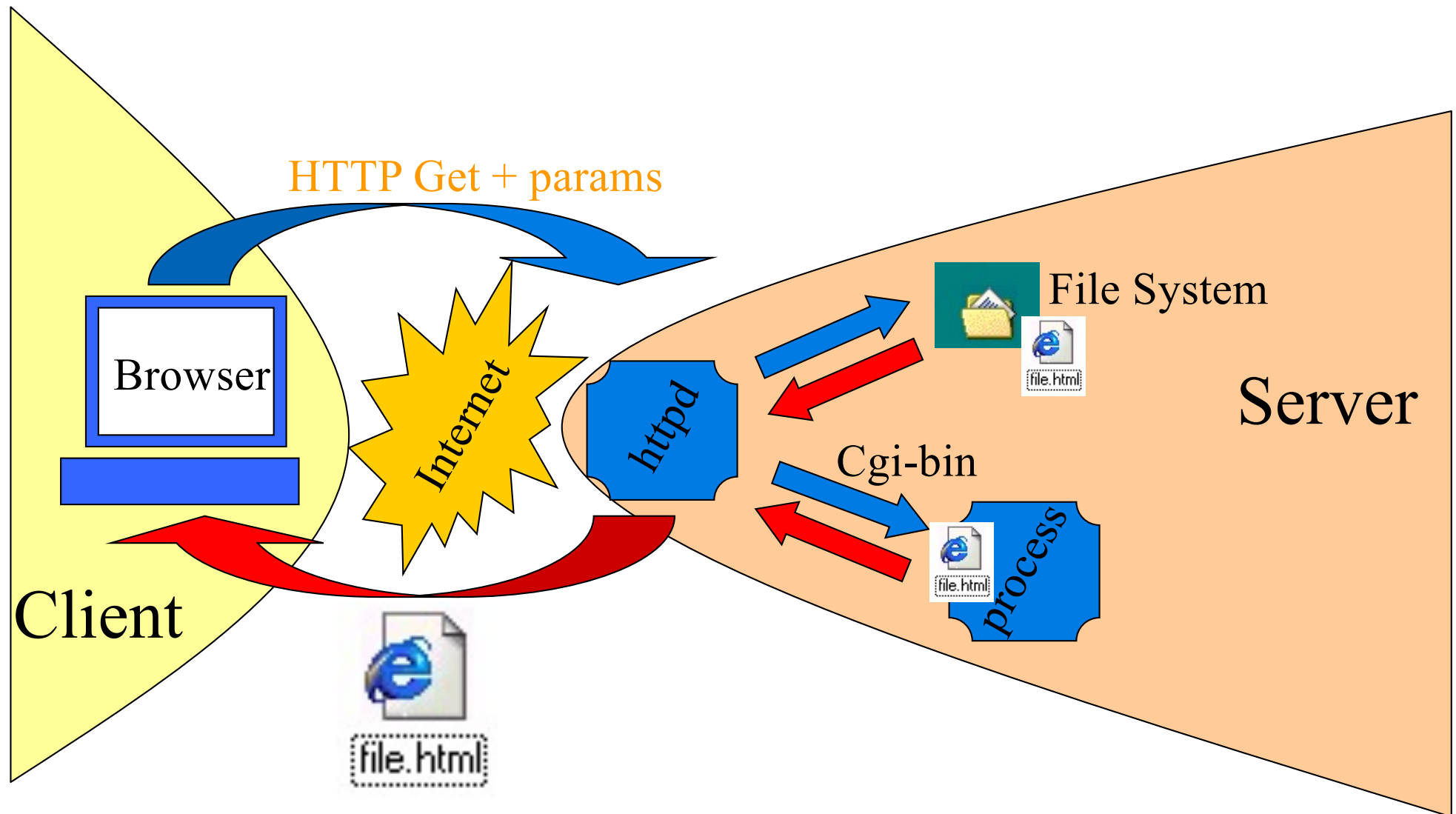


Dynamic content: programming the web servers

Q

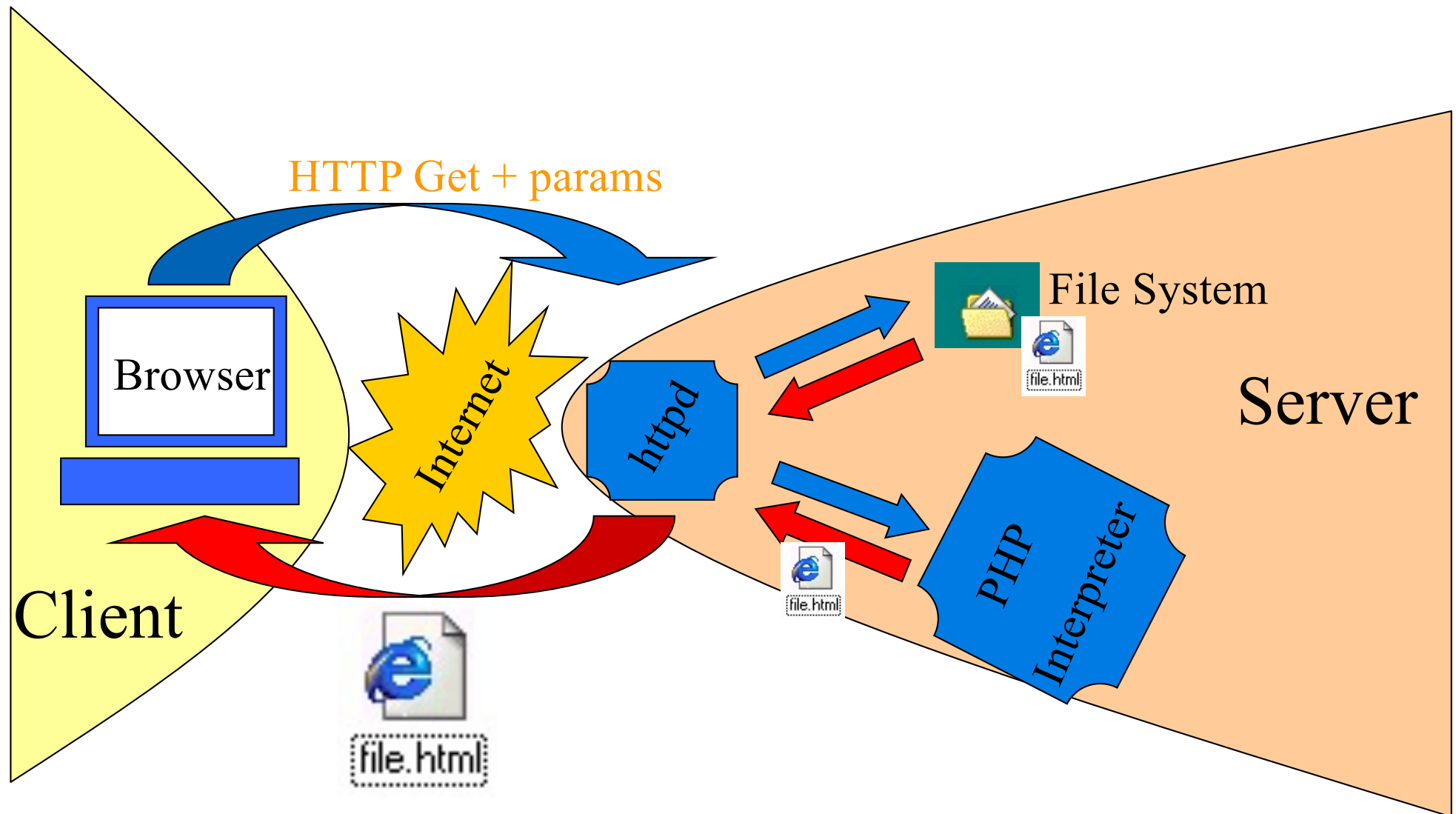
How can we obtain dynamic responses from a Web server?

The original web architecture: dynamic pages



Evolution 1: dynamically create (interlinked) documents

The original web architecture: dynamic pages



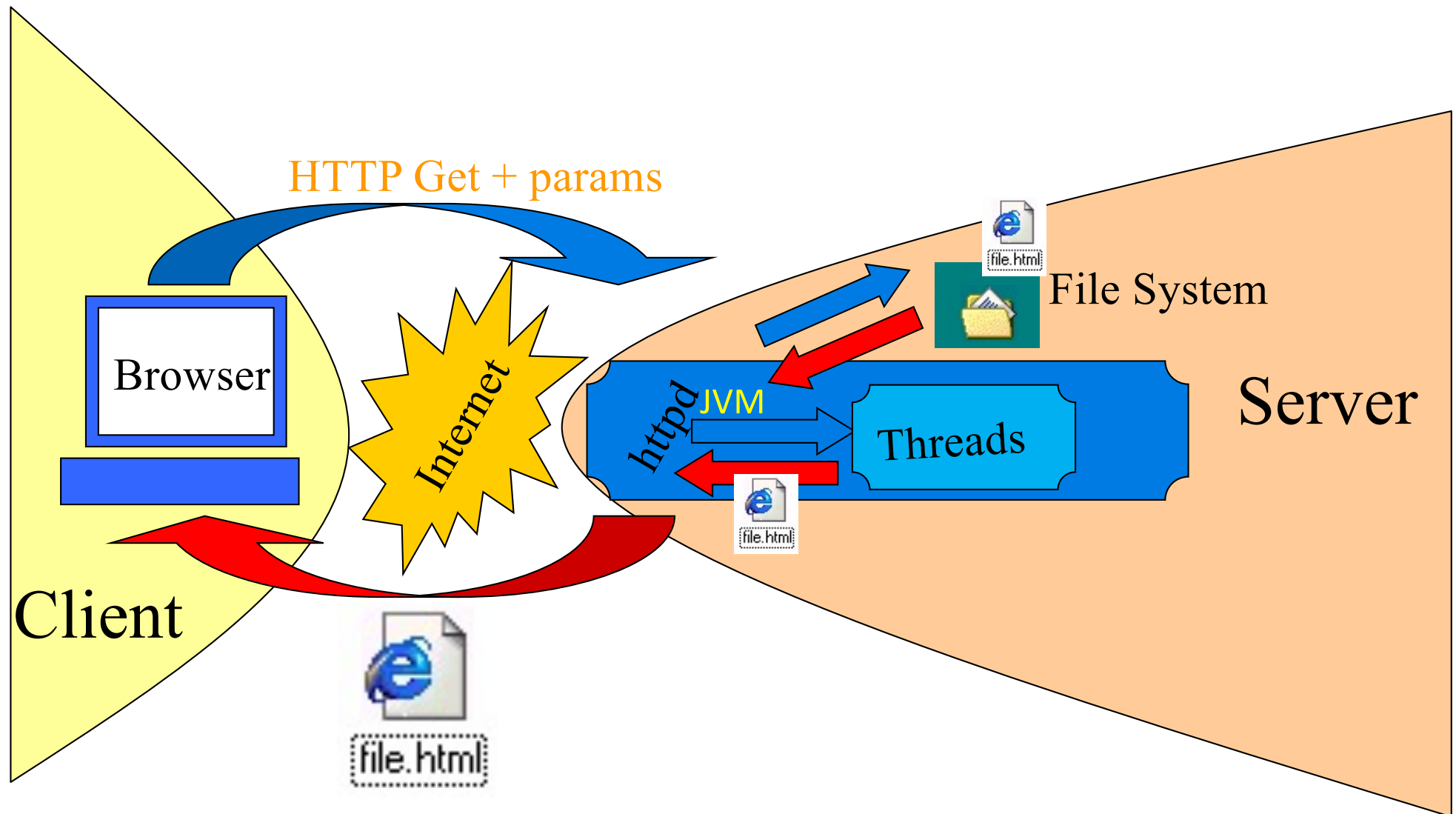
Evolution 1: dynamically create (interlinked) documents

A third, more efficient solution

Instead of spawning a process, extent the Web Server to allow for threads, and create a callback-based framework

Give developers API to implement the callbacks

The original web architecture: dynamic pages



Evolution 1: dynamically create (interlinked) documents

Q

What is a Servlet ?

Servlets

A Servlet is a class that exposes methods (called doX, where X is an HTTP method), which are called by a Web Server in a thread.

The server must be compliant with the J2EE specifications.

It is able to inject code in a WebServer Thread

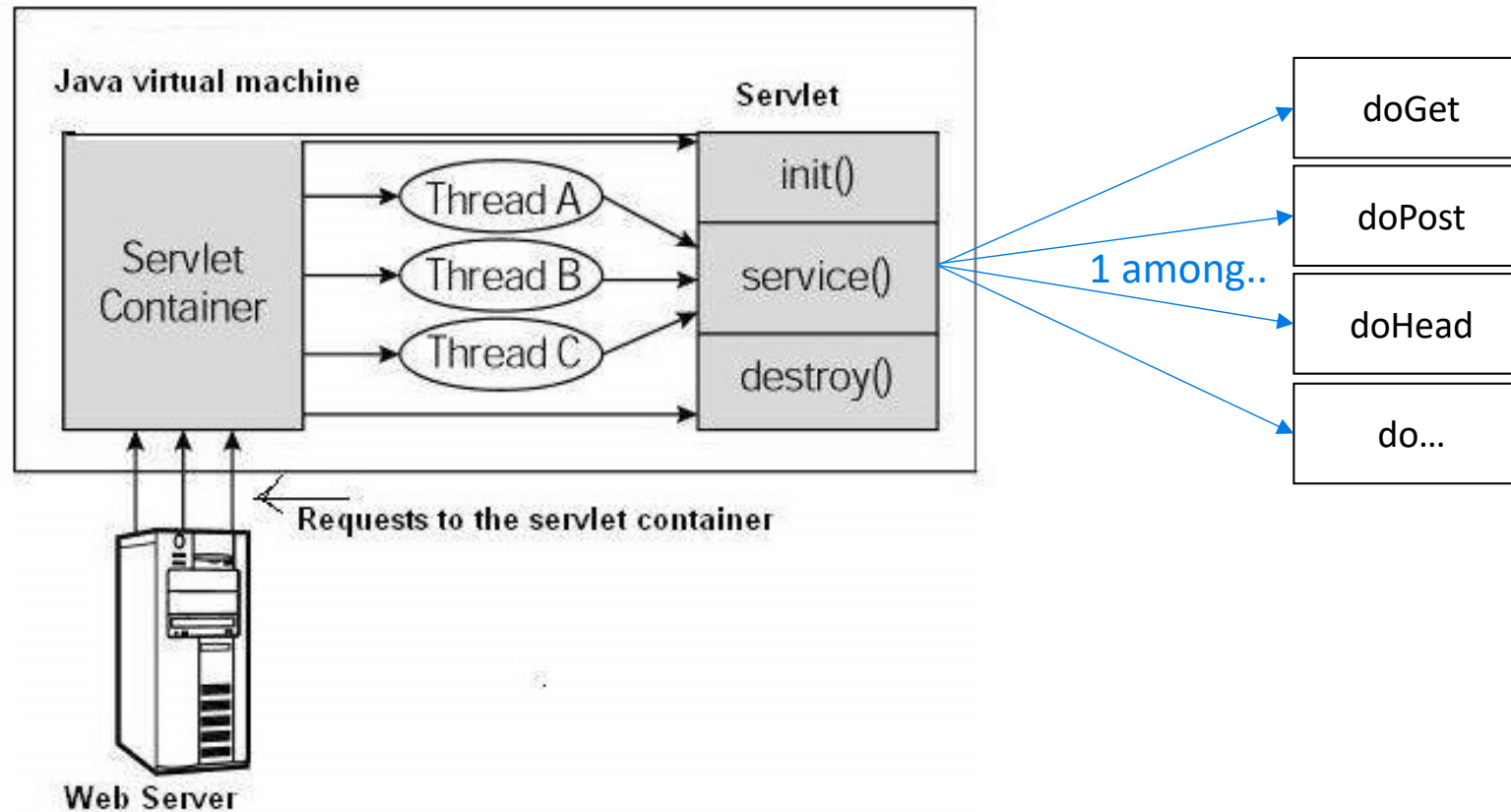
Documentation:

<https://jakarta.ee/specifications/servlet/5.0/jakarta-servlet-spec-5.0.pdf>

This code is part of the class HttpServlet

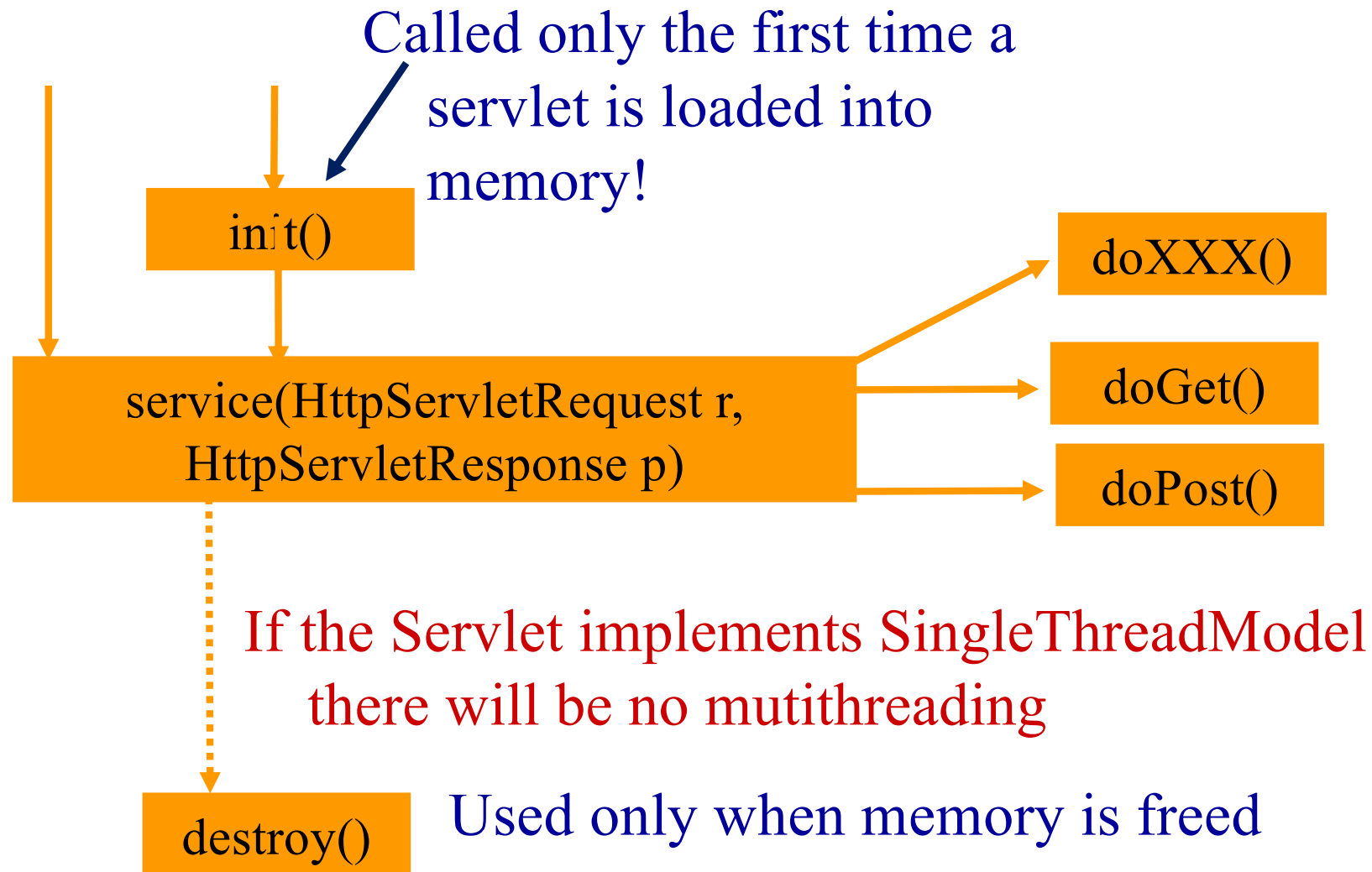
```
protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
{
    String method = req.getMethod ();
    if (method.equals ("GET")) {
        ...
        if (ifModifiedSince == -1 || lastModified == -1) doGet (req, resp);
    } else if (method.equals ("HEAD")) { ... ; doHead (req, resp);
    } else if (method.equals ("POST")) { doPost (req, resp);
    } else if (method.equals ("PUT")) { doPut(req, resp);
    } else if (method.equals ("DELETE")) {doDelete(req, resp);
    } else if (method.equals ("OPTIONS")) {doOptions(req,resp);
    } else if (method.equals ("TRACE")) { doTrace(req,resp);
    } else {
        resp.sendError (HttpServletResponse.SC_NOT_IMPLEMENTED,
            "Method '" + method + "' is not defined in RFC 2068");
    }
}
```

Servlet lifecycle



See <https://www.tutorialspoint.com/servlets/servlets-life-cycle.htm>

Servlet Lifecycle



Q

How can we write a Servlet ?

Servlets

1. Create an HttpServlet subclass
2. Associate an URL to your class
3. Override the doGet (or doPost, doHead...)
4. Use the HttpServletRequest parameter to read the HTTP request and extract info
5. Use the HttpServletResponse parameter to write the HTTP response
6. Deploy the Servlet in a servlet container (a suitable web server)

Q

How can we deploy a Servlet with IntelliJ ?

Apache Tomcat

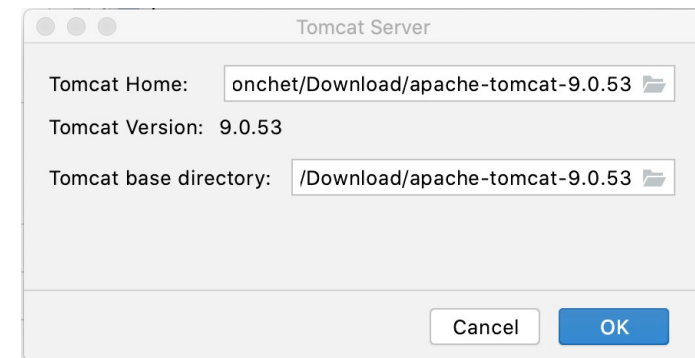
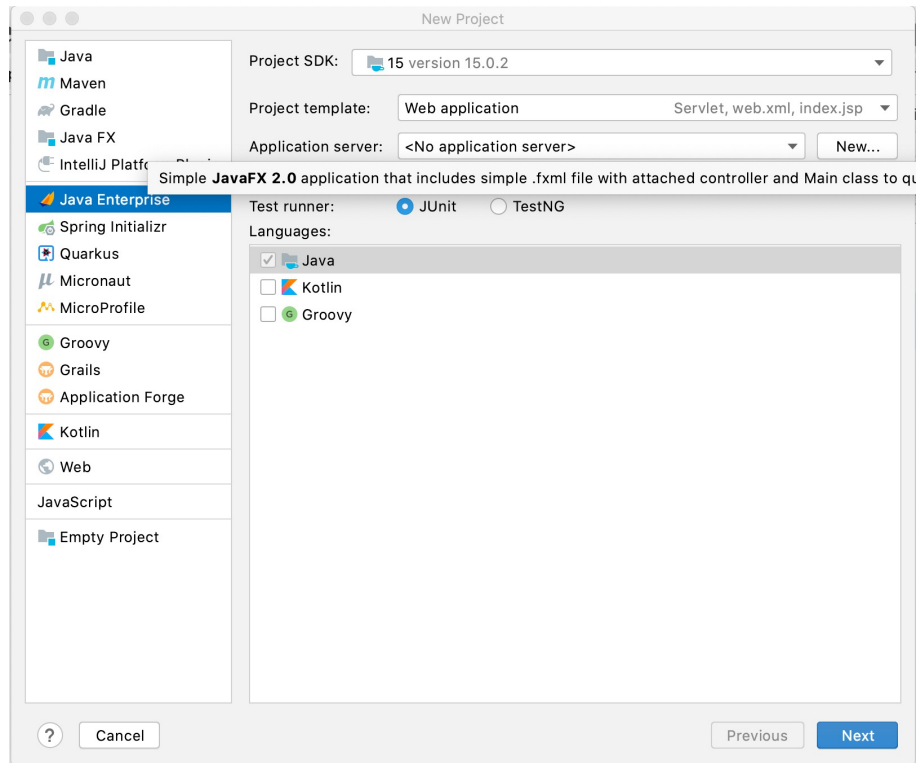
<https://tomcat.apache.org/whichversion.html>

Apache Tomcat Versions

Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Authentication (JASIC) Spec	Apache Tomcat Version	Latest Released Version	Supported Java Versions
6.0	3.1	5.0	TBD	TBD	10.1.x	10.1.0-M5 (alpha)	11 and later
5.0	3.0	4.0	2.0	2.0	10.0.x	10.0.11	8 and later
4.0	2.3	3.0	1.1	1.1	9.0.x	9.0.53	8 and later
3.1	2.3	3.0	1.1	1.1	8.5.x	8.5.71	7 and later
3.1	2.3	3.0	1.1	N/A	8.0.x (superseded)	8.0.53 (superseded)	7 and later
3.0	2.2	2.2	1.1	N/A	7.0.x (archived)	7.0.109 (archived)	6 and later (7 and later for WebSocket)
2.5	2.1	2.1	N/A	N/A	6.0.x (archived)	6.0.53 (archived)	5 and later
2.4	2.0	N/A	N/A	N/A	5.5.x (archived)	5.5.36 (archived)	1.4 and later
2.3	1.2	N/A	N/A	N/A	4.1.x (archived)	4.1.40 (archived)	1.3 and later
2.2	1.1	N/A	N/A	N/A	3.3.x (archived)	3.3.2 (archived)	1.1 and later

SELECT VERSION 9 (Why?)

Creating new Project - 1



Creating new project -2

New Project

Version: **Java EE 8**

Libraries and Frameworks:

- ☐ JSON Processing (JSON-P) (1.1.4)
- ☐ Message Service (JMS) (2.0.1)
- ☐ Model View Controller (MVC) (1.0.0)
- ☐ Persistence (JPA) (2.2)
- ☐ RESTful Web Services (JAX-RS) (2.1.1)
- ☐ Security (1.0)
- ☐ Server Faces (JSF) (2.3)
- ☒ Servlet (4.0.1)
- ☐ Transaction (JTA) (1.3)
- ☐ WebSocket (1.1)
- ☐ XML Web Services (JAX-WS) (2.3.1)

▼ Implementations

- ☐ Apache BVal (2.0.5)
- ☐ Apache CXF REST Server (JAX-RS) (3.3.7)
- ☐ Apache CXF XML Web Service (JAX-WS) (3.3.7)

Full Platform
Includes most of Java EE specifications (JSR 366).
Compatible servers: GlassFish 5.x, WebSphere AS Liberty 18.0.0.2, Wildfly 14.x, JBoss Enterprise AP 7.3, WebLogic Server 14.1.1.0.

[Web site](#) [Specification](#)

? Cancel Previous Next

New Project

Name: **MyFirstServletProject**

Location: **~/Download/MyFirstServletProject**

Artifact Coordinates

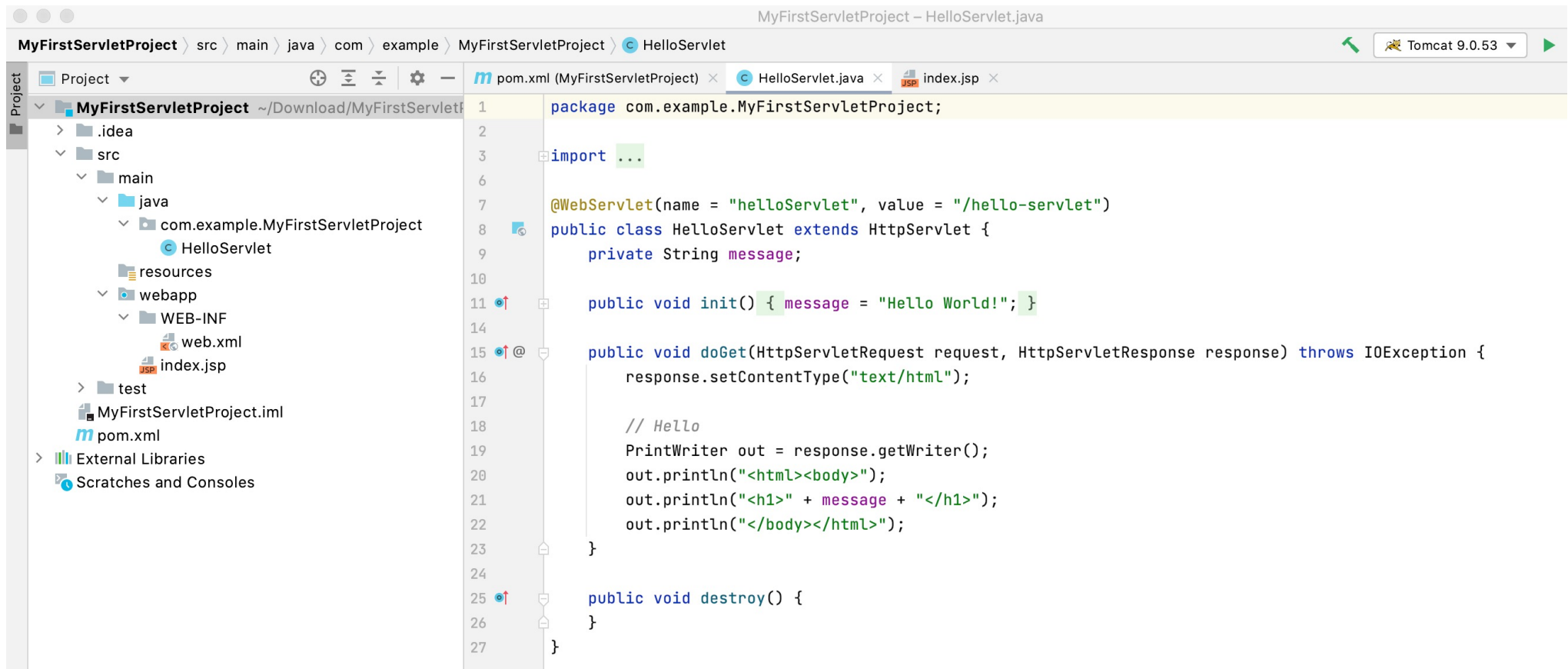
Group: **com.example**
The name of the artifact group, usually a company domain

Artifact: **MyFirstServletProject**
The name of the artifact within the group, usually a project name

Version: **1.0-SNAPSHOT**

? Cancel Previous Finish

Creating new project - 3



The screenshot shows an IDE window titled 'MyFirstServletProject - HelloServlet.java'. The breadcrumb navigation at the top indicates the path: 'MyFirstServletProject > src > main > java > com > example > MyFirstServletProject > HelloServlet'. The left sidebar displays the project structure, including folders like '.idea', 'src', 'main', 'resources', 'webapp', and 'WEB-INF', along with files like 'pom.xml', 'MyFirstServletProject.iml', 'web.xml', and 'index.jsp'. The main editor area shows the code for 'HelloServlet.java'.

```
1 package com.example.MyFirstServletProject;
2
3 import ...
4
5
6
7 @WebServlet(name = "helloServlet", value = "/hello-servlet")
8 public class HelloServlet extends HttpServlet {
9     private String message;
10
11     public void init() { message = "Hello World!"; }
12
13
14
15     public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
16         response.setContentType("text/html");
17
18         // Hello
19         PrintWriter out = response.getWriter();
20         out.println("<html><body>");
21         out.println("<h1>" + message + "</h1>");
22         out.println("</body></html>");
23     }
24
25     public void destroy() {
26     }
27 }
```

Index.jsp

For now, let's consider it as a simple HTML page

```
m pom.xml (MyFirstServletProject) x HelloServlet.java x JSP index.jsp x
1 <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
2 <!DOCTYPE html>
3 <html>
4 <head>
5     <title>JSP - Hello World</title>
6 </head>
7 <body>
8 <h1><%= "Hello World!" %>
9 </h1>
10 <br/>
11 <a href="hello-servlet">Hello Servlet</a>
12 </body>
13 </html>
```

HelloServlet.java

```
1 package com.example.MyFirstServletProject;
2
3 import ...
4
5
6
7 @WebServlet(name = "helloServlet", value = "/hello-servlet")
8 public class HelloServlet extends HttpServlet {
9     private String message;
10
11     public void init() { message = "Hello World!"; }
12
13
14
15     public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
16         response.setContentType("text/html");
17
18         // Hello
19         PrintWriter out = response.getWriter();
20         out.println("<html><body>");
21         out.println("<h1>" + message + "</h1>");
22         out.println("</body></html>");
23     }
24
25     public void destroy() {
26     }
27 }
```

if necessary...

- 1) cd into your tomcat bin directory
- 2) make all .sh files executable

```
MR-MacBookPro:~ ronchet$ cd /Users/ronchet/Download/apache-tomcat-9.0.53/bin
MR-MacBookPro:bin ronchet$ ls -la
total 1760
drwxr-xr-x@ 29 ronchet  staff    928 Sep  6 23:09 .
drwxr-xr-x@ 17 ronchet  staff    544 Sep 16 10:36 ..
-rw-r--r--@  1 ronchet  staff  34705 Sep  6 23:09 bootstrap.jar
-rw-r--r--@  1 ronchet  staff   1703 Sep  6 23:09 catalina-tasks.xml
-rw-r--r--@  1 ronchet  staff  16840 Sep  6 23:09 catalina.bat
-rw-r--r--@  1 ronchet  staff  25294 Sep  6 23:09 catalina.sh
-...
-rw-r--r--@  1 ronchet  staff   2026 Sep  6 23:09 version.bat
-rw-r--r--@  1 ronchet  staff   1908 Sep  6 23:09 version.sh
MR-MacBookPro:bin ronchet$ chmod a+x *.sh
MR-MacBookPro:bin ronchet$
```



Q

How do we deploy a WebApp manually ?

Create a user

- Make sure IntelliJ is quit.
- edit file [TOMCAT-HOME]/conf/tomcat-users.xml
- towards the end, add a line like

```
49 <!--
50 <role rolename="tomcat"/>
51 <role rolename="role1"/>
52 <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
53 <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
54 <user username="role1" password="<must-be-changed>" roles="role1"/>
55 -->
56 <user username="ronchet" password="secret" roles="standard,manager-script,manager-gui" />
57 </tomcat-users>
58
```

- start tomcat:
- cd file [TOMCAT-HOME]/bin
- ./startup.sh

```
[MR-MacBookPro:bin ronchet$ ./startup.sh
Using CATALINA_BASE:   /Users/ronchet/Download/apache-tomcat-9.0.53
Using CATALINA_HOME:   /Users/ronchet/Download/apache-tomcat-9.0.53
Using CATALINA_TMPDIR: /Users/ronchet/Download/apache-tomcat-9.0.53/temp
Using JRE_HOME:        /Users/ronchet/Library/Java/JavaVirtualMachines/adopt-o
njdk-15.0.2/Contents/Home
Using CLASSPATH:        /Users/ronchet/Download/apache-tomcat-9.0.53/bin/bootstra
p.jar:/Users/ronchet/Download/apache-tomcat-9.0.53/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
MR-MacBookPro:bin ronchet$
```


Tomcat is running...

localhost:8080

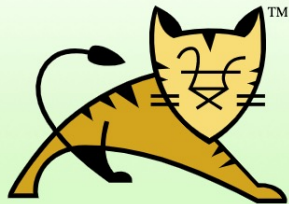
[Home](#) [Documentation](#) [Configuration](#) [Examples](#) [Wiki](#) [Mailing Lists](#)

[Find Help](#)

Apache Tomcat/9.0.53



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:

[Security Considerations How-To](#)

[Manager Application How-To](#)

[Clustering/Session Replication How-To](#)

[Server Status](#)

[Manager App](#)

[Host Manager](#)

Developer Quick Start

[Tomcat Setup](#)

[First Web Application](#)

[Realms & AAA](#)

[JDBC DataSources](#)

[Examples](#)

[Servlet Specifications](#)

[Tomcat Versions](#)

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

Documentation

[Tomcat 9.0 Documentation](#)

[Tomcat 9.0 Configuration](#)

Getting Help

[FAQ](#) and [Mailing Lists](#)

The following mailing lists are available:

to shut it down:

```
cd file [TOMCAT-HOME]/bin  
./shutdown.sh
```



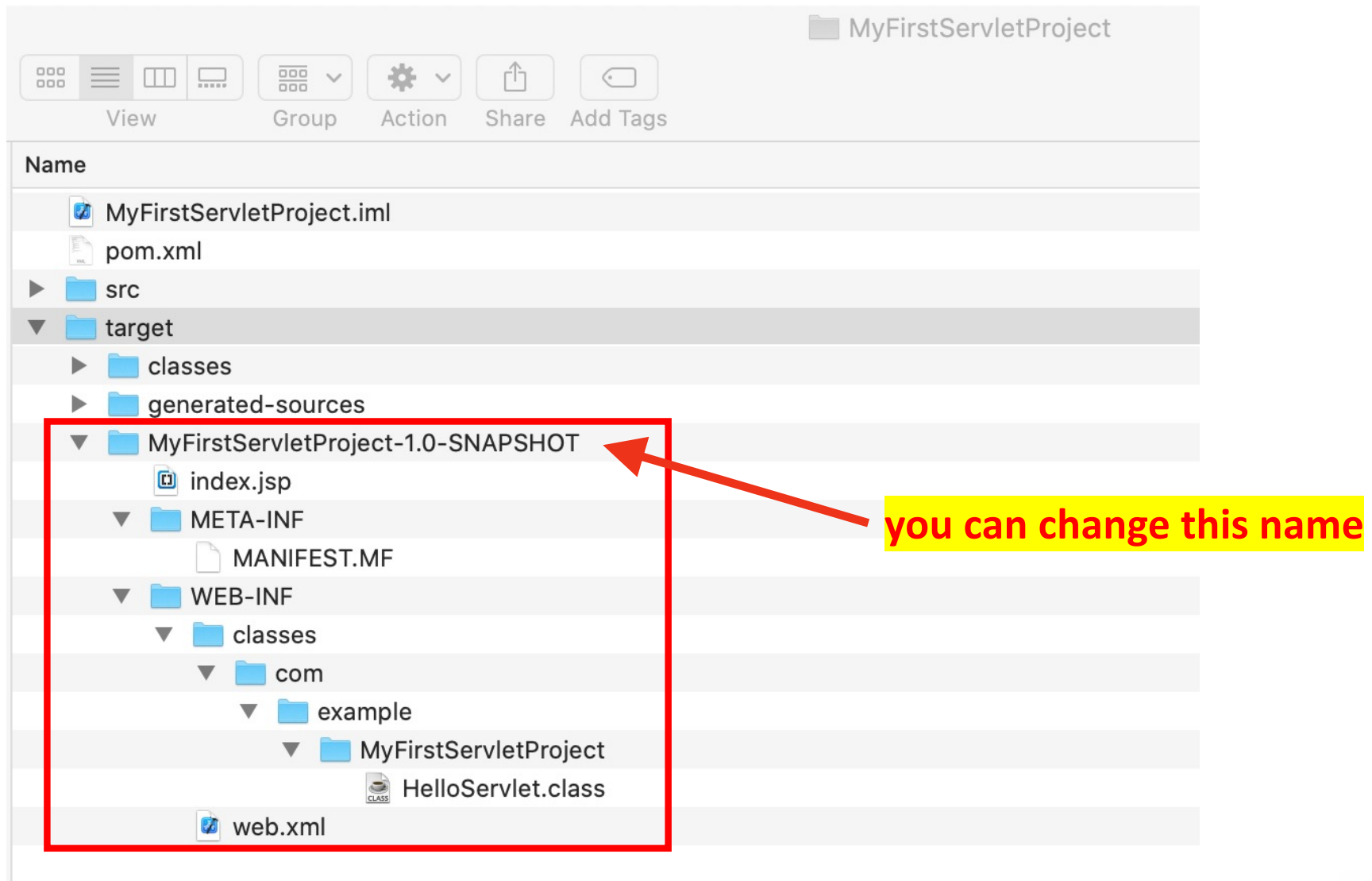
Servlet Deployment

- By default, a servlet application is located at the path
`<Tomcat-installationdirectory>/webapps/ROOT`
and the class file would reside in
`<Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.`

If you have a fully qualified class name
of **com.myorg.MyServlet**, then this servlet class must
be located in
`WEB-INF/classes/com/myorg/MyServlet.class.`



Servlet Deployment



Our app is deployed!



Tomcat Web Application Manager

Message: OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes
/MyFirstServletProject	None specified		true	1	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes
/docs	None specified	Tomcat Documentation	true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes
/examples	None specified	Servlet and JSP Examples	true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes
/manager	None specified	Tomcat Manager Application	true	2	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes

E

**Example: let's write a Servlet
that does not use parameters**

```
package it.unitn.disi.ronchet.myservlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "ReadPost", urlPatterns = {"/ReadPost"})
public class ReadPost extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

```

/**
 * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
 * methods.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html><head>");
        out.println("<title>Servlet ReadPost</title>");
        out.println("</head><body>");
        out.println("<h1>Servlet ReadPost at "+request.getContextPath()+"</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}

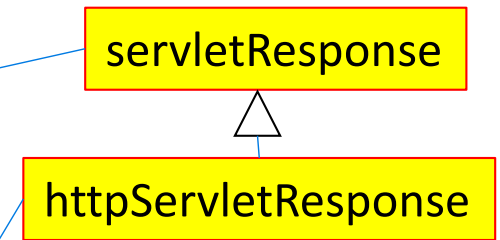
```



What can we use **httpServletResponse** for?

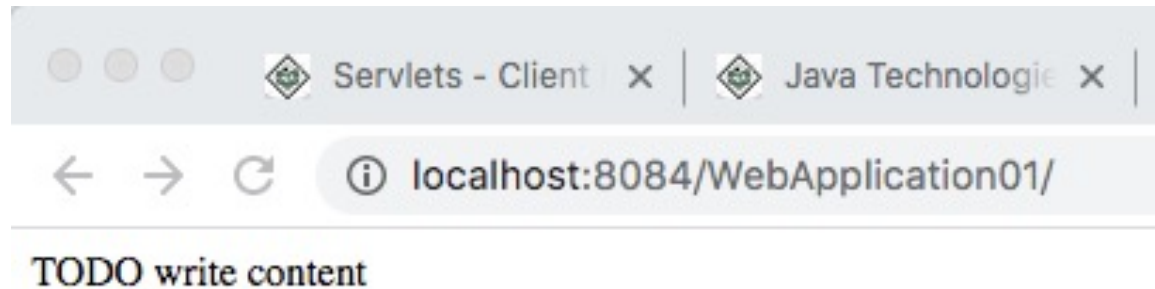
- `getWriter`
- `setCharacterEncoding`
- `setContentLength`
- `setContentType`
- `setLocale`

- `addHeader`
- `sendError`
- `sendRedirect`
- `setStatus`
- `Session`
- `addCookie`



See <https://docs.oracle.com/javaee/7/api/javax/servlet/ServletResponse.html>

Output



Servlet ReadPost at /WebApplication01

**Next step:
let's get the parameters
contained in the request**

E

**Example: let's write a Servlet
that reads the parameters contained in
the request**

Parameters passing

```
<html>
<body>
<form action="http://localhost:8084/WebApplication01/ReadPost" method="post">
  <label for="fname">First name:</label>
  <input type="text" name="fname"><br><br>
  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
</form>
</body>
</html>
```



form.html

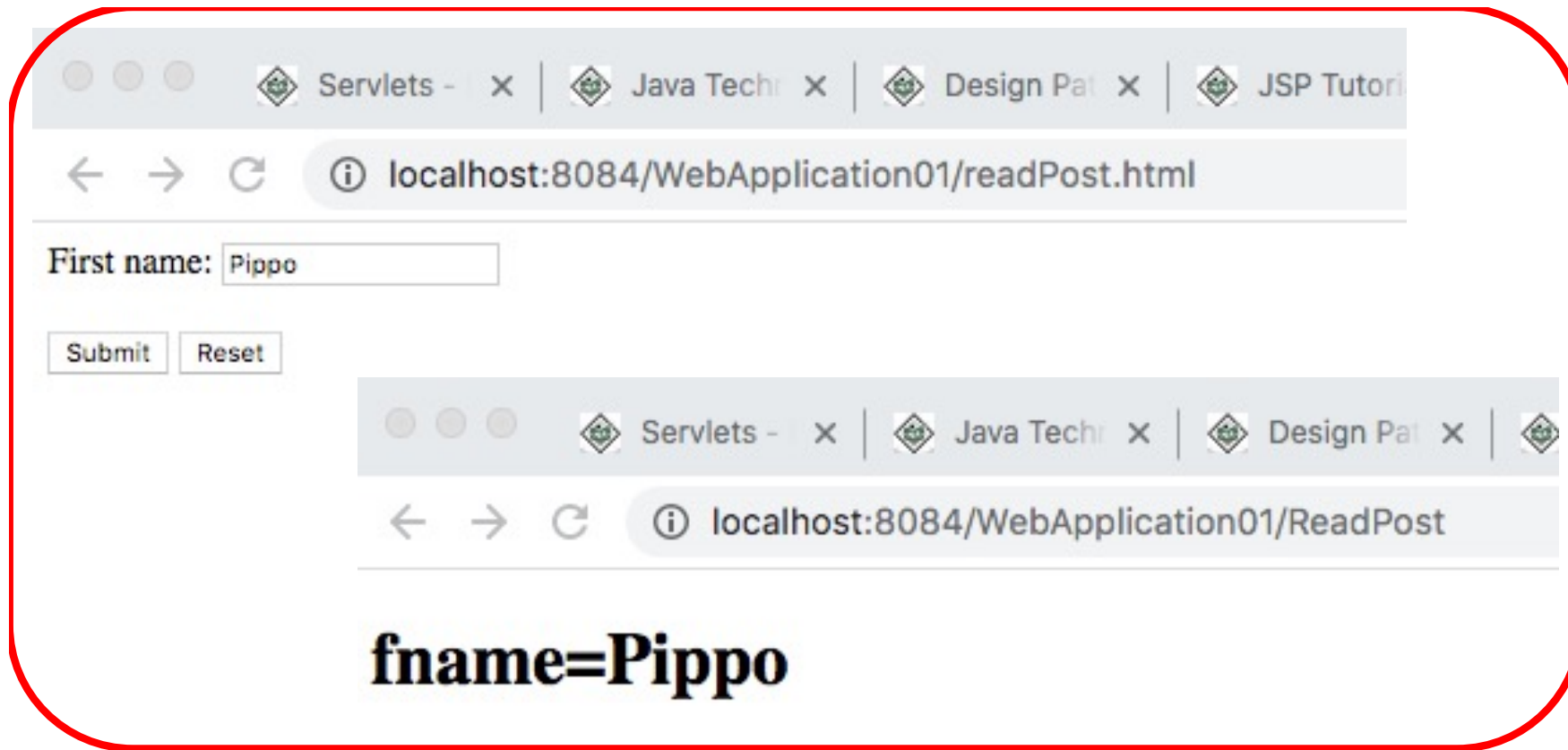
File | /Users/ronchet/Downloads/form.html

First name:

```

protected void processRequest(HttpServletRequest request,
                             HttpServletResponse response)
    throws ServletException, IOException {
    String name = request.getParameter("fname");
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html><head>");
        out.println("<title>Servlet ReadPost</title>");
        out.println("</head><body>");
        out.println("<h1> fname="+name+"</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}

```



Post

Servlets - x | Java Techn x | Design Pat x | JSP Tutori

localhost:8084/WebApplication01/readPost.html

First name:

localhost:8084/WebApplication01/ReadPost

fname=Pippo

Post

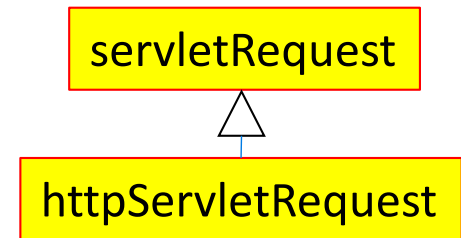
Get

Servlets - x | Java Techn x | Design Pat x | JSP Tutori x

localhost:8084/WebApplication01/ReadPost?fname=Minnie

fname=Minnie

Getting parameters from **HttpServletRequest**:



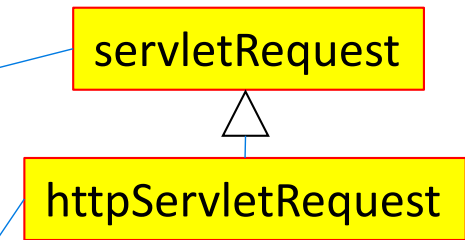
String	getParameter(String name) Returns the value of a request parameter as a <code>String</code> , or <code>null</code> if the parameter does not exist.
Map<String,String[]>	getParameterMap() Returns a <code>java.util.Map</code> of the parameters of this request.
Enumeration<String>	getParameterNames() Returns an <code>Enumeration</code> of <code>String</code> objects containing the names of the parameters contained in this request.
String[]	getParameterValues(String name) Returns an array of <code>String</code> objects containing all of the values the given request parameter has, or <code>null</code> if the parameter does not exist.

See <https://docs.oracle.com/javaee/7/api/javax/servlet/ServletRequest.html>

What else can I obtain from **HttpServletRequest**?

- Character encoding
- Content length
- Content Type
- Locale
- Protocol
- Local/remote name, address, port
- Server name
- Is Secure ?

- Headers
- Method
- Path Info
- Query String
- Session
- Cookie



See <https://docs.oracle.com/javaee/7/api/javax/servlet/ServletRequest.html>

Q

How do we factor out some code ?

Factoring out some HTML

- How can we avoid this horrible stuff?

```
out.println("<!DOCTYPE html>");  
out.println("<html>");  
out.println("<head>");  
out.println("<title>Servlet ReadPost</title>");  
out.println("</head>");  
out.println("<body>");  
out.println("<h1> fname="+name+"</h1>");  
out.println("</body>");  
out.println("</html>");
```

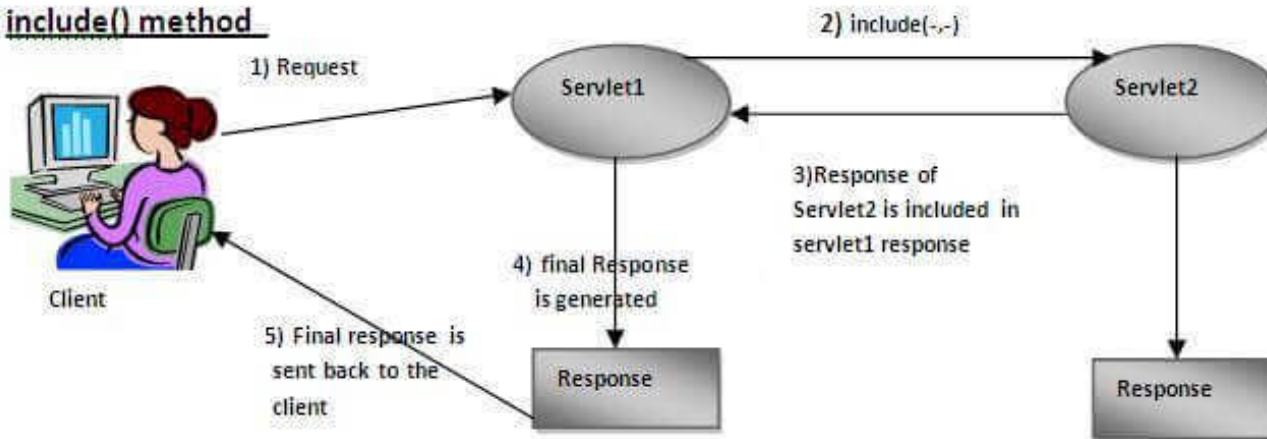
Factoring out some HTML

```
....xm ReadPost.java x Demo1.java x fragment2.html x fragment1.html x Counter.java x readPost.html x
1      <!DOCTYPE html>
2      <html>
3      <head>
4          <title>TODO supply a title</title>
5          <meta charset="UTF-8">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      </head>
8      <body>
9          <h2>fragment 1</h2>
```

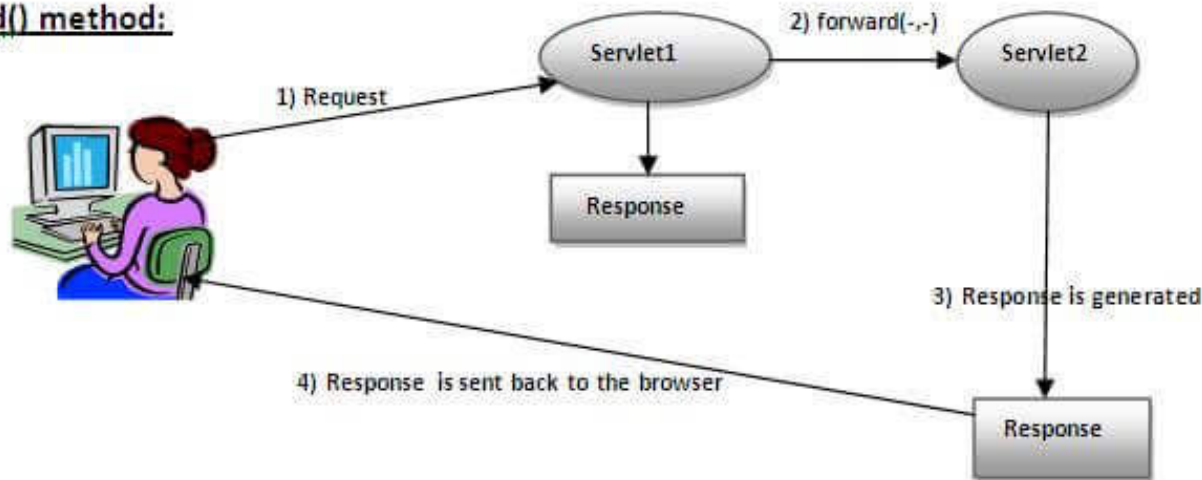
```
....xm ReadPost.java x Demo1.java x fragment2.html x
💡      <h2>fragment 2</h2>
2      </body>
3      </html>
4
```

Include vs forward

include() method



forward() method:



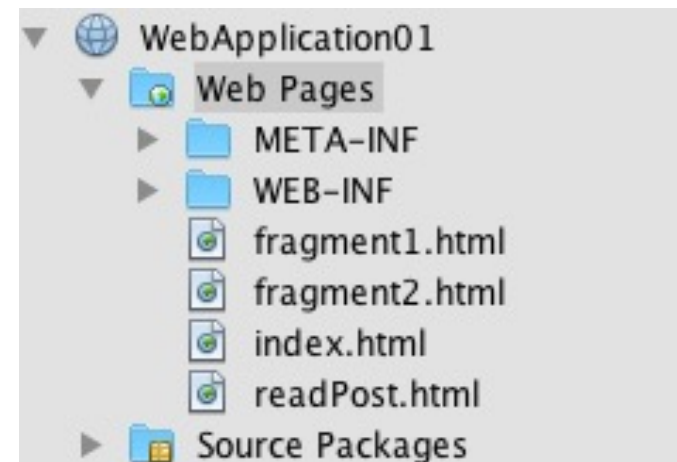
images from
[/www.javatpoint.com](http://www.javatpoint.com)

example: see <https://www.javatpoint.com/requestdispatcher-in-servlet>

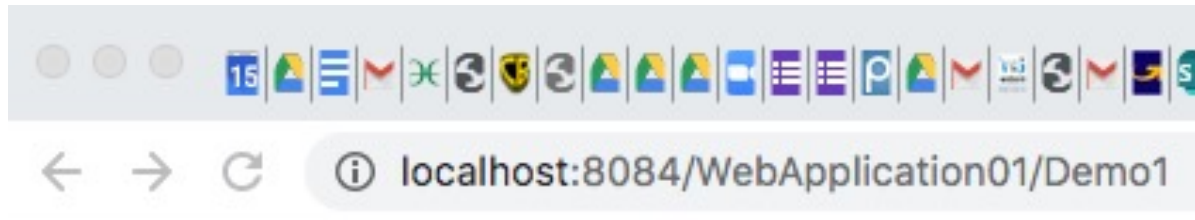


@Override

```
protected void doGet(HttpServletRequest request,
HttpServletRequest response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        request.getRequestDispatcher("/fragment1.html")
            .include(request, response);
        out.println("Servlet generated content");
        request.getRequestDispatcher("/fragment2.html")
            .include(request, response);
    }
}
```



output



fragment 1

Servlet generated content

fragment 2

Q

How do we monitor the execution?

```
public void log(String msg) {  
    getServletContext().log(msg);  
}
```

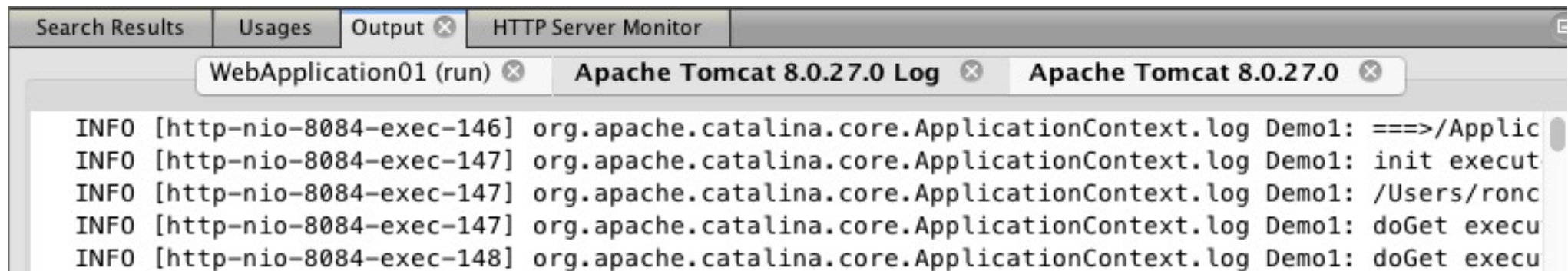
@Override

```
public void destroy() {  
    log("destroy executed");  
}
```

@Override

```
public void init() {  
    log("init executed");  
}
```

Logging




```
public void log(String msg) {  
    filterConfig.getServletContext().log(msg);  
}
```

Logging in filters

