Dynamic content: programming the web servers

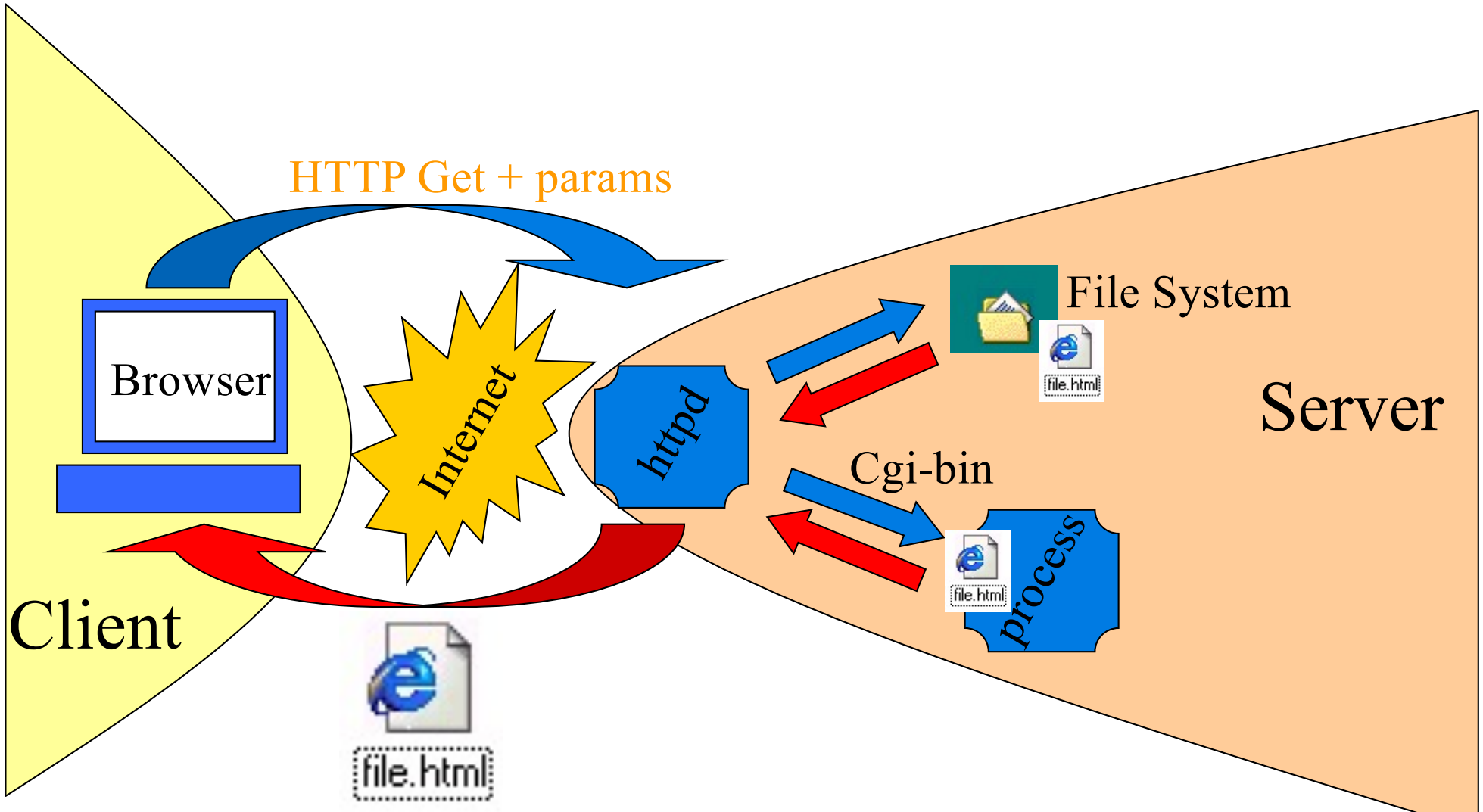
Step 2:
let’s use our web server to generate dynamic pages

Dynamic pages: the main idea:

We want to obtain NON STATIC information from the server. This implies executing some code on it, and send the results to the user.

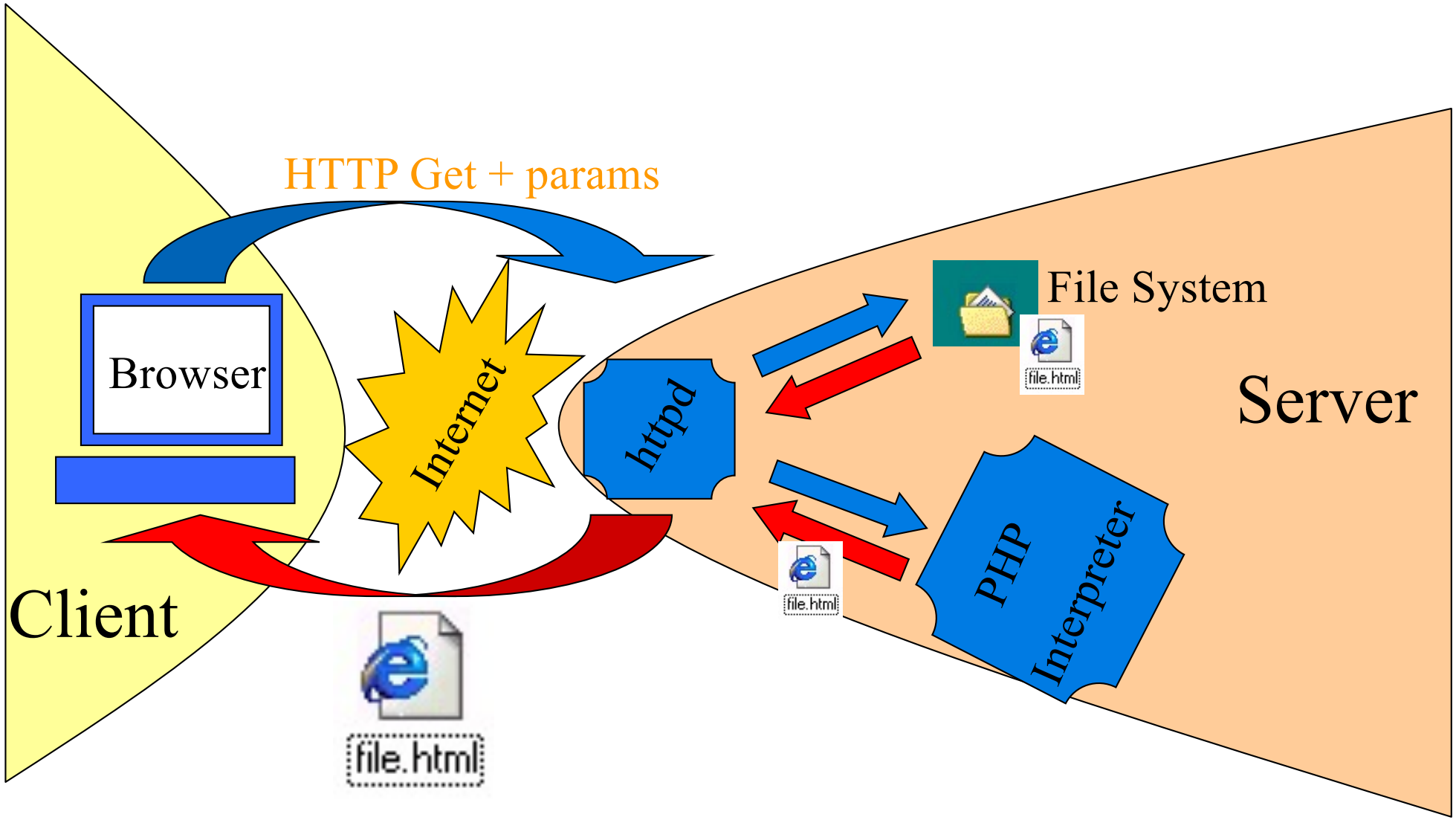
e.g.: what's the time?

The original web architecture: dynamic pages



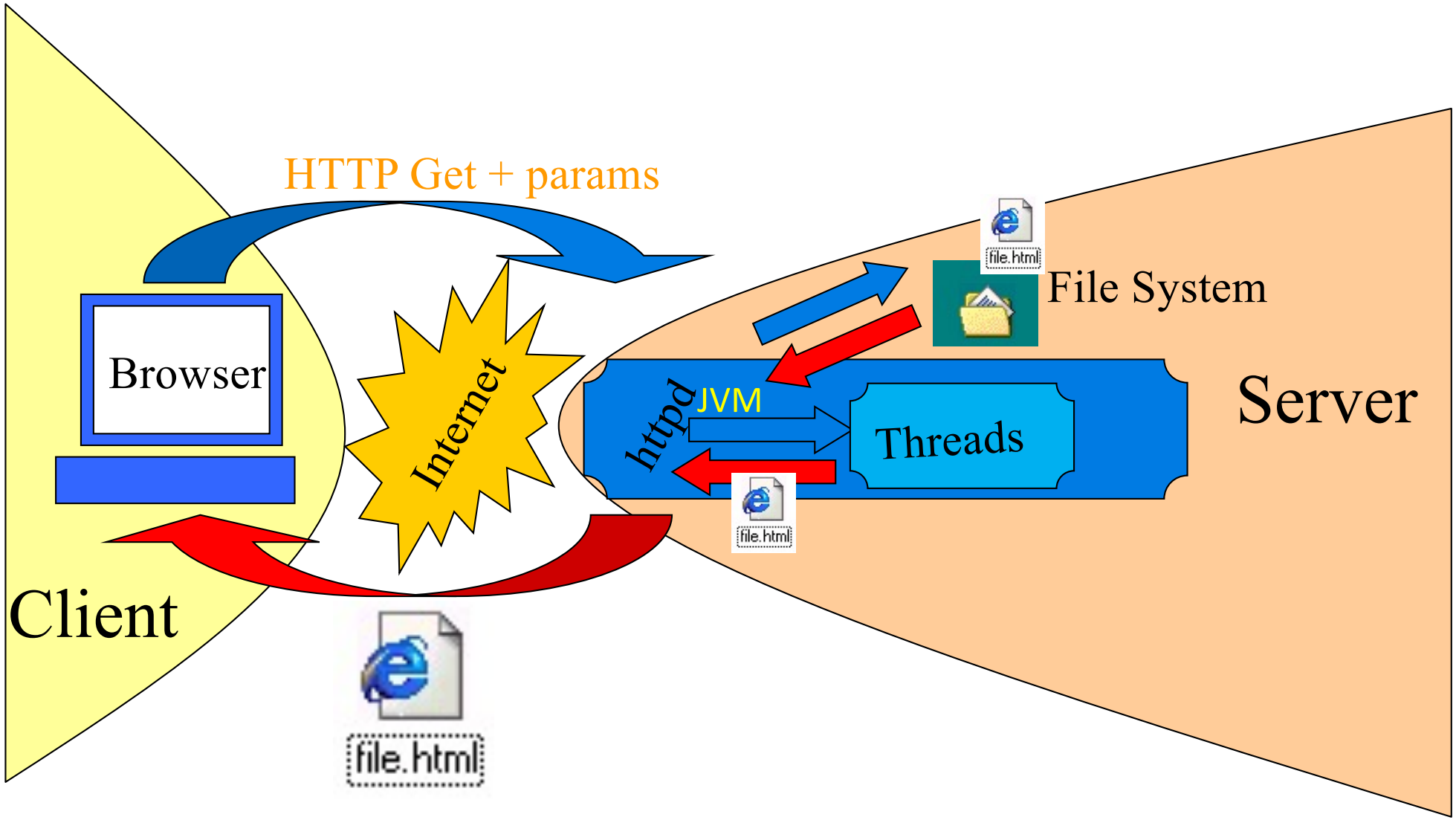
Evolution 1: dynamically create (interlinked) documents

The original web architecture: dynamic pages



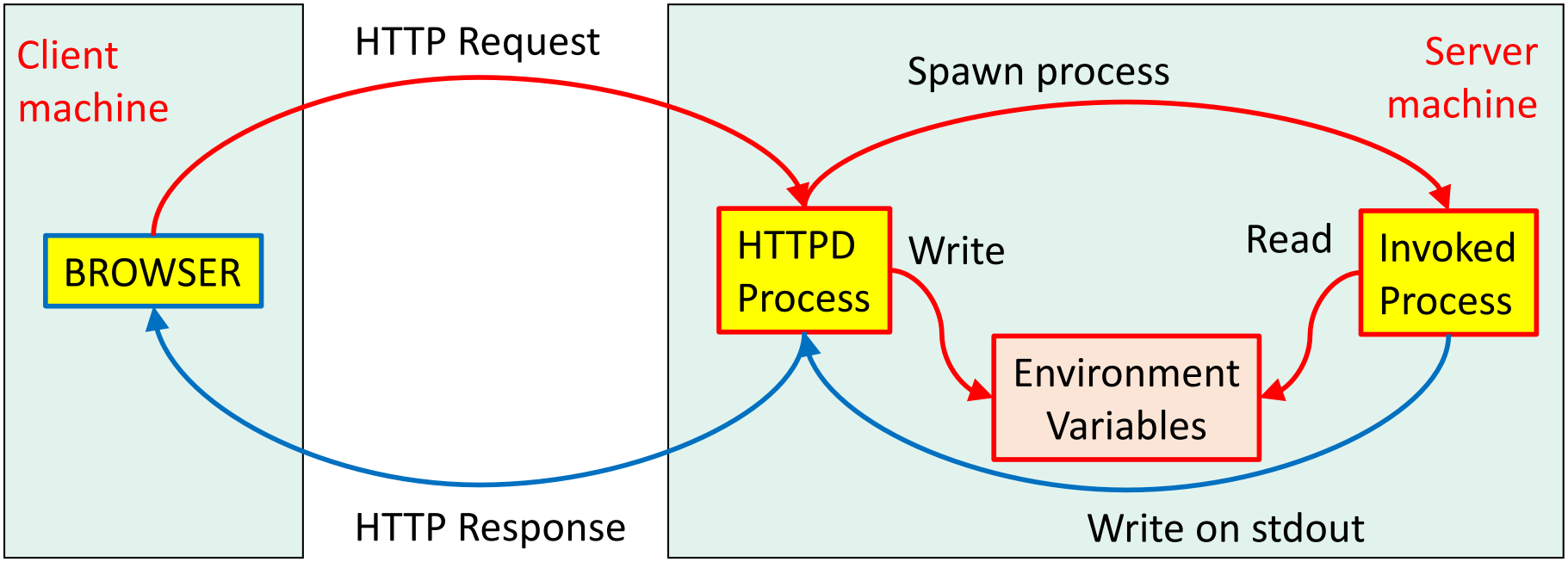
Evolution 1: dynamically create (interlinked) documents

The original web architecture: dynamic pages

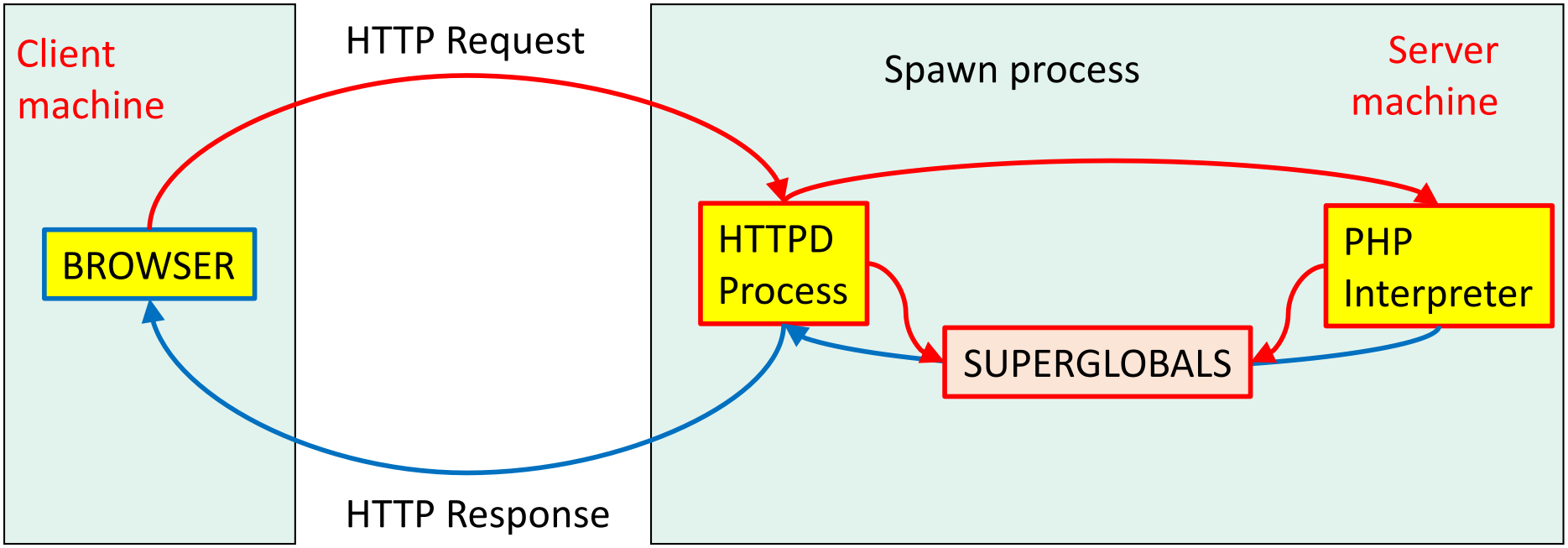


Evolution 1: dynamically create (interlinked) documents

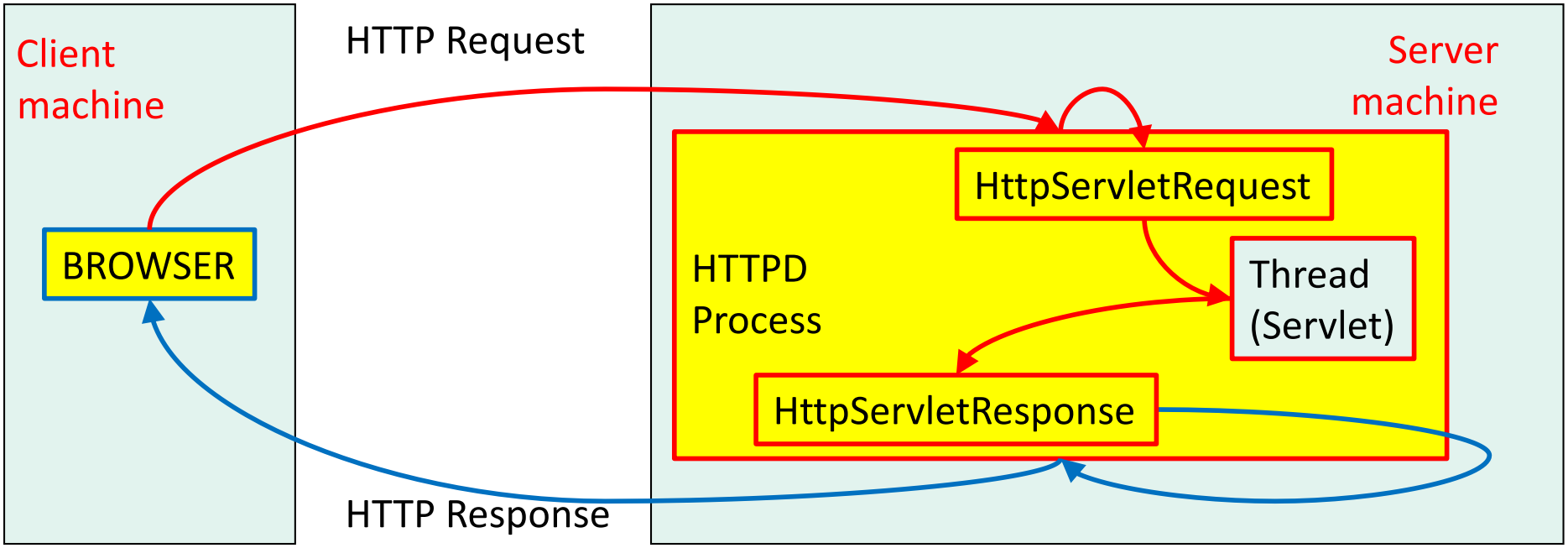
Getting info about the request



Getting info about the request



Getting info about the request



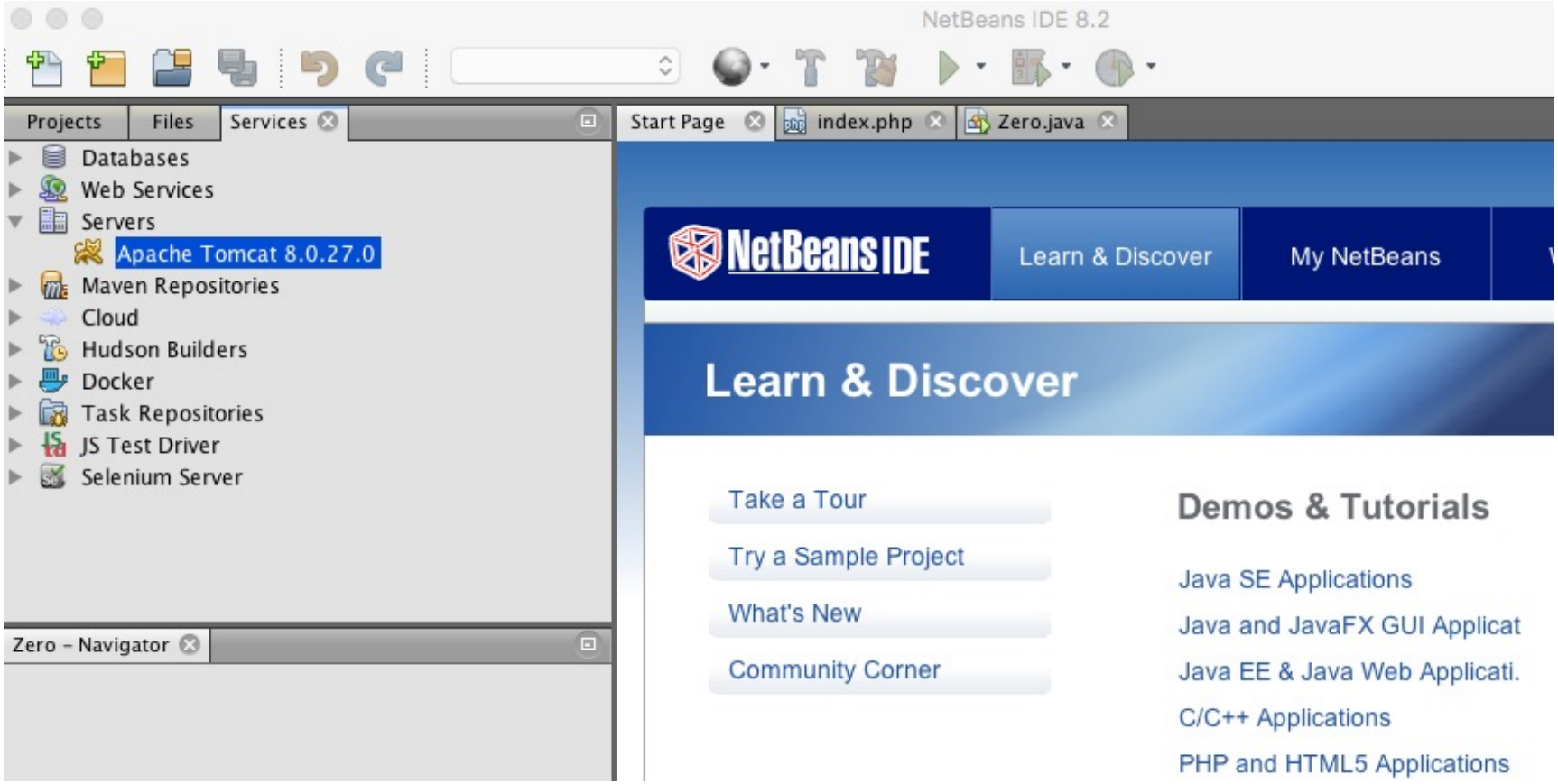
- Apache **Tomcat** is a **Java servlet container**, and is run on a **Java Virtual Machine**, or **JVM**.
- **Tomcat** utilizes the Java servlet specification to execute servlets generated by requests, often with the help of JSP pages, allowing dynamic content to be generated much more efficiently than with a CGI script.

Apache Tomcat

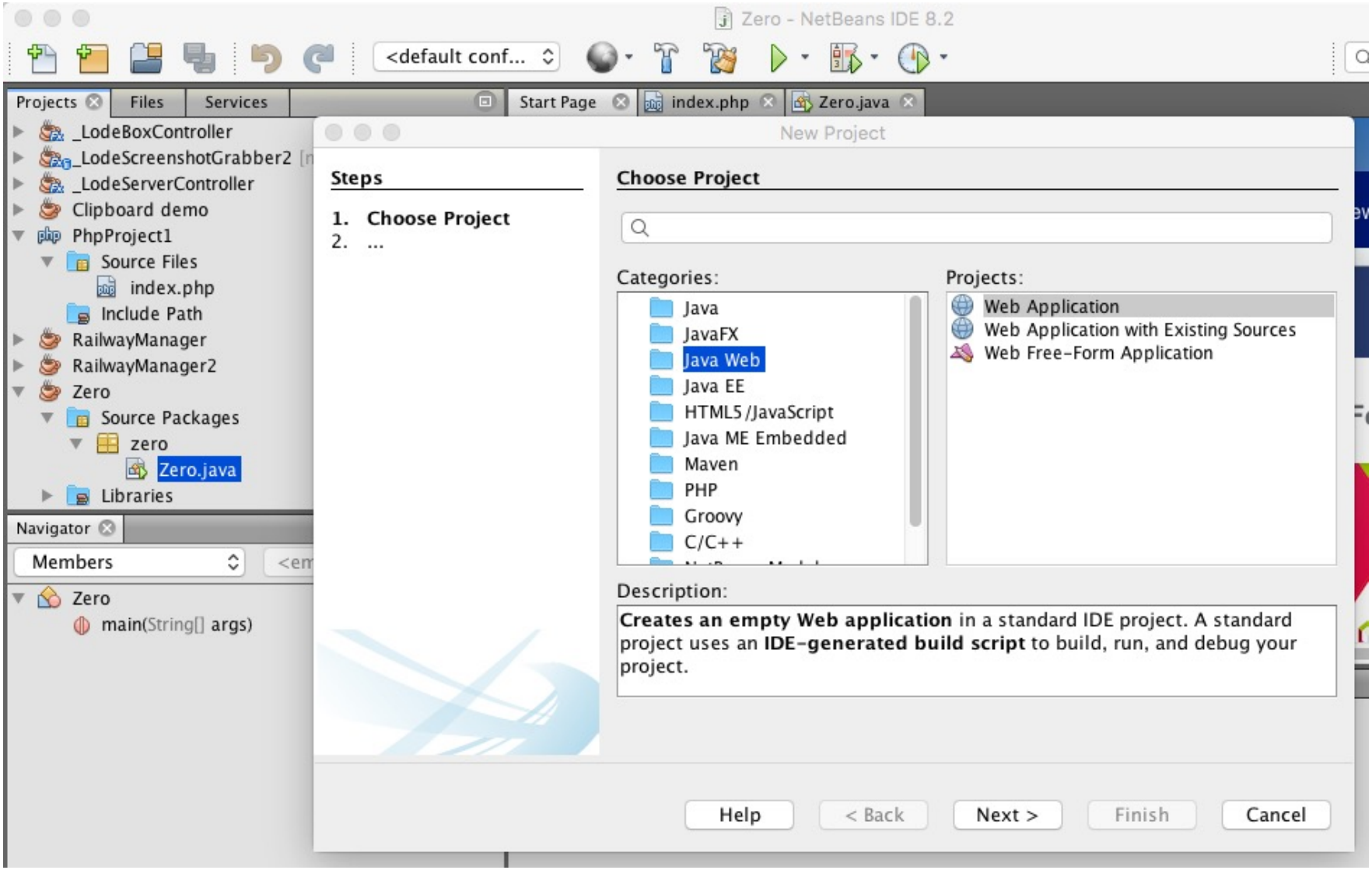
https://tomcat.apache.org/download-80.cgi

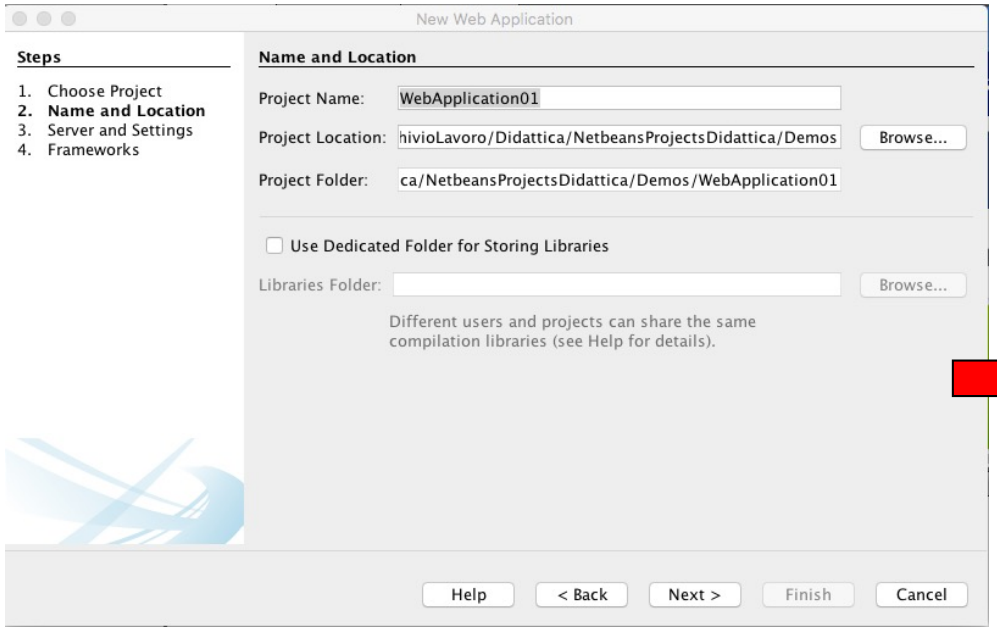
Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Authentication (JASIC) Spec	Apache Tomcat Version	Latest Released Version	Supported Java Versions
5.0	3.0	4.0	2.0	2.0	10.0.x	10.0.4	8 and later
4.0	2.3	3.0	1.1	1.1	9.0.x	9.0.44	8 and later
3.1	2.3	3.0	1.1	1.1	8.5.x	8.5.64	7 and later
3.1	2.3	3.0	1.1	N/A	8.0.x (superseded)	8.0.53 (superseded)	7 and later
3.0	2.2	2.2	1.1	N/A	7.0.x	7.0.108	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

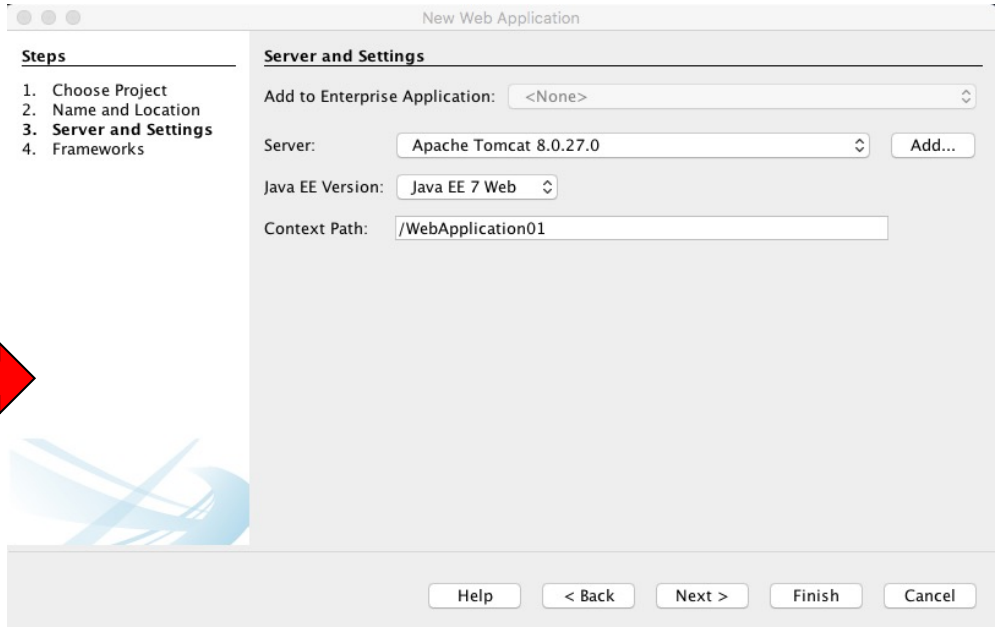
Apache Tomcat in Netbeans 8.2

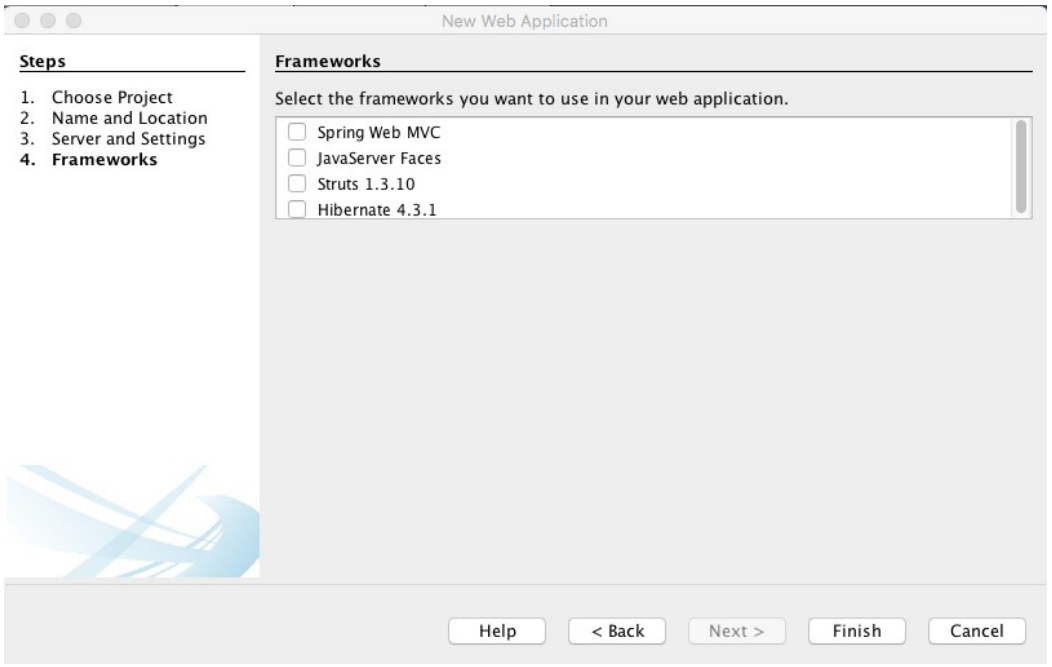


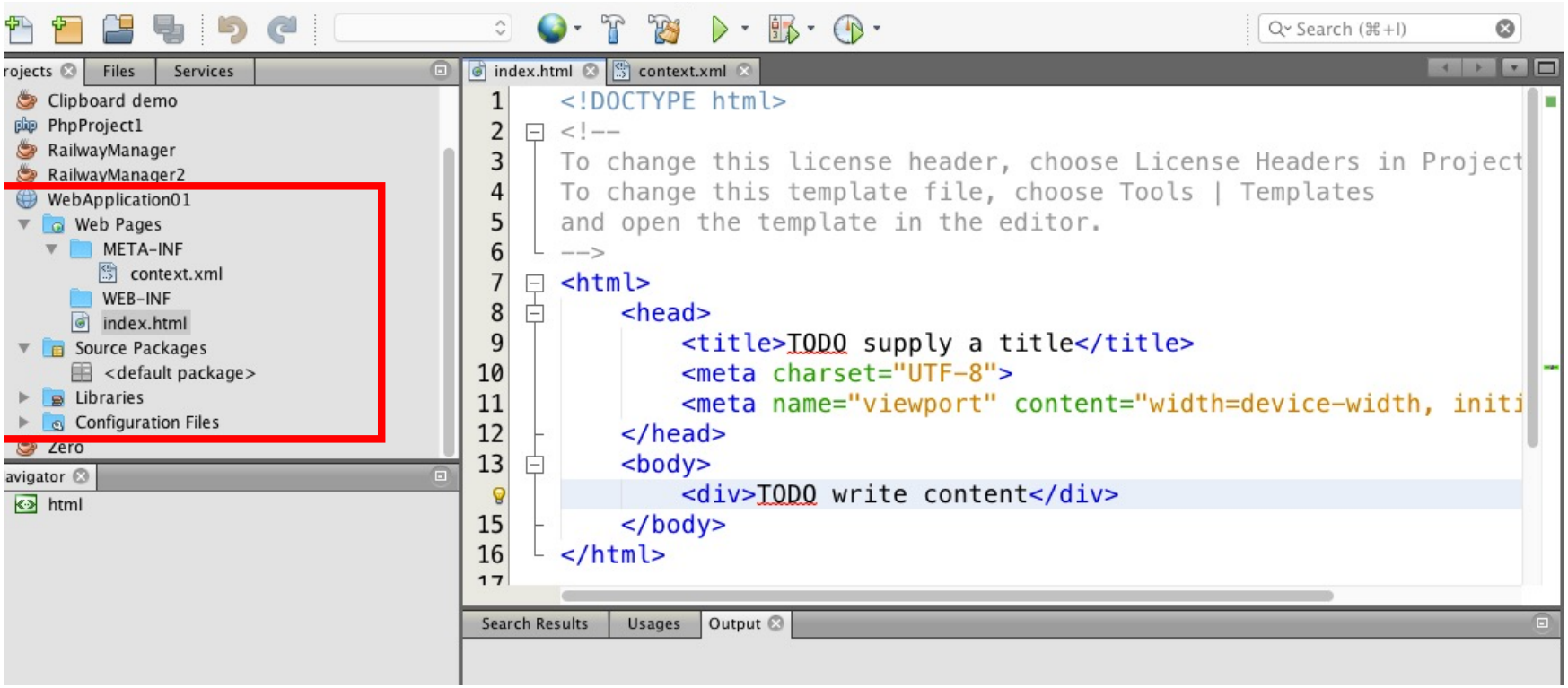
Create new web project

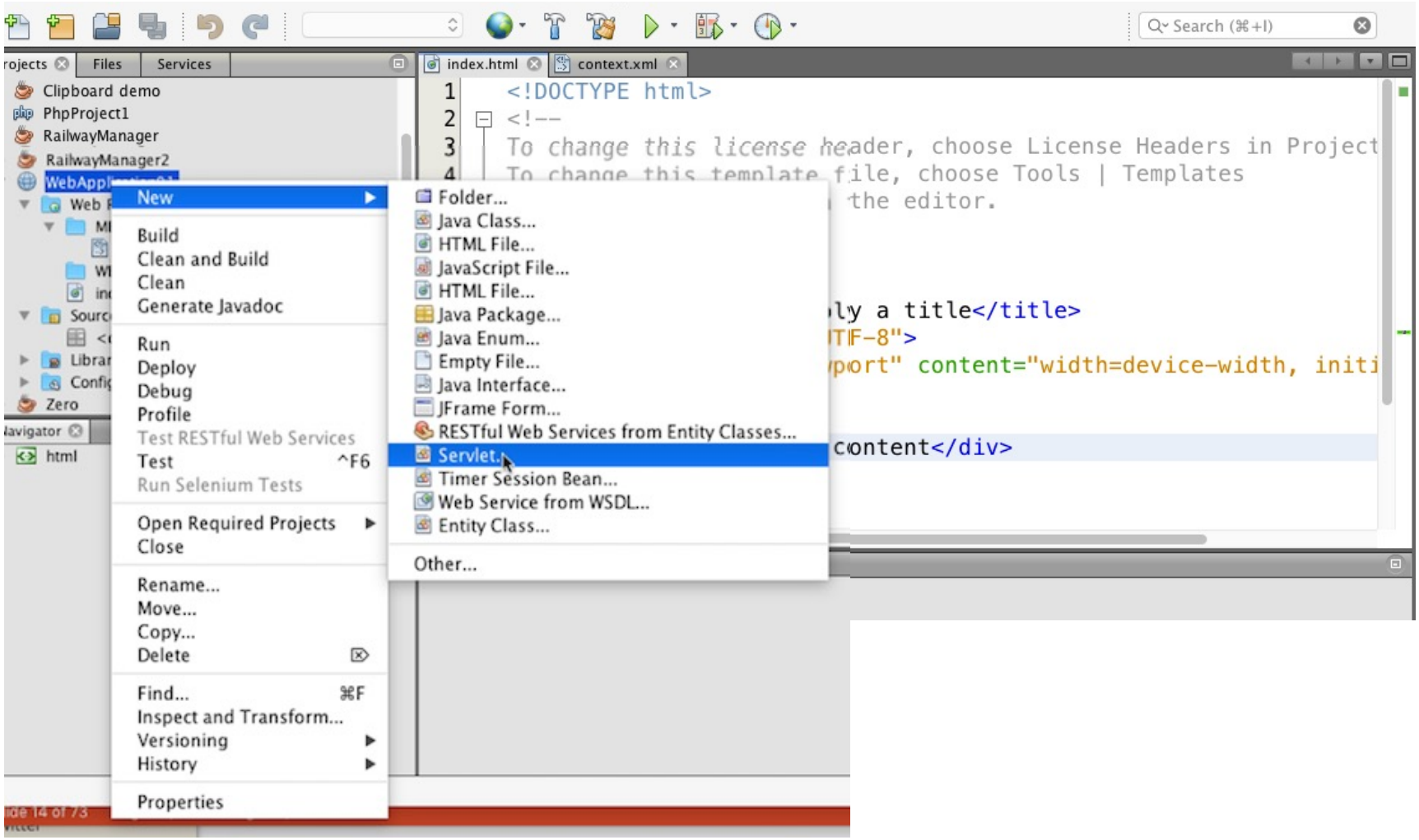


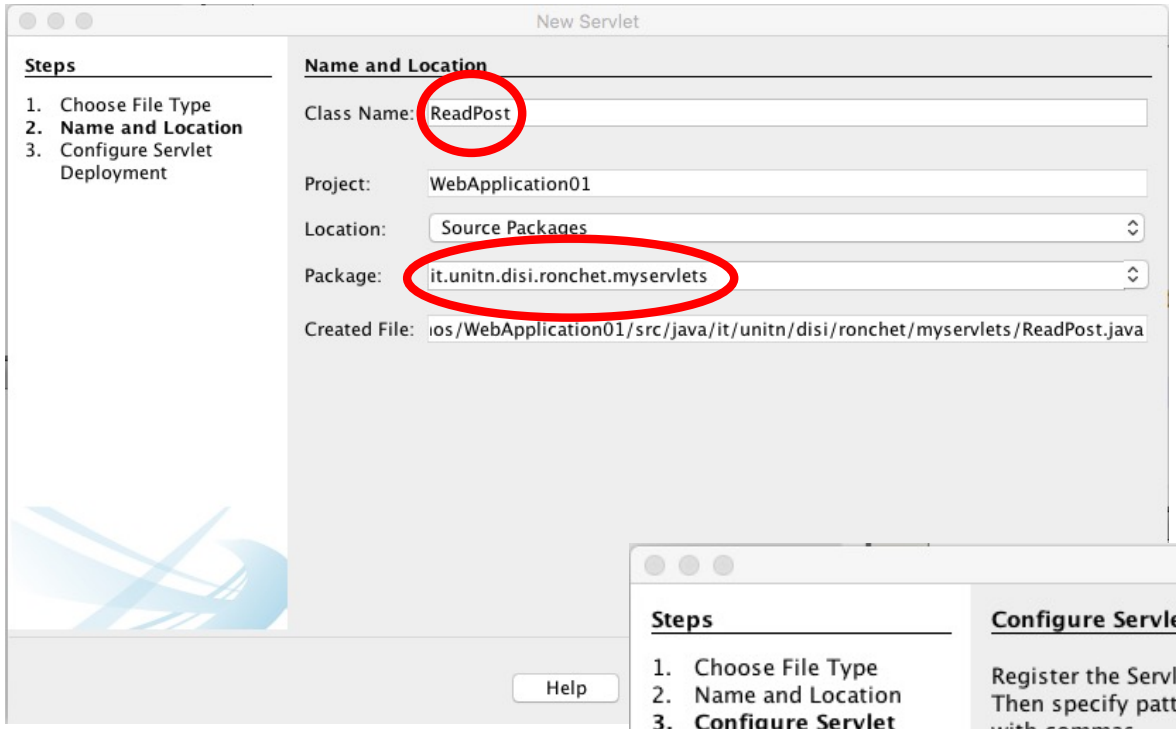

New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Name and Location
Project Name: WebApplication01
Project Location: hivioLavoro/Didattica/NetbeansProjectsDidattica/Demos
Browse...
Project Folder: ca/NetbeansProjectsDidattica/Demos/WebApplication01
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries (see Help for details).
Help
< Back
Next >
Finish
Cancel

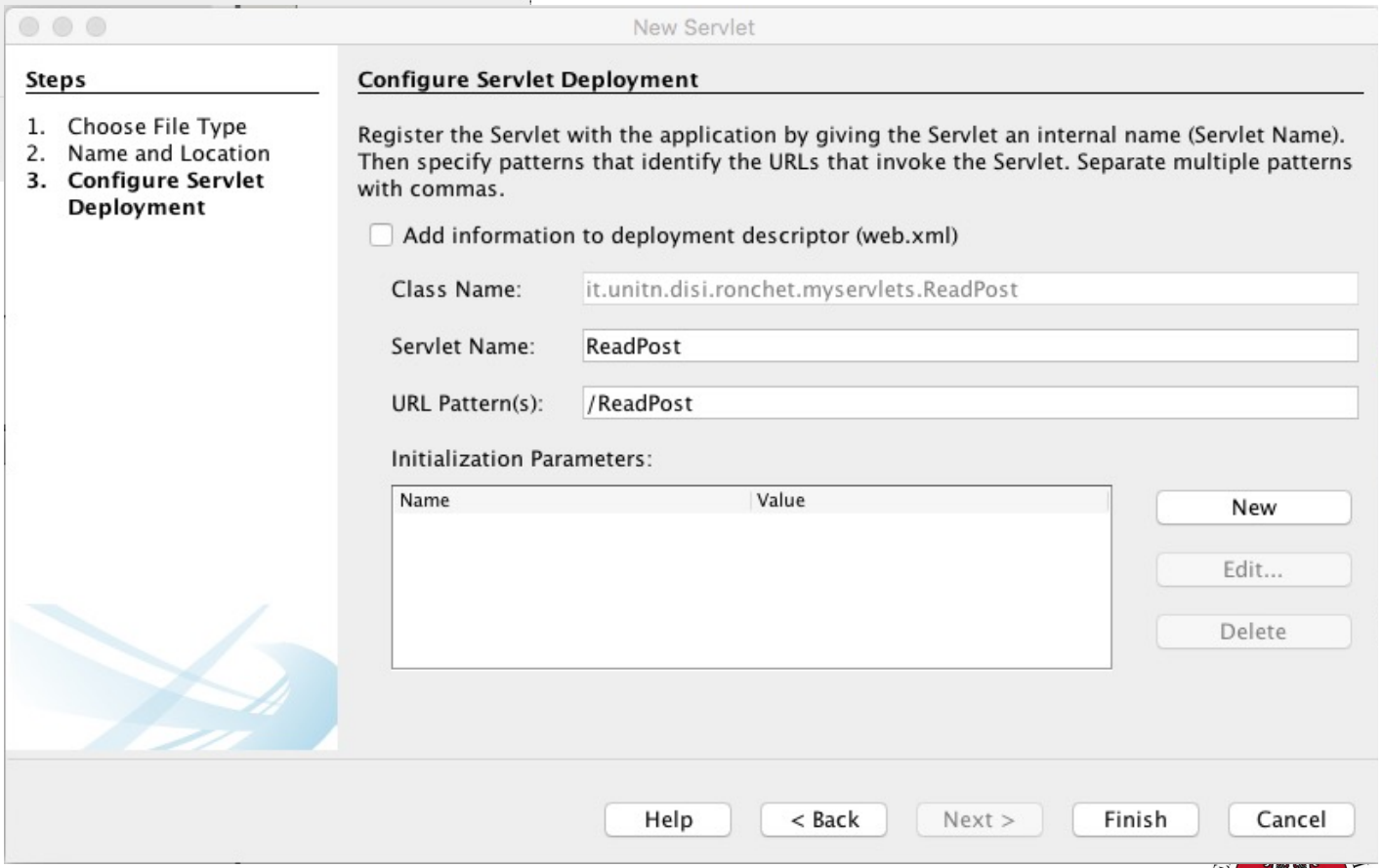

New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Server and Settings
Add to Enterprise Application: <None>
Server: Apache Tomcat 8.0.27.0
Add...
Java EE Version: Java EE 7 Web
Context Path: /WebApplication01
Help
< Back
Next >
Finish
Cancel


New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Frameworks
Select the frameworks you want to use in your web application.
Spring Web MVC
JavaServer Faces
Struts 1.3.10
Hibernate 4.3.1
Help
< Back
Next >
Finish
Cancel


Search (⌘+I)
Files
Services
Clipboard demo
PhpProject1
RailwayManager
RailwayManager2
WebApplication01
Web Pages
META-INF
context.xml
WEB-INF
index.html
Source Packages
<default package>
Libraries
Configuration Files
Zero
html
index.html
context.xml
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initi
</head>
<body>
TODO write content
</body>
</html>
Search Results
Usages
Output


Projects
Files
Services
Clipboard demo
PhpProject1
RailwayManager
RailwayManager2
New
Build
Clean and Build
Clean
Generate Javadoc
Run
Deploy
Debug
Profile
Test RESTful Web Services
Test ^F6
Run Selenium Tests
Open Required Projects
Close
Rename...
Move...
Copy...
Delete
Find... ⌘F
Inspect and Transform...
Versioning
History
Properties
Folder...
Java Class...
HTML File...
JavaScript File...
HTML File...
Java Package...
Java Enum...
Empty File...
Java Interface...
JFrame Form...
RESTful Web Services from Entity Classes...
Servlet...
Timer Session Bean...
Web Service from WSDL...
Entity Class...
Other...
index.html
context.xml
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project
To change this template file, choose Tools | Templates
the editor.
content


New Servlet
Steps
1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment
Name and Location
Class Name: ReadPost
Project: WebApplication01
Location: Source Packages
Package: it.unitn.disi.ronchet.myservlets
Created File: ios/WebApplication01/src/java/it/unitn/disi/ronchet/myservlets/ReadPost.java
Help

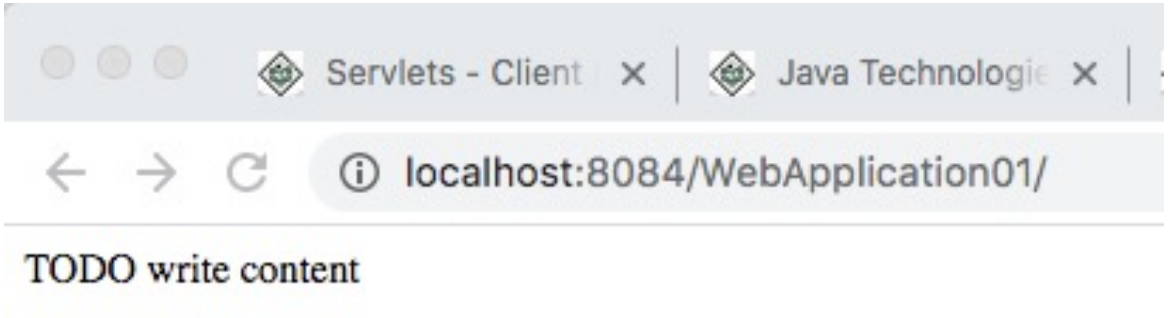

New Servlet
Steps
1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment
Configure Servlet Deployment
Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.
Add information to deployment descriptor (web.xml)
Class Name: it.unitn.disi.ronchet.myservlets.ReadPost
Servlet Name: ReadPost
URL Pattern(s): /ReadPost
Initialization Parameters:
Name
Value
New
Edit...
Delete
Help
< Back
Next >
Finish
Cancel

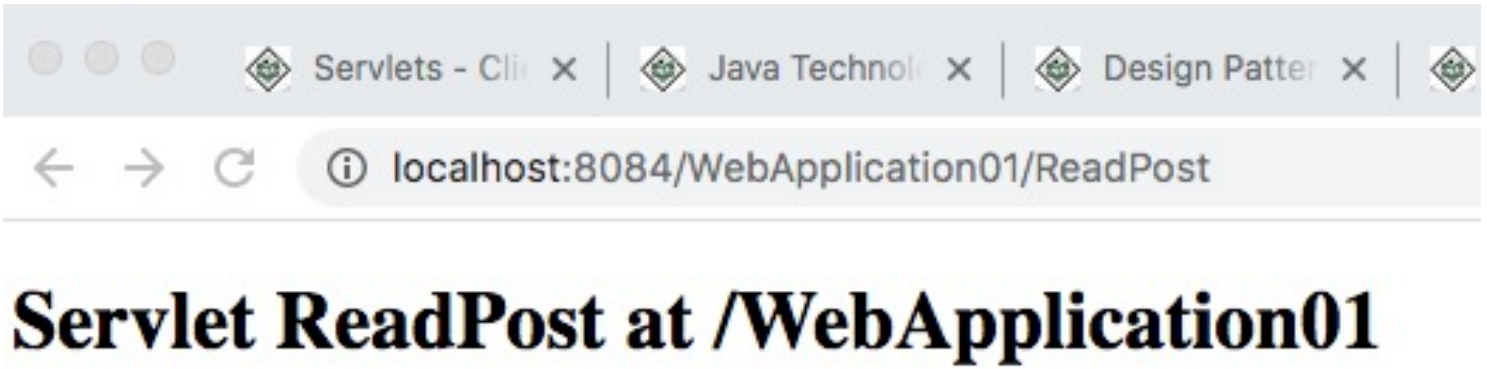
```java
package it.unitn.disi.ronchet.myservlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@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);
    }
```

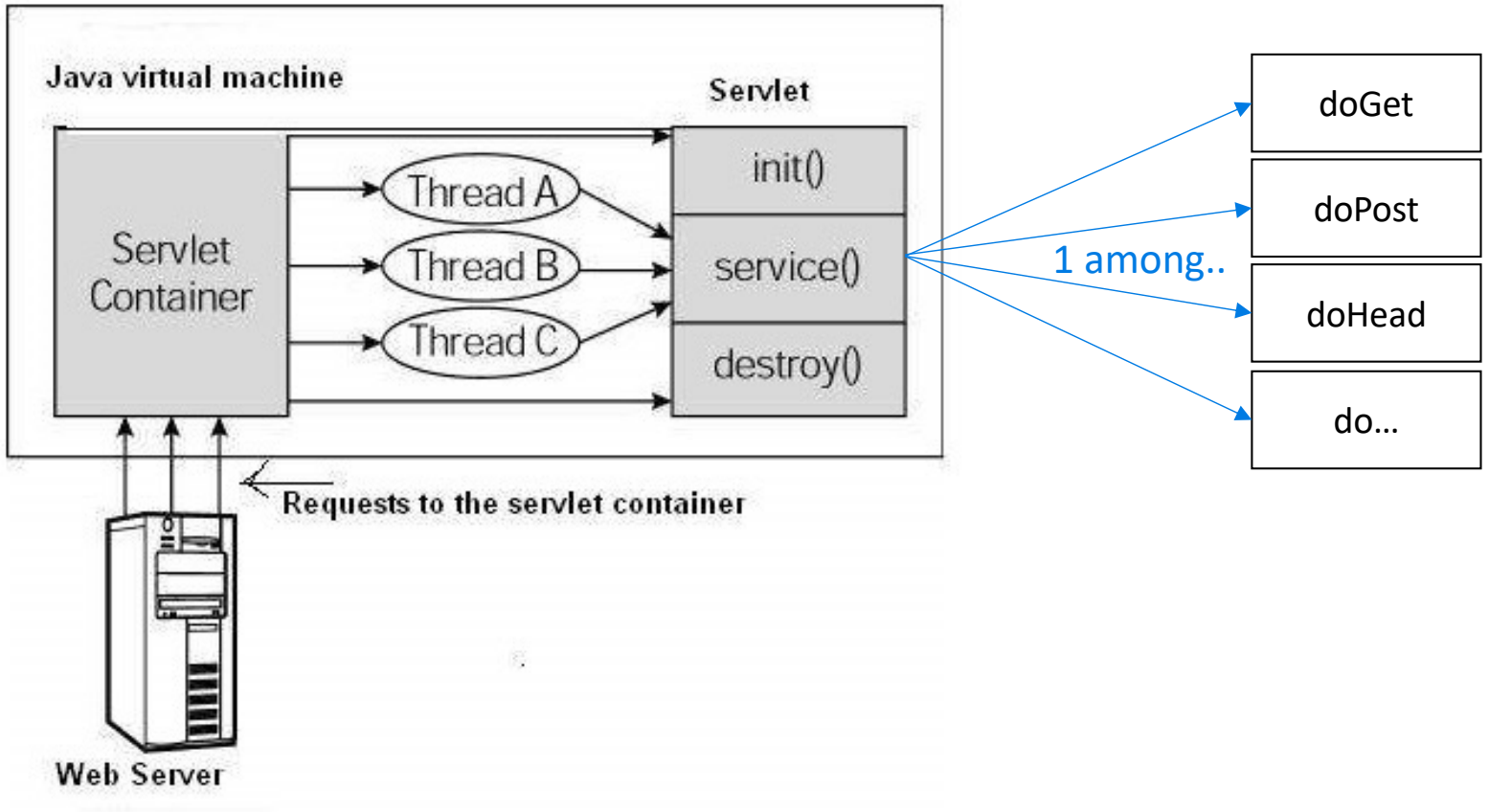

```
/**
 * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
 * methods.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
protected void processRequest(HttpServletRequest 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>");
    }
}}
```

Output


Servlets - Client
Java Technologie
localhost:8084/WebApplication01/
TODO write content


Servlets - Cli
Java Technol
Design Patte
localhost:8084/WebApplication01/ReadPost
Servlet ReadPost at /WebApplication01

Servlet lifecycle



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

Servlet Deployment

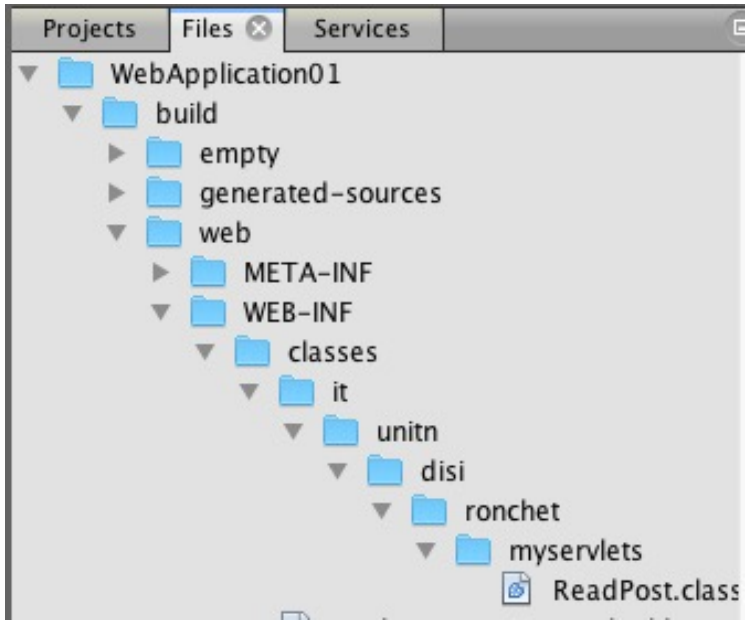
- By default, a servlet application is located at the path

<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.

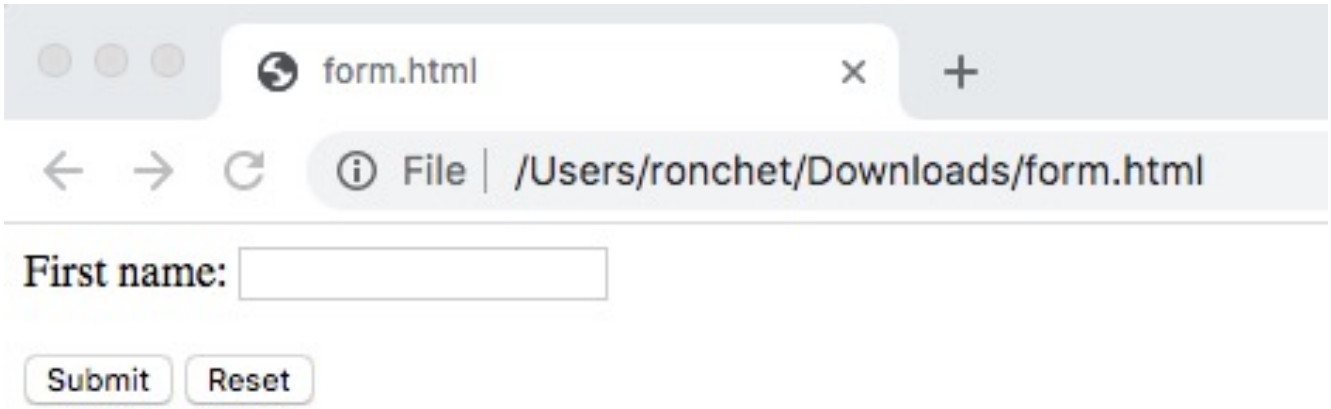


Step 3:
let’s pass parameters in our request

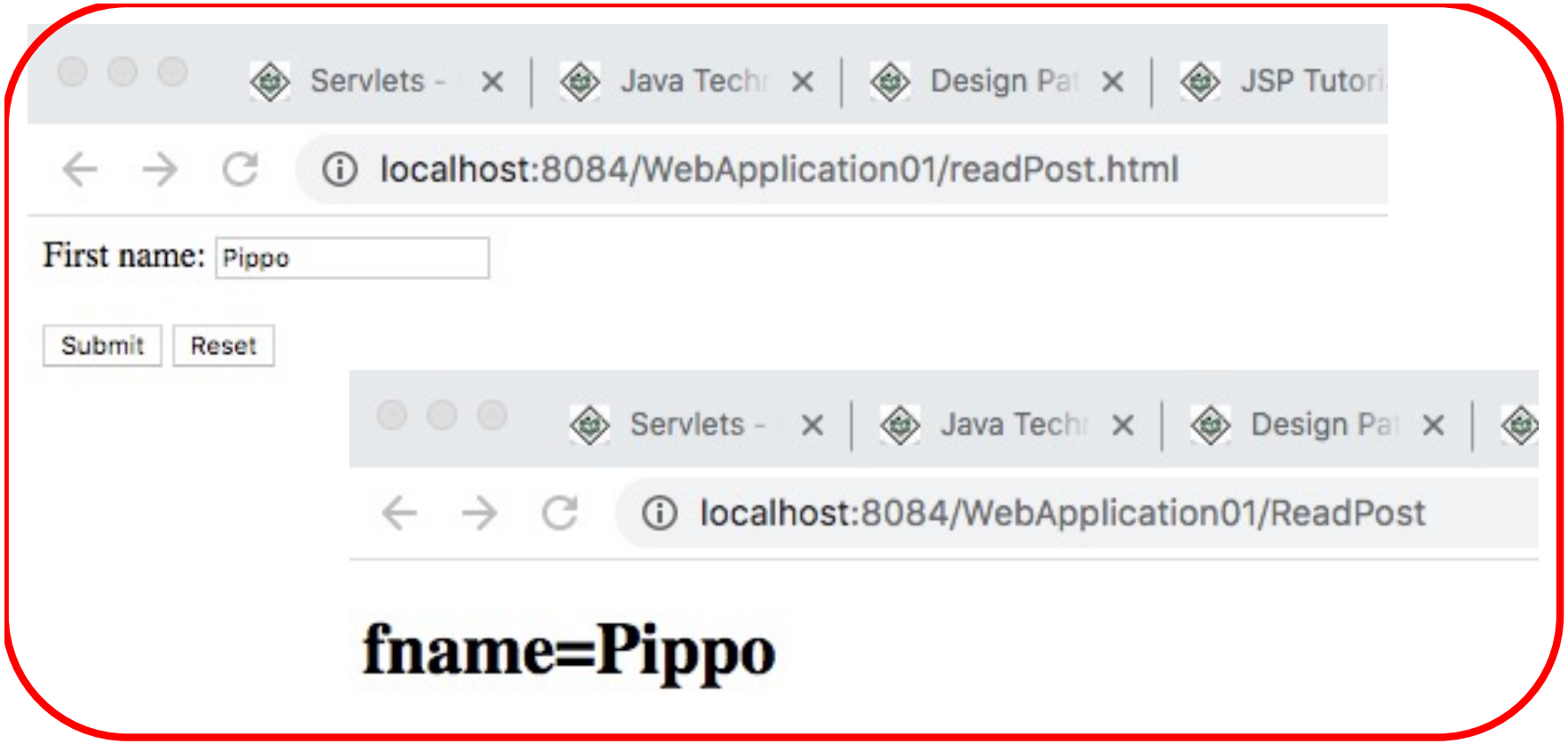
HTML Forms

Parameters passing

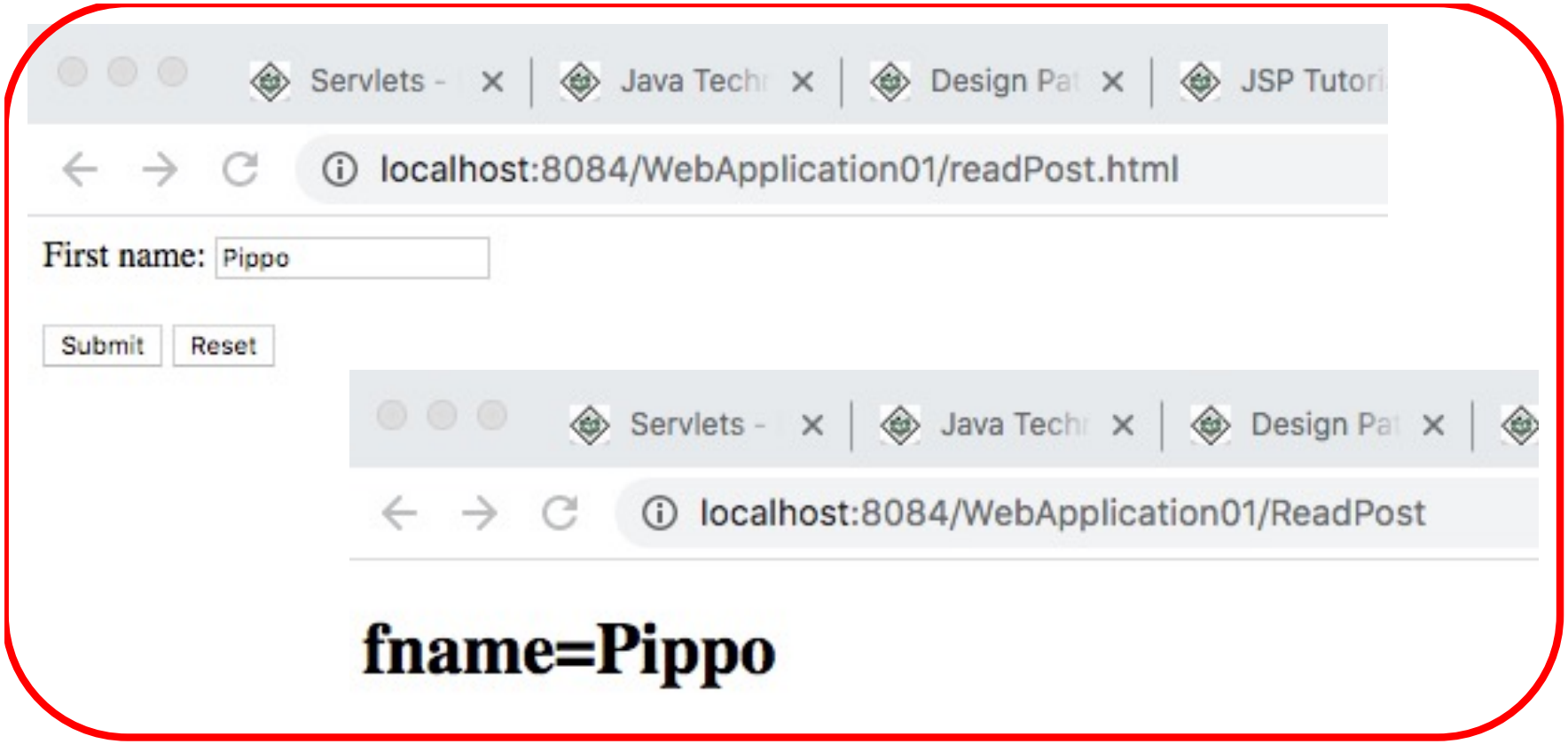
```
<html>
<body>
<form action="http://localhost:8084/WebApplication01/ReadPost" method="post">
   <label for="fname">First name:</label>
   <input type="text" name="fname"><br><br>
   <input type="submit" value="Submit">
   <input type="reset" value="Reset">
</form>
</body>
</html>
```



```
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>");
    }
}}
```

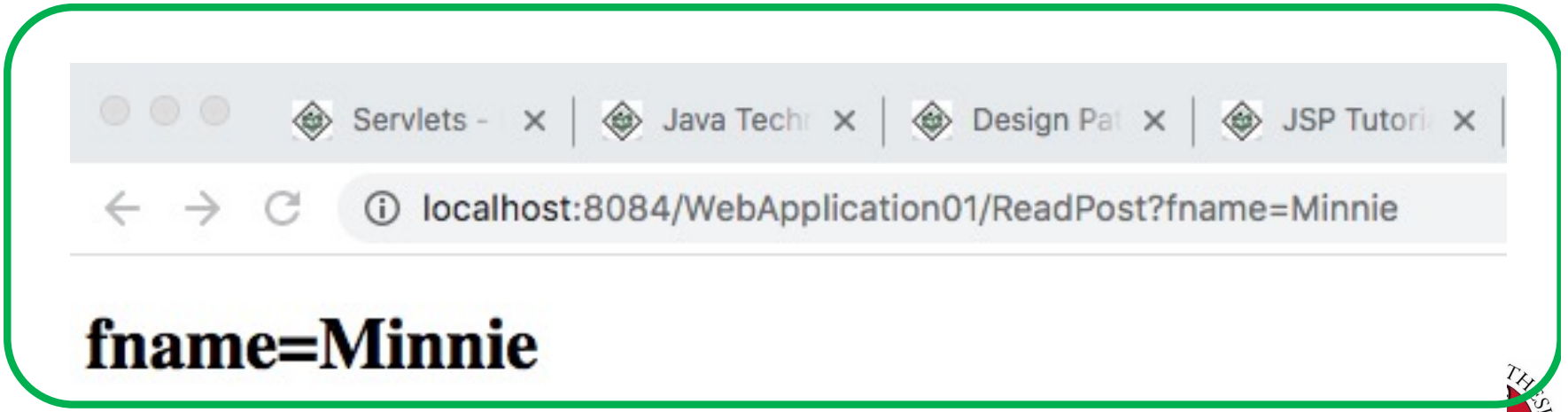


Servlets -
Java Techr
Design Pat
JSP Tutori
localhost:8084/WebApplication01/readPost.html
First name: Pippo
Submit
Reset
Servlets -
Java Techr
Design Pat
localhost:8084/WebApplication01/ReadPost
fname=Pippo

Post

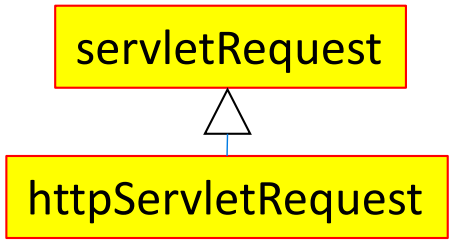

Servlets -
Java Tech
Design Pat
JSP Tutori
localhost:8084/WebApplication01/readPost.html
First name: Pippo
Submit
Reset
Servlets -
Java Tech
Design Pat
localhost:8084/WebApplication01/ReadPost
fname=Pippo

Post

Get


Servlets -
Java Tech
Design Pat
JSP Tutori
localhost:8084/WebApplication01/ReadPost?fname=Minnie
fname=Minnie

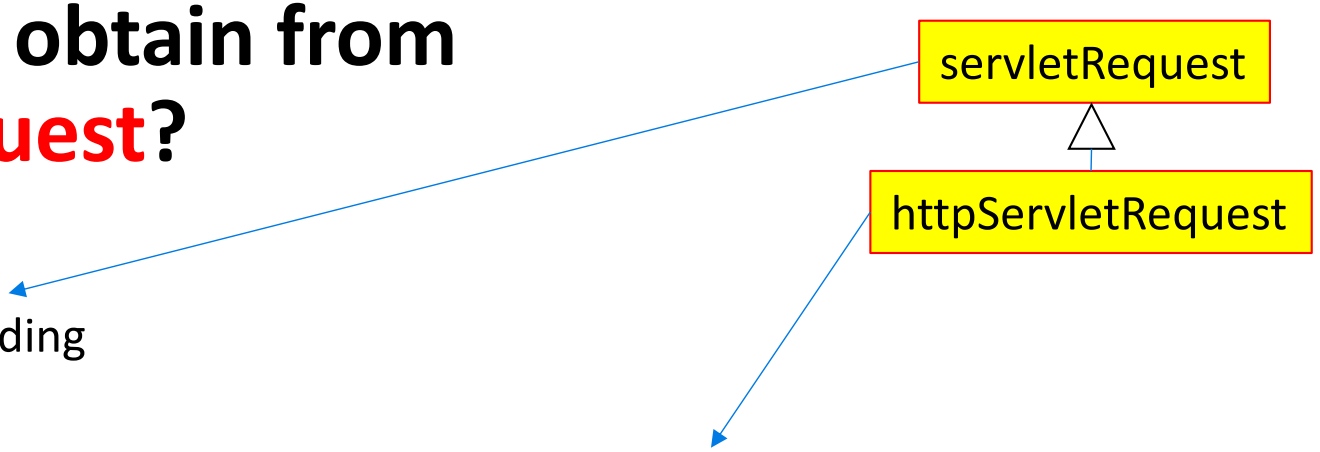
Getting parameters from httpServletRequest:



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

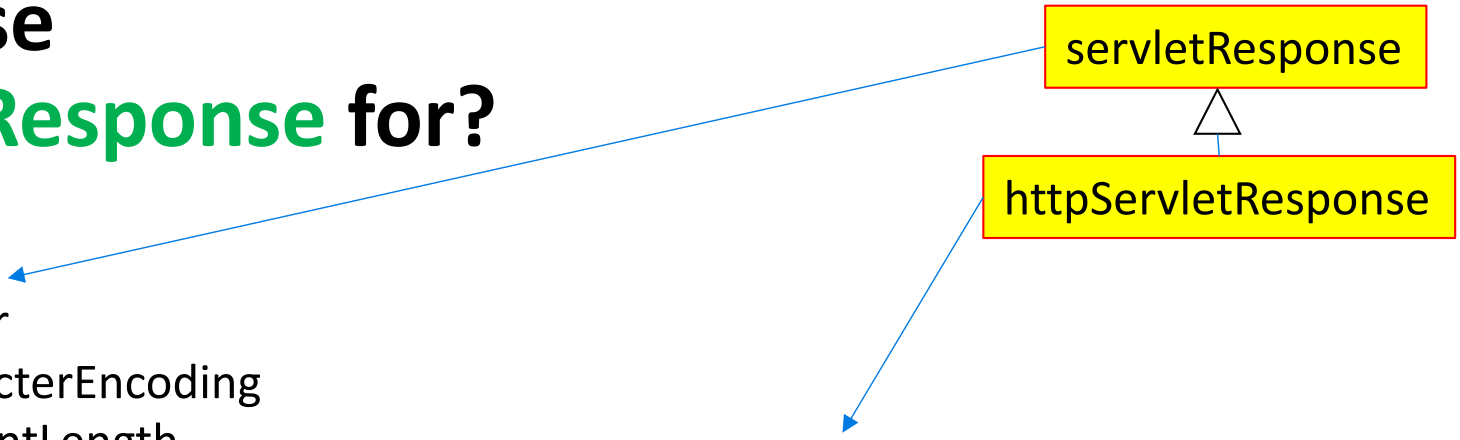


- Character encoding
- Content length
- Content Type
- Locale
- Protocol
- Local/remote name, address, port
- Server name
- Is Secure ?

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

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

What can use httpServletResponse for?



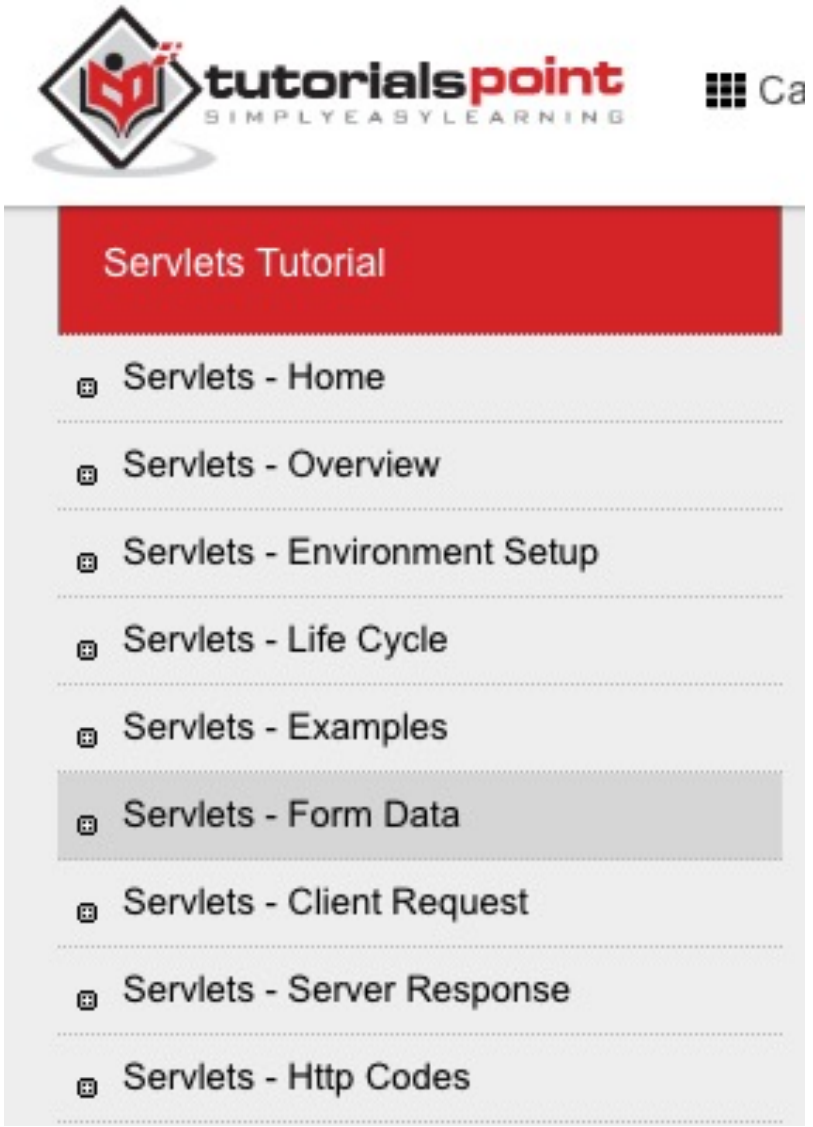
- getWriter
- setCharacterEncoding
- setContentLength
- setContentType
- setLocale

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

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

Readings

https://www.tutorialspoint.com/servlets/index.htm



Stop here!

Getting deeper with Servlets

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

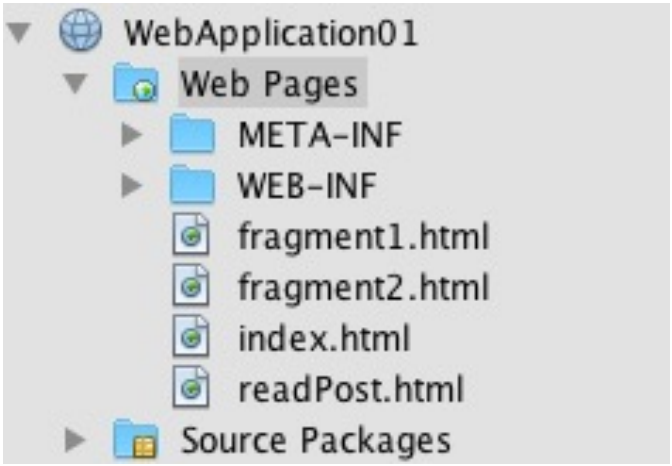
....xm | ReadPost.java | Demo1.java | fragment2.html | fragment1.html | Counter.java | readPost.html

```
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>
```

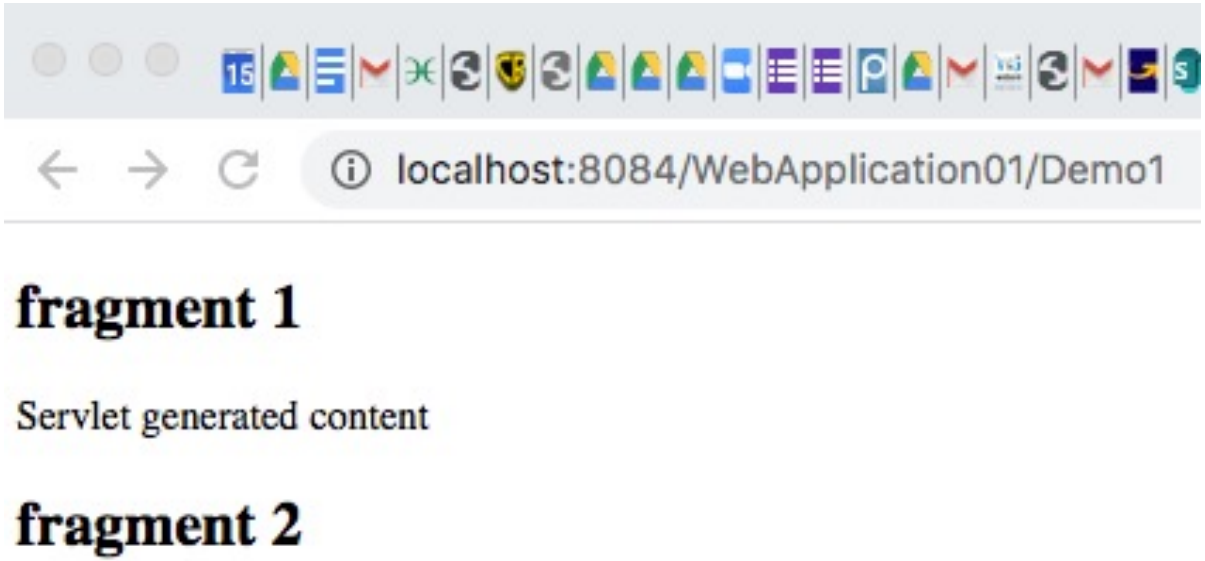
....xm | ReadPost.java | Demo1.java | fragment2.html

```
            <h2>fragment 2</h2>
2       </body>
3   </html>
4
```

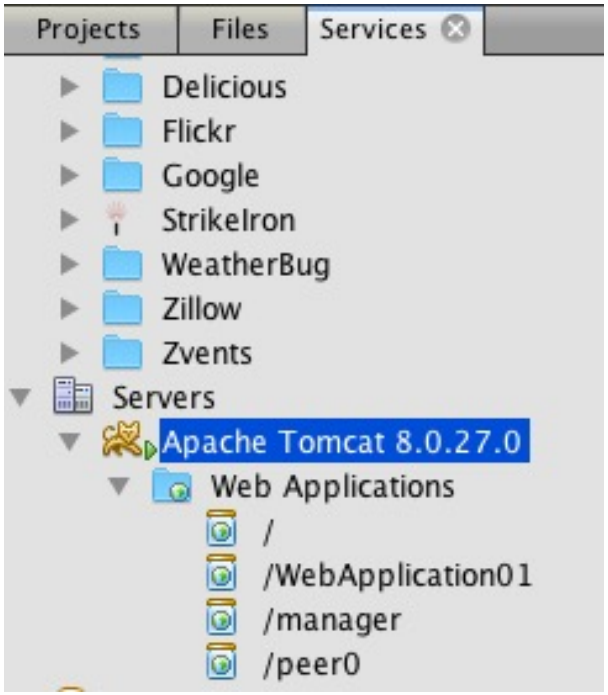
```
@Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            request.getRequestDispatcher("/fragment1.html")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

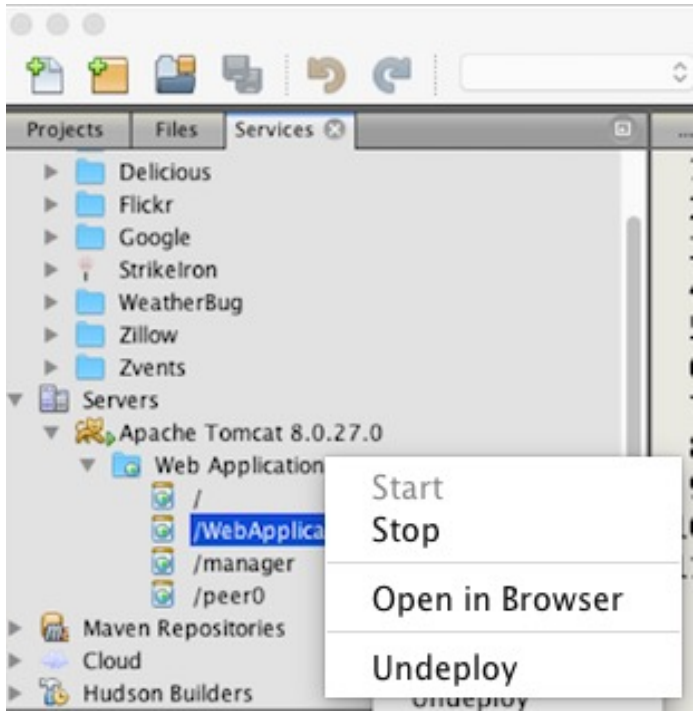


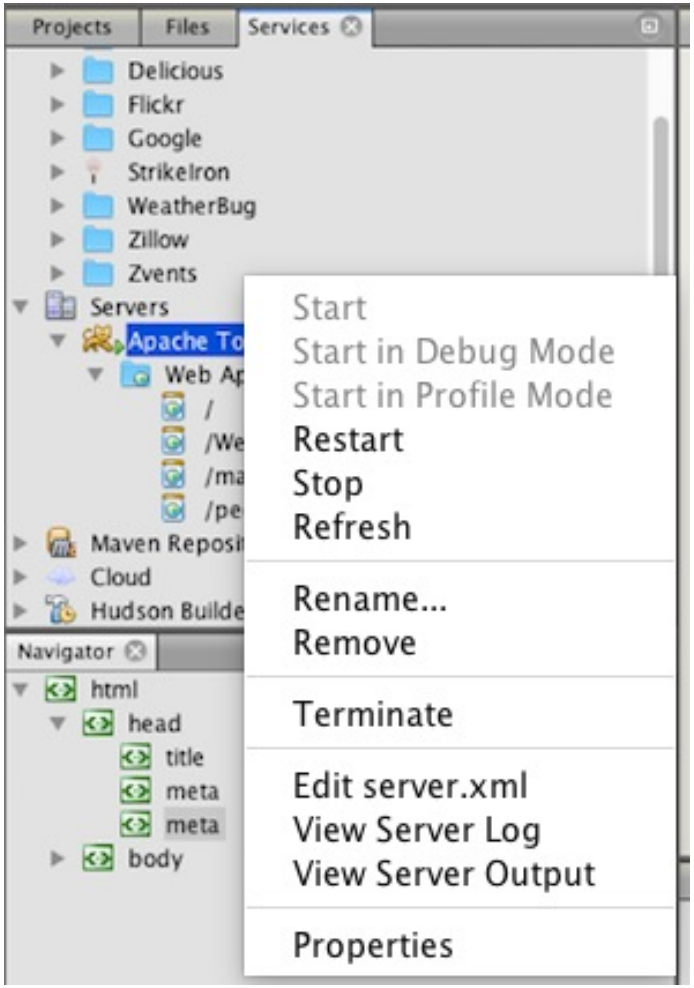
output


localhost:8084/WebApplication01/Demo1
fragment 1
Servlet generated content
fragment 2

Netbeans Services







Starting and stopping Tomcat
Undeploying WebApps