

# Servlets, Sessions, and Cookies

## Lecture 8

cs193i – Internet Technologies

Summer 2004

Stanford University

## Administrative Stuff

- HW #3 due August 2
- Lab #3 due August 4
- Local SCPD students must take final on-campus

## Cookies and Privacy

- Cookies are good
  - Remember who you are and your preferences
  - Session tracking
- Cookies are bad
  - When developer is not careful  
(store password, credit card info, etc...)
  - When people abuse them  
(track information about you)

## The Big Picture

- Making Web Applications Better
- For the Developer
  - More features, better API, ...
  - Time-To-Market (beat your competitors)
- For the End-User
  - Better Continuity
  - Better User Experience

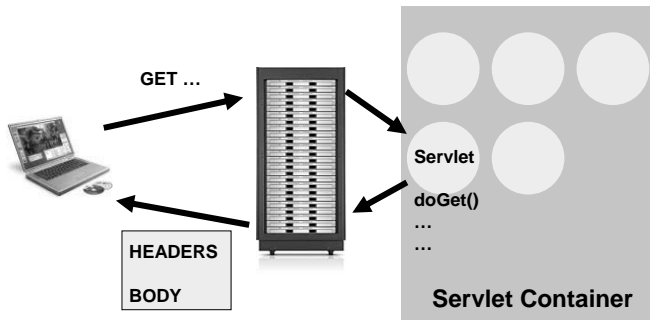
## Why Java Servlets Instead of CGI?

- Efficient, Convenient, Powerful, Portable, Secure, Inexpensive
  - Lightweight threads instead of OS threads created
  - Single copy of code brought into memory for all threads versus per thread
  - Data (session state) can be stored across threads within servlet container
  - Java is portable and secure
  - Requires little expense once servlet container integrated with web server

## Servlet Structure

- Java Servlet Objects on Server Side
- Managed by Servlet Container
  - Loads/unloads servlets
  - Directs requests to servlets
- Request → doGet()
- Each request is run as its own thread

## Web App with Servlets



## 5 Simple Steps for Java Servlets

1. Subclass off HttpServlet
2. Override doGet(...) method
3. HttpServletRequest
  - getParameter("paramName")
4. HttpServletResponse
  - set Content Type
  - ...
  - get PrintWriter
  - send text to client via PrintWriter
5. Don't use instance variables

## Servlet/JSP Container

- Java Servlet 2.4
- JavaServer Pages 2.0
- Tomcat is the basis for the official reference implementation



## HelloWorld.java

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

public class HelloWorldExample 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>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"white\">");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## RequestHeaderExample.java

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

public class RequestHeaderExample extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        Enumeration e = request.getHeaderNames();
        while (e.hasMoreElements()) {
            String name = (String)e.nextElement();
            String value = request.getHeader(headerName);
            out.println(name + " = " + value );
        }
    }
}
```

## Servlet Lifecycle (Creation)

- Single instance created
- init() method called
- You can override init() in your subclass of HttpServlet to do some initial code...
- init() is NOT called again on further requests

## Servlet Lifecycle (Service Method)

- On each request, the server spawns a new thread and calls `service()`
- `service()` checks HTTP request type and calls appropriate `doXXXX` (Get, Post, Put...)
- don't override `service` (unless you really know what you're doing)

## Servlet Lifecycle (doGet(), doPost())

- Real meat of the web app is here
- `doPost()` can call `doGet()`, or viceversa
- no `doHead()`... system uses headers of `doGet()` result

## Servlet Lifecycle (destroy())

- For some reason (servlet idle, etc) the server may want to remove the servlet from memory
- `destroy()` allows you to close DB connections, wrap up, etc...
- Don't count on `destroy` to write persistent state (server may crash before you ever get here!)

## Accessing Request Components

- `getParameter("param1")`
- `getCookies() => Cookie[]`
- `getContentTypeLength()`
- `getContentType()`
- `getHeaderNames()`
- `getMethod()`

## Environment Variables

- JavaServlets do not require you to use the clunky environment variables used in CGI
- Individual functions:
  - `PATH_INFO` `req.getPathInfo()`
  - `REMOTE_HOST` `req.getRemoteHost()`
  - `QUERY_STRING` `req.getQueryString()`
  - ...

## Setting Response Components

- Set status first!
  - `setStatus(int)`  
`HttpServletResponse.SC_OK...`
  - `sendError(int, String)`
  - `sendRedirect(String url)`

## Setting Response Components

- Set headers
  - `setHeader(...)`
  - `setContentType("text/html")`
- Output body
  - `PrintWriter out = response.getWriter();`
  - `out.println("<HTML><HEAD>...")`

## J2EE API

- <http://java.sun.com/j2ee/1.4/docs/api/index.html>
- `HttpServletResponse`, `HttpServletRequest`, `HttpServlet`, `HttpSession...`

## Developing Servlets (Start w/ baby steps)

- Install Tomcat
- Run Tomcat
- Run examples

## Creating Your Own Servlet

- Write new servlet (e.g. `Hi.java`)
- Make sure Tomcat jar files are in your classpath
- Compile servlet (`javac Hi.java`)
- Edit `web.xml`
- Restart the Tomcat Server/Servlet Container
- `http://<host>:8080/<webappname>/servlet/Hi`

## Debugging

- use `out.println` to the html
- print to a socket on localhost...

## Five Minute Break

## Continuity Problem

- Session: A user sits down, enters a website, does some work, exits
- HTTP Stateless
  - Does Keep-Alive Help?

## Client vs. Server Side

- Client Side
  - Store Variable=Value Bindings in HTML Page, or Cookies
- Server Side
  - Store Variable=Value Bindings in DB/Server Memory
  - Store Session ID on Client Side, to identify Client

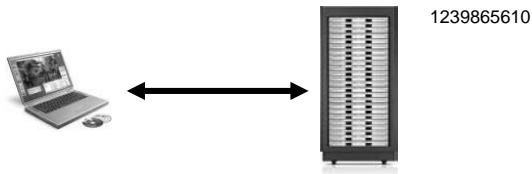
## Three Typical Solutions

- Cookies
- URL Rewriting
- Hidden Fields

## HTTP Cookies Grab-bag

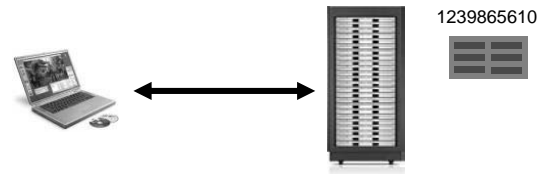
- Lifetime
  - Session – not written to file system
  - Persistent – written to user preferences
- Only returns cookie to requesting domain
- Cookie must be specified by content
- No special characters in cookie

## HTTP Cookies



```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

## HTTP Cookies



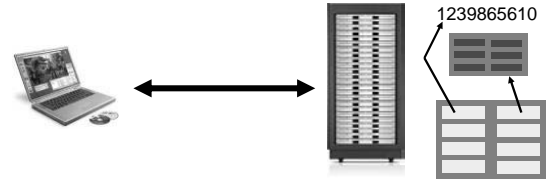
```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

## HTTP Cookies



```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

## HTTP Cookies



```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

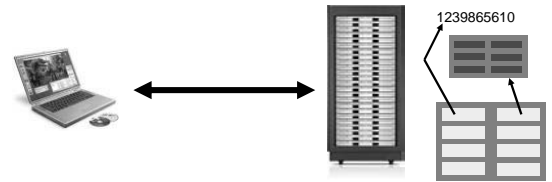
## HTTP Cookies



```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

JSESSIONID → 1239865610

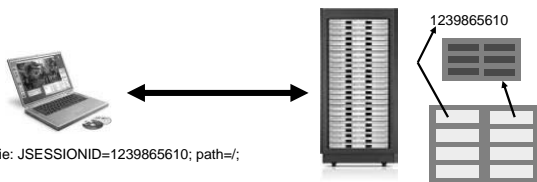
## HTTP Cookies



```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

JSESSIONID → 1239865610  
PATH → /

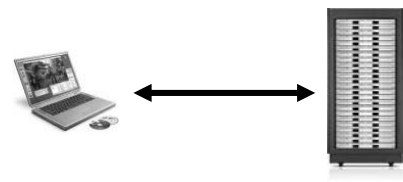
## HTTP Cookies



Set-Cookie: JSESSIONID=1239865610; path=/;

```
String sID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = findTableStoringSessions();
globalTable.put(sID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

## HTTP Cookies



Cookie: JSESSIONID=1239865610;

```
// On request
String sID = request.getCookie("JSESSIONID");
Hashtable globalTable = findTableStoringSessions();
Hashtable sInfo = (Hashtable) globalTable.get(sID);
```

# HTTP Cookies



Cookie: JSESSIONID=1239865610;

```
// On request
String sID = request.getCookie("JSESSIONID");
Hashtable globalTable = findTableStoringSessions();
Hashtable sInfo = (Hashtable) globalTable.get(sID);
```

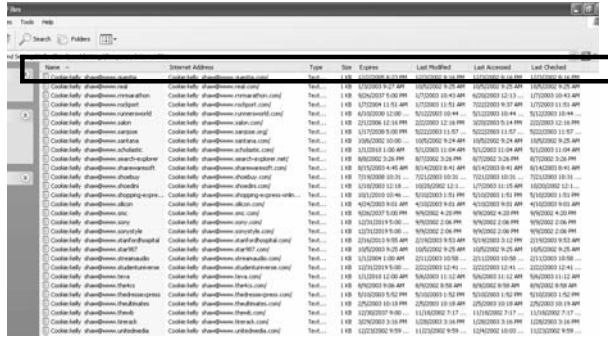
# HTTP Cookies



Cookie: JSESSIONID=1239865610;

```
// On request
String sID = request.getCookie("JSESSIONID");
Hashtable globalTable = findTableStoringSessions();
Hashtable sInfo = (Hashtable) globalTable.get(sID);
```

# In-Browser Cookie Management



# URL Rewriting

- Rewrite all URLs in response to contain SessionID
  - `http://foo.com/servlet/cart?id=123xyz`
- Parse out session ID from request line
- `encodeURL()` in `HttpResponse` object will rewrite session-id onto URL
- Limitations
  - Always include `?sessionId=238423984`
  - e.g. `http://www.amazon.com/exec/obidos/subst/home/home.html/103-0036360-1119059`

# URL Rewriting

```
<TABLE border=0 width=100% cellpadding=1 cellspacing=0 bgcolor=#cccc99 ><TR> <TD width=100%><TABLE border=0 cellpadding=4 cellspacing=0 bgcolor=#cccc99><TR> <TD align=left bgcolor=#eeeeec <small class="small" href=http://www.amazon.com/exec/obidos/account-access-login/ref=pd_ys_h_c/103-4591077-2490203>Your Account</a></b> </TD> </TR> <TR> <TD bgcolor=#ffffff align=top width=100%><table width=100% border=0 cellpadding=2 cellspacing=0> <tr align=top>
```

# Hidden Form Fields

- `<input type="hidden" name="session" value="...">`
- ```
<form method="POST" action="/exec/obidos/handle-buy-box/ref=bp_adq/103-4591077-2490203">
  <input type="hidden" name="colid" value="">
  <input type="hidden" name="template-name" value="">
  <input type="hidden" name="store-name" value="gateway">
  <input type="hidden" name="maw" value="1">
  <input type="hidden" name="coliid" value="">
  <input type="hidden" name="dropdown-selection" value="default-address">
  <table border="0" width="100%" cellspacing="0" cellpadding="6">
```

## Java Servlet Solution

- Session tracking API built on top of URL rewriting or cookies
  - Look up HttpSession object associated with current request (or create new one)
    - All cookie/URL rewriting mechanics hidden
  - Look up information associated with a session
  - Associate information with a session

## Look up Session Info

```
HttpSession session = request.getSession(true);
ShoppingCart sc = (ShoppingCart)
session.getAttribute("shoppingCart");
if (cart == null) {
    cart = new ShoppingCart();
    session.setAttribute("shoppingCart", cart);
}
...
// do something with your shopping cart object
```

## HttpSession Methods

- public String getId()
- public boolean isNew()
- public long getCreationTime()
- public long getLastAccessedTime()
- public int getMaxInactiveInterval()
- public void setMaxInactiveInterval(int secs)
- public void invalidate()

## Associate Info w/ Session

```
HttpSession session = request.getSession(true);
session.setAttribute("referringPage",
    request.getHeader("Referer"));

ShoppingCart cart =
    (ShoppingCart)session.getAttribute("previousItems");

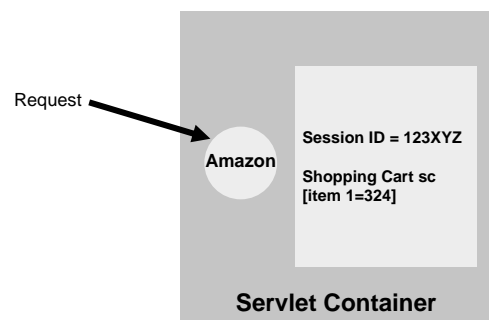
if (cart == null) {
    cart = new ShoppingCart();
    session.setAttribute("previousItems", cart);
}

String itemID = request.getParameter("itemID");
if (itemID != null) {
    cart.addItem(Catalog.getItem(itemID));
}
```

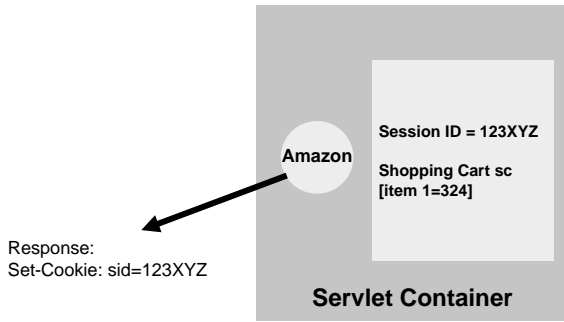
## Session Termination

- Automatic! After a long enough interval (getMaxInactiveInterval)

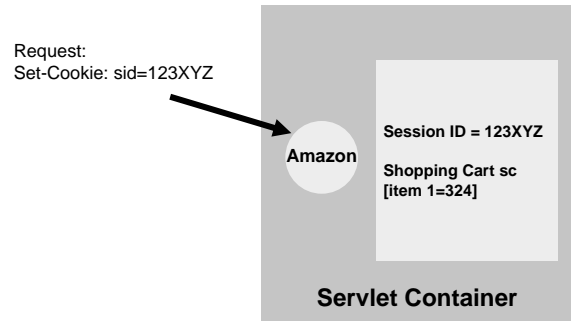
## Session Tracking



## Session Tracking



## Session Tracking



## Session Tracking

