Dynamic content: programming the web servers

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

Q

How can we obtain dynamic responses from a Web server?

Dynamic pages: the basic idea:

1) We add to the URL the parameters which express our query
2) We write an executable, which somehow accesses the (GET/POST) parameters and generates an HTML page
3) We associate the URL with an executable

e.g.: CGI, what's the time?

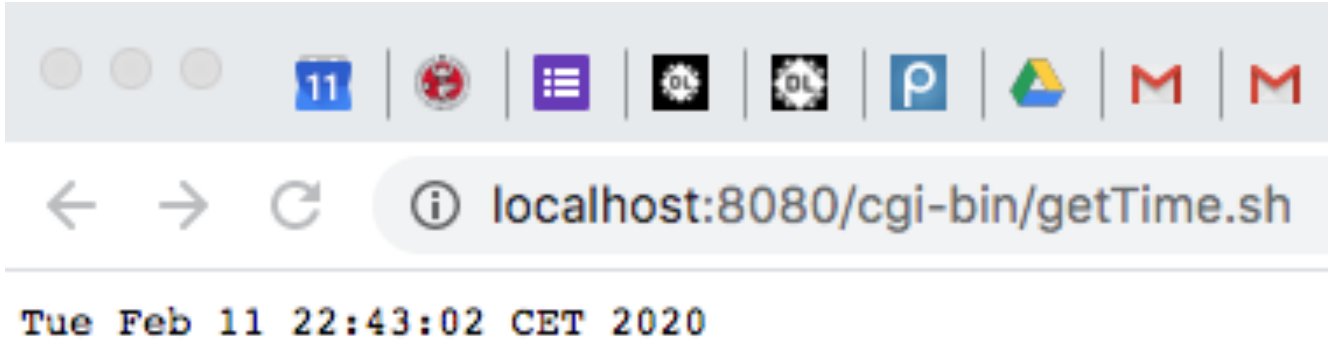
Creating a dynamic page

```
MR-MBP-14955:local ronchet$ cd /Applications/XAMPP/xamppfiles/cgi-bin
MR-MBP-14955:htdocs ronchet$ vi getTime.sh
MR-MBP-14955:htdocs ronchet$ cat getTime.sh
#!/bin/sh
echo "Content-type: text/plain; charset=iso-8859-1"
echo
echo "<HTML><BODY><H1>"
echo `date`
echo "</H1></BODY></HTML>"
```

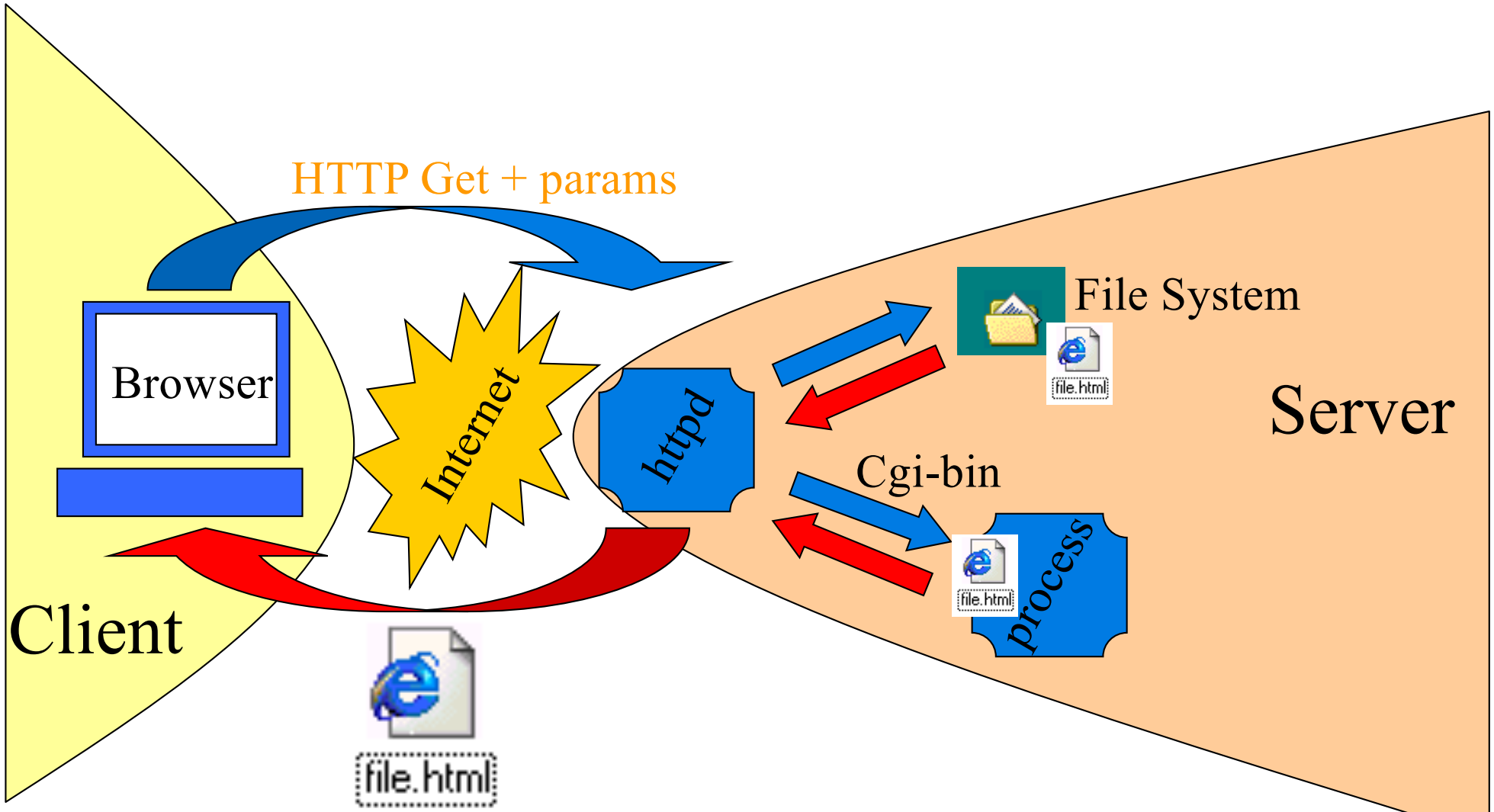
Edit file

Show content

embed HTML into the code



The original web architecture: dynamic pages



Evolution 1: dynamically create (interlinked) documents

Another solution: a simpler idea

Instead of embedding

HTML into the code, we could embed the code into HTML…

Template page

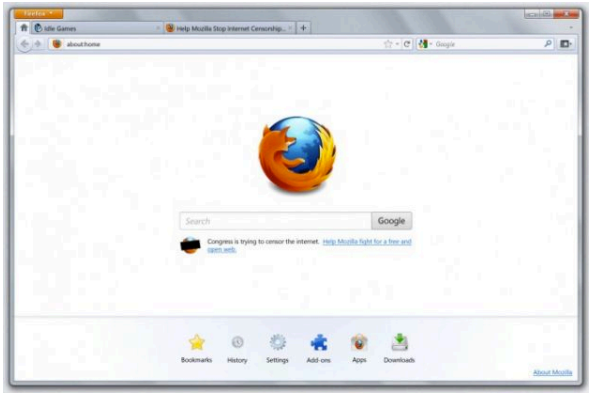
```
<HTML>
SOME HTML STUFF
%SOME CODE%
MORE HTML STUFF
</HTML>
```

```
<HTML>
SOME HTML STUFF
CODE OUTPUT
MORE HTML STUFF
</HTML>
```

A simpler idea

Augment the web server with an engine capable of parsing a web page, and executing code in it

Request URL



Retrieve template page from file system

```
<HTML>
SOME HTML STUFF
%SOME CODE%
MORE HTML STUFF
</HTML>
```

Evaluate code

```
<HTML>
SOME HTML STUFF
CODE OUTPUT
MORE HTML STUFF
</HTML>
```

Response

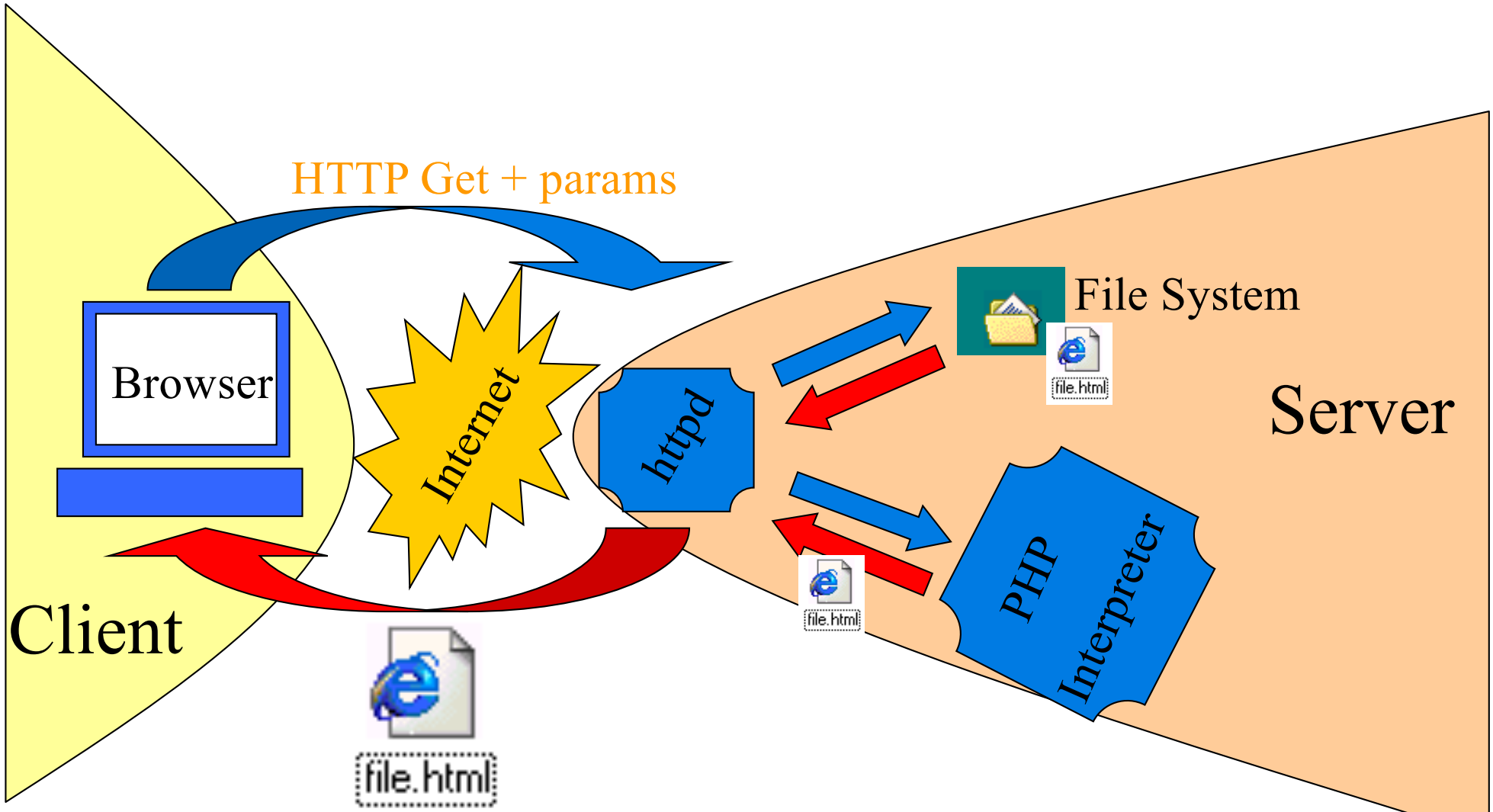
example: PHP

whatsTheTime.php

```
<!DOCTYPE html>
<html>
<body>
<i>Sir, the current time is
<?php
echo  date("h:i:sa");
?>
</i><br/>
(as far as I know!)
</body>
</html>
```

Sir, the current time is 04:20:11pm
(as far as I know!)

The original web architecture: dynamic pages



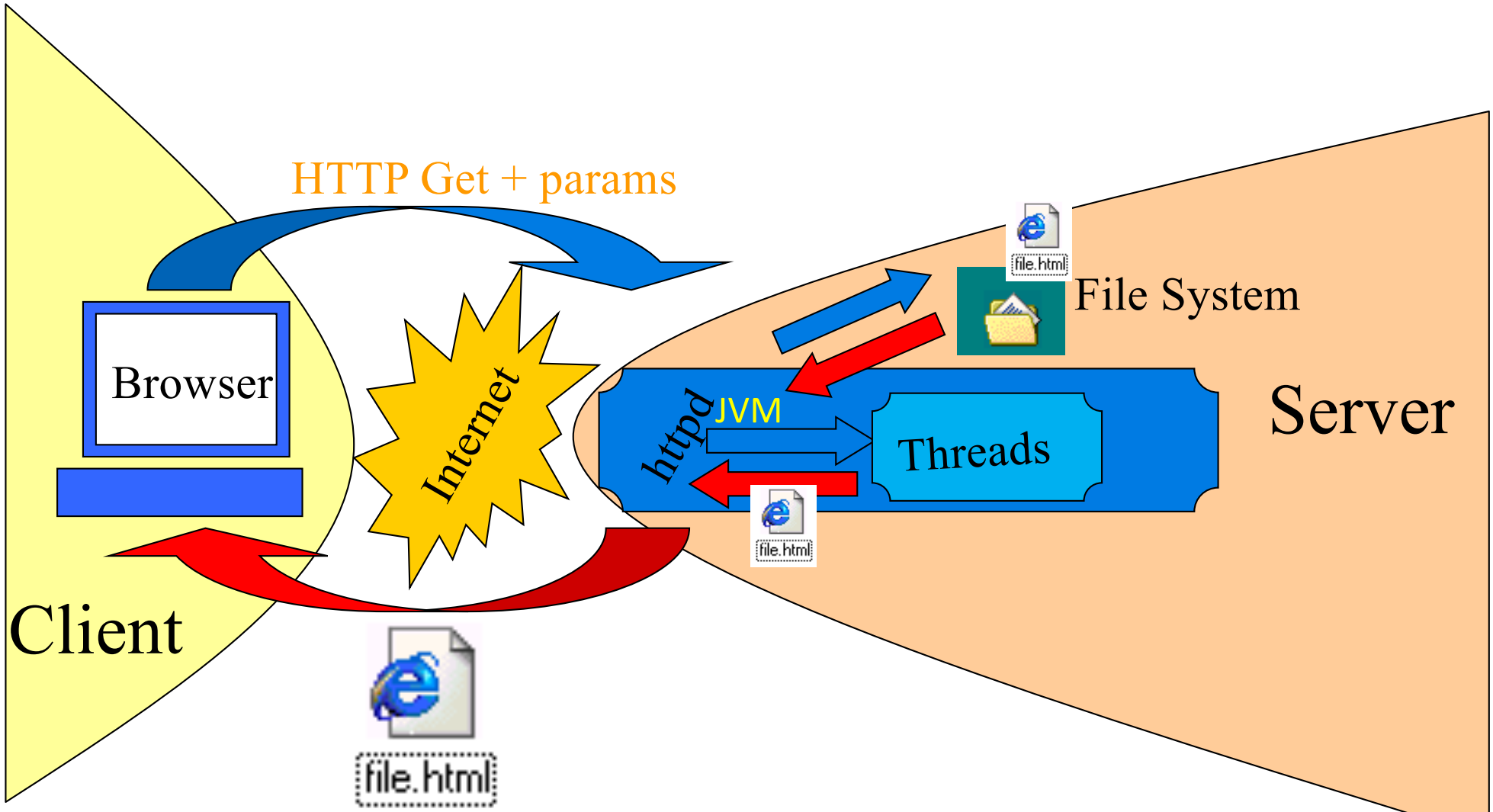
Evolution 1: dynamically create (interlinked) documents

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

How can we write a Servlet ?

Servlets

1. Create an HttpServlet subclass
2. Associate an URL to your class
3. implement 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)

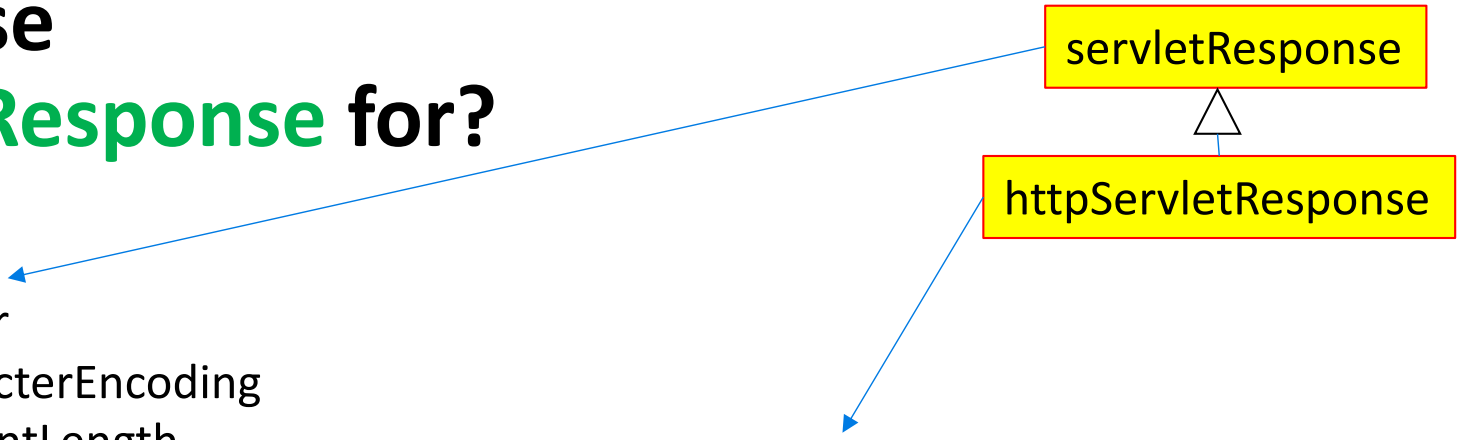
Example: let's write a Servlet that does not use parameters

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


What can use httpServletResponse for?

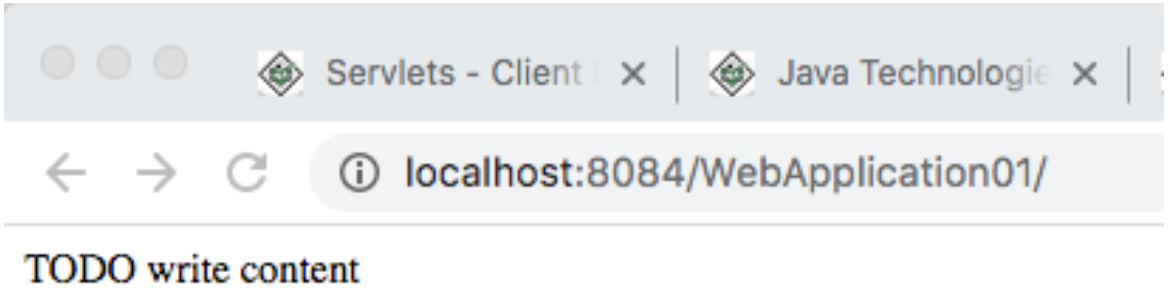


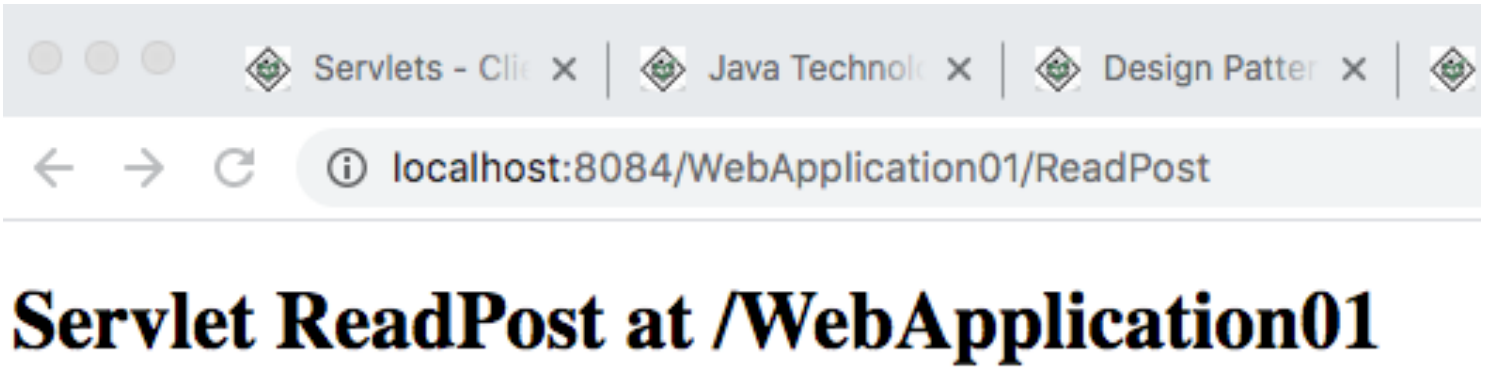
- getWriter
- setCharacterEncoding
- setContentLength
- setContentType
- setLocale

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

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

Output

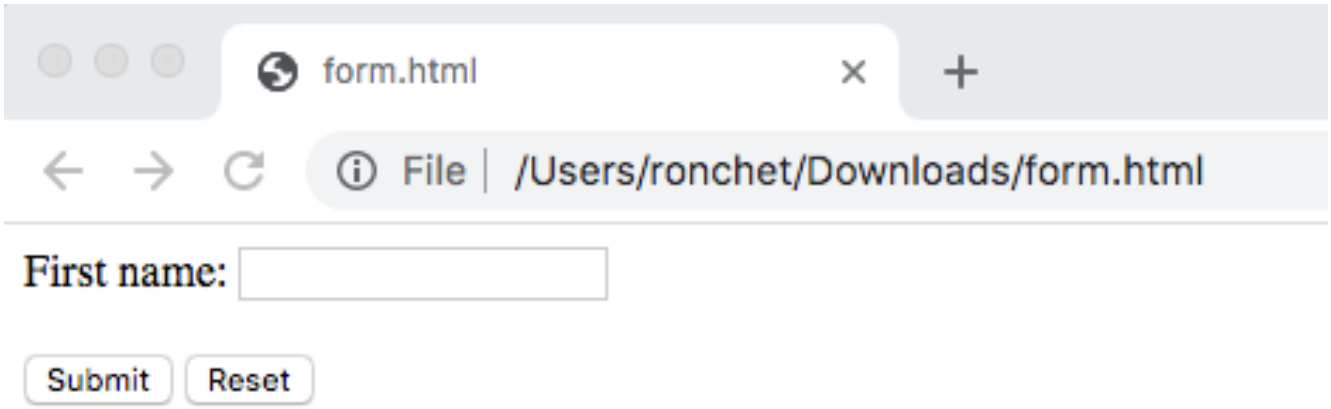

Servlets - Client
Java Technologie
localhost:8084/WebApplication01/
TODO write content


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

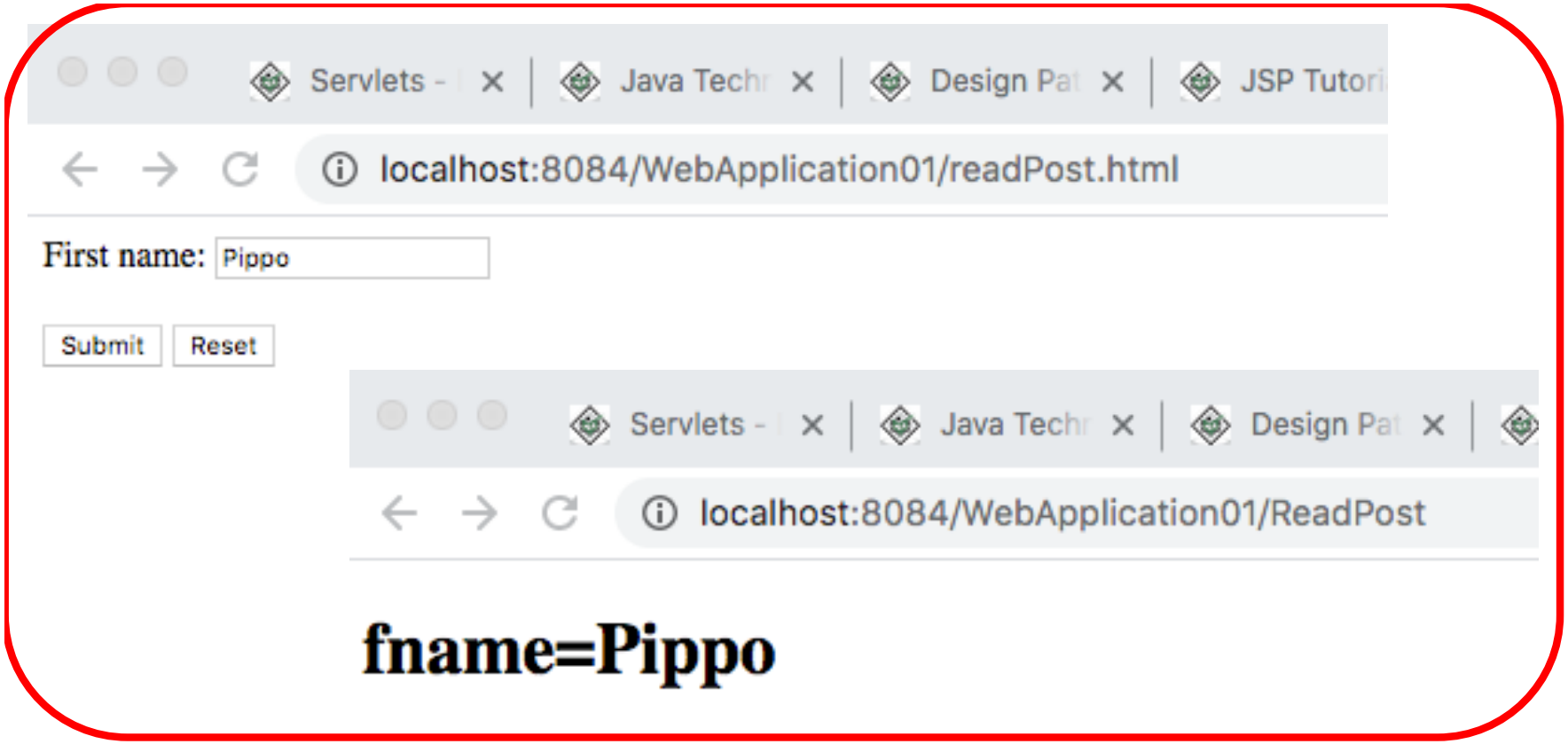
Next step:
let’s get the parameters contained in the request

Parameters passing

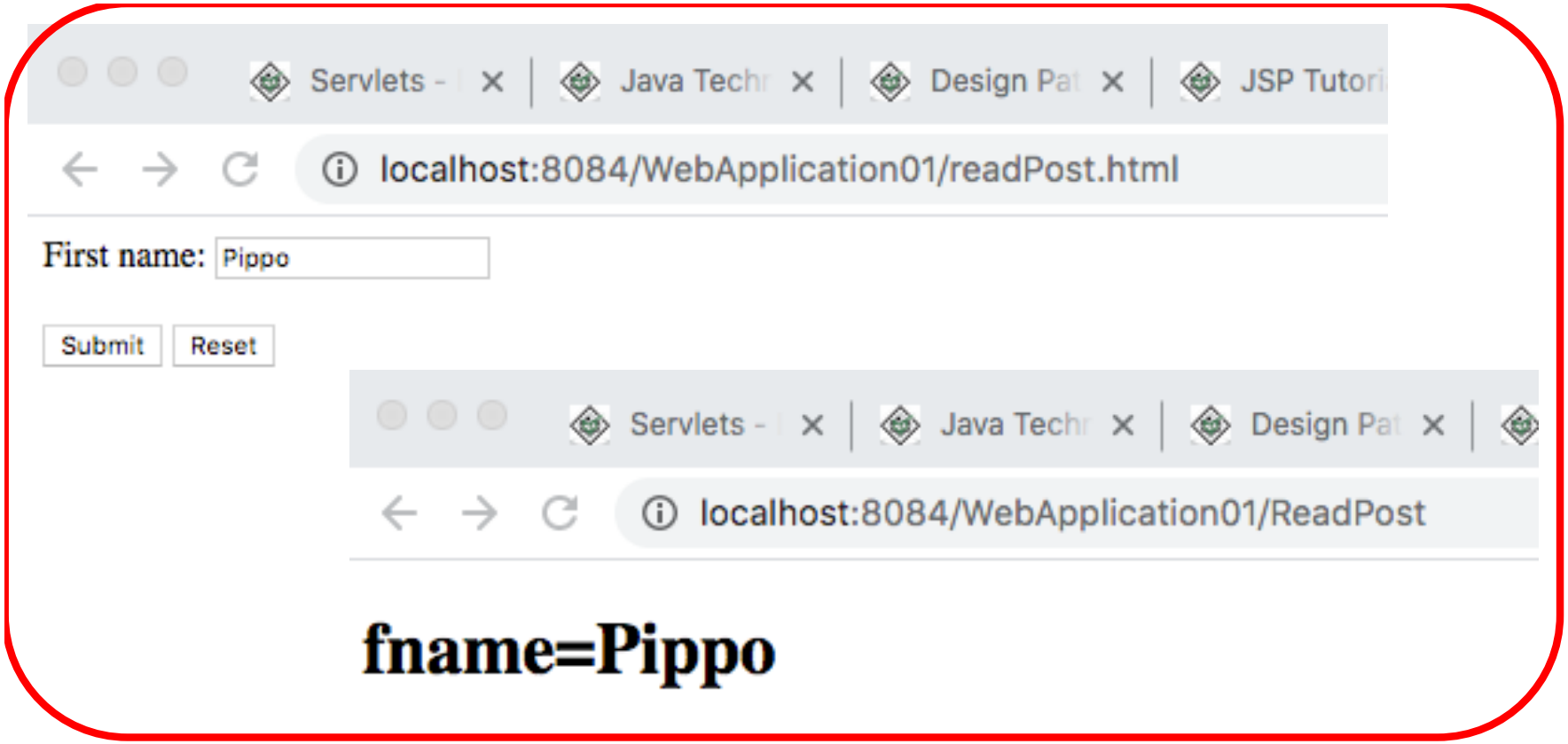
```
<html>
<body>
<form action="http://localhost:8084/WebApplication01/ReadPost" method="post">
   <label for="fname">First name:</label>
   <input type="text" name="fname"><br><br>
   <input type="submit" value="Submit">
   <input type="reset" value="Reset">
</form>
</body>
</html>
```



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

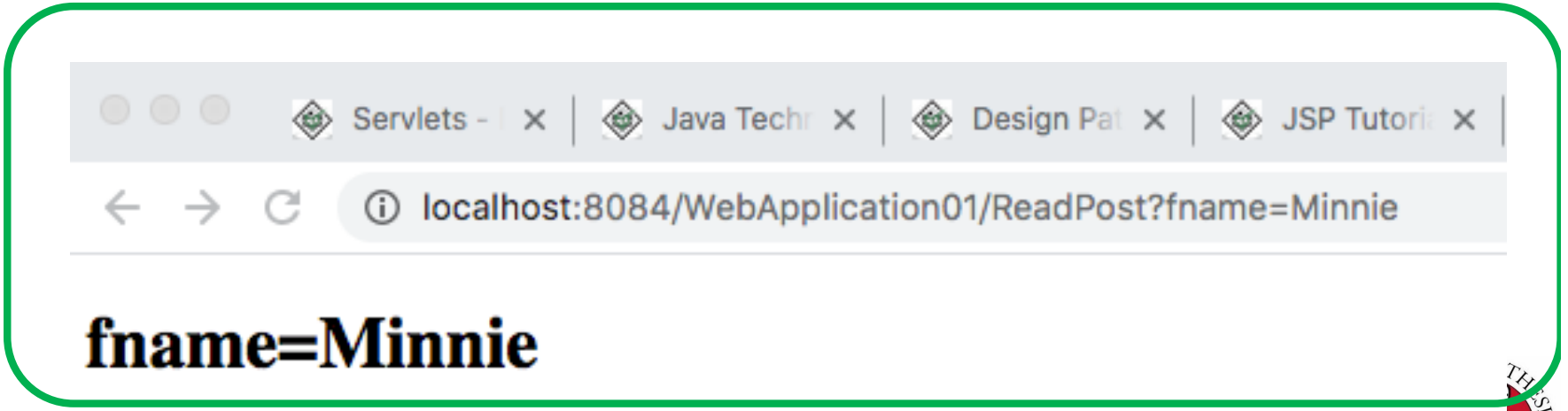


Servlets -
Java Techr
Design Pat
JSP Tutori
localhost:8084/WebApplication01/readPost.html
First name: Pippo
Submit
Reset
Servlets -
Java Techr
Design Pat
localhost:8084/WebApplication01/ReadPost
fname=Pippo

Post

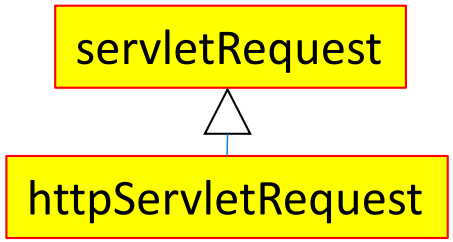

Servlets -
Java Techr
Design Pat
JSP Tutori
localhost:8084/WebApplication01/readPost.html
First name: Pippo
Submit
Reset
Servlets -
Java Techr
Design Pat
localhost:8084/WebApplication01/ReadPost
fname=Pippo

Post

Get


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

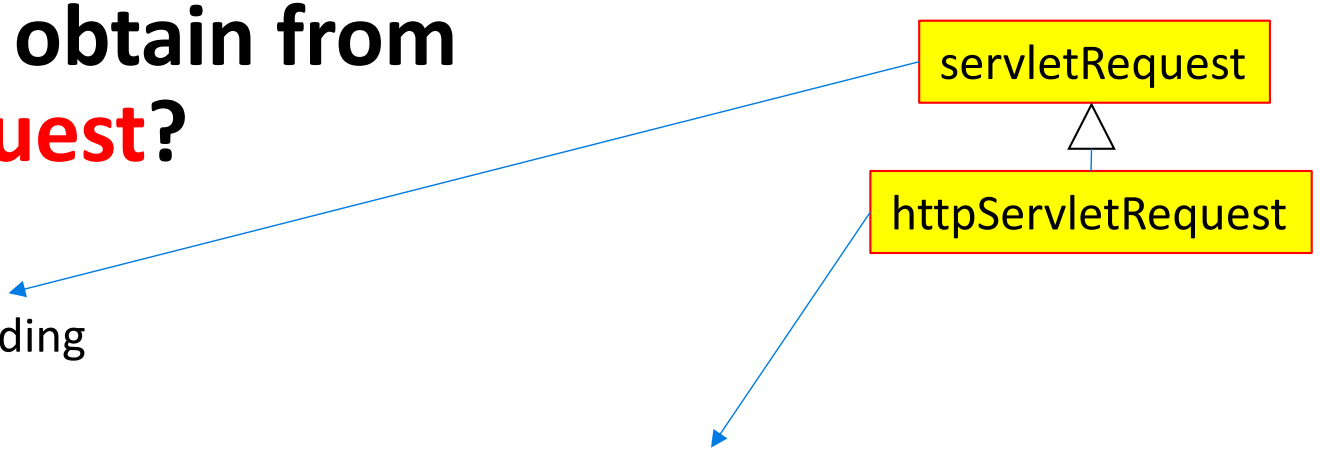
Getting parameters from httpServletRequest:



`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

How can we deploy a Servlet ?

Apache Tomcat

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

Apache Tomcat Versions

Apache Tomcat® is an open source software implementation of a subset of the Jakarta EE (formally Java EE) technologies. Different versions of Apache Tomcat are available for different versions of the specifications. The mapping between the specifications and the respective Apache Tomcat versions is:

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	10.0.0-M1	8 and later
4.0	2.3	3.0	1.1	1.1	9.0.x	9.0.31	8 and later
3.1	2.3	3.0	1.1	1.1	8.5.x	8.5.51	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	7.0.100	6 and later (7 and later for WebSocket)

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.