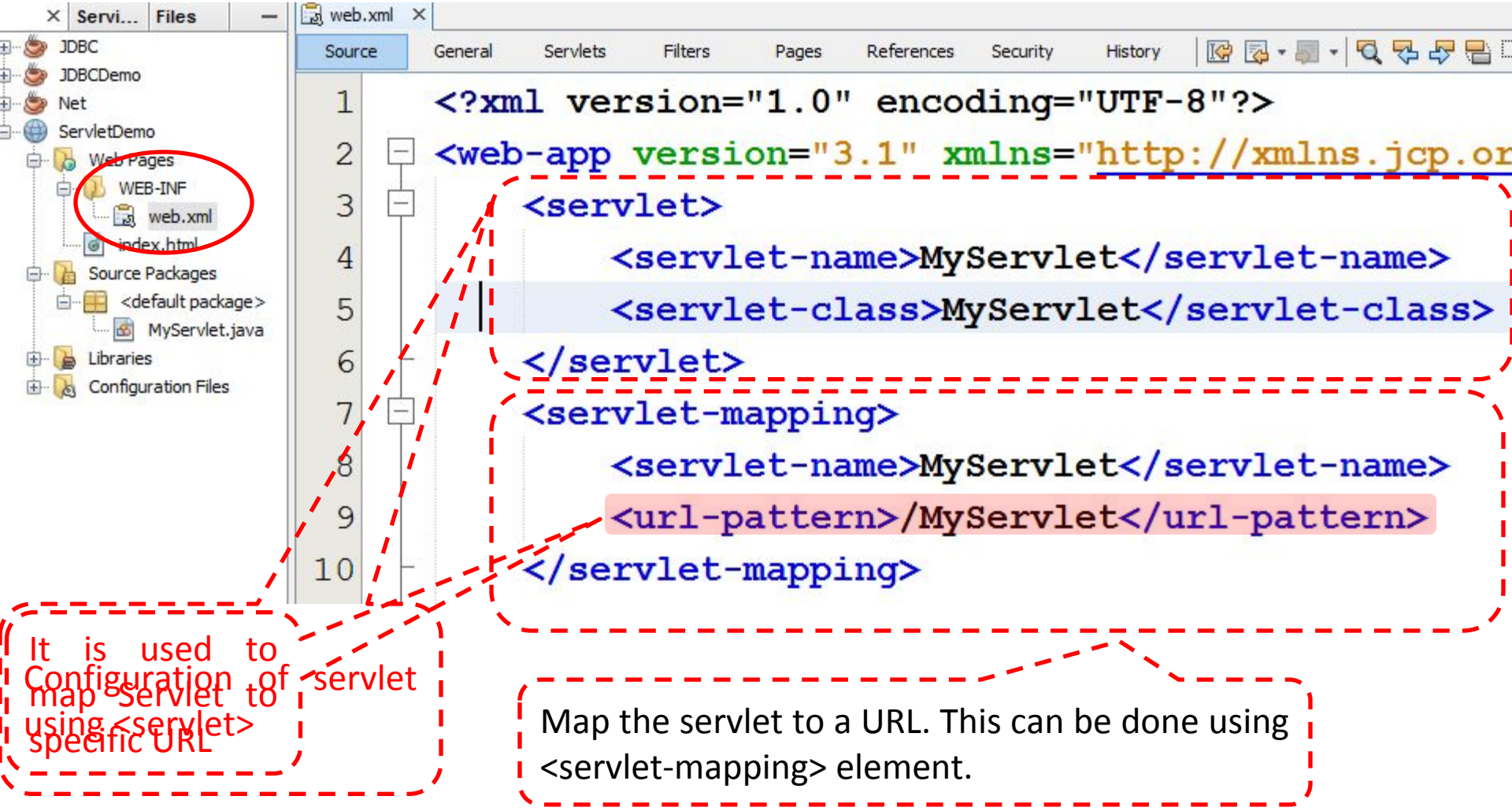
Steps for Servlet Program

Step 9: index.html



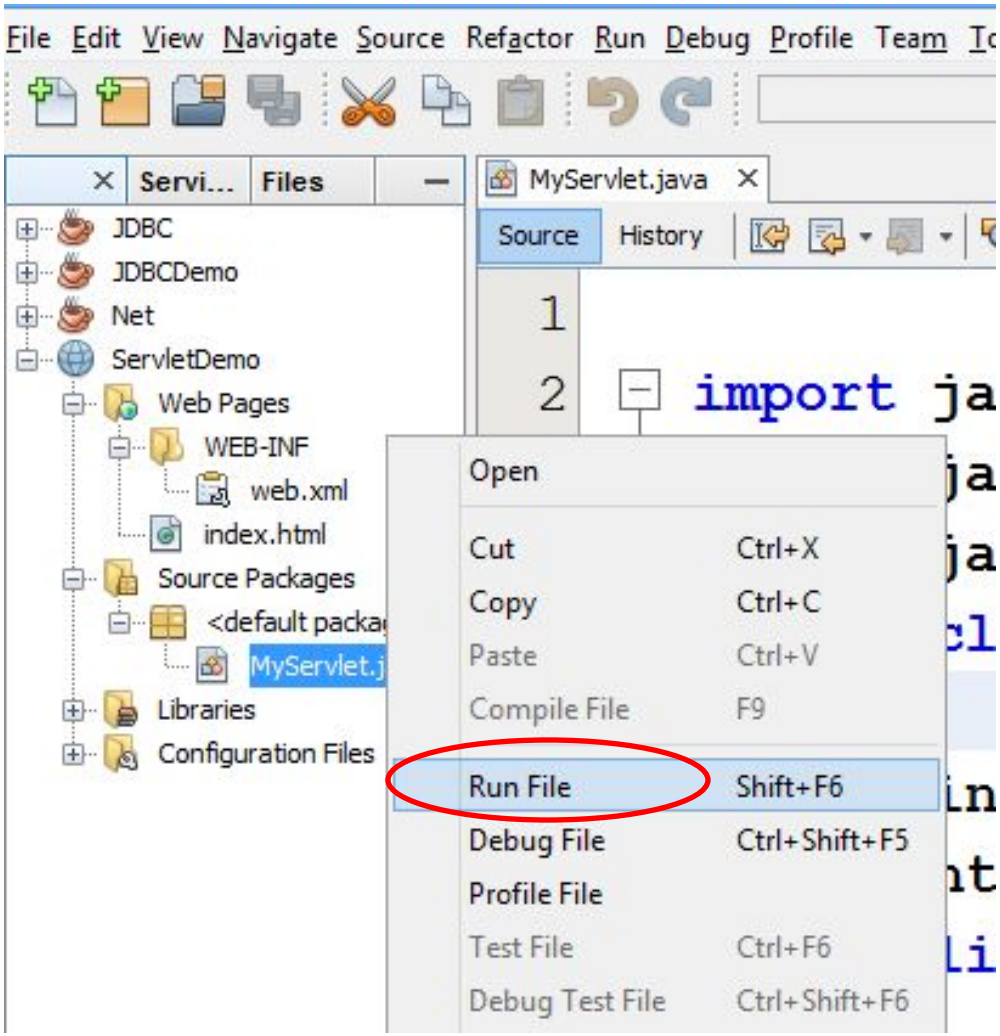
Steps for Servlet Program

Step 10: open web.xml



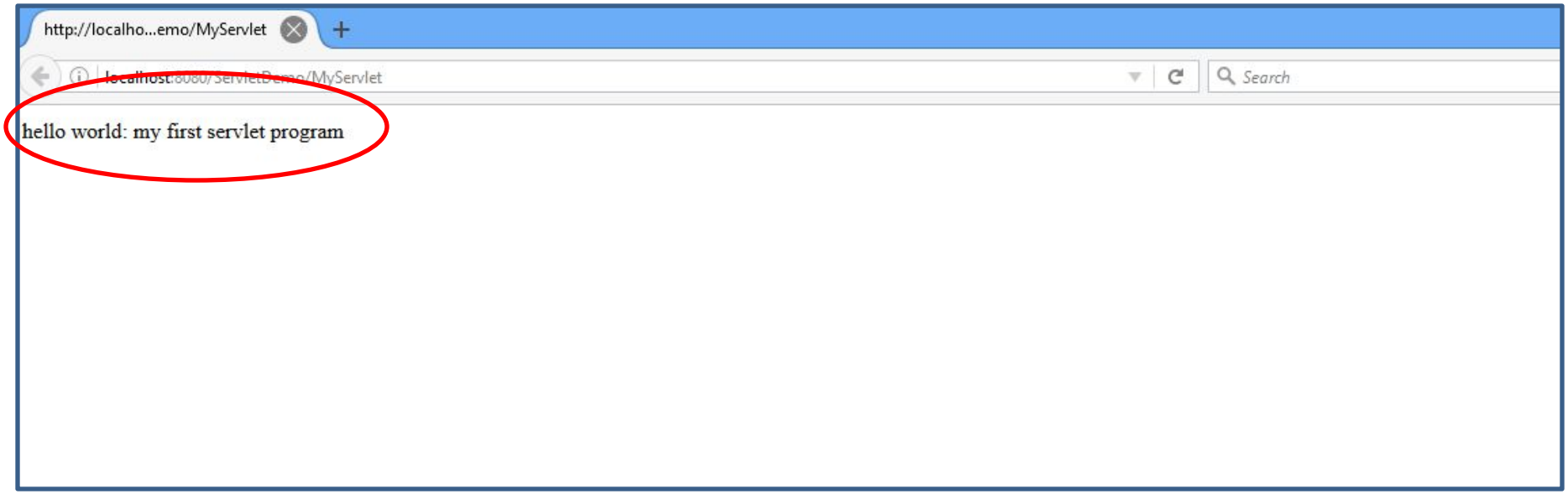
Steps for Servlet Program

Step 11: Run your application, right click on your Project and select **Run**



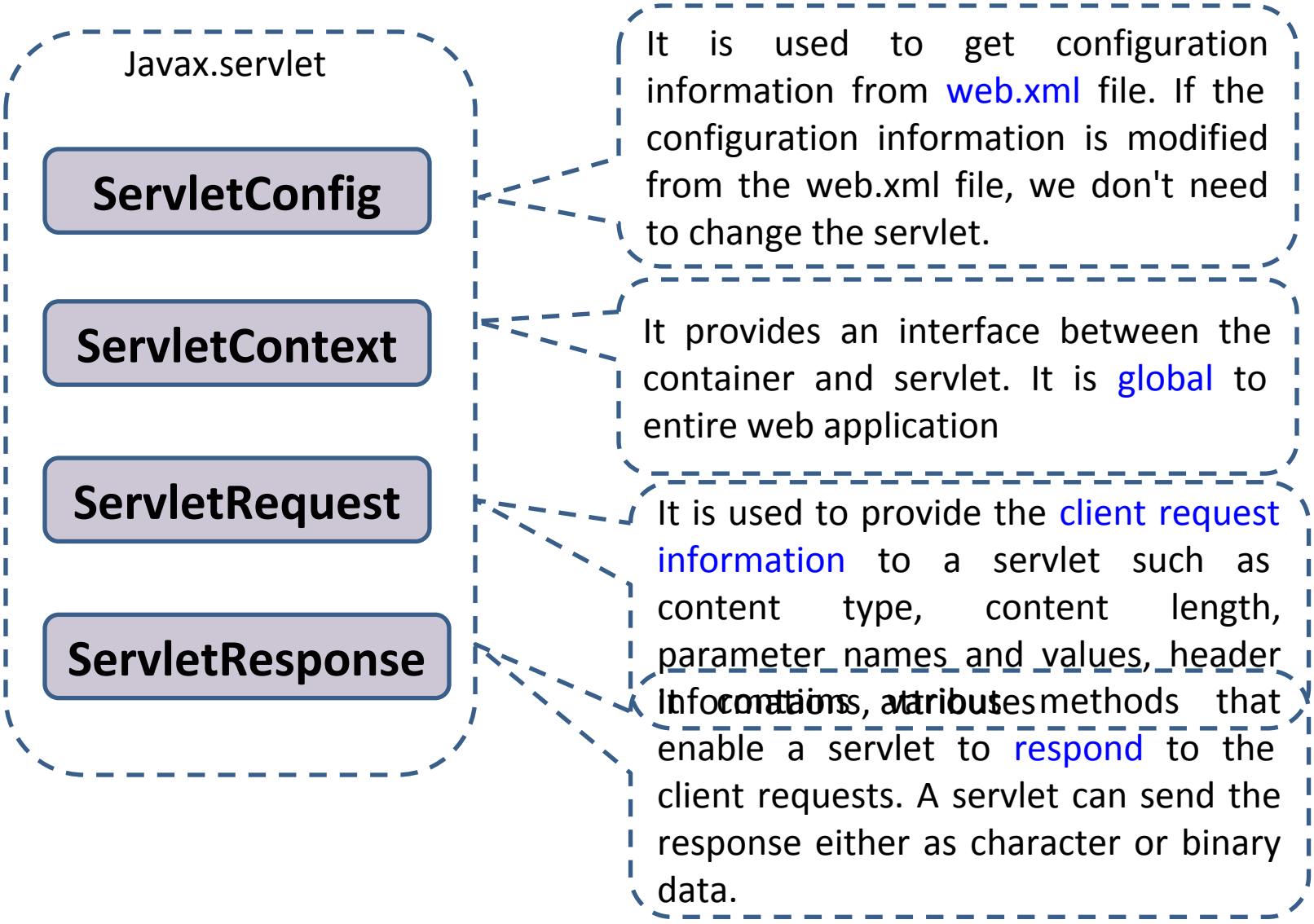
Steps for Servlet Program

Output:



javax.servlet Interface

javax.servlet Interface



Types of Servlet

- **Generic Servlet**
  - javax.servlet (package)
  - extends javax.servlet.Servlet
  - service method

```
service(ServletRequest req, ServletResponse res)
```

- **Http Servlet**
  - javax.servlet.http (package)
  - extends javax.servlet.HttpServlet
  - doGet(), doPost()

```
doGet(HttpServletRequest req,HttpServletResponse res)
doPost(HttpServletRequest req,HttpServletResponse res)
```

Generic Servlet: Method Summary

void **init**(ServletConfig config)	It is used to initialize the servlet. It is called once, automatically, by the network service each time it loads the servlet.
abstract void **service** (ServletRequest request, ServletResponse response)	It provides service for the incoming request. It is invoked at each time when user requests for a servlet.
void **destroy**()	It is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
ServletConfig **getServletConfig**()	returns the object of ServletConfig.
ServletContext **getServletContext**()	returns the object of ServletContext.
String **getInitParameter**(String name)	returns the parameter value for the given parameter name.
Enumeration **getInitParameterNames**()	returns all the parameters defined in the web.xml file.
String **getServletName**()	returns the name of the servlet object.

HttpServlet: Method Summary

protected void **service**(HttpServletRequest req, HttpServletResponse res)	It receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
protected void **doGet**(HttpServletRequest req, HttpServletResponse res)	handles the GET request. It is invoked by the web container.
protected void **doPost**(HttpServletRequest req, HttpServletResponse res)	handles the POST request. It is invoked by the web container.

GenericServlet vs HttpServlet

GenericServlet	HttpServlet
javax.servlet.GenericServlet	javax.servlet.http.HttpServlet
It defines a generic, protocol-independent servlet.	It defines a HTTP protocol specific servlet.
GenericServlet is a super class of HttpServlet class.	HttpServlet is a sub class of GenericServlet class.
Can handle all types of protocols	only HTTP specific protocols.
It supports only one abstract method:service()	It support doGet(), doPost() etc.

GTU Questions

1.	GenericServlet vs HttpServlet [3]	Sum'16 Win'18 Win'19

Deployment Descriptor

web.xml

Deployment Descriptor

- Located @ WEB-INF directory
- File known as web.xml
- It controls the behavior of Java Servlet
- What does it contain?
  - XML Header
  - DOCTYPE
  - Web-app element

    The Web-app element should contain a servlet element with 3 sub-element.

  1. <servlet-name>: name used to access java servlet
  2. <servlet-class>: class name of java servlet
  3. <init-param>: for initialization parameter

Deployment Descriptor: web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <servlet>
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>MyServlet</servlet-class>
        <init-param>
            <param-name>name</param-name>
            <param-value>cxcy</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyServlet</servlet-name>
        <url-pattern>/MyServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

xml header

Configures a web application.

Document Type Definition

Name used to access Java Servlet

Name of servlet .java class

Used to pass parameters to a servlet from the web.xml file.

map the servlet to a URL or URL pattern

Controls behavior of Servlet

Program to call servlet from html file

Servlet Program

Write a java Servlet program to call servlet from html hyperlink.

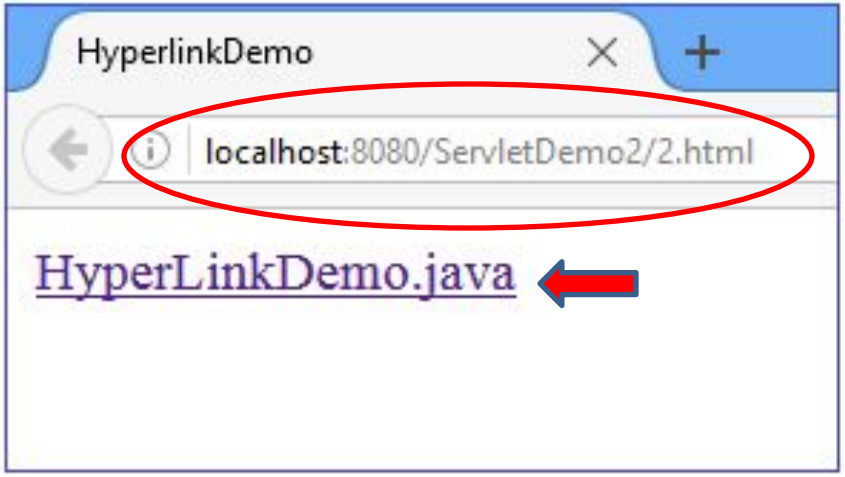
2.html

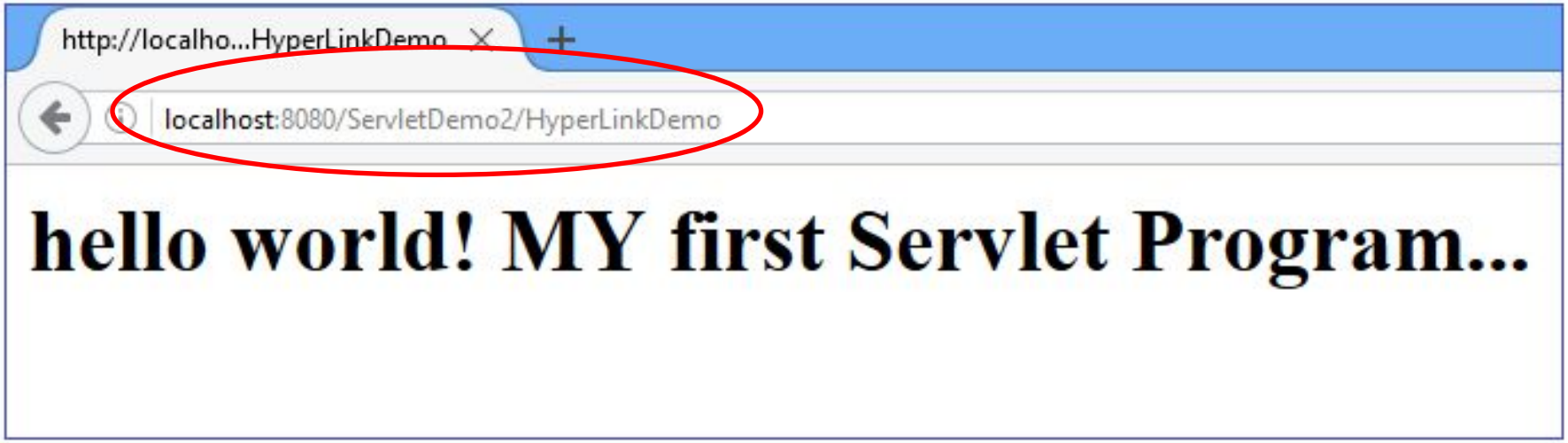
```
<html>
 <head>
  <title> HyperLinkDemo </title>
 </head>
 <body>
   <a href =
"/ServletDemo2/HyperLinkDemo">HyperLinkDemo.java
</a>
 </body>
</html>
```

Servlet Program: HyperLinkDemo.java

```
1.  import javax.servlet.*;
2.  import javax.servlet.http.*;
3.  import java.io.*;
4.  public class HyperLinkDemo extends HttpServlet
5.  {   String msg="";
6.      PrintWriter out;
7.      public void init(ServletConfig config)throws ServletException
8.      {   msg="hello world! MY first Servlet Program...";
9.      }
10.     public void doGet(HttpServletRequest request,HttpServletResponse
                  response) throws ServletException,IOException
11.     {   response.setContentType("text/html");
12.         out=response.getWriter();
13.         out.println("<h1>"+msg+"</h1>");
14.     }
15.     public void destroy()
16.     {   out.close();
17.     }}
```

Servlet Program: Output

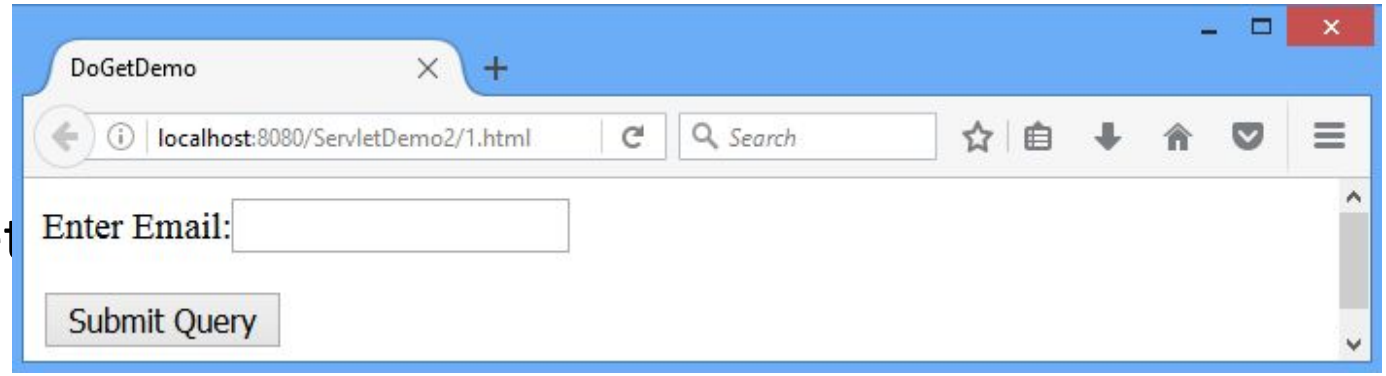




doGet()

HttpServlet : 1.html

```
<html>
   <head>
      <title> DoGet
</head>
   <body>
      <form action="/ServletDemo2/DoGetDemo">
         Enter Email:<input type="text" name="email">
         <p><input type="submit"></p>
      </form>
   </body>
</html>
```

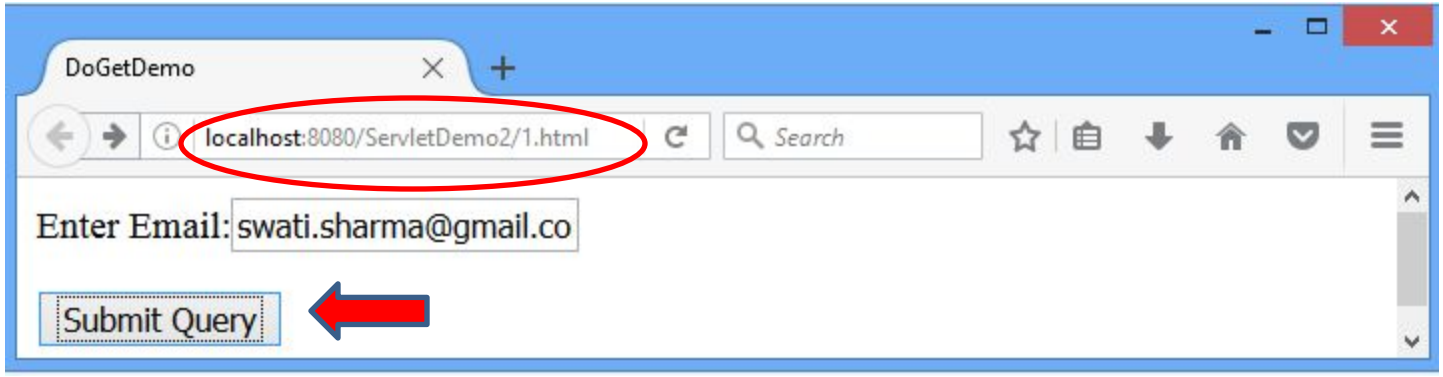


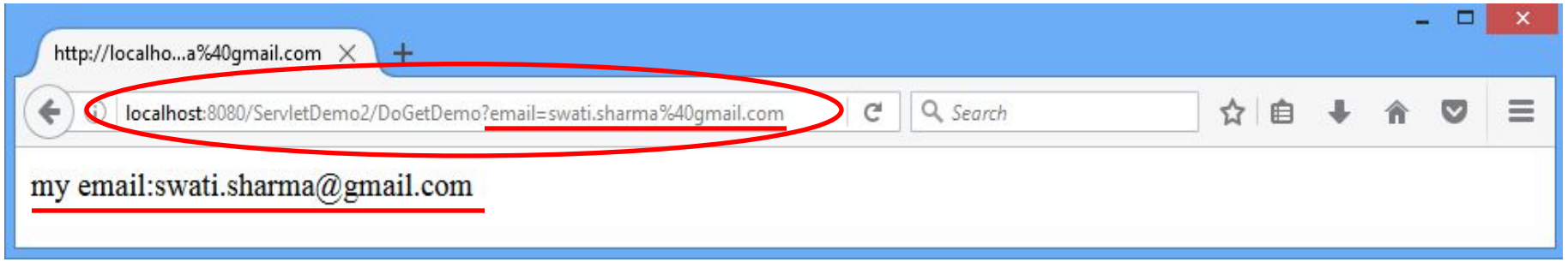
HttpServlet: DoGetDemo.java

```
1.  import javax.servlet.*;
2.  import javax.servlet.http.*;
3.  import java.io.*;
4.  public class DoGetDemo extends HttpServlet
5.  {     PrintWriter out;
6.      public void init(ServletConfig config)throws ServletException
7.      {     }
8.      public void doGet(HttpServletRequest request,HttpServletResponse
                                response)
9.              throws ServletException,IOException
10.     {
11.         String email=request.getParameter("email");
12.         response.setContentType("text/html");
13.         out =response.getWriter();
14.         out.println("my email:"+email);
15.     }
16.     public void destroy()
17.     {         out.close();     }
18.   }
```

String **getParameter**(String name)
Returns the value of a request parameter as a String

Output

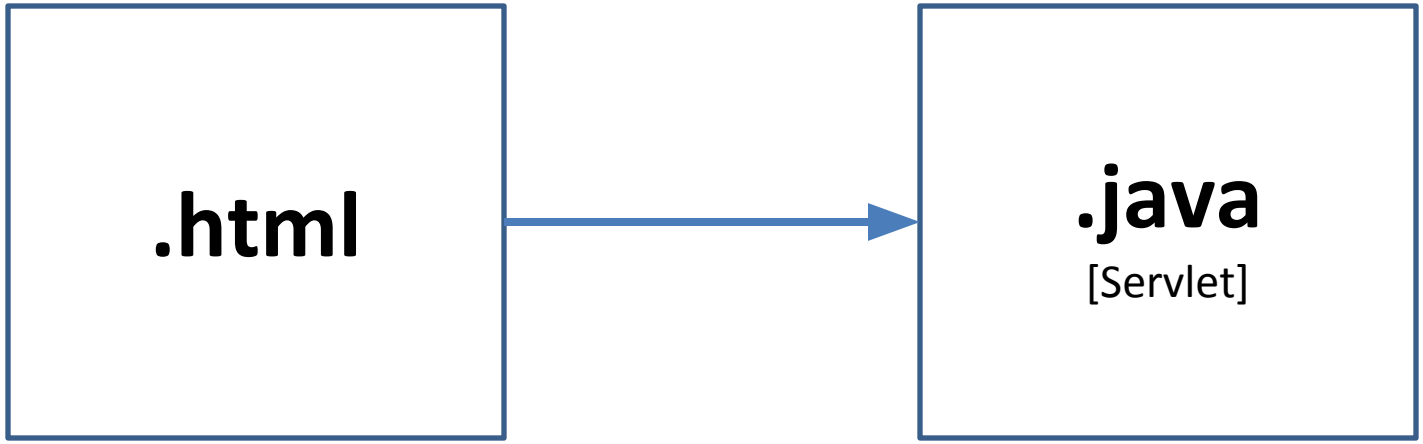

DoGetDemo
localhost:8080/ServletDemo2/1.html
Search
Enter Email: swati.sharma@gmail.co
Submit Query


http://localho...a%40gmail.com
localhost:8080/ServletDemo2/DoGetDemo?email=swati.sharma%40gmail.com
Search
my email:swati.sharma@gmail.com

doPost()

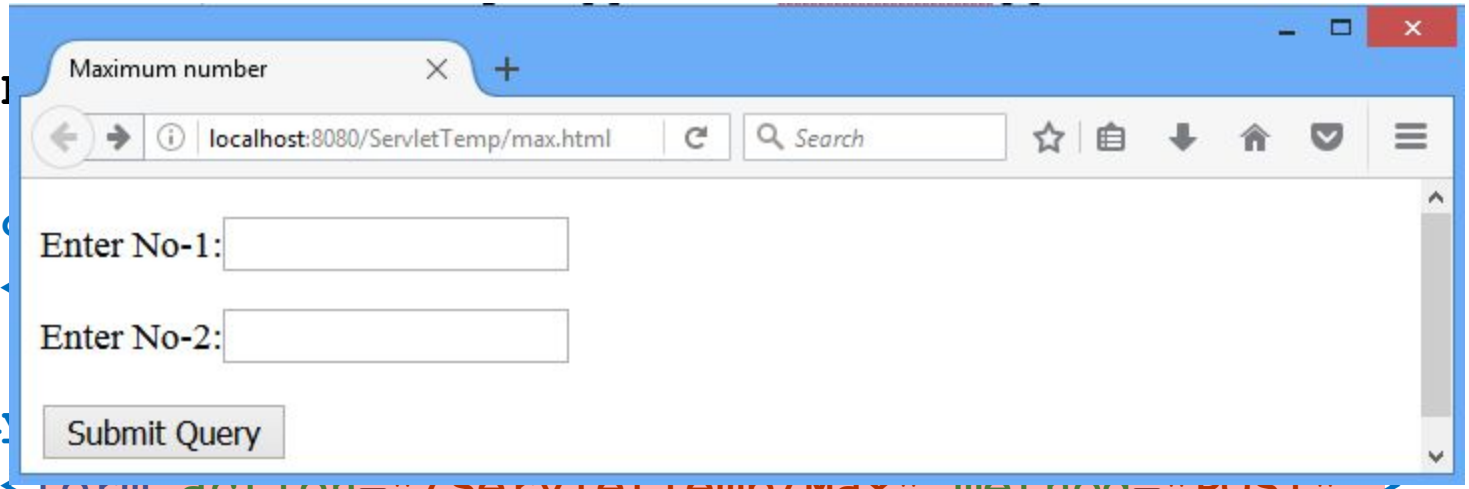
Write a Servlet program to enter two numbers and find maximum among them.

Servlet program: doPost()



Servlet program using doPost()

max.html



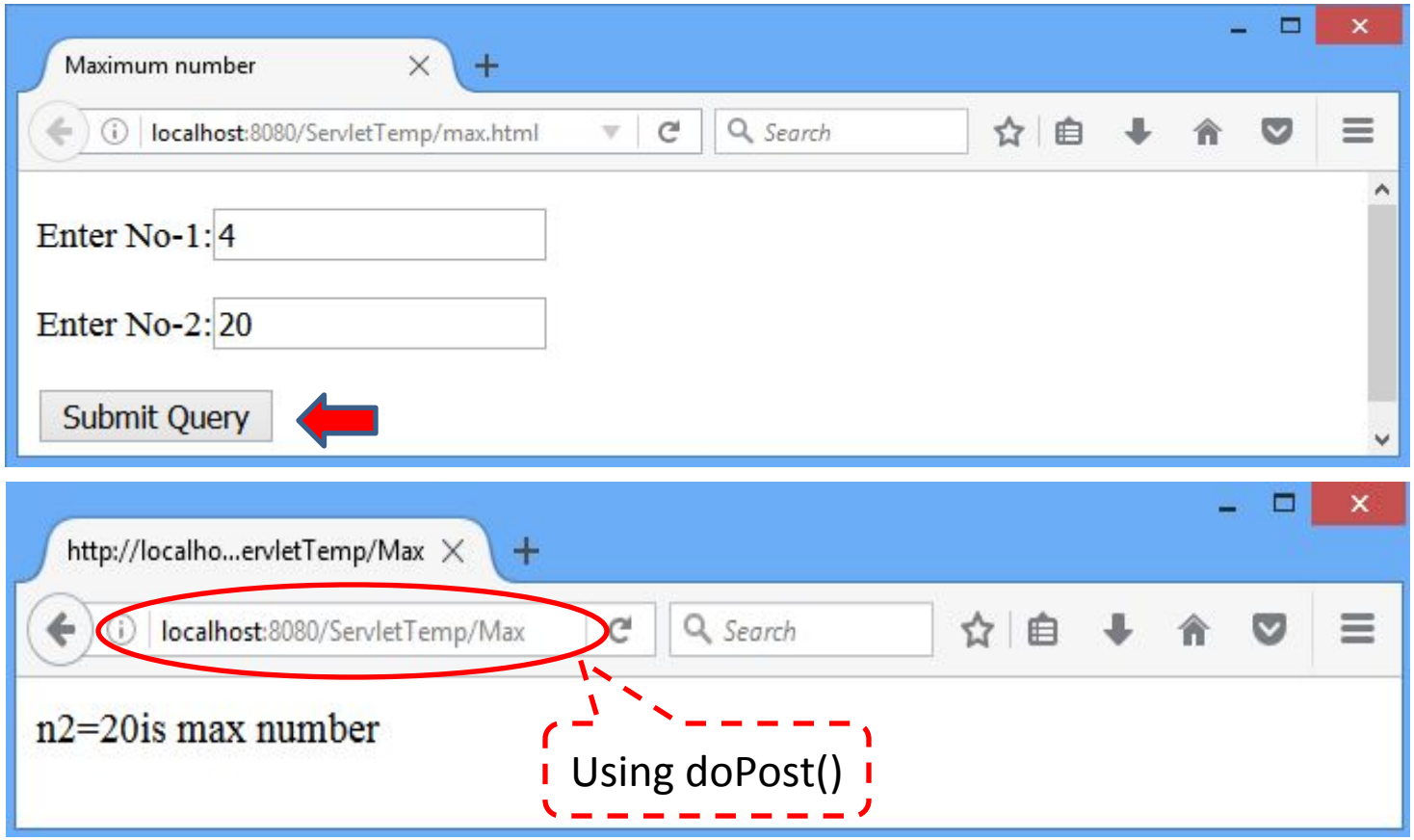

```
1.  <!DOCTYPE
2.  <html>
3.      <head
4.          <
5.  </head>
6.      <body
7.          <form action="/ServletTemp/Max" method="POST" >
8.              <p>Enter No-1:<input type="text"
                    name="no1"></p>
9.              <p>Enter No-2:<input type="text"
                    name="no2"></p>
10.             <p><input type="submit"></p>
11.         </form>
12.     </body>
13. </html>
```

Servlet program using doPost()

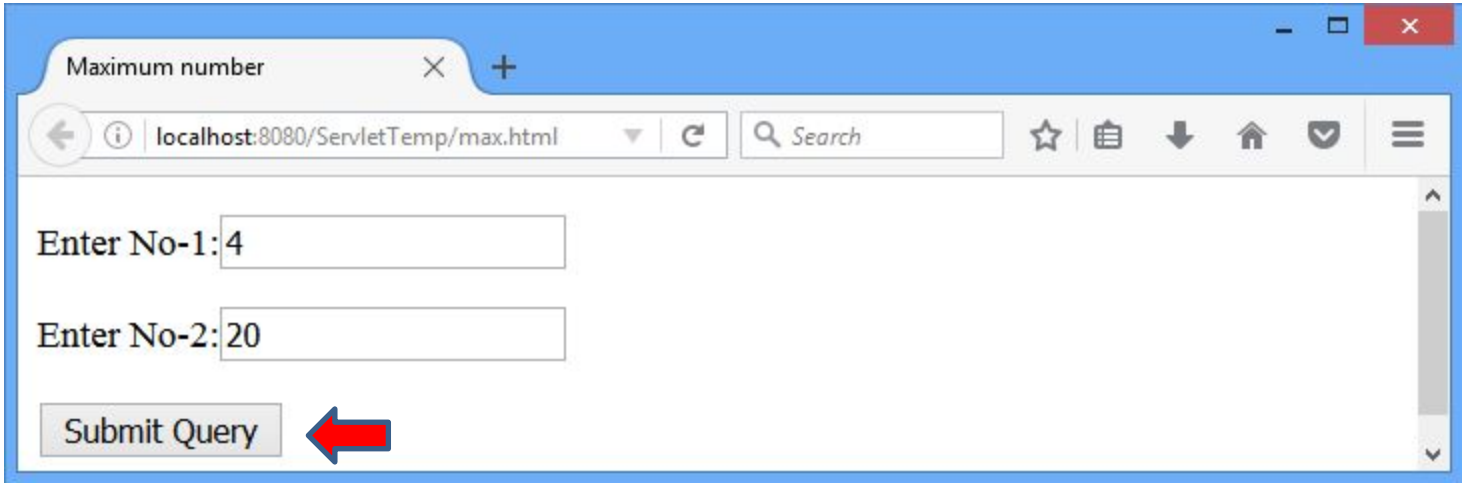
```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.  public class Max extends HttpServlet
5.  {   public void doPost(HttpServletRequest request, HttpServletResponse
                          response)throws ServletException,IOException
6.      {   int n1=0,n2=0;
7.          response.setContentType("text/html");
8.          PrintWriter out=response.getWriter();
9.          n1=Integer.parseInt(request.getParameter("no1"));
10.         n2=Integer.parseInt(request.getParameter("no2"));
11.         if(n1>n2)
12.             out.println("n1="+n1+"is max number");
13.         else if(n2>n1)
14.             out.println("n2="+n2+"is max number");
15.         else if(n1==n2)
16.             out.println("n1= "+n1+"and n2=  "+n2+"are equal numbers");
17.     }
18. }
```

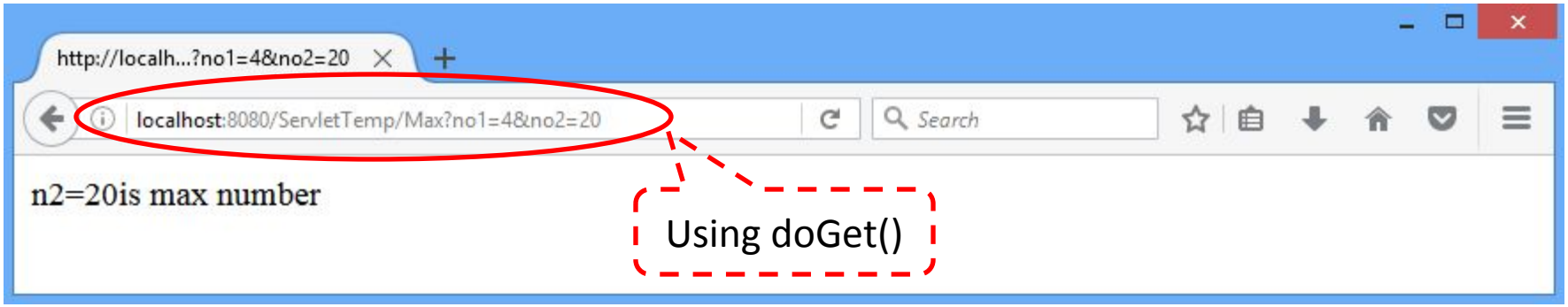
Servlet program using doPost()

Executing max.html



Servlet program using doGet()





ServletConfig Interface

Servlet Config

- It is used to get configuration information from web.xml file.
- If the configuration information is modified from the web.xml file, we don't need to change the servlet.

Method

String getInitParameter(String name)	Returns the parameter value for the specified parameter name.

Example

```
String str = config.getInitParameter("name")
```

web.xml
<init-param>
<param-name>**name**</param-name>

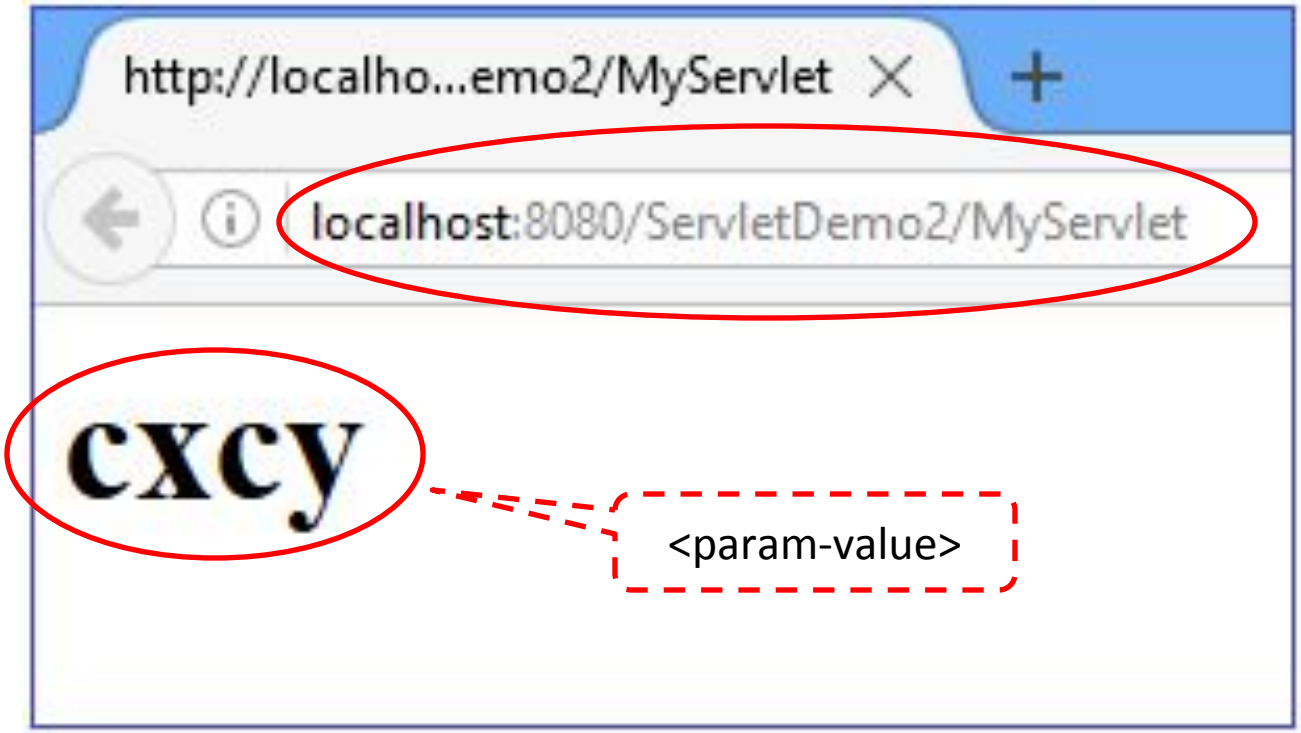
Servlet Config: web.xml

```
<web-app>
    <servlet>
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>MyServlet</servlet-class>
        <init-param>
            <param-name>name</param-name>
            <param-value>cxcy</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyServlet</servlet-name>
        <url-pattern>/MyServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Servlet Config: MyServlet.java

```
1.  import javax.servlet.*;
2.  import javax.servlet.http.*;
3.  import java.io.*;
4.  public class MyServlet extends HttpServlet
5.  {   String msg;
6.      PrintWriter out;
7.      public void init(ServletConfig config)throws
                    ServletException
8.      {       msg = config.getInitParameter("name"); }
9.      public void doGet(HttpServletRequest request ,
           HttpServletResponse response) throws
10.                 ServletException,IOException
11.     {   response.setContentType("text/html");
12.         out = response.getWriter();
13.         out.println("<h1>"+ msg +"</h1>");
14.     }
15.     public void destroy()
16.     {       out.close();    }}
```

Servlet Config: Output



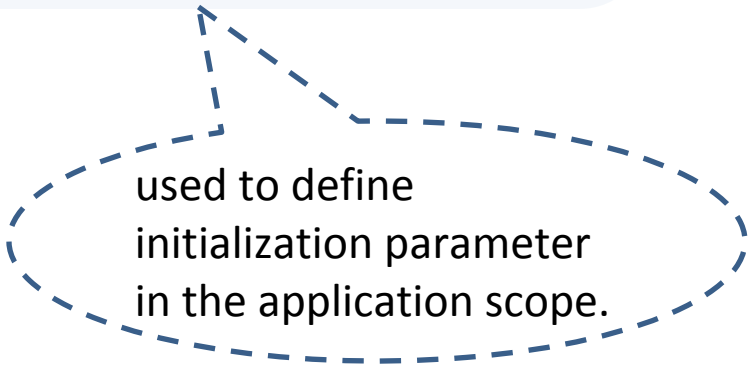
ServletContext Interface

ServletContext Interface

- ServletContext is created by the web container at time of deploying the project.
- It can be used to get configuration information from web.xml file.
- There is only one ServletContext object per web application.
- If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.

Context Parameter Initialized inside web.xml

```
<web-app>
  ...
  <context-param>
    <param-name>parametername</param-name>
    <param-value>parametervalue</param-value>
  </context-param>
  ...
 <servlet>
      ...
  </servlet>
</web-app>
```



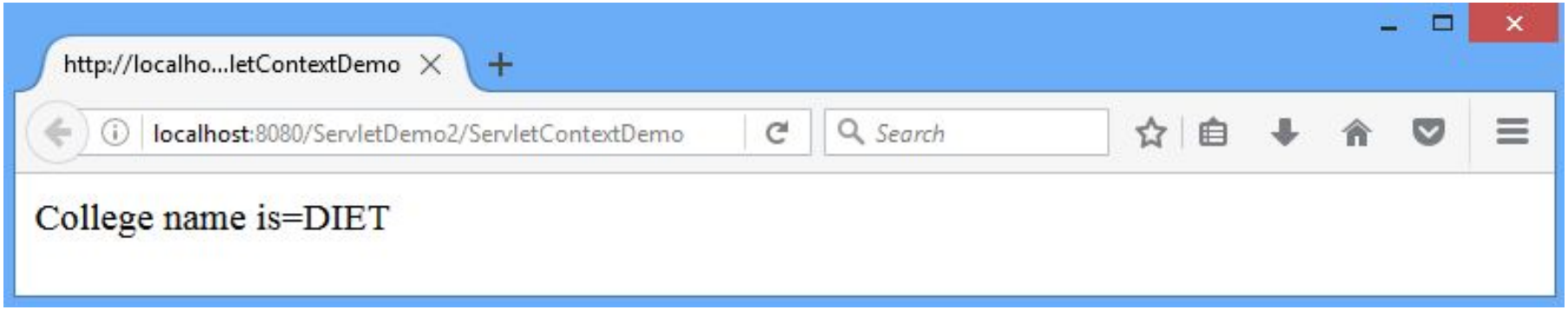
web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>ServletContextDemo</servlet-name>
        <servlet-class>ServletContextDemo</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ServletContextDemo</servlet-name>
        <url-pattern>/ServletContextDemo</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>name</param-name>
        <param-value>DIET</param-value>
    </context-param>
</web-app>
```

ServletContextDemo.java

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.  public class ServletContextDemo extends HttpServlet{
5.      public void doGet(HttpServletRequest req,HttpServletResponse
                  res)  throws ServletException,IOException
6.  {   res.setContentType("text/html");
7.      PrintWriter out=res.getWriter();
8.      //creating ServletContext object
9.      ServletContext context=getServletContext();
10.     //Getting the value of the initialization parameter and
    printing it
11.     String college=context.getInitParameter("name");
12.     out.println("College name is="+college);
13.     out.close();
14.   }}
```

Output


http://localho...letContextDemo
localhost:8080/ServletDemo2/ServletContextDemo
Search
College name is=DIET

Servlet Config vs Servlet Context

Servlet Config	Servlet Context
ServletConfig object is one per servlet class	ServletContext object is global to entire web application
Object of ServletConfig will be created during initialization process of the servlet	Object of ServletContext will be created at the time of web application deployment
Scope**:** As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed.	***Scope*****:** As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server.
We should give request explicitly, in order to create ServletConfig object for the first time	ServletContext object will be available even before giving the first request
In web.xml – <*init-param*> tag will be appear under <*servlet-class*> tag	In web.xml – <*context-param*> tag will be appear under <*web-app*> tag

Servlet Program

- Write a java program to accept one String from ServletConfig object and another from ServletContext object and Concat both the String. Display the String and String Length.

HttpServletRequest

Methods

HttpServletRequest: Methods

String **getContextPath**()	Returns the portion of the request URI that indicates the context of the request.
Enumeration **getHeaderNames**()	Returns an enumeration of all the header names this request contains.
String **getHeader**(String name)	Returns the value of the specified request header as a String.
String **getQueryString**()	Returns the query string that is contained in the request URL after the path.
String **getServletPath**()	Returns the part of this request's URL that calls the servlet. This path starts with a "/" character and includes either the servlet name or a path to the servlet
String **getMethod**()	Returns the name of the HTTP method with which this request was made, for example GET or POST

HttpServletRequest: Methods

String **getContextPath**()	Returns the portion of the request URI that indicates the context of the request.

Example

```
public void doGet(HttpServletRequest request,
            HttpServletResponse response)
{
    out.println("<p>request.getContextPath():"
+request.getContextPath()+"</p>");
}
```

Output

```
    request.getContextPath():/ServletTemp
```

HttpServletRequest: Methods

Enumeration **getHeaderNames**()	Returns an enumeration of all the header names this request contains.

Example

```
public void doGet(HttpServletRequest request,
            HttpServletResponse response)
{
     Enumeration h=request.getHeaderNames();
        while(h.hasMoreElements())
    {
        String paramName = (String)h.nextElement();
        out.print("<p>" + paramName + "\t");
        String paramValue = request.getHeader(paramName);
        out.println( paramValue + "</p>\n");
    }
}
```

Output

1. host localhost:8080
2. user-agent Mozilla/5.0 (Windows NT 6.2; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0
3. accept text/html,application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
4. accept-language en-US,en;q=0.5
5. accept-encoding gzip, deflate
6. connection keep-alive
7. upgrade-insecure-requests 1

HttpServletRequest: Methods

String **getHeader**(String name)	Returns the value of the specified request header as a String.

Example

```
public void doGet(HttpServletRequest request,
            HttpServletResponse response)
{
   out.println("<p>request.getHeader(): "
    +request.getHeader("host")+"</p>");
      out.println("<p>request.getHeader(): "
      +request.getHeader("referer")+"</p>");
}
```

Output

```
request.getHeader():host=localhost:8080
request.getHeader():referer=http://localhost:8080
                  /ServletTemp/servletmeth.html
```

HttpServletRequest: Methods

String **getQueryString**()	Returns the query string that is contained in the request URL after the path.

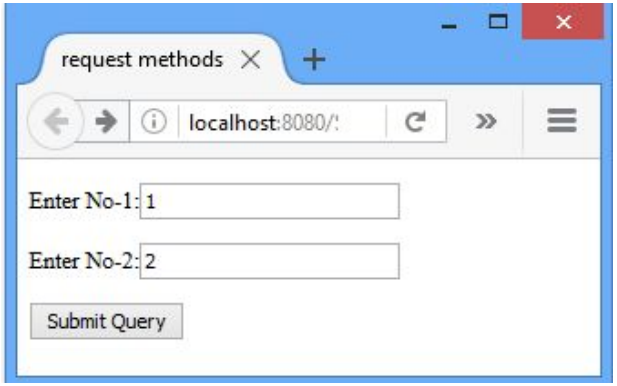
Example

```
public void doGet(HttpServletRequest request,
            HttpServletResponse response)
{
out.println("<p>request.getQueryString():"
       +request.getQueryString()+"</p>");

}
```

Output

```
request.getQueryString(): no1=1&no2=2
```



HttpServletRequest: Methods

String **getServletPath**()	Returns the part of this request's URL that calls the servlet. This path starts with a "/" character and includes either the servlet name or a path to the servlet

Example

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
{
    out.println("<p>request.getServletPath():"
        +request.getServletPath()+"</p>");
}
```

Output

```
request.getServletPath(): /ServletMeth
```

HttpServletRequest: Methods

String **getMethod**()	Returns the name of the HTTP method with which this request was made, for example GET or POST

Example

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
{
   out.println("<p>request.getMethod():"
      +request.getMethod()+"</p>");
}
```

Output

```
   request.getMethod(): GET
```

Servlet Collaboration

RequestDispatcher Interface

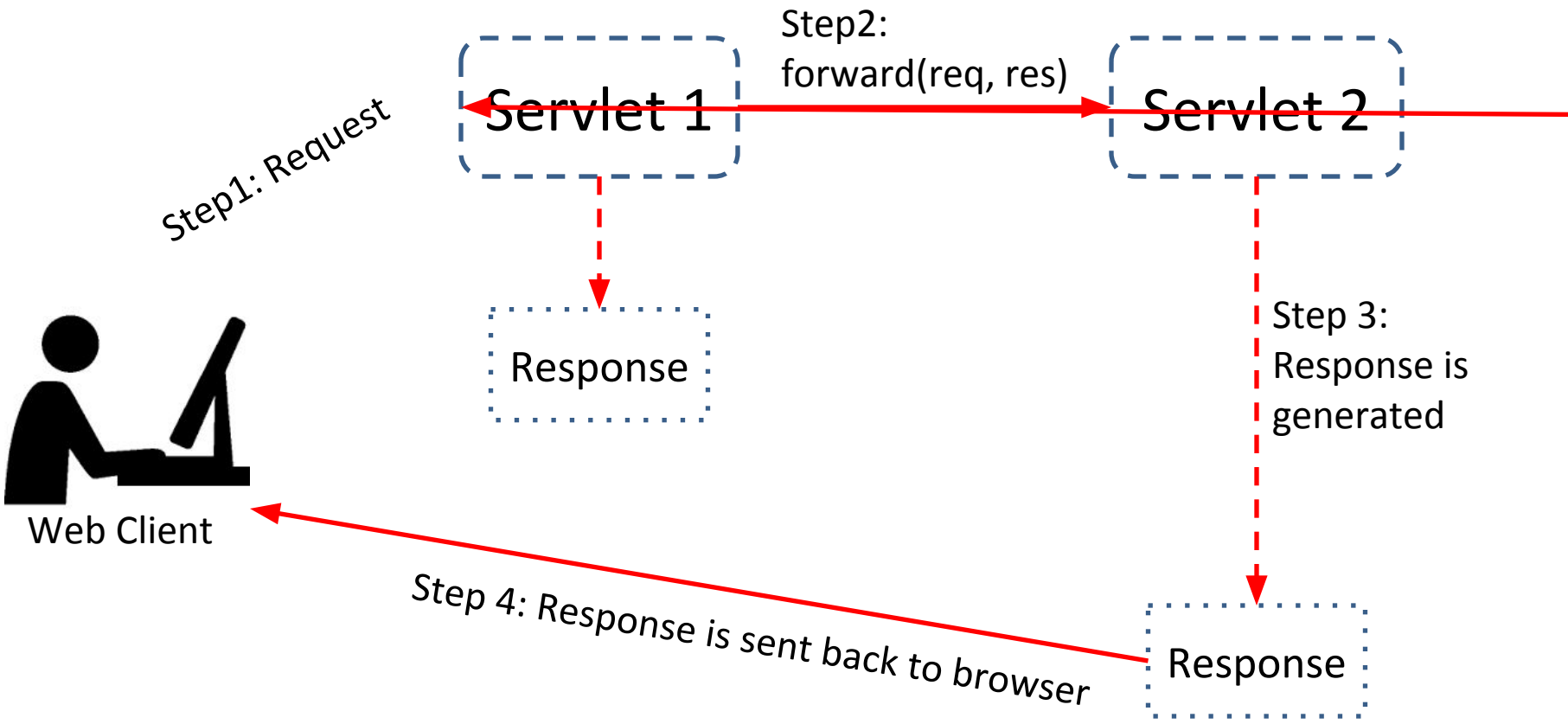
javax.servlet.RequestDispatcher Interface

- The RequestDispatcher interface provides the facility of dispatching the request to another resource.
- Resource can be HTML, Servlet or JSP.
- This interface can also be used to include the content of another resource.
- It is one of the way of servlet collaboration.

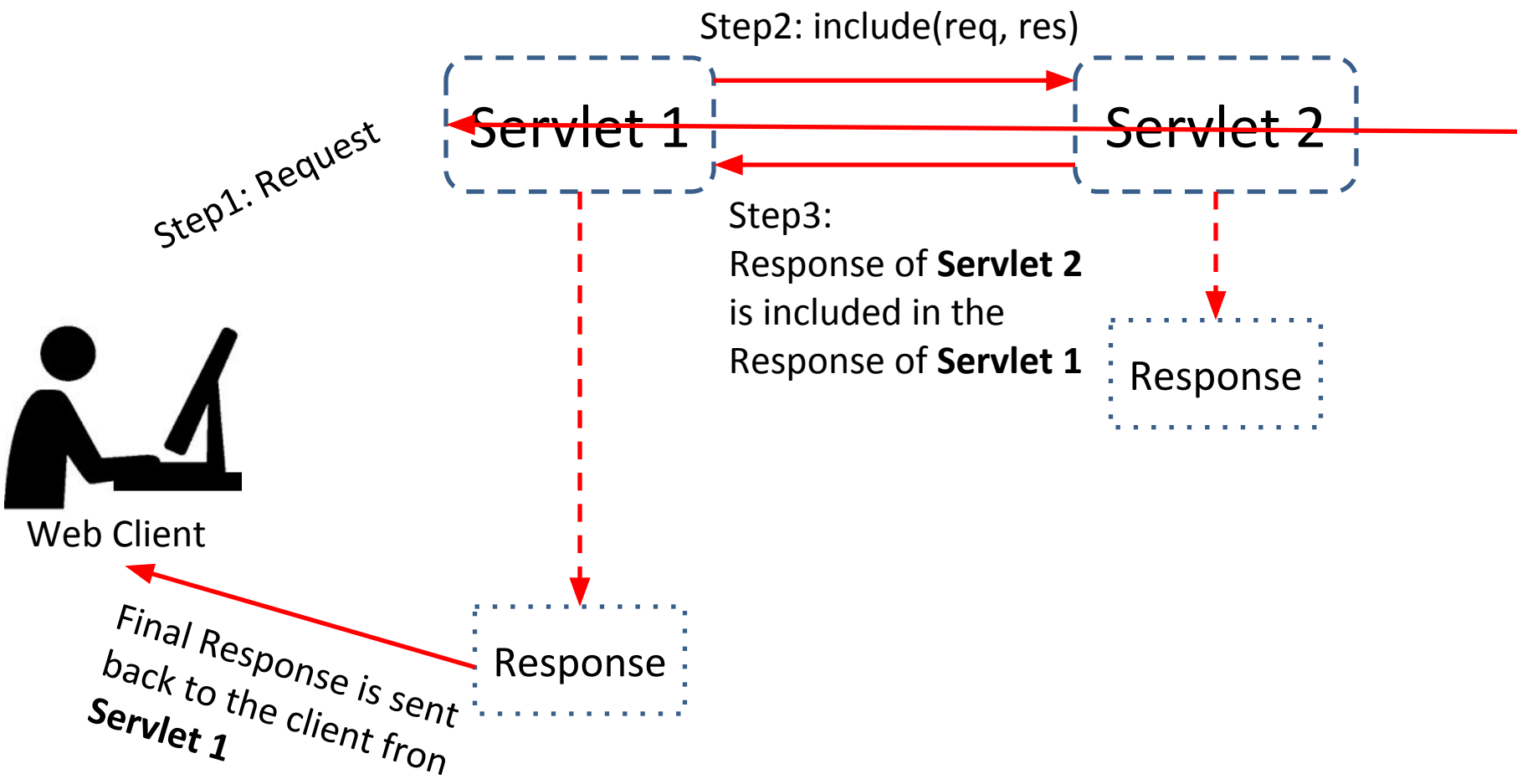
RequestDispatcher :Method

void **forward**(ServletRequest request, ServletResponse response) throws ServletException, IOException	Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
void **include**(ServletRequest request, ServletResponse response) throws ServletException, IOException	Includes the content of a resource (Servlet, JSP page, or HTML file) in the response.

RequestDispatcher: forward()



RequestDispatcher: include()



How to get the object of RequestDispatcher?

The **getRequestDispatcher**() method of ServletRequest interface returns the object of RequestDispatcher.

Syntax

```
RequestDispatcher getRequestDispatcher(String resource)
```

Example

Name of Servlet specified in <url-pattern>

```
RequestDispatcher rd=request.getRequestDispatcher("servlet2");
rd.forward(request, response);//method may be include/forward
```

RequestDispatcher: forward()

Example: forward()

```
RequestDispatcher rd =
request.getRequestDispatcher("servlet2");
rd.forward(request, response);


RequestDispatcher rd   =
request.getRequestDispatcher("/1.html");
rd.forward(request, response);
```

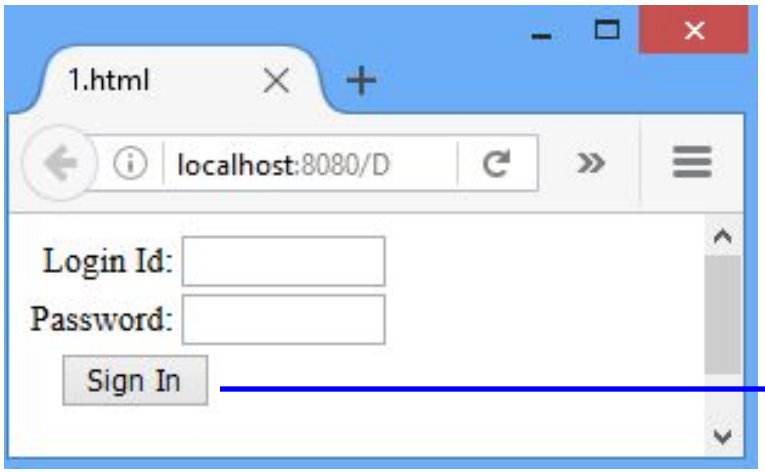
RequestDispatcher: include()

Example: include()

```
RequestDispatcher rd=
request.getRequestDispatcher("servlet2");
rd.include(request, response);
```

```
RequestDispatcher rd=
request.getRequestDispatcher("/1.html");
rd.include(request, response);
```

RequestDispatcher: Servlet Program



Validate Servlet
[CallServlet.java]

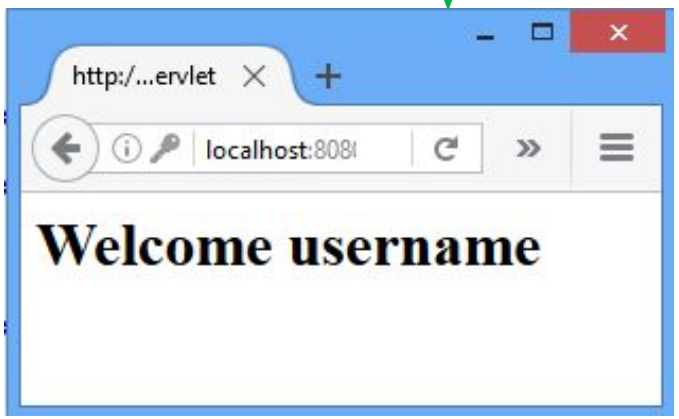
IsValid
?

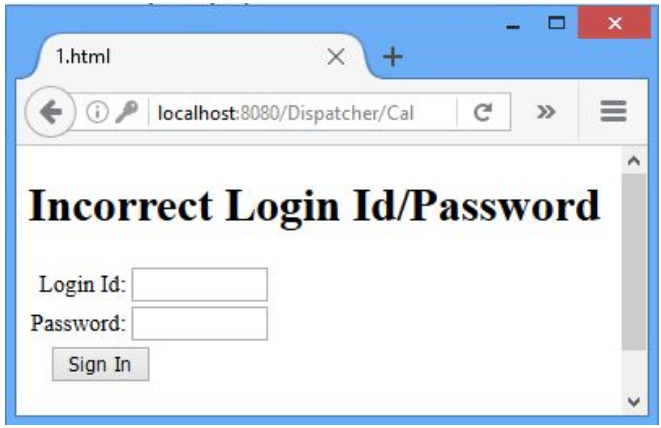
Ye
s

No

[forward: FwdDemo.java]

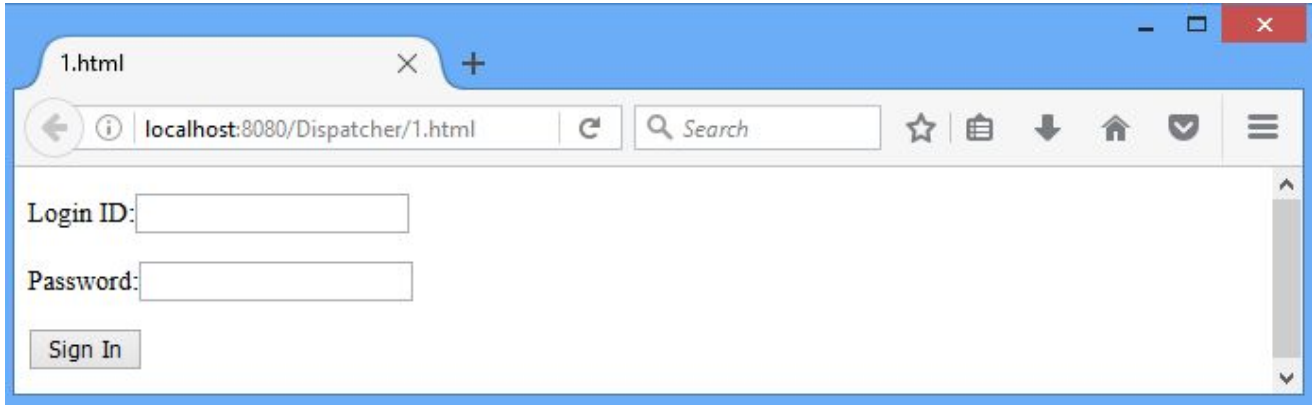
[include: 1.html]





RequestDispatcher: 1.html

```
1.  <html>
2.      <head>
3.          <title>1.html</title>
4.      </head>
5.      <body>
6.          <form action="/Dispatcher/CallServlet"
                        method="POST">
7.              <p>Login ID:<input type="text" name="login"></p>
8.              <p>Password:<input type="text" name="pwd"></p>
9.              <p><input type="submit" value="Sign In"></p>
10.         </form>
11.     </body>
12. </html>
```

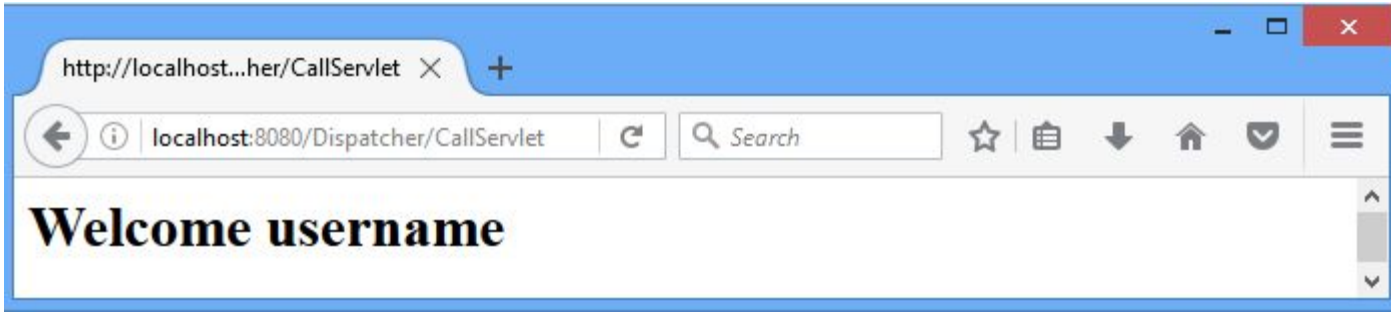


RequestDispatcher: Validate Servlet

```
1.  public class CallServlet extends HttpServlet
2.  {   public void doPost(HttpServletRequest request,
            HttpServletResponse response)
3.  throws ServletException,IOException
4.      {   response.setContentType("text/html");
5.          PrintWriter out=response.getWriter();
6.          RequestDispatcher rd;
7.          String login=request.getParameter("login");
8.          String pwd=request.getParameter("pwd");
9.          if(login.equals("java") && pwd.equals("servlet"))
10.         {   rd=request.getRequestDispatcher("FwdDemo");
11.             rd.forward(request, response);}//if
12.         else
13.         {   out.println("<p><h1>Incorrect Login Id/Password
                            </h1></p>");
14.             rd=request.getRequestDispatcher("/1.html");
15.             rd.include(request, response); }//else }//dopost }
```

RequestDispatcher: fwdDemo.java

```
1.  import javax.servlet.*;
2.  import javax.servlet.http.*;
3.  import java.io.*;
4.  public class FwdDemo extends HttpServlet{
5.  public void doPost(HttpServletRequest request,
            HttpServletResponse response)
6.                      throws ServletException,IOException
7.      {   response.setContentType("text/html");
8.          PrintWriter out=response.getWriter();
9.          String username=request.getParameter("login");
10.         out.println("<h1>"+"Welcome "+username+"</h1>");
11.     }
12. }
```



RequestDispatcher: web.xml

```
1.  <web-app>
2.      <servlet>
3.          <servlet-name>FwdDemo</servlet-name>
4.          <servlet-class>disp.FwdDemo</servlet-class>
5.      </servlet>
6.      <servlet>
7.          <servlet-name>CallServlet</servlet-name>
8.          <servlet-class>disp.CallServlet</servlet-class>
9.      </servlet>

10.     <servlet-mapping>
11.         <servlet-name>FwdDemo</servlet-name>
12.         <url-pattern>/FwdDemo</url-pattern>
13.     </servlet-mapping>
14.     <servlet-mapping>
15.         <servlet-name>CallServlet</servlet-name>
16.         <url-pattern>/CallServlet</url-pattern>
17.     </servlet-mapping>
18. </web-app>
```

Servlet Collaboration

sendRedirect()

javax.servlet.http.HttpServletResponse

SendRedirect

- The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

Syntax

```
void sendRedirect(String location)
                    throws IOException
```

Example

```
response.sendRedirect("http://www.abc.ac.in");
response.sendRedirect("/1.html");//relative path
response.sendRedirect("http://localhost:8080/1.html");
                    //absolute path
```

sendRedirect(): Example

```
1.  public class Redirect extends HttpServlet
2.  {   public void doGet( HttpServletRequest request,
                HttpServletResponse response)
3.                      throws ServletException,IOException
4.      {   response.setContentType("text/html");
5.          PrintWriter out=response.getWriter();
6.          String login=request.getParameter("login");
7.          String pwd=request.getParameter("pwd");
8.          if(login.equals("java") && pwd.equals("servlet"))
9.          {  response.sendRedirect("/Dispatcher/Welcome");
10.         }
11.         else
12.         response.sendRedirect("/Dispatcher/redirect.html");
13.     } //doGet
14. }
```