

Servlet 2

Servlets

Similar to CGI

Run on the Server

More features

Run faster, since avoid "startup" costs per hit

(there are Apache modules that make this same optimization for Perl CGI's)

Java oriented

Servlet Structure

Java servlet objects exist on the server side

The servlets are installed in a running Java VM with servlet infrastructure --

we're not going to worry about how it works too much. We just write the servlet, install it, and it should work. (as we did with CGI's and the CGI server)

Request -> doGet()

When a client request comes in, it is sent to the servlet for handling, the doGet() notification is sent to the servlet in its own thread.

It's fast -- the servlet object is probably already exists in memory, so there's no setup -- just send the doGet()

Because the servlet object is present in memory the whole time, it can build "session" (below) state in memory that exists across multiple requests

Threading / ivars

By default, each request comes in on its own thread

There is one servlet object

However, there can be multiple requests running doGet() on that servlet at one time

For the most part, this is not a problem and gives better performance

However, it means that you should not use instance vars in the servlet object to store things that are part of a particular session -- use the session object itself (below) to store session information

1. Subclass off HttpServlet

2. Override doGet() message

Sent on client hit with the GET method

Also, override doPost(), and have it call doGet()

3. HttpServletRequest

Represents the client request

send `getParameter("paramName")` to search for form bindings -- returns null if no binding

(later) We'll get cookies the client sends us from the request as well

4. HttpServletResponse

a. Set the content type and other HTTP header attributes

b. Get `PrintWriter` from response object

c. Send `println()` messages to the `printwriter` to send text to the client

HTTP header before content

It's important that (a) be completed before doing (b) and (c), since (b) will start the sending of the HTTP header/content system. Anything that's going to be in the HTTP header must be set before servlet starts sending the HTTP header/content bytes.

5. No Instance Variables

This goes against the OOP style, but servlets should not have instance variables -- we'll use the "session" object below to store state instead of instance variables.

HelloServlet.java

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

public class HelloServlet extends HttpServlet
{

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        // Set the content type on the HTTP response
        response.setContentType("text/html");

        // Generate the HTML part of the response
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");

        String title = "Hello World";

        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor=white>");

        out.println("<h1>" + title + "</h1>");
    }
}
```

```

out.println("<p>Hello World has left the building");

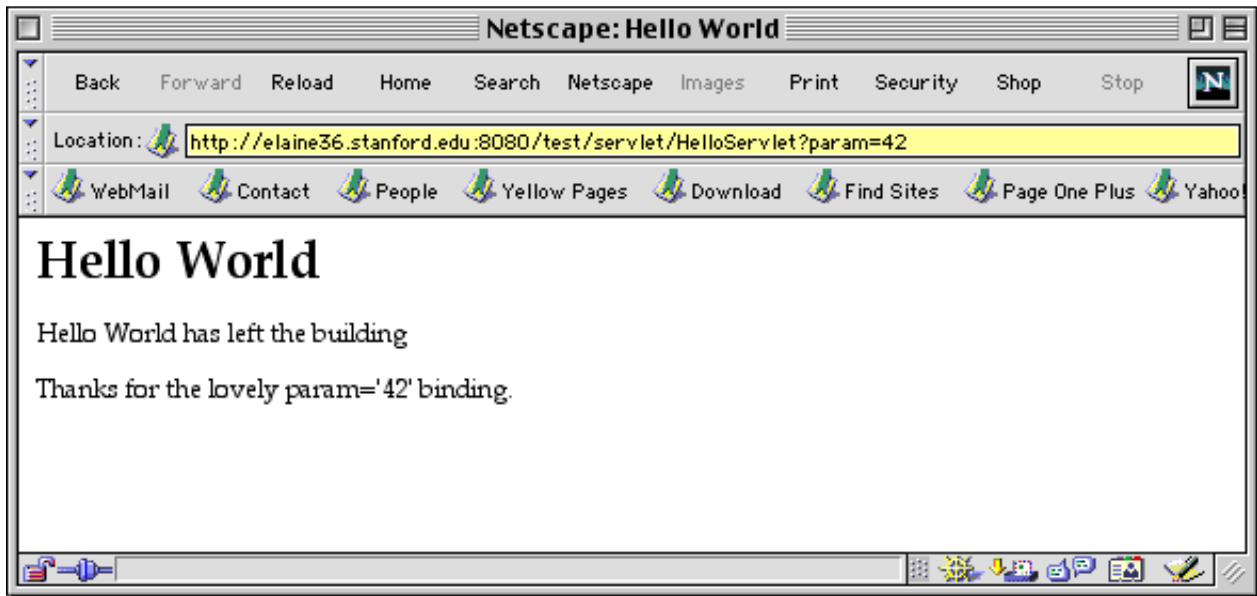
// Pull a binding off the request (may be null)
String param = request.getParameter("param");

if (param != null) {
    out.println("<p>Thanks for the lovely param='" + param + "' binding.");
}

out.println("</body>");
out.println("</html>");
}

// Override doPost() -- just have it call doGet()
public void doPost(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}
}

```



Sessions

Servlet system can associate a "session" with a sequence of client requests
 The servlet system can do this with cookies, although you don't really need to know that

The session object is like a hash table that persists from one request to the next
 Store things you want to remember in it

request.getSession(boolean)

Check for/create a Session Object

getSession(false) checks for an existing session

getSession(true) checks, and then creates one if necessary

session.getAttribute(key)

Returns the object stored in the session under the given String key, or null

The return type is Object -- cast it to its proper type -- e.g. Tracker below

session.setAttribute(key, value);

Install a key/value binding in the session

Tracker class

Custom class that stores stuff we want to store for this session

Stored in the session under the key "tracker"

Counts the number of transactions: upCount(), getCount()

Keeps a Vector of the names of the pages we've visited just to show off

HelloSession.java

```
// HelloSession.java

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

/*
 Represents three pages: A, B, and C -- the form
 has a button to go to each of the three.

 Creates a "session" object on the first client interaction.

 Uses the session to keep track of sequence of visits over the whole
 session.
*/

public class HelloSession extends HttpServlet {

 // One instance of the Tracker class is what is stored in the
 // session. The Tracker keeps track of the number of interactions
 // and an array of the sequence of pages visited.
 // Java note: you can declare nested utility classes like this
 private class Tracker {

 private int count;
 private Vector series; // Vector is a dynamic array that stores Objects
 // Vector responds to: size(), addElement(), and elementAt()

 public Tracker() {
 count = 0;
 }
 }
 }
 }
```

```

    series = new Vector();
}

public void upCount() {
    count++;
}

public int getCount() {
    return(count);
}

public void add(Object object) {
    series.addElement(object);
}

public void printSeries(PrintWriter out) {
    for (int i=0; i<series.size(); i++) {
        out.println(series.elementAt(i) + " ");
    }
}

}

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

    out.println("<html>");
    out.println("<head>");

    String title = "Hello Session";

    out.println("<title>" + title + "</title>");
    out.println("</head>");
    out.println("<body bgcolor=white>");

    out.println("<h1>" + title + "</h1>");

    // See if there is a session already
    // If not, create the session and put a new tracker in it
    // If there is an existing session, get the tracker out of it
    Tracker tracker = null;
    HttpSession session = request.getSession(false);

    if (session == null) {
        out.println("Hello there! I haven't seen you before have I? Creating a
session.");
        session = request.getSession(true);
        tracker = new Tracker();

        session.setAttribute("tracker", tracker);
    }
    else {
        tracker = (Tracker) session.getAttribute("tracker");
    }
}

```

```

    }

    // At this point, a session and tracker object are established
    // Increment the number of visits and print the greeting
    tracker.upCount();
    String time;
    if (tracker.getCount() == 1) time = " time ";
    else time = " times ";
    out.println("Dang if we haven't served you " +
               tracker.getCount() + time + "this session");

    // Standard technique to look at which button was pressed to
    // return the right page (the page string is for the visit tracking)
    String page;
    if (request.getParameter("gotob") != null) {
        doB(out);
        page = "b";
    }
    else if (request.getParameter("gotoc") != null) {
        doC(out);
        page = "c";
    }
    else {
        doA(out);          // the page of last resort
        page = "a";
    }

    tracker.add(page);

    // Just to show off the session state we are keeping,
    // echo the series of visits for this whole session
    out.println("<hr>Order of visits so far...");
    tracker.printSeries(out);

    // Print the form with the A B C buttons last
    doForm(out);

    out.println("</body>");
    out.println("</html>");
}

// Separated out utilities for the HTML generation...

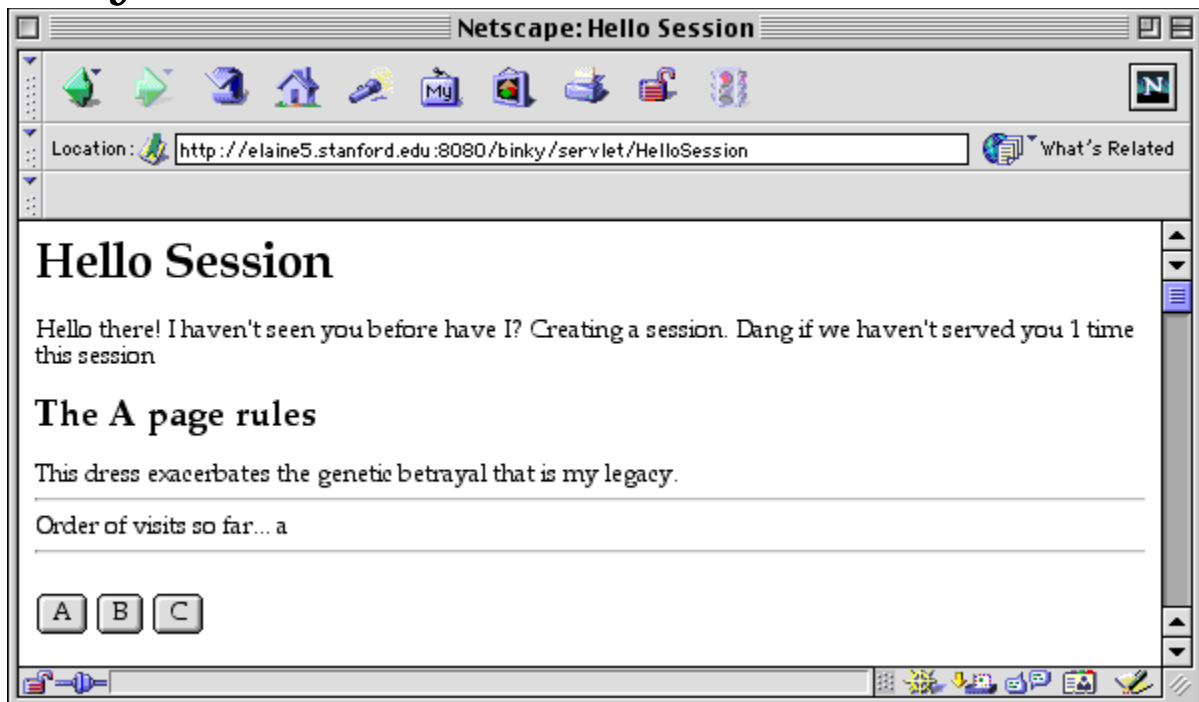
public void doForm(PrintWriter out) {
    out.println("<hr><form><input name=gotoa type=submit value=A>");
    out.println("<input name=gotob type=submit value=B>");
    out.println("<input name=gotoc type=submit value=C></form>");
}

public void doA(PrintWriter out) {
    out.println("<h2>The A page rules</h2>");
    out.println("<p>This dress exacerbates the genetic betrayal that is my
legacy.!</p>");
}

```

```
}  
  
public void doB(PrintWriter out) {  
    out.println("<h2>The B page rules</h2>");  
    out.println("<p>Clattu barrada nickto.");  
}  
  
public void doC(PrintWriter out) {  
    out.println("<h2>The C page rules</h2>");  
    out.println("<p>Picture this -- I'm hiding naked in a refrigerator.");  
}  
}
```

Very First Hit



A Later Hit

