

# Servlet

# HTTP(Hyper Text Transfer Protocol)

---



HTTP Request



GET /HelloClientServlet HTTP/1.1

...



HTTP Response

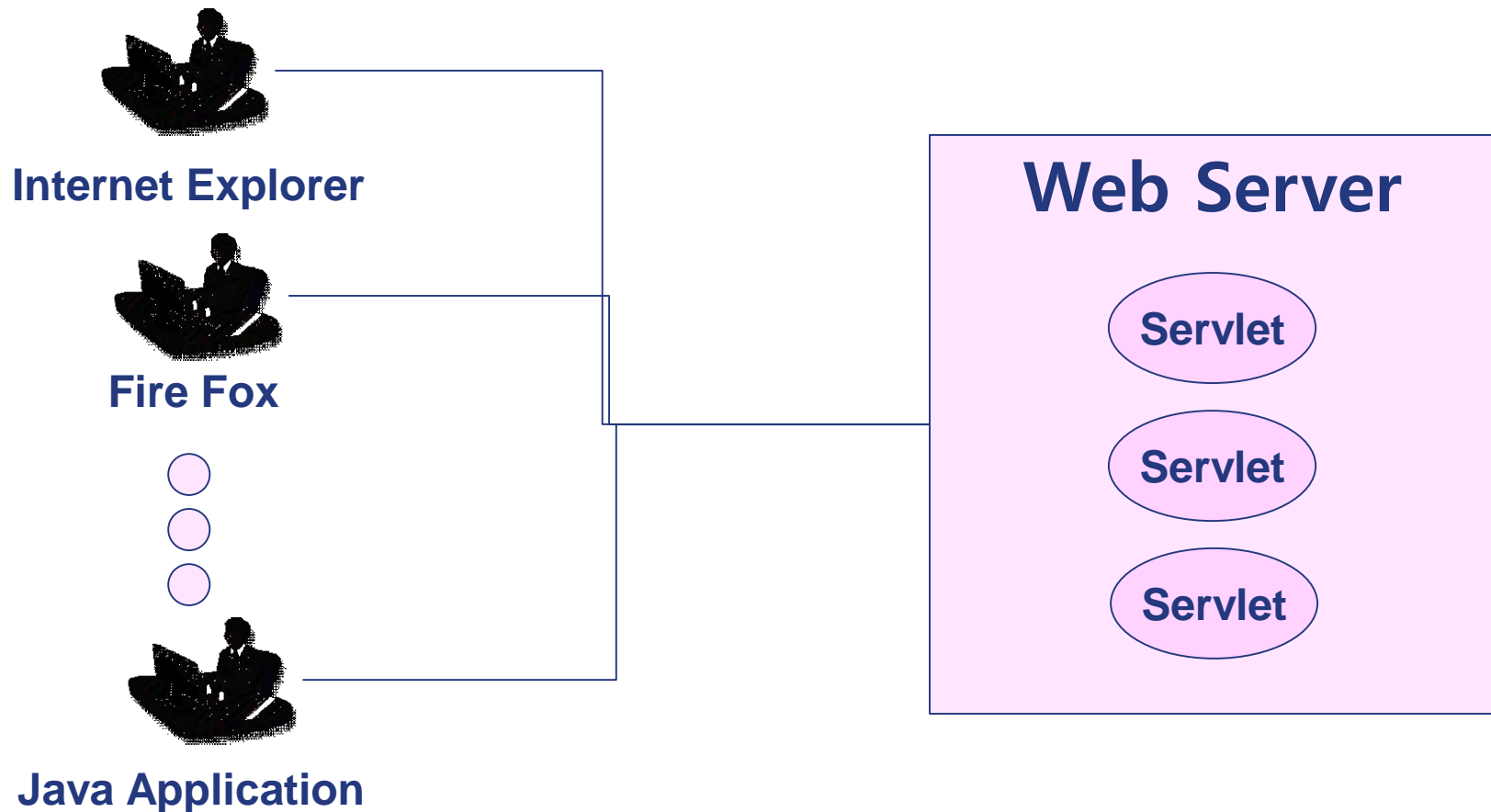
HTTP/1.1 200 OK

...

```
<HTML> <HEAD> <TITLE>Hello Client!</TITLE> </HEAD>
<BODY>
<H1>Hello </H1>
</H2> Client! </H2>
</BODY> </HTML>
```

# Servlet

- ❖ Servlets are modules of Java code that run in web server.



# Servlet - Example

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
@WebServlet("/HelloClientServlet")
public class HelloClientServlet extends HttpServlet {
    protected void doGet (HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {

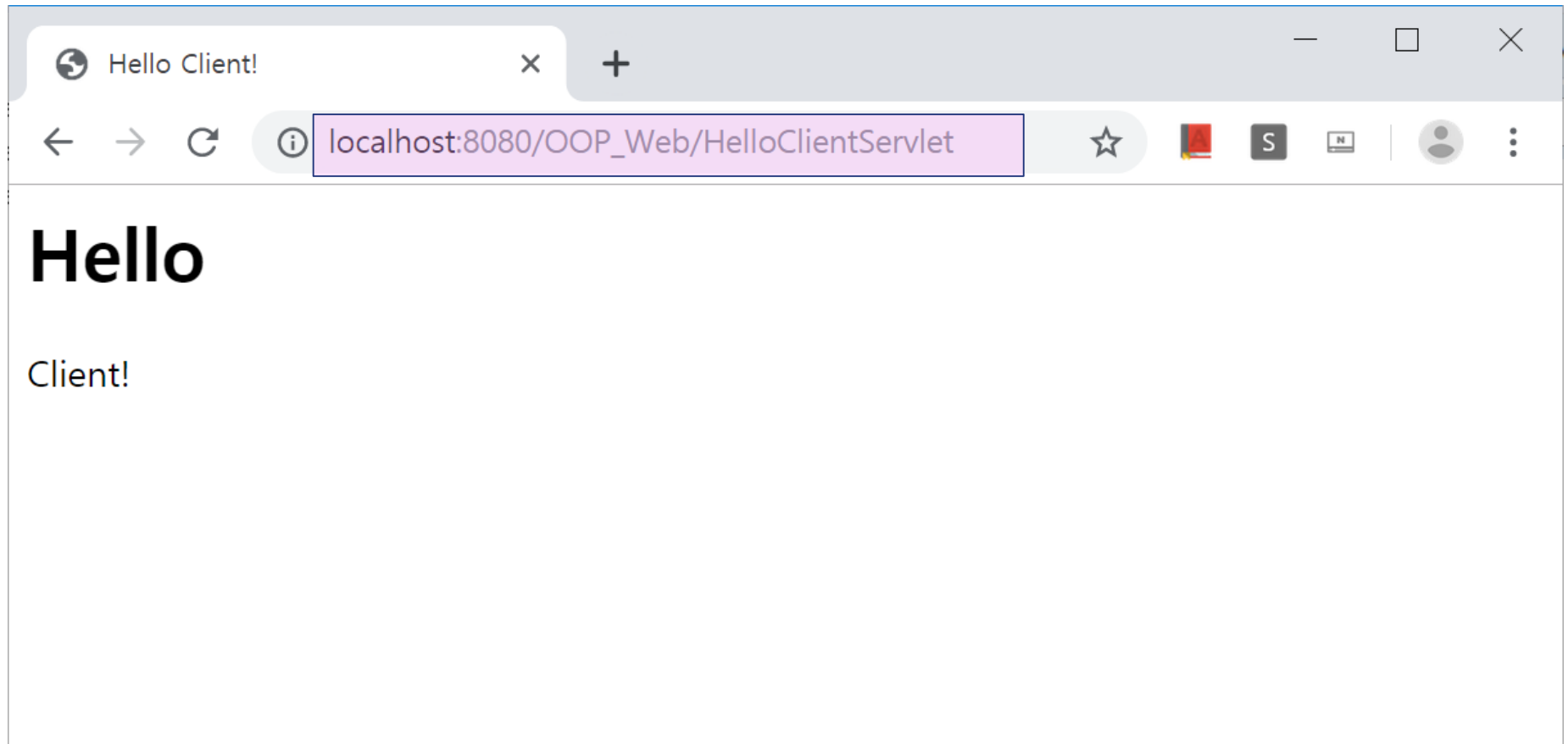
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println("<HTML> <HEAD> <TITLE>Hello Client!</TITLE>" +
            "</HEAD>\n<BODY>"
            + "<H1>Hello </H1>"
            + "</H2> Client! </H2>\n"
            + "</BODY> </HTML>");
        out.close();
    }
}
```

# Output

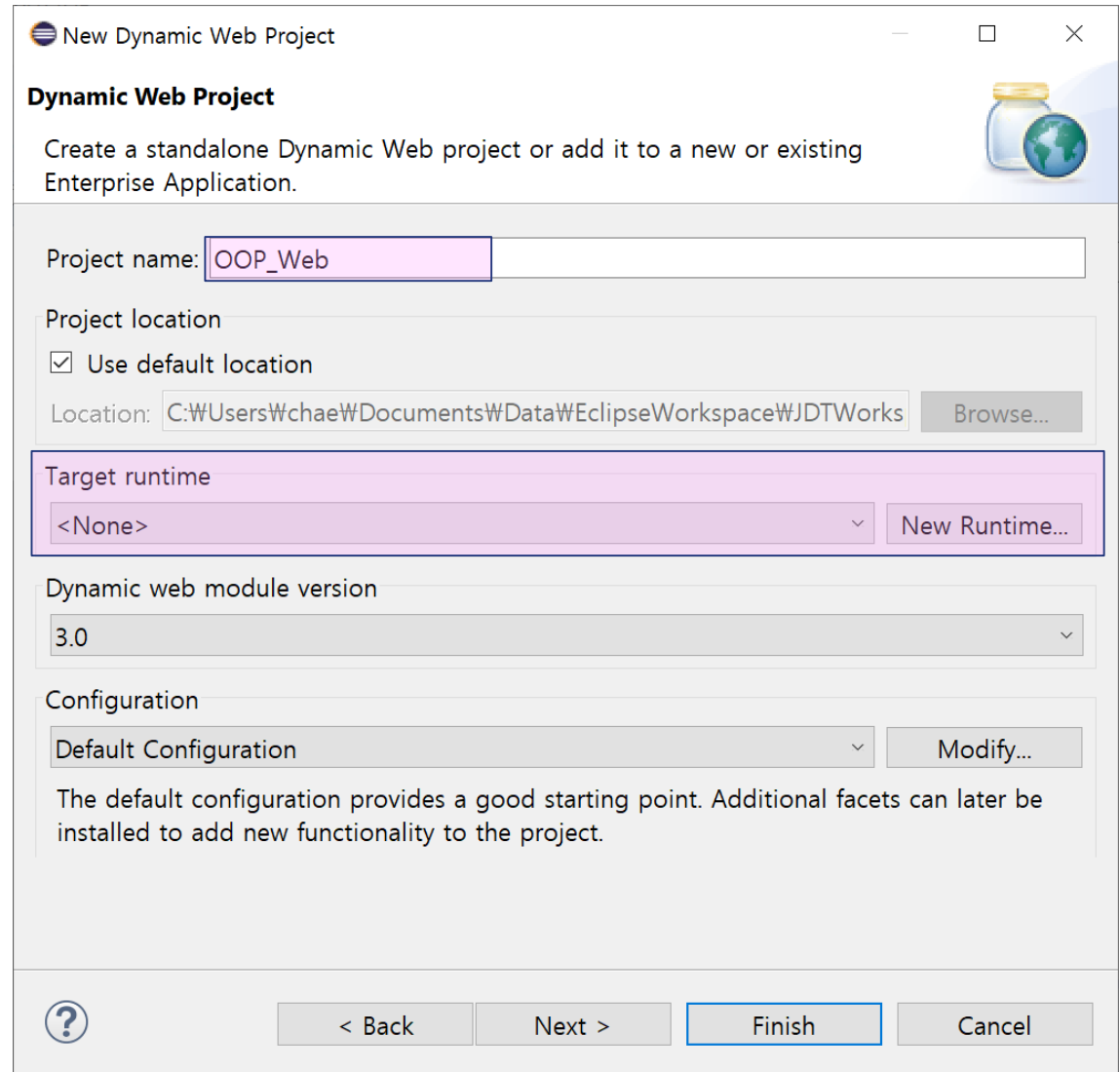
---

[http://localhost:8080/OOP\\_Web/HelloClientServlet](http://localhost:8080/OOP_Web/HelloClientServlet)



# Eclipse - Dynamic Web Project

❖ New – Other | Web –  
Dynamic Web Project



The screenshot shows the 'New Dynamic Web Project' dialog box in Eclipse. The title bar reads 'New Dynamic Web Project'. Below the title bar, the text 'Dynamic Web Project' is followed by a description: 'Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.' To the right of this text is an icon of a jar and a globe. The dialog is divided into several sections: 'Project name:' with a text field containing 'OOP\_Web'; 'Project location' with a checked 'Use default location' checkbox and a 'Location:' field showing 'C:\Users\Wchae\Documents\Data\EclipseWorkspace\JDTWorks' and a 'Browse...' button; 'Target runtime' with a dropdown menu set to '<None>' and a 'New Runtime...' button; 'Dynamic web module version' with a dropdown menu set to '3.0'; and 'Configuration' with a dropdown menu set to 'Default Configuration' and a 'Modify...' button. At the bottom, there is a help icon (question mark), and four buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

New Dynamic Web Project

**Dynamic Web Project**

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: OOP\_Web

Project location

☒ Use default location

Location: C:\Users\Wchae\Documents\Data\EclipseWorkspace\JDTWorks Browse...

Target runtime

<None> New Runtime...

Dynamic web module version

3.0

Configuration

Default Configuration Modify...

The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.

? < Back Next > Finish Cancel

# Web Server Installation

New Server Runtime Environment

**New Server Runtime Environment**

Define a new server runtime environment

Select the type of runtime environment:

type filter text

- Apache
  - Apache Tomcat v3.2
  - Apache Tomcat v4.0
  - Apache Tomcat v4.1
  - Apache Tomcat v5.0
  - Apache Tomcat v5.5
  - Apache Tomcat v6.0
  - Apache Tomcat v7.0
  - Apache Tomcat v8.0

Apache Tomcat v8.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.

☐ Create a new local server

? < Back Next > Finish Cancel

New Server Runtime Environment

**Tomcat Server**

Specify the installation directory

Name:  
Apache Tomcat v8.0

Tomcat installation directory:  
 Browse...

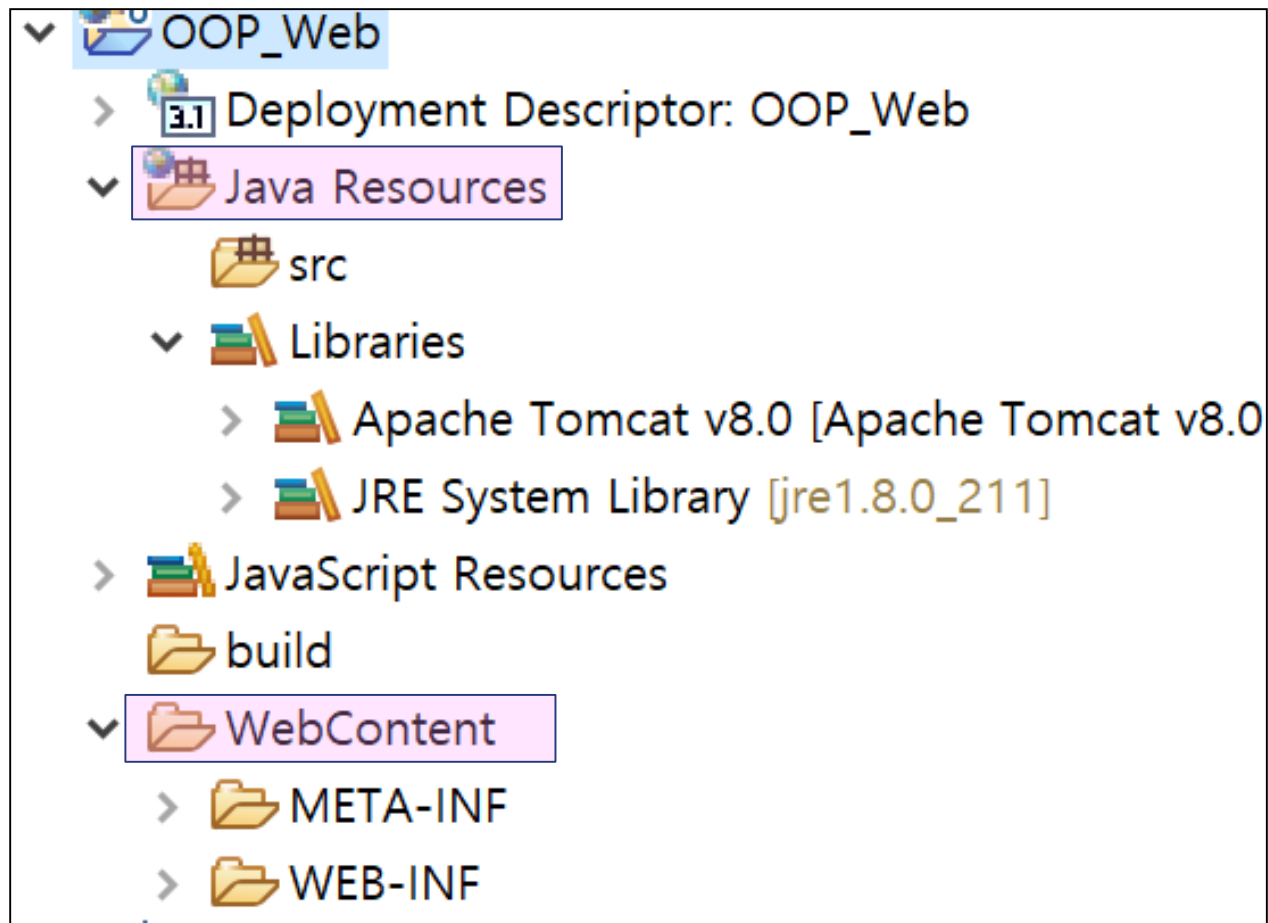
apache-tomcat-8.0.36 Download and Install..

JRE:  
Workbench default JRE Installed JREs...

? < Back Next > Finish Cancel

# Dynamic Web Project - Structure

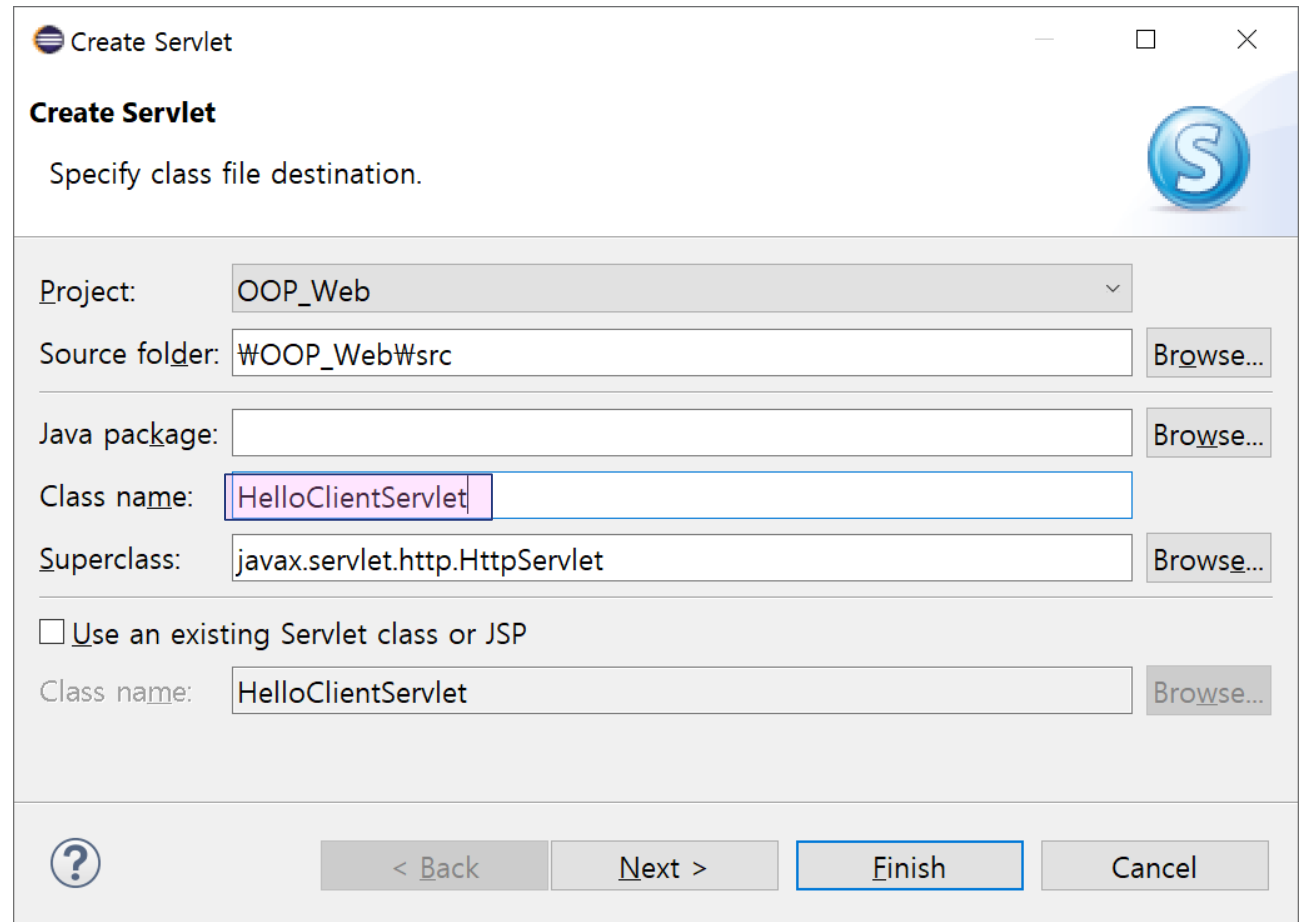
---





# Create Servlet

## ❖ OOP-Web – New – Servlet



The image shows a 'Create Servlet' dialog box from an IDE. It has a title bar with a standard icon and window controls. The main area is titled 'Create Servlet' and contains the instruction 'Specify class file destination.' To the right of this text is a blue circular icon with a white 'S'. Below the instruction are several input fields: 'Project:' with a dropdown menu showing 'OOP\_Web'; 'Source folder:' with a text box containing 'WOOO\_PWebWsrc' and a 'Browse...' button; 'Java package:' with an empty text box and a 'Browse...' button; 'Class name:' with a text box containing 'HelloClientServlet' and a 'Browse...' button; and 'Superclass:' with a text box containing 'javax.servlet.http.HttpServlet' and a 'Browse...' button. Below these fields is a checkbox labeled 'Use an existing Servlet class or JSP'. Under this checkbox is another 'Class name:' text box containing 'HelloClientServlet' and a 'Browse...' button. At the bottom of the dialog is a footer bar containing a help icon (a question mark in a circle) and four buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), and 'Cancel'.

**Create Servlet**

Specify class file destination.

Project: OOP\_Web

Source folder: WOOO\_PWebWsrc [Browse...](#)

Java package: [Browse...](#)

Class name: HelloClientServlet [Browse...](#)

Superclass: javax.servlet.http.HttpServlet [Browse...](#)















☐ Use an existing Servlet class or JSP

Class name: HelloClientServlet [Browse...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

# Servlet Source File

---

- ▼  OOP\_Web
  - >  3.1 Deployment Descriptor: OOP\_Web
  - ▼  Java Resources
    - ▼  src
      - ▼  (default package)
        - >  HelloClientServlet.java
    - ▼  Libraries
      - >  Apache Tomcat v8.0 [Apache Tomcat v8.0]
      - >  JRE System Library [jre1.8.0\_211]
  - >  JavaScript Resources
  - >  build
  - ▼  WebContent
    - >  META-INF
    - >  WEB-INF

# Edit the Servlet

```
@WebServlet("/HelloClientServlet")
public class HelloClientServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloClientServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>Hello Client!</TITLE>" +
            "</HEAD><BODY>"
            + "<H1>Hello </H1>"
            + "</H2> Client! </H2>"
            + "</BODY></HTML>");
        out.close();
    }
}
```

# Run Servlet

## ❖ OOP\_Web – Run as – Run on Server

**Run On Server**

Select which server to use

How do you want to select the server?

☐ Choose an existing server

☒ Manually define a new server

Select the server type:

type filter text

- Tomcat v7.0 Server
- Tomcat v8.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Apache Tomcat v8.0 at localhost

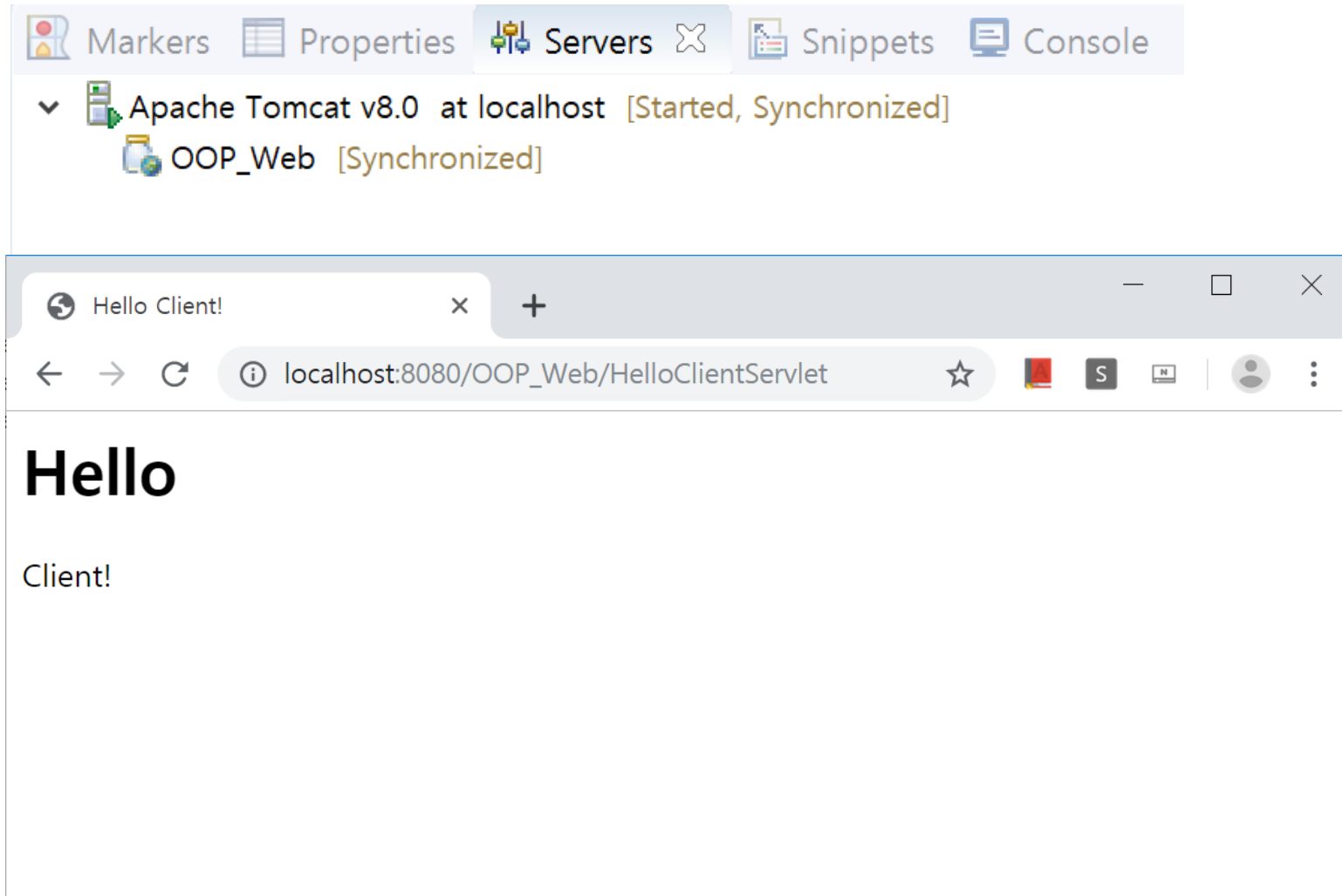
Server runtime environment: Apache Tomcat v8.0 [Add...](#)

[Configure runtime environments...](#)

☒ Always use this server when running this project

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

# Run Servlet



---

# MORE EXAMPLES

# ClientInfoServlet

---



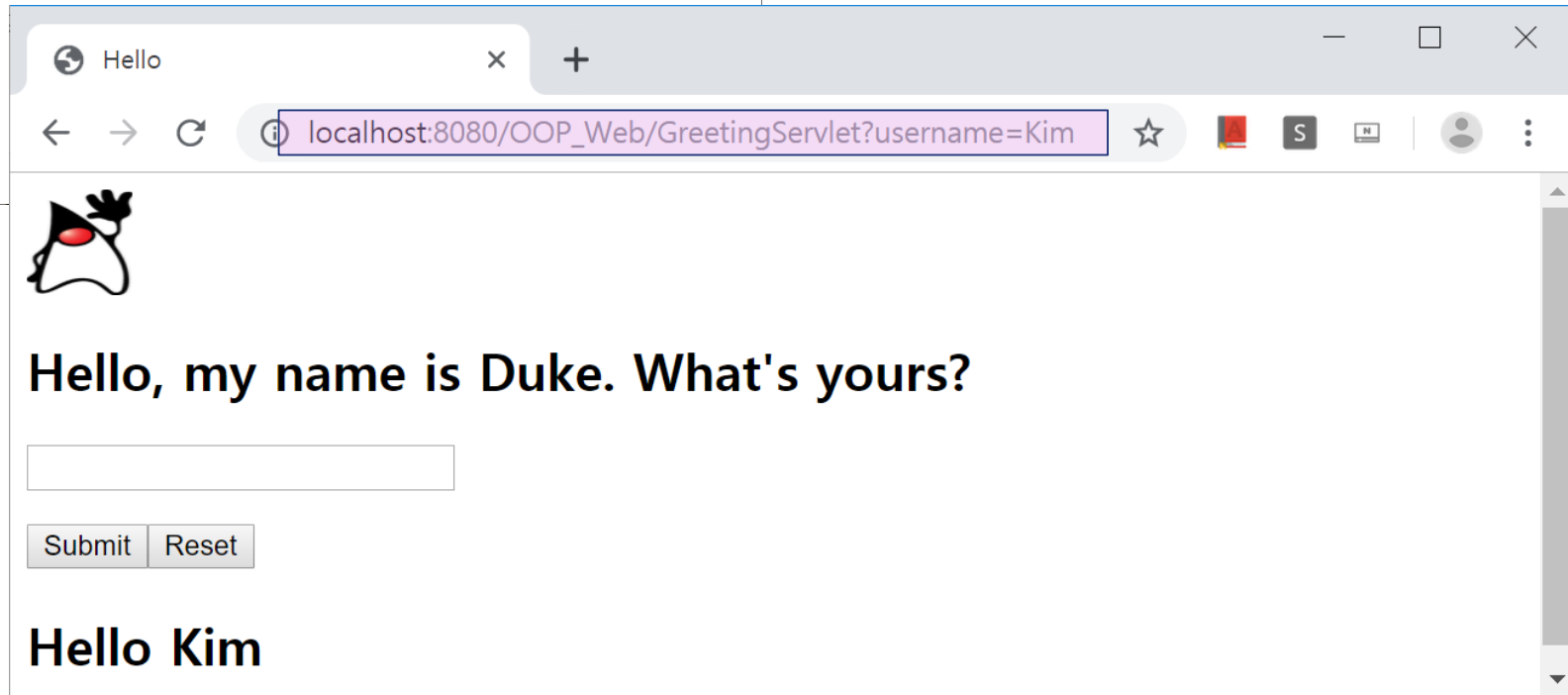
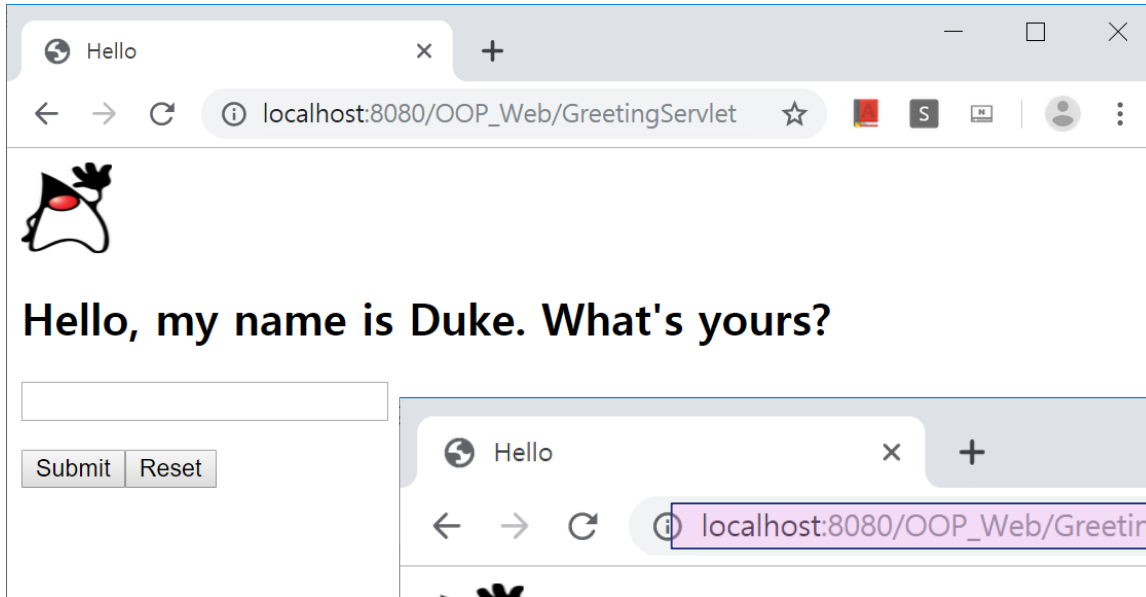
# Servlet For Dynamic Content

---

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ClientInfoServlet extends HttpServlet {
    protected void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML> <HEAD> <TITLE>Hello Client!</TITLE>" +
            "</HEAD> <BODY>Hello Client!<P> <HR>");
        out.println("You are from " + request.getRemoteHost() + ":" +
            request.getRemotePort() + " at " +
            request.getRequestURL() );
        out.println("</BODY> </HTML>");
        out.close();
    }
}
```



# GreetingServlet



```

public class GreetingServlet extends HttpServlet {
    public void doGet (
        HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        response.setBufferSize(8192);

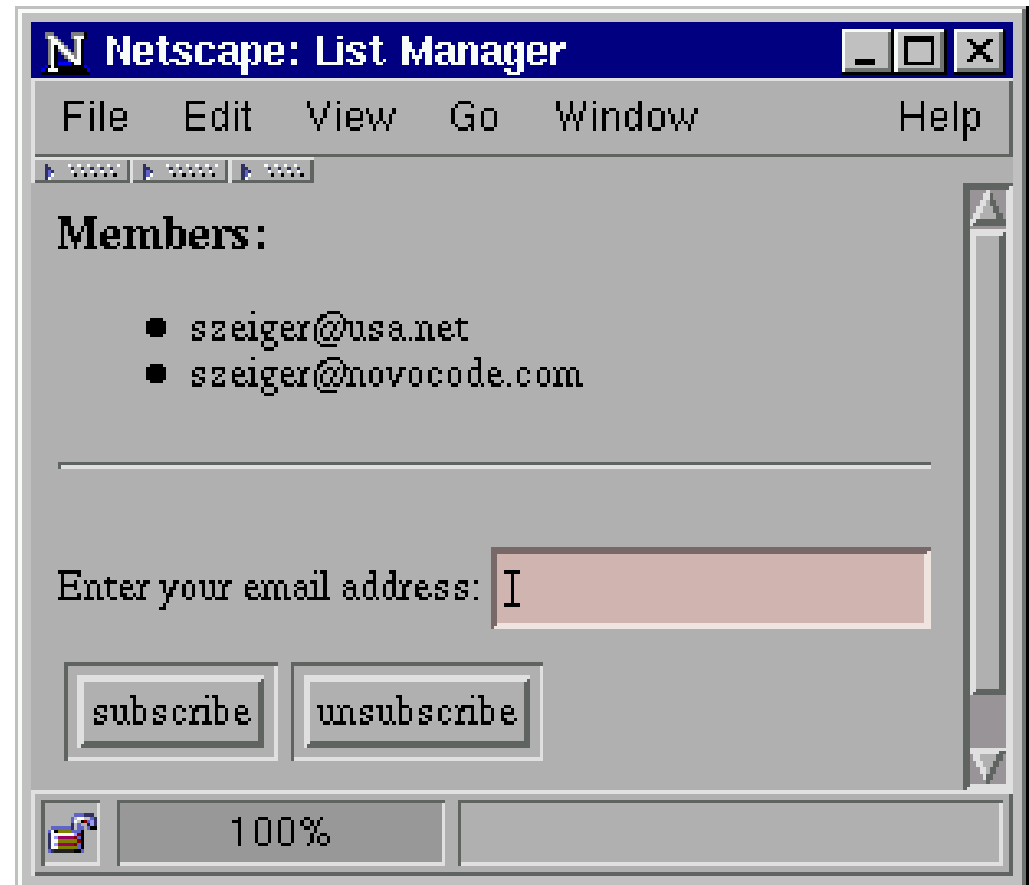
        PrintWriter out = response.getWriter();
        out.println("<html>" + "<head> <title>Hello</title> </head>");
        out.println(
            "<body> <img src=\\\"duke.waving.gif\\\" alt=\\\"Duke waving\\\"> "
            + "<h2>Hello, my name is Duke. What's yours? </h2>"
            + "<form method=\\\"get\\\">"
            + "<input type=\\\"text\\\" name=\\\"username\\\" size=\\\"25\\\">"
            + "<p> </p>" + "<input type=\\\"submit\\\" value=\\\"Submit\\\">"
            + "<input type=\\\"reset\\\" value=\\\"Reset\\\">" + "</form>");

        String username = request.getParameter("username");
        if ((username != null) && (username.length() > 0)) {
            out.println("<H2> <B>Hello " + username + "</B> </H2>");
        }
        out.println("</body> </html>");
        out.close();
    }
}


```



# Form Processing Servlet

- ❖ process form data
- ❖ manage persistent data
- ❖ use init parameters




# Servlet Parameters

 Create Servlet



## Create Servlet

Enter servlet deployment descriptor specific information.



Name:

Description:

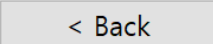

Initialization parameters:

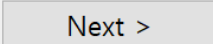
Name	Value	Description

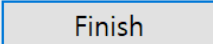
URL mappings:

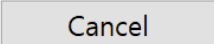
/ListManagerServlet
---------------------

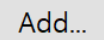
☐ Asynchronous Support


 < Back

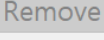
 Next >

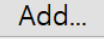
 Finish

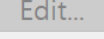
 Cancel

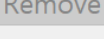
 Add...


 Edit...

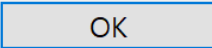
 Remove

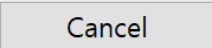
 Add...

 Edit...

 Remove

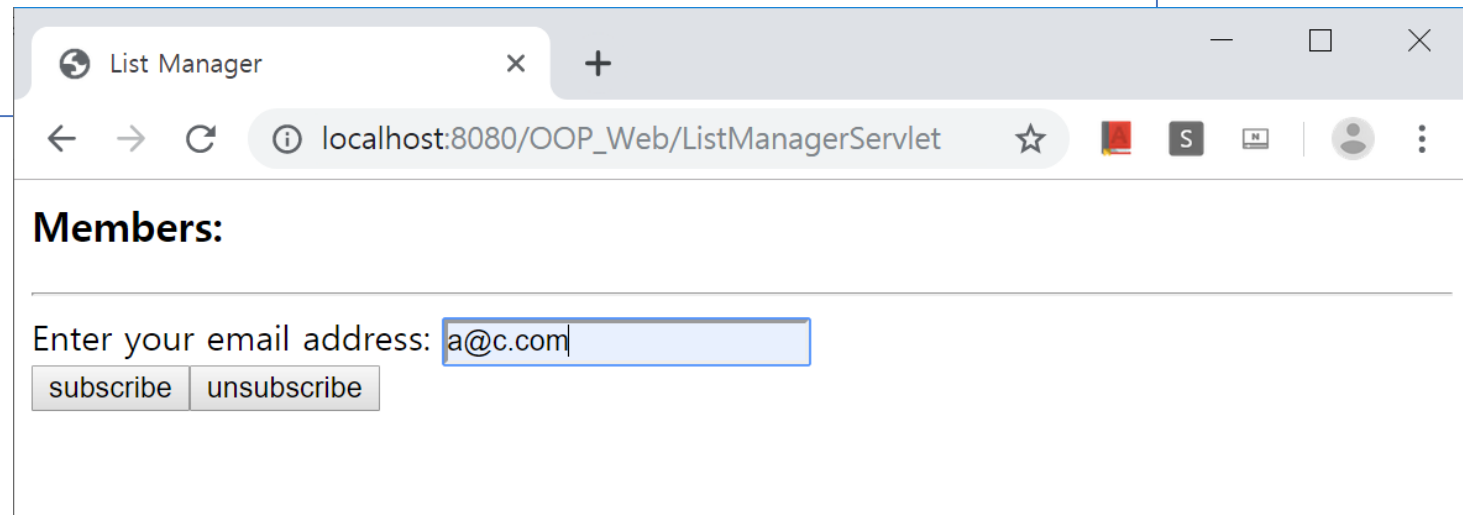
 Initialization Parameters

 OK

 Cancel

```
@WebServlet(  
    urlPatterns = { "/ListManagerServlet" },  
    initParams = {  
        @WebInitParam(name = "addressfile", value = "address")  
    })  
public class ListManagerServlet extends HttpServlet {  
    private Vector addresses;  
    private String filename;  
    public void init(ServletConfig config) throws ServletException {  
        super.init(config);  
        filename = config.getInitParameter("addressfile");  
        try {  
            ObjectInputStream in =  
                new ObjectInputStream(new FileInputStream(filename));  
            addresses = (Vector)in.readObject();  
            in.close();  
        }  
        catch(IOException e) { addresses = new Vector(); }  
        catch(ClassNotFoundException e) {addresses = new Vector(); }  
    }  
}
```

```
protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException 37: {
    res.setContentType("text/html");
    res.setHeader("pragma", "no-cache");
    PrintWriter out = res.getWriter();
    out.print("<HTML> <HEAD> <TITLE> List Manager</TITLE> </HEAD>");
    out.print("<BODY> <H3> Members:</H3> <UL>");
    for(int i=0; i<addresses.size(); i++) out.print("<LI>" + addresses.elementAt(i));
    out.print("</UL> <HR> <FORM METHOD=POST>");
    out.print("Enter your email address: <INPUT TYPE=TEXT NAME=email> <BR>");
    out.print("<INPUT TYPE=SUBMIT NAME=action VALUE=subscribe>");
    out.print("<INPUT TYPE=SUBMIT NAME=action VALUE=unsubscribe>");
    out.print("</FORM> </BODY> </HTML>");
    out.close();
}
```



List Manager

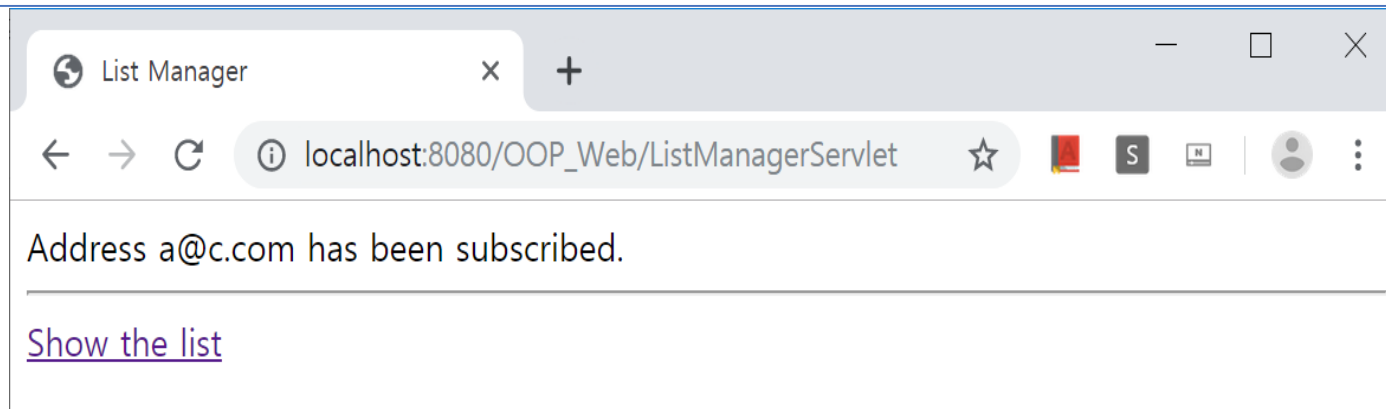
localhost:8080/OOP\_Web/ListManagerServlet

### Members:

Enter your email address:

```
protected void doPost (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String email = req.getParameter("email");
    String msg;
    if(email == null) { res.sendError(res.SC_BAD_REQUEST,
        "No email address specified."); return; }
    if(req.getParameter("action").equals("subscribe")) {
        if ( subscribe(email) ) msg = "Address " + email + " has been subscribed.";
        else {
            res.sendError(res.SC_BAD_REQUEST, "Address " + email +
                " was already subscribed.");
            return;
        }
    }
    else { // unsubscribe is requested.
        if ( unsubscribe(email) ) msg = "Address " + email + " has been removed.";
        else {
            res.sendError(res.SC_BAD_REQUEST, "Address " + email +
                " was not subscribed."); return;
        }
    }
}
```

```
res.setContentType("text/html"); res.setHeader("pragma", "no-cache");
PrintWriter out = res.getWriter();
out.print("<HTML> <HEAD> <TITLE>List Manager</TITLE> </HEAD> <BODY>");
out.print(msg);
out.print("<HR> <A HREF=#\"");
out.print(req.getRequestURI());
out.print("#\">Show the list</A> </BODY> </HTML>");
out.close();
}
```





```
private synchronized boolean subscribe (String email) throws IOException {  
    if ( addresses.contains(email) ) return false;  
    addresses.addElement(email);  
    save();  
    return true;  
}
```

```
private synchronized boolean unsubscribe (String email) throws IOException {  
    if ( !addresses.removeElement(email) ) return false;  
    save();  
    return true;  
}
```

```
private void save() throws IOException {  
    ObjectOutputStream out =  
        new ObjectOutputStream(new FileOutputStream(filename));  
    out.writeObject(addresses);  
    out.close();  
}
```

```
}
```

# GET vs POST

	GET	POST
Visibility	Data is visible in the URL	Data is not displayed in the URL
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes. Maximum URL length is 2048 characters	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure because data sent is part of the URL. Never use GET when sending sensitive information!	POST is a little safer because the parameters are not stored in browser history or in web server logs
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)

[https://www.w3schools.com/tags/ref\\_httpmethods.asp](https://www.w3schools.com/tags/ref_httpmethods.asp)