

# Servlets

---

## "Session" Theory

### Session = overall dialog

Keep information associated with the overall dialog.

#### e.g. Cart

As the user clicks around, they keep adding things to their cart. Maintain a list of the cart contents. Ideally, this works even as they hit the back button (problem with hidden field storage).

#### e.g. Login

On the first page, the user types in their username/password. For all subsequent pages, we want to use that users prefs.

## Server Side Session Storage — ID

Instead of storing the data in the form hidden field, etc. store the state on the server side in, say, a database. In the form, just store the ID of the session. This gets around some of the back-button problems.

### #1 - Hidden Fields

### #2 - URL re-writing

#### Modify HTML

Modify every URL on the generated HTML to include the session id.  
At the end of the url put .../cgi-bin/foo.pl/session-id=34567

#### Brute-Force

This is a bit clumsy, but it works.  
Problems if they come to the site from some other url, such as on another site.

### #3 Cookies

#### Store Server->Client

#### Client Stores

Have the client store the session-id in a cookie.

#### On request, Client->Server

On subsequent requests, the client sends the cookie back to the server, reminding the server of what session this dialog is a part of.

## Servlet Theory

Java servlet objects exist on the server side

The servlets are installed in a running 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)

When a client request comes in, it is sent to the servlet for handling

It's fast -- the servlet is probably already running in memory

Because the Java VM is present the whole time, it can build "session" state in memory that exists across multiple requests

### 1. Subclass off HttpServlet

### 2. Override doGet() message

sent on client hit

### 3. HttpServletRequest

Represents the client request

send `getParameter("paramName")` to search for form bindings  
(later) We'll get cookies the client sends to us out of here too

### 4. HttpServletResponse

Get `PrintWriter` from response object

Send `println()` messages to the `PrintWriter` to send text to the client

### 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.

## ==Hello Servlet

```
// HelloServlet

/*
 * A simple sevlet that returns a single page.
 * It looks for a binding called "param" and if present incorporates
 * it into its response.
 */

import java.io.*;
import java.text.*;
import java.util.*;
```

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

public class Hello extends HttpServlet {

    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 World";

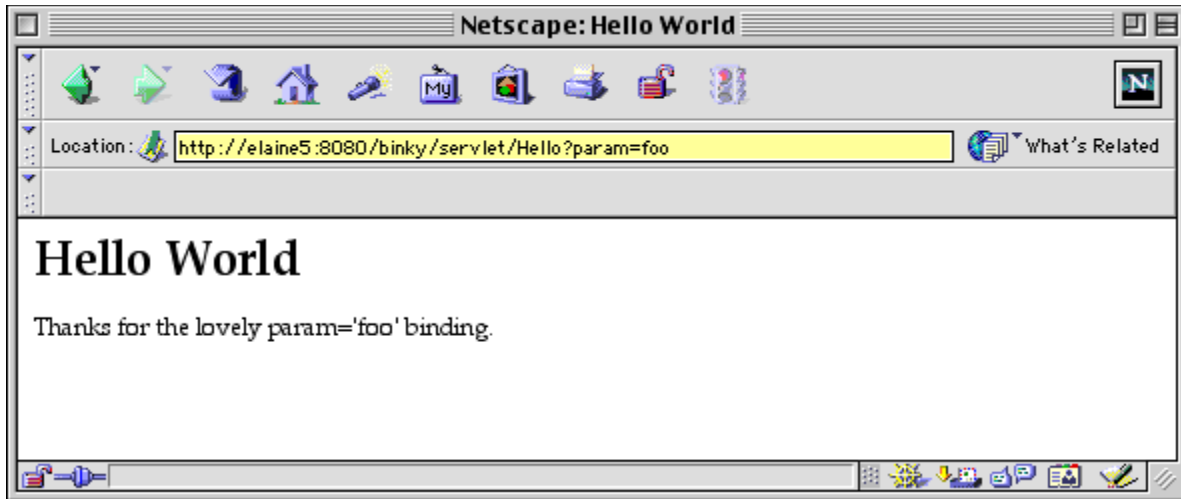
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor=white>");

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

        String param = request.getParameter("param");

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

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



## Sessions

### 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

The session object is like a hashtable that exists across several transactions

### session.getValue(key)

### session.putValue(key, value);

## 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 Example

```
// HelloSession

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.putValue("tracker", tracker);
}
else {
    tracker = (Tracker) session.getValue("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...

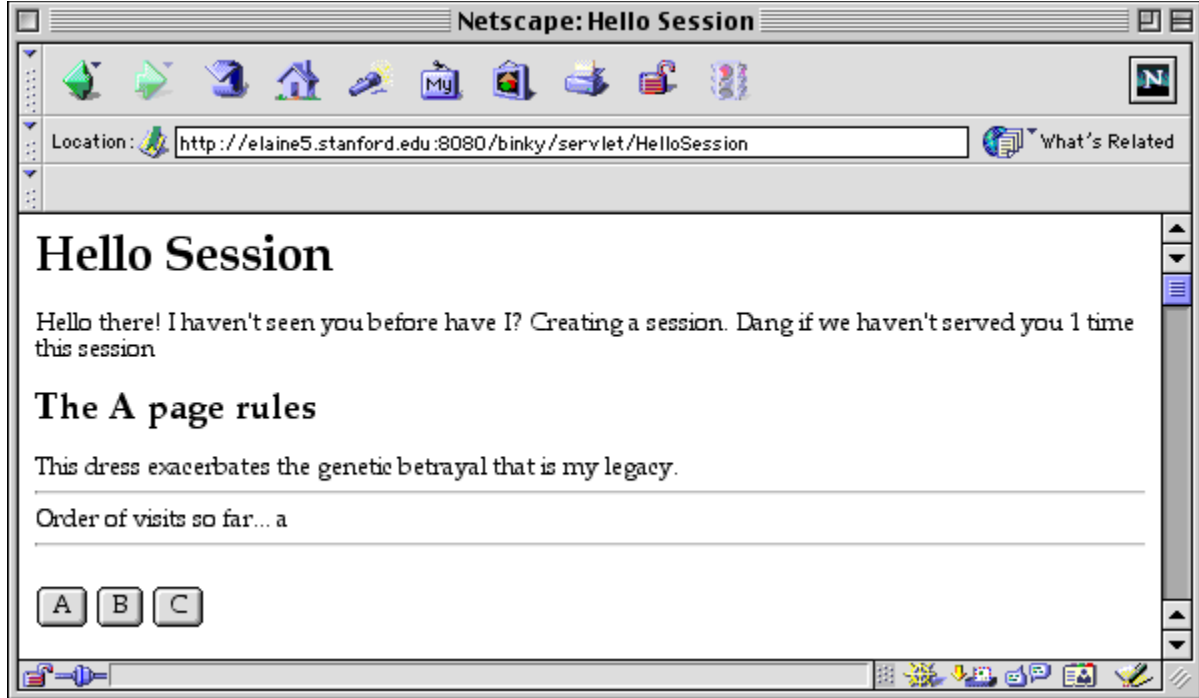
public void doForm(PrintWriter out) {
    out.println("<hr><form><input name=gotoa type=submit value=A>");
    out.println("<input name=gotob type=submit value=B>");
    out.println("<input name=gotoc type=submit value=C></form>");
}

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.</p>");
}

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.</p>");
}
}
```

## Very First Hit



## A Later Hit

