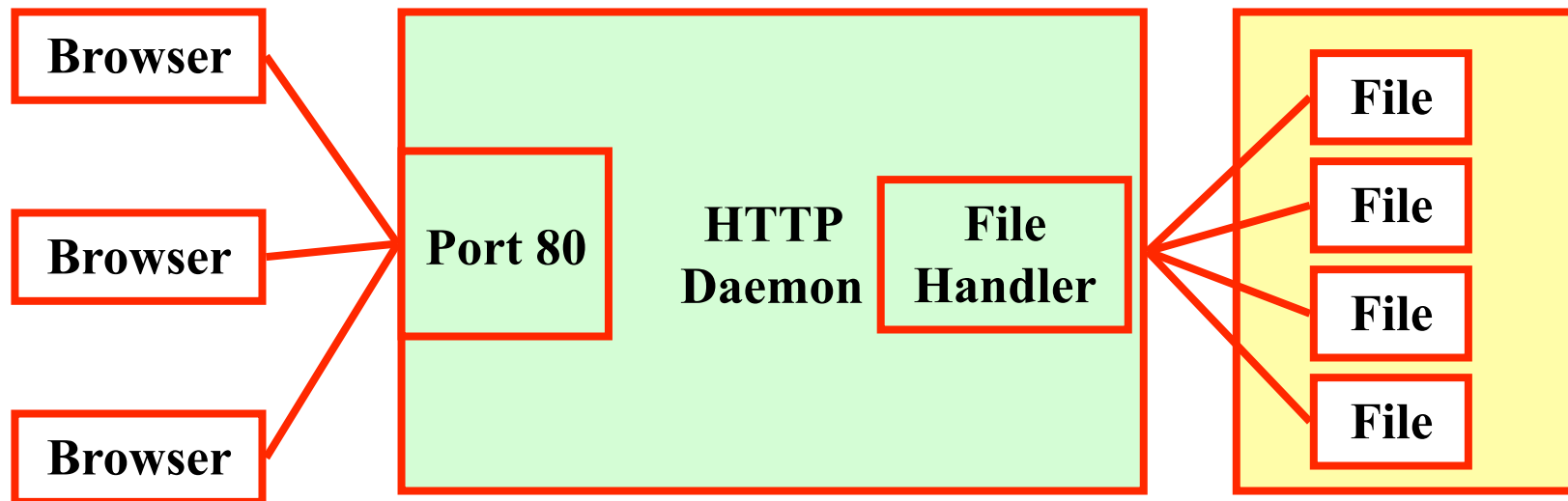
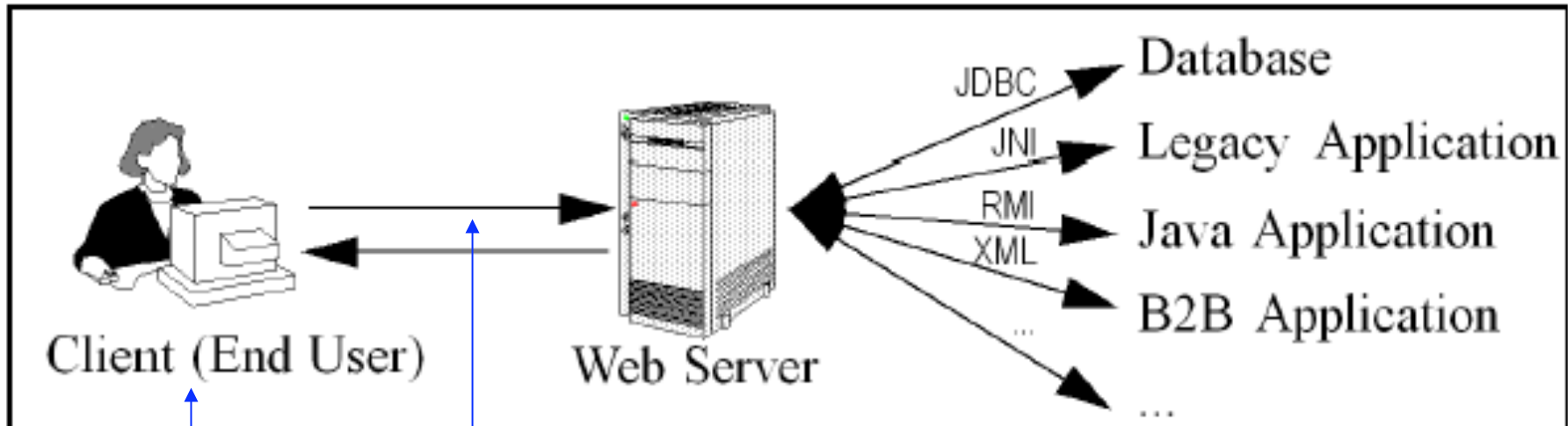


Web-based Applications

Web Servers with Static Web Pages



Dynamic Web Applications



**Web
Browser**

HTTP

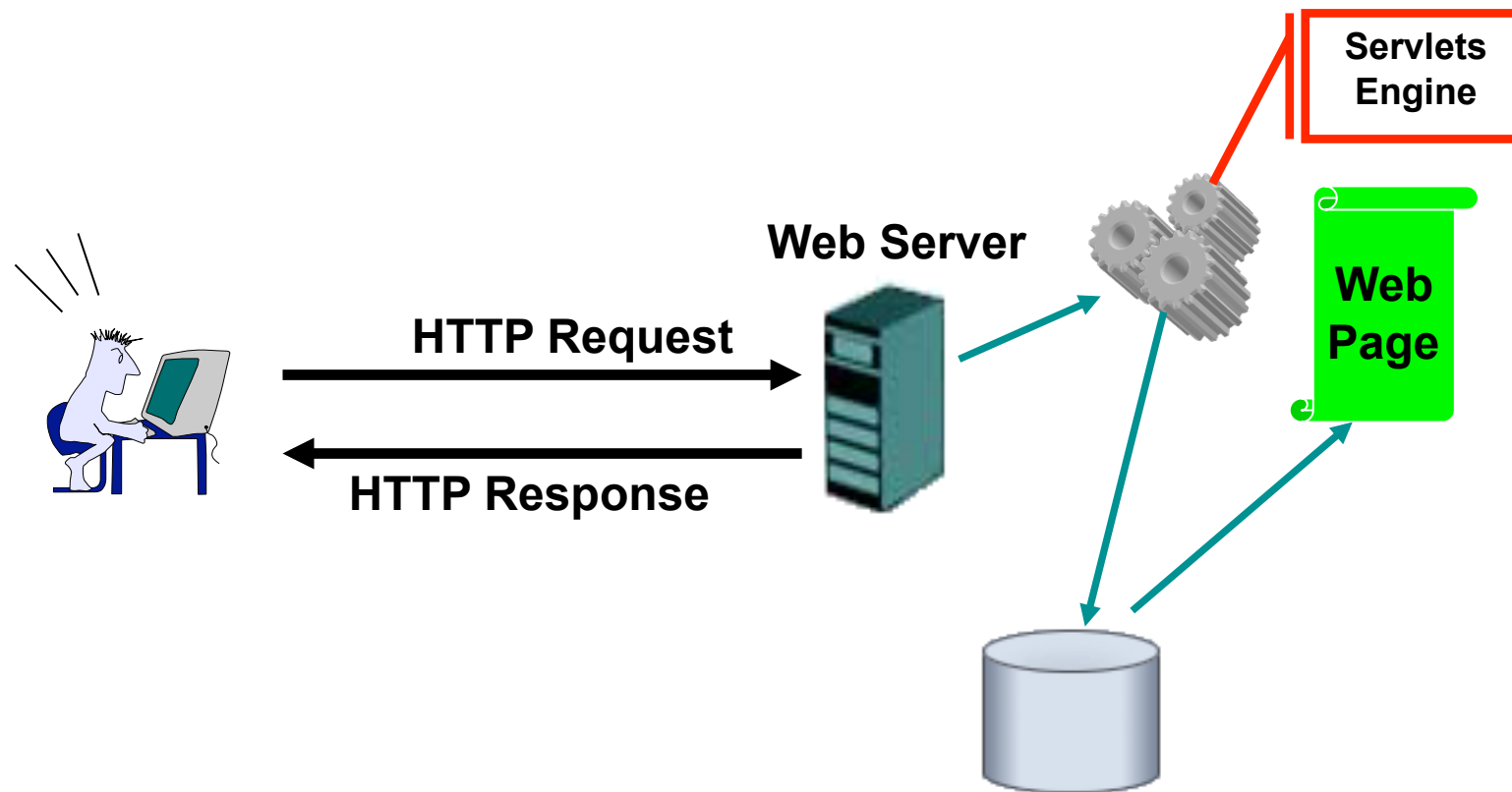
TECNOLOGIAS:

- Microsoft Active Server Pages (ASP.Net)
- Scripting Languages: PHP, Perl
- Servlets & Java ServerPages (JSP)
- Ruby-on-Rails
- Django or TurboGears
-



Java Servlets

Building Web Pages *on-the-fly* with Servlets



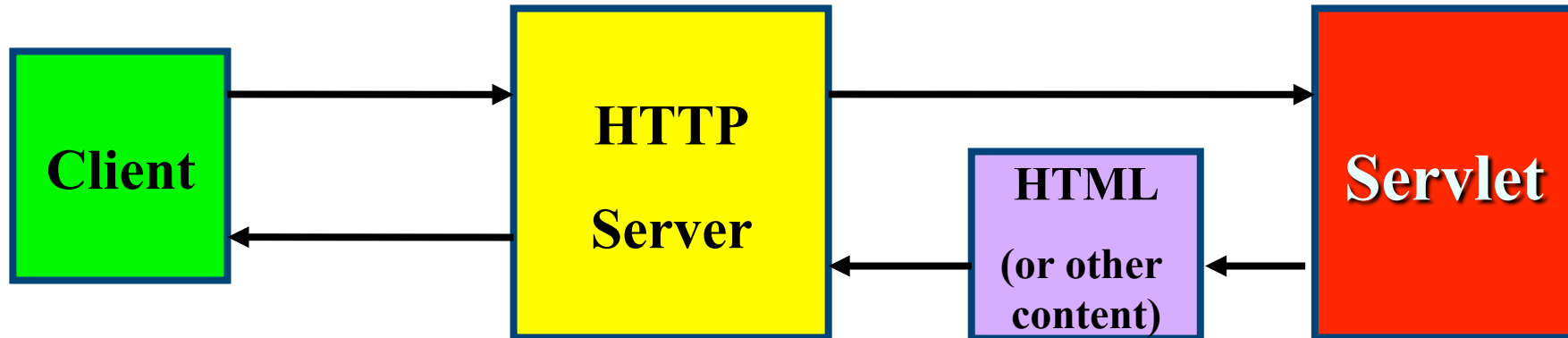
Servlets

- A **Servlet** is a Web component that is managed by a container/engine, and generates dynamic content.
- **Java ServerPages** builds upon the Servlet framework and provides page generation functionality similar to that of ASP.

Servlets: Web Objects

- Servlets are loaded once; object persists between requests.
- Benefits:
 - keeps memory footprint small;
 - eliminates object creation overhead and delays in servicing client request;
 - enables persistence of resources that the servlet may use (e.g. database connections);
- Scalability due to multi-threading.

Servlet - Web Server Architecture



- Read data sent by the user.
- Look up info about the request component of the HTTP Request.
- Generate results.
- Format results in a document.
- Set HTTP response parameters.
- Send document to client.

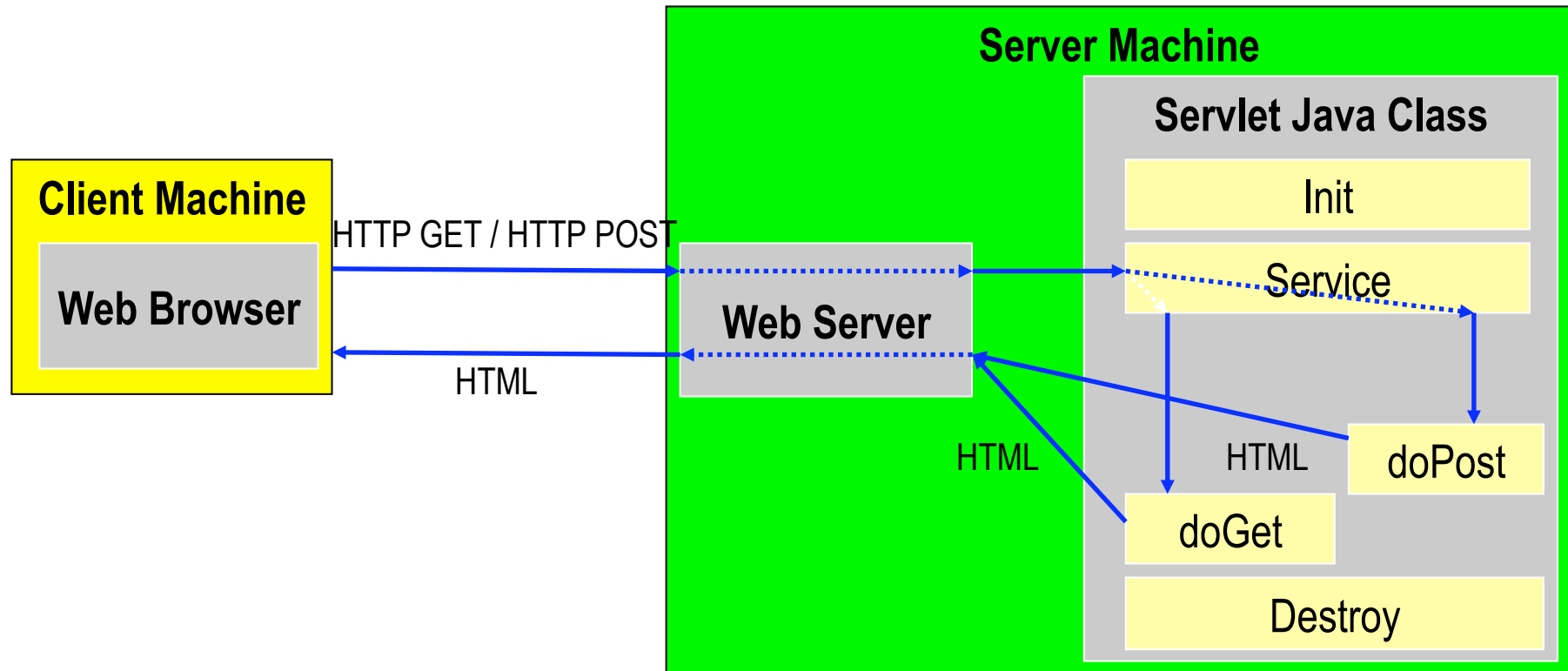
HttpServlet

- You can extend a class that already implements this interface:

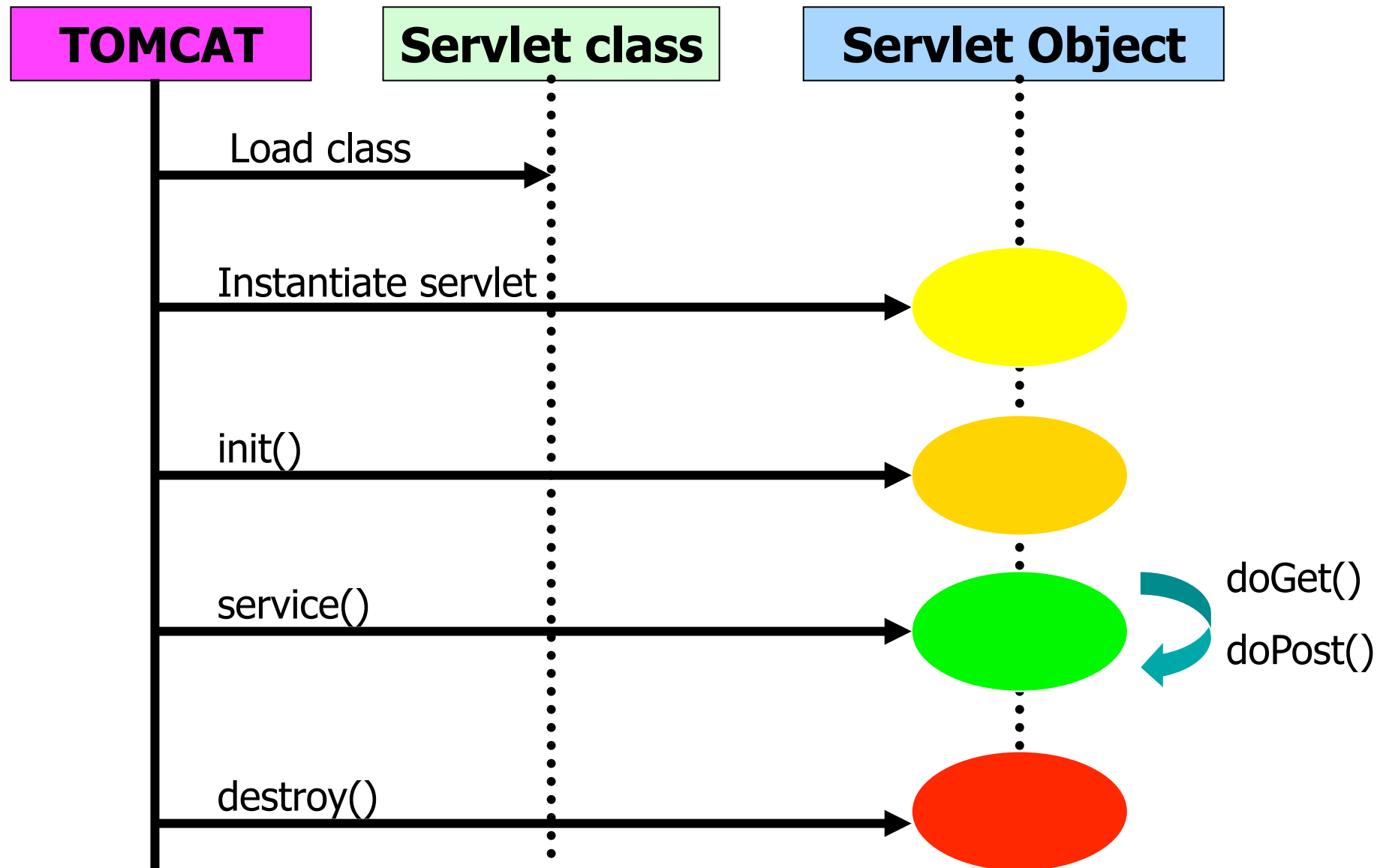
HttpServlet

- A servlet should extend **HttpServlet**.
- Should also override **doGet** and/or **doPost**

Arquitectura



Lifecycle of a Servlet



HttpServlet

- How to process HTTP requests:
 - **doGet** ("GET" requests)
 - **doPost** ("POST" requests)
 - **doPut** ("PUT" requests)
 - **doDelete** ("DELETE" requests)
 - **doHead** ("HEAD" requests)
 - **doOptions** ("OPTIONS" requests)
 - **doTrace** ("TRACE" requests)

-HelloServlet-

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException
    {
        PrintWriter out = response.getWriter();
        out.println("Latada!!!!");
    }
}
```

A Simple Servlet

- **HttpServletRequest**

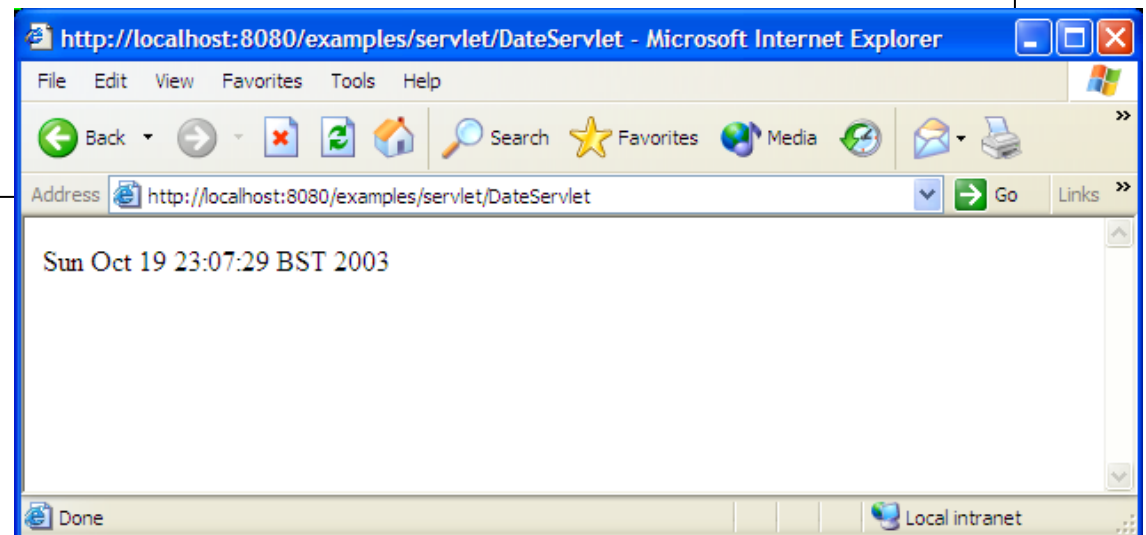
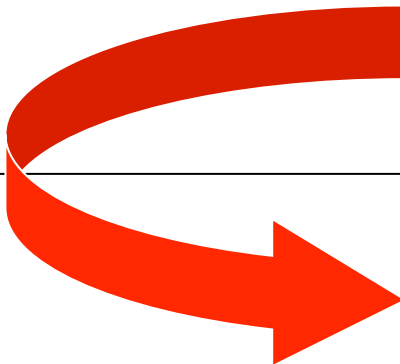
- has methods which can be used to obtain information about the incoming request (form data, request header, client host name etc).

- **HttpServletResponse**

- represents the response to the client
- lets you set HTTP status codes (eg. 404)
- lets you obtain access to an OutputStream for sending data to the browser (PrintWriter)

-DateServlet-

```
public class DateServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException
    {
        PrintWriter out = response.getWriter();
        out.println(new Date().toString());
    }
}
```



Now, in HTML Format...

```
<HTML>  
<HEAD>  
  <TITLE>DateServlet</TITLE>  
</HEAD>  
  
<BODY>  
<H1> data/hora no servidor.... </H1>  
</BODY>  
</HTML>
```


- DateServlet -

```
public class DateServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println(  
            "<HTML>\n" + "<HEAD><TITLE>DateServlet</TITLE></HEAD>" +  
            "<BODY>\n" +  
            "<H1>" + new Date().toString() + "</H1>\n" +  
            "</BODY></HTML>");  
    }  
}
```

Servlet Requests



ServletRequest

- Has methods which can be used to obtain information about the incoming request:
 - Form data,
 - request headers,
 - client host name etc.

Get Information

String getMethod() – obtain the HTTP request is it a GET or POST

String getParameter(String name) - get a parameter (covered later)

String getRemoteHost() - client's host name

String getRemoteAddr() - IP address of client

String getServerName() - servers name

int getServerPort() - port

String getHeader("User-Agent") - name of client's web browser

String getHeader("Referer") - URL of page that called the servlet

some more methods ...

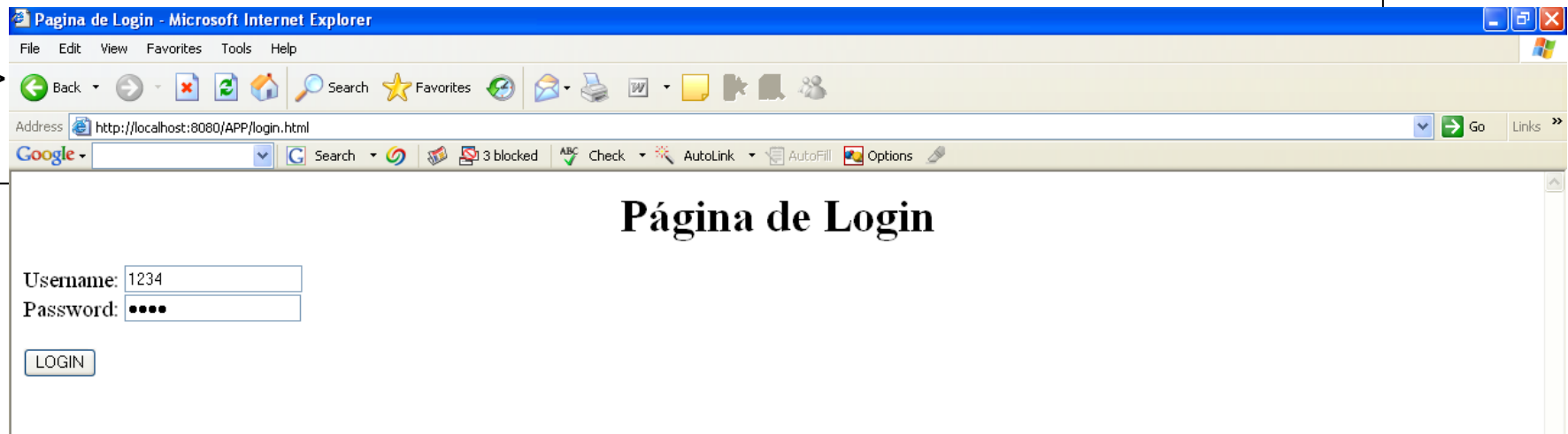
Exemplo: A nossa página de login

```
<html>
<head>
<title> Pagina de Login </title>
</head>
<body>

<h1 align="center"> Página de Login </h1>

<form action="Login" method="POST">
  Username: <input type="text" name="user" /> <br/>
  Password: <input type="password" name="pass" /> <br/> <br/>
  <input type="SUBMIT" value="LOGIN"/>
</form>

</body>
</html>
```



Servlet: Login

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Login extends HttpServlet{
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
        IOException, ServletException {

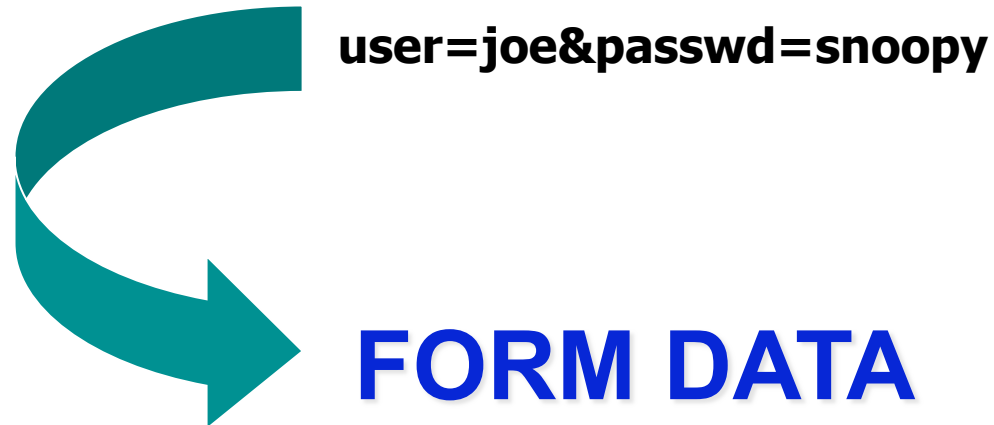
        String username = request.getParameter("user");
        String password = request.getParameter("pass");

        PrintWriter out = response.getWriter();
        out.println("username = " + username);
        out.println("password= " + password);

    }
}
```

Handling Form Data

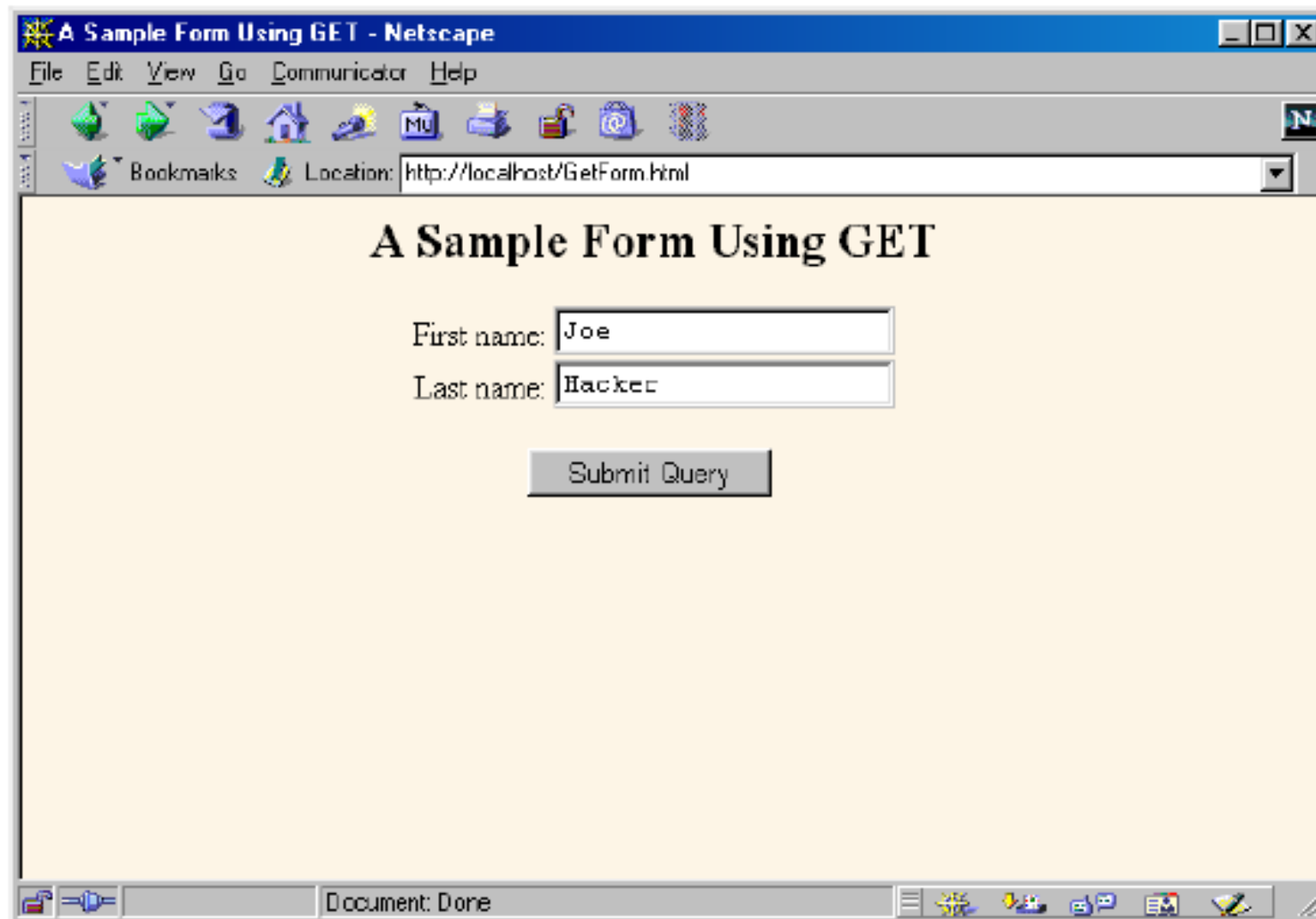
`http://host/login?user=joe&passwd=snoopy`



Form Data

- Parameter information is accessed by:
`request.getParameter(String paramName)`
- This is the same method whether it is a GET or POST request.
- Return value is a String representing the parameter value.

HTML Form with GET



The screenshot shows a Netscape browser window with the title "A Sample Form Using GET - Netscape". The address bar displays "http://localhost/GetForm.html". The form content is as follows:

A Sample Form Using GET

First name:

Last name:

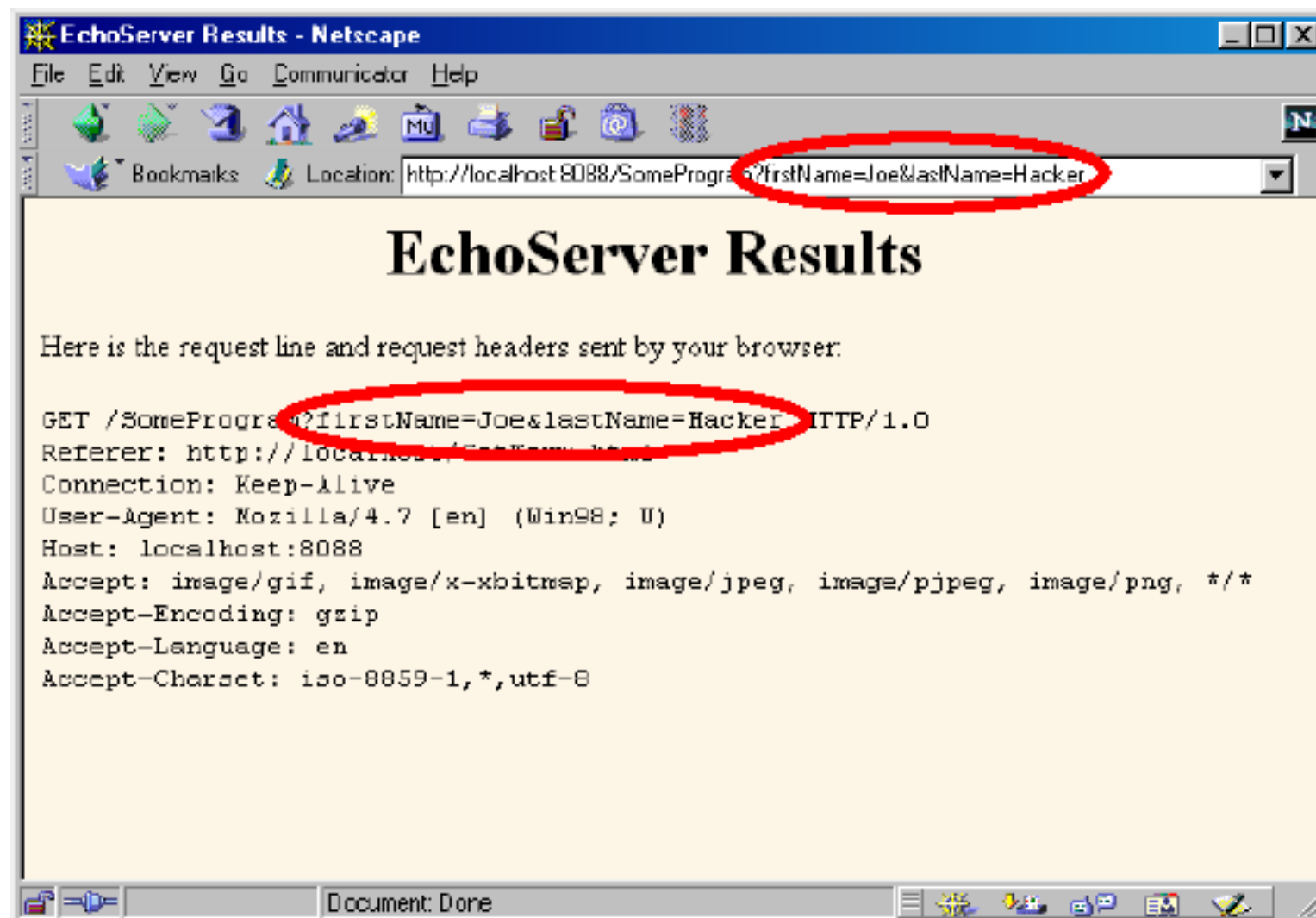
The status bar at the bottom indicates "Document: Done".

HTML Form with GET...

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>A Sample Form Using GET</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using GET</H2>

<FORM ACTION="http://localhost:8088/SomeProgram">
  <CENTER>
    First name:
    <INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
    Last name:
    <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
    <INPUT TYPE="SUBMIT"> <!-- Press this to submit form -->
  </CENTER>
</FORM>
</BODY></HTML>
```

HTML Form with GET...

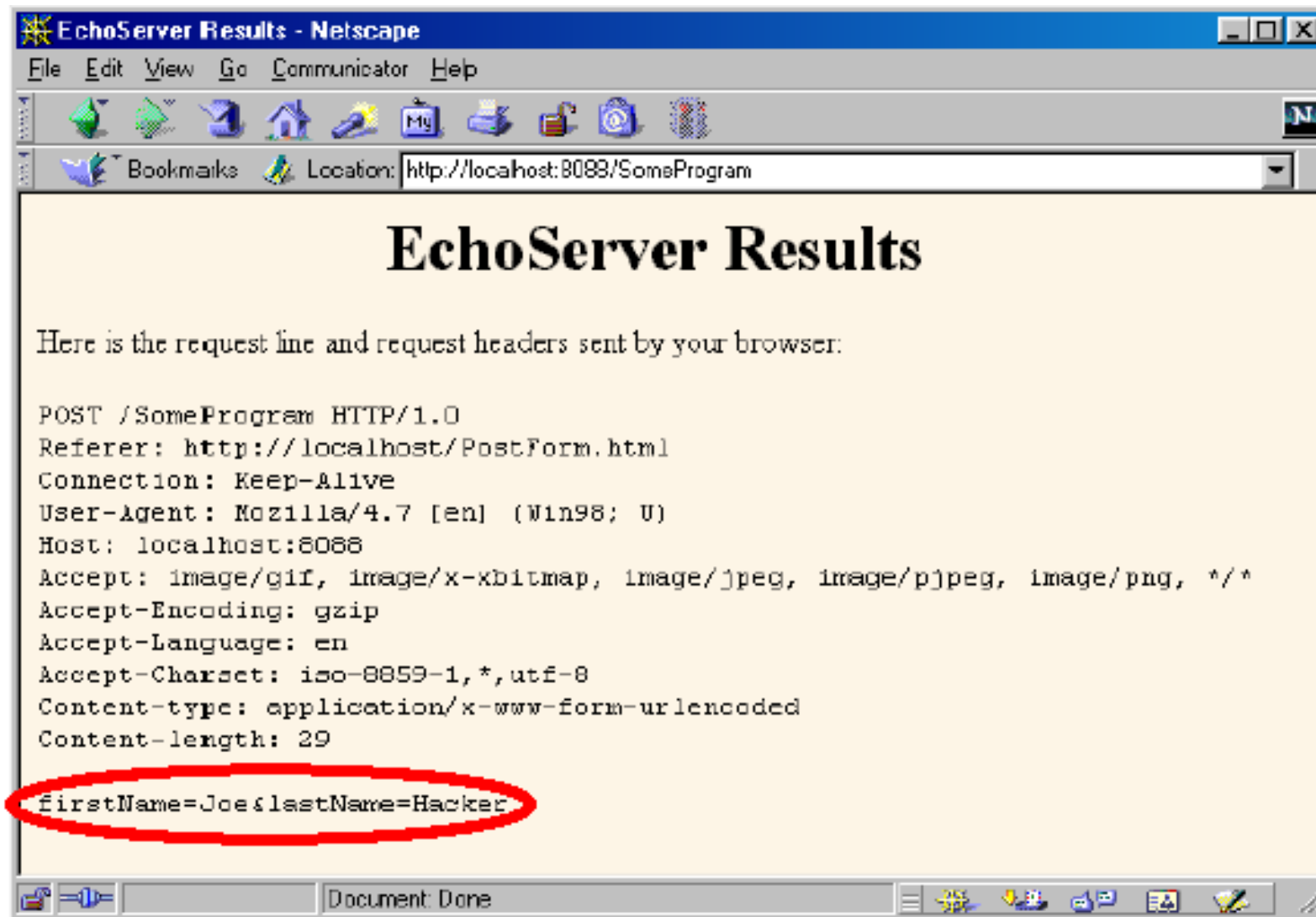


HTML Form with POST

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>A Sample Form Using POST</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using POST</H2>

<FORM ACTION="http://localhost:8080/SomeProgram" METHOD="POST">
<CENTER>
First name:
<INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
Last name:
<INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
<INPUT TYPE="SUBMIT">
</CENTER>
</FORM>
</BODY></HTML>
```

Sending POST Data



Servlet: get the parameters...

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Reading Parameters";

    out.println("<HTML>" +
        "<BODY><H1 ALIGN=CENTER>" + title + "</H1>\n" +
        "<UL>\n" +
        "  <LI><B>firstName</B>: "

        + request.getParameter("firstName") + "\n" +

        "  <LI><B>lastName</B>: "

        + request.getParameter("lastName") + "\n" +

        "</UL>\n" +
        "</BODY></HTML>");
}
```

FORMs HTML: Input Fields

```
<html>
<body>

<form name="input" action="html_form_action.asp" method="get">

Type your first name:
<input type="text" name="FirstName" value="Mickey" size="20">
<br>Type your last name:
<input type="text" name="LastName" value="Mouse" size="20">
<br>
<input type="submit" value="Submit">

</form>

<p>
If you click the "Submit" button, you will send your input to a new page
called html_form_action.asp.
</p>

</body>
</html>
```

Type your first name:

Type your last name:

If you click the "Submit" button, you will send your input to a new page called html_form_action.asp.

FORMs HTML: Checkbox

```
<html>
<body>

<form name="input" action="html_form_action.asp" method="get">
I have a bike:
<input type="checkbox" name="Bike" checked="checked"><br>
I have a car:
<input type="checkbox" name="Car">
<br>
<input type="submit" value="Submit">
</form>

<p>
If you click the "Submit" button, you send your input to a new page
called html_form_action.asp
</p>

</body>
</html>
```

I have a bike: ☒

I have a car: ☐

If you click the "Submit" button, you send your input to a new page called
html_form_action.asp.

FORMs HTML: RadioButton

```
<html>
<body>

<form name="input" action="html_form_action.asp" method="get">
Male:
<input type="radio" name="Sex" value="Male" checked="checked">
<br>
Female:
<input type="radio" name="Sex" value="Female">
<br>
<input type="submit" value="Submit">
</form>

<p>
If you click the "Submit" button, you will send your input to a new page
called html_form_action.asp.
</p>

</body>
</html>
```

Male: ☒
Female: ☐

If you click the "Submit" button, you will send your input to a new page called html_form_action.asp.

Mais Info sobre HTML Forms:

- http://www.w3schools.com/html/html_forms.asp

HTTP Response



Setting Status Codes

- **response.setStatus(int statusCode)**
Use a constant for the code, not an explicit int.
E.g., SC_OK, SC_NOT_FOUND, etc.
- **response.sendError(int code, String message)**
Wraps message inside small HTML document
- **response.sendRedirect(String url)**
Sets Location header also

Common Status Codes

- **200 (OK)**
 - Everything is fine; document follows.
- **204 (No Content)**
 - Browser should keep displaying previous document.
- **301 (Moved Permanently)**
- **302 (Found)**
 - Requested document temporarily moved elsewhere
 - Browsers go to new location automatically.
 - Servlets should use `sendRedirect`,
- **401 (Unauthorized)**
 - Browser tried to access password-protected page without proper Authorization header.
- **404 (Not Found)**

Response Header

- The Response header is used to submit a variety of information back to client
 - **setHeader(String name, String value)**
 - sets a Header with given name & value. Overwrites existing values.
- HTTP Response headers include:
 - Cache-Control, Content-Type, Content-Length, Content-Location, Connection, Content-Encoding, Allow

Other Response Methods

- Some convenience methods provided are:
 - **setContentType**
 - sets the Content-Type Header which specifies the Multipurpose Internet Mail Extension(MIME) type of the response document eg. **text/html** for **HTML pages**.

Response convenience methods

- **sendError**
 - send some error information to the client. It sets the header and content body to appropriate values.
- **sendRedirect**
 - sets headers and content body to redirect the client to a different URL.

Cookies



Why Cookies?...

Basic Idea

- Servlet sends a simple name and value to client.
 - Client returns same name and value when it connects to same site
-
- Identifying a user during a session.
 - Avoiding username and password.
 - Customizing a portal site.
 - Focusing advertising.

Security Threat?

- Cookies are never interpreted or executed in any way, and thus cannot be used to insert viruses or attack your system in any way.
- Browsers generally only accept 20 cookies per site and 300 cookies total, and each cookie is limited to 4KB, cookies cannot be used to fill up someone's disk or launch other denial of service attacks.

Add a Cookie to the Response Header

```
Cookie userCookie = new Cookie("user", "uid1234");  
response.addCookie(userCookie);
```

HTTP RESPONSE

HTTP REQUEST

```
// To read incoming cookies:  
request.getCookies();  
  
// returns an array of Cookie objects.
```

get/set Cookie Attributes

- getComment/setComment
- getDomain/setDomain
- getMaxAge/setMaxAge
- getName/setName
- getPath/setPath
- getSecure/setSecure
- getValue/setValue
- getVersion/setVersion

Session Tracking



Techniques for Session Tracking

- **Cookies**

- **URL Rewriting**

You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.

Ex: `http://host/path/file.html;jsessionid=1234`

- **Hidden form fields**

`(<INPUT TYPE="HIDDEN" NAME="session" VALUE="...">.`

When the form is submitted, the specified name and value are included in the GET or POST data.

Session Tracking API

- Session objects live on the server
- Use `request.getSession(true)` to get either existing or new session
- Hashtable-like mechanism lets you store arbitrary objects inside session:
 - `setAttribute` stores values
 - `getAttribute` retrieves values

HttpSession Object

```
HttpSession session = request.getSession();  
HttpSession session = request.getSession(false);
```

Methods to use with HttpSession:

```
getId()  
getAttribute()  
setAttribute()  
getAttributeNames()  
setMaxInactiveInterval() // seconds  
isNew()  
getCreationTime()  
getLastAccessedTime()  
invalidate()
```

Using Sessions...

```
HttpSession s=request.getSession()  
    // creates a new session  
    // or if the session already exists returns the session id  
  
if(s.isNew())        // true if no response yet with that session  
    .... // this is a new session  
  
else  
    .... // existing session
```

Alternativa: Sessions...

```
HttpSession s=request.getSession(false)
```

```
    // returns an existing session
```

```
    // or null if there is no session associated with this client
```

```
if(s == null)
```

```
    ....
```

```
    s = request.getSession();
```

```
else
```

```
    .... // there is an existing session
```

Session Management

- `session.setMaxInactiveInterval(60);`
 - `// sessão permite 1 minuto de inatividade`
- `session.invalidate()`
 - `// invalidar uma sessão`