Session 4 – Web Session

Agenda - Handling Cookies

- **The potential of cookies**
- **The problems with cookies**
- **Sending cookies to browser**
- **Reading cookies from browser**
- **Simple cookie-handling servlets**
- **Cookie utilities**
- **Methods in the Cookie API**
- **A customized search engine front end**

The Potential of Cookies

- **Idea**
  - Servlet sends a simple name and value to client.
  - Client returns same name and value when it connects to same site (or same domain, depending on cookie settings).
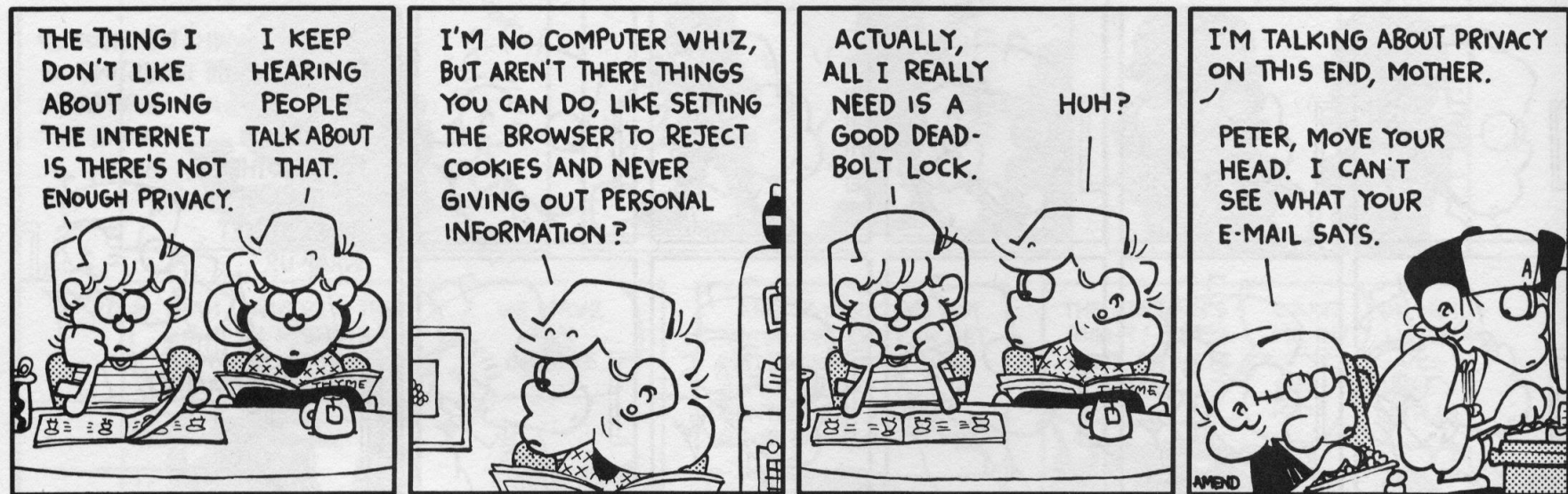
- **Typical Uses of Cookies**
  - Identifying a user during an e-commerce session
    - Servlets have a higher-level API for this task
  - Avoiding username and password
  - Customizing a site
  - Focusing advertising

Cookies and Focused Advertising
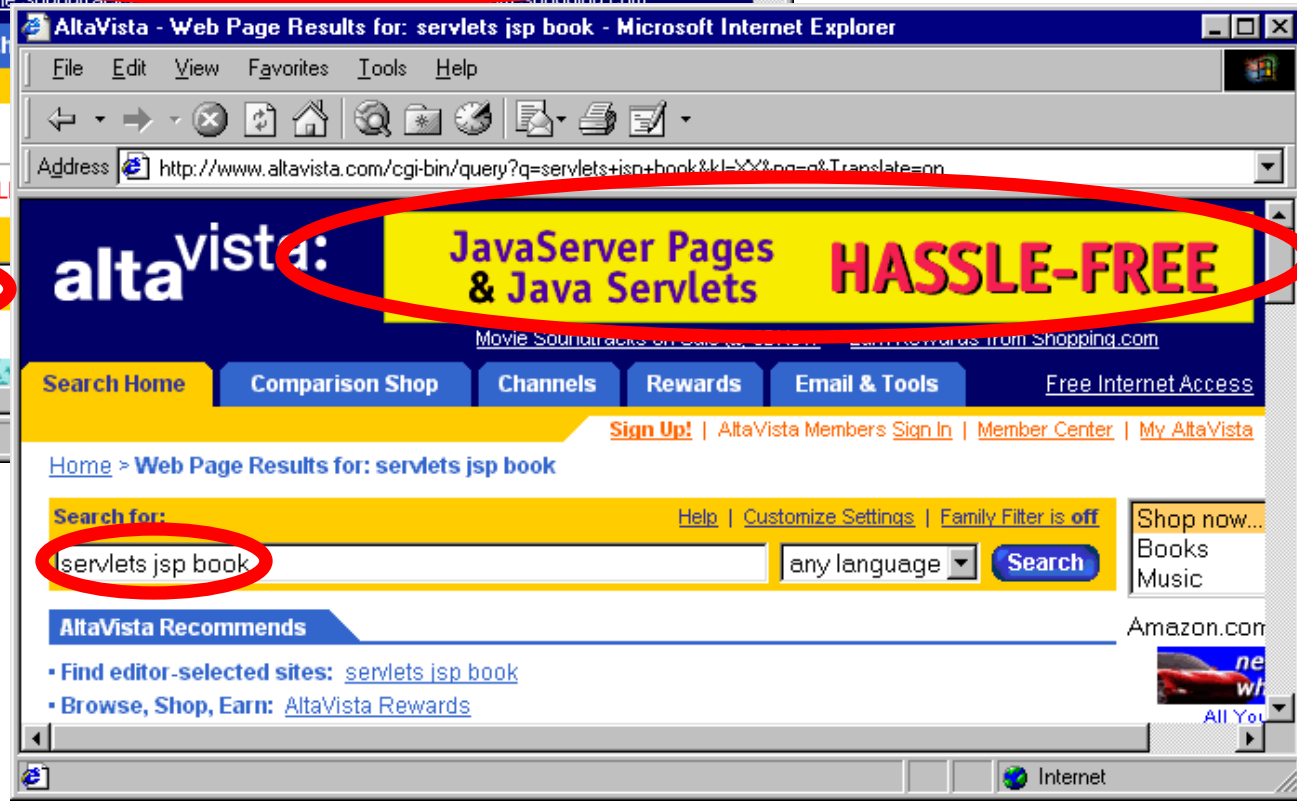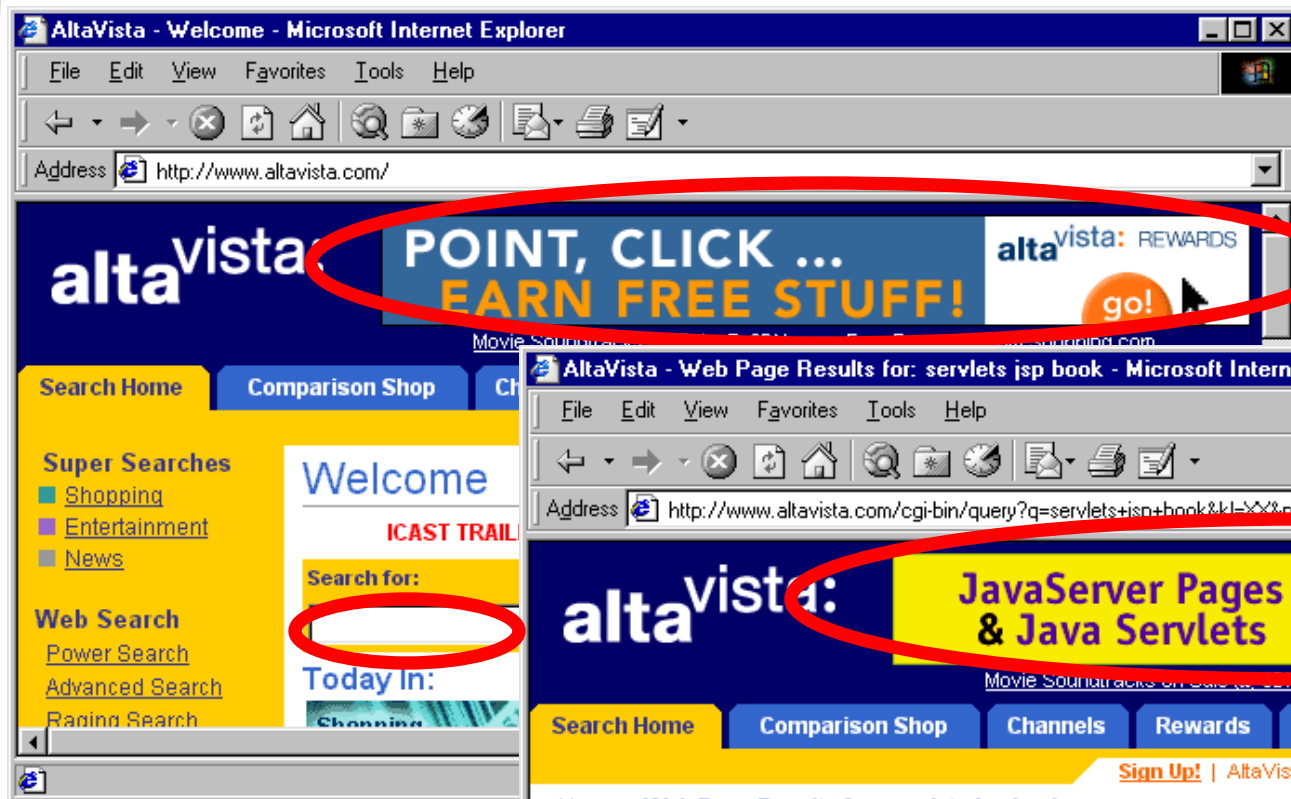
Cookies and Privacy

FoxTrot © 1998 Bill Amend. Reprinted with permission of Universal Press Syndicate. All rights reserved.

Some Problems with Cookies

- **The problem is privacy, not security.**
  - Servers can remember your previous actions
  - If you give out personal information, servers can link that information to your previous actions
  - Servers can share cookie information through use of a cooperating third party like doubleclick.net
  - Poorly designed sites store sensitive information like credit card numbers directly in cookie
  - JavaScript bugs let hostile sites steal cookies (old browsers)
- **Moral for servlet authors**
  - If cookies are not critical to your task, avoid servlets that totally fail when cookies are disabled
  - Don't put sensitive info in cookies

Sending Cookies to Browser

- **Standard approach:**

```
Cookie c = new Cookie("name", "value");
c.setMaxAge(...); // Means cookie persists on disk
// Set other attributes.
response.addCookie(c);
```

- **Simplified approach:**

  – Use LongLivedCookie class:

```
public class LongLivedCookie extends Cookie {
  public static final int SECONDS_PER_YEAR =
    60*60*24*365;

  public LongLivedCookie(String name, String value) {
    super(name, value);
    setMaxAge(SECONDS_PER_YEAR);
  }
}
```

Reading Cookies from Browser

- ## Standard approach:

```java
Cookie[] cookies = request.getCookies();
if (cookies != null) {
  for(int i=0; i<cookies.length; i++) {
    Cookie c = cookies[i];
    if (c.getName().equals("someName")) {
      doSomethingWith(c);
      break;
    }
  }
}
```
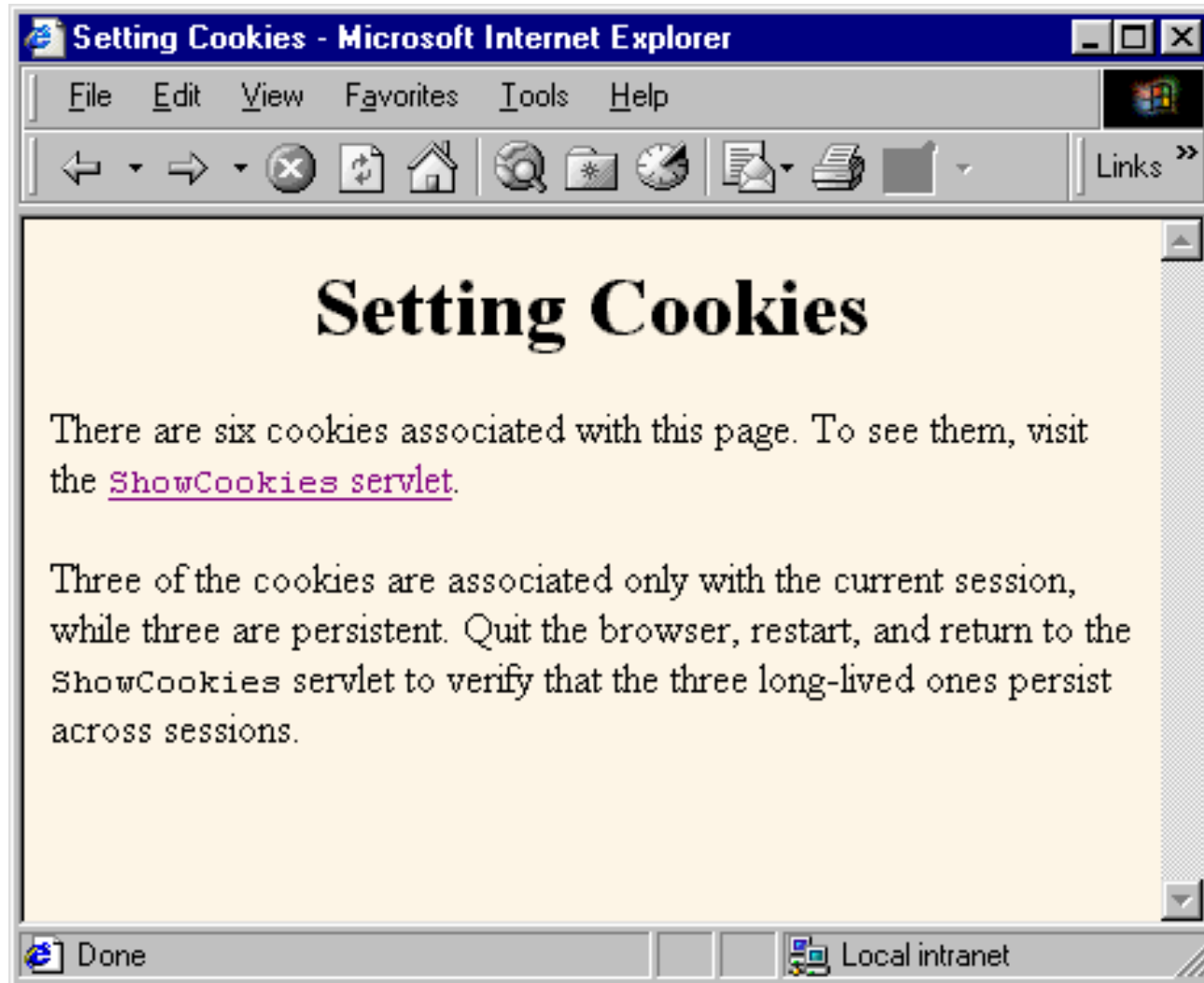
- ## Simplified approach:

  - Extract cookie or cookie value from cookie array by using ServletUtilities.getCookieValue or ServletUtilities.getCookie

Simple Cookie-Setting Servlet

```java
public class SetCookies extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException {
    for(int i=0; i<3; i++) {
      Cookie cookie = new Cookie("Session-Cookie-" + i,
                                "Cookie-Value-S" + i);
      response.addCookie(cookie);
      cookie = new Cookie("Persistent-Cookie-" + i,
                          "Cookie-Value-P" + i);
      cookie.setMaxAge(3600);
      response.addCookie(cookie);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println(...);
```

Web Programming – IU – IT – 2014

Result of Cookie-Setting Servlet

Setting Cookies - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Links »

Setting Cookies

There are six cookies associated with this page. To see them, visit the ShowCookies servlet.

Three of the cookies are associated only with the current session, while three are persistent. Quit the browser, restart, and return to the ShowCookies servlet to verify that the three long-lived ones persist across sessions.

Done   Local intranet

Simple Cookie-Viewing Servlet

```java
public class ShowCookies extends HttpServlet {
  public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
     throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Active Cookies";
    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>Cookie Name\n" +
              "  <TH>Cookie Value");
```

Simple Cookie-Viewing Servlet (Continued)

```java
Cookie[] cookies = request.getCookies();
if (cookies != null) {
  Cookie cookie;
  for(int i=0; i<cookies.length; i++) {
    cookie = cookies[i];
    out.println("<TR>\n" +
                "  <TD>" + cookie.getName() + "\n" +
                "  <TD>" + cookie.getValue());
  }
}
out.println("</TABLE></BODY></HTML>");
  }
}
```

Result of Cookie-Viewer (Before & After Restarting Browser)

Cookie Utilities

- **Problem**
  - getCookies returns an array of cookies
  - You almost always only care about one particular cookie
- **Solution**
  - Static methods to
    - Extract a cookie value given a cookie name (default value if no match)
    - Extract a Cookie object given a cookie name (null if no match)

ServletUtilities.getCookieValue

```java
public static String getCookieValue(Cookie[] cookies,
                                    String cookieName,
                                    String defaultVal) {
  if (cookies != null) {
    for(int i=0; i<cookies.length; i++) {
      Cookie cookie = cookies[i];
      if (cookieName.equals(cookie.getName()))
        return(cookie.getValue());
    }
  }
  return(defaultVal);
}
```

ServletUtilities.getCookie

```java
public static Cookie getCookie(Cookie[] cookies,
                               String cookieName) {
  if (cookies != null) {
    for(int i=0; i<cookies.length; i++) {
      Cookie cookie = cookies[i];
      if (cookieName.equals(cookie.getName()))
        return(cookie);
    }
  }
  return(null);
}
```

Methods in the Cookie API

- ## getDomain/setDomain
  - Lets you specify domain to which cookie applies. Current host must be part of domain specified.

- ## getMaxAge/setMaxAge
  - Gets/sets the cookie expiration time (in seconds). If you fail to set this, cookie applies to current browsing session only. See LongLivedCookie helper class given earlier.

- ## getName
  - Gets the cookie name. There is no setName method; you supply name to constructor. For incoming cookie array, you use getName to find the cookie of interest.

Methods in the Cookie API (Continued)

- **getPath/setPath**
  - Gets/sets the path to which cookie applies. If unspecified, cookie applies to URLs that are within or below directory containing current page.

- **getSecure/setSecure**
  - Gets/sets flag indicating whether cookie should apply only to SSL connections or to all connections.

- **getValue/setValue**
  - Gets/sets value associated with cookie. For new cookies, you supply value to constructor, not to setValue. For incoming cookie array, you use getName to find the cookie of interest, then call getValue on the result. If you set the value of an incoming cookie, you still have to send it back out with response.addCookie.

A Customized Search Engine Interface

- **Front end remembers settings for search engine, search string, and hits per page**

  – Front end *uses* cookies

  – Back end *sets* cookies

  – In real life, don't really show previous queries!

Web Session

Front End to SearchEngines Servlet

```java
public class SearchEnginesFrontEnd extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException {
    Cookie[] cookies = request.getCookies();
    String searchString =
      ServletUtilities.getCookieValue(cookies,
                                      "searchString",
                                      "Java Programming");
    String numResults =
      ServletUtilities.getCookieValue(cookies,
                                      "numResults",
                                      "10");
    String searchEngine =
      ServletUtilities.getCookieValue(cookies,
                                      "searchEngine",
                                      "google");
```

Front End to SearchEngines Servlet (Continued)

```
...
out.println
  (...
   "<FORM ACTION=\"/servlet/" +
     "coreservlets.CustomizedSearchEngines\">\n" +
   "<CENTER>\n" +
   "Search String:\n" +
   "<INPUT TYPE=\"TEXT\" NAME=\"searchString\"\n" +
   "       VALUE=\"" + searchString + "\"><BR>\n" +
   "Results to Show Per Page:\n" +
   "<INPUT TYPE=\"TEXT\" NAME=\"numResults\"\n" +
   "       VALUE=" + numResults + " SIZE=3><BR>\n" +
   "<INPUT TYPE=\"RADIO\" NAME=\"searchEngine\"\n" +
   "       VALUE=\"google\"" +
   checked("google", searchEngine) + ">\n" +
   ...);
```

Customized SearchEngines Servlet (Back End)

```java
public class CustomizedSearchEngines extends HttpServlet {
  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;
    }
    Cookie searchStringCookie =
      new LongLivedCookie("searchString", searchString);
    response.addCookie(searchStringCookie);
    ...
  }
}
```

Summary

- **Cookies involve name/value pairs sent from server to browser and returned when the same page, site, or domain is visited later**

- **Let you**
  - Track sessions (use higher-level API)
  - Permit users to avoid logging in at low-security sites
  - Customize sites for different users
  - Focus content or advertising

- **Setting cookies**
  - Call Cookie constructor, set age, call response.addCookie

- **Reading cookies**
  - Call request.getCookies, check for null, look through array for matching name, use associated value

Agenda - Session Tracking

- **The purpose of session tracking**
- **Rolling your own session tracking**
- **The session tracking API**
- **Per-client access counts**
- **Shopping carts and online stores**

Session Tracking and E-Commerce

- ## Why session tracking?
  - When clients at on-line store add item to their shopping cart, how does server know what's already in cart?
  - When clients decide to proceed to checkout, how can server determine which previously created cart is theirs?
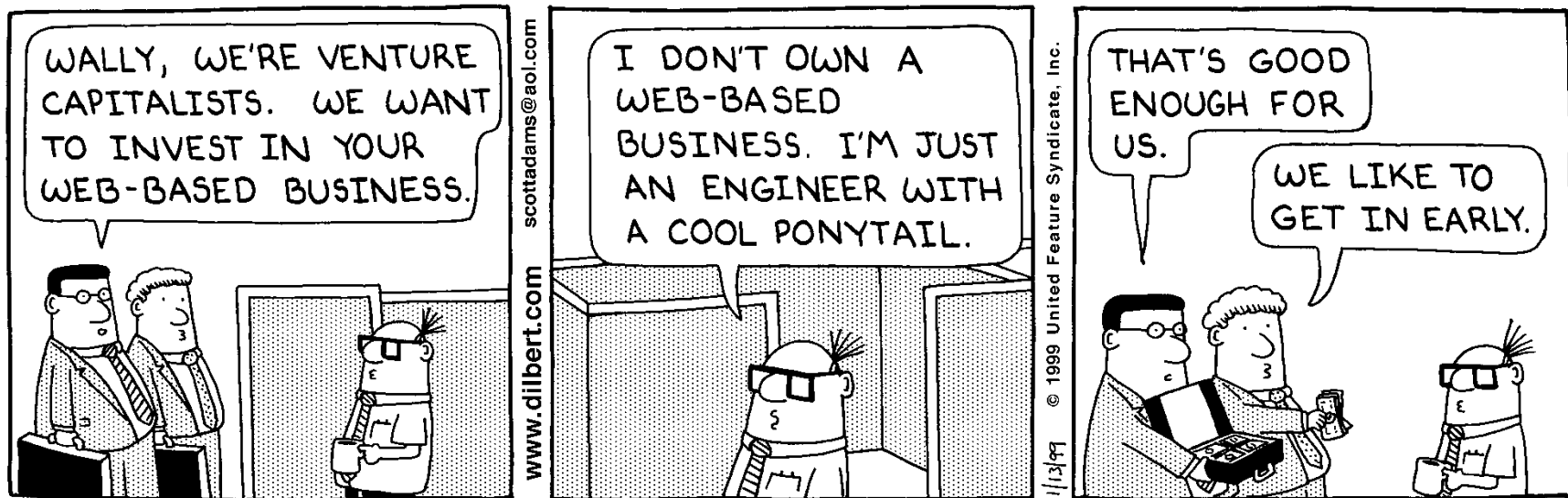
Dilbert used with permission of United Syndicates Inc.

Rolling Your Own Session Tracking: Cookies

- ## Idea: associate cookie with data on server

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

- ## Still to be done:

  – Extracting cookie that stores session identifier
  – Setting appropriate expiration time for cookie
  – Associating the hash tables with each request
  – Generating the unique session identifiers

Rolling Your Own Session Tracking: URL-Rewriting

- **Idea**
  - Client appends some extra data on the end of each URL that identifies the session
  - Server associates that identifier with data it has stored about that session
  - E.g., http://host/path/file.html;jsessionid=1234
- **Advantage**
  - Works even if cookies are disabled or unsupported
- **Disadvantages**
  - Lots of tedious processing
  - Must encode all URLs that refer to your own site
  - Links from other sites and bookmarks can fail

Rolling Your Own Session Tracking: Hidden Form Fields

- **Idea:**

  ```
  <INPUT TYPE="HIDDEN" NAME="session" VALUE="...">
  ```

- **Advantage**
  - Works even if cookies are disabled or unsupported

- **Disadvantages**
  - Lots of tedious processing
  - All pages must be the result of form submissions

The Session Tracking API

- **Session objects live on the server**
- **Automatically associated with client via cookies or URL-rewriting**
  - Use request.getSession(true) to get either existing or new session
    - Behind the scenes, the system looks at cookie or URL extra info and sees if it matches the key to some previously stored session object. If so, it returns that object. If not, it creates a new one, assigns a cookie or URL info as its key, and returns that new session object.
- **Hashtable-like mechanism lets you store arbitrary objects inside session**
  - setAttribute (putValue in 2.1) stores values
  - getAttribute (getValue in 2.1) retrieves values

Accessing Session Data

```
HttpSession session = request.getSession(true);
ShoppingCart cart =
   (ShoppingCart)session.getAttribute("shoppingCart");
if (cart == null) { // No cart already in session
  cart = new ShoppingCart();
  session.setAttribute("shoppingCart", cart);
}
doSomethingWith(cart);
```

HttpSession Methods

- **getAttribute (getValue in old servlet spec 2.1)**
  - Extracts a previously stored value from a session object. Returns null if no value is associated with given name.
- **setAttribute (putValue in ver. 2.1)**
  - Associates a value with a name. Monitor changes: values implement HttpSessionBindingListener.
- **removeAttribute (removeValue in ver. 2.1)**
  - Removes values associated with name.
- **getAttributeNames (getValueNames in 2.1)**
  - Returns names of all attributes in the session.
- **getId**
  - Returns the unique identifier.

HttpSession Methods (Continued)

- **isNew**
  - Determines if session is new to *client* (not to *page*)
- **getCreationTime**
  - Returns time at which session was first created
- **getLastAccessedTime**
  - Returns time at which session was last sent from client
- **getMaxInactiveInterval, setMaxInactiveInterval**
  - Gets or sets the amount of time session should go without access before being invalidated
- **invalidate**
  - Invalidates the session and unbinds all objects associated with it

A Servlet Showing Per-Client Access Counts

```java
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  String title = "Session Tracking Example";
  HttpSession session = request.getSession(true);
  String heading;
  Integer accessCount =
    (Integer)session.getAttribute("accessCount");
  if (accessCount == null) {
    accessCount = new Integer(0);
    heading = "Welcome, Newcomer";
  } else {
    heading = "Welcome Back";
    accessCount = new Integer(accessCount.intValue() + 1);
  }
  session.setAttribute("accessCount", accessCount);
```

First Visit to ShowSession Servlet

Session Tracking Example - Netscape

File   Edit   View   Go   Communicator   Help

Welcome, Newcomer

Information on Your Session:

Info Type	Value
ID	To1010mC6608409833952431At
Creation Time	Mon Apr 16 10:41:25 EDT 2001
Time of Last Access	Mon Apr 16 10:41:25 EDT 2001
Number of Previous Accesses	0

Document: Done
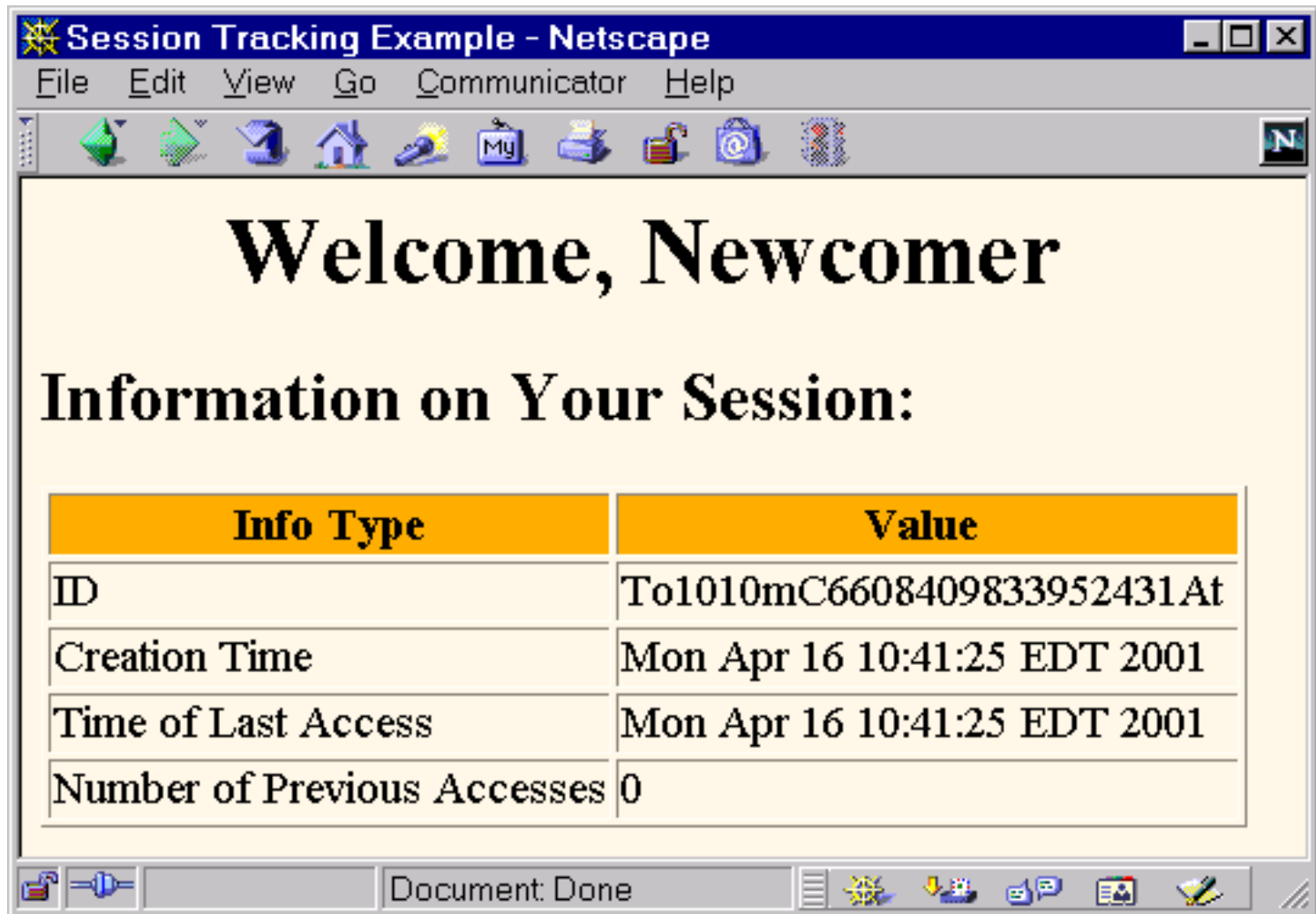
Eleventh Visit to ShowSession Servlet

Session Tracking and Shopping Carts
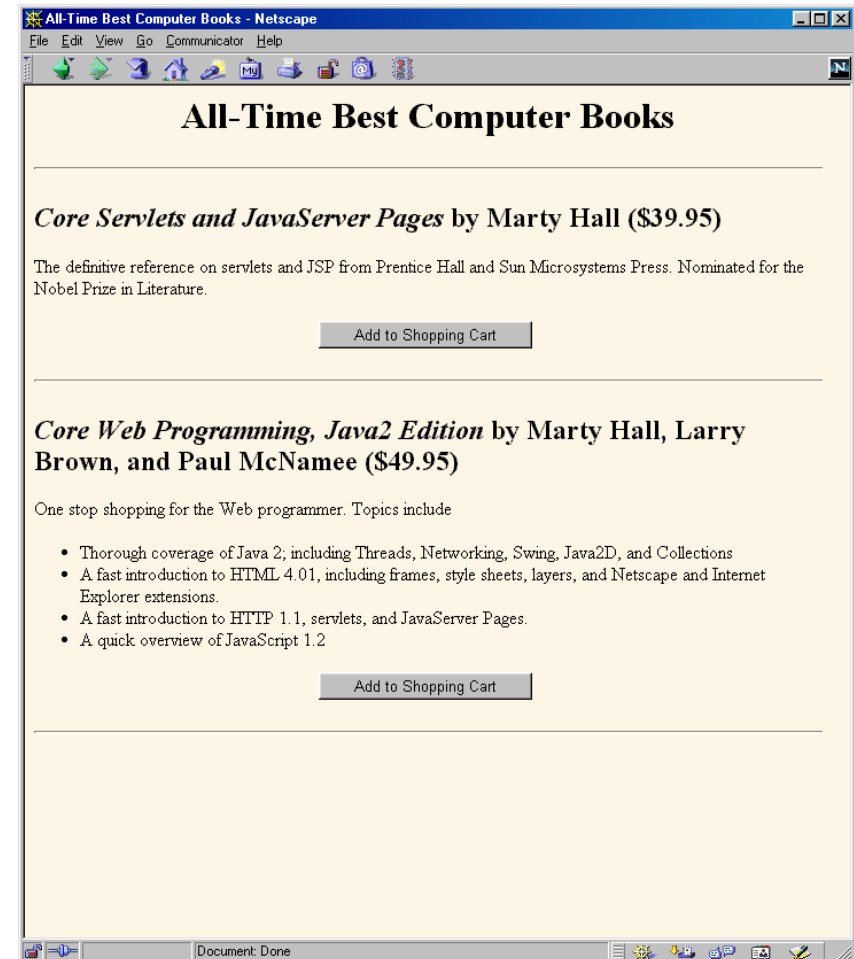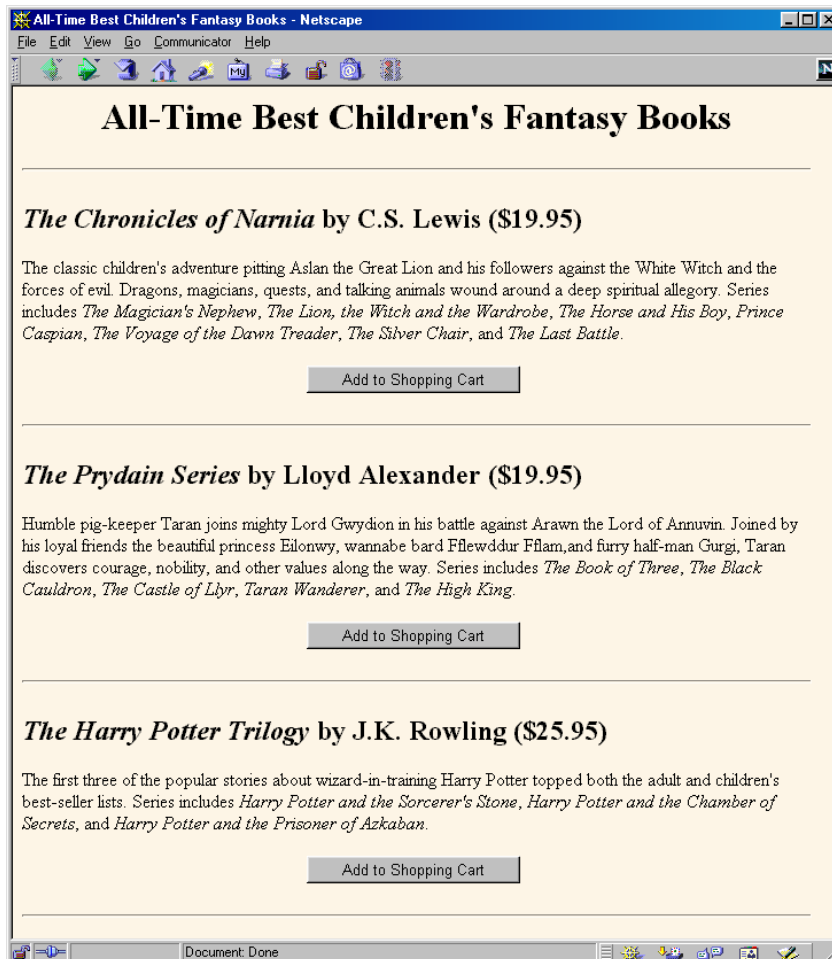
Session Tracking and Shopping Carts (Continued)

What Changes if Server Uses URL Rewriting?

- **Session tracking code:**
  - No change
- **Code that generates hypertext links back to same site:**
  - Pass URL through response.encodeURL.
    - If server is using cookies, this returns URL unchanged
    - If server is using URL rewriting, this appends the session info to the URL
    - E.g.:
      ```
      String url = "order-page.html";
      url = response.encodeURL(url);
      ```
- **Code that does sendRedirect to same site:**
  - Pass URL through response.encodeRedirectURL

Summary

- **Although it usually uses cookies behind the scenes, the session tracking API is higher-level and easier to use than the cookie API**
  - If server supports URL-rewriting, *your* code unchanged
- **Session information lives on server**
  - Cookie or extra URL info associates it with a user
- **Obtaining session**
  - request.getSession(true)
- **Associating values with keys**
  - session.setAttribute (or session.putValue)
- **Finding values associated with keys**
  - session.getAttribute (or session.getValue)
    - Always check if this value is null