Top of Form

**Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).

**Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.

There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

**What is a Servlet?**

Servlet can be described in many ways, depending on the context.

* Servlet is a technology which is used to create a web application.
* Servlet is an API that provides many interfaces and classes including documentation.
* Servlet is an interface that must be implemented for creating any Servlet.
* Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
* Servlet is a web component that is deployed on the server to create a dynamic web page.



**Servlet Interface**

**Servlet interface provides** commonbehaviorto all the servlets.Servlet interface defines methods that all servlets must implement.

Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

**Methods of Servlet interface**

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

**Method Description**

public void init(ServletConfig config) initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.

public void service(ServletRequest request,ServletResponse response) provides response for the incoming request. It is invoked at each request by the web container.

public void destroy() is invoked only once and indicates that servlet is being destroyed.

public ServletConfig getServletConfig() returns the object of ServletConfig.

public String getServletInfo() returns information about servlet such as writer, copyright, version etc.

**Servlet Example by implementing Servlet interface**

*File: First.java*

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
4. **public** **class** First **implements** Servlet{
5. ServletConfig config=**null**;
7. **public** **void** init(ServletConfig config){
8. **this**.config=config;
9. System.out.println("servlet is initialized");
10. }
12. **public** **void** service(ServletRequest req,ServletResponse res)
13. **throws** IOException,ServletException{
15. res.setContentType("text/html");
17. PrintWriter out=res.getWriter();
18. out.print("<html><body>");
19. out.print("<b>hello simple servlet</b>");
20. out.print("</body></html>");
22. }
23. **public** **void** destroy(){System.out.println("servlet is destroyed");}
24. **public** ServletConfig getServletConfig(){**return** config;}
25. **public** String getServletInfo(){**return** "copyright 2007-1010";}
27. }

**GenericServlet class**

**GenericServlet** class implements **Servlet**, **ServletConfig** and **Serializable** interfaces. It provides the implementation of all the methods of these interfaces except the service method.

GenericServlet class can handle any type of request so it is protocol-independent.

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

**Methods of GenericServlet class**

There are many methods in GenericServlet class. They are as follows:

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7. **public ServletContext getServletContext()** returns the object of ServletContext.
8. **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
9. **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
10. **public String getServletName()** returns the name of the servlet object.
11. **public void log(String msg)** writes the given message in the servlet log file.
12. **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

**Servlet Example by inheriting the GenericServlet class**

*File: First.java*

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
4. **public** **class** First **extends** GenericServlet{
5. **public** **void** service(ServletRequest req,ServletResponse res)
6. **throws** IOException,ServletException{
8. res.setContentType("text/html");
10. PrintWriter out=res.getWriter();
11. out.print("<html><body>");
12. out.print("<b>hello generic servlet</b>");
13. out.print("</body></html>");
15. }
16. }

**HttpServlet class**

The HttpServlet class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

**Methods of HttpServlet class**

There are many methods in HttpServlet class. They are as follows:

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.
2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** handles the HEAD request. It is invoked by the web container.
6. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.
7. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
8. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.
9. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.
10. **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

Bottom of Form

Top of Form

**Life Cycle of a Servlet (Servlet Life Cycle)**

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.

As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

**1) 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.

**2) 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.

**3) 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. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

**public** **void** init(ServletConfig config) **throws** ServletException

**4) service method is invoked**

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

1. **public** **void** service(ServletRequest request, ServletResponse response)
2. **throws** ServletException, IOException

**5) destroy method is invoked**

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

1. **public** **void** destroy()

Bottom of Form

Top of Form

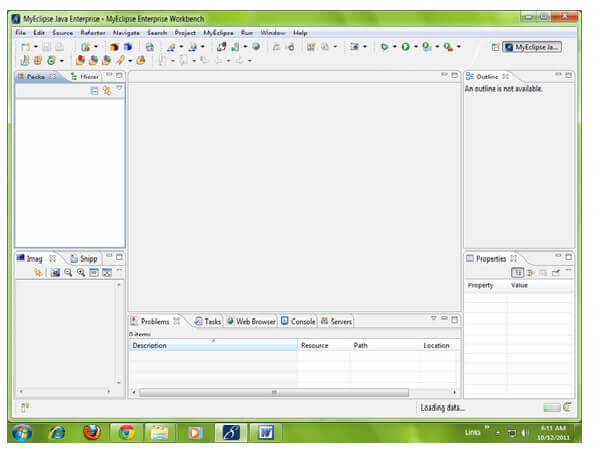
**Creating Servlet in myeclipse IDE**

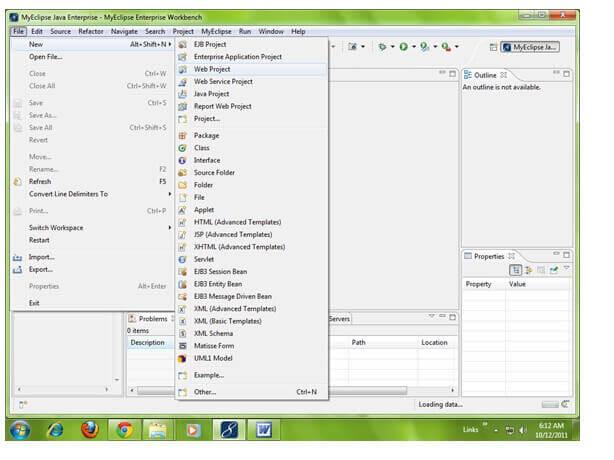
You need to follow the following steps to create the servlet in the myeclipse IDE. The steps are as follows:

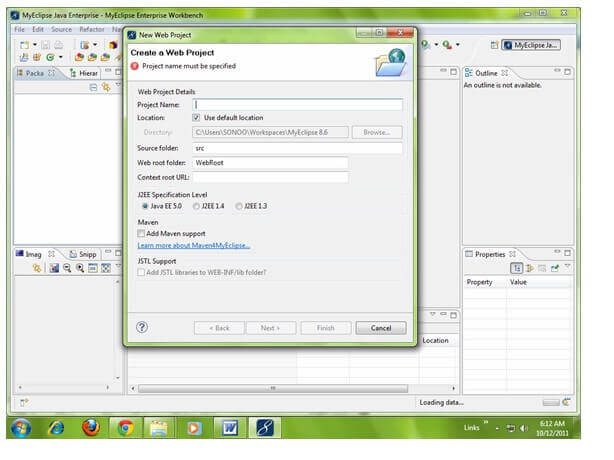
* Create a web project
* create a html file
* create a servlet
* start myeclipse tomcat server and deploy project

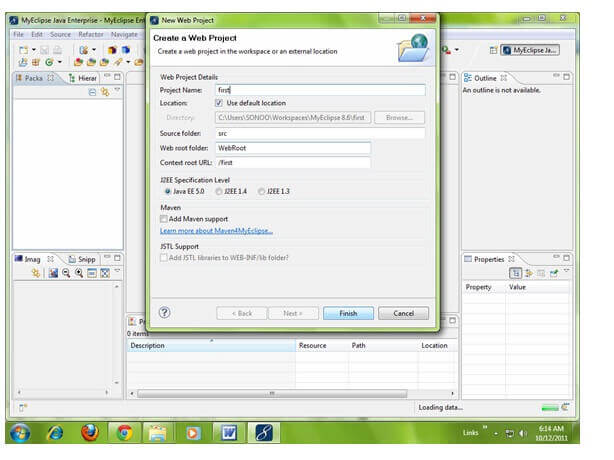
**1) Create the web project:**

For creating a web project click on File Menu -> New -> web project -> write your project name e.g. first -> Finish.





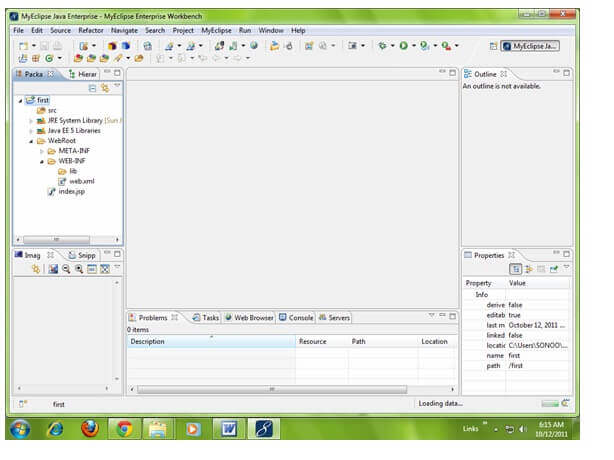




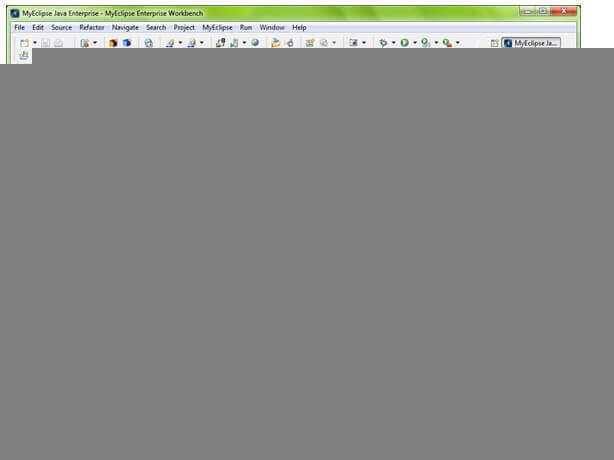
**2) Create the html file:**

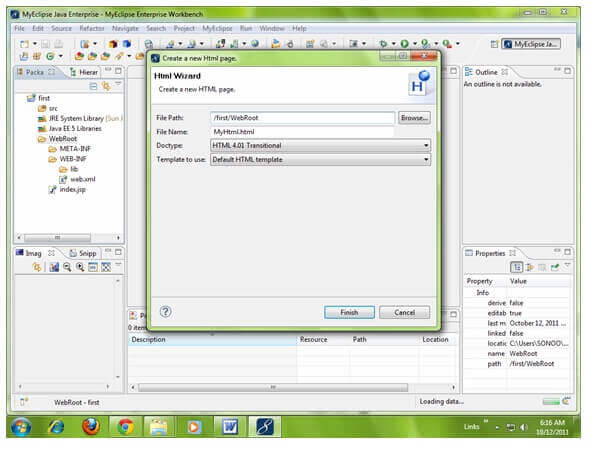
As you can see that a project is created named first. Now let's explore this project.



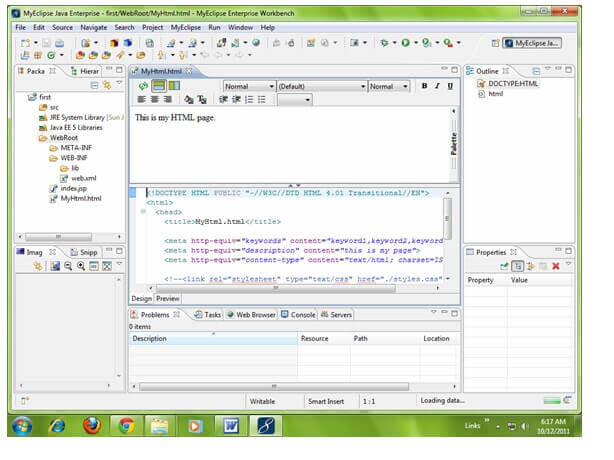


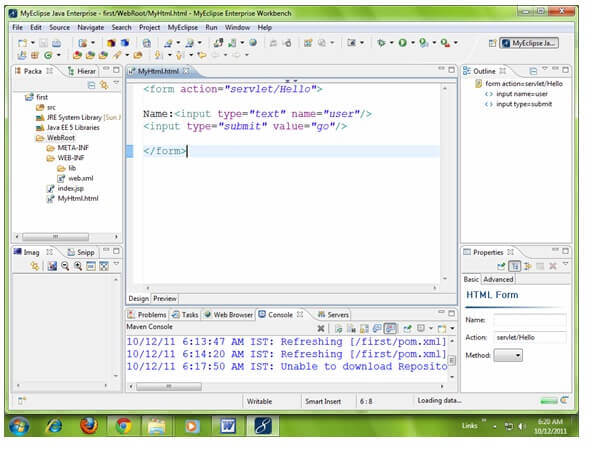
For creating a html file, right click on WebRoot -> New -> html -> write your html file name e.g. MyHtml.html -> Finish.





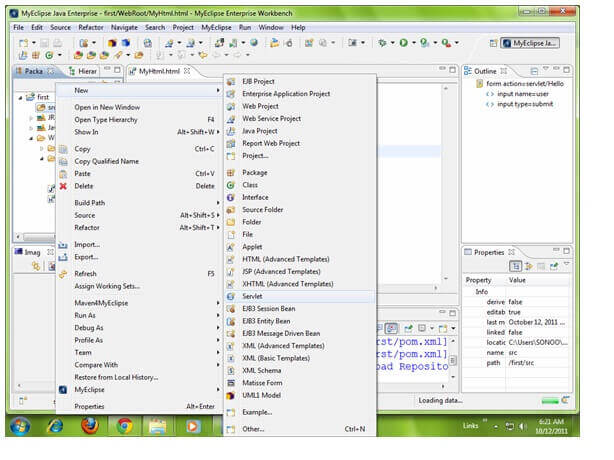
As you can see that a html file is created named MyHtml.html. Now let's write the html code here.

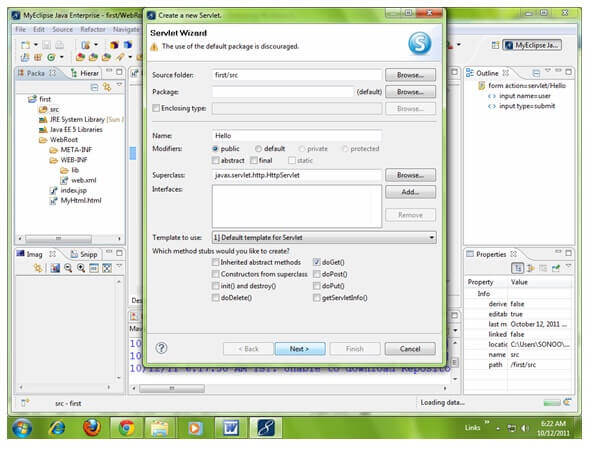


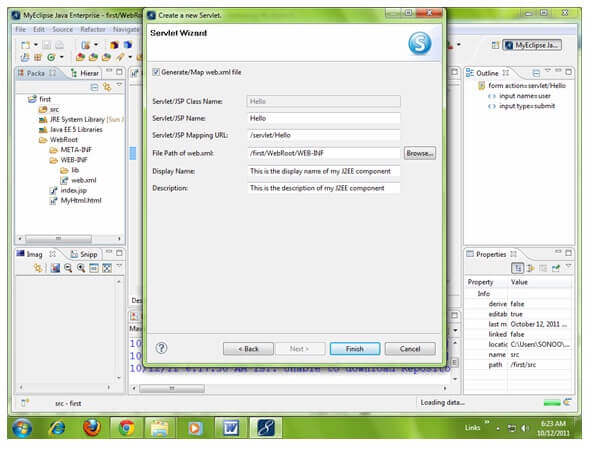


**3) Create the servlet:**

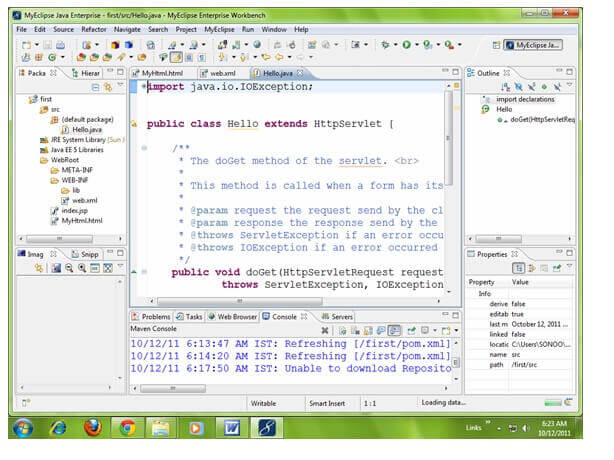
For creating a servlet click on File Menu -> New -> servlet -> write your servlet name e.g. Hello -> uncheck all the checkboxes except doGet() -> next -> Finish.

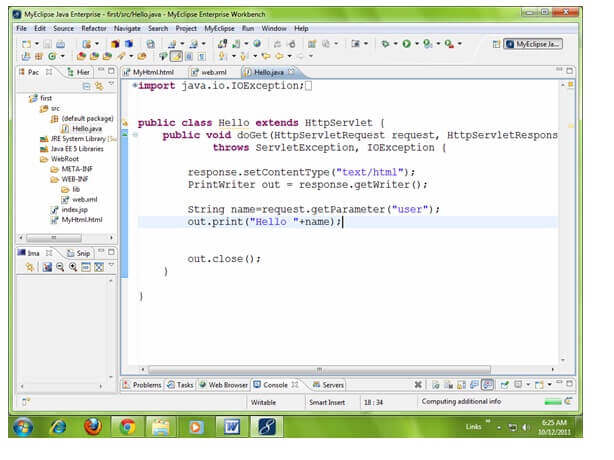




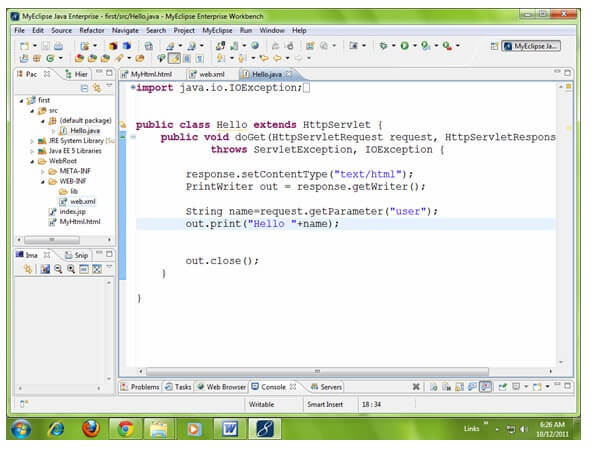


As you can see that a servlet file is created named Hello.java. Now let's write the servlet code here.

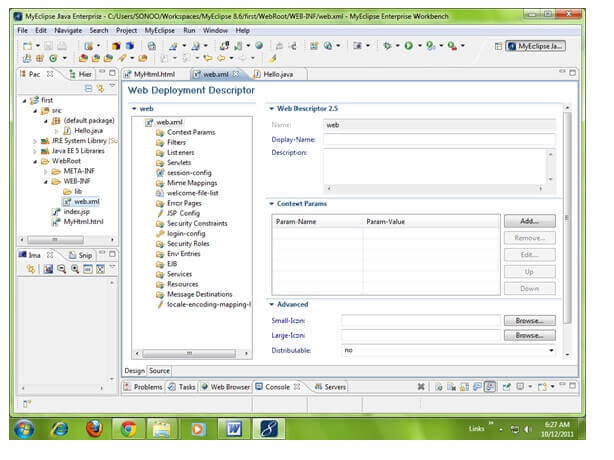




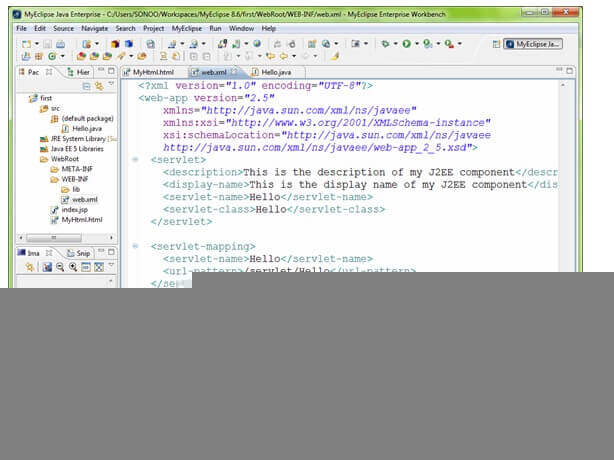
Now let's make the MyHtml.html file as the default page of our project. For this, open web.xml file and change the welcome file name as MyHtml.html in place of index.jsp.

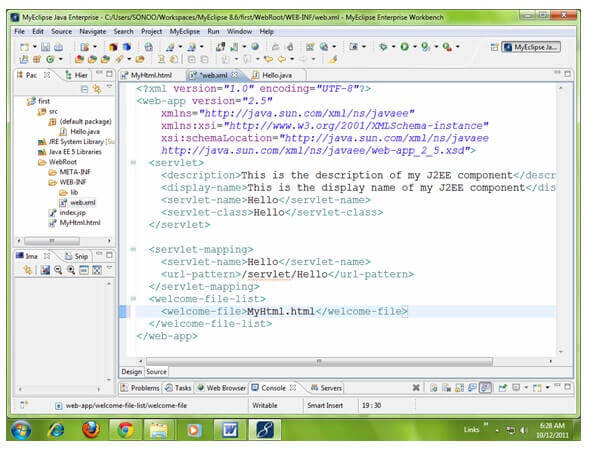


Click on the source tab to see the source code.



Now change the welcome file as MyHtml.html in place of index.jsp.

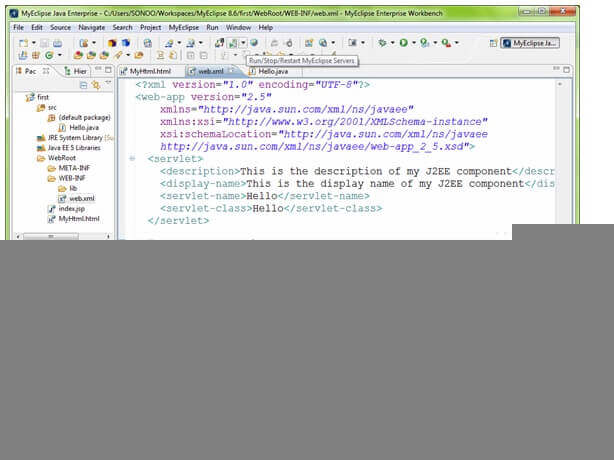


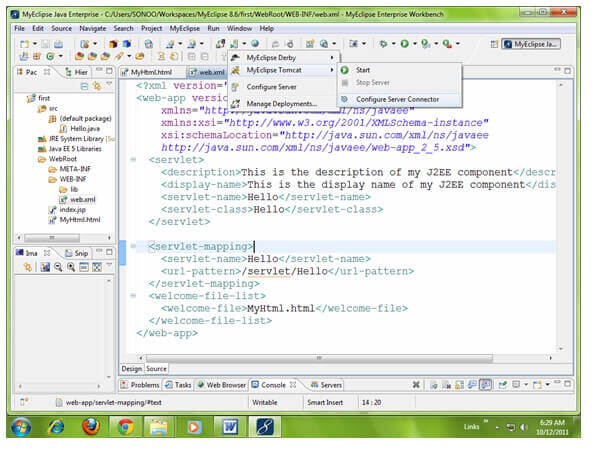


**4) Start the server and deploy the project:**

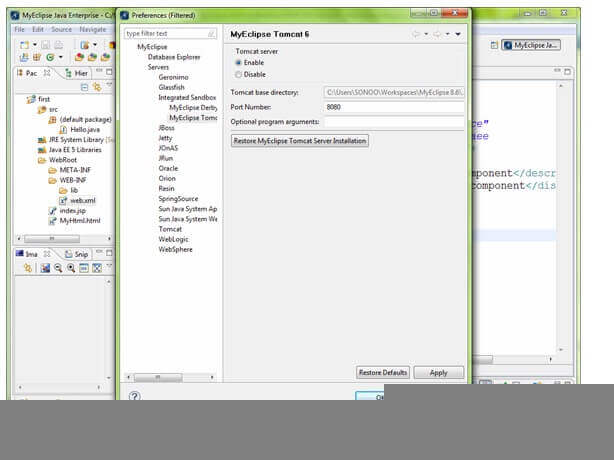
For starting the server and deploying the project in one step Right click on your project -> Run As -> MyEclipse server application.

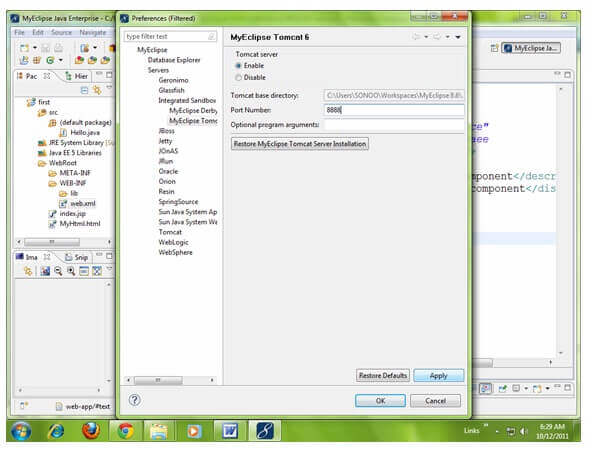
The default port of myeclipse tomcat is 8080, if you have installed oracle on your system, the port no. will conflict so let's first change the port number of myeclipse tomcat server. For changing the port number click on the start server icon at the left hand side of browser icon -> myeclipse tomcat -> Configure server connector -> change the port number as 8888 in place of 8080 -> apply -> ok.



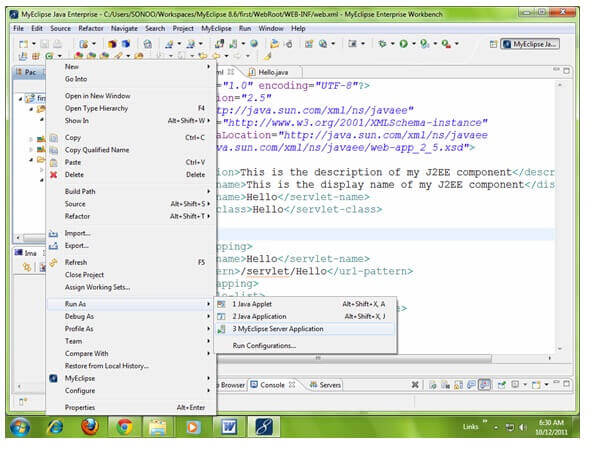


Now change the port number as 8888 in place of 8080 -> apply -> ok.

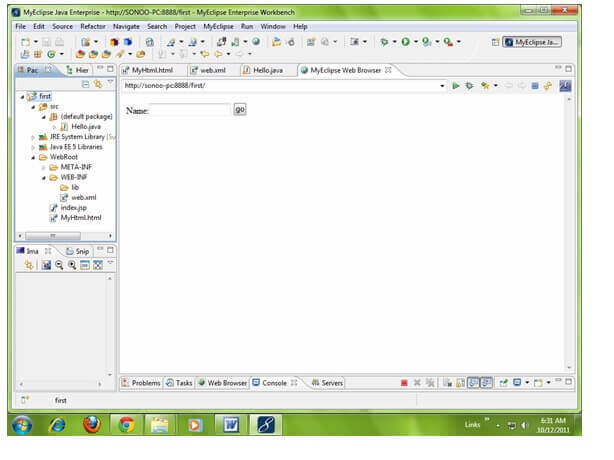


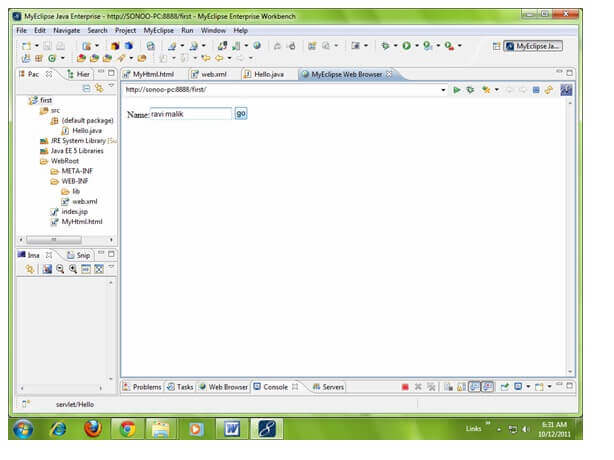


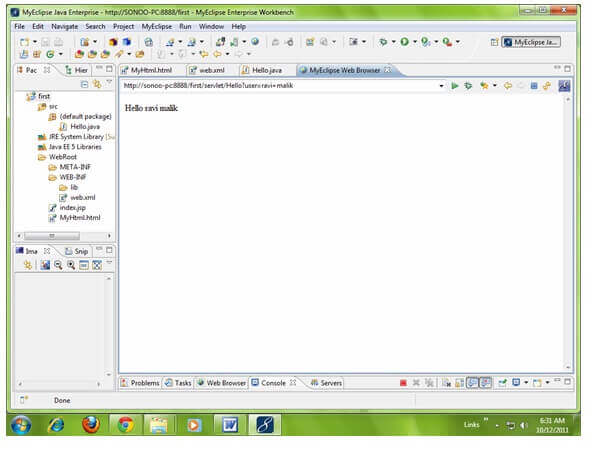
Now port number have been changed. For starting the server Right click on your project -> Run As -> MyEclipse server application.



As you can see that default page of your project is open, write your name -> go.







Bottom of Form

Top of Form

**Video Links:**

**Reference Links:**

https://www.javatpoint.com/life-cycle-of-a-servlet

https://www.javatpoint.com/HttpServlet-class

https://www.javatpoint.com/GenericServlet-class

https://www.javatpoint.com/Servlet-interface

https://beginnersbook.com/2014/04/genericservlet-class/

https://docs.oracle.com/javaee/7/tutorial/servlets002.htm

https://javaee.github.io/servlet-spec/downloads/servlet-3.1/Final/servlet-3\_1-final.pdf

Bottom of Form