Vietnam National University of HCMC
International University
School of Computer Science and Engineering

Web Application Development (IT093IU)

Assoc. Prof. Nguyen Van Sinh
Email: nvsinh@hcmiu.edu.vn

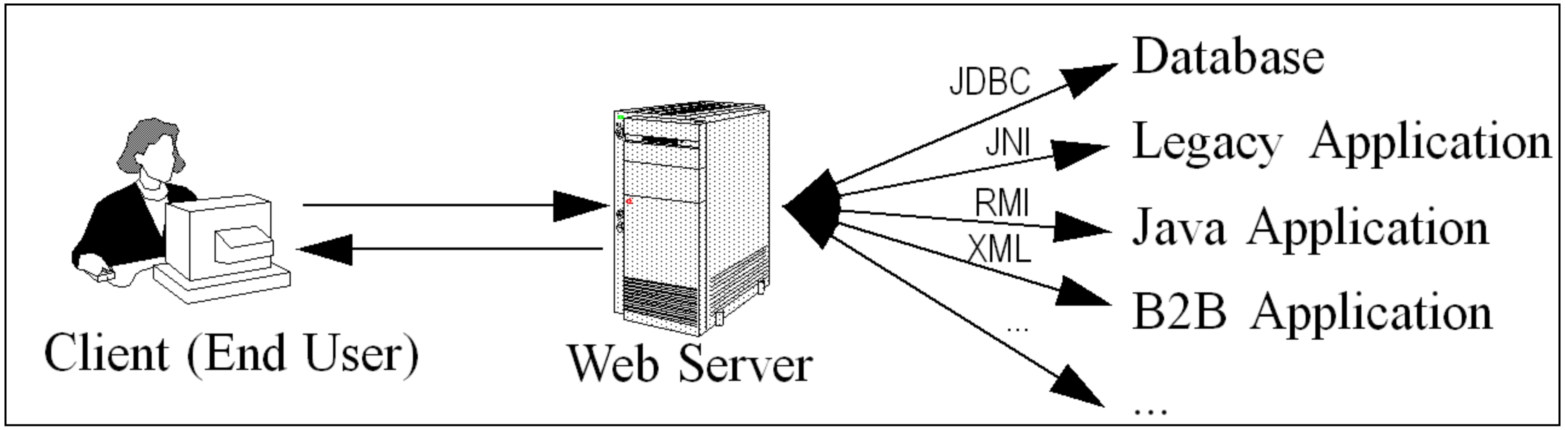
(Semester 2, 2023-2024)

Servlet (lecture 6 and 7)

- What servlets are all about
- Advantages of servlets
- What JSP is all about
- Free servlet and JSP engines
- Compiling and invoking servlets
- Servlet structure, examples
- Servlet lifecycle
- Initializing servlets
- Debugging servlets

Duties of the servlet

- Read explicit data sent by client (form data)
- Read implicit data sent by client (request headers)
- Generate the results
- Send the explicit data back to client (HTML)
- Send the implicit data to client (status codes and response headers)



- It is processed completely on the server side

Why it build a web page dynamically?

- The web page is based on data submitted by the user
  - E.g., results page from search engines and order-confirmation pages at on-line stores
- The web page is derived from data that changes frequently
  - E.g., a headlines-weather report or news page
- The web page uses information from databases or other server-side sources
  - E.g., an e-commerce site could use a servlet to build a Web page that lists the current price and availability of each item that is for sale.

The advantages of servlets

- Efficient
  - Threads instead of OS processes, one servlet copy, persistence
- Convenient
  - Lots of high-level utilities
- Powerful
  - Sharing data, pooling, persistence
- Portable
  - Run on virtually all operating systems and servers
- Secure
  - No shell escapes, no buffer overflows
- Inexpensive
  - There are plenty of free and low-cost servers.

Extending the power of servlets: java server pages (JSP)

- Idea:
  - Use regular HTML for most of page
  - Mark dynamic content with special tags
  - Details in second half of course

```
<HTML>
<HEAD><TITLE>Welcome to Our Store</TITLE></HEAD>
<BODY>
<H1>Welcome to Our Store</H1>
<SMALL>Welcome,
<!-- User name is "New User" for first-time visitors -->
<%= coreservlets.Utils.getUserNameFromCookie(request) %>
To access your account settings, click
<A HREF="Account-Settings.html">here.</A></SMALL>
<P>
Regular HTML for  rest of on-line store’s Web page
</BODY></HTML>
```

Server-side java is driving the web

Get on board or get out of the way

The references for servlets and JSP

- Apache Tomcat
  - http://tomcat.apache.org
- References:
  - https://docs.oracle.com/javaee/6/api/javax/servlet/Servlet.html
  - http://www.servlets.com
  - https://www.javaguides.net/p/servlet-tutorial.html
  - https://www.javatpoint.com/servlet-tutorial
  - https://www.tutorialspoint.com/servlets/index.htm

Compiling and Invoking Servlets

The following steps to create, compile, invoke and access a servlet:

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

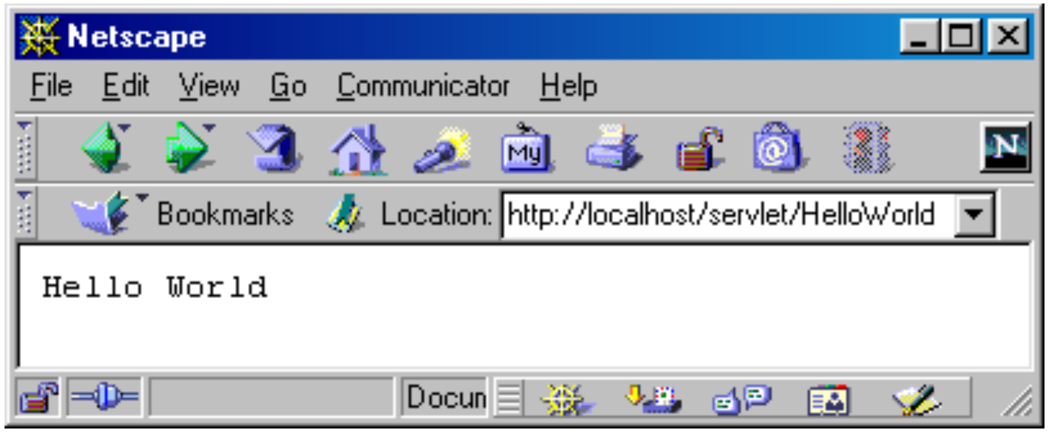
… it is easy implemented on NetBeans

Simple Servlet Template

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletTemplate extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

    // Use "request" to read incoming HTTP headers
    // (e.g. cookies) and HTML form data (query data)
    // Use "response" to specify the HTTP response status
    // code and headers (e.g. the content type, cookies).

    PrintWriter out = response.getWriter();
    // Use "out" to send content to browser
  }
}
```

A simple Servlet to print a plain text

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

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



http://localhost/servlet/HelloWorld

```
Hello World
```

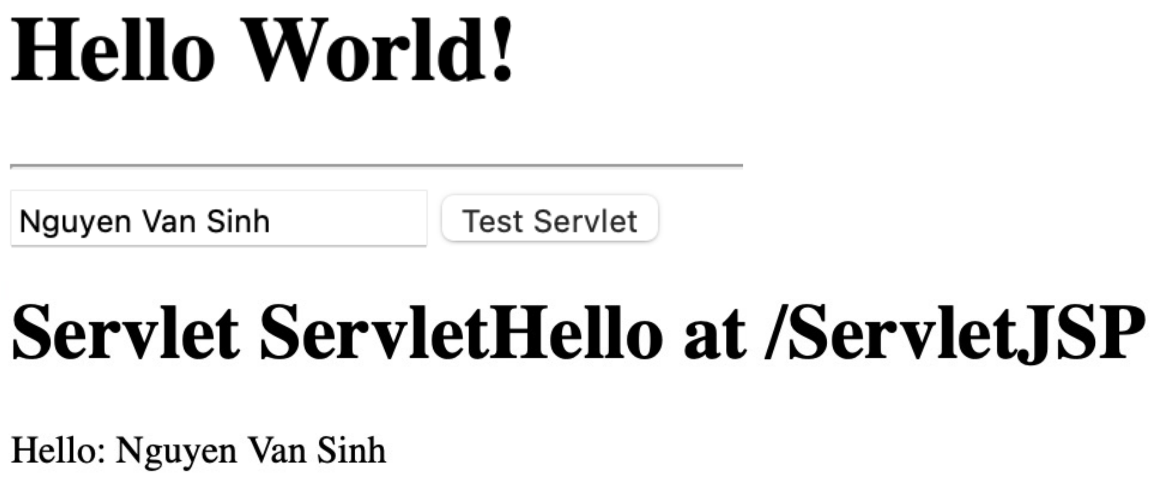
Example: a jsp page call a servlet

TestServlet.jsp

```
<body>
    <h1>Hello World!</h1><hr>
    <form method="post" action="ServletHello">
    <input type="text" name="user">
    <input type="submit" name="Login" value="Test Servlet">
</form>
</body>
```

ServletHello.java

```
out.println("Hello: " + request.getParameter("user"));
```



Hello World!

Servlet ServletHello at /ServletJSP

Hello: Nguyen Van Sinh

Generating HTML

- Set the Content-Type header
  - Use response.setContentType("text/html");
- Output HTML
  - Be sure to include the DOCTYPE
- Use an HTML validation service
  - http://validator.w3.org/
  - If your servlets are behind a firewall, you can run them, save the HTML output, and use a file upload form to validate.

A Servlet That Generates HTML

```
public class HelloWWW extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Hello WWW</TITLE></HEAD>\n" +
                "<BODY>\n" +
                "<H1>Hello WWW</H1>\n" +
                "</BODY></HTML>");
  }
}
```

Packaging Servlets

- Move the files to a subdirectory that matches the intended package name
  - For example, I'll use the coreservlets package for most of the rest of the servlets in this course. So, the class files need to go in a subdirectory called coreservlets.
- Insert a package statement in the class file
  - E.g., top of HelloWWW2.java:
    `package coreservlets;`
- Set CLASSPATH to top-level directory
  - E.g., C:\Servlets+JSP.
- Include package name in URL
  - http://localhost/servlet/coreservlets.HelloWWW2

Some simple HTML-building utilities

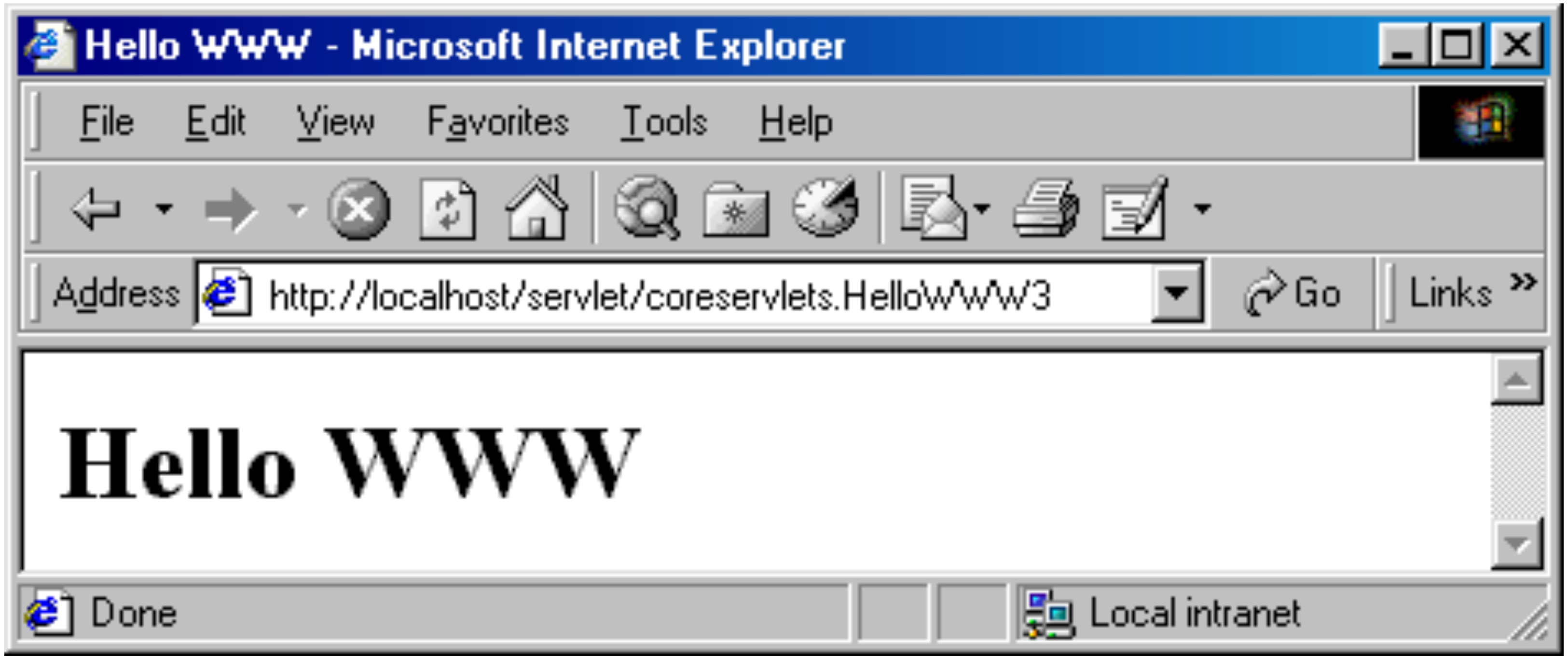
```
public class ServletUtilities {
  public static final String DOCTYPE =
    "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
    "Transitional//EN\">";

  public static String headWithTitle(String title) {
    return(DOCTYPE + "\n" +
           "<HTML>\n" +
           "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n");
  }
  ...
}
```

- Don’t go overboard
  - Complete HTML generation packages usually work poorly
  - The JSP framework is a better solution

Hello WWW with ServletUtilities

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

public class HelloWWW3 extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println(ServletUtilities.headWithTitle("Hello WWW") +
                "<BODY>\n" +
                "<H1>Hello WWW</H1>\n" +
                "</BODY></HTML>");
  }
}
```

Hello WWW Result

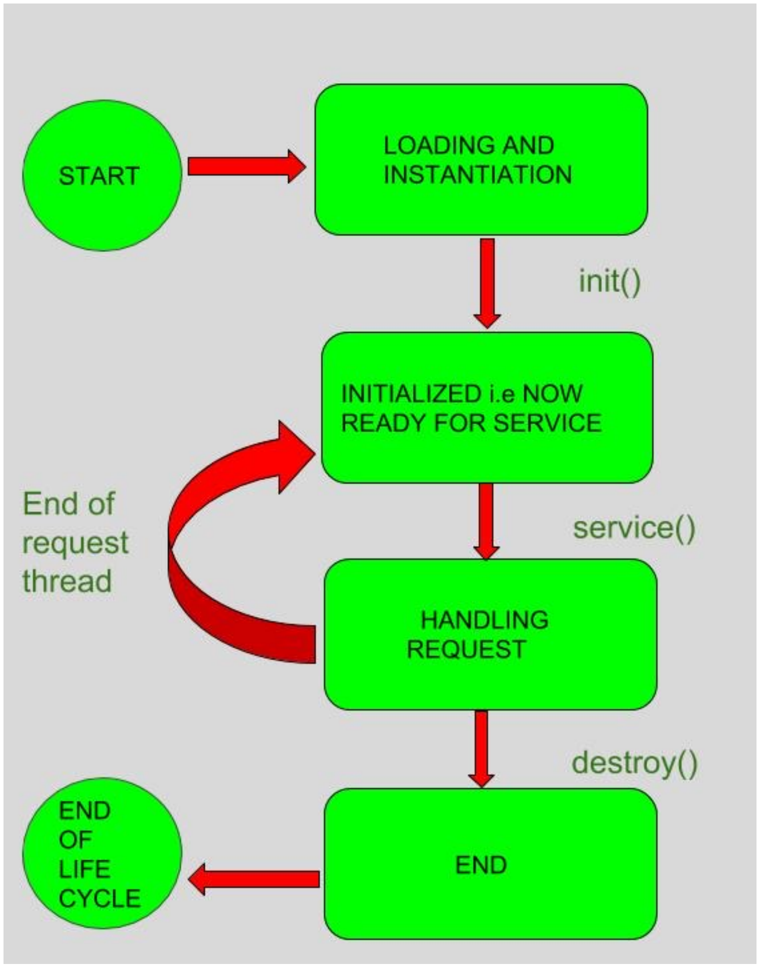

Hello WWW - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
Links
Done
Local intranet

http://localhost/servlet/coreservlets.HelloWWW3

Hello WWW

The Servlet Life Cycle

- init
  - Executed once when the servlet is first loaded.
    Not called for each request.
- service
  - Called in a new thread by server for each request. Dispatches to doGet, doPost, etc.
    Do not override this method!
- doGet, doPost, etc.,
  - Handles GET, POST, etc. requests.
  - Override these to provide desired behavior.
- destroy
  - Called when server deletes servlet instance.
    Not called after each request.



Why you should *not* override service

- You can add support for other services later by adding doPut, doTrace, etc.
- You can add support for modification dates by adding a getLastModified method
- The service method gives you automatic support for:
  - HEAD requests
  - OPTIONS requests
  - TRACE requests
- Alternative: have doPost call doGet

Initializing Servlets

- Common in real-life servlets
  - E.g., initializing database connection pools.
- Use ServletConfig.getInitParameter to read initialization parameters
- Set init parameters in web.xml
  - …/WEB-INF/web.xml
  - Many servers have custom interfaces to create web.xml
- It is common to use init even when you don’t read init parameters
  - See modification date example in *Core Servlets and JavaServer Pages* Chapter 2

Debugging Servlets

- Use print statements; run server on desktop
- Integrated debugger in IDE
- Look at the HTML source
- Return error pages to the client
  - for any problem
- Use the log file
  - log("message") or log("message", Throwable)
- Separate the request and response data .
  - Request: see EchoServer at www.coreservlets.com
  - Response: see WebClient at www.coreservlets.com
- Stop and restart the server

Summary

- Servlets are efficient, portable, powerful, and widely accepted in industry
- Regardless of deployment server, run a free server on your desktop for development
- Getting started:
    - Set your CLASSPATH
        - Servlet JAR file (existing libraries)
        - Top of your package hierarchy
    - Put class files in proper location
        - .../WEB-INF/classes
    - Use proper URL, usually http://*host*/servlet/*ServletName*
- Download existing servlet first time
    - Start with HelloWWW from www.coreservlets.com

Summary (Continued)

- Main servlet code goes in doGet or doPost:
  - The HttpServletRequest contains the incoming information
  - The HttpServletResponse lets you set outgoing information
    - Call setContentType to specify MIME type
    - Call getWriter to obtain a Writer pointing to client
- One-time setup code goes in init
  - Servlet gets initialized and loaded once
  - Servlet gets invoked multiple times
  - Initialization parameters set in web.xml (covered in detail in *More Servlets & JavaServer Pages* Chapter 5)

Handling the Client Request: Form Data

Agenda

- Why form data is important
- Processing form data in traditional CGI
- Processing form data in servlets
- Reading individual request parameters
- Reading all request parameters
- Real-life servlets: handling malformed data
- Filtering HTML-specific characters

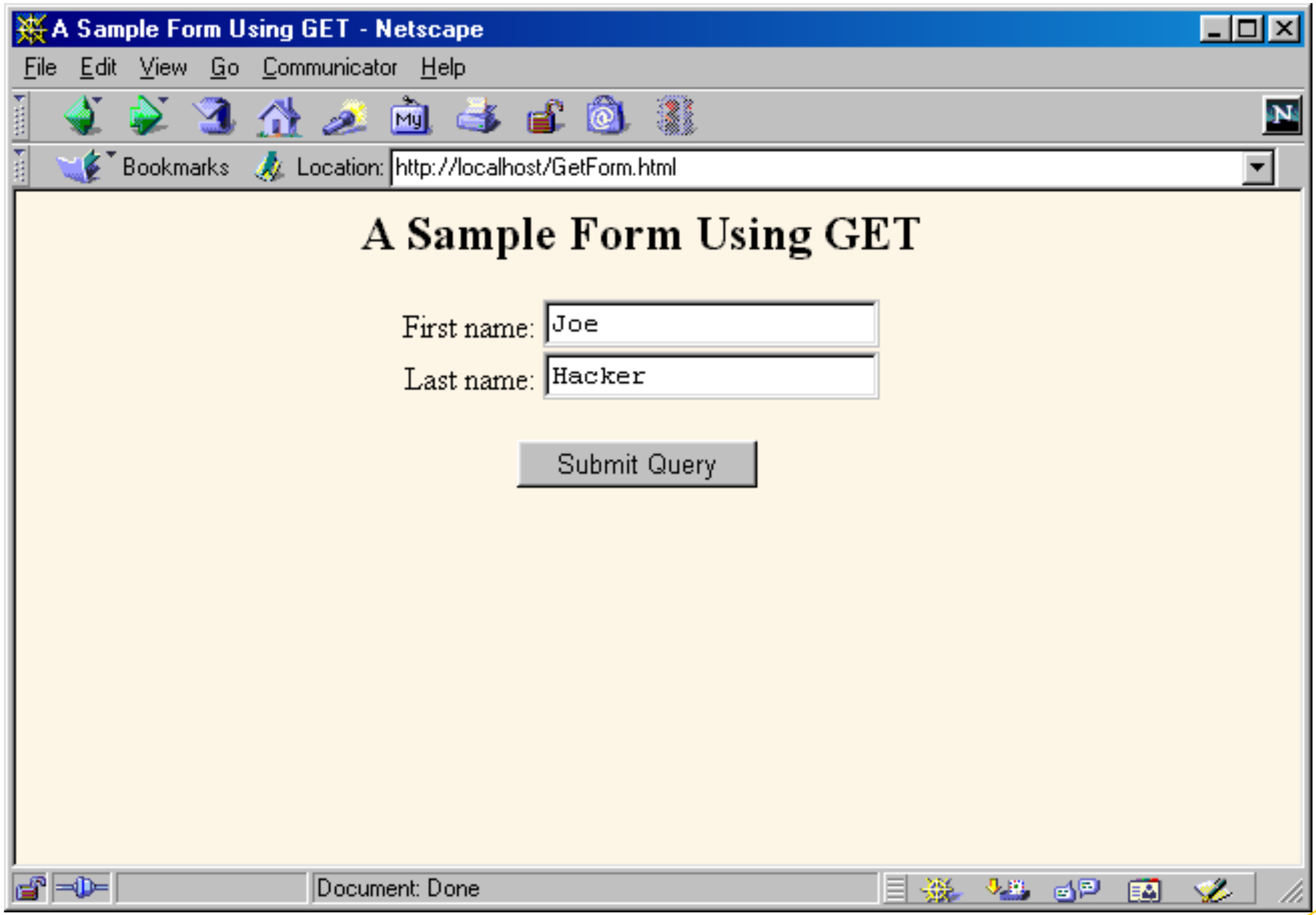
The role of form data

- Example URL at online travel agent
  - http://host/path?user=Marty+Hall&origin=bwi&dest=lax
  - Names come from HTML author; values usually come from end user
- Parsing form (query) data in traditional CGI
  - Read the data one way (QUERY_STRING) for GET requests, another way (standard input) for POST requests
  - Chop pairs at ampersands, then separate parameter names (left of the equal signs) from parameter values (right of the equal signs)
  - Need special cases for omitted values (param1=val1&param2=val2&param3=val3) and repeated parameters (param1=val1&param2=val2&param1=val3)

Creating Form Data: HTML Forms

```
<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: Initial Result


A Sample Form Using GET - Netscape
File Edit View Go Communicator Help
Bookmarks
Location:

http://localhost/GetForm.html

A Sample Form Using GET

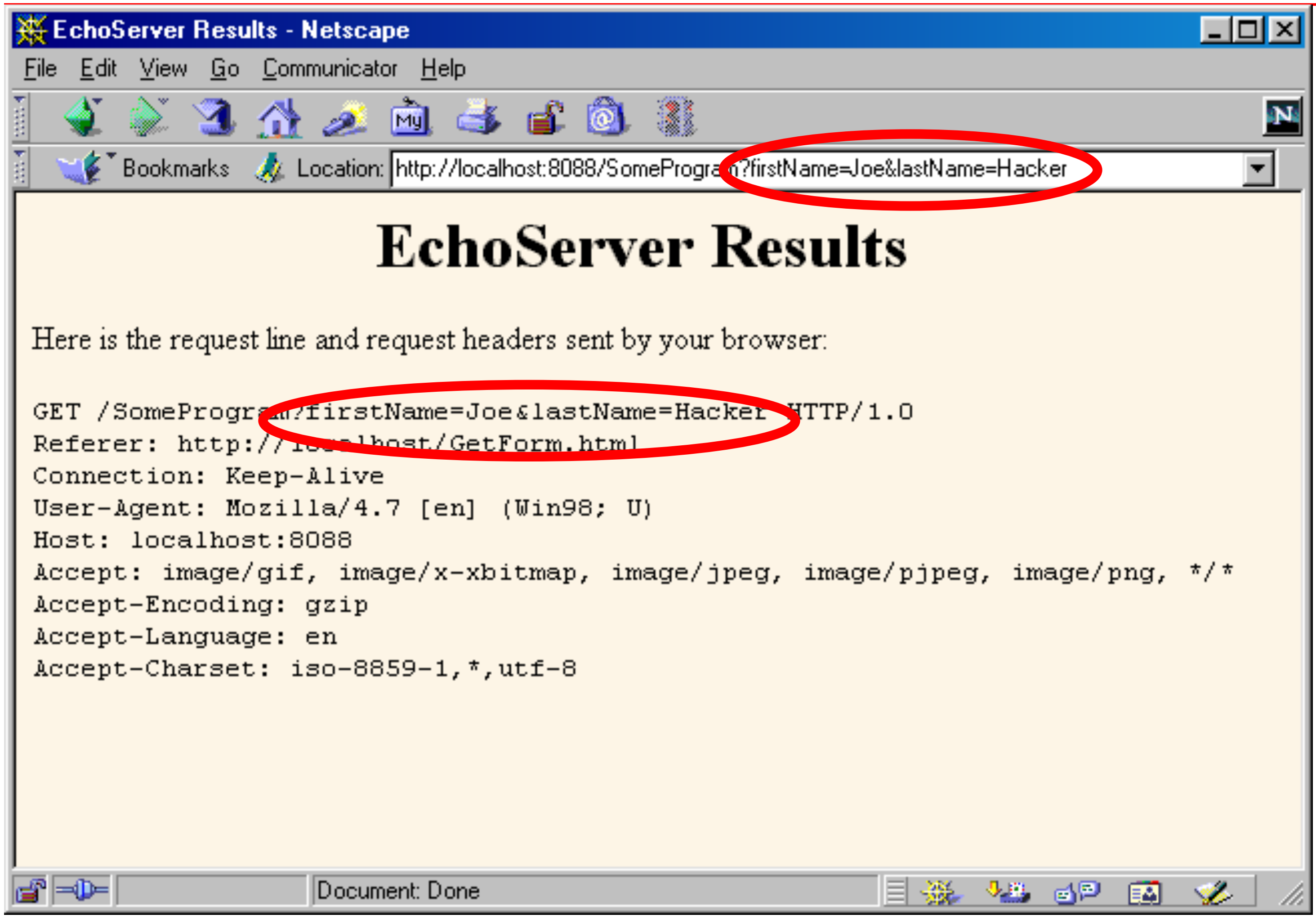
First name: Joe

Last name: Hacker

Submit Query

Document: Done

HTML Form: Submission Result (Data Sent to EchoServer)


EchoServer Results - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://localhost:8088/SomeProgram?firstName=Joe&lastName=Hacker
Document: Done

EchoServer Results

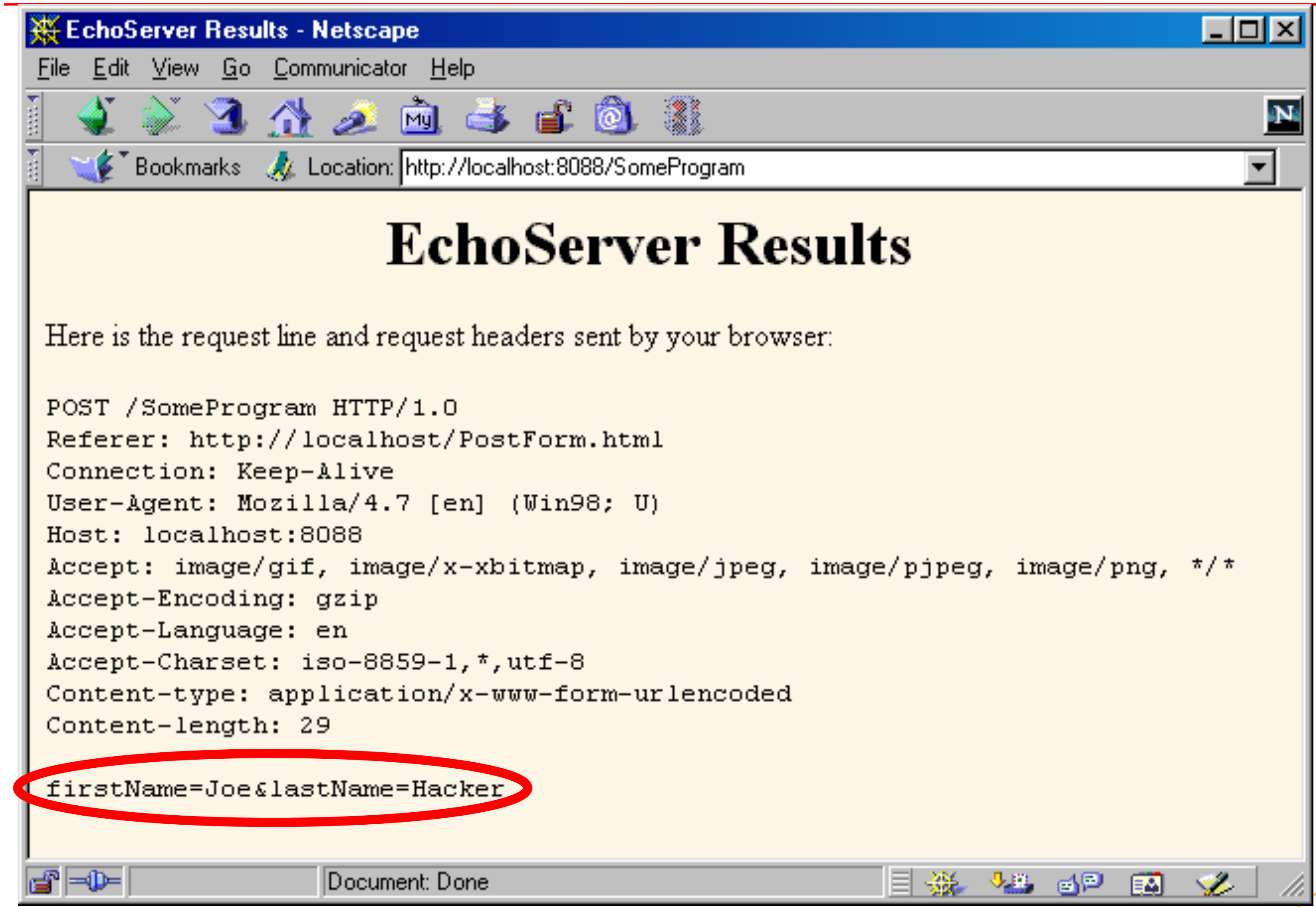
Here is the request line and request headers sent by your browser:

```
GET /SomeProgram?firstName=Joe&lastName=Hacker HTTP/1.0
Referer: http://localhost/GetForm.html
Connection: Keep-Alive
User-Agent: Mozilla/4.7 [en] (Win98; U)
Host: localhost:8088
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Sending POST Data

```
<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:8088/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


EchoServer Results - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost:8088/SomeProgram
Document: Done

EchoServer Results

Here is the request line and request headers sent by your browser:

```
POST /SomeProgram HTTP/1.0
Referer: http://localhost/PostForm.html
Connection: Keep-Alive
User-Agent: Mozilla/4.7 [en] (Win98; U)
Host: localhost:8088
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 29

firstName=Joe&lastName=Hacker
```

Reading form data in servlets

- request.getParameter("name")
  - Returns URL-decoded value of first occurrence of name in query string
  - Works identically for GET and POST requests
  - Returns null if no such parameter is in query
- request.getParameterValues("name")
  - Returns an array of the URL-decoded values of all occurrences of name in query string
  - Returns a one-element array if param not repeated
  - Returns null if no such parameter is in query
- request.getParameterNames()
  - Returns Enumeration of request params

Handling input in multiple languages

- Use server's default character set

```
String firstName =
  request.getParameter("firstName");
```

- Convert from English (Latin-1) to Japanese

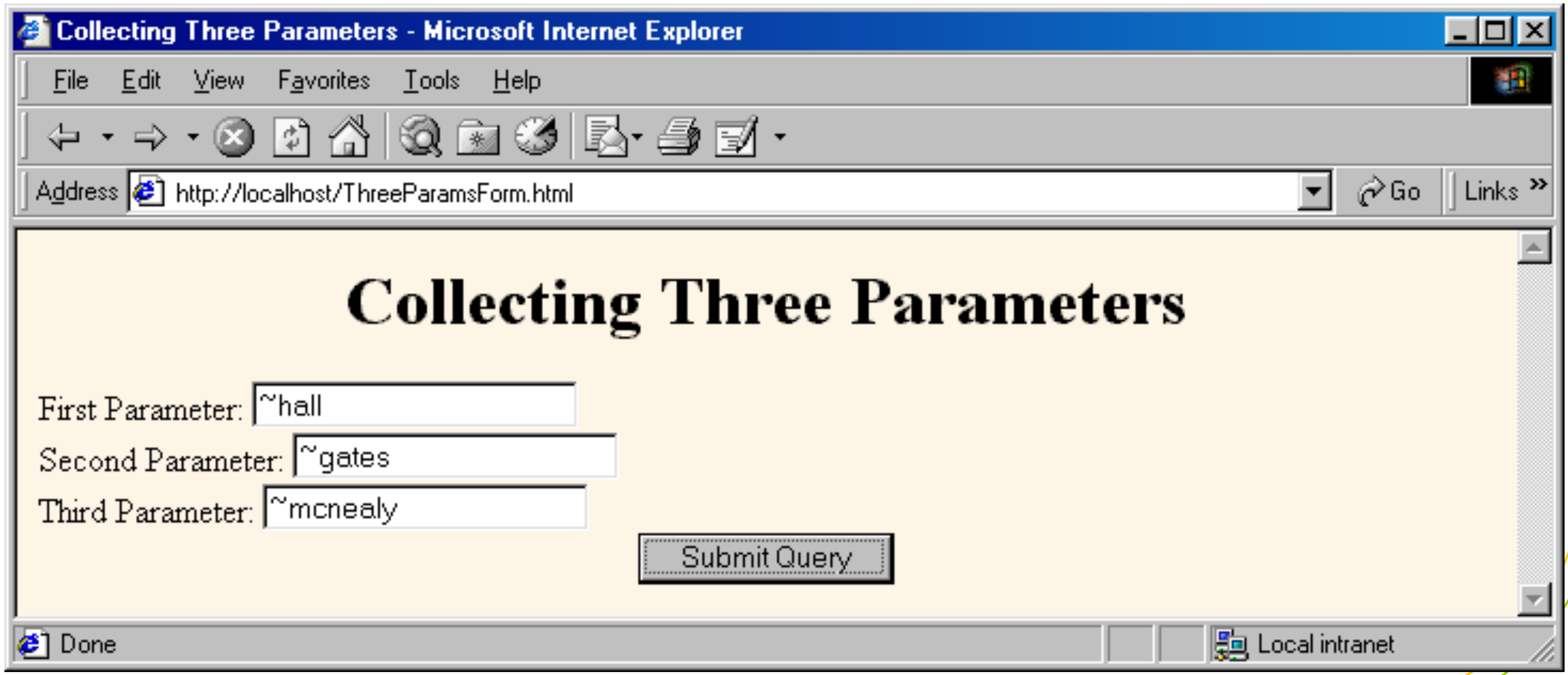
```
String firstNameWrongEncoding =
  request.getParameter("firstName");
String firstName =
  new String(firstNameWrongEncoding.getBytes(),
             "Shift_JIS");
```

- Accept either English or Japanese

```
request.setCharacterEncoding("JISAutoDetect");
String firstName =
  request.getParameter("firstName");
```

An HTML Form With Three Parameters

```
<FORM ACTION="/servlet/coreservlets.ThreeParams">
  First Parameter:  <INPUT TYPE="TEXT" NAME="param1"><BR>
  Second Parameter: <INPUT TYPE="TEXT" NAME="param2"><BR>
  Third Parameter:  <INPUT TYPE="TEXT" NAME="param3"><BR>
  <CENTER><INPUT TYPE="SUBMIT"></CENTER>
</FORM>
```


Collecting Three Parameters - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
Links
Done
Local intranet

http://localhost/ThreeParamsForm.html

Collecting Three Parameters

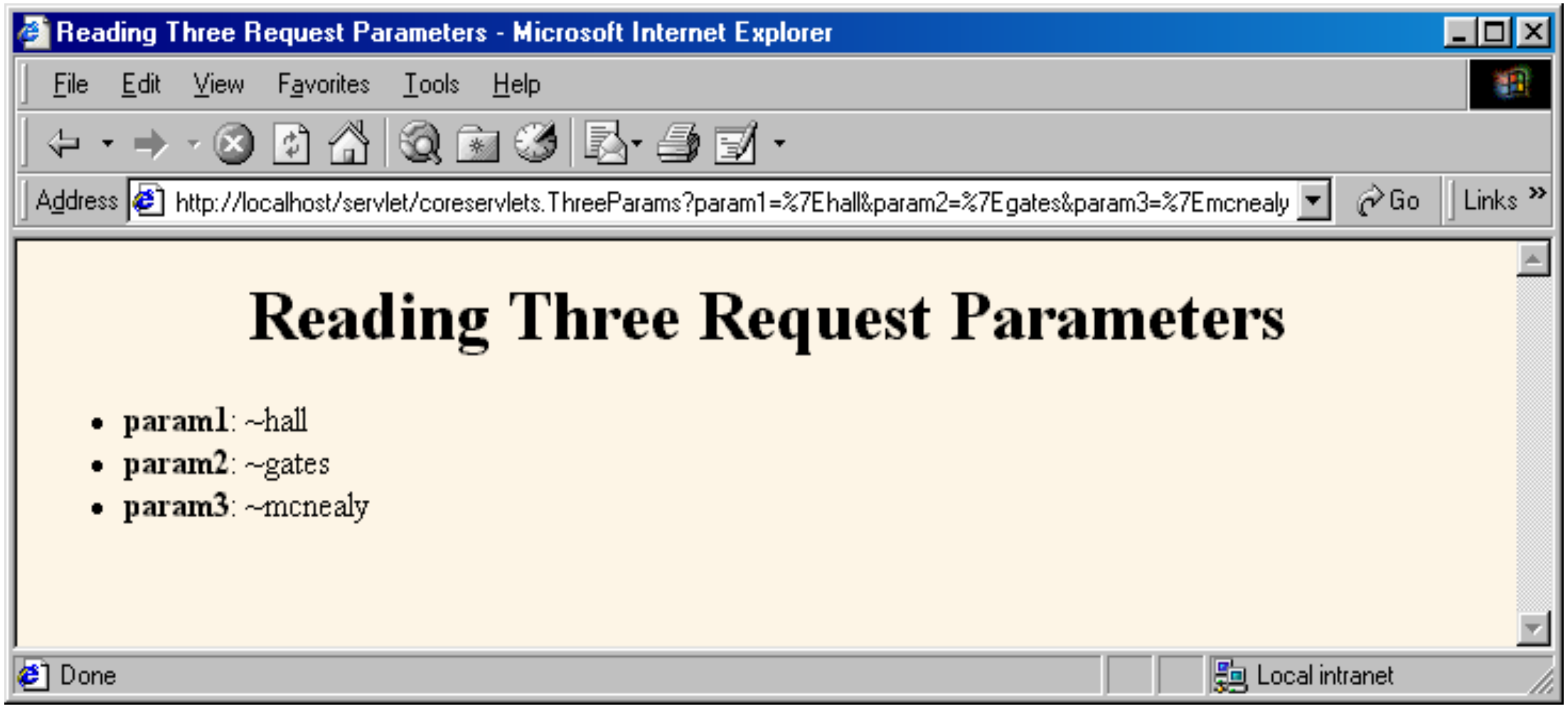
First Parameter: ~hall
Second Parameter: ~gates
Third Parameter: ~mcnealy

Submit Query

Reading the three parameters

```
public class ThreeParams extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Reading Three Request Parameters";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                "<UL>\n" +
                "  <LI><B>param1</B>: "
                + request.getParameter("param1") + "\n" +
                "  <LI><B>param2</B>: "
                + request.getParameter("param2") + "\n" +
                "  <LI><B>param3</B>: "
                + request.getParameter("param3") + "\n" +
                "</UL>\n" +
                "</BODY></HTML>"); }}
```

Reading three parameters: Result


Reading Three Request Parameters - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
Links
Done
Local intranet

http://localhost/servlet/coreservlets.ThreeParams?param1=%7Ehall&param2=%7Egates&param3=%7Emcnealy

Reading Three Request Parameters

- **param1**: ~hall
- **param2**: ~gates
- **param3**: ~mcnealy

Reading all parameters

```
public class ShowParameters extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Reading All Request Parameters";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                "<TABLE BORDER=1 ALIGN=CENTER>\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "<TH>Parameter Name<TH>Parameter
   Value(s)");
```

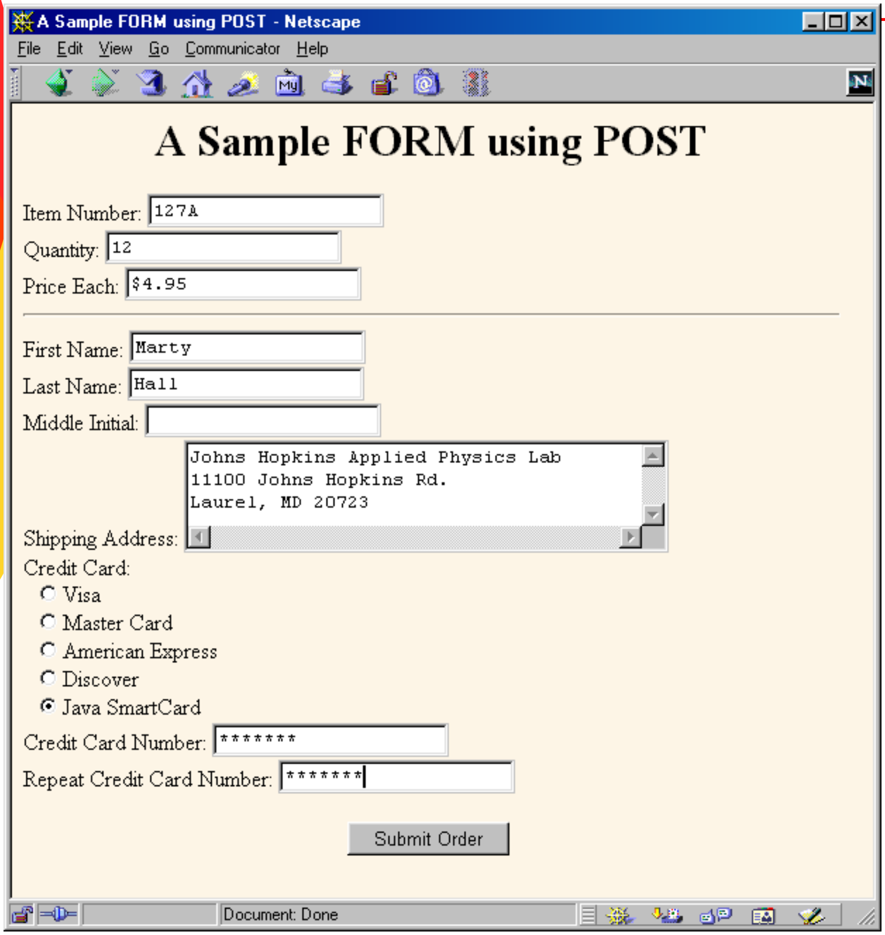
Reading all parameters (continued)

```
 Enumeration paramNames =
request.getParameterNames();
  while(paramNames.hasMoreElements()) {
    String paramName =
(String)paramNames.nextElement();
    out.print("<TR><TD>" + paramName + "\n<TD>");
    String[] paramValues =
      request.getParameterValues(paramName);
    if (paramValues.length == 1) {
      String paramValue = paramValues[0];
      if (paramValue.length() == 0)
        out.println("<I>No Value</I>");
      else
        out.println(paramValue);
```

Reading all parameters (continued)

```
      } else {
        out.println("<UL>");
        for(int i=0; i<paramValues.length; i++) {
          out.println("<LI>" + paramValues[i]);
        }
        out.println("</UL>");
      }
    }
    out.println("</TABLE>\n</BODY></HTML>");
  }

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

Result of showParameters servlet


A Sample FORM using POST - Netscape
File Edit View Go Communicator Help
Submit Order
Document: Done

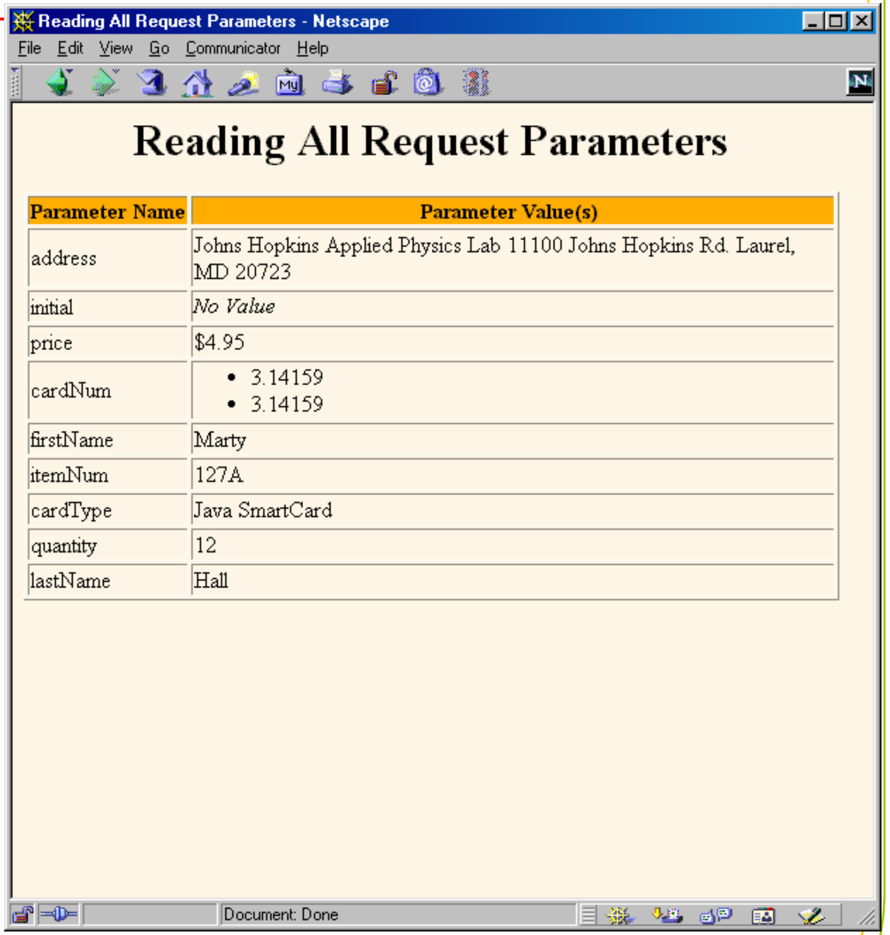
A Sample FORM using POST

Item Number:	127A
Quantity:	12
Price Each:	$4.95
First Name:	Marty
Last Name:	Hall
Middle Initial:	
Shipping Address:	Johns Hopkins Applied Physics Lab 11100 Johns Hopkins Rd. Laurel, MD 20723

Credit Card:
- Visa
- Master Card
- American Express
- Discover
- [✓ Java SmartCard]

Credit Card Number: *******

Repeat Credit Card Number: *******


Reading All Request Parameters - Netscape
File Edit View Go Communicator Help
Document: Done

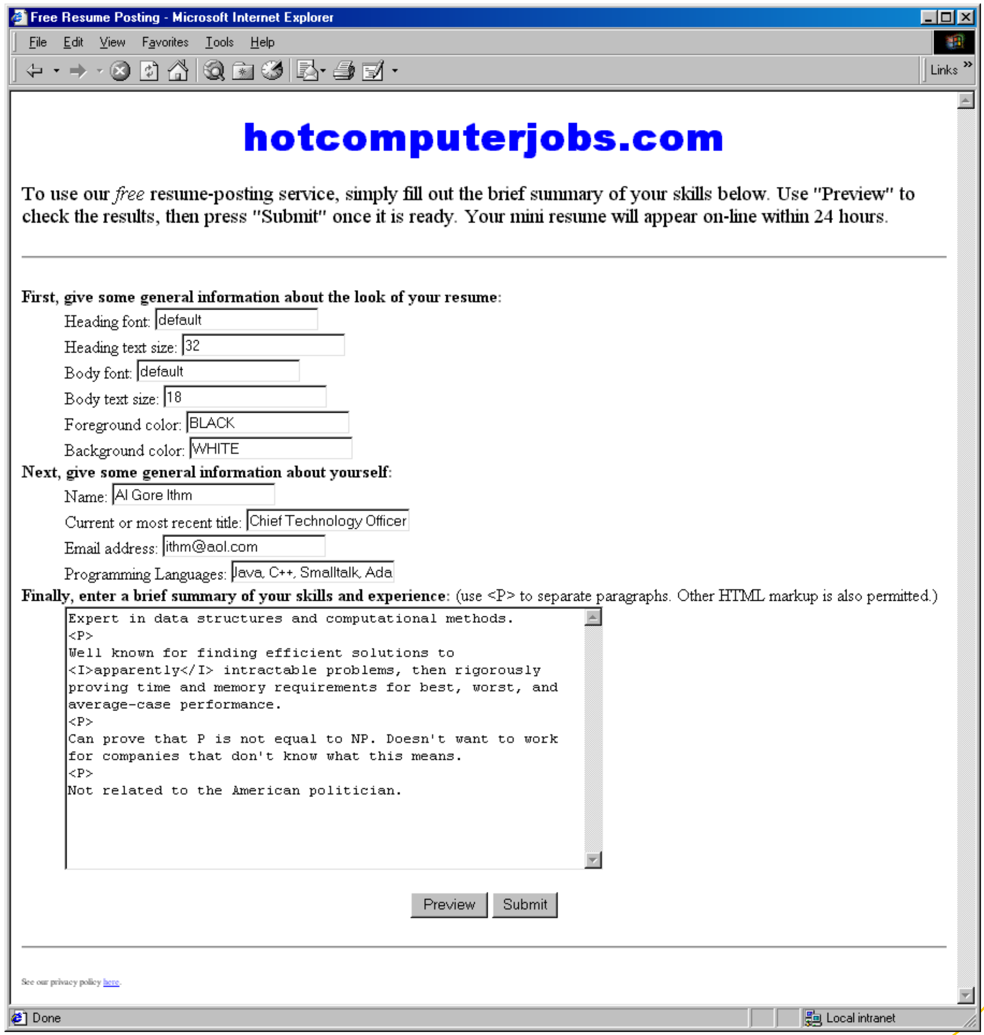
Reading All Request Parameters

Parameter Name	Parameter Value(s)
address	Johns Hopkins Applied Physics Lab 11100 Johns Hopkins Rd. Laurel, MD 20723
initial	*No Value*
price	$4.95
cardNum	• 3.14159 • 3.14159
firstName	Marty
itemNum	127A
cardType	Java SmartCard
quantity	12
lastName	Hall

Note that order of parameters in Enumeration does not match order they appeared in Web page

Posting service: Front End

- Gathers resume formatting and content information


Free Resume Posting - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
Local intranet

hotcomputerjobs.com

To use our *free* resume-posting service, simply fill out the brief summary of your skills below. Use "Preview" to check the results, then press "Submit" once it is ready. Your mini resume will appear on-line within 24 hours.

First, give some general information about the look of your resume:

Heading font: default
Heading text size: 32
Body font: default
Body text size: 18
Foreground color: BLACK
Background color: WHITE

Next, give some general information about yourself:

Name: Al Gore Ithm
Current or most recent title: Chief Technology Officer
Email address: ithm@aol.com
Programming Languages: Java, C++, Smalltalk, Ada

Finally, enter a brief summary of your skills and experience: (use <P> to separate paragraphs. Other HTML markup is also permitted.)

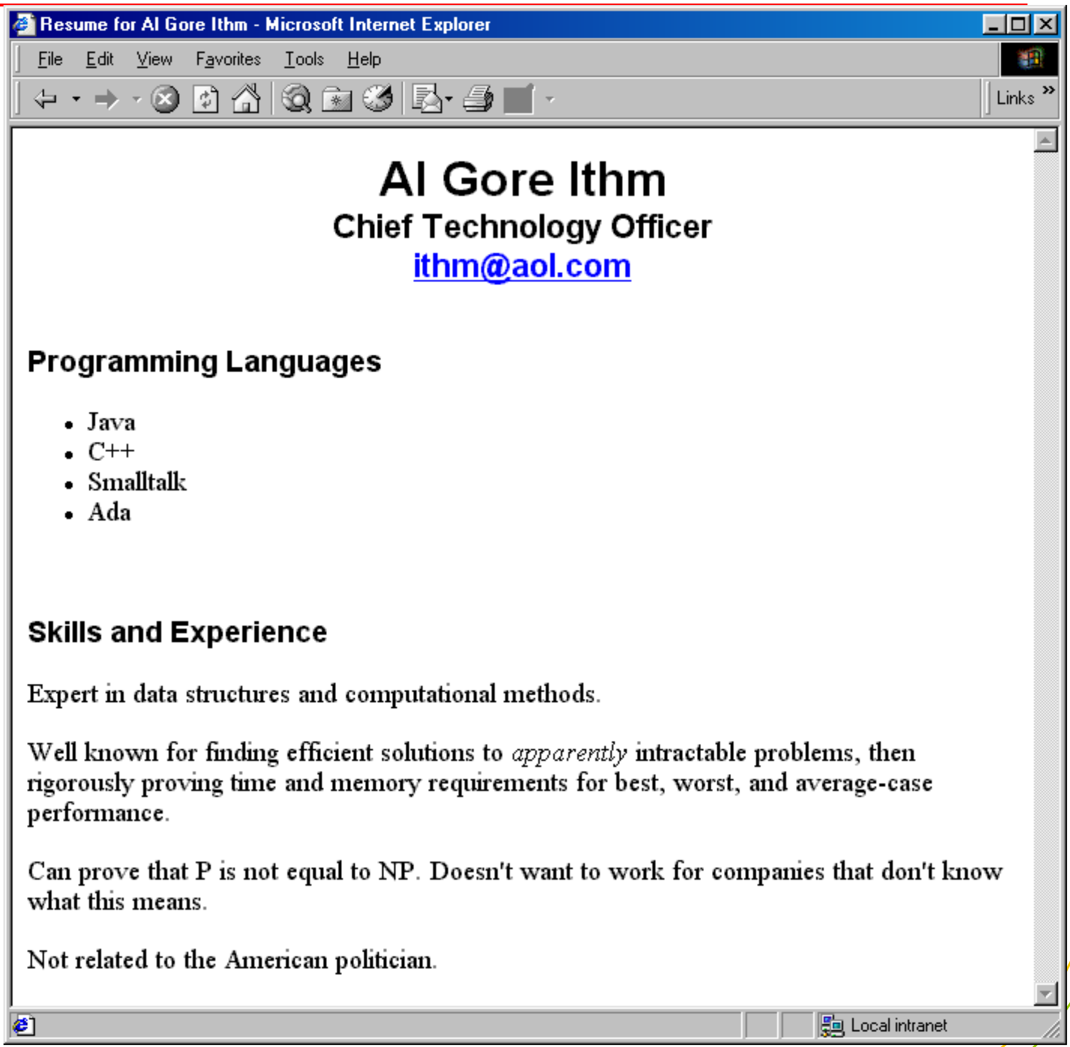
```
Expert in data structures and computational methods.
<P>
Well known for finding efficient solutions to
<I>apparently</I> intractable problems, then rigorously
proving time and memory requirements for best, worst, and
average-case performance.
<P>
Can prove that P is not equal to NP. Doesn't want to work
for companies that don't know what this means.
<P>
Not related to the American politician.
```

Preview Submit

See our privacy policy here.

Posting Service: Back End

- Previews result or stores resume in database


Resume for Al Gore Ithm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Local intranet

Al Gore Ithm
Chief Technology Officer
ithm@aol.com

Programming Languages

- Java
- C++
- Smalltalk
- Ada

Skills and Experience

Expert in data structures and computational methods.

Well known for finding efficient solutions to *apparently* intractable problems, then rigorously proving time and memory requirements for best, worst, and average-case performance.

Can prove that P is not equal to NP. Doesn't want to work for companies that don't know what this means.

Not related to the American politician.

Point: check for missing data

- Textfield was not in HTML form at all
  - request.getParameter returns null
- Textfield was empty when form was submitted
  - Request.getParameter returns an empty String
- Example Check

```
String value =
  request.getParameter("someName");
if ((value != null) &&
   (!value.equals("")) {
  …
}
```

Posting service: Servlet Code

```
private void showPreview(HttpServletRequest request,
                         PrintWriter out) {
  String headingFont = request.getParameter("headingFont");
  headingFont = replaceIfMissingOrDefault(headingFont, "");
  ...
  String name = request.getParameter("name");
  name = replaceIfMissing(name, "Lou Zer");
  String title = request.getParameter("title");
  title = replaceIfMissing(title, "Loser");
  String languages = request.getParameter("languages");
  languages = replaceIfMissing(languages, "<I>None</I>");
  String languageList = makeList(languages);
  String skills = request.getParameter("skills");
  skills = replaceIfMissing(skills, "Not many, obviously.");
  ...
}
```

Filtering strings for HTML-specific characters

- You cannot safely insert arbitrary strings into servlet output
  - < and > can cause problems anywhere
  - & and " can cause problems inside of HTML attributes
- You sometimes cannot manually translate
  - The string is derived from a program excerpt or another source where it is already in some standard format
  - The string is derived from HTML form data
- Failing to filter special characters from form data makes you vulnerable to *cross-site scripting attack*

https://www.3pillarglobal.com/insights/security-vulnerabilities-java-based-web-applications
https://www.netsparker.com/blog/web-security/sql-injection-vulnerability/

Filtering code (ServletUtilities.java)

```
public static String filter(String input) {
   StringBuffer filtered = new
 StringBuffer(input.length());
   char c;
   for(int i=0; i<input.length(); i++) {
     c = input.charAt(i);
     if (c == '<') {
       filtered.append("&lt;");
     } else if (c == '>') {
       filtered.append("&gt;");
     } else if (c == '"') {
       filtered.append("&quot;");
     } else if (c == '&') {
       filtered.append("&amp;");
     } else {
       filtered.append(c);
     }
   }
   return(filtered.toString());
}
```

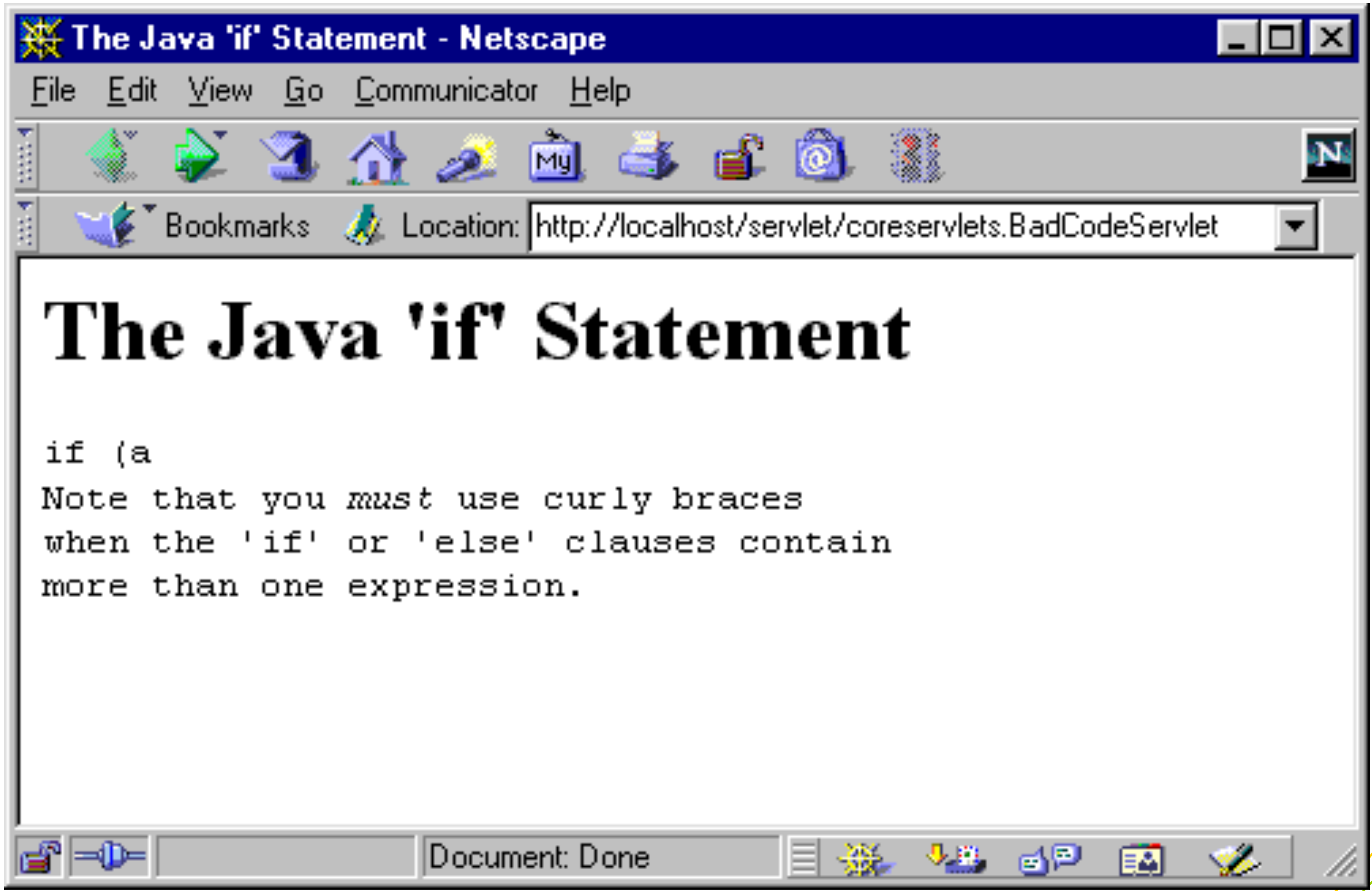
Servlet that fails to filter

```
public class BadCodeServlet extends
   HttpServlet {
  private String codeFragment =
    "if (a<b) {\n" +
    "  doThis();\n" +
    "} else {\n" +
    "  doThat();\n" +
    "}\n";

  public String getCodeFragment() {
    return(codeFragment);
  }
```

Servlet that fails to filter (continued)

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  String title = "The Java 'if' Statement";
  out.println(ServletUtilities.headWithTitle(title) +
              "<BODY>\n" +
              "<H1>" + title + "</H1>\n" +
              "<PRE>\n" +
              getCodeFragment() +
              "</PRE>\n" +
              "Note that you <I>must</I> use curly braces\n"
 +
              "when the 'if' or 'else' clauses contain\n" +
              "more than one expression.\n" +
              "</BODY></HTML>");
 }
```

Servlet that fails to filter (Result)

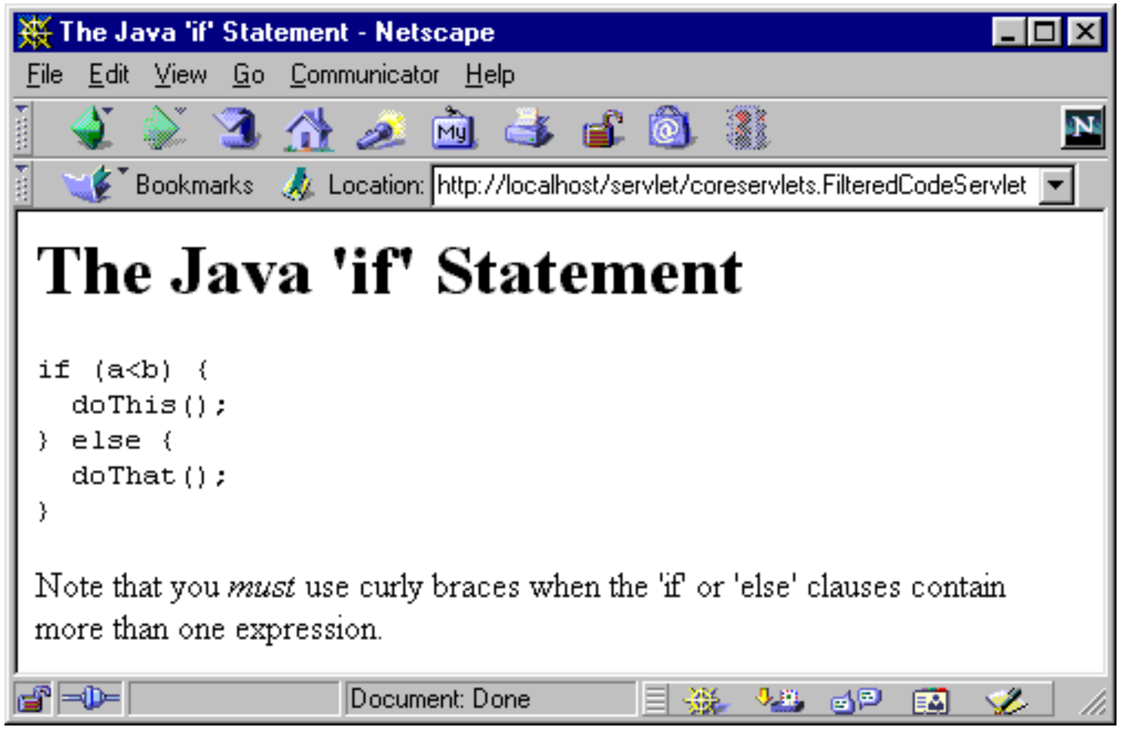

The Java 'if' Statement - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/servlet/coreservlets.BadCodeServlet
Document: Done

The Java 'if' Statement

```
if (a
Note that you must use curly braces
when the 'if' or 'else' clauses contain
more than one expression.
```

Servlet that properly filters

```
public class FilteredCodeServlet extends BadCodeServlet {
  public String getCodeFragment() {

   return(ServletUtilities.filter(super.getCodeFragment()));
  }
}
```



The Java 'if' Statement - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/servlet/coreservlets.FilteredCodeServlet
Document: Done

The Java 'if' Statement

```
if (a<b) {
  doThis();
} else {
  doThat();
}
```

Note that you *must* use curly braces when the 'if' or 'else' clauses contain more than one expression.

Summary

- Query data comes from HTML forms as URL- encoded name/value pairs
- Servlets read data by calling request.getParameter("name")
  - Results in value as entered into form, not as sent over network. I.e. *not* URL-encoded.
- Always check for missing or malformed data
  - Missing: null or empty string
  - Special case: query data that contains special HTML characters
    - Need to be filtered if query data will be placed into resultant HTML page

Handling the Client Request: HTTP Request Headers

Agenda

- Idea of HTTP request headers
- Reading request headers from servlets
- Example: printing all headers
- Common HTTP 1.1 request headers
- Example: compressing Web pages
- Example: password-protecting Web pages

Handling the Client Request: HTTP Request Headers

- Example HTTP 1.1 Request

```
GET /search?keywords=servlets+jsp HTTP/1.1
Accept: image/gif, image/jpg, */*
Accept-Encoding: gzip
Connection: Keep-Alive
Cookie: userID=id456578
Host: www.somebookstore.com
Referer:
  http://www.somebookstore.com/findbooks.html
User-Agent: Mozilla/4.7 [en] (Win98; U)
```

- It shouldn't take a rocket scientist to realize that you need to understand HTTP to be effective with servlets or JSP

Reading request headers: methods in HttpServletRequest

- General
  - getHeader
  - getHeaders (2.2 only, https://www.oracle.com/technetwork/articles/javase/servletapi-137835.html)
  - getHeaderNames
- Specialized
  - getCookies
  - getAuthType and getRemoteUser
  - getContentLength
  - getContentType
  - getDateHeader
  - getIntHeader
- Related info
  - getMethod, getRequestURI, getProtocol

Checking for missing headers

- HTTP 1.0
  - *All* request headers are optional
- HTTP 1.1
  - Only `Host` is required
- Conclusion
  - *Always* check that request.getHeader is non-null before trying to use it

```
String val = request.getHeader("some
  name");
if (val != null) {
    …
}
```

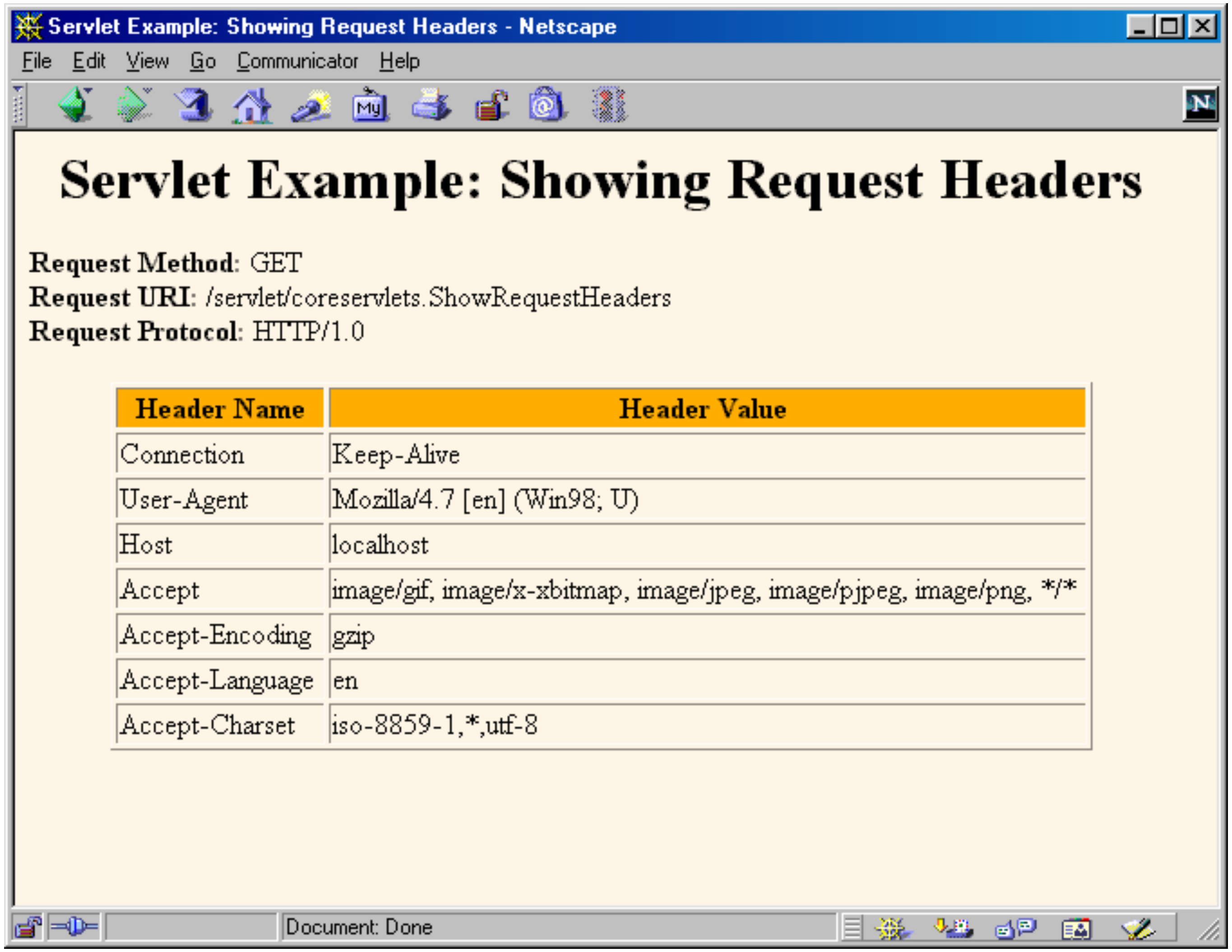
Printing all headers

```
public class ShowRequestHeaders extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Servlet Example: Showing Request Headers";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                "<B>Request Method: </B>" +
                request.getMethod() + "<BR>\n" +
                "<B>Request URI: </B>" +
                request.getRequestURI() + "<BR>\n" +
                "<B>Request Protocol: </B>" +
                request.getProtocol() + "<BR><BR>\n" +
```

Printing all headers (continued)

```
                "<TABLE BORDER=1 ALIGN=CENTER>\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "<TH>Header Name<TH>Header Value");
    Enumeration headerNames = request.getHeaderNames();
    while(headerNames.hasMoreElements()) {
      String headerName = (String)headerNames.nextElement();
      out.println("<TR><TD>" + headerName);
      out.println("    <TD>" + request.getHeader(headerName));
    }
    out.println("</TABLE>\n</BODY></HTML>");
  }

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

Printing all headers: typical netscape result

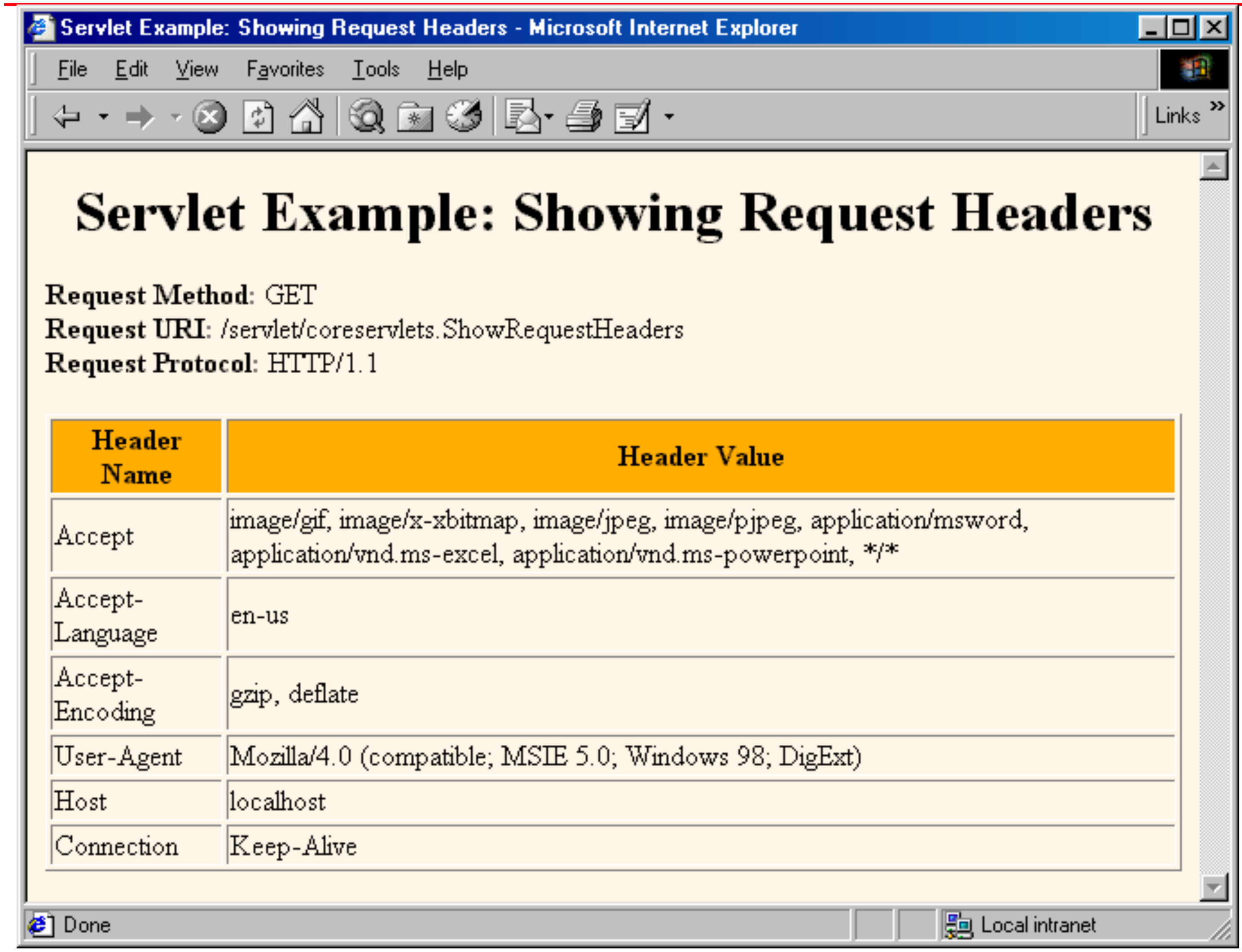

Servlet Example: Showing Request Headers - Netscape
File Edit View Go Communicator Help
Document: Done

Servlet Example: Showing Request Headers

Request Method: GET
Request URI: /servlet/coreservlets.ShowRequestHeaders
Request Protocol: HTTP/1.0

Header Name	Header Value
Connection	Keep-Alive
User-Agent	Mozilla/4.7 [en] (Win98; U)
Host	localhost
Accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding	gzip
Accept-Language	en
Accept-Charset	iso-8859-1,*,utf-8

Printing all headers: typical internet explorer result


Servlet Example: Showing Request Headers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
Local intranet

Servlet Example: Showing Request Headers

Request Method: GET
Request URI: /servlet/coreservlets.ShowRequestHeaders
Request Protocol: HTTP/1.1

Header Name	Header Value
Accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword, application/vnd.ms-excel, application/vnd.ms-powerpoint, */*
Accept-Language	en-us
Accept-Encoding	gzip, deflate
User-Agent	Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
Host	localhost
Connection	Keep-Alive

SecretServlet (registered name of ProtectedPage servlet)

```
public class ProtectedPage extends HttpServlet {
  private Properties passwords;
  private String passwordFile;

  public void init(ServletConfig config)
      throws ServletException {
    super.init(config);
    try {
      passwordFile =
        config.getInitParameter("passwordFile");
      passwords = new Properties();
      passwords.load(new FileInputStream(passwordFile));
    } catch(IOException ioe) {}
  }
```

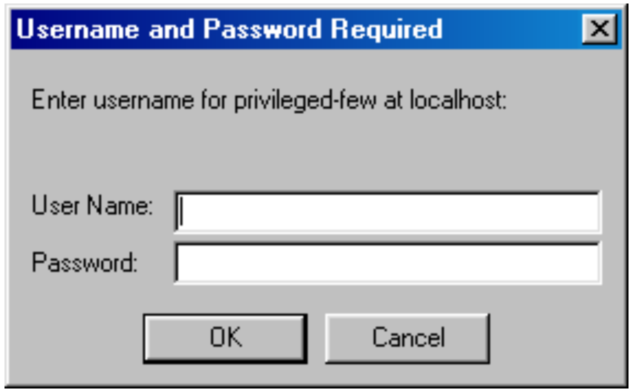
SecretServlet (continued)

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  String authorization =
    request.getHeader("Authorization");
  if (authorization == null) {
    askForPassword(response);
  } else {
    String userInfo =
      authorization.substring(6).trim();
    BASE64Decoder decoder = new BASE64Decoder();
    String nameAndPassword =
      new String(decoder.decodeBuffer(userInfo));
    // Check name and password
```

SecretServlet (continued)

```
private void askForPassword
                              (HttpServletResponse
response) {
  // SC_UNAUTHORIZED is 401

response.setStatus(response.SC_UNAUTHORIZED)
;
  response.setHeader("WWW-Authenticate",
                        "BASIC
realm=\"privileged-few\"");
}
```

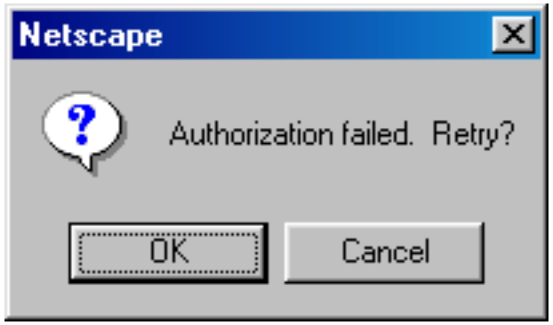
SecretServlet in action


Username and Password Required

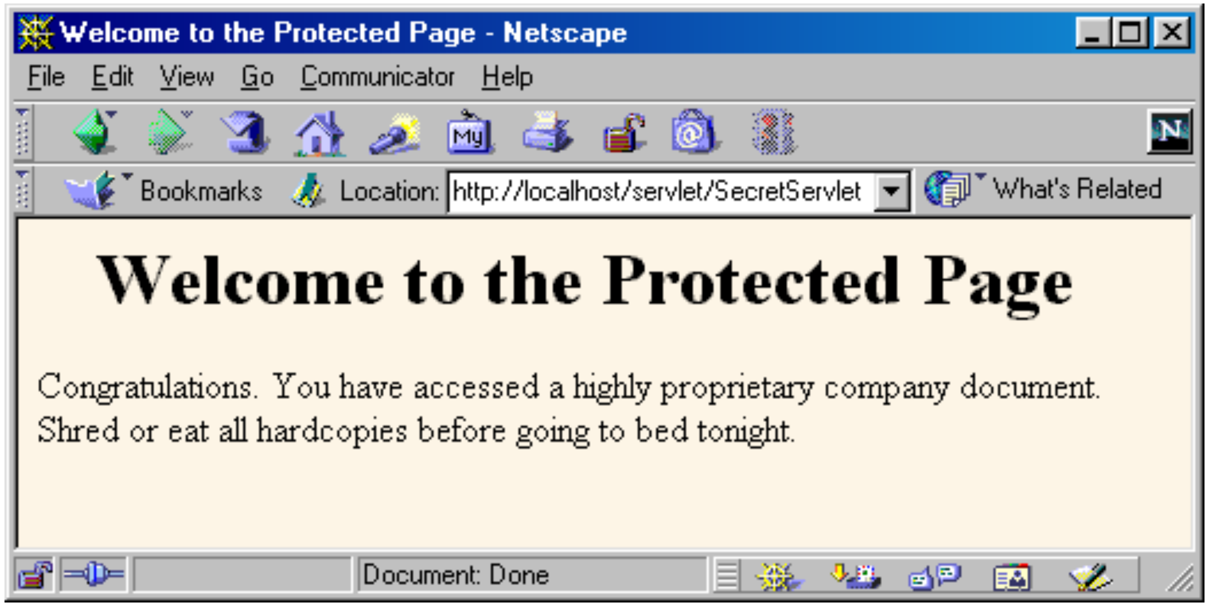
Enter username for privileged-few at localhost:

User Name:

Password:


Netscape

Authorization failed. Retry?


Welcome to the Protected Page - Netscape

http://localhost/servlet/SecretServlet

Welcome to the Protected Page

Congratulations. You have accessed a highly proprietary company document. Shred or eat all hardcopies before going to bed tonight.

Document: Done

Summary

- Many servlet tasks can *only* be accomplished by making use of HTTP headers coming from the browser
- Use request.getHeader for arbitrary header
  - Remember to check for `null`
- Cookies, authorization info, content length, and content type have shortcut methods
- Most important headers you read directly
  - Accept
  - Accept-Encoding
  - Connection
  - Referer
  - User-Agent

Generating the HTTP Response

Agenda

- Idea of HTTP status codes
- Setting status codes from servlets
- Common HTTP 1.1 status codes
- A common front end to various Web search engines
- Idea of HTTP response headers
- Setting response headers from servlets
- Common HTTP 1.1 response headers
- Persistent servlet state and auto-reloading pages

Generating the server response: HTTP status codes

- Example HTTP 1.1 Response

```
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE ...>
<HTML>
...
</HTML>
```

- Changing the status code lets you perform a number of tasks not otherwise possible
  - Forward client to another page
  - Indicate a missing resource
  - Instruct browser to use cached copy
- Set status *before* sending document

Setting status codes

- response.setStatus(int statusCode)
  - Use a constant for the code, not an explicit int. Constants are in HttpServletResponse
  - Names derived from standard message. E.g., SC_OK, SC_NOT_FOUND, etc.
- response.sendError(int code,
  String message)
  - Wraps message inside small HTML document
- response.sendRedirect(String url)
  - Relative URLs permitted in 2.2 and later
  - Sets Location header also

Common HTTP 1.1 status codes

- 200 (OK)
  - Everything is fine; document follows.
  - Default for servlets.
- 204 (No Content)
  - Browser should keep displaying previous document.
- 301 (Moved Permanently)
  - Requested document permanently moved elsewhere (indicated in Location header).
  - Browsers go to new location automatically.

Common HTTP 1.1 status codes

- 302 (Found)
  - Requested document temporarily moved elsewhere (indicated in Location header).
  - Browsers go to new location automatically.
  - Servlets should use sendRedirect, not setStatus, when setting this header. See example.
- 401 (Unauthorized)
  - Browser tried to access password-protected page without proper Authorization header. See example in book.
- 404 (Not Found)
  - No such page. Servlets should use sendError to set this.
  - Problem: Internet Explorer 5.0.
  - Fun and games: http://www.plinko.net/404/

A front end to various search engines: code

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  String searchString =
    request.getParameter("searchString");
  if ((searchString == null) ||
      (searchString.length() == 0)) {
    reportProblem(response, "Missing search string.");
    return;
  }
  searchString = URLEncoder.encode(searchString);
  String numResults =
    request.getParameter("numResults");
  ...
  String searchEngine =
    request.getParameter("searchEngine");
```

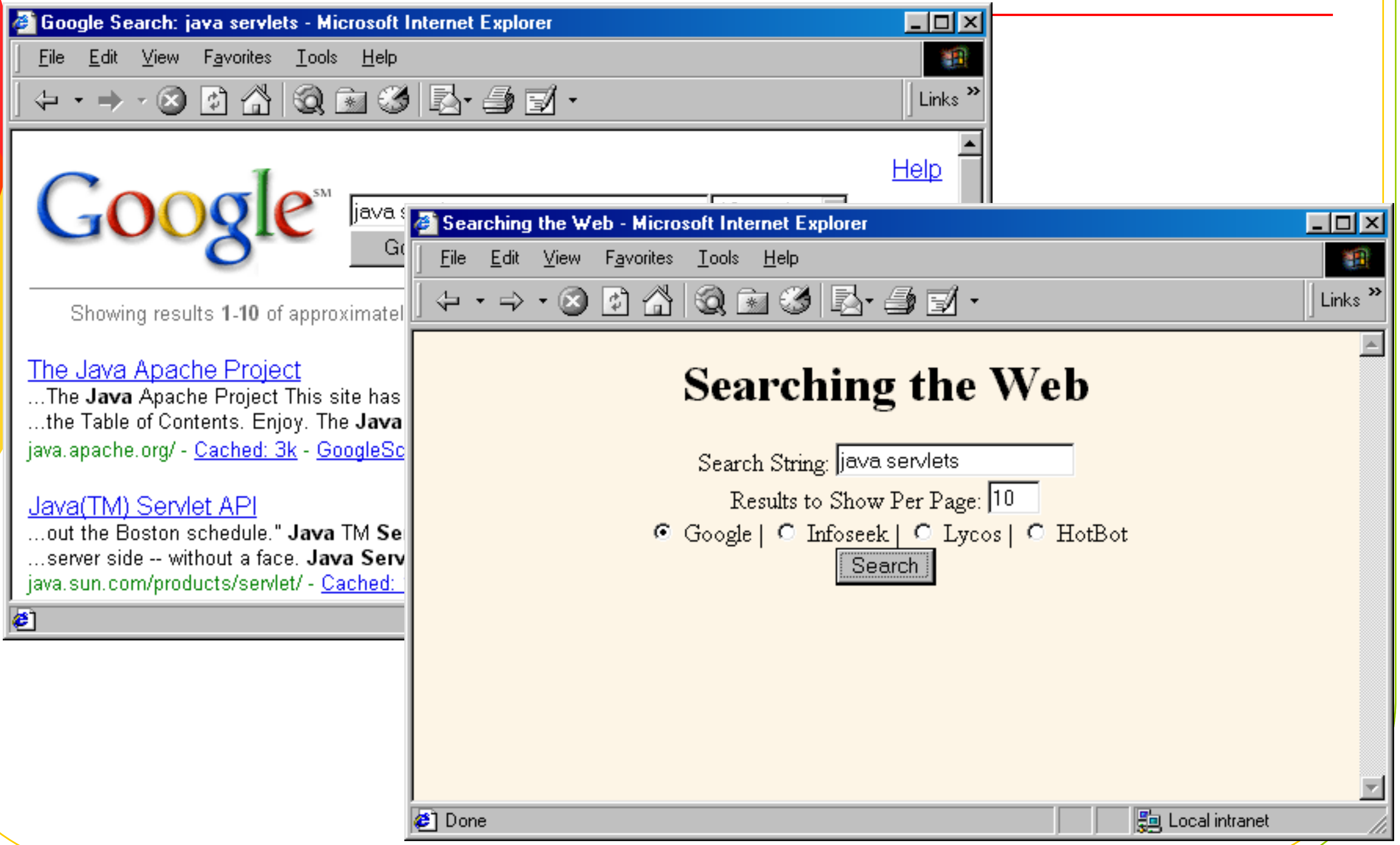
A front end to various search engines: code

```
SearchSpec[] commonSpecs =
  SearchSpec.getCommonSpecs();
for(int i=0; i<commonSpecs.length; i++) {
  SearchSpec searchSpec = commonSpecs[i];
  if (searchSpec.getName().equals(searchEngine)) {
    String url =
      searchSpec.makeURL(searchString, numResults);
    response.sendRedirect(url);
    return;
  }
}
reportProblem(response,
                    "Unrecognized search engine.");
```

A front end to various search engines: code

```
private void reportProblem(HttpServletResponse
 response,
                                           String message)
      throws IOException {
   response.sendError(response.SC_NOT_FOUND,
                              message);
}
```

Front end to search engines: result of legal request



Google Search: java servlets - Microsoft Internet Explorer

Showing results **1-10** of approximatel

The Java Apache Project
...The **Java** Apache Project This site has
...the Table of Contents. Enjoy. The **Java**
java.apache.org/ - Cached: 3k - GoogleSc

Java(TM) Servlet API
...out the Boston schedule." **Java** TM **Se**
...server side -- without a face. **Java Serv**
java.sun.com/products/servlet/ - Cached:

Searching the Web - Microsoft Internet Explorer

Searching the Web

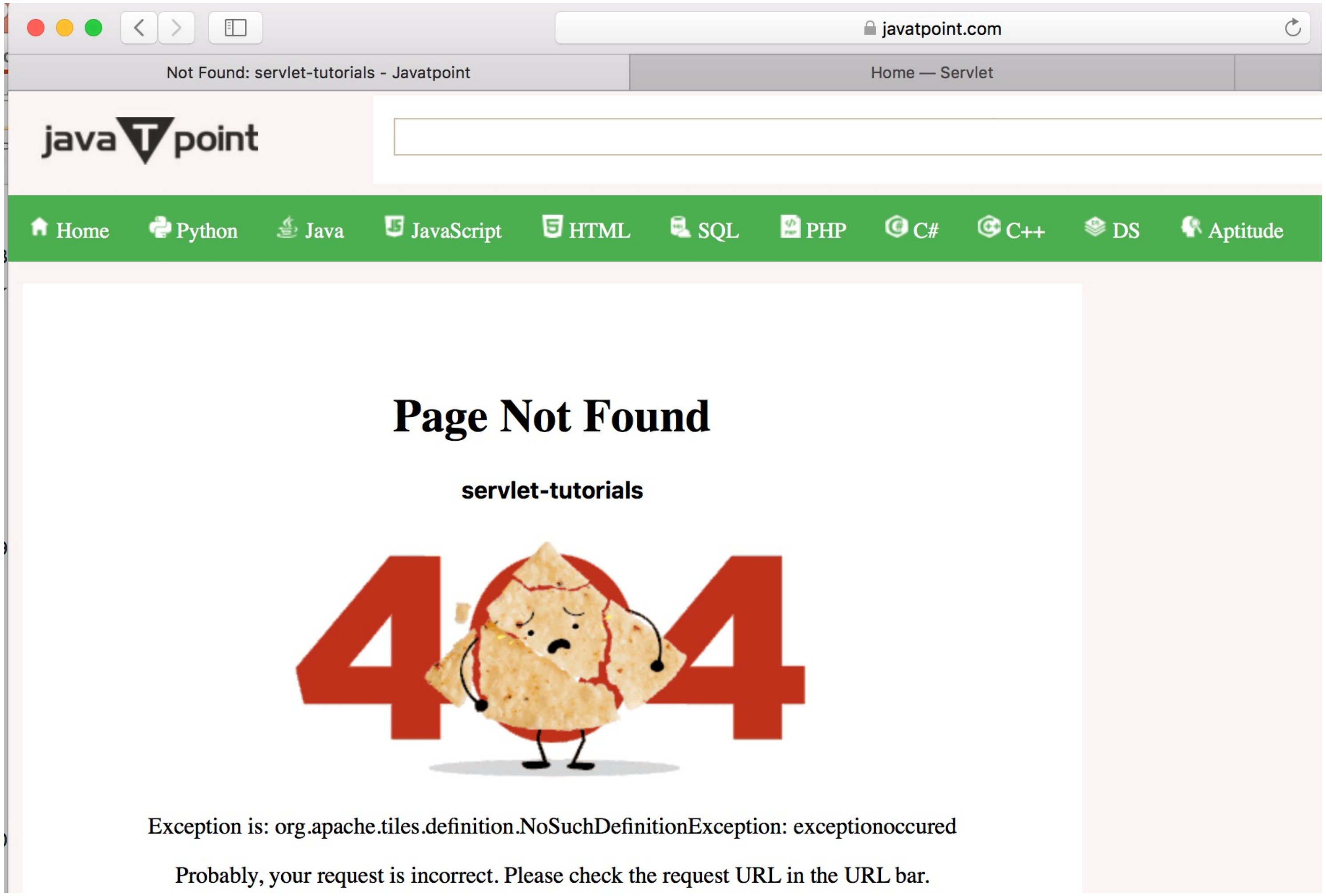
Search String: java servlets

Results to Show Per Page: 10

[✓ Google] | Infoseek | Lycos | HotBot

Search

Front end to search engines: result of illegal request



Page Not Found

servlet-tutorials

Exception is: org.apache.tiles.definition.NoSuchDefinitionException: exceptionoccured

Probably, your request is incorrect. Please check the request URL in the URL bar.

Generating the server response: HTTP response headers

- Purposes
  - Give forwarding location
  - Specify cookies
  - Supply the page modification date
  - Instruct the browser to reload the page after a designated interval
  - Give the document size so that persistent HTTP connections can be used
  - Designate the type of document being generated
  - Etc.

Setting arbitrary response headers

- public void setHeader(String headerName, String headerValue)
  - Sets an arbitrary header.
- public void setDateHeader(String name, long millisecs)
  - Converts milliseconds since 1970 to a date string in GMT format.
- public void setIntHeader(String name, int headerValue)
  - Prevents need to convert int to String before calling setHeader.
- addHeader, addDateHeader, addIntHeader
  - Adds new occurrence of header instead of replacing. Servlets 2.2/2.3 only.

Setting common response headers

- setContentType
  - Sets the Content-Type header.
    Servlets almost always use this.
    See table of common MIME types.
- setContentLength
  - Sets the Content-Length header.
    Used for persistent HTTP connections.
    See Connection request header.
- addCookie
  - Adds a value to the Set-Cookie header.
    See separate section on cookies.
- sendRedirect
  - Sets the Location header (plus changes status code).

Common MIME types

Type	**Meaning**
application/msword	Microsoft Word document
application/octet-stream	Unrecognized or binary data
application/pdf	Acrobat (.pdf) file
application/postscript	PostScript file
application/vnd.ms-excel	Excel spreadsheet
application/vnd.ms-powerpoint	Powerpoint presentation
application/x-gzip	Gzip archive
application/x-java-archive	JAR file
application/x-java-vm	Java bytecode (.class) file
application/zip	Zip archive
audio/basic	Sound file in .au or .snd format
audio/x-aiff	AIFF sound file
audio/x-wav	Microsoft Windows sound file
audio/midi	MIDI sound file
text/css	HTML cascading style sheet
text/html	HTML document
text/plain	Plain text
text/xml	XML document
image/gif	GIF image
image/jpeg	JPEG image
image/png	PNG image
image/tiff	TIFF image
video/mpeg	MPEG video clip
video/quicktime	QuickTime video clip

Common HTTP 1.1 response headers

- Cache-Control (1.1) and Pragma (1.0)
  - A no-cache value prevents browsers from caching page. Send both headers or check HTTP version.
- Content-Encoding
  - The way document is encoded. Browser reverses this encoding before handling document. See compression example earlier.
- Content-Length
  - The number of bytes in the response.
  - See setContentLength on previous slide.
  - Use ByteArrayOutputStream to buffer document before sending it, so that you can determine size. See discussion of the Connection request header and detailed example in book.

Common HTTP 1.1 response headers

- Content-Type
  - The MIME type of the document being returned.
  - Use setContentType to set this header.
- Expires
  - The time at which document should be considered out-of-date and thus should no longer be cached.
  - Use setDateHeader to set this header.
- Last-Modified
  - The time document was last changed.
  - Don’t set this header explicitly; provide a getLastModified method instead.
    See example in CSAJSP Chapter 2.

Common HTTP 1.1 response headers

- Location
  - The URL to which browser should reconnect.
  - Use sendRedirect instead of setting this directly.
- Refresh
  - The number of seconds until browser should reload page. Can also include URL to connect to.
    See following example.
- Set-Cookie
  - The cookies that browser should remember. Don’t set this header directly; use addCookie instead. See next section.
- WWW-Authenticate
  - The authorization type and realm needed in Authorization header. See example in *CSAJSP* Section 4.5.

Persistent servlet state and auto-reloading pages

- Idea: generate list of large (e.g., 150-digit) prime numbers
  - Show partial results until completed
  - Let new clients make use of results from others
- Demonstrates use of the Refresh header.
- Shows how easy it is for servlets to maintain state between requests.
  - Very difficult in traditional CGI.
- Also illustrates that servlets can handle multiple simultaneous connections
  - Each request is in a separate thread.

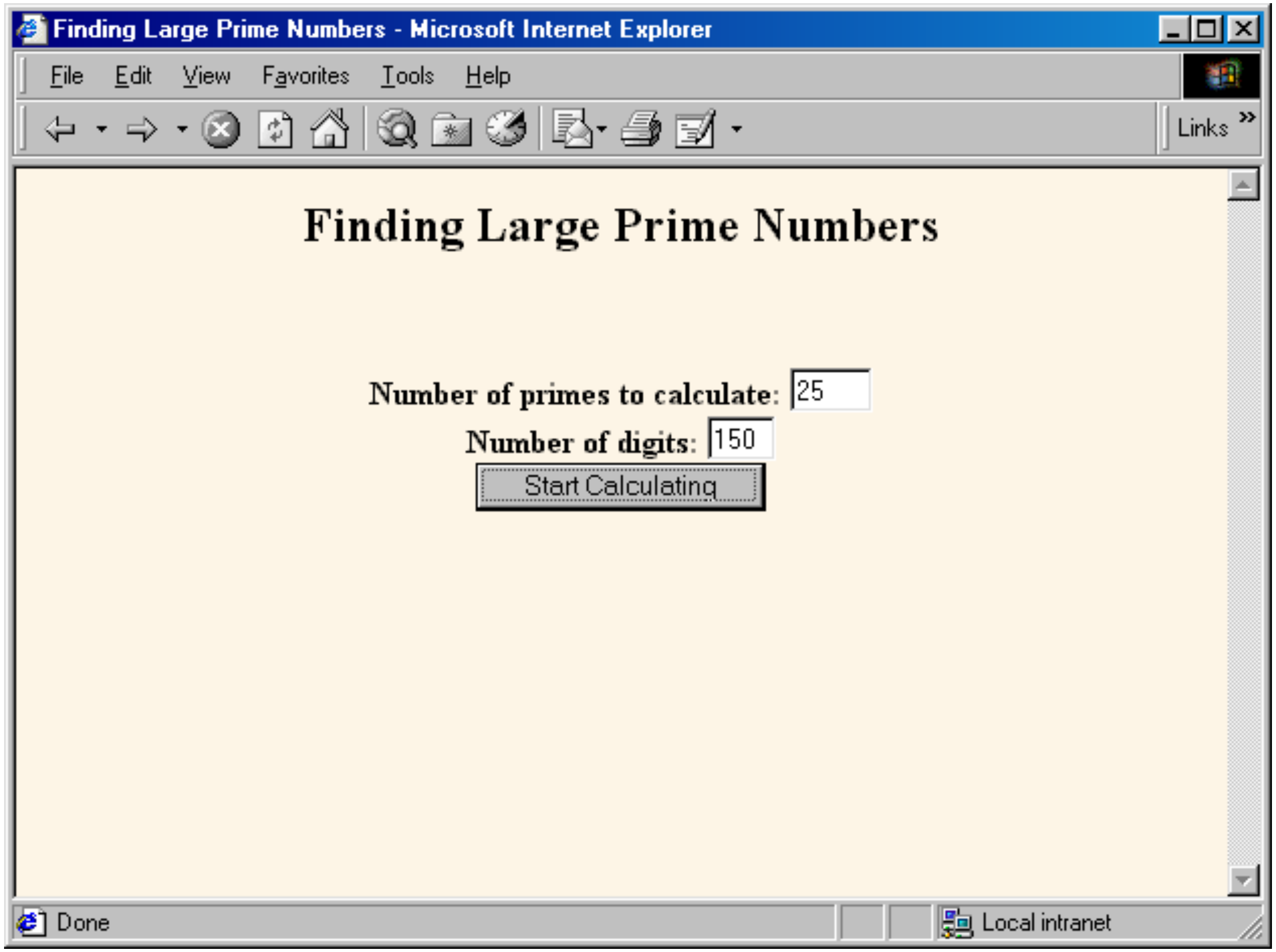
Generating prime numbers: source code

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  int numPrimes =
    ServletUtilities.getIntParameter(request,
                                     "numPrimes", 50);
  int numDigits =
    ServletUtilities.getIntParameter(request,
                                     "numDigits", 120);
  // findPrimeList is synchronized
  PrimeList primeList =
    findPrimeList(primeListVector, numPrimes,
 numDigits);
  if (primeList == null) {
    primeList = new PrimeList(numPrimes, numDigits,
 true);
```

Generating prime numbers: source code

```
  synchronized(primeListVector) {
    if (primeListVector.size() >= maxPrimeLists)
      primeListVector.removeElementAt(0);
    primeListVector.addElement(primeList);
  }
}
Vector currentPrimes = primeList.getPrimes();
int numCurrentPrimes = currentPrimes.size();
int numPrimesRemaining = (numPrimes - numCurrentPrimes);
boolean isLastResult = (numPrimesRemaining == 0);
if (!isLastResult) {
  response.setHeader("Refresh", "5");
}
response.setContentType("text/html");
PrintWriter out = response.getWriter();
// Show List of Primes found ...
```

Prime number servlet: front end



Finding Large Prime Numbers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
Local intranet

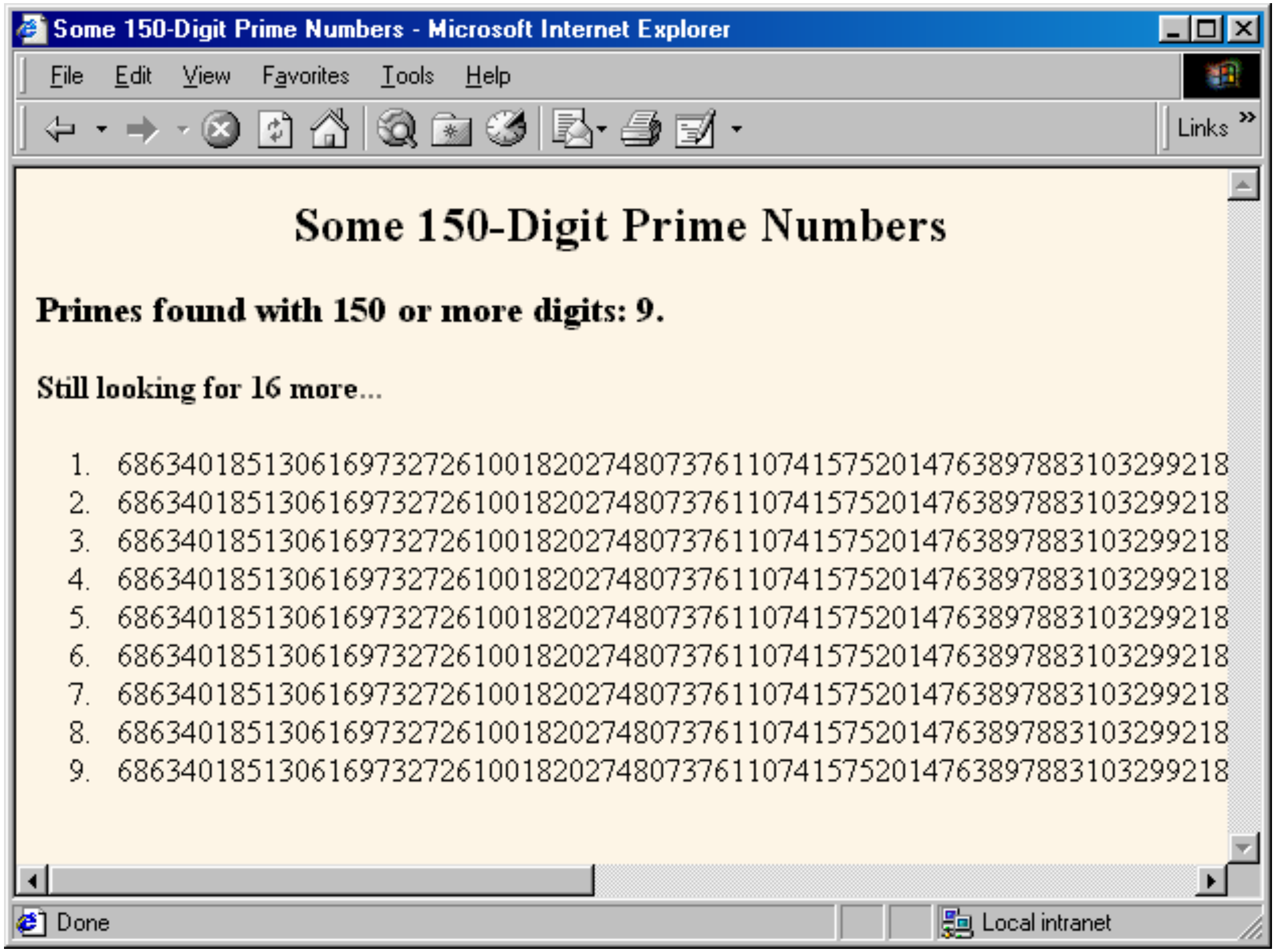
Finding Large Prime Numbers

Number of primes to calculate: 25

Number of digits: 150

Start Calculating

Prime number servlet: initial result


Some 150-Digit Prime Numbers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
Local intranet

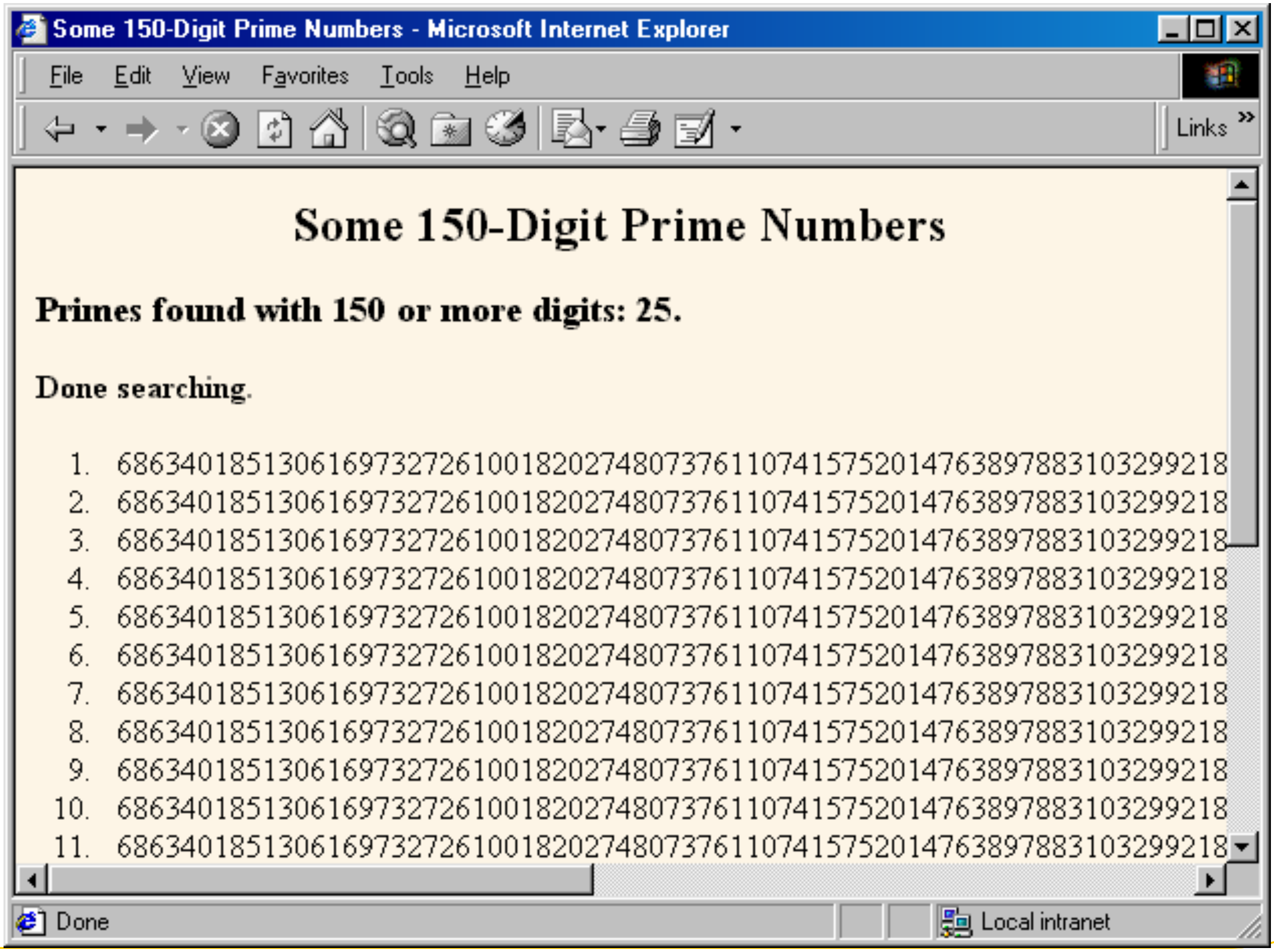
Some 150-Digit Prime Numbers

Primes found with 150 or more digits: 9.

Still looking for 16 more...

1. 686340185130616973272610018202748073761107415752014763897883103299218
2. 686340185130616973272610018202748073761107415752014763897883103299218
3. 686340185130616973272610018202748073761107415752014763897883103299218
4. 686340185130616973272610018202748073761107415752014763897883103299218
5. 686340185130616973272610018202748073761107415752014763897883103299218
6. 686340185130616973272610018202748073761107415752014763897883103299218
7. 686340185130616973272610018202748073761107415752014763897883103299218
8. 686340185130616973272610018202748073761107415752014763897883103299218
9. 686340185130616973272610018202748073761107415752014763897883103299218

Prime number servlet: final result


Some 150-Digit Prime Numbers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
Local intranet

Some 150-Digit Prime Numbers

Primes found with 150 or more digits: 25.

Done searching.

1. 68634018513061697327261001820274807376110741575201476389788310329218
2. 68634018513061697327261001820274807376110741575201476389788310329218
3. 68634018513061697327261001820274807376110741575201476389788310329218
4. 68634018513061697327261001820274807376110741575201476389788310329218
5. 68634018513061697327261001820274807376110741575201476389788310329218
6. 68634018513061697327261001820274807376110741575201476389788310329218
7. 68634018513061697327261001820274807376110741575201476389788310329218
8. 68634018513061697327261001820274807376110741575201476389788310329218
9. 68634018513061697327261001820274807376110741575201476389788310329218
10. 68634018513061697327261001820274807376110741575201476389788310329218
11. 68634018513061697327261001820274807376110741575201476389788310329218

Summary

- Many servlet tasks can *only* be accomplished through use of HTTP status codes and headers sent to the browser
- Two parts of the response
  - Status line
    - In general, set via response.setStatus
    - In special cases, set via response.sendRedirect and response.sendError
  - Response headers
    - In general, set via response.setHeader
    - In special cases, set via response.setContentType, response.setContentLength, response.addCookie, and response.sendRedirect

Summary

- Most important status codes
  - 200 (default)
  - 302 (forwarding; set via sendRedirect)
  - 401 (password needed)
  - 404 (not found; set via sendError)
- Most important headers you set directly
  - Cache-Control and Pragma
  - Content-Encoding
  - Content-Length
  - Expires
  - Refresh
  - WWW-Authenticate