# Java Servlets

**Part I Server and Servlet Basics**

**Part II Session Tracking and Servlet Collaboration**

**Design Patterns along the way**
**Erich Gamma, Richard Helm, Ralph Johnson**
**John Vlissides (Gang of Four)**

## Part I : Server and Servlet Basics

- NetworkServer.java and EchoServer.java
- PostForm.html
- GetForm.html
- More HTML form examples

# NetworkServer.java

```
// NetworkServer.java        Adapted from "Core Servlets
// and Java Server Pages"
// by Marty Hall

import java.net.*;
import java.io.*;

public class NetworkServer {

    private int port;
    private int maxConnections;
```

No web server.
Just this code.

```
protected void setPort(int port) { this.port = port; }

public int getPort() { return port; }

protected void setMaxConnections(int max) {
                maxConnections = max;
}

public int getMaxConnections() { return maxConnections; }

public NetworkServer(int port, int maxConnections) {
  setPort(port);
  setMaxConnections(maxConnections);
}
```

```
// Wait for a connections until maxConnections.
// On each connection call handleConnection() passing
// the socket. If maxConnections == 0 loop forever

  public void listen() {
        int i = 0;
        try {
             ServerSocket listener = new ServerSocket(port);
             Socket server ;
             while((i++ < maxConnections) || (maxConnections == 0)) {
                  server = listener.accept();   // wait for connection
                  handleConnection(server);
             }
        } catch (IOException ioe) {
              System.out.println("IOException : " + ioe);
              ioe.printStackTrace();
        }
   }
```

```
// Open readers and writers to socket.
// Display client's host name to console.
// Read a line from the client and display it on the console.
// Send "Generic network server" to the client.
// Override this method.

protected void handleConnection(Socket server)
                                        throws IOException {

     BufferedReader in = new BufferedReader(
                    new InputStreamReader(
                        server.getInputStream() ));

     PrintWriter out = new PrintWriter(
                         server.getOutputStream(),true);
```

InputStream for reading bytes

Flush buffer on println

Readers and Writers to work with characters

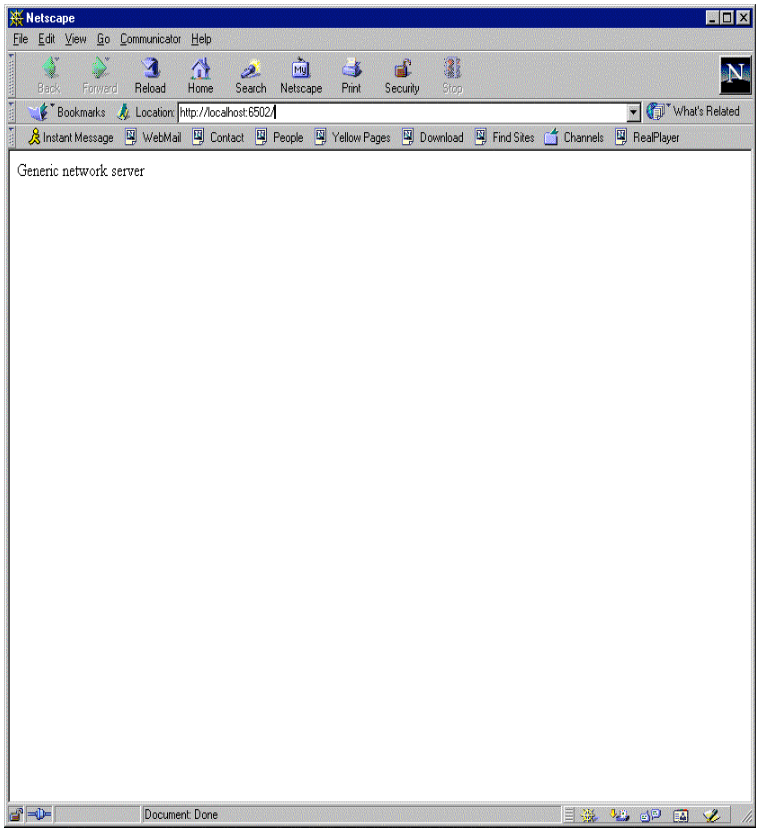
```
System.out.println("Generic network server: got connection from "+
                    server.getInetAddress().getHostName() + "\n" +
                    "with first line '" + in.readLine() + "'");
     out.println("Generic network server");
     server.close();
   }

   public static void main(String args[]) {
      NetworkServer test = new NetworkServer(6502, 5);
      test.listen();
   }
}
```

To server's console.

To client.

# Compile, Run and Visit


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:6502/
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
Document: Done

Generic network server

Client

Server

```
C:\McCarthy\www\46-928\examples\networking>java NetworkServer
Generic network server: got connection from localhost
with first line 'GET / HTTP/1.0'
```

## Decorator Design Pattern

In the code above, the InputStream object is wrapped or decorated with two other objects.

From the Gang of Four:

“Attach additional responsibilities to
an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.”

The wrapping object has the same interface as the object it wraps. It adds functionality before or after the call to the wrapped object.

# EchoServer.java

/* From Core Servlets, Marty Hall

An HTTP **Request** header example   Notes

GET /path/file.html  HTTP/1.0
Accept:  text/html
Accept:  audio/x
User-agent:  MacWeb
Request terminated by two returns

HTTP defines dozens of possible headers.

The whitespace is required.
Accept header fields tell the server MIME types (Multipurpose Internet Mail Extension) that are handled by the browser.

Still no web server

# EchoServer.java

An HTTP **Response** header example

```
HTTP 1.0 200 OK              ←  Response code
Server: NCSA/1.4.2
MIME-version: 1.0
Content-type: text/html      ←  MIME type
Content-length: 107
                             ←  Blank line
<html>
:                            ←  The client must interpret
:                               this MIME encoded data.
</html>
```

## HTTP General form

```
<method> <resource identifier> <HTTP Version> <crlf>
[<Header> : <value>] <crlf>

:  :   :
[<Header> : <value>] <crlf>
   a blank line
[entity body]
```

The resource identifier field specifies the name of the target resource; it's the URL stripped of the protocol and the server domain name. When using the GET method, this field will also contain a series of name=value pairs separated by ‘&’. When using a POST method, the entity body contains these pairs.

The HTTP version identifies the protocol used by the client.
*/

```
// Adapted from Core Servlets and JavaServerPages
// by Marty Hall, chapter 16

import java.net.*;
import java.io.*;
import java.util.StringTokenizer;

public class EchoServer extends NetworkServer {

  protected int maxRequestLines = 50;    // Post data is brought in
                                         // as a single string.
  protected String serverName = "EchoServer";

  public static void main(String a[]) {

     int port = 6502;
     new EchoServer(port,0);    // loop forever
  }
```

```
public EchoServer(int port, int maxConnections) {
      super(port,maxConnections);      // call base class constructor
      listen();                        // call base class listen()
   }                                   // listen calls handleConnection()

   // Overrides base class handleConection and is called by listen()
   public void handleConnection(Socket server) throws IOException {

         // Assign readers and writers to the socket
         BufferedReader in = new BufferedReader(
                                 new InputStreamReader(
                                       server.getInputStream() ));
         PrintWriter out = new PrintWriter(server.getOutputStream(),true)
         // Announce connection to console
         System.out.println(serverName + " got connection from "+
                        server.getInetAddress().getHostName() + "\n");
```

```
String inputLines[] = new String[maxRequestLines];
int i;
for(i = 0; i < maxRequestLines; i++) {
      inputLines[i] = in.readLine();
      if(inputLines[i] == null) break; // client closed connection
      if(inputLines[i].length() == 0) {  // blank line
                                         // maybe done or maybe post
            if(usingPost(inputLines)) {
                  // readPostData reads into a single string
                  // at location i+1
                  readPostData(inputLines,i,in);
                  // i was not changed in the procedure so
                  // bump it one past the post data string
                  i = i + 2;
            }
             break;                      // we're done either way
      }
}
```

```
        printHeader(out);                          // HTTP + HTML
        for(int j = 0; j < i; j++) {
            out.println(inputLines[j]);            //Request Data
        }

        printTrailer(out);                         // Closing HTML
        server.close();
    }
```

HTTP Response headers plus HTML.

```
private void printHeader(PrintWriter out) {
        out.println(
              "HTTP/1.0 200 OK\r\n"           +
              "Server: " + serverName + "\r\n" +
              "Content-Type: text/html\r\n"    + "\r\n" +
              "<!DOCTYPE HTML PUBLIC "         +
              "\"-//W3C//DTD HTML 4.0 Transitional//EN\">\n" +
              "<HTML>\n"                       +
              "<HEAD>\n"                       +
              "   <TITLE>" + serverName + " Results</TITLE>\n" +
              "</HEAD>\n"                      +
              "\n"  + "<BODY BGCOLOR=\"#FDF5E6\">\n"   +
              "<H1 ALIGN=\"CENTER\">" + serverName +
              " Results</H1>\n" +
              "Here is your request line and request headers\n" +
              "sent by your browser:\n" +
              "<PRE>" );              // honors whitespace
   }
```

```
private void printTrailer(PrintWriter out) {      // Close HTML

        out.println("</PRE>\n"   +
                "</BODY>\n"  +
                "</HTML>\n");
   }
                                                   // Checks if post
   private boolean usingPost(String[] inputs) {
        return (inputs[0].toUpperCase().startsWith("POST"));
   }
```

```
// Read the post data as a single array of char and place it all
// in one string.

   private void readPostData (String inputs[], int i, BufferedReader in)
                     throws IOException {

         int contentLength = contentLength(inputs);
         char postData[] = new char[contentLength];
         in.read(postData, 0, contentLength);

         // All of the post data is converted to a single string
         inputs[++i] = new String(postData,0,contentLength);
   }
```

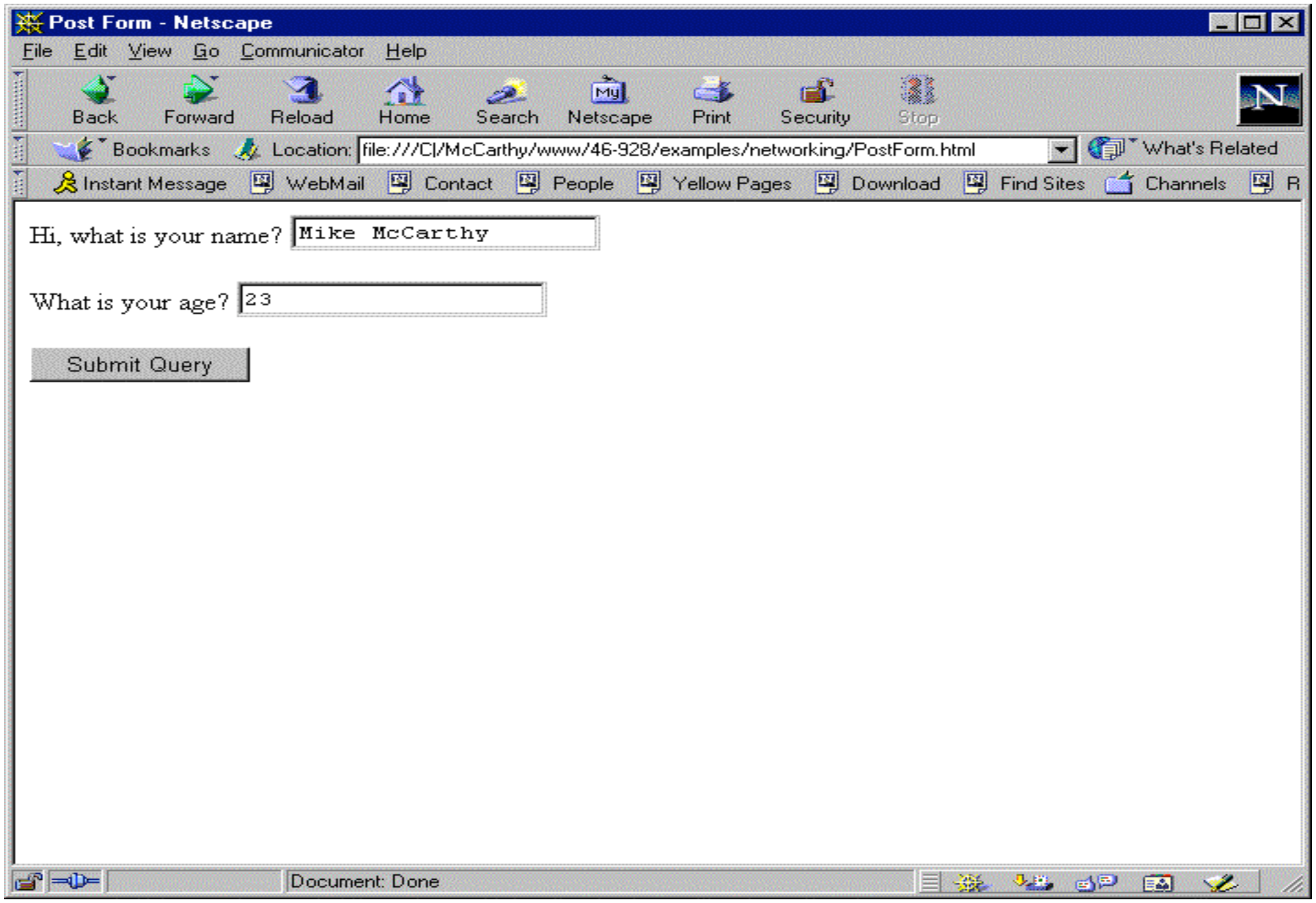
```
// The header fields may arrive in any order.
// Search for and return the CONTENT-LENGTH.
   private int contentLength(String inputs[]) {
          String input;
          for(int i = 0; i < inputs.length; i++) {
             if(inputs[i].length() == 0) break;
             input = inputs[i].toUpperCase();
             if(input.startsWith("CONTENT-LENGTH")) return (getLengtl
          }
          return (0);
   }
   // Return the integer associated with the second token.
   private int getLength(String length) {
         StringTokenizer tok = new StringTokenizer(length);
         tok.nextToken();
         return (Integer.parseInt(tok.nextToken()));
   }
}
```

# PostForm.html

```
<!-- PostForm.html -->
<html>
<head>
<title>Post Form</title>
</head>
<body>
   <form method="post" action="http://localhost:6502">
      Hi, what is your name?
      <input type="text" name = "name"> <p>
      What is your age?
      <input type="text" name = "age"> <p>
      <input type = "submit">
   </form>
</body>
</html>
```

Visit the port

# PostForm.html Browser


Post Form - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels R
Submit Query

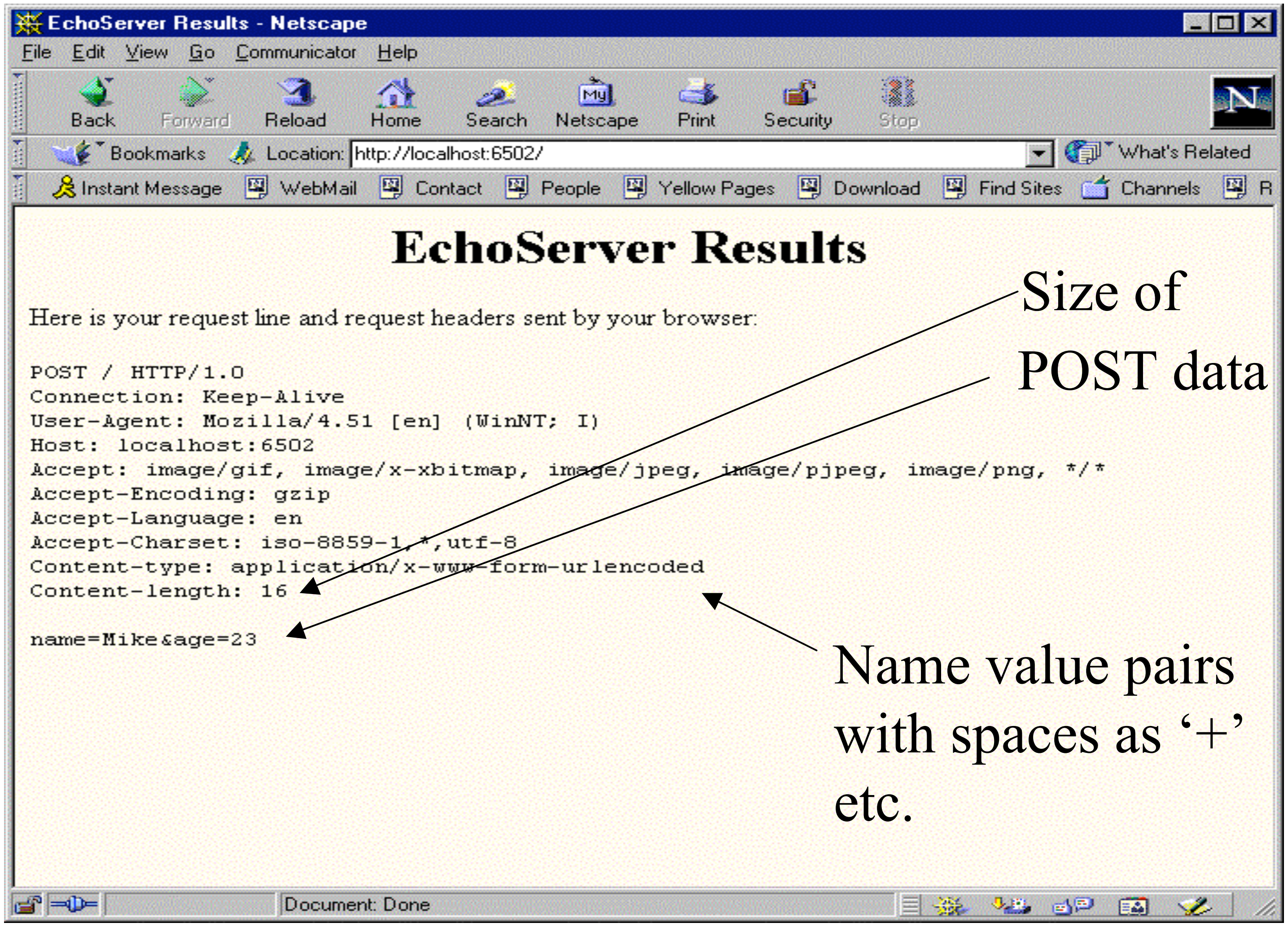
Location: file:///C|/McCarthy/www/46-928/examples/networking/PostForm.html

Hi, what is your name? Mike McCarthy

What is your age? 23

Document: Done

## EchoServer Response Using POST


EchoServer Results - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:6502/ What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

**EchoServer Results**

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

```
POST / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Host: localhost:6502
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: 16

name=Mike&age=23
```

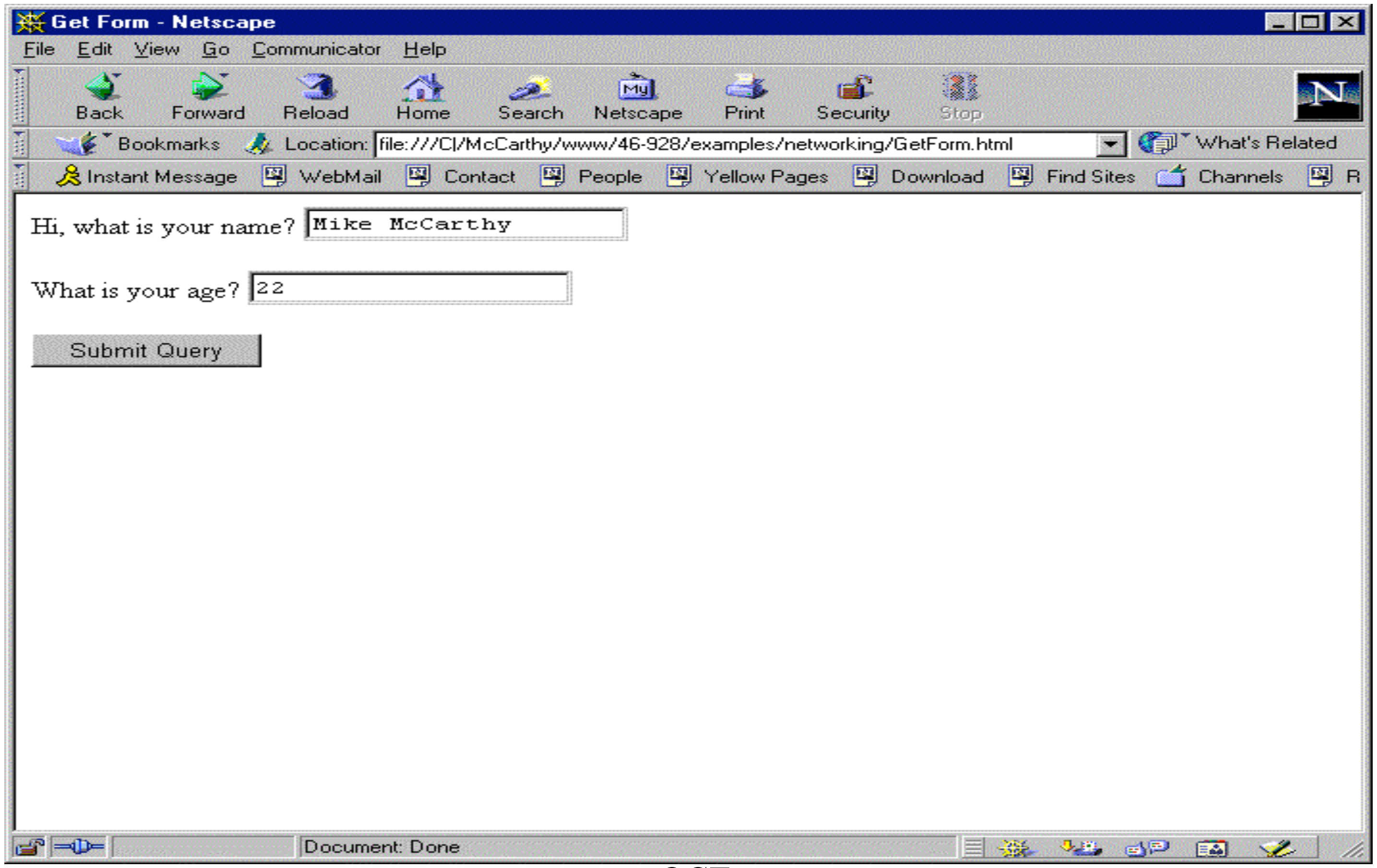
Size of POST data

Name value pairs with spaces as '+' etc.

## GetForm.html

```
<!-- GetForm.html -->
<html>
<head>
<title>Get Form</title>
</head>
<body>
   <form method="get" action="http://localhost:6502">
       Hi, what is your name?
       <input type="text" name = "name"> <p>
       What is your age?
       <input type="text" name = "age"> <p>
       <input type = "submit">
   </form>
</body>
</html>
```

# GetForm.html Browser


Get Form - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location:
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels R

file:///C|/McCarthy/www/46-928/examples/networking/GetForm.html

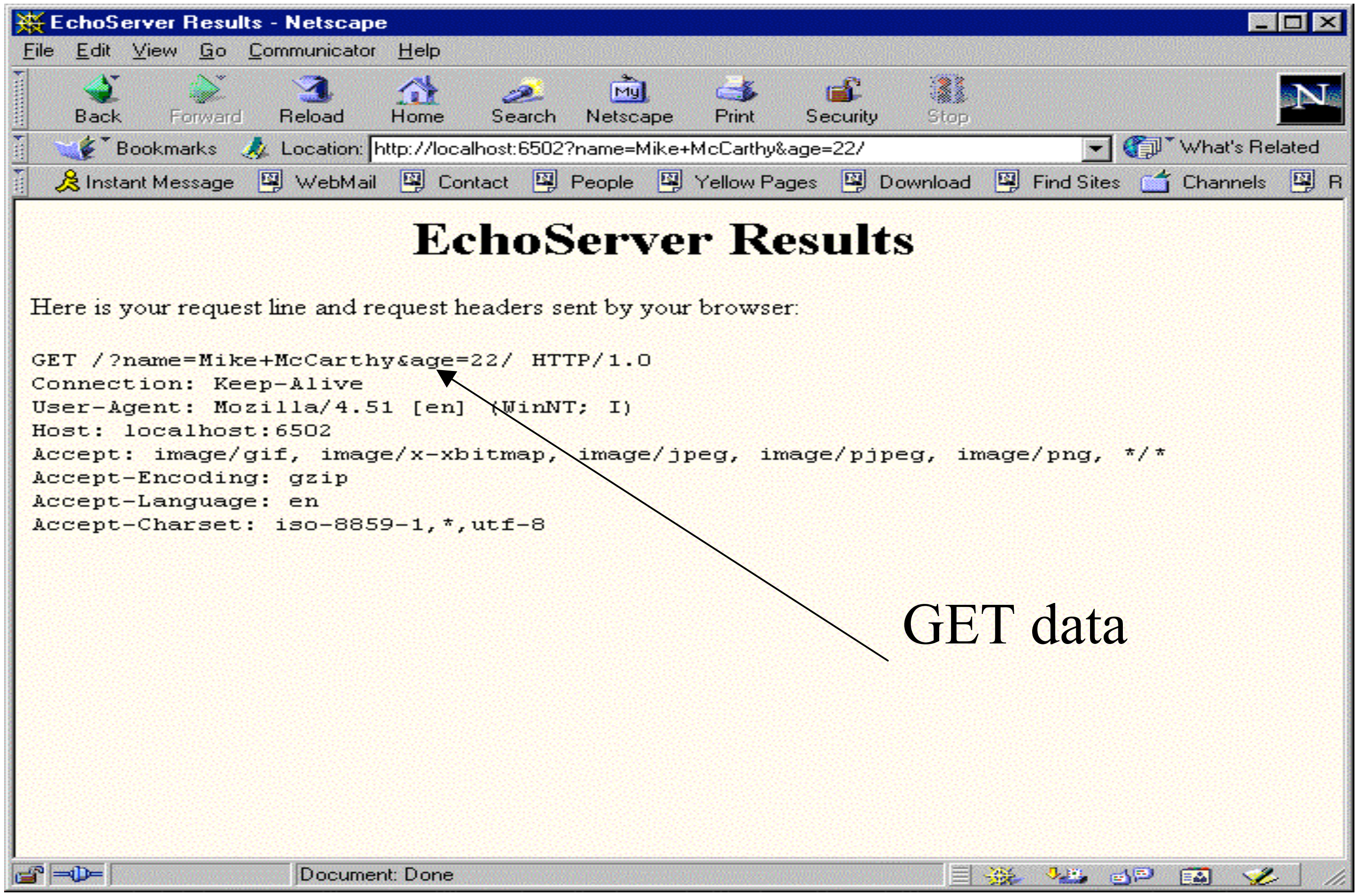
Hi, what is your name? Mike McCarthy

What is your age? 22

Submit Query

Document: Done

# EchoServer Response Using GET


EchoServer Results - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:6502?name=Mike+McCarthy&age=22/
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done
GET data

**EchoServer Results**

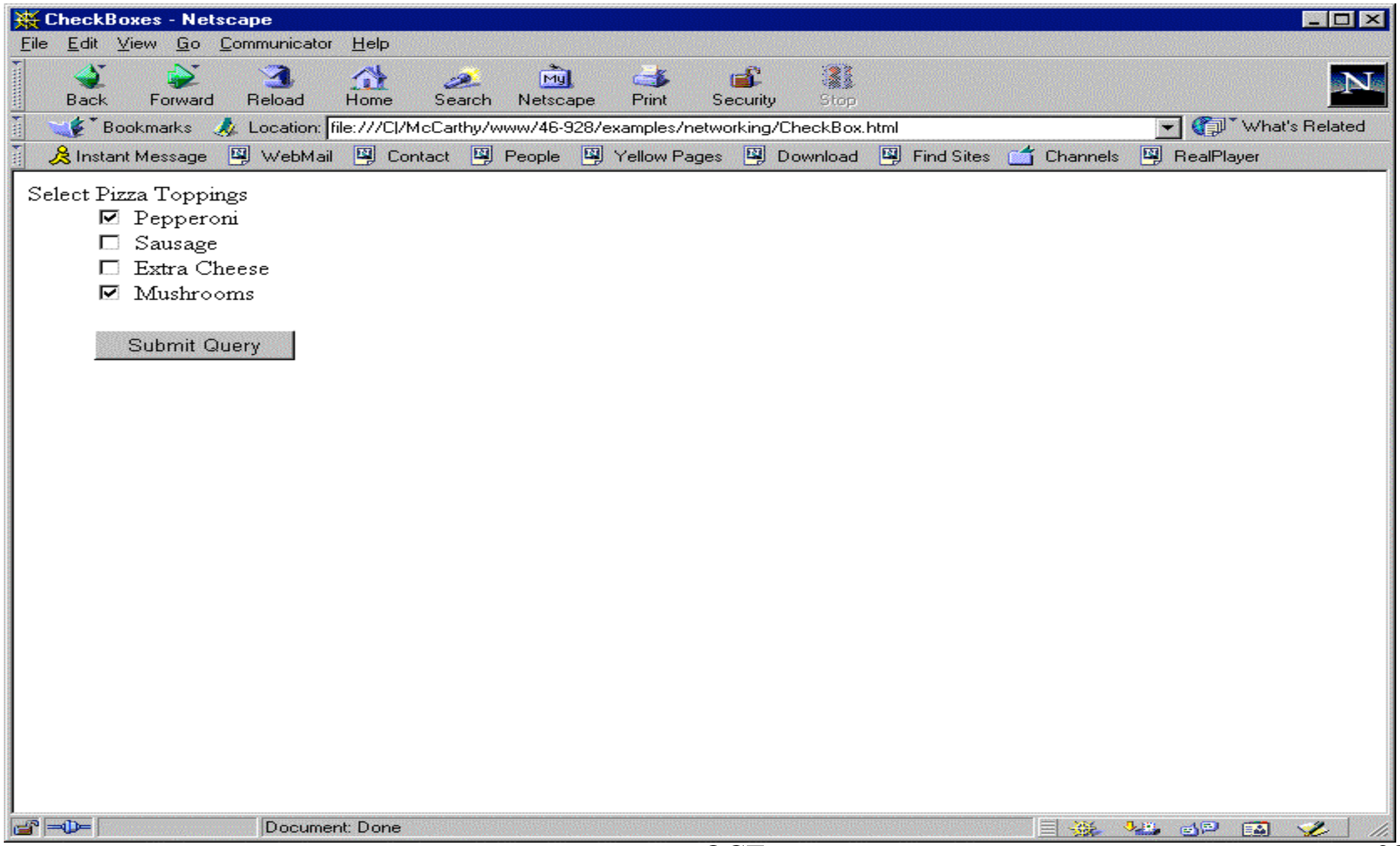
Here is your request line and request headers sent by your browser:

```
GET /?name=Mike+McCarthy&age=22/ HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Host: localhost:6502
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
```

## A Form With Checkboxes

```
<!-- CheckBox.html -->
<html>
<head>
<title>CheckBoxes</title>
</head>
<body BGCOLOR="WHITE">
   <form action="http://localhost:6502">
     <dl>
        <dt> Select Pizza Toppings </dt>
        <dd><Input type = "CheckBox" name = "Pepperoni"> Pepperoni
        <dd><Input type = "CheckBox" name = "Sausage"> Sausage
        <dd><Input type = "CheckBox" name = "Extra Cheese"> Extra Cheese
        <dd><Input type = "CheckBox" name = "Mushrooms"> Mushrooms
        <p> <input type = "submit">
     </dl>
   </form>
</body>
</html>
```

# CheckBoxes Browser

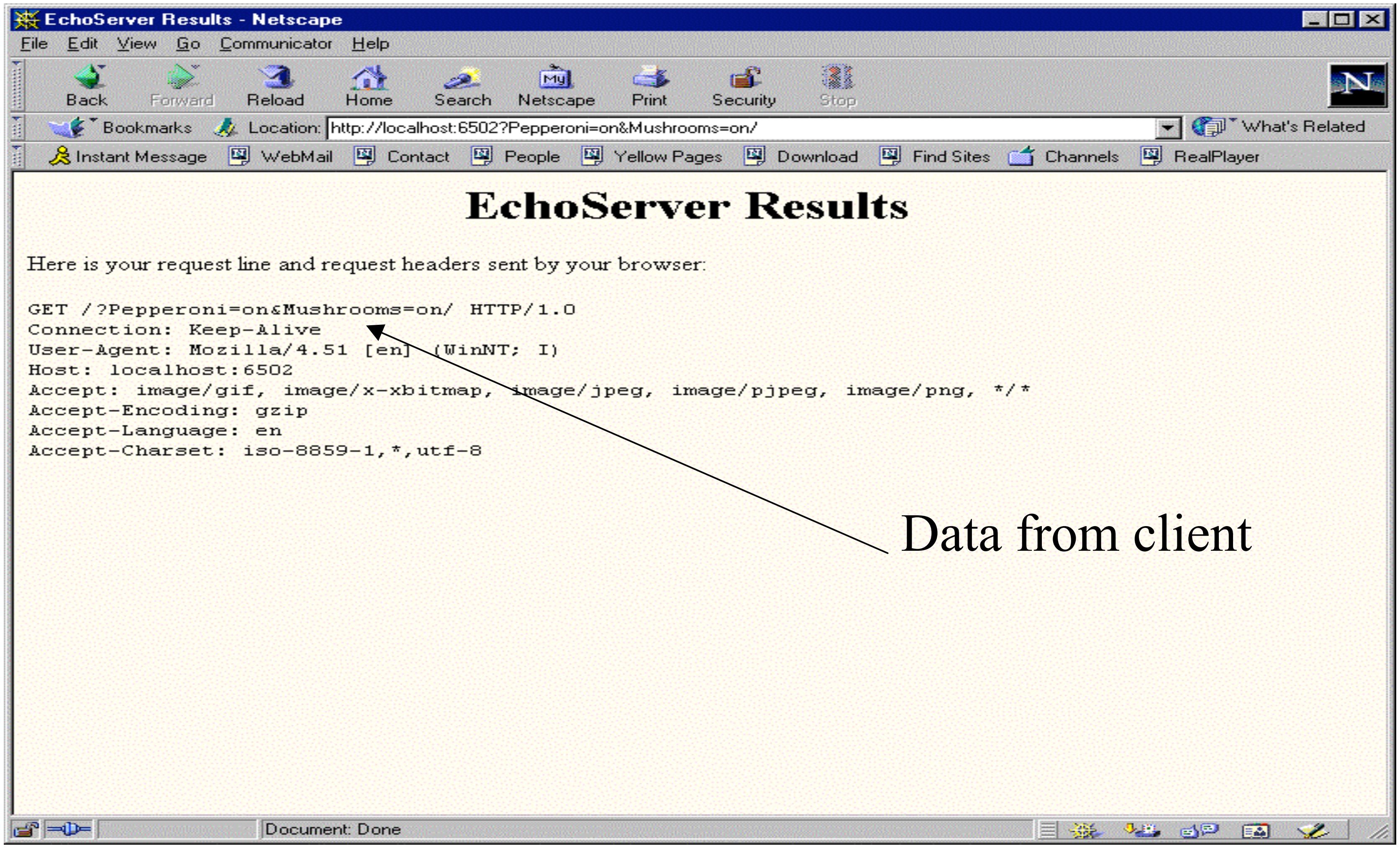

CheckBoxes - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///C|/McCarthy/www/46-928/examples/networking/CheckBox.html
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer

Select Pizza Toppings

- [✓] Pepperoni
- [ ] Sausage
- [ ] Extra Cheese
- [✓] Mushrooms

Submit Query

# CheckBox Response


EchoServer Results - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:6502?Pepperoni=on&Mushrooms=on/
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
Document: Done

**EchoServer Results**

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

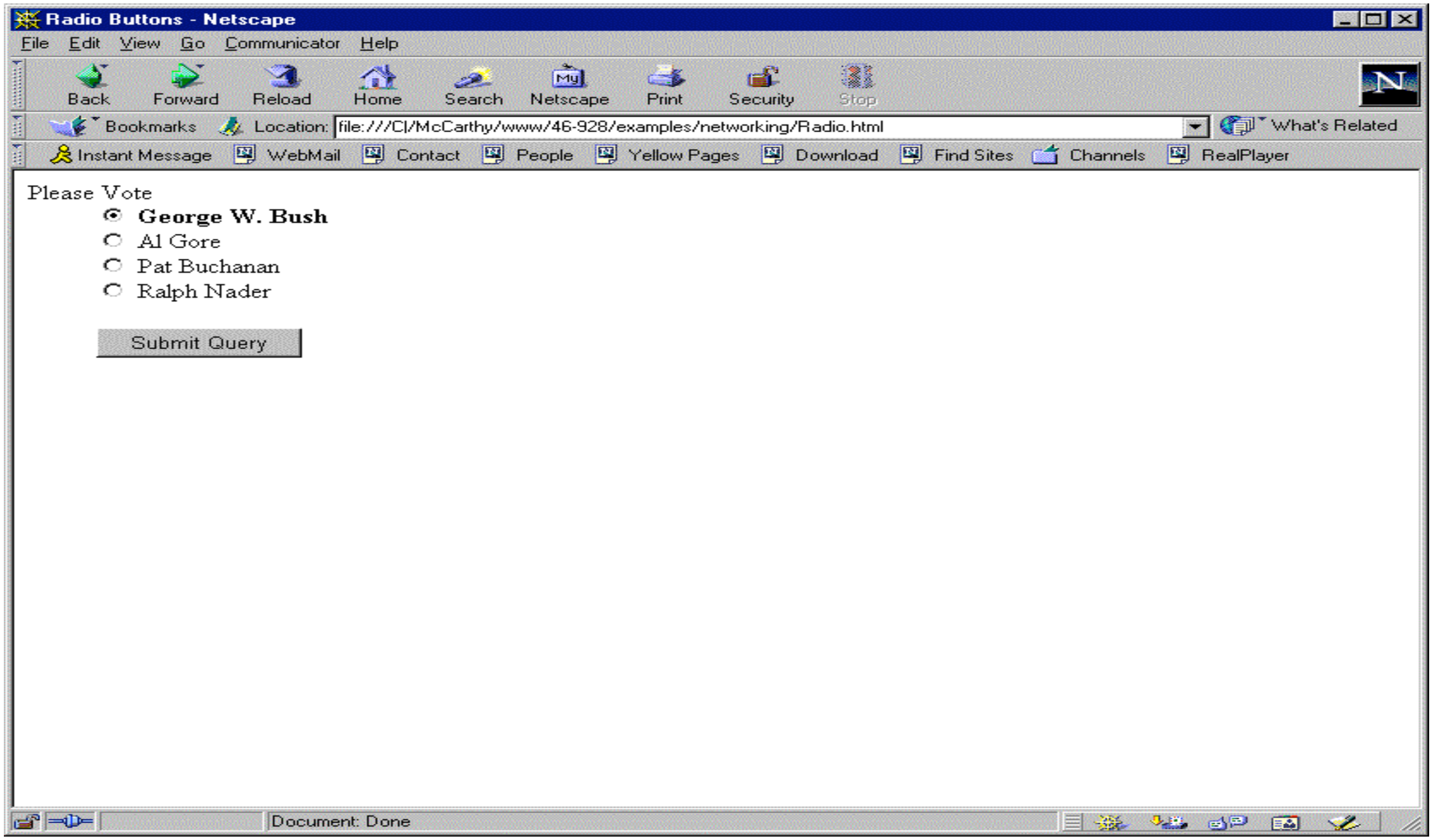
```
GET /?Pepperoni=on&Mushrooms=on/ HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Host: localhost:6502
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Data from client

# RadioBoxes HTML

```
<!-- Radio.html -->
<html>
<head>
<title>Radio Buttons</title>
</head>
<body BGCOLOR="WHITE">
  <form action="http://localhost:6502">
   <dl>                                               <!– Definition list -->
     <dt> Please Vote </dt>                           <!- The term to be defined left margin-->
                                                      <!-- Item definitions indented and below -->
     <dd><Input type = "Radio" name = "president" value= "Bush"> <b>George W. Bush</b>
     <dd><Input type = "Radio" name = "president" value = "Gore"> Al Gore
     <dd><Input type = "Radio" name = "president" value = "Buchanan"> Pat Buchanan
     <dd><Input type = "Radio" name = "president" value = "Nader"> Ralph Nader
     <p> <input type = "submit">
   </dl>
  </form>
</body>
</html>
```

# RadioBoxes Browser

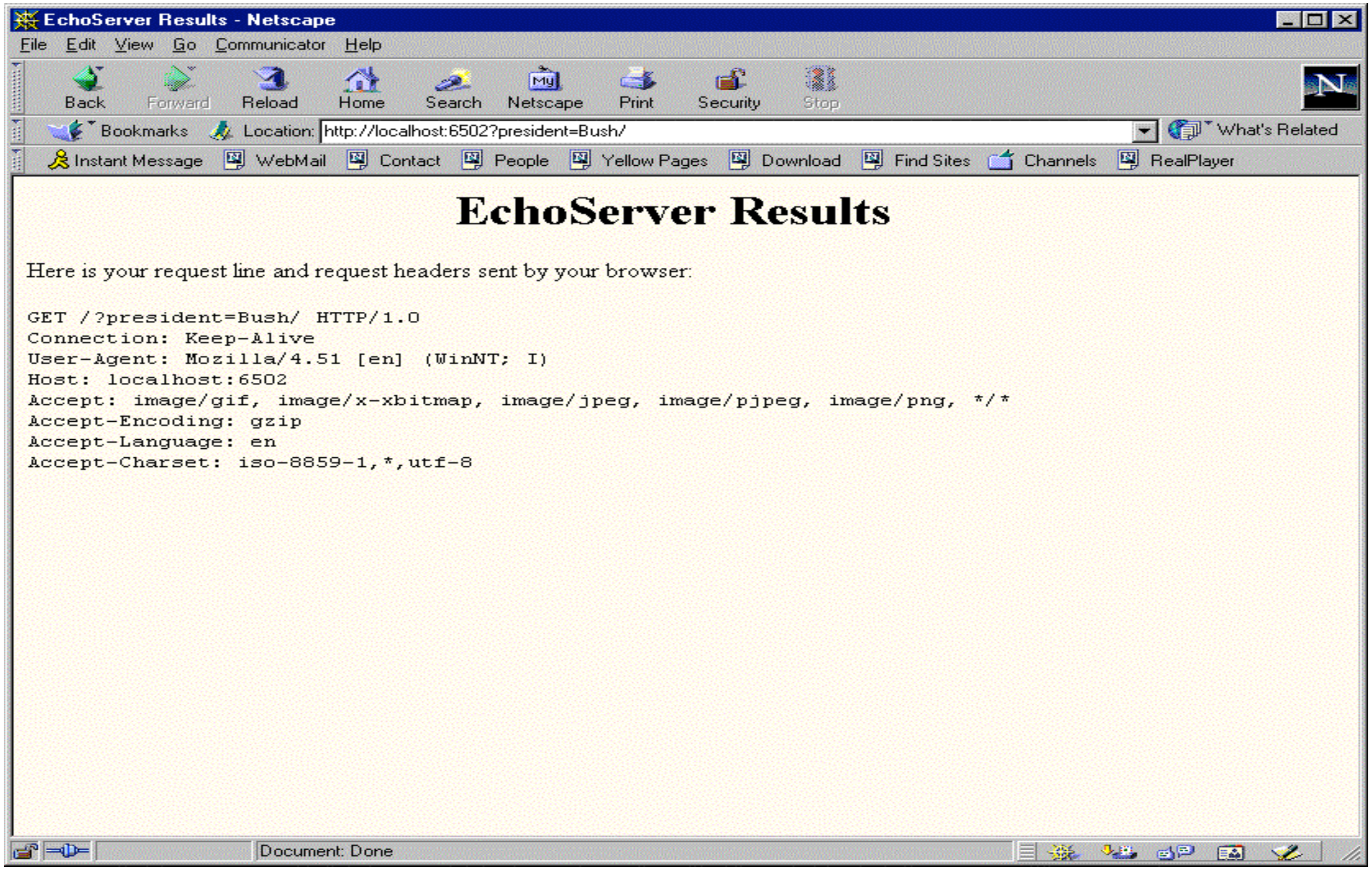

Radio Buttons - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///C|/McCarthy/www/46-928/examples/networking/Radio.html
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer

Please Vote

[✓ **George W. Bush**]
Al Gore
Pat Buchanan
Ralph Nader

Submit Query
Document: Done

# EchoServer Response



**EchoServer Results**

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

```
GET /?president=Bush/ HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Host: localhost:6502
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
```

## Template Design Pattern

In the code above, listen() calls handleConnection().
J2EE Servlets are another example of the Template Design Pattern.

From the Gang of Four:

"Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses."

An implemented method calls abstract methods which are implemented by a subclass.

The HttpServlet class provides a method called service() which calls one of 7 methods called doXXX(). The application programmer implements selected doXXX() methods.

# Reading Form Data With Servlets Under Tomcat

```
// **QueryData.java** -- Handle the voting form in radio.html
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class QueryData extends HttpServlet {

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

We have less work to do. We'll just implement doPost and doGet.

```
public void doGet(HttpServletRequest req,
              HttpServletResponse response)
              throws ServletException,
              IOException
 {
     String newPresident = req.getParameter("president");


     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     String docType = "<!DOCTYPE HTML PUBLIC \"//W3C//DTD" +
                      "HTML 4.0 ";
                      docType += "Transitional//EN\">\n";
```

If doPost() is called we'll treat it as a call on doGet().

What is a DTD and why is it here?

```
out.println(docType + "<HTML>\n" +
        "<HEAD><TITLE>Presidential Servlet" + "</TITLE>” +
        “</HEAD>\n" +
        "<BODY>\n" +
        "<H1>The new president is  "+ newPresident + "</H1>\n" +
        "</BODY></HTML>");
  }
}
```

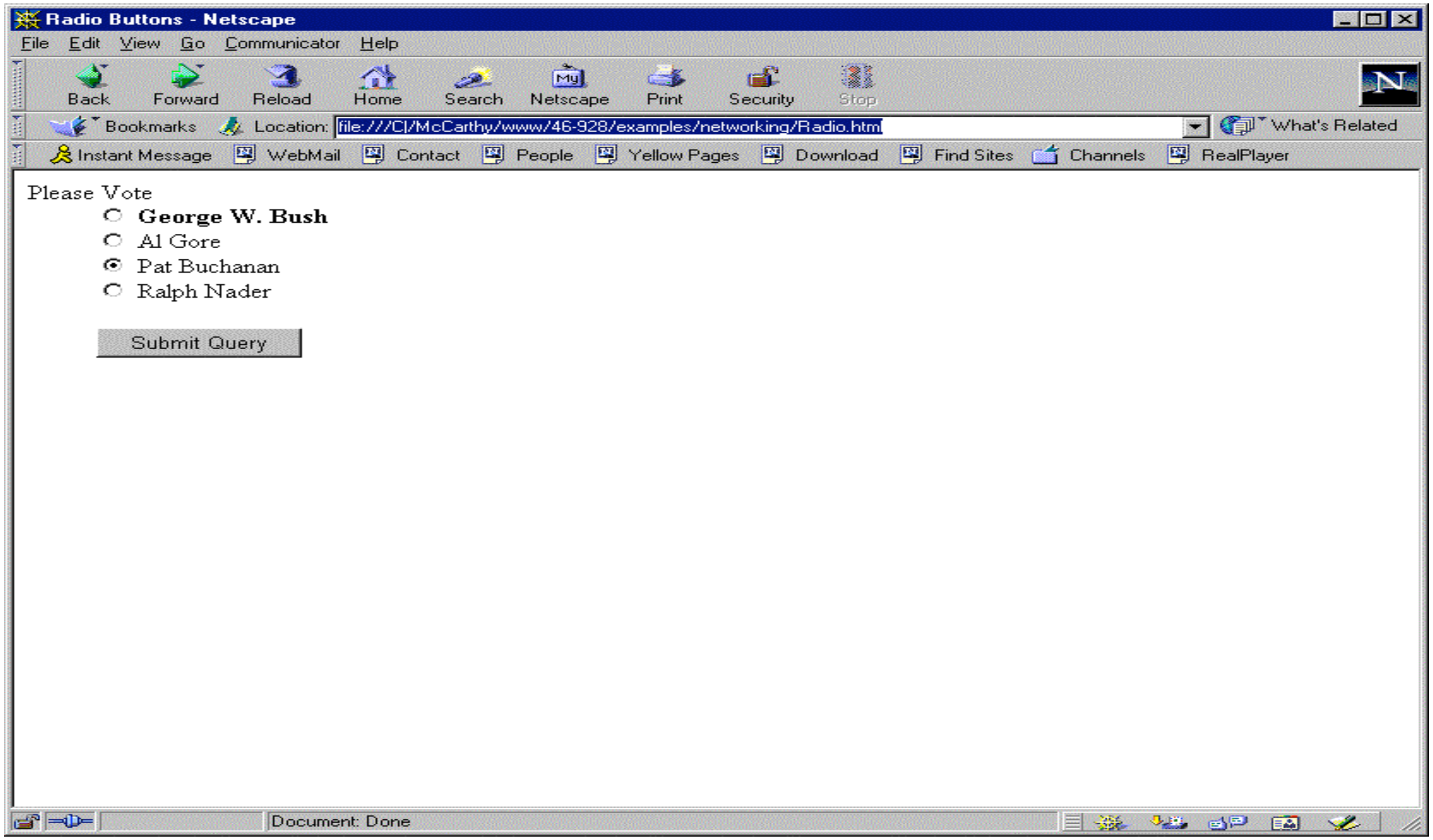
## <!-- Radio.html (Modified for servlets)-->

Tomcat's port

servlet

```
<html>
<head>
<title>Radio Buttons</title>
</head>
<body BGCOLOR="WHITE">
  <form action="http://localhost:8080/CoolServlet/QueryData">
    <dl>
      <dt> Please Vote </dt>
      <dd><Input type = "Radio" name = "president" value= "Bush">
                                                  <b>George W. Bush</b>
      <dd><Input type = "Radio" name = "president" value = "Gore"> Al Gore
      <dd><Input type = "Radio" name = "president" value = "Buchanan"> Pat Bucha
      <dd><Input type = "Radio" name = "president" value = "Nader"> Ralph Nader
      <p> <input type = "submit">
    </dl>
  </form>
</body>
</html>
```

# Radio HTML in the browser

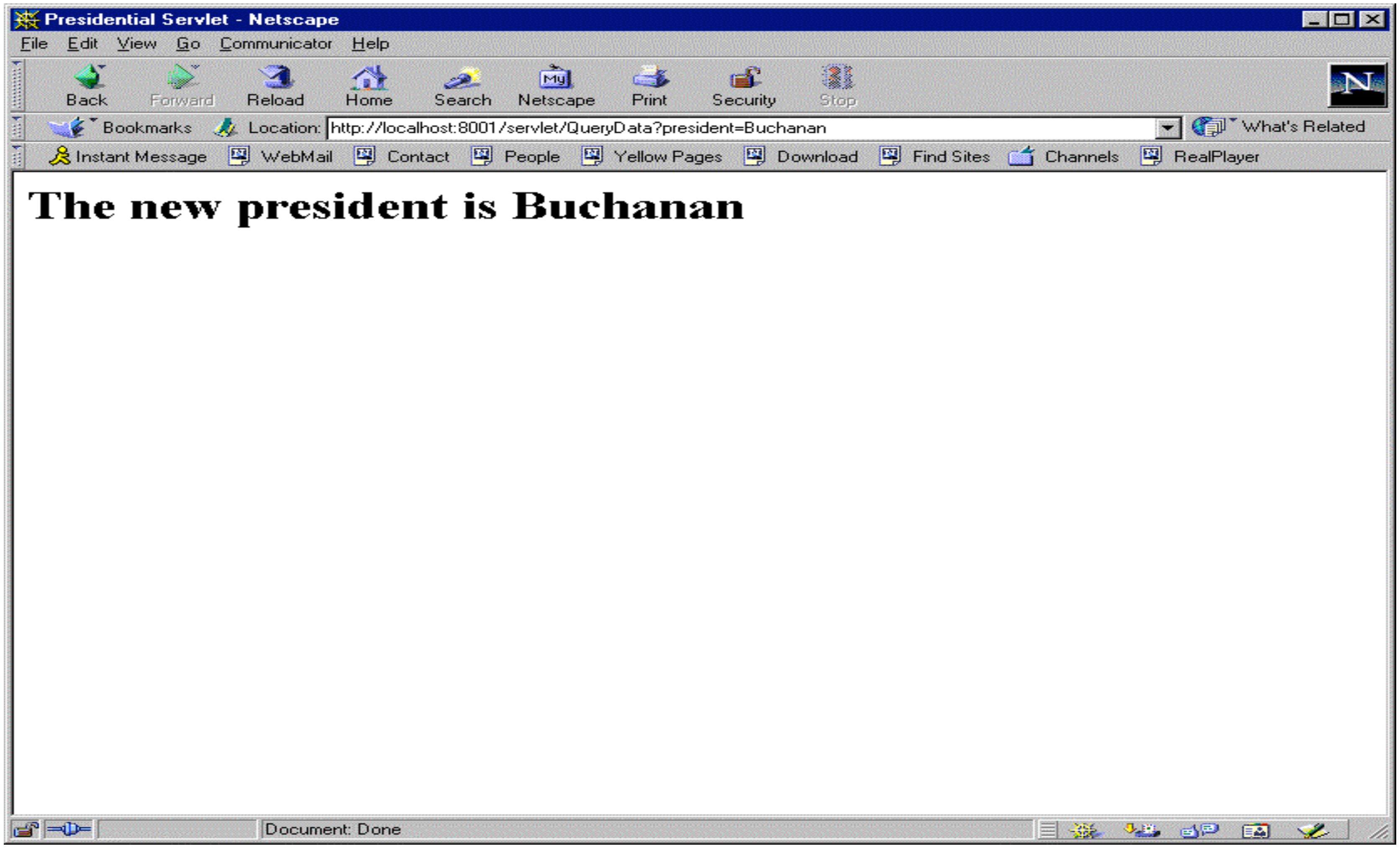

Radio Buttons - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///C|/McCarthy/www/46-928/examples/networking/Radio.html
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
Document: Done

Please Vote

- ○ **George W. Bush**
- ○ Al Gore
- ● Pat Buchanan
- ○ Ralph Nader

Submit Query

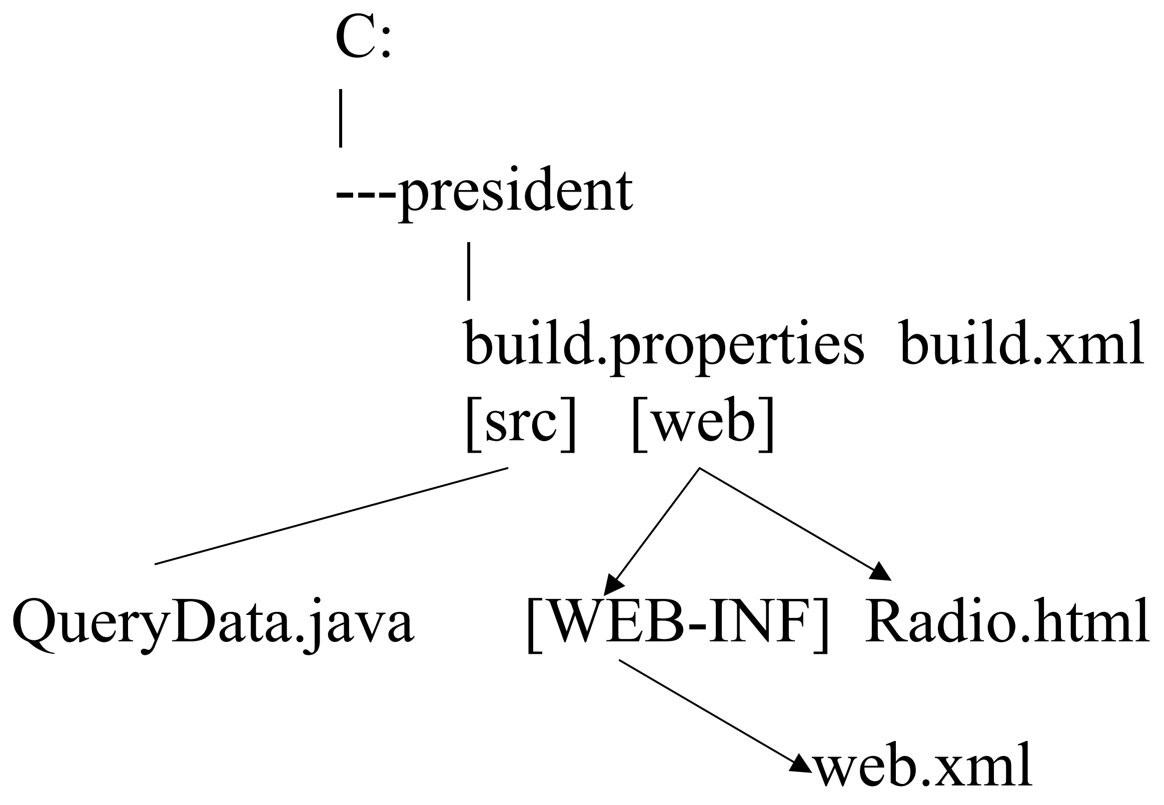
# The Servlet's Response



Presidential Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:8001/servlet/QueryData?president=Buchanan
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
Document: Done

**The new president is Buchanan**

# Organizing the J2EE File Structure



## A J2EE build.properties File

```
# Context path to install this application on
app.path=/CoolServlet

# Tomcat 5 installation directory
catalina.home=C:\Tomcat 5.5

# Manager webapp username and password
manager.username=xxxxxxx
manager.password=xxxxxxx
```

## A J2EE web.xml File

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
 Copyright 2002 Sun Microsystems, Inc. All rights reserved.
 SUN PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
-->
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
   "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
   <web-app>
   <servlet>
      <servlet-name>QueryDataId</servlet-name>
      <servlet-class>QueryData</servlet-class>
      <load-on-startup/>
   </servlet>
   <servlet-mapping>
      <servlet-name>QueryDataId</servlet-name>
      <url-pattern>/QueryData/*</url-pattern>
   </servlet-mapping>
</web-app>
```

## Running Ant on build.xml

Produces a build directory with the following structure:

```
build
 |    Radio.html
 |
 └──────WEB-INF
     |
     |    web.xml
     |
     ├──────classes
     |        QueryData.class
     |
     └──────lib
```

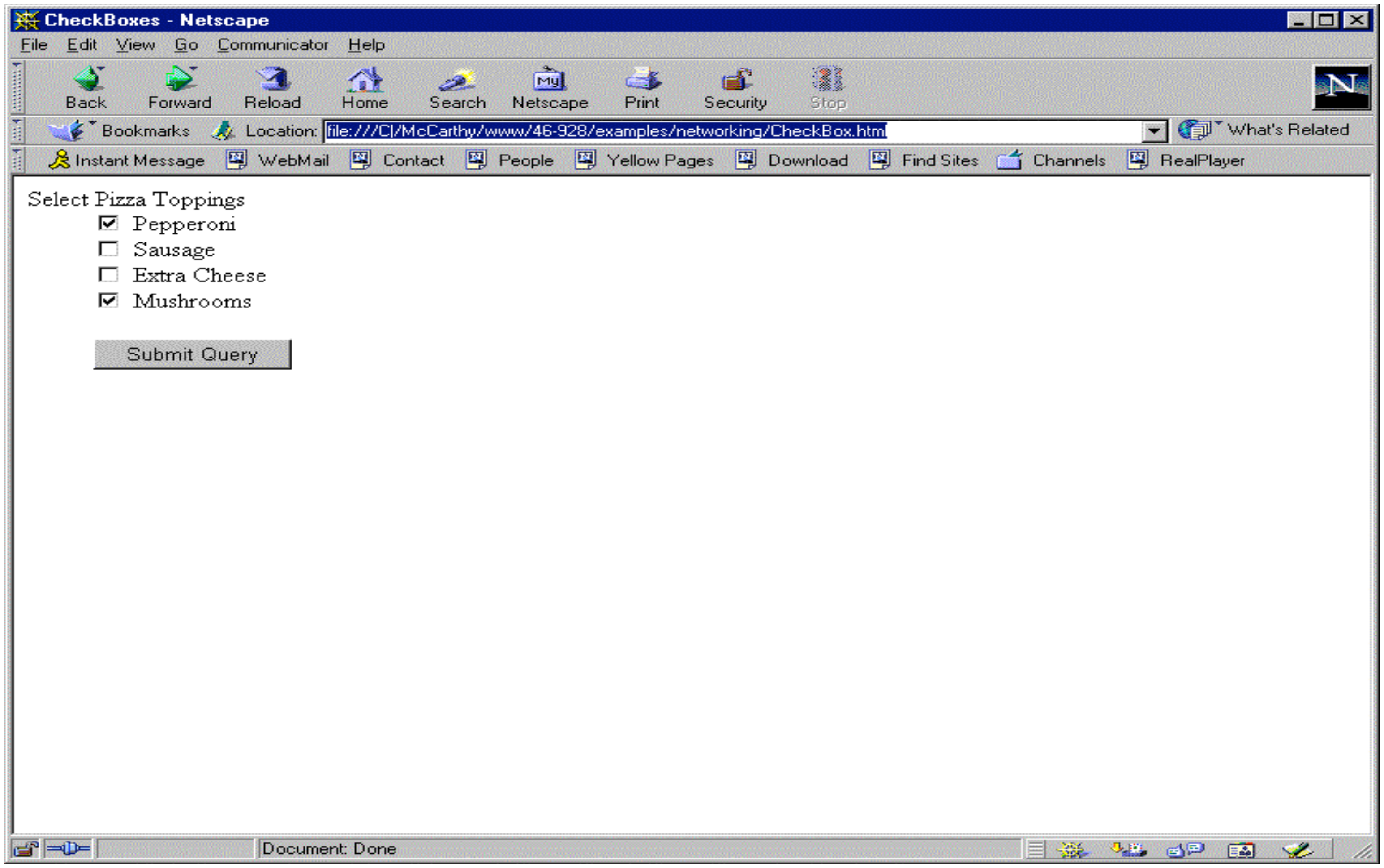
How does Ant compare with BPEL? What is BPEL?

## Handling CheckBoxes

servlet

```
<!-- CheckBox.html -->
<html>
<head>
<title>CheckBoxes</title>
</head>
<body BGCOLOR="WHITE">
   <form action="http://localhost:8080/servlet/PizzaData">
     <dl>
       <dt> Select Pizza Toppings </dt>
       <dd><Input type = "CheckBox" name = "Pepperoni"> Pepperoni
       <dd><Input type = "CheckBox" name = "Sausage"> Sausage
       <dd><Input type = "CheckBox" name = "Extra Cheese"> Extra Cheese
       <dd><Input type = "CheckBox" name = "Mushrooms"> Mushrooms
       <p> <input type = "submit">
     </dl>
   </form>
</body>
</html>
```

# Pizza Toppings

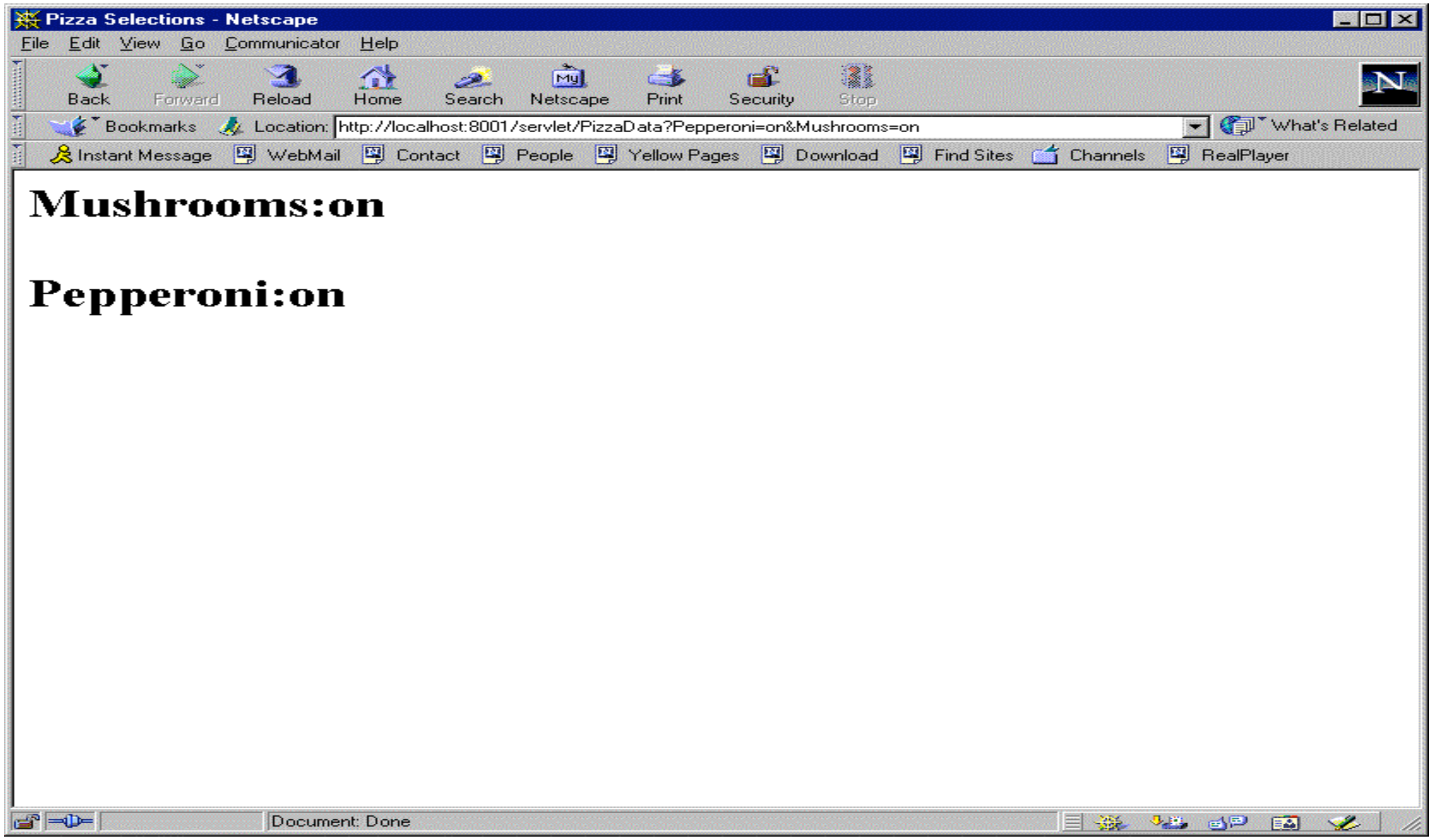

CheckBoxes - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///C|/McCarthy/www/46-928/examples/networking/CheckBox.html
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
Document: Done

Select Pizza Toppings

- [✓ Pepperoni]
- Sausage
- Extra Cheese
- [✓ Mushrooms]

Submit Query

# Servlet Response


Pizza Selections - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location:
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
Document: Done

```
http://localhost:8001/servlet/PizzaData?Pepperoni=on&Mushrooms=on
```

**Mushrooms:on**

**Pepperoni:on**

## PizzaData Servlet

```
// PizzaData.java -- Handle the toppings selection from pizza.html
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

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

         doGet(req, response);
   }
```

```
public void doGet(HttpServletRequest req,
                  HttpServletResponse response)
                  throws ServletException,
                  IOException
  {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     String finalString = "";

     Enumeration paramNames = req.getParameterNames();

     while(paramNames.hasMoreElements()) {
         String paramName = (String) paramNames.nextElement();
         finalString += paramName + ":" ;

         finalString += req.getParameter(paramName) + "<p>";
     }
```

Enumerate over the input.

```
String docType = "<!DOCTYPE HTML PUBLIC \"//W3C//DTD”
                        + “ HTML 4.0 ";
              docType += "Transitional//EN\">\n";

        out.println(docType +
                  "<HTML>\n" +
                  "<HEAD><TITLE>Pizza Selections" + "</TITLE>” +
                   “</HEAD>\n" +
                  "<BODY>\n" +
                  "<H1>" + finalString + "</H1>\n" +
                  "</BODY></HTML>");
  }

}
```

## Iterator Design Pattern

From the Gang of Four:

“Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.”

Where is the iterator pattern used in the code above?

## Part II Session Tracking and Servlet Collaboration

- First we will use a shared object
- Then we’ll use the new Session Tracking API

## Session Tracking with Servlets

HTTP is a stateless protocol.

We must have each user introduce themselves in some way.

We’ll look at traditional session tracking and then look at the Session Tracking API.

## Traditional Session Tracking

- User Authorization
- Hidden Form fields
- URL Rewriting
- Persistent cookies

We'll look at the first and last.

## User Authorization

- The web server requests the user name and password. The information is available to any servlet that needs it.
- The browser resends the name and password with each subsequent request.
- Data about the user and the user’s state can be saved in a shared object.

## Shared Objects

- A convenient way to store data associated with a user.
- There are likely to be many servlets running.
- They can collaborate through a shared object.
- Only one instance of the shared object should exist.
- It has to be available (in the classpath) of the servlets that needs it.
- It will be used by several threads and therefore should protect itself against simultaneous access.
- We'll look at a shared object and two servlets that use it.

## Singleton Design Pattern

The shared object will act like a global variable - available to any servlet that needs to read from it or write to it.

From the Gang of Four:

"Ensure a class only has one instance, and provide a global point of access to it."

An implemented method calls abstract methods which are implemented by a subclass.

The HttpServlet class provides a method called service() which calls one of 7 methods called doXXX(). The application programmer implements selected doXXX() methods.

## VisitTracker.java

```
// Servlet collaboration can be done through a shared object.
// Any servlet has access to this object and it only has one
// instance.
// It maintains a hash table of names and dates.

// Sections of code that must not be executed simultaneously
// are called critical sections. Java provides the synchronized
// keyword to protect these critical sections. For a synchronized
// instance method, Java obtains an exclusive lock on the class
// instance.

import java.util.*;
```

When will this constructor be called?

```
public class VisitTracker  {

    private Map nameDatePairs;
    private static VisitTracker instance = new VisitTracker();

    private VisitTracker() {          // private constructor
        nameDatePairs = new HashMap();
    }

    public static VisitTracker getInstance() { return instance; }

    synchronized public void addVisit(String userName) {

        nameDatePairs.put(userName, new Date());
    }
synchronized public Date lastVisit(String name) {

            Date d = (Date)nameDatePairs.get(name);
            return d;

      }
}
```

# User Authorization

- Administered by the web server – Tomcat
- Edit Tomcat's deployment descriptor
- From within the servlet use String name = req.getRemoteUser(); to access the user name.
- We have to assign user names and passwords.

  tomcat-users.xml

```
<tomcat-users>
  <user name="tomcat" password="tomcat" roles="tomcat" />
  <user name="role1"  password="tomcat" roles="role1"  />
  <user name="both"   password="tomcat" roles="tomcat,role1" />
  <user name="mike"   password="tomcat" roles="student" />
</tomcat-users>
```

- The following will keep track of the date of the last visit.

```
// UserAuthorizationDemo.java
// This servlet reads from Tomcat and finds the name of the
// authorized user. It then adds it to a hash table storing
// the time of this visit. It makes use of VisitTracker.

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

public class UserAuthorizationDemo extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
```

```
   res.setContentType("text/plain");
   PrintWriter out = res.getWriter();
   String name = req.getRemoteUser();    // ask the server
   if(name == null) {
        System.out.println("The system administrator should protect" +
                          " this page.");
   }
   else {

        out.println("This user was authorized by the server:" + name);
        VisitTracker visit = VisitTracker.getInstance();
        Date last = visit.lastVisit(name);
        if(last == null) out.println("Welcome, you were never here before");
        else out.println("Your last visit was on " + last);
        visit.addVisit(name);
     }
   }
}
```

## Cookies

- A cookie is a bit of information sent by a web server to a browser that can later be read back from that browser.
- The server can take that bit of information and use it as a key to recover information about prior visits. This information may be in a database or a shared object.
- Cookies are read from the request object by calling getCookies() on the request object.
- Cookies are placed in the browser by calling addCookie() on the response object.

## Using Cookies

```
// CookieDemo.java

// This servlet uses a cookie to determine when the
// last visit by this browser occurred. It makes use of
// the VisitTracker object.

// Cookies normally expire as soon as the browser exits.
// We want the cookie to last one year and so we use
// setMaxAge(seconds) on the cookie.

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

```
public class CookieDemo extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse re
                    throws ServletException, IOException {

            res.setContentType("text/plain");
            PrintWriter out = res.getWriter();

            Cookie[] c = req.getCookies();
            // If this person has been here before then we should have
            // a cookiedemouser field assigned to a unique id.

            String id = null;
```

```
if (c!=null) {   // we may have the cookie we are after

   for (int i=0;i<c.length;i++) {

       if (c[i].getName().equals("cookiedemouser")) {

          id = c[i].getValue();
       }
       break;
   }
}
```

```
    if (id == null) {
            // They have not been here before and need a
            // cookie. We get a unique string (with respect
            // to this host)and make sure it is of the 'query string' form
            // It uses the clock. Don’t turn the clock back!
            String uid = new java.rmi.server.UID().toString();
            id = java.net.URLEncoder.encode(uid);
            Cookie oreo = new Cookie("cookiedemouser",id);
            oreo.setMaxAge(60*60*24*365);
            res.addCookie(oreo);
     }
     VisitTracker visit = VisitTracker.getInstance();
     Date last = visit.lastVisit(id);
     if(last == null) out.println("Welcome, you were never here befo
     else out.println("Your last visit was on " + last);
     visit.addVisit(id);
  }
}
```

## The New Session Tracking API

- Support may vary depending on the server.
- Implemented with cookies or with URL rewriting if cookies fail (URL rewriting requires help from the servlet).
- Every user of the site is associated with a javax.servlet.http.HttpSession object
- The session object can hold any arbitrary set of Java objects.
- Servlets collaborate by accessing the session object.
- The following example abstracts away shared object concerns.
- All valid sessions are grouped together in a HttpSessionContext object

## The Session Tracking API

```
// SessionDemo.java
// The session object associated with this user/browser is available
// to other servlets.

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

public class SessionDemo extends HttpServlet {
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {

        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();

        // Get the current session object. Create one if none exists.
        HttpSession session = req.getSession(true);

        // Get the Date associated with this session
        Date d = (Date)session.getAttribute("dateofvisit");

        if(d == null)  out.println("Your first time, welcome!");

        else  out.println("Your last visit was on " + d);

        session.setAttribute("dateofvisit", new Date());
    } }
```