Servlets & Tomcat

Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

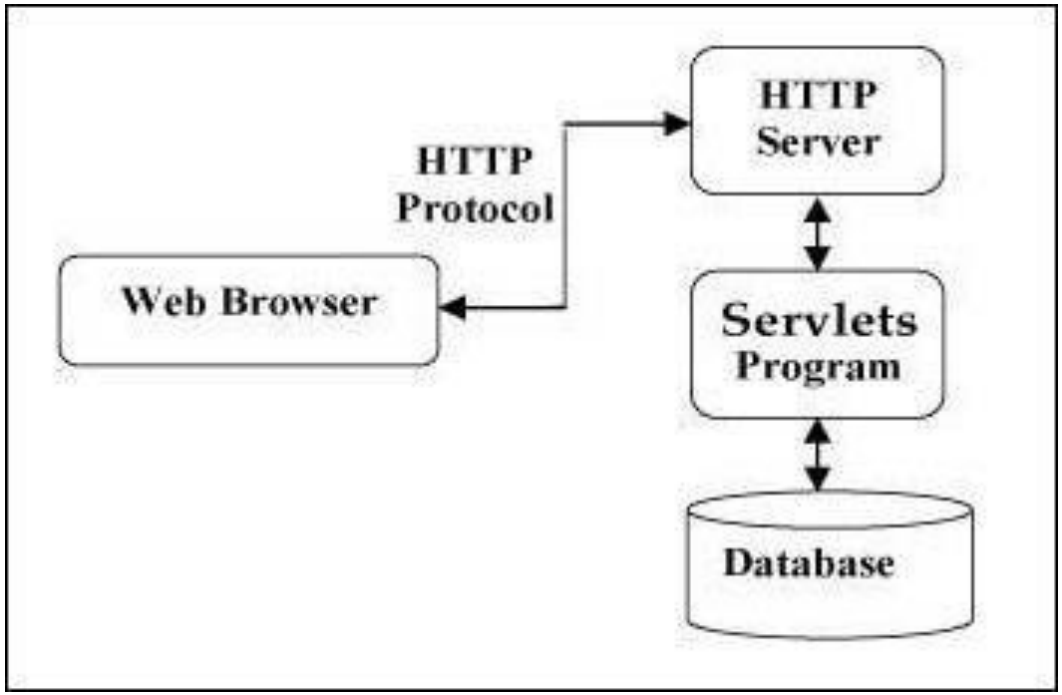
What are Servlets?

- Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.
- Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Advantages of Servlet over CGI

- Performance is significantly better.
- Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
- Servlets are platform-independent because they are written in Java.
- servlets are trusted.
- The full functionality of the Java class libraries is available to a servlet.

Servlets Architecture



Servlets Packages

- Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.
- Servlets can be created using the packages
  - **javax.servlet**
  - **javax.servlet.http**

Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code

Servlet Life Cycle

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

The init() Method

- The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards.
- So, it is used for one-time initializations.
- When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate.
- The init() method simply creates or loads some data that will be used throughout the life of the servlet.

- The init method definition looks like this −

```
public void init() throws ServletException {
  // Initialization code...
}
```

The service() Method

- 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 formatted response back to the client.
- When server receives a request for a servlet, the service() method checks the HTTP request type (GET, POST) and calls doGet, doPost, methods as appropriate.

- **Here is the signature of this method –**
- public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException { }

The doGet() Method

- A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

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

The doPost() Method

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

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

The destroy() Method

- 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 database connections, halt background threads, and perform other such cleanup activities.
- After the destroy() method is called, the servlet object is marked for garbage collection.

- The destroy method definition looks like this −

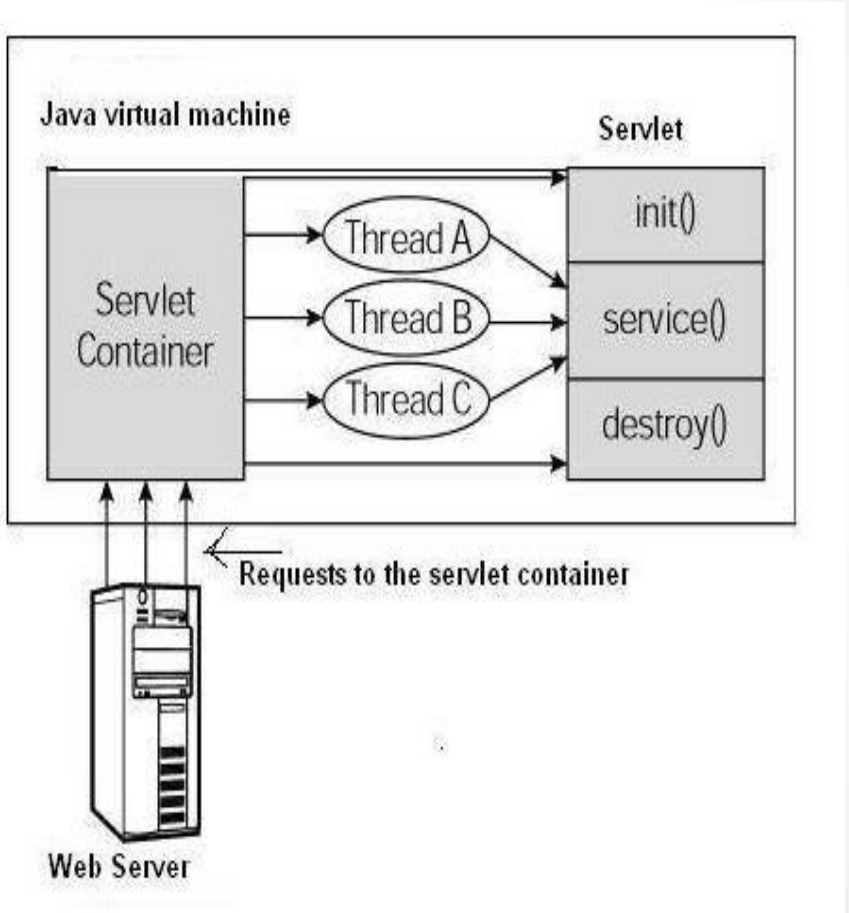
```
public void destroy() {
  // Finalization code...
}
```

Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

Architecture Diagram

- First the HTTP requests coming to the server are delegated to the servlet container.
- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

Requirements

Before running Servlet your machine requires following tools

- 1> Eclipse (IDE-integrated development environment)
- 2> Tomcat (Web Server)

Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

Apache-Tomcat

- Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF).
- Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL (Expression Language) a simple non-procedural scripting language that can be used to evaluate dynamic expressions within a JSP page, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.

Tomcat- Components

- Catalina
- Coyote
- Jasper
- Cluster
- High availability
- Web application

Tomcat- Components

Catalina

- Catalina is Tomcat's servlet container.
- Catalina is the powerhouse of Tomcat. It’s where the real magic happens.
- When a request arrives, Catalina decides whether it’s a servlet or a JSP (JavaServer Page). If it is a servlet, Catalina takes charge.
- Catalina manages the lifecycle of servlets, including loading, initialization, and handling requests.
- Catalina implements Sun Microsystems's specifications for servlet and JavaServer Pages (JSP).
- In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles assigned to those users.
- Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.

Tomcat- Components

Coyote

- Coyote is a Connector component for Tomcat that supports the HTTP 1.1 protocol as a web server.
- This allows Catalina, nominally a Java Servlet or JSP container, to also act as a plain web server that serves local files as HTTP documents.
- Coyote listens for incoming connections to the server on a specific TCP port and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client.

Tomcat- Components

Jasper

- Jasper is Tomcat's JSP Engine.
- Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.
- From Jasper to Jasper 2, important features were added:
  - JSP Tag library pooling - Each tag markup in JSP file is handled by a tag handler class.
  - Background JSP compilation - While recompiling modified JSP Java code, the older version is still available for server requests.
  - Recompile JSP when included page changes - Pages can be inserted and included into a JSP at runtime
  - JDT Java compiler - Jasper 2 can use the Eclipse JDT Java compiler

Tomcat- Components

- **These components are added from Tomcat 7.0 version**
- **Cluster**
- This component has been added to manage large applications.
- **High availability**
- A high-availability feature has been added to facilitate the scheduling of system upgrades (e.g. new releases, change requests) without affecting the live environment.
- **Web application**
- It has also added user- as well as system-based web applications enhancement to add support for deployment across the variety of environments.

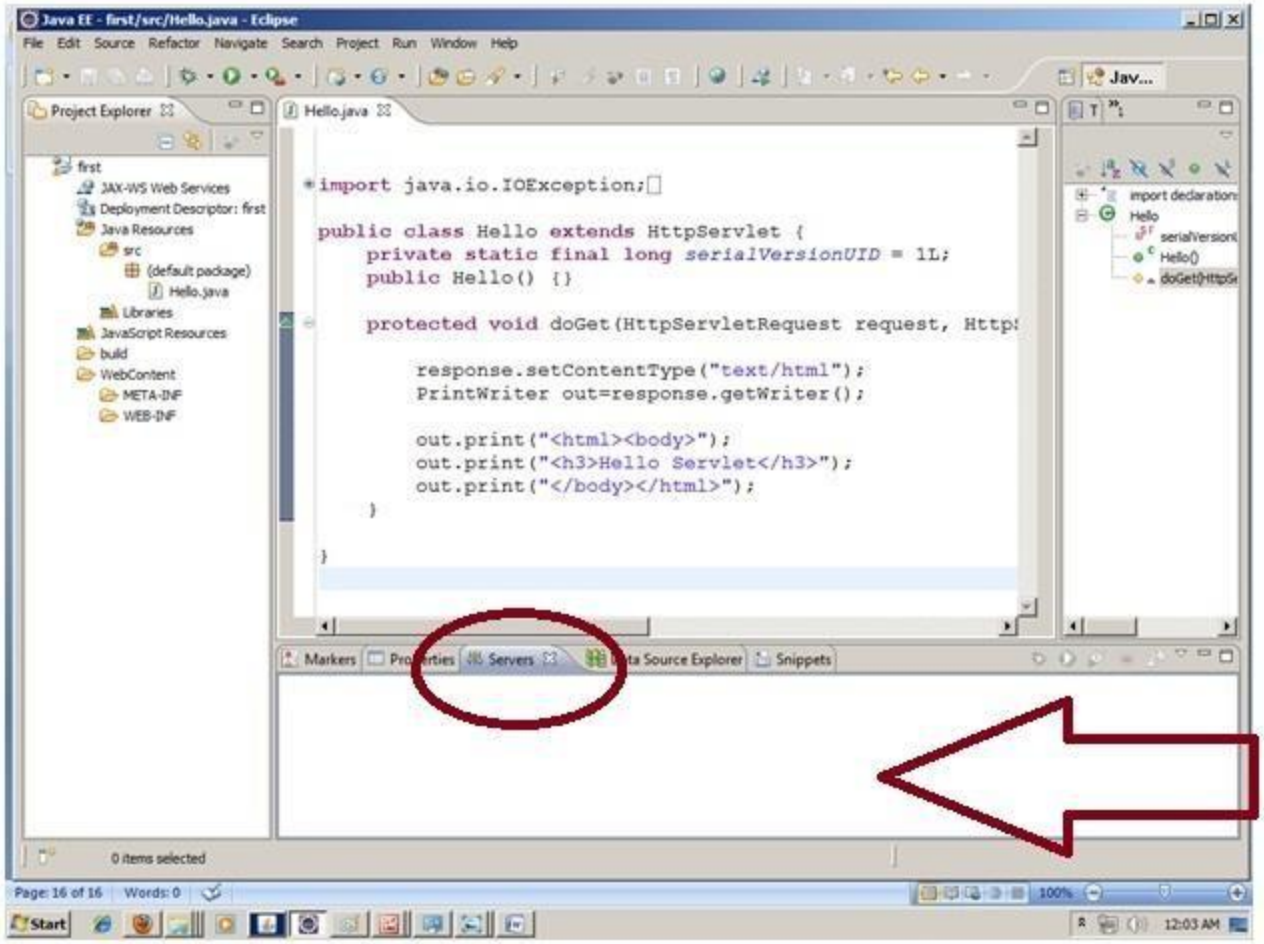
Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

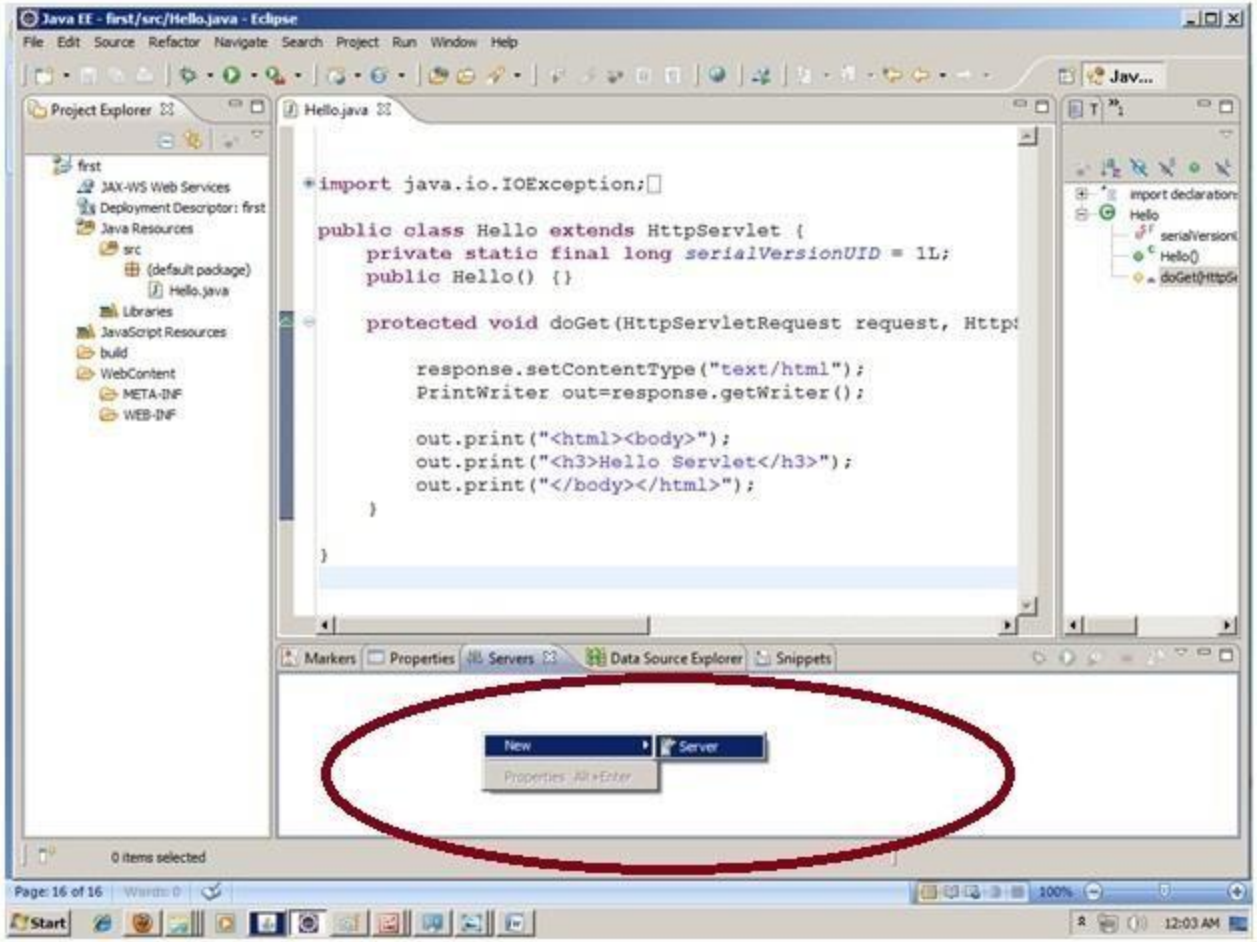
How to configure tomcat server in Eclipse ? (One time Requirement)

- If you are using **Eclipse IDE first time**, you need to configure the tomcat server First.
- For configuring the tomcat server in eclipse IDE,
- **click on servers tab at the bottom side of the IDE -> right click on blank area -> New -> Servers -> choose tomcat then its version -> next -> click on Browse button -> select the apache tomcat root folder previous to bin -> next -> addAll -> Finish.**

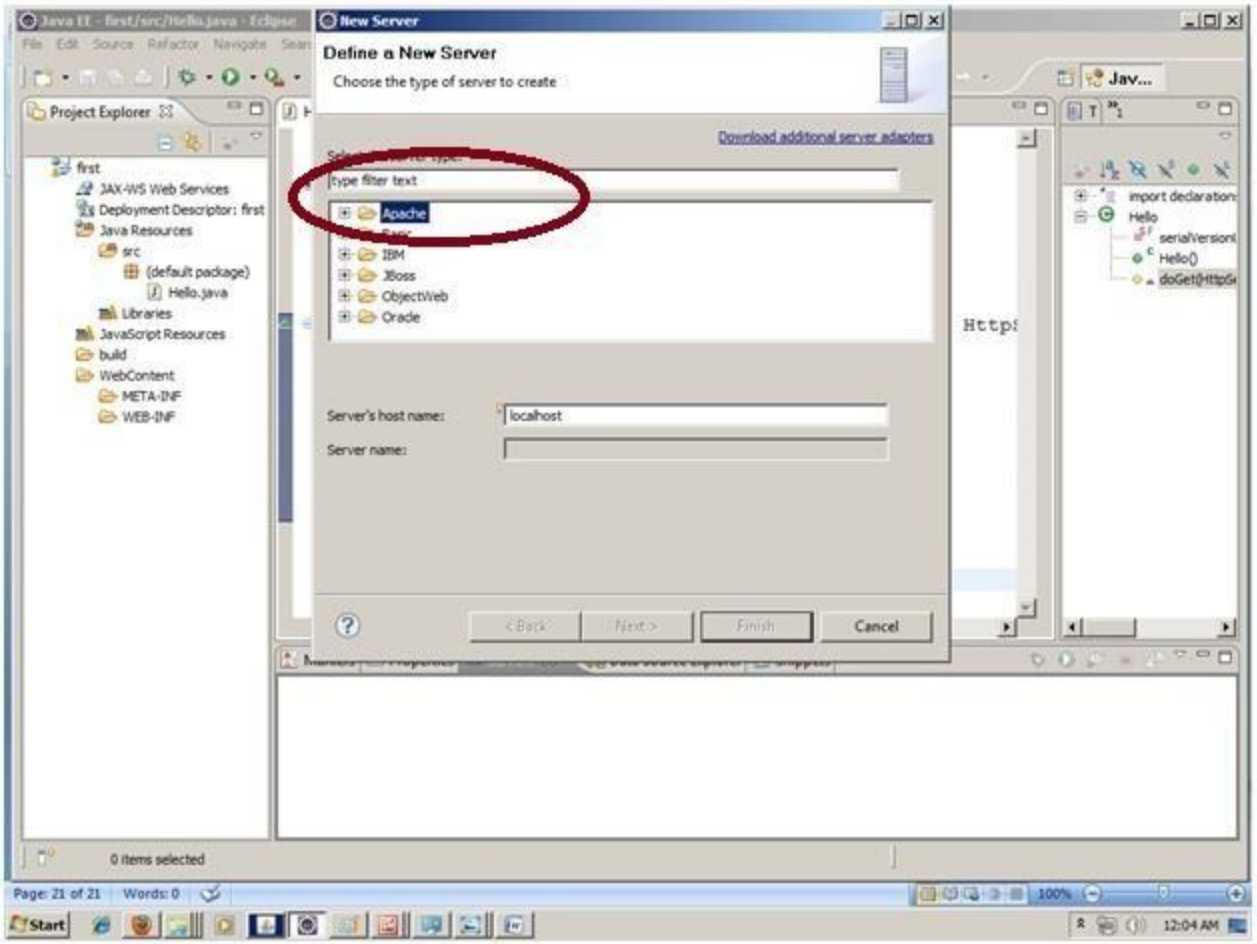
How to configure tomcat server in Eclipse ?



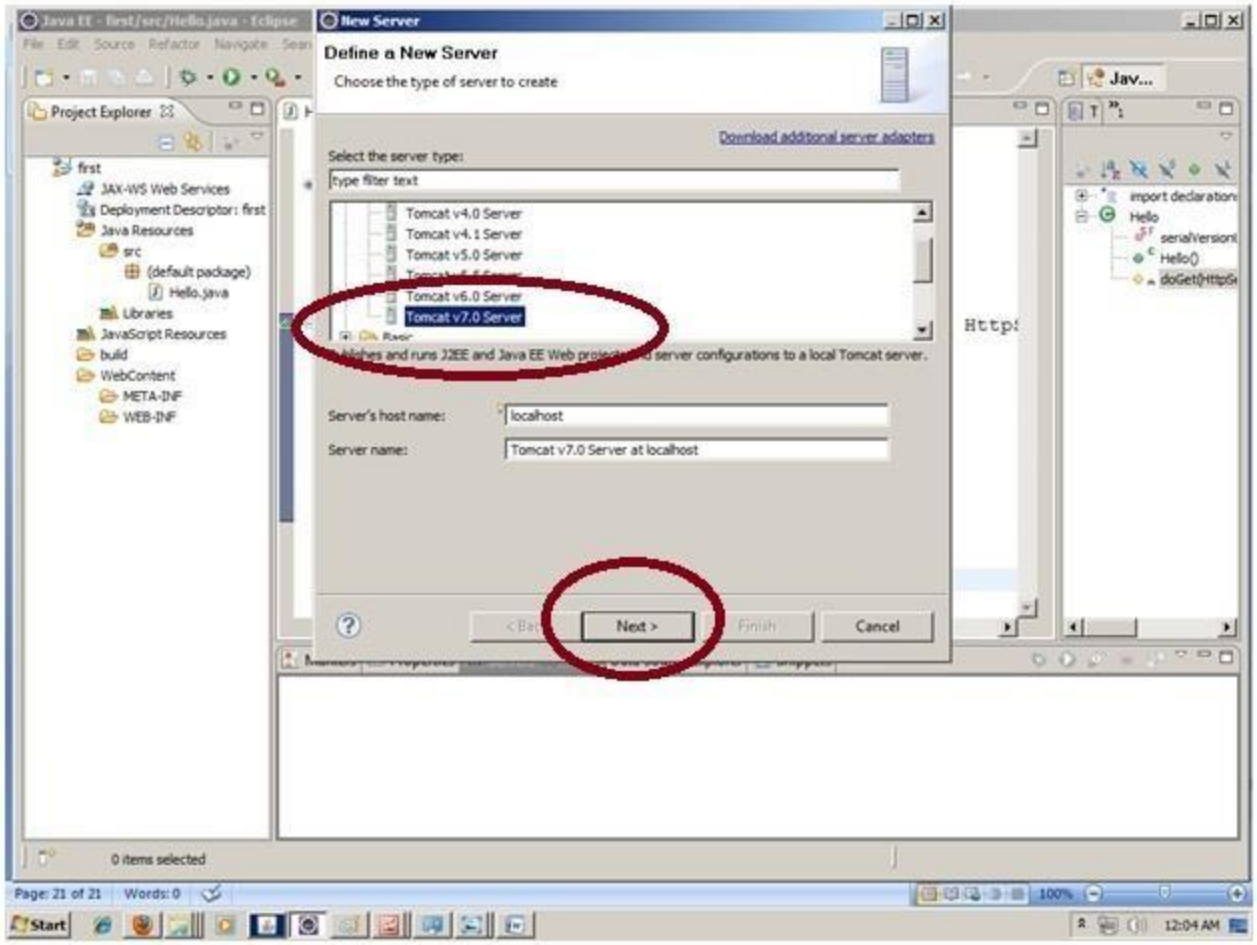
How to configure tomcat server in Eclipse ?



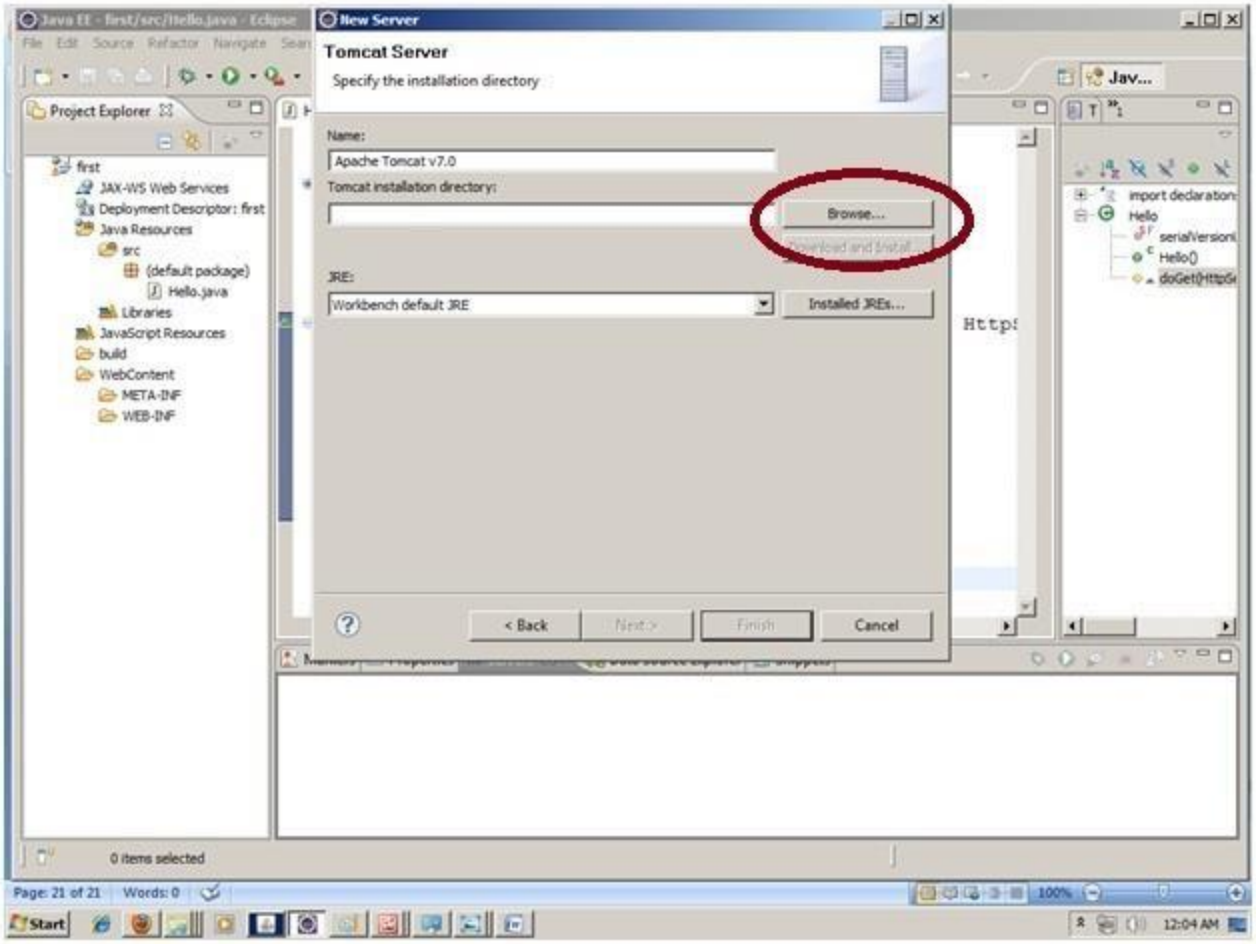
How to configure tomcat server in Eclipse ?



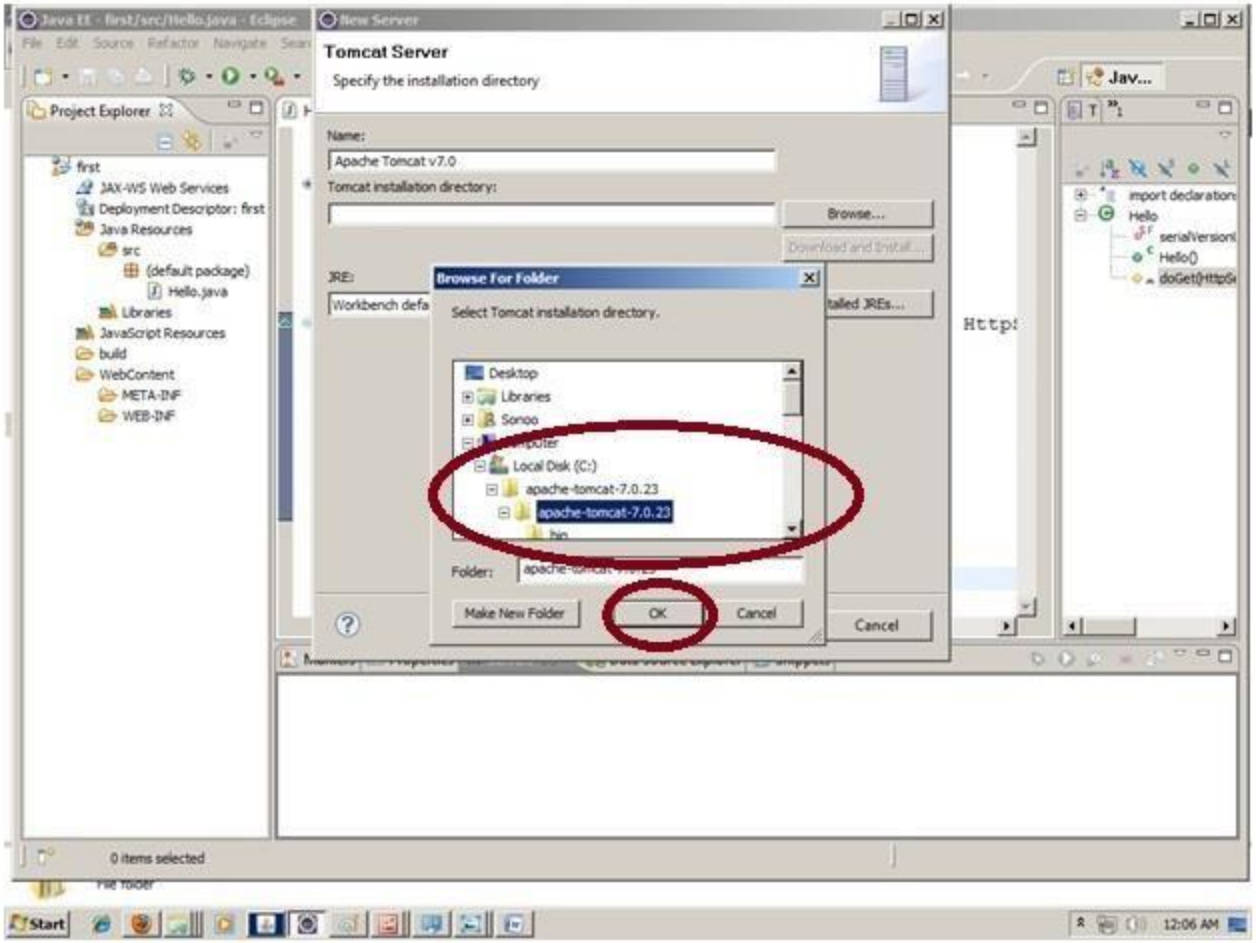
How to configure tomcat server in Eclipse ?



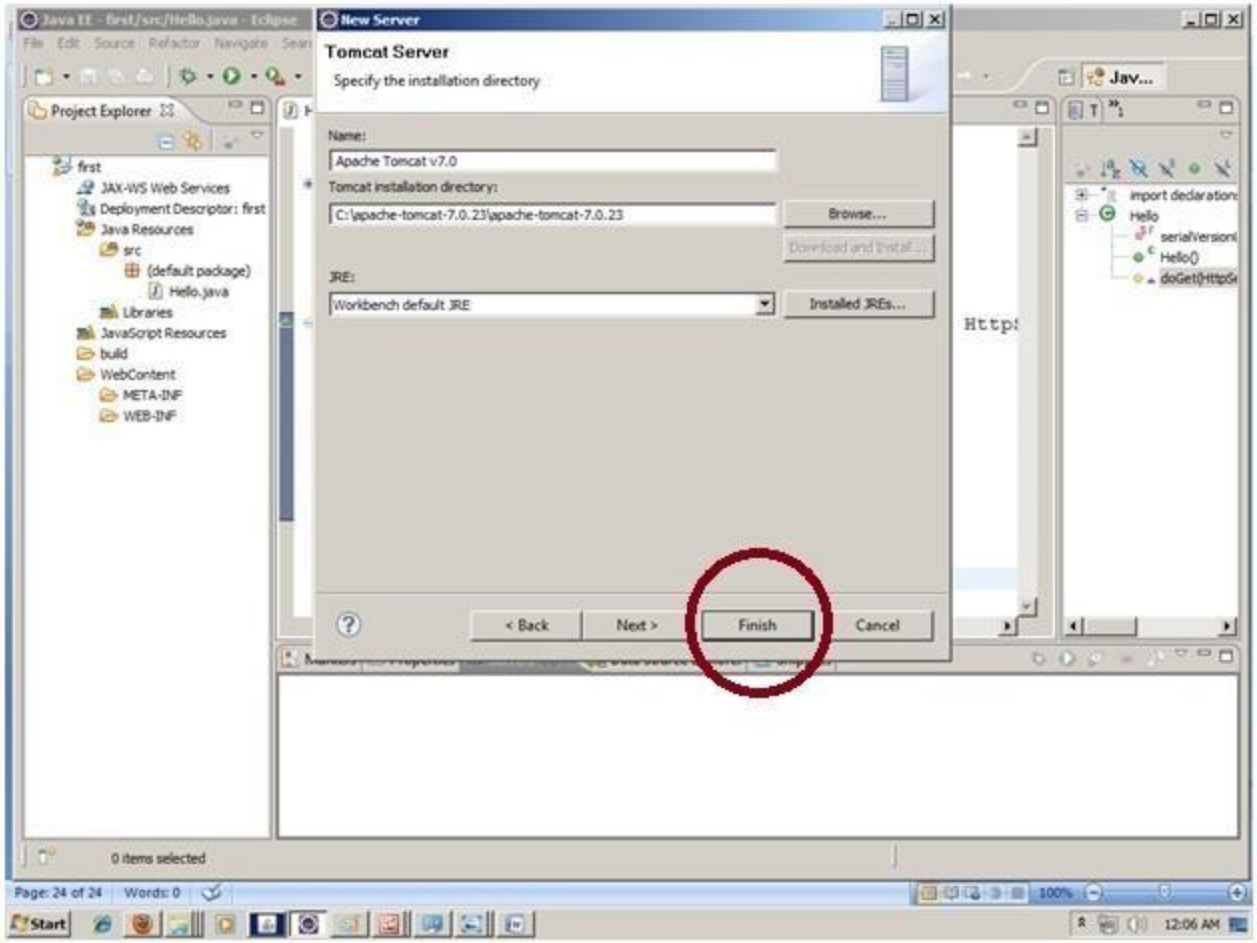
How to configure tomcat server in Eclipse ?



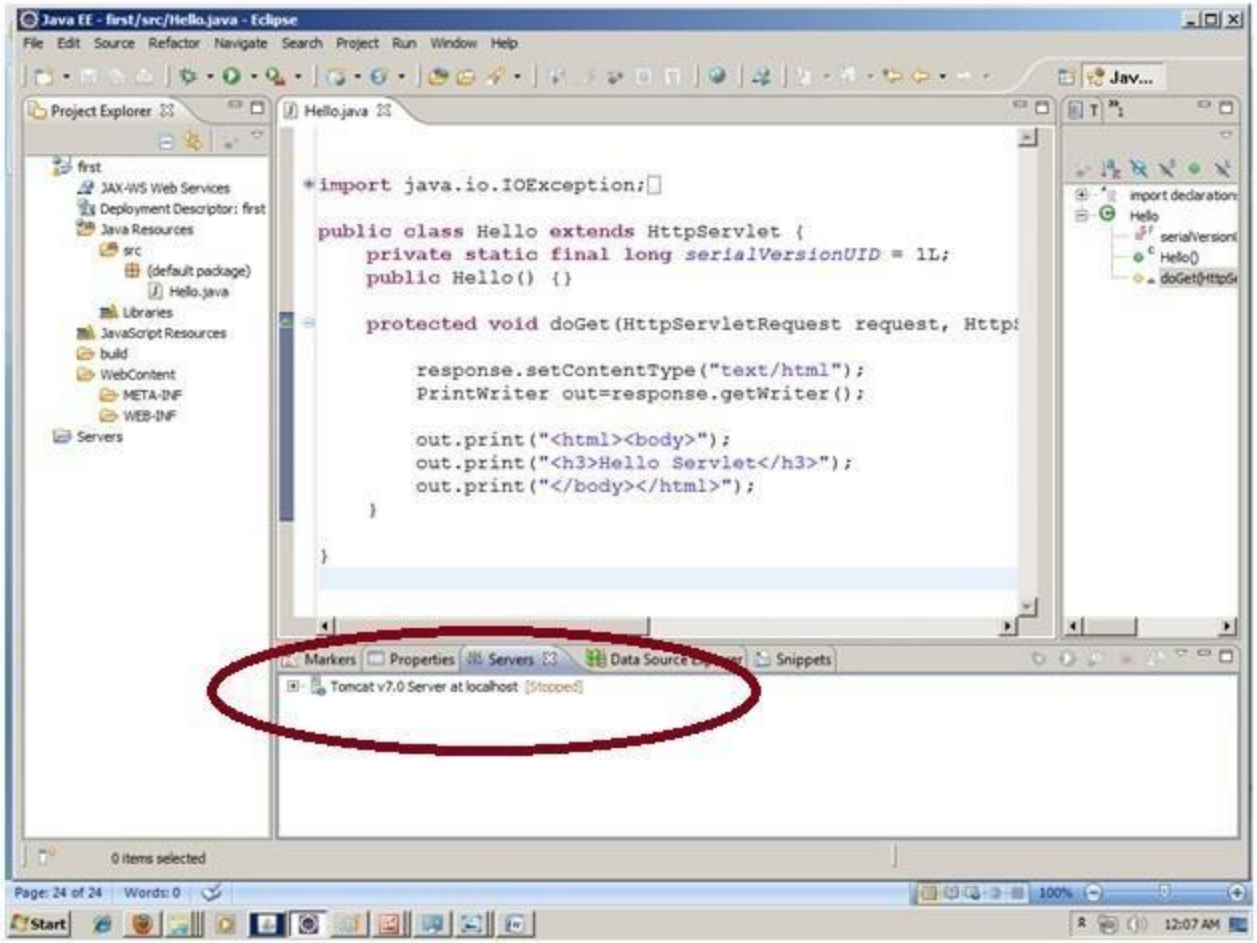
How to configure tomcat server in Eclipse ?



How to configure tomcat server in Eclipse ?



How to configure tomcat server in Eclipse ?



Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

Steps to run servlet in Eclipse

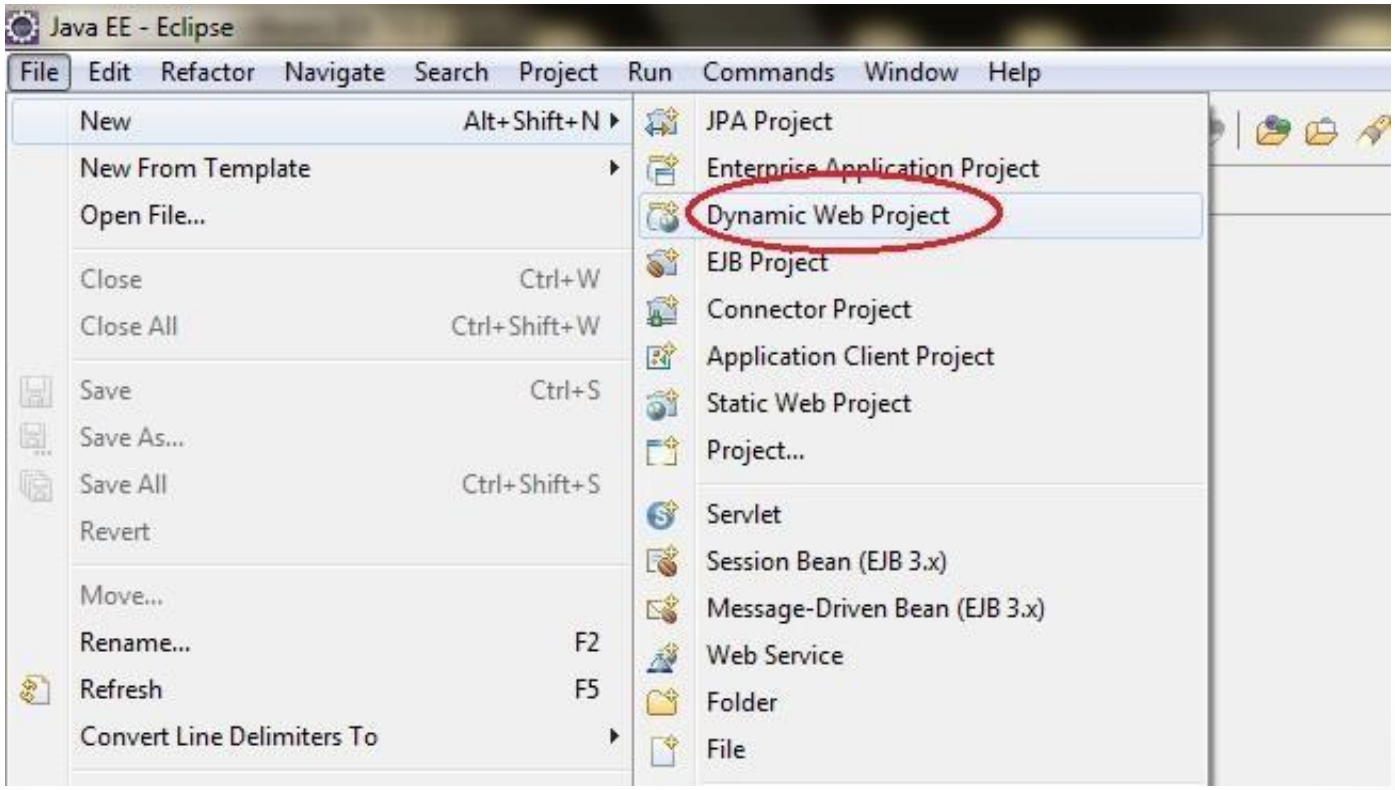
- Create a Dynamic web project
- create a servlet
- add servlet-api.jar file
- Run the servlet

Steps to run servlet in Eclipse

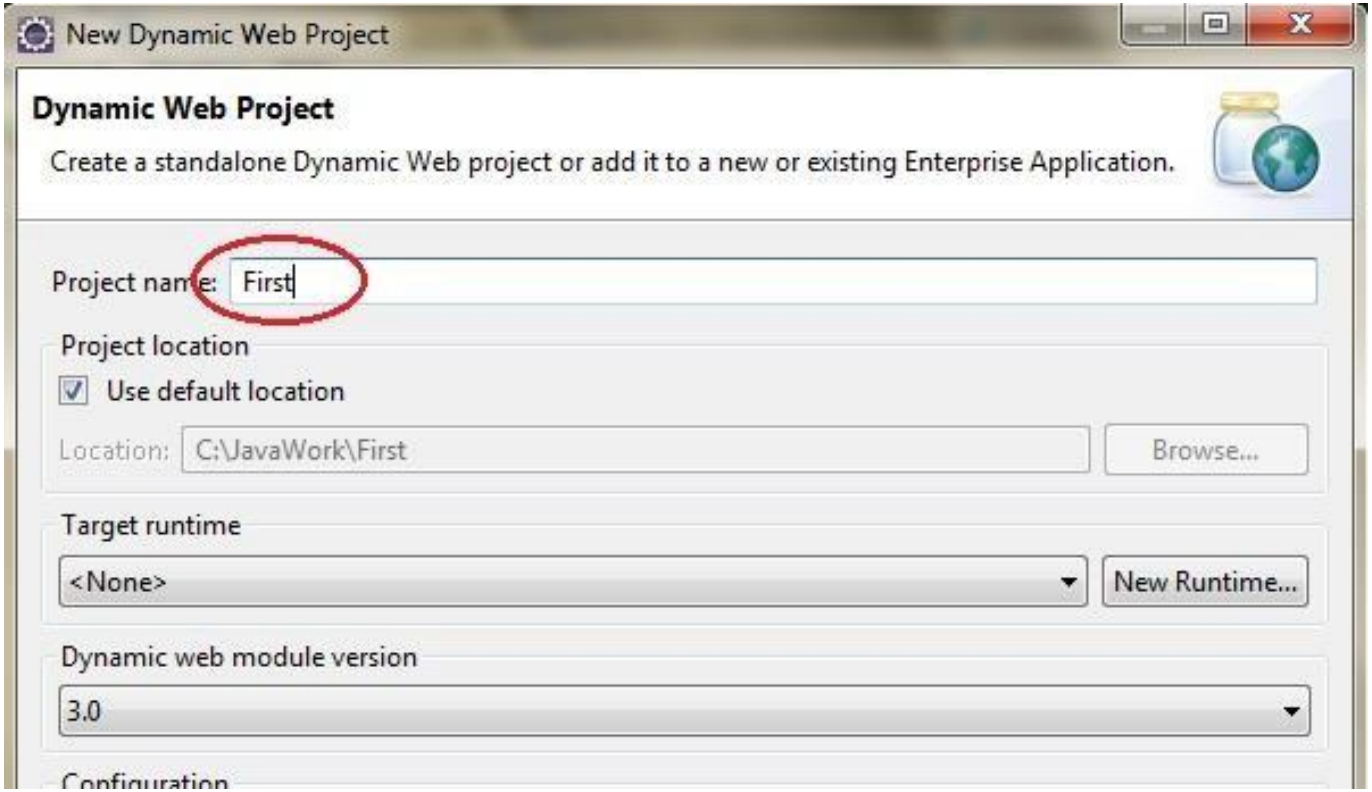
- Create a Dynamic web project
- create a servlet
- add servlet-api.jar file
- Run the servlet

Create the dynamic web project

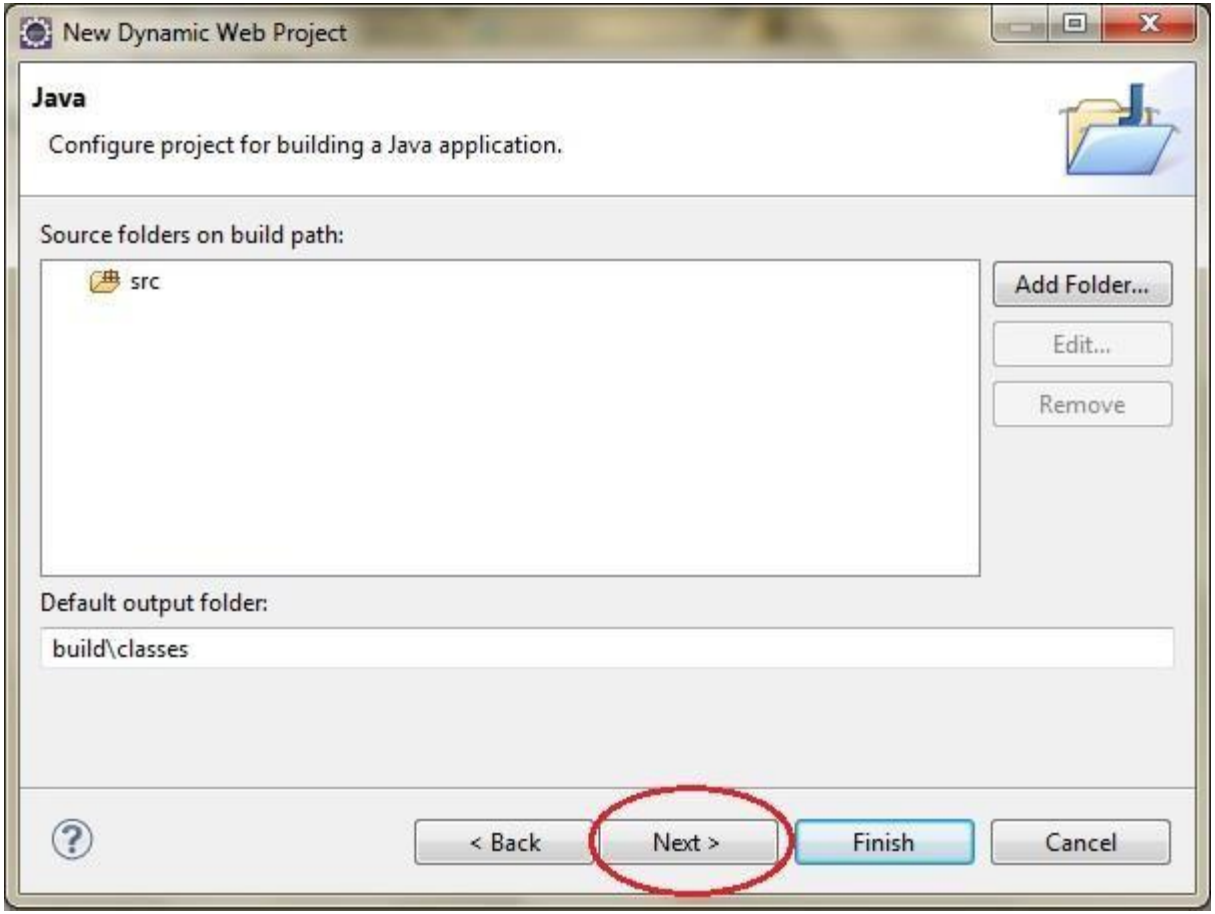
Start Eclipse and In Menu : **File** -> **New** -> **Dynamic Web Project**



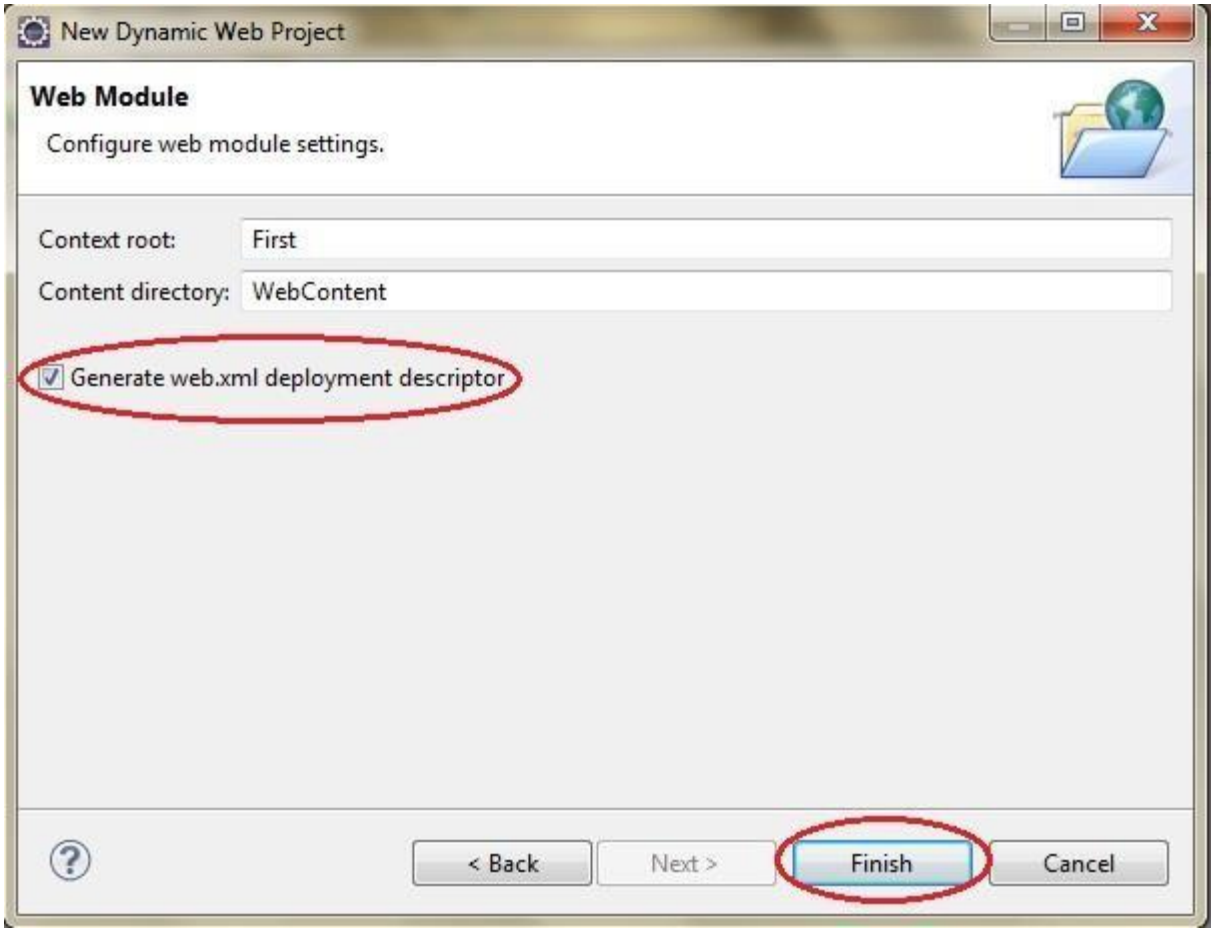
Create the dynamic web project



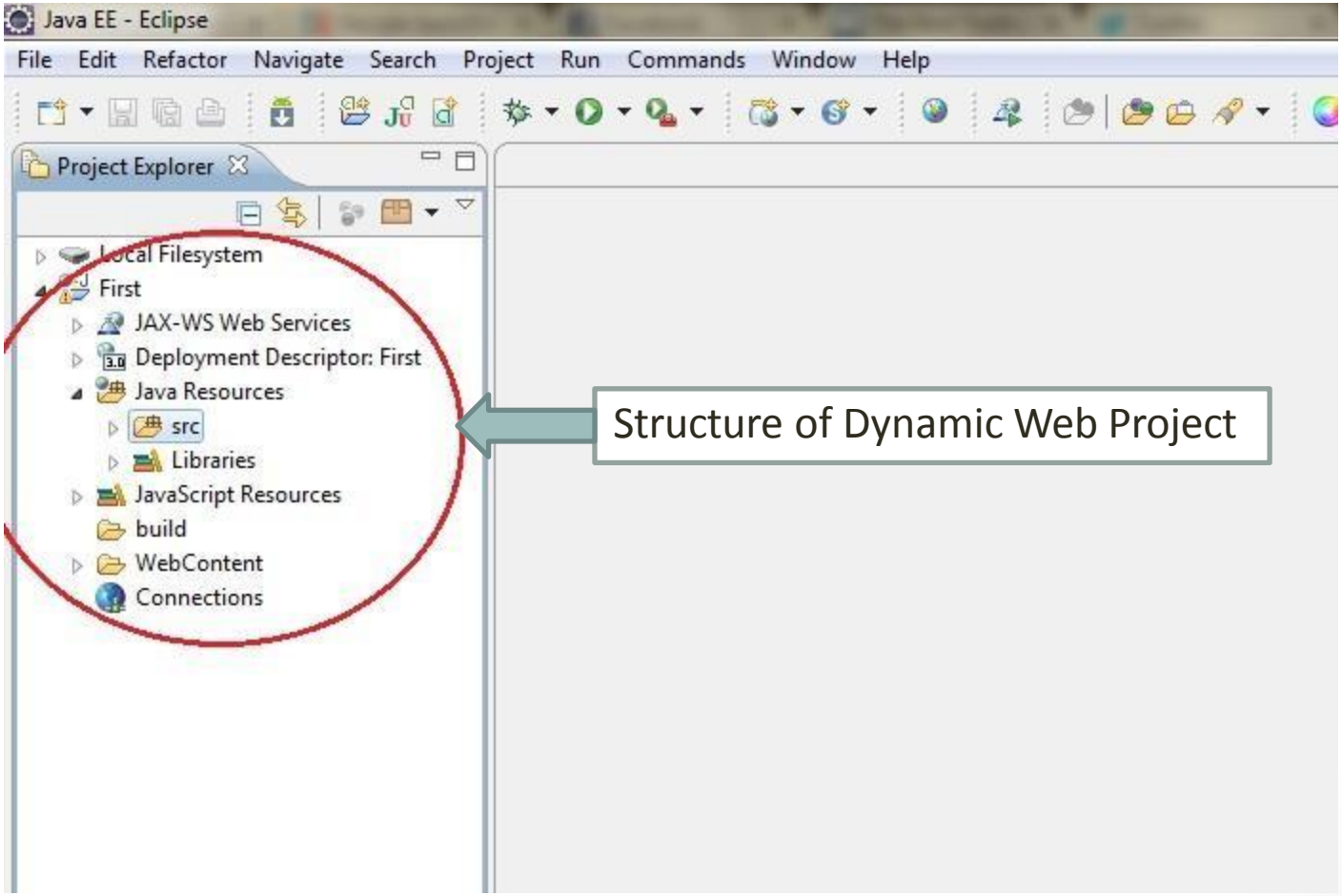
Create the dynamic web project



Create the dynamic web project



Create the dynamic web project

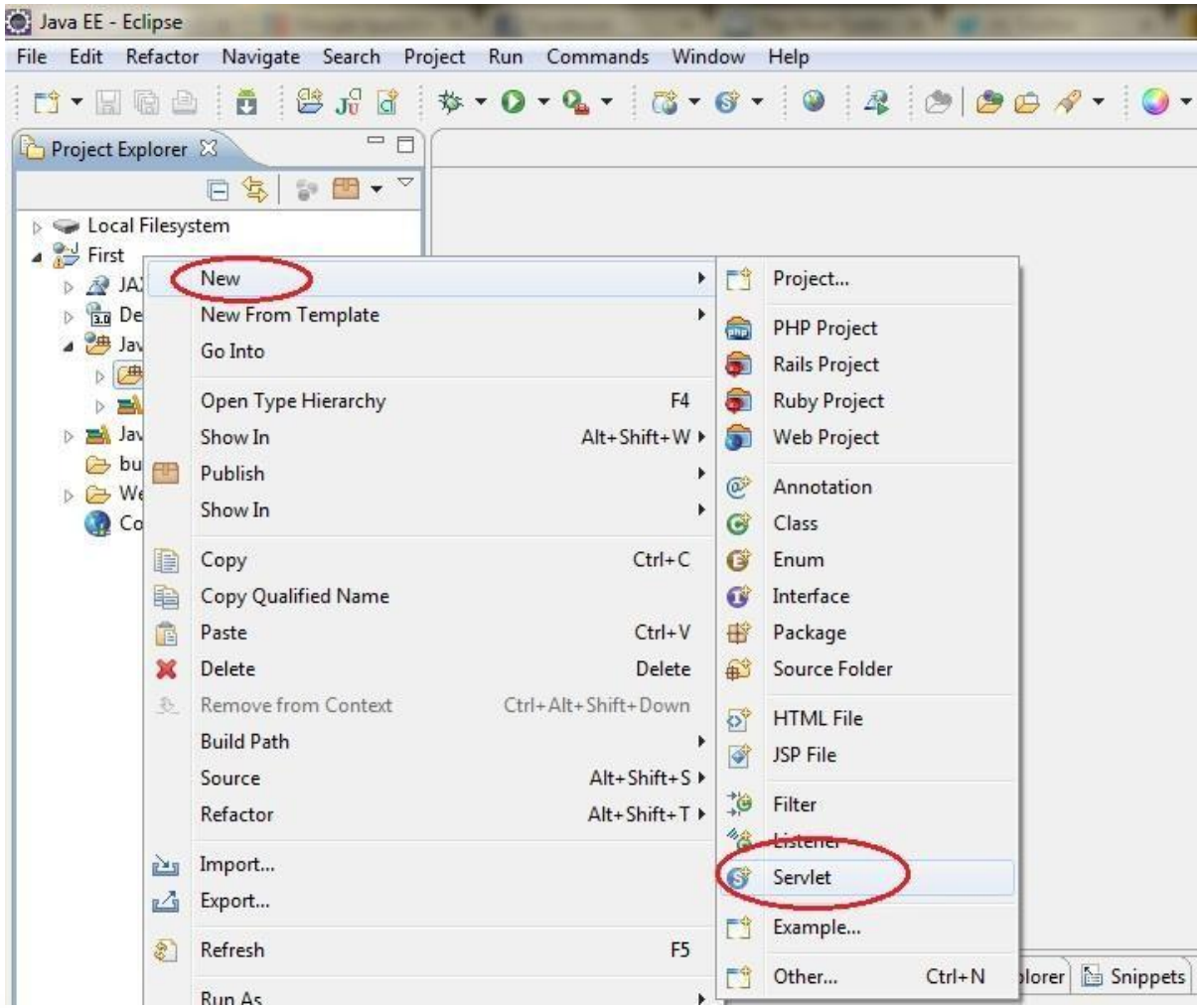


Steps to run servlet in Eclipse

- Create a Dynamic web project
- create a servlet
- add servlet-api.jar file
- Run the servlet

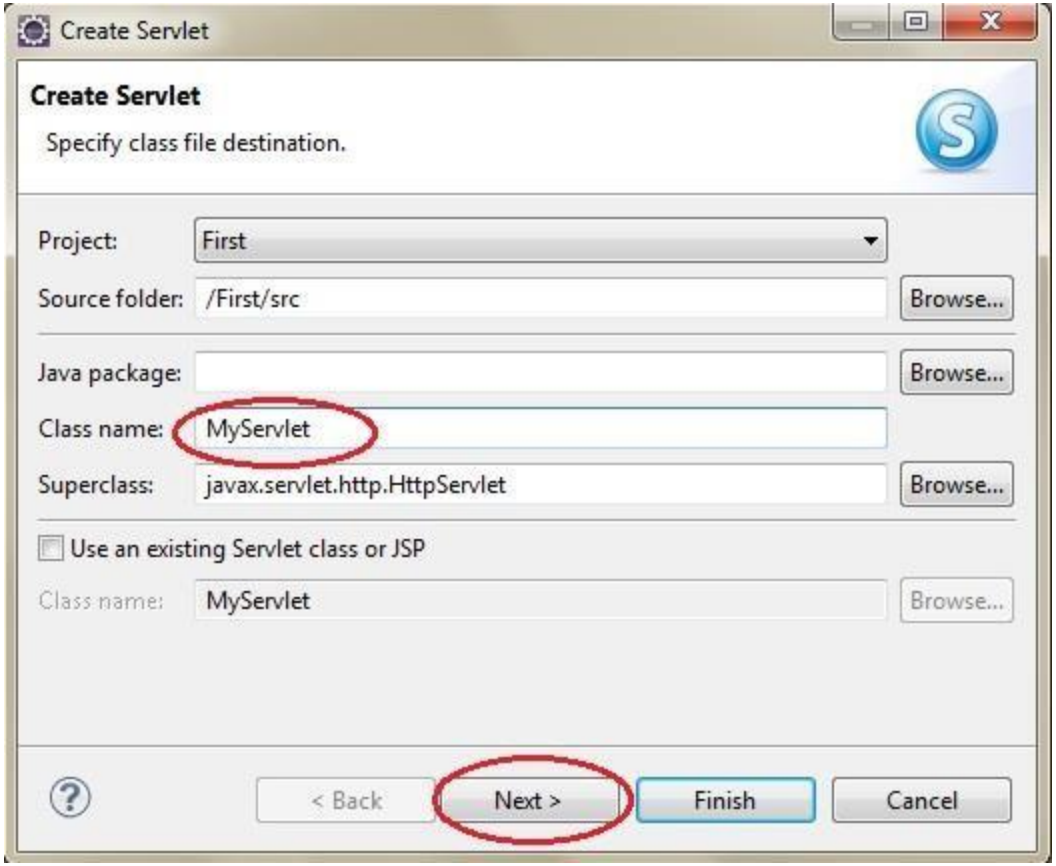
Create a servlet

Right click on Src -> New -> Sevlet

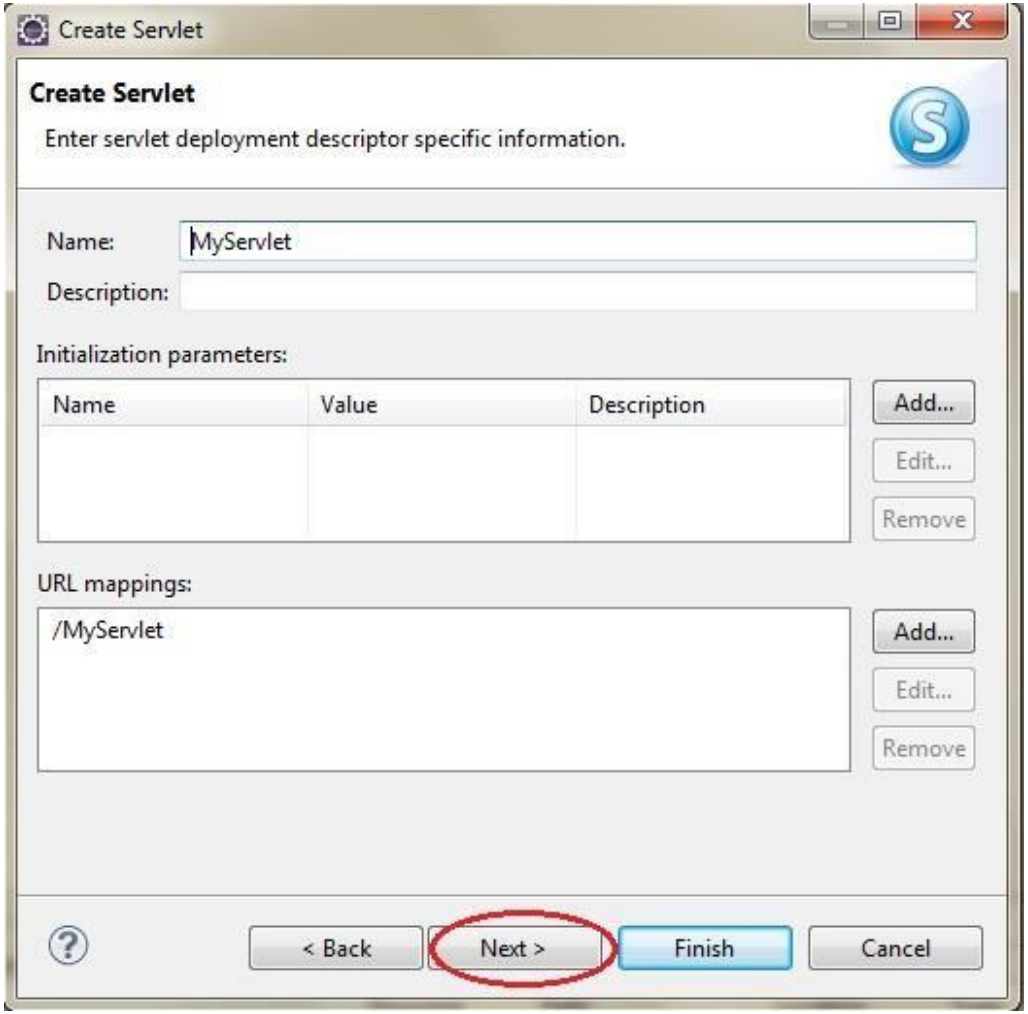


Create a servlet

Give name of class

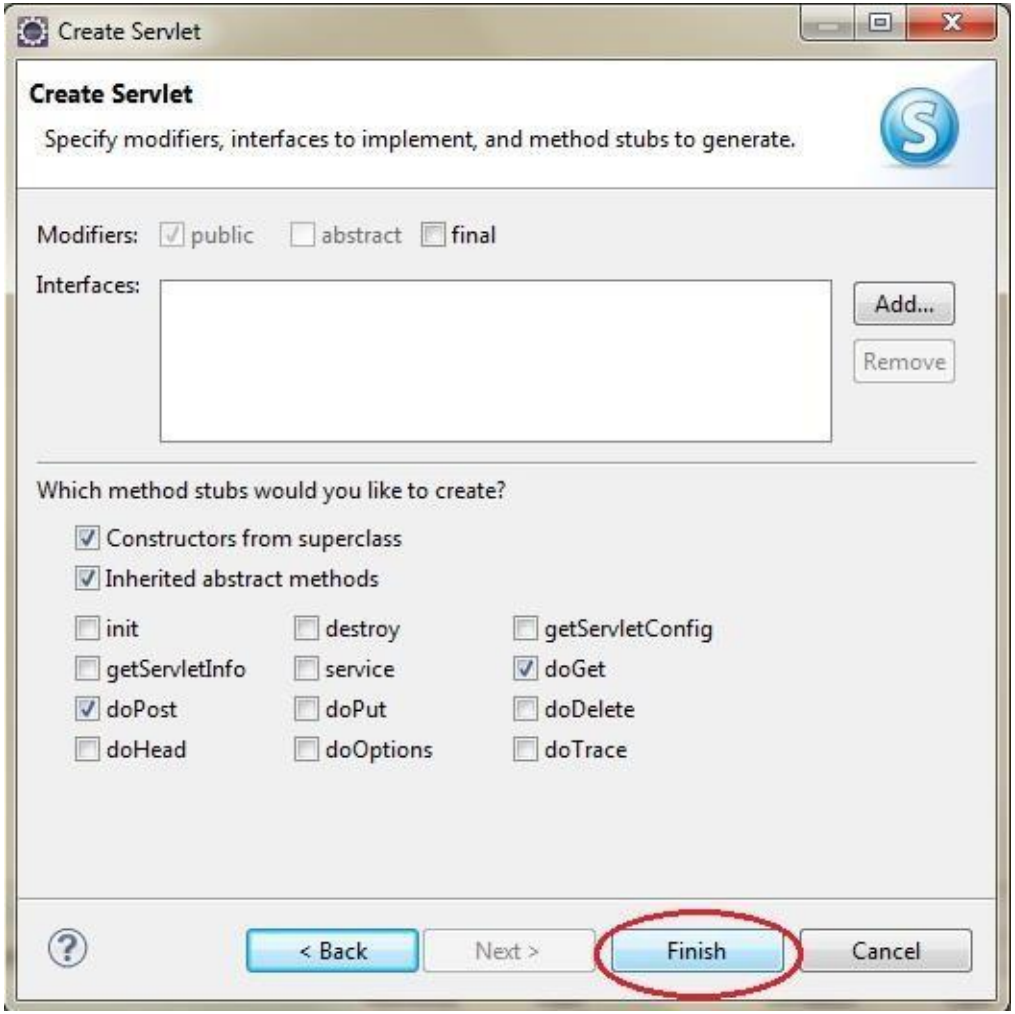


Create a servlet



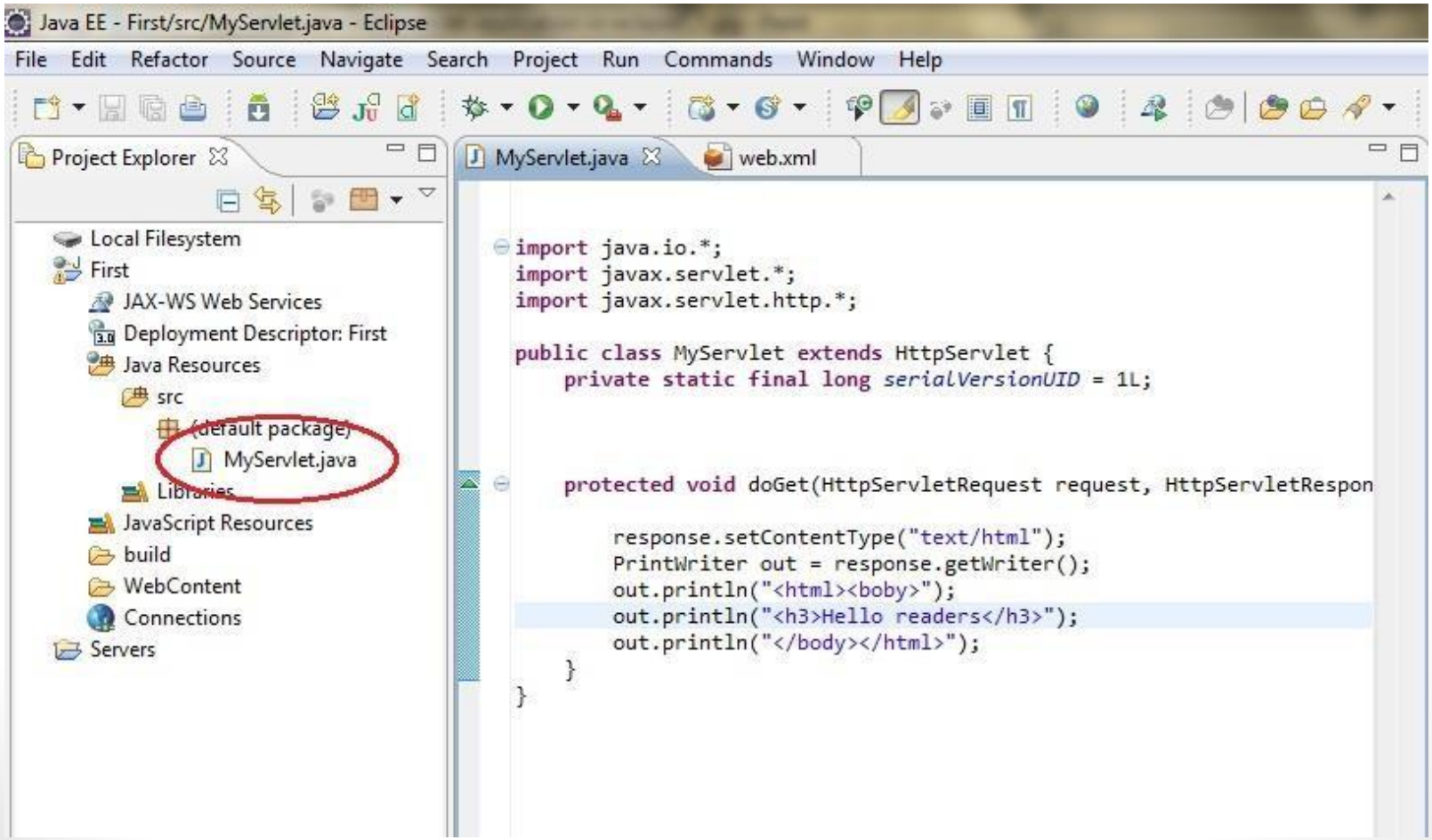
Create a servlet

Either select doGet or doPost



Create a servlet

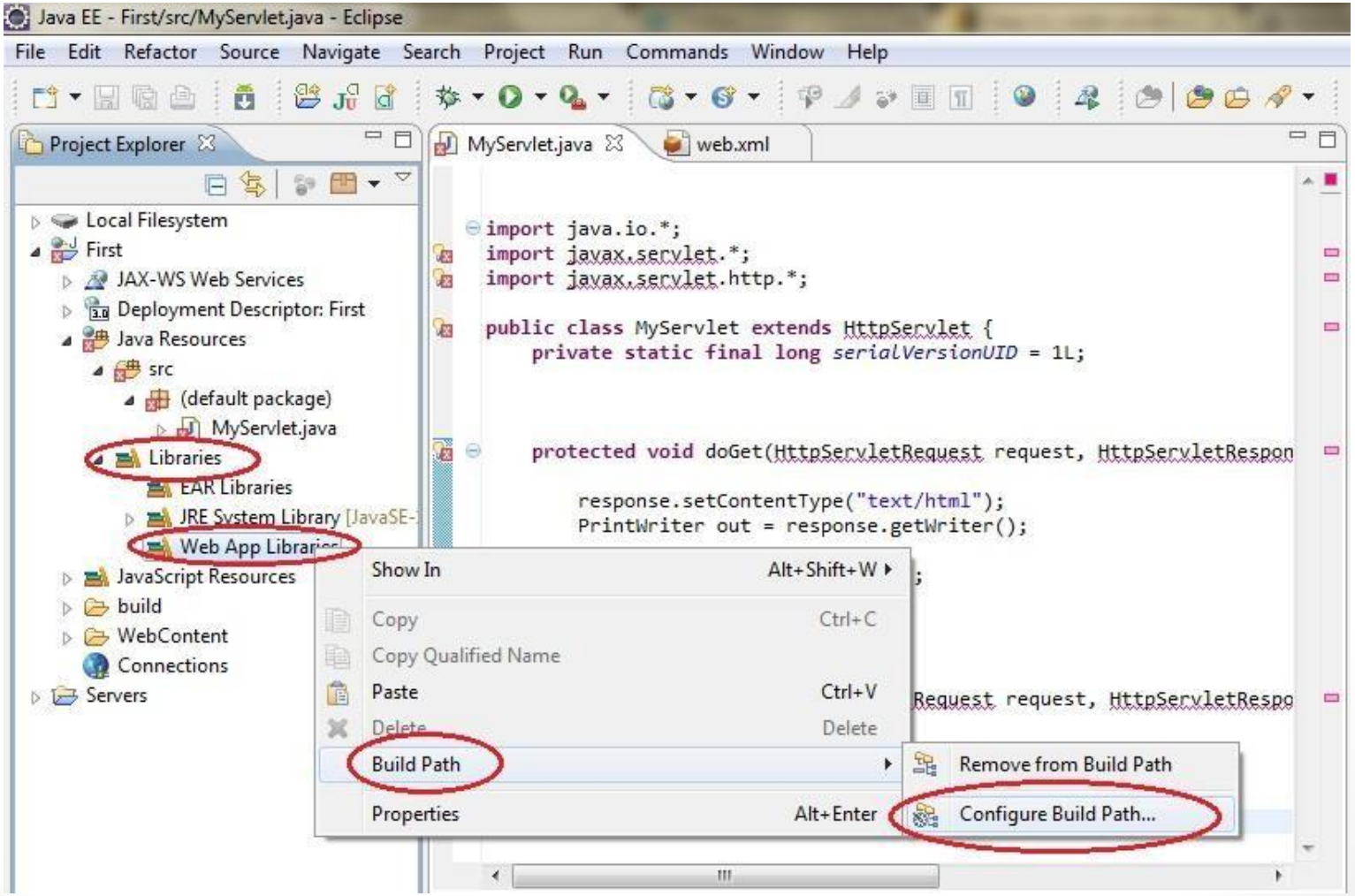
Servlet is created, you can write the code now.



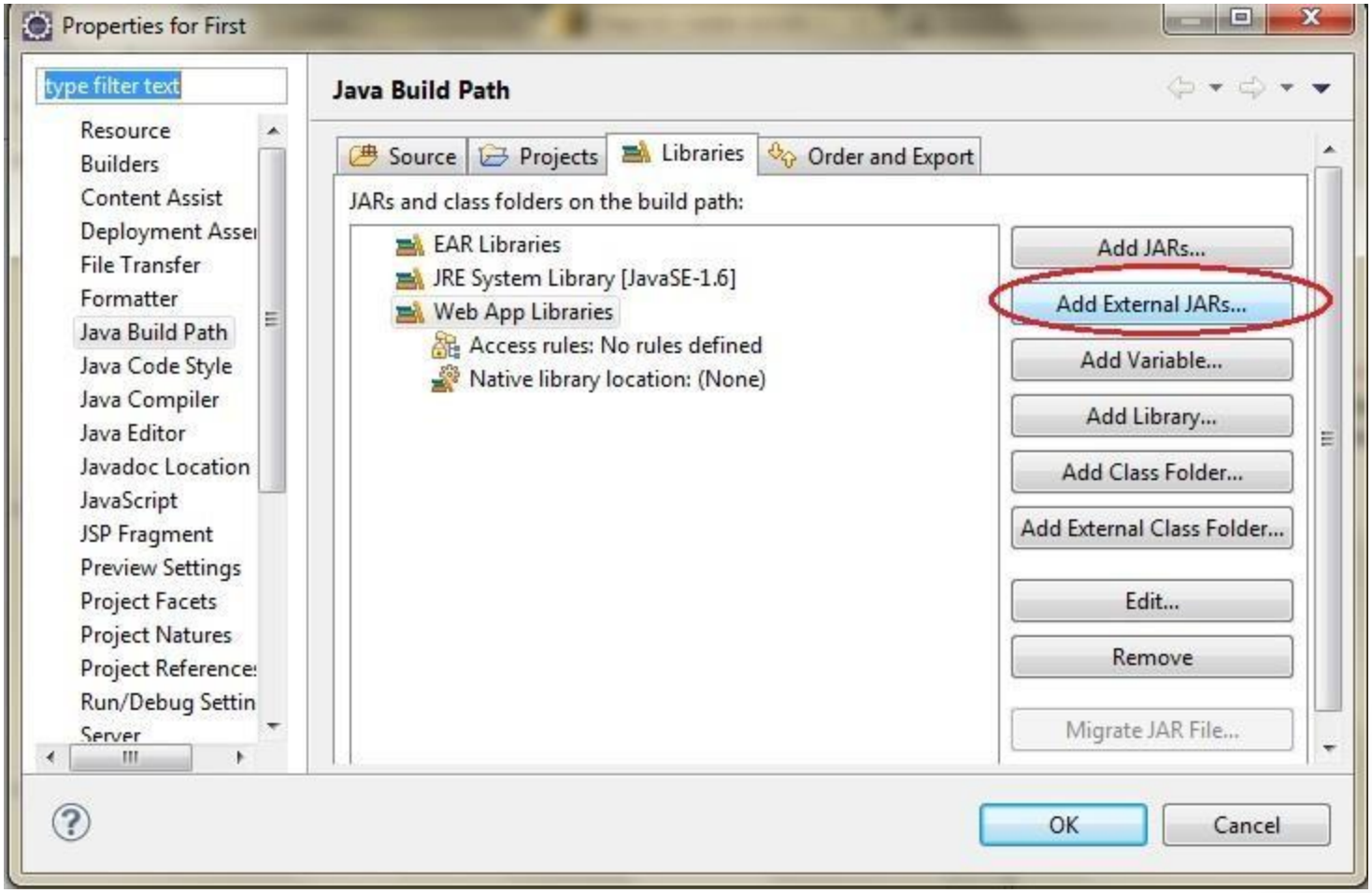
Steps to run servlet in Eclipse

- Create a Dynamic web project
- create a servlet
- add servlet-api.jar file
- Run the servlet

Add servlet-api.jar file

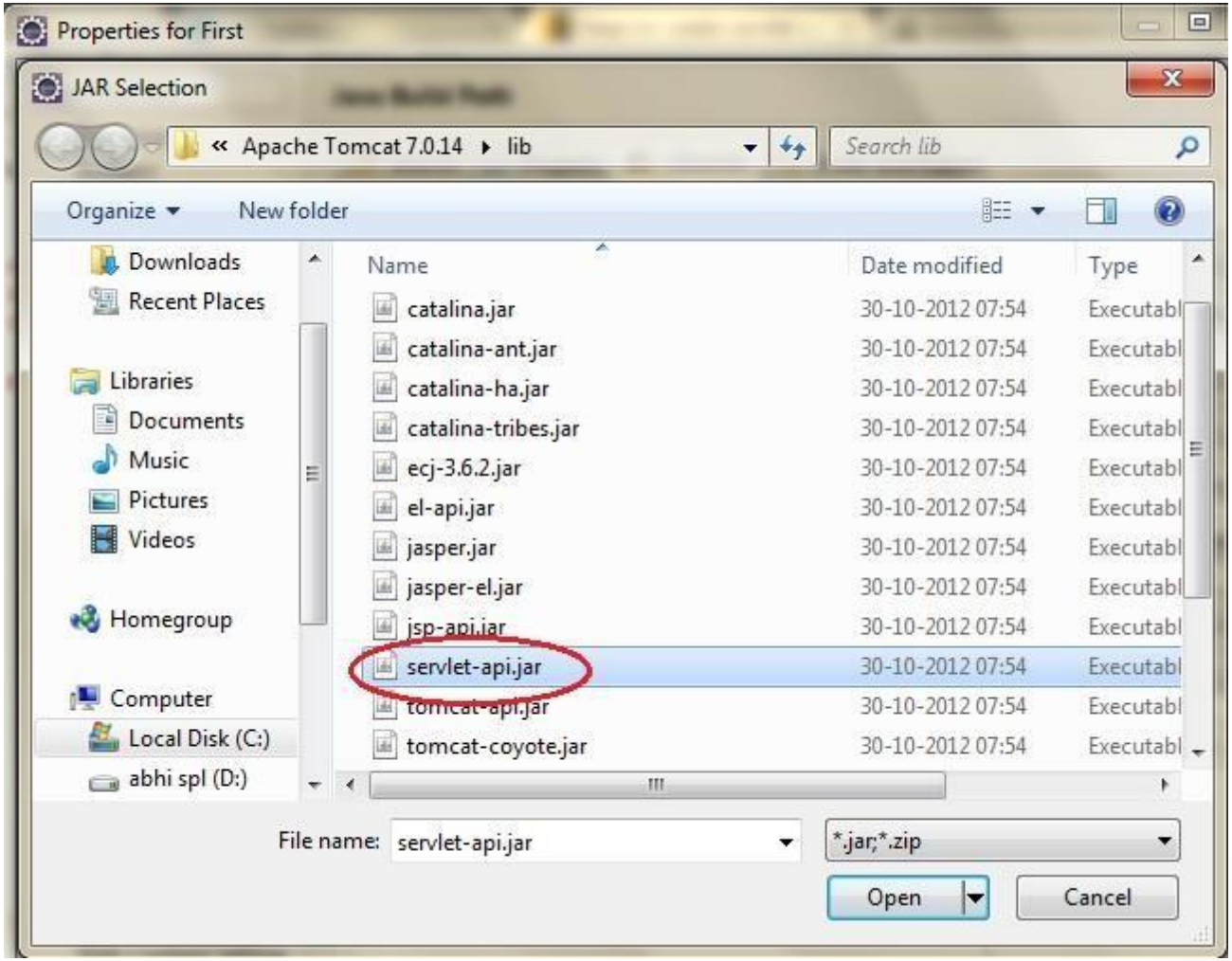


Add servlet-api.jar file

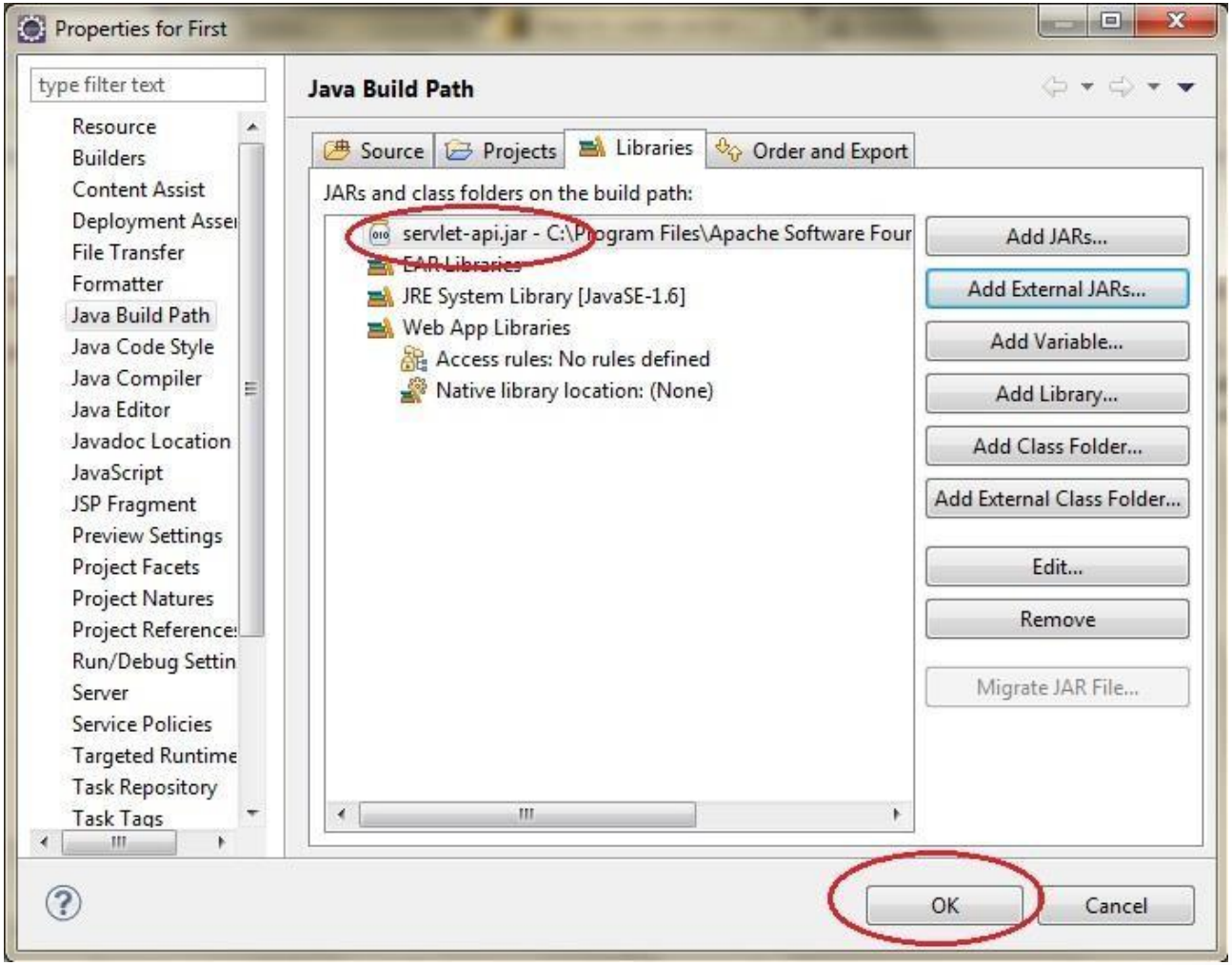


Add servlet-api.jar file

In Tomcat folder goto Lib folder and select servlet-api.jar file



Add servlet-api.jar file

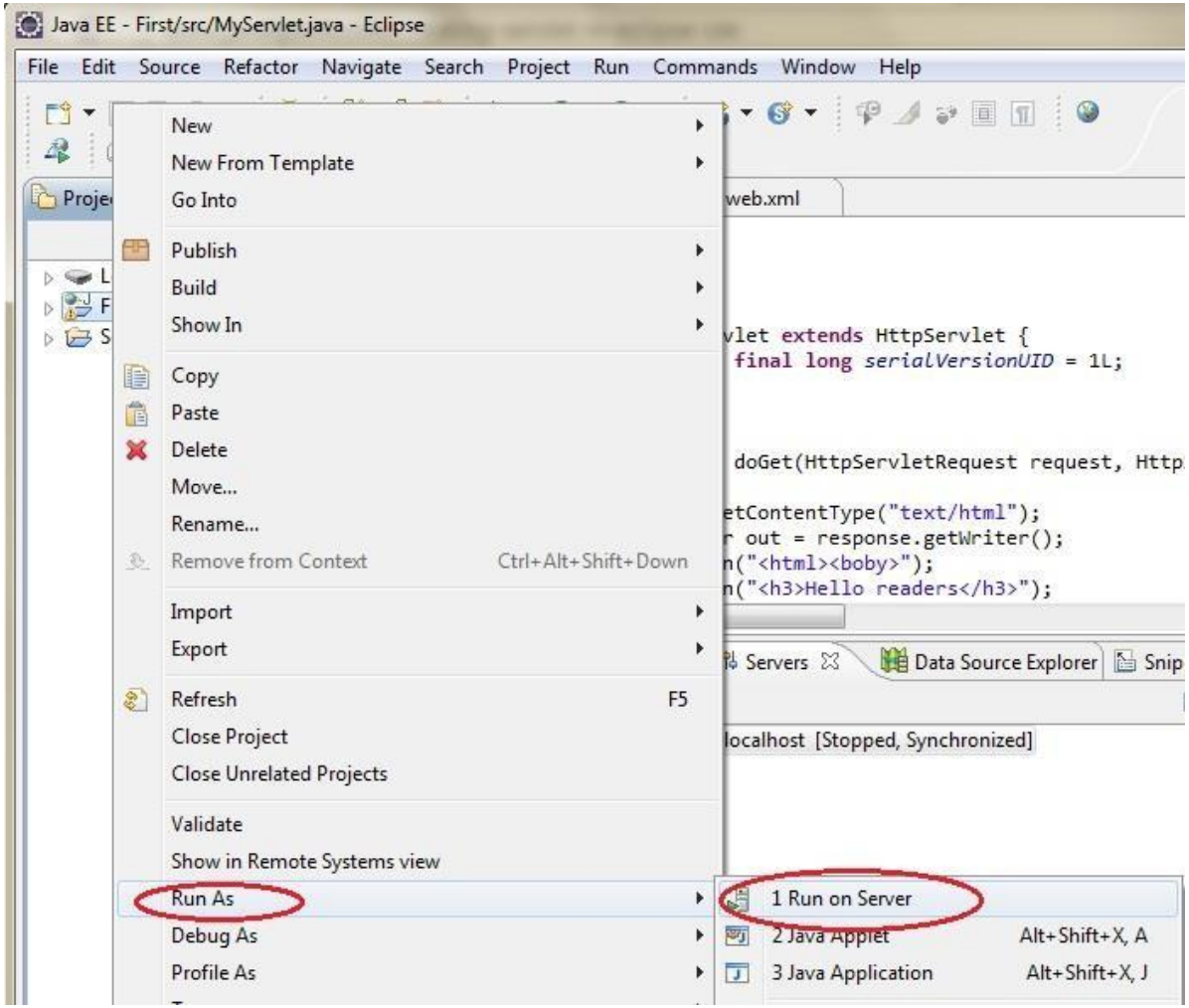


Steps to run servlet in Eclipse

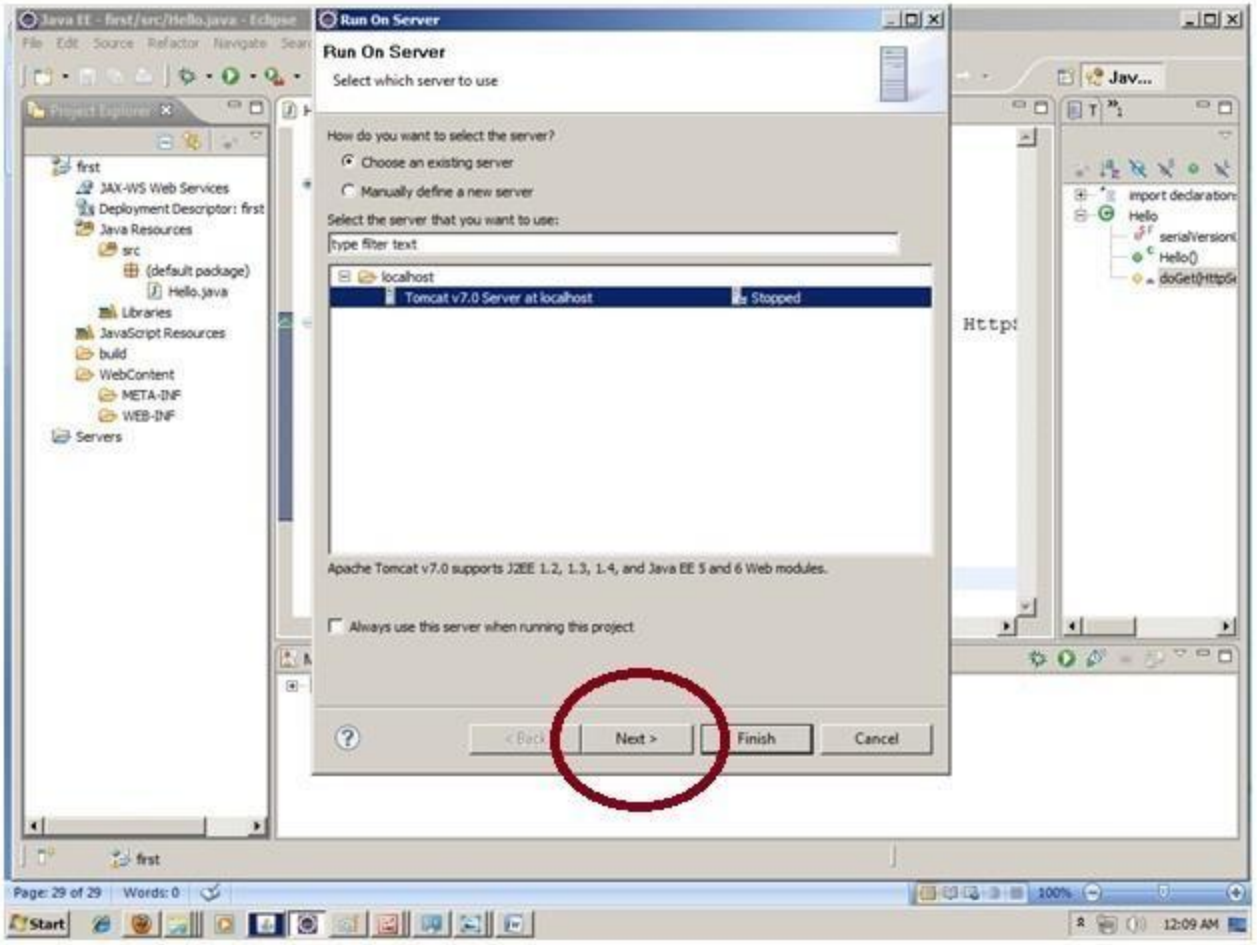
- Create a Dynamic web project
- create a servlet
- add servlet-api.jar file
- Run the servlet

Run the servlet

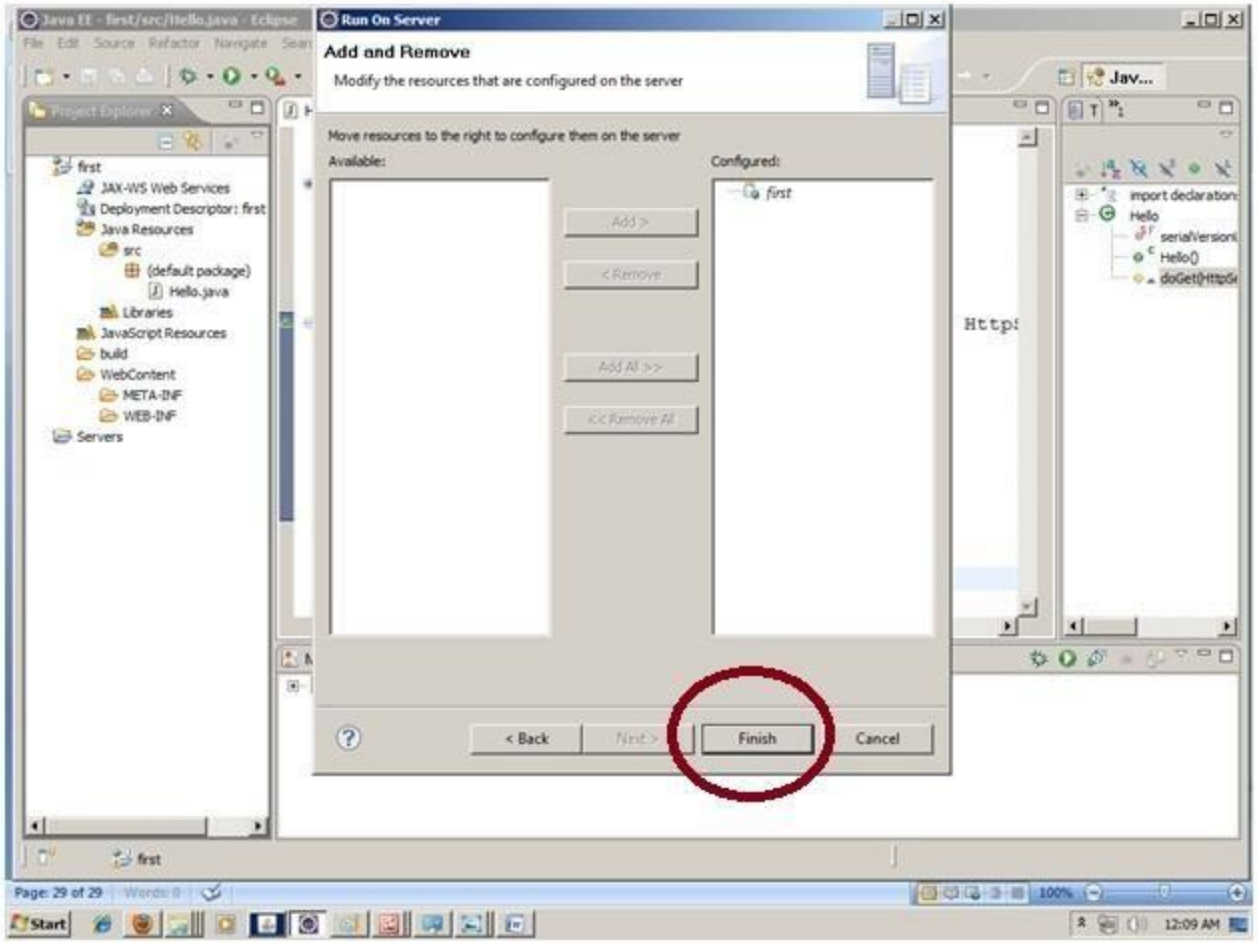
Right click on project name -> click Run As -> Run on Server



Run the servlet



Run the servlet



Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

Example 1-

To Print Hello World directly

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
   public void init() throws ServletException {  }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException,  IOException {
      response.setContentType("text/html");
        PrintWriter out = response.getWriter();
       out.println("<h1> Hello World </h1>");
   }
   public void destroy() { }
}
```

Example 2-

To Print Hello World using init method

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
   private String message;
   public void init() throws ServletException
        { message = "Hello World"; }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType("text/html");
         PrintWriter out = response.getWriter();
          out.println("<h1>" + message + "</h1>");   }
   public void destroy() { }
}
```

Reading Form Data using Servlet

- **getParameter()** – You call request.getParameter() method to get the value of a form parameter.
- **getParameterValues()** – Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames()** – Call this method if you want a complete list of all parameters in the current request.

Example 3-

To read data from HTML file and print that..

- It requires 2 files:
  - 1. HTML (s1.html)File
  - 2. Servlet (s2.java) File
- First Run HTML file and after clicking on submit button it will run servel(.java) file.

Example 3-

To read data from HTML file and print that..

- **S1.html (html code)**

```
<html>
<form method="post" action="s2">
Enter Your Name
<input type=text name="t1">
<br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Example 3-

To read data from HTML file and print that..

- **S2.java (Servlet Code)**

```
import java.io.*;
import javax.servlet.*;
public class s2 extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
 {
response.setContentType("text/html");
String a= request.getParameter("t1");
PrintWriter out= response.getWriter();
out.print("<br>Your Name is: "+a);
 }
}
```

Example 4 – (HTML code)

To read data from HTML checkbox values print that

- <!DOCTYPE html>
- <html>
- <body>
- <form method="post" action=s1>
- Hobbies
- <input type="checkbox" name="t1" value="Cricket"> Cricket
- <input type="checkbox" name="t1" value="Music">Music
- <input type="checkbox" name="t1" value="Dance">Dance
- <input type="submit" value="submit">
- </form>
- </body>
- </html>

Name of Servlet File

Example 4 – (servlet code)

To read data from HTML checkbox values print that

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out=response.getWriter();
String[] values=request.getParameterValues("t1");

        for(int i=0;i<values.length;i++)
       {
          out.println("<li>"+values[i]+"</li>");
       }
}
}
```

Example 5 – (HTML code)

To print complete list of all parameters

- <!DOCTYPE html>
- <html>
- <body>
- <form method="post" action=s1>
- Hobbies
- Enter Name <input type="text" name="t1" >
- Enter Address <input type="text" name="t2" >
- Enter Class <input type="text" name="t3" >
- <input type="submit" value="submit">
- </form>
- </body>
- </html>

Example 5 – (Servlet code)

To print complete list of all parameters

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
{
  response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   Enumeration e = request.getParameterNames();
   while(e.hasMoreElements())
   {
     Object obj = e.nextElement();
     out.println((String) obj+ "<br>");
     }
}
```

Out Line

- Servlet Introduction
- Servlet Lifecycle
- Servlet Architecture
- Software Requirement to run Servlet
- Tomcat Introduction
- Tomcat Configuration in Eclipse
- Steps to run servlets in Eclipse
- Sample code
- Session Management

Session Tracking (Management)

- **Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.
- Http protocol is a stateless so we need to maintain state using session tracking techniques.
- Each time user requests to the server, server treats the request as the new request.
- So we need to maintain the state of an user to recognize to particular user.

- **Why use Session Tracking?**
- **To recognize the user** It is used to recognize the particular user.

Session Tracking Techniques

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Session Tracking- Using Cookies

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- **How Cookie works**
  - In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, **we recognize the user as the old user**.

Session Tracking- Using Cookies

- **Types of Cookie**
  - Non-persistent cookie
  - Persistent cookie
- **Non-persistent cookie**
  - It is **valid for single session** only. It is removed each time when user closes the browser.
- **Persistent cookie**
  - It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Session Tracking- Using Cookies

- **Advantage of Cookies**
  - Simplest technique of maintaining the state.
  - Cookies are maintained at client side.
- **Disadvantage of Cookies**
  - It will not work if cookie is disabled from the browser.
  - Only textual information can be set in Cookie object.

Session Tracking- Using Cookies

Useful Methods of Cookie class

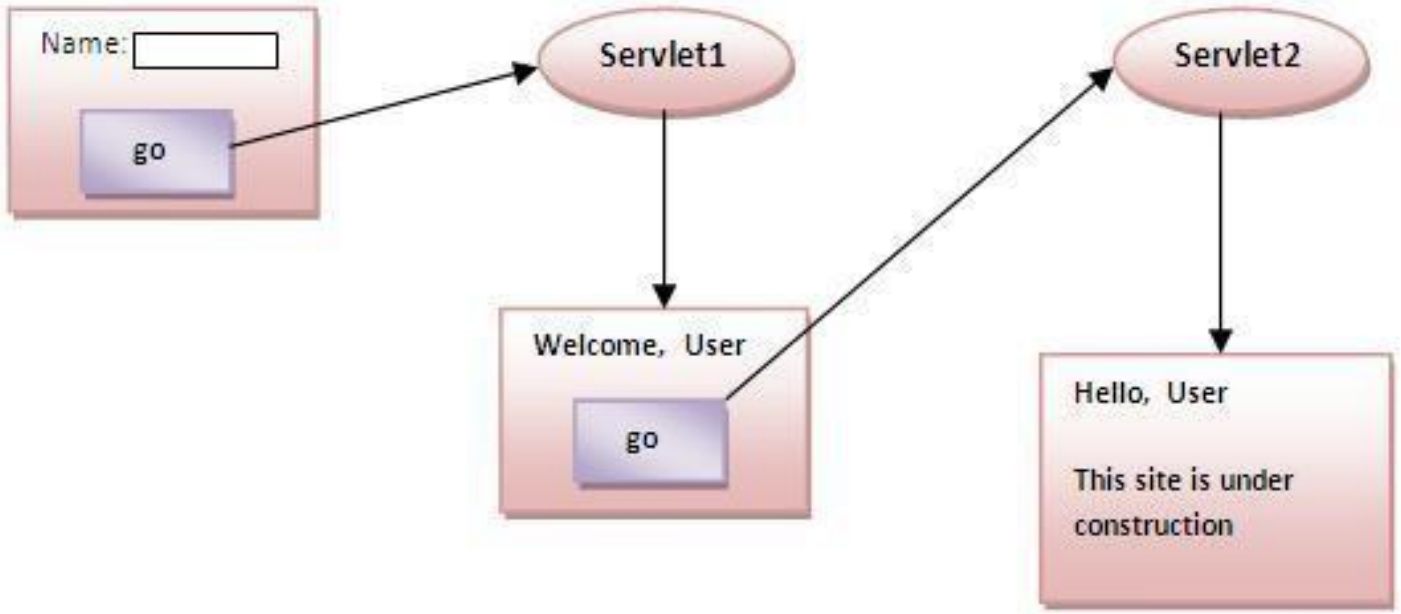
Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Session Tracking- Using Cookies

- **How to create Cookie?**
  - Cookie ck=new Cookie("user","sonu");//creating cookie object
  - response.addCookie(ck);//adding cookie in the response
- **How to delete Cookie?**
  - Cookie ck=new Cookie("user","");//deleting value of cookie
  - ck.setMaxAge(0);//changing the maximum age to 0 seconds
  - response.addCookie(ck);//adding cookie in the response
- **How to get Cookies?**
  - Cookie ck[]=request.getCookies();
  - for(int i=0;i<ck.length;i++){
  - out.print("
"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
  - }

Session Tracking- Using Cookies

Simple example of Servlet Cookies



Session Tracking- Using Cookies

Simple example of Servlet Cookies

- **index.html**

```
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
```

Session Tracking- Using Cookies

Simple example of Servlet Cookies

Servlet1.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
  public class FirstServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response){
    try{
      response.setContentType("text/html");
    PrintWriter out  =  response.getWriter();
    String n=request.getParameter("userName");
    out.print("Welcome "+n);

    Cookie ck=new Cookie("uname",n);//creating cookie object
    response.addCookie(ck);//adding cookie in the response

    //creating submit button
    out.print("<form action='servlet2'>");
    out.print("<input type='submit' value='go'>");
    out.print("</form>");
     out.close();
   }catch(Exception e){System.out.println(e);}
  }
}
```

Session Tracking- Using Cookies

Simple example of Servlet Cookies

Servlet2.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SecondServlet extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response){
    try{

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    Cookie ck[]=request.getCookies();
    out.print("Hello "+ck[0].getValue());

    out.close();
    }catch(Exception e){System.out.println(e);}
    }
  }
```

Session Tracking Techniques

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Session Tracking- Hidden Form Fields

- In case of Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.
- In such case, we store the information in the hidden field and get it from another servlet.

- **Synax**
- <input type="hidden" name="uname" value="Bhavana">

- Here, uname is the hidden field name and Bhavana is the hidden field value.

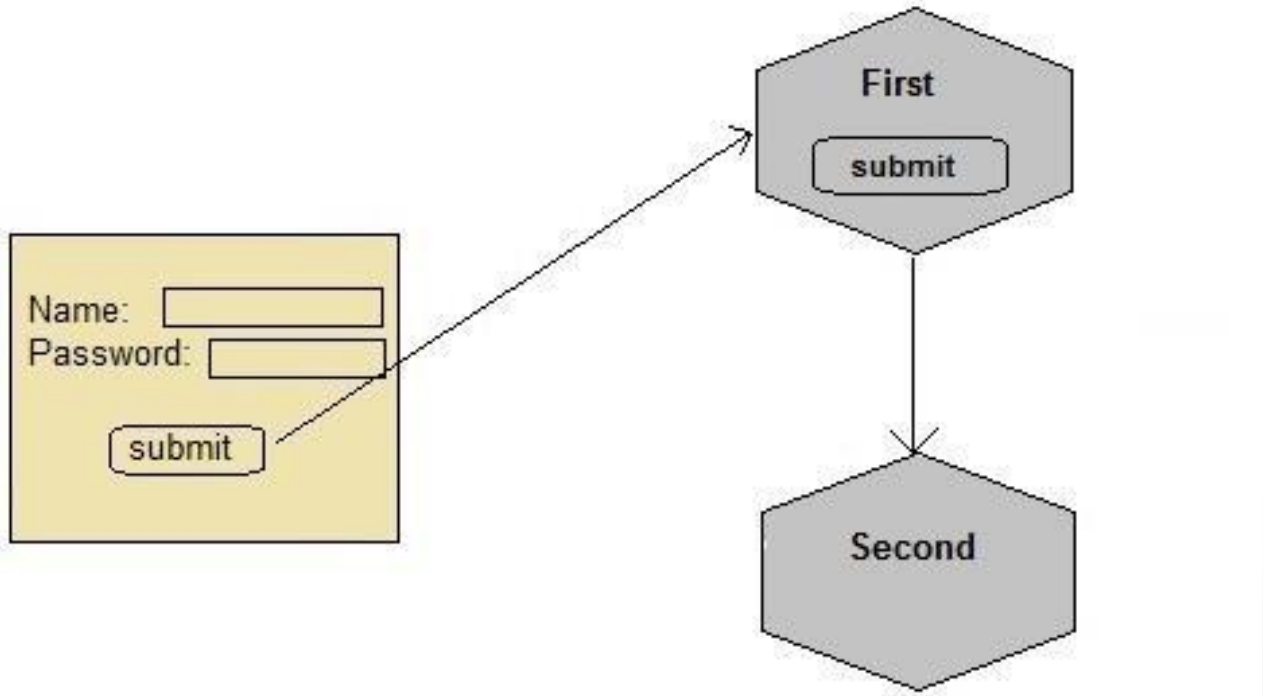
Session Tracking- Hidden Form Fields

- **Advantage of Hidden Form Field**
  - It will always work whether cookie is disabled or not.

- **Disadvantage of Hidden Form Field:**
  - It is maintained at server side.
  - Extra form submission is required on each pages.
  - Only textual information can be used.

Session Tracking- Hidden Form Fields

Example of passing username



Session Tracking- Hidden Form Fields

Example of passing username

- **index.html**

```
<form method="post" action="First">
Name:<input type="text" name="user" /><br/>
Password:<input type="text" name="pass" ><br/>
<input type="submit" value="submit">
</form>
```

Session Tracking- Hidden Form Fields

Example of passing username

- **First.java**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class First extends HttpServlet {
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
       String user = request.getParameter("user");
      //creating a new hidden form field
      out.println("<form action='Second'>");
      out.println("<input type='hidden' name='user' value='"+user+"'>");
      out.println("<input type='submit' value='submit' >");
      out.println("</form>");
   }
}
```

Session Tracking- Hidden Form Fields

Example of passing username

- **Second.java**

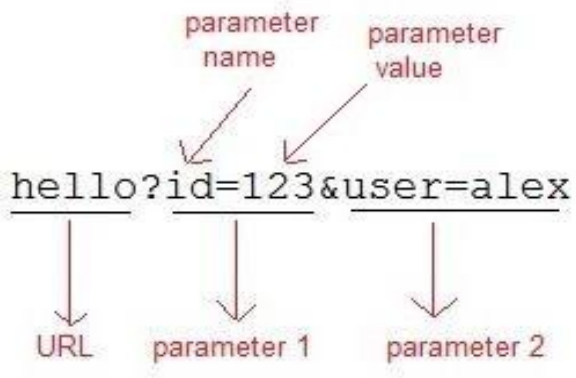
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Second extends HttpServlet {
 protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        //getting parameter from the hidden field
        String user = request.getParameter("user");
        out.println("Welcome "+user);
    }
}
```

Session Tracking Techniques

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Session Tracking- URL Rewriting

- If the client has disabled cookies in the browser then session management using cookie wont work.
- In that case **URL Rewriting** can be used as a backup. **URL rewriting** will always work.
- In URL rewriting, a token(parameter) is added at the end of the URL.
- The token consist of name/value pair separated by an equal(=) sign.
- **For Example:**



Session Tracking- URL Rewriting

- When the User clicks on the URL having parameters, the request goes to the **Web Container** with extra bit of information at the end of URL.
- The **Web Container** will fetch the extra part of the requested URL and use it for session management.
- The getParameter() method is used to get the parameter value at the server side.
- **Advantage of URL Rewriting**
  - It will always work whether cookie is disabled or not (browser independent).
  - Extra form submission is not required on each pages.
- **Disadvantage of URL Rewriting**
  - It will work only with links.
  - It can send only textual information.

Session Tracking- URL Rewriting Example

- **index.html**

- <form method="post" action="validate">
- Name:<input type="text" name="user" />
- Password:<input type="text" name="pass" >
- <input type="submit" value="submit">
- </form>

Session Tracking- URL Rewriting Example

- **Validate.java**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet extends HttpServlet {
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      String name = request.getParameter("user");
      String pass = request.getParameter("pass");
       if(pass.equals("1234"))       {
         response.sendRedirect("First?user_name="+name+"");
      }
   }
}
```

Session Tracking- URL Rewriting Example

- **First.java**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class First extends HttpServlet {
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      String user = request.getParameter("user_name");
      out.println("Welcome "+user);
   }
}
```

Session Tracking Techniques

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Session Tracking- HttpSession

- The servlet container uses this interface to create a session between an HTTP client and an HTTP server.
- The session persists for a specified time period, across more than one connection or page request from the user.

- You would get HttpSession object by calling the public method **getSession()** of HttpServletRequest, as below –

```
HttpSession session = request.getSession();
```

Session Tracking- HttpSession

- **How to get the HttpSession object ?**
  - **public HttpSession getSession():**Returns the current session associated with this request, or if the request does not have a session, creates one.
- **Commonly used methods of HttpSession interface**
  - **public String getId():**Returns unique identifier value.
  - **public long getCreationTime():**Returns the time when this session was created.
  - **public long getLastAccessedTime():**Returns the last time the client sent a request associated with this session.
  - **public void invalidate():**Invalidates this session then unbinds any objects bound to it.

Session Tracking- HttpSession

Example Hit Count

- **HTML File**

```
<h3>Hit Count Example with HttpSession</h3>
  <form method="get" action="HitCount">
   Click for Hit Count
    <input type="submit" value="GET HITS">
   </form>
</body>
```

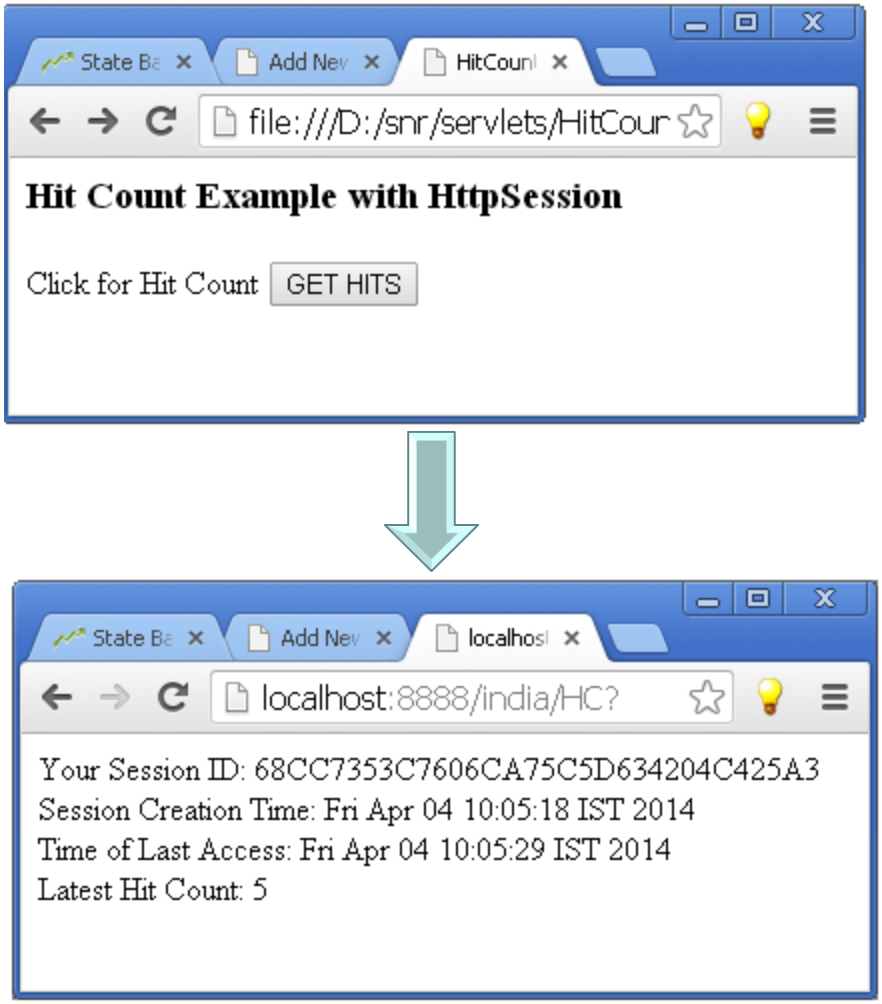
Session Tracking- HttpSession

Example Hit Count

```
public class HitCount extends HttpServlet
{   public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException,  IOException    {
   res.setContentType("text/html") ;
   PrintWriter out = res.getWriter( );
   HttpSession session = req.getSession();
    Integer hitNumber = (Integer) session.getAttribute("rama");

   if  (hitNumber == null)     {     hitNumber = new Integer(1);    }
   else    {   hitNumber = new Integer(hitNumber.intValue()+1) ; }
    session.setAttribute("rama", hitNumber);            // storing the value with session object

    out.println("Your Session ID: " + session.getId()); // never changes in the whole session
  out.println("<br>Session Creation Time: " + new Date(session.getCreationTime()));
  out.println("<br>Time of Last Access: " + new Date(session.getLastAccessedTime()));
  out.println("<br>Latest Hit Count: " + hitNumber);      // increments by 1 for every hit
```

Session Tracking- HttpSession

Example Hit Count output



Servlet- Data Storage

Almost all web applications (servlets or related dynamic web server software) store and retrieve data –

- Typical web app uses a data base management system (DBMS),
- Another option is to use the file system

Servlet- Concurrency

One common web application problem: concurrency

Concurrency means multiple computations are happening at the same time.

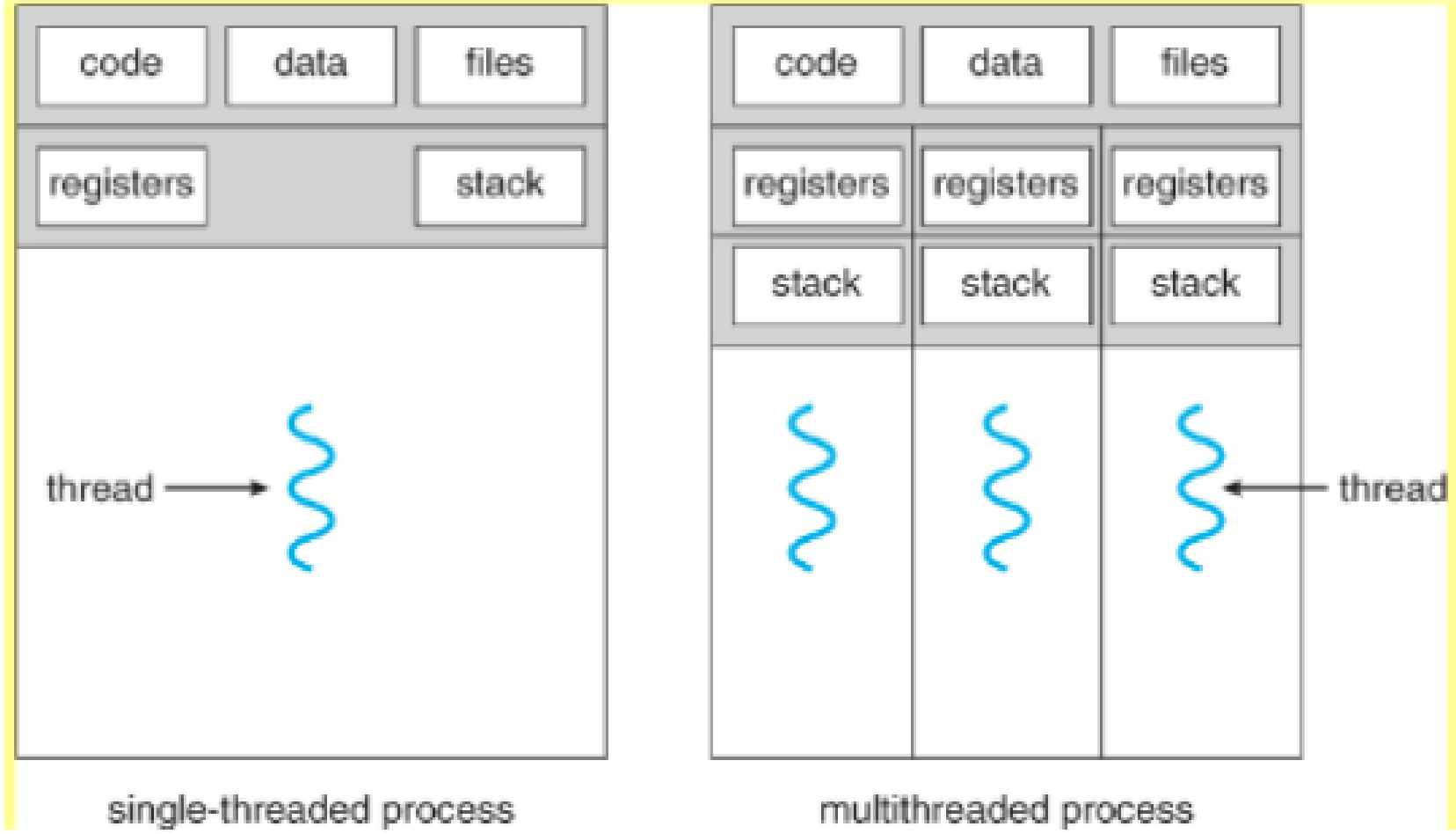
- Ex: Web browser loading several images at the same time, or web browser can respond to your mouse clicks while it is still downloading information from a web server are examples of concurrent processing in action on the client side.

- On a server side, multiple requests to the same servlet may be executed at the same time. So concurrency container or web server is multithreaded.

Cont…

- A thread is a single execution process.
- It is a basic unit of CPU utilization, consisting of own program counter, a stack, and a set of registers.
- A program is multithreaded when multiple threads execute a single instance of a program.

Cont…



Single-threaded and multithreaded processes

Threading Issues

- Two Threads running in HelloCounter Concurrently

- The initial value of visits Is assumed to be 17

Thread Synchronization

- A servlet must be capable of serving more than one client at a time.
- If several clients issue requests at the same time, methods will serve each client in a different thread.
- service(), doGet(), and do Post() can handle many concurrent clients.
- It uses lock mechanism to synchronize the threads.

Database (MySQL)

- MySQL is a database system used for developing web-based software applications.
- MySQL used for both small and large applications.
- MySQL is a relational database management system (RDBMS).
- MySQL is fast, reliable, and flexible and easy to use.
- MySQL supports standard SQL (Structured Query Language).
- MySQL is free to download and use.
- MySQL was developed by Michael Widenius and David Axmark in 1994.
- MySQL is presently developed, distributed, and supported by Oracle Corporation.
- MySQL Written in C, C++.

Cont…

- MySQL server design is multi-layered with independent modules.
- MySQL is fully multithreaded by using kernel threads. It can handle multiple CPUs if they are available.
- MySQL provides transactional and non-transactional storage engines.
- MySQL has a high-speed thread-based memory allocation system.
- MySQL supports in-memory heap table.
- MySQL Handles large databases.
- MySQL Server works in client/server or embedded systems.
- MySQL Works on many different platforms.
-

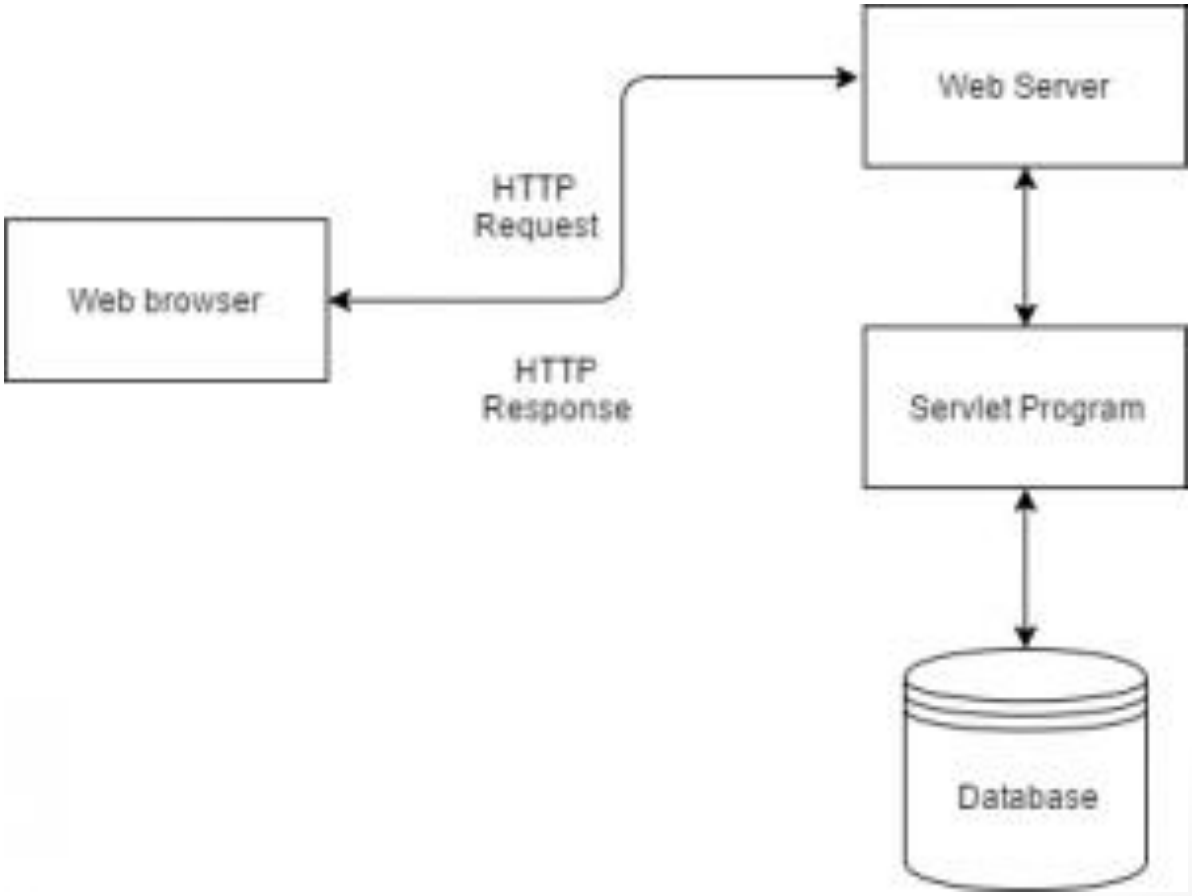
Java Servlet

- Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.

- Properties of Servlets are as follows:
  - Servlets work on the server-side.
  - Servlets are capable of handling complex requests obtained from the webserver.

Cont…

Servlet Architecture is can be depicted from the image itself as provided below as follows:



Cont…

- Execution of Servlets basically involves six basic steps:
- The clients send the request to the webserver.
- The web server receives the request.
- The web server passes the request to the corresponding servlet.
- The servlet processes the request and generates the response in the form of output.
- The servlet sends the response back to the webserver.
- The web server sends the response back to the client and the client browser displays it on the screen.

Cont…

To start with interfacing Java Servlet Program with JDBC Connection:

- Proper JDBC Environment should set-up along with database creation.
- To do so, download the mysql-connector.jar file from the internet,
- As it is downloaded, move the jar file to the apache-tomcat server folder,
- Place the file in **lib** folder present in the apache-tomcat directory.
- **To start with the basic concept of interfacing:**
  - **Step 1: Creation of Database and Table in MySQL**
  - **Step 2: Implementation of required Web-pages**
  - **Step 3: Creation of Java Servlet program with JDBC Connection**

    To create a JDBC Connection steps are:
    - Import all the packages
    - Register the JDBC Driver
    - Open a connection
    - Execute the query, and retrieve the result
    - Clean up the JDBC Environment
  - **Step 4: To use this class method, create an object in Java Servlet program**
  - **Step 5: Get the data from the HTML file**

AJAX

What is AJAX?

- AJAX = Asynchronous JavaScript And XML.
- AJAX is not a programming language.
- AJAX just uses a combination of: A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX - Technologies

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

JavaScript:

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page
- Glue for the whole AJAX operation.

DOM:

- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

CSS:

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

XMLHttpRequest:

- JavaScript object that performs asynchronous interaction with the server.

AJAX – Real Time Examples

Here is a list of some famous web applications that make use of AJAX.

Google Maps :

- A user can drag an entire map by using the mouse, rather than clicking on a button.

Google Suggest :

- As you type, Google offers suggestions. Use the arrow keys to navigate the results.

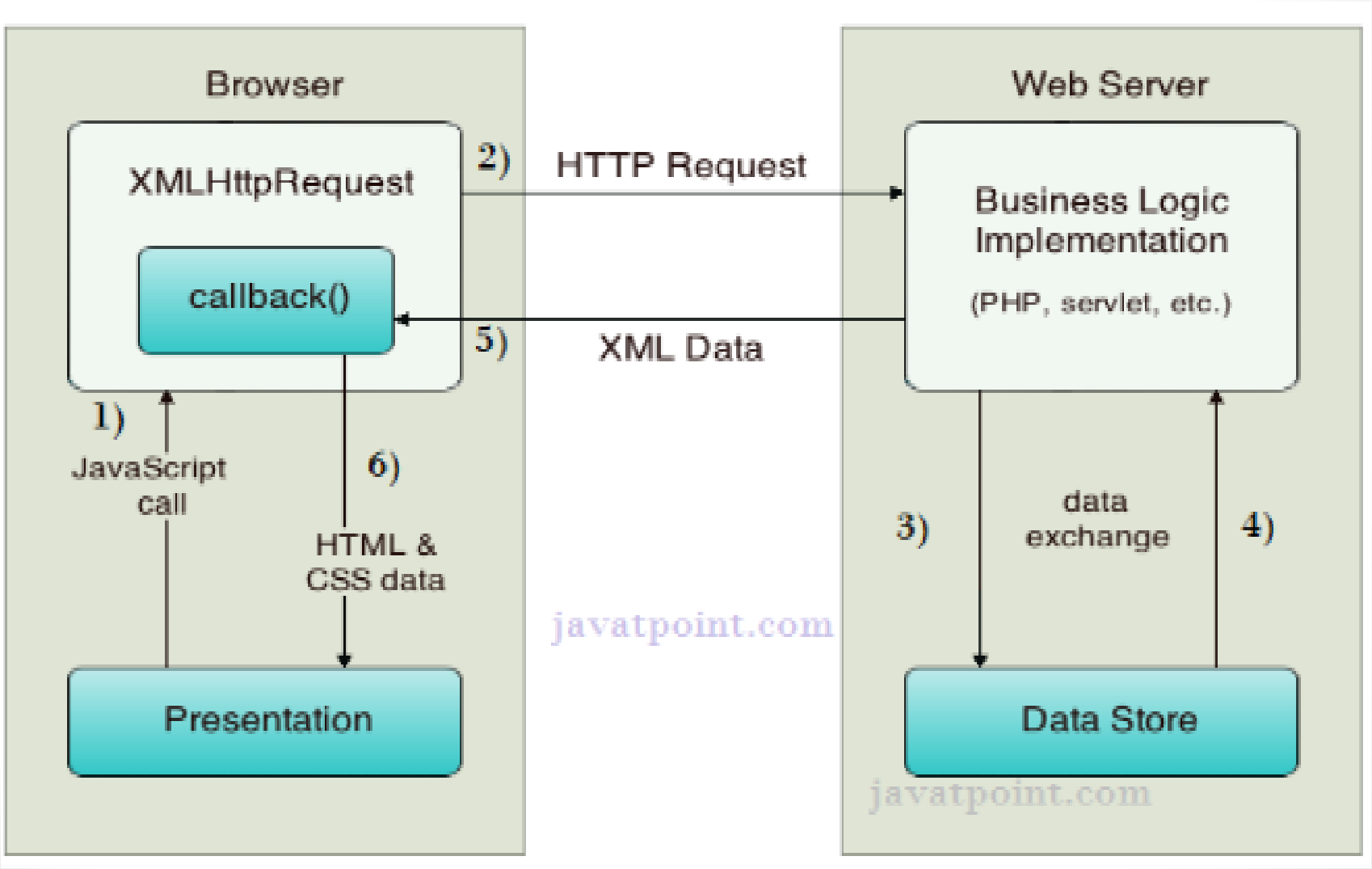
Gmail :

- Gmail is a webmail built on the idea that emails can be more intuitive, efficient, and useful.

Yahoo Maps (new) :

- Now it's even easier and more fun to get where you're going!

How AJAX Works



AJAX Processing Steps

Steps of AJAX Operation:

- A client event occurs.
- An XMLHttpRequest object is created.
- The XMLHttpRequest object is configured.
- The XMLHttpRequest object makes an asynchronous request to the Webserver. The Webserver returns the result containing XML document.
- The XMLHttpRequest object calls the callback() function and processes the result. The HTML DOM is updated.

Ajax Example- table.html

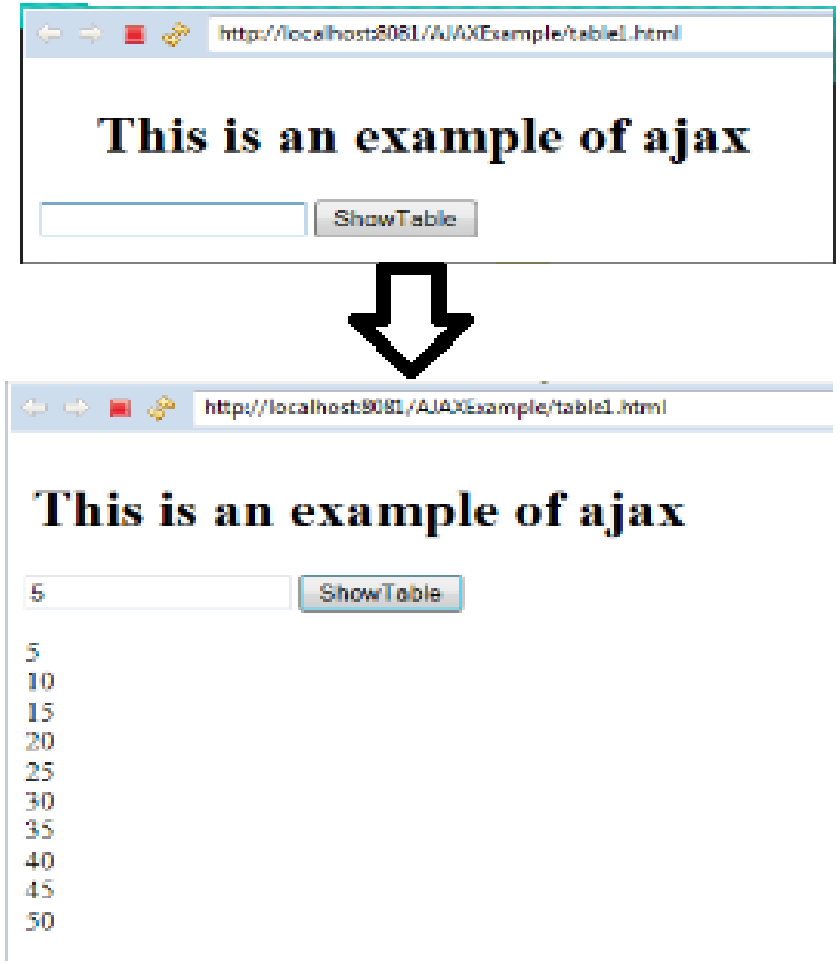
```
<html>
<head>
<script>
var request;

function sendInfo() {
var v=document.f1.t1.value;
var url="index.jsp?val="+v;

if(window.XMLHttpRequest){
   request=new XMLHttpRequest();
  }

request.onreadystatechange=getInfo;
request.open("GET",url,true);
request.send();
}
```

```
function
if  request.readyState==4) {
var val=request.responseText;
document.getElementById('amit').innerHTML=val;
   }
 }
 </script>
</head>

<body>
<h1>This is an example of ajax</h1>
<form name="f1">
<input type="text" name="t1">
<input type="button" value="ShowTable"
onClick="sendInfo()">
</form>
<span id="amit"> </span>
 </body>
</html>
```

AJAX Example- index.jsp

```
<%
int n=Integer.parseInt(request.getParameter("val"));
 for(int i=1;i<=10;i++)
 out.print(i*n+"<br>");
%>
```

AJAX Example Output



XML – XML is a data format, not a programming language.

- XML stands for extensible Markup language.
- This scripting language is similar to HTML. That means, this scripting language contains various tags. But these tags are not predefined tags, in fact user can define his own tags.
- Thus HTML is designed for representation of data on the web page whereas the XML is designed for transport or to store data.
- XML is a W3C Recommendation

Uses of XML

- XML is used to display the meta contents i.e. XML describes the content of the document.
- XML is useful in exchanging data between the applications.
- The data can be extracted from database and can be used in more than one application. Different applications can perform different tasks on this data.

Advantages of XML

- XML document is human readable and we can edit any XML document in simple text editors.
- The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl
- Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner
- XML files are independent of an operating system.

XML Key components

1) Element Tag

A tag is a markup (language) construct that begins with < and ends with >. Tags come in three flavors:
- start tag, such as <section>
- End tag, such as </section>
- Empty element tag, such as <line-break />

2) Element

- In XML, the basic entity is element. The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.
- Syntax of writing any element for opening tag is <element name>
- Syntax of writing any element for closing tag is </element name>

eg: — <student> I am learning Web Technology </student>

- Every XML must have a single root element
- The other elements are called the container elements.
- There can be one or more elements inside the container elements such elements are called the child elements
- An empty tag can be defined by putting a / (forward slash) before closing bracket

eg :-

```
<Person>
    <Personal_Info>
        <Name> My name is Anu </Name>
        <City> I live in Nashik </City>
    </Personal_Info>

    <Hobby>
        <First> I like reading </First>
        <Second> I like programming </Second>
    </Hobby>
</Person>
```

Attribute

The attributes must be enclosed within double quotes or a single quote

- An attribute is a markup construct consisting of a name-value pair that exists within a start-tag or empty element tag.

eg :- `<Person flag = "true">`

`</Person>`

flag = name.
true = value.

eg :-

`<step number = "3" />`

`</step>`

~~Flag~~ = number = name
3 = value

eg :- `<img src = "rose.jpg" />`

rose.jpg = ~~name~~
src = name.

PAGE NO.....

DATE...../...../.....

XML Declaration

XML documents may begin with an XML declaration that describes some information about themselves.

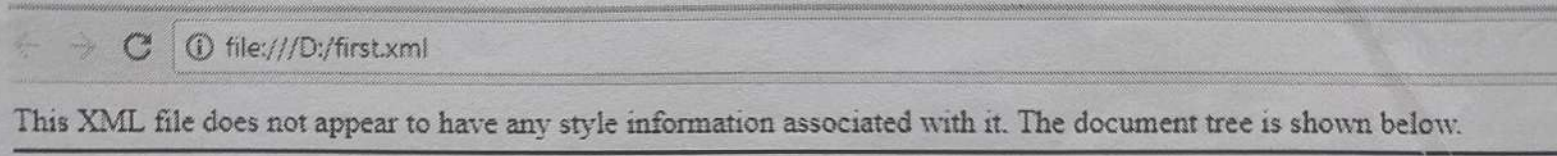
eg:- <?xml version = "1.0" encoding = "UTF-8"?>

Comments in XML

Syntax is same as used for HTML:

<!-- ---- -->

```
<?xml version="1.0" encoding="UTF-8"?>
<contact_info>
        <name>Ajinkya</name>
        <company>Infosys</company>
        <phone> 123456789</phone>
</contact_info>
```



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<contact_info>
    <name>Ajinkya</name>
    <company>Infosys</company>
    <phone>123456789</phone>
  </contact_info>
```

DATE. ...../...../.....

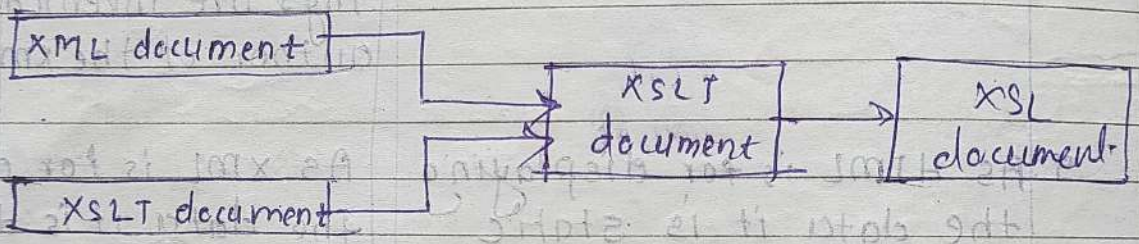
Difference between HTML and XML

	HTML	XML
1.	HTML stands for Hypertext Markup Language	XML stands for eXtensible Markup Language
2.	HTML is designed to display data with the focus on look & feel of data.	XML is used to transport and store data, with focus on what data is
3.	HTML is case insensitive	XML is case sensitive.
4.	HTML has predefined tags	XML has custom tags can be defined and the tags are invented by the author of the XML document
5.	As HTML is for displaying the data it is static	As XML is for carrying the data it is dynamic
6.	It can not preserve white space	It can preserve the white space.

Transforming XML into XSLT

- XSLT (eXtensible Stylesheet Language Transformation) is the standard style sheet language for XML.
- XSLT is better than CSS.
- XSLT helps to add / remove elements and attributes to or from the output file.
- Also the elements can be rearranged and sort, execute tests and decide which elements to hide or display.
- XSLT used to transform xml data from one format into other.

How XSLT Works



- XSLT processor take two input documents one is XML document and another is XSLT document.
- The XSLT document is nothing but a program and xml document is nothing but the input data, thus this program works on the XML input data.
- Then some or whole part of the XML document is selected, modified & merged with XSLT program document in order to produce another document.
- This newly produced document is provided as input to the XSLT processor which in turn produce another document called the XSL document.
- The XSL document is used along with the

application so that particular application can be displayed on the web browser in some desired manner.

- The XSLT document makes use of templates using which particular code can be described in xml document.
- xml document and then particular code in XSLT is executed when the match is found.
- XSLT processor processes the xml document sequentially by reading each line one by one.
- The XSLT model can be described as template driven or data driven model

XSL Elements

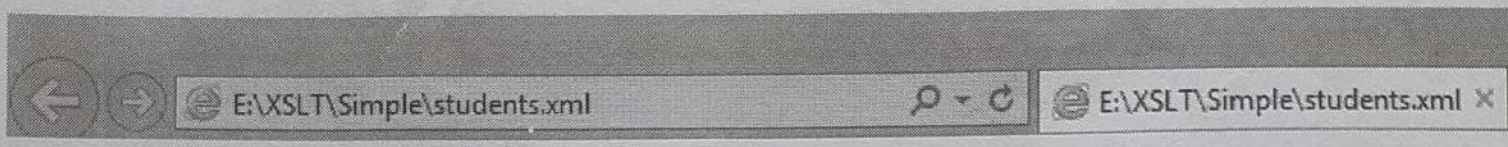
- The <xsl:template> is for building the templates
- The match attribute is associated with the template element. The match attribute can also be used to define a template for the entire xml document.
- The match="/" defines the whole document.

students.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="students.xsl"?>
<class>
        <student>
        <rno>1</rno>
        <name>Kunal</name>
        <address>Nashik</address>
        <marks>85</marks>
        </student>

        <student>
        <rno>2</rno>
        <name>Monali</name>
        <address>Pune</address>
        <marks>70</marks>
        </student>

        <student>
        <rno>3</rno>
        <name>Sonali</name>
        <address>Mumbai</address>
        <marks>90</marks>
        </student>
</class>
```

students.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="/">

        <html>
                <body>
                <h3>Students</h3>
                <table border="2">
                <tr>
                <th>Roll_no</th>
                <th>Name</th>
                <th>Address</th>
                <th>Marks</th>
                </tr>

                <tr>
                <td>**</td>
                <td>**</td>
                <td>**</td>
                <td>**</td>
                </tr>

                </table>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```



Students

Roll_no	Name	Address	Marks
**	**	**	**

students.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="/">

        <html>
                <body>
                <h3>Students</h3>
                <table border="2">
                <tr>
                <th>Roll_no</th>
                <th>Name</th>
                <th>Address</th>
                <th>Marks</th>
                </tr>

                <tr>
                <td><xsl:value-of select="class/student/rno"/></td>
                <td><xsl:value-of select="class/student/name"/></td>
                <td><xsl:value-of select="class/student/address"/></td>
                <td><xsl:value-of select="class/student/marks"/></td>
                </tr>

                </table>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

The value-of is used to extract the value of xml element and add it to output stream of XSL transformation

Students

Roll_no	Name	Address	Marks
1	Kunal	Nashik	85

for-each

20

students.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="/">

        <html>
                <body>
                <h3>Students</h3>
                <table border="2">
                <tr>
                <th>Roll_no</th>
                <th>Name</th>
                <th>Address</th>
                <th>Marks</th>
                </tr>

                <xsl:for-each select="class/student">

                <tr>
                <td><xsl:value-of select="rno"/></td>
                <td><xsl:value-of select="name"/></td>
                <td><xsl:value-of select="address"/></td>
                <td><xsl:value-of select="marks"/></td>
                </tr>
                </xsl:for-each>

                </table>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

The for-each element allows to traverse through each element of the node

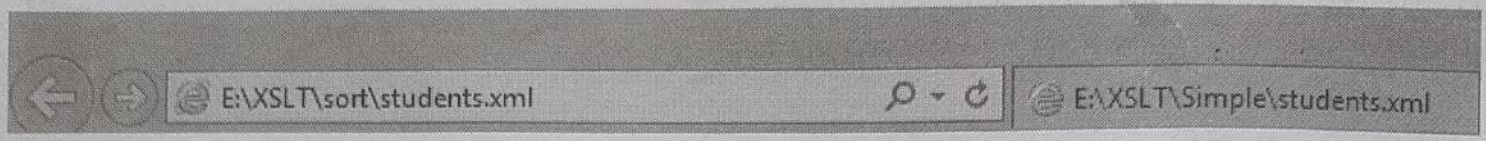
Students

Roll_no	Name	Address	Marks
1	Kunal	Nashik	85
2	Monali	Pune	70
3	Sonali	Mumbai	90

students.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="/">

        <html>
                <body>
                <h3>Students</h3>
                <table border="2">
                <tr>
                <th>Roll_no</th>
                <th>Name</th>
                <th>Address</th>
                <th>Marks</th>
                </tr>

                <xsl:for-each select="class/student">
                <xsl:sort select="marks"/>

                <tr>
                <td><xsl:value-of select="rno"/></td>
                <td><xsl:value-of select="name"/></td>
                <td><xsl:value-of select="address"/></td>
                <td><xsl:value-of select="marks"/></td>
                </tr>
                </xsl:for-each>

                </table>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

Sorted data can be displayed using <xsl:sort>



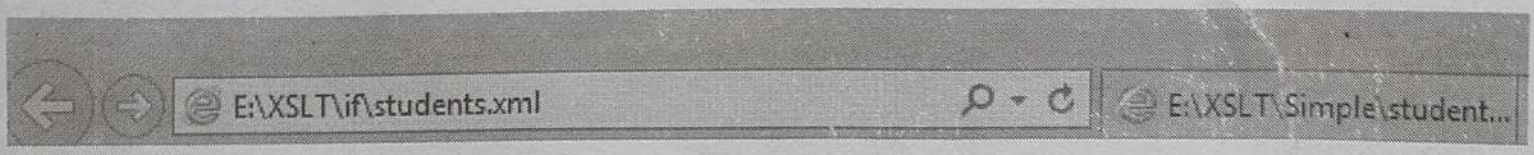
Students

Roll_no	Name	Address	Marks
2	Monali	Pune	70
1	Kunal	Nashik	85
3	Sonali	Mumbai	90

students.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="/">

        <html>
                <body>
                <h3>Students</h3>
                <table border="2">
                <tr>
                <th>Roll_no</th>
                <th>Name</th>
                <th>Address</th>
                <th>Marks</th>
                </tr>

                <xsl:for-each select="class/student">
                <xsl:if test="marks>77">

                <tr>
                <td><xsl:value-of select="rno"/></td>
                <td><xsl:value-of select="name"/></td>
                <td><xsl:value-of select="address"/></td>
                <td><xsl:value-of select="marks"/></td>
                </tr>

                </xsl:if>
                </xsl:for-each>

                </table>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

It is used to select specific record based on some condition



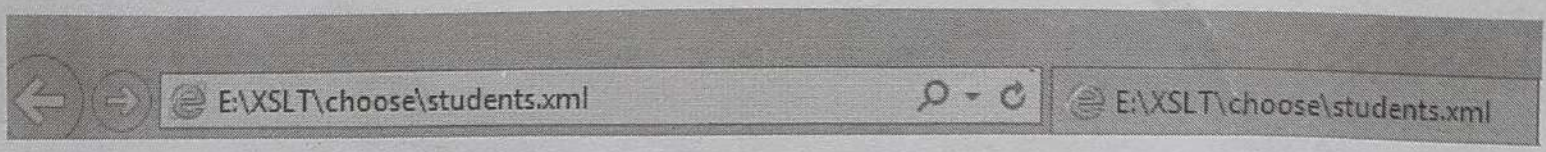
Students

Roll_no	Name	Address	Marks
1	Kunal	Nashik	85
3	Sonali	Mumbai	90

students.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="/">

        <html>
                <body>
                <h3>Students</h3>
                <table border="2">
                <tr>

                <th>Name</th>

                <th>Marks</th>
                </tr>

                <xsl:for-each select="class/student">


                <tr>

                <td><xsl:value-of select="name"/></td>
                <xsl:choose>
                <xsl:when test="marks &lt; 77">


                <td><xsl:value-of select="marks"/></td>
                </xsl:when>
                <xsl:otherwise>
                <td><xsl:value-of select="marks"/></td>

                </xsl:otherwise>

                </xsl:choose>

                </tr>


                </xsl:for-each>

                </table>
                </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

It is used along with <xsl:when> & <xsl:otherwise> to express multiple conditional tests.



Students

Name	Marks
Kunal	85
Monali	70
Sonali	90

DTD

- The document type definition is used to define the basic building block of any xml document
- Using DTD we can specify the various elements types, attributes and their relationship with one another.
- Basically DTD is used to specify the set of rules for structuring data in any XML file.

eg:- If we want to put some information about some students in XML file then, generally we use tag student followed by his/her name, address, standard and marks. That means we are actually specifying the manner by which the information should be arranged in the XML file. and for this purpose Document Type Definition is used.

Building blocks of XML are

1) Elements

The basic entity is element. The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

2) Attribute

The attributes are generally used to specify the values of element. These are specified within the double quotes of tag.

eg :- <flag type = "True">

3) CDATA

CDATA stands for character data. This character data will be parsed by the parser.

4) PCDATA

It stands for Parsed Character Data (i.e. text). Any parsable character data should not contain the markup characters.. The markup characters are < or > or &. If we want to use less than, greater than or ampersand characters then make use of <, > or &

There are two ways by which DTD can be defined.

1) Internal DTD

Consider following xml document ED emot.xml

We are defining DTD internally

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name, address, std, marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
]>
<student>
<name> Anand </name>
<address> Pune </address>
<std> Second </std>
<marks> 70 percent </marks>
</student>
```

O/P

This xml file does not appear to have any style information associated with it. The document tree is shown below.

```
<student>
<name> Anand </name>
<address> Pune </address>
<std> second </std>
<marks> 70 percent </marks>
</student>
```

External DTD

In this type, an external DTD file is created and its name must be specified in the corresponding xml file.

1) Creation of DTD file (student.dtd)
   - open suitable text editor (notepad).

```
<!ELEMENT student (name, address, std, marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
```

Now save this file as student.dtd

2) Creation of XML Document (Demo.xml)

```
<?xml version="1.0"?>
<!DOCTYPE student SYSTEM "student.dtd">
```

The external DTD file is created

<student>
<name> Anand </name>
<address> Pune </address>
<std> second </std>
<marks> 70 percent </marks>
</student>

(3) Using some web browser open the xml document.

O/P

This XML file does not appear to have any style information associated with it. The document tree is shown below

<student>
<name> Anand </name>
<address> Pune </address>
<std> second </std>
<marks> 70 percent </marks>
</student>

Merits of DTD

1) DTD's are used to define the structural components of xml document
2) These are relatively simple and compact
3) DTD's can be defined inline & hence can be embeded directly in the xml document.

Demerits of DTD

1) The DTDs are very basic and hence cannot be much specific for complex documents
2) The language that DTD uses is not an xml document, hence various frameworks used by xml cannot be supported by the DTDs.
3. The DTD cannot define the type of data contained within the xml document. Hence using DTD we cannot specify

Whether the element is numeric or string or etc
4) There are some XML processor which do not understand DTD's
5) The DTDs are not aware of namespace concept

Schema

The XML schemas are used to represent the structure of XML document.

- The purpose of XML schema is to define the building blocks of an XML document. These can be used as an alternative to XML DTD.
- The XML schema language is called as XML Schema definition (XSD) language
- XML schema defines elements, attributes, elements having child elements, order of child elements. It also defines fixed and default values of elements and attributes.
- XML schema also allows the developer to use data types.

How to write a simple schema

We will first write a simple xsd file in which the desired structure of the XML document is defined.

1. This file contains the complex type with four child simple type elements.

XML schema (studentschema.xsd)

```
<?xml version = "1.0" ?>
<xs:schema
xmlns:xs = "http://www.w3.org/2001/XMLSchema">
    <xs:element name = "Student" >
        <xs:complexType>
        <xs:sequence>
            <xs:element name = "name" type = "xs:string" />
            <xs:element name = "address" type = "xs:string" />
            <xs:element name = "std" type = "xs:string" />
            <xs:element name = "marks" type = "xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

xs is qualifier used to identity the schema elements and types.

The document element of schema is xs:schema
The xs:schema is root element. It takes the attributes xmlns:xs which have the value http://www.w3.org/2001/XMLSchema.

- This declaration indicates that document should follow the rules of XML schema.
- The XML schema rules are defined by the W3C recommendation in year 2001.
- xs:element is used to define the xml element. In above case the element student is of complex type who have four child elements: name, address, std and marks. All these elements are of simple type string

PAGE NO.....
DATE...../...../.....

(2) Now develop xml document in which the desired values to the xml elements can be given.

myschema.xml

version & encoding is specified

```
<?xml version="1.0" encoding="UTF-8"?>
<student xmlns-xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentschema.xsd">
<name> Anand </name>
<address> Pune </address>
<std> second </std>
<marks> 70 percent </marks>
</student>
```

indicates that xml document is an instance of xml.

it has come from the namespace

This attribute used in To tie this xml document with the some schema definition

The value that can be passed to this attribute is the name of xsd file "studentschema.xsd"

three are child elements

o/p

This xml file does not appear to have any style information associated with it. The document tree is shown below

```
<student xsi:noNamespaceSchemaLocation="
                                studentschema.xsd">
<name> Anand </name>
<address> Pune </address>
<std> second </std>
<marks> 70 percent </marks>
</student>
```

Data Types

Various data types that can be used to specify the data types of an element are -

String
Date
Numeric
Boolean

String data type

It is used to define the element containing characters, lines, tabs or white spaces.

eg:-
```
<?xml version="1.0 encoding="UTF-8"?>
<xs:schema xmlns:xs:"http://www.w3.org/2001/
                                XMLSchema">
<xs:element name="Student_name" type:"xs:string"/>
</xs:schema>
```

(2) Date

It is used to specify the date. The format of this date is yyyy-mm-dd where yyyy denotes the year, mm denotes the month & dd specifies the day.

eg:-
```
<?
<xs:

<xs:element name="Date_of_birth" type="xs:date"/>
</xs:schema>
```

3] Numeric

If we want to use numeric value for some element then we can use the data types as either decimal or integer

```
<?
{
  <xs:element name = "My_marks" type = "xs:decimal" />
</xs:schema>
```

Using decimal data type we can specify the value that may contain some decimal point. integer data type is used to specify the numeric values that are without any decimal point

4] Boolean

It is used to specify the true or false values.

```
<?
<xs:

  <xs:element name = "Flag" type = "xs:boolean" />
</xs:schema>
```

Advantages

1. The schemas are more specific
2. The schema provide the support for data types
3. The schema is aware of namespaces
4. The XML schema is written in XML itself and has a large number of built in & derived types.
5. The XML schema is W3C recommendation. Hence it is supported by various XML validator & XML processors.

Disadvantages

1. The XML schema is complex to design and hard to learn.
2. The XML document cannot be if the corresponding schema file is absent.
3. Maintaining the schema for large and cumplex operations sometimes slows down the processing of XML document.

JSON

- It stands for JavaScript Object Notation.
- Using JSON we can store & retrieve the data.
- It is extended from JavaScript language, but it can be used by many languages like Python, Ruby, PHP and Java.
- It is text based, lightweight data interchange format.
- It is language independent.
- It is easy to read & write
- It is easy for machines to parse & generate.
- It uses the conventions that are familiar to the languages like C, C++, Java, Javascript, Perl, Python & so on.

Structure of JSON

JSON is built on <u>two structures</u>

1. A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list or associative array.
2. An ordered list of values. In most languages this is realized as an array, vector, list or sequence.

JSON object

- JSON object holds the key-value pair
- Each key is represented as string and value can be of any datatype.
- The key and value are separated by colon.

syntax:-

```
{string : value, ---}
```

eg:- "Age":20

The object is written within the { } curly brackets

JSON object containing value of diff. data types

```
{
"student":{
        "name":"abc",
        "roll-no":10,
        "Indian":true
}
}
```

JSON Nested object

```
{
  "student":{
        "name":"abc",
        "rollno":10,
        "address":{
                "street":"Shivaji Nagar",
                "city":"Nashik",
                "pincode":422005
        }
  }
}
```