

Core Servlets and JavaServer Pages / 2e
Volume 1: Core Technologies
Marty Hall • Larry Brown

Handling Cookies

Agenda

- **Understanding the benefits and drawbacks of cookies**
- **Sending outgoing cookies**
- **Receiving incoming cookies**
- **Tracking repeat visitors**
- **Specifying cookie attributes**
- **Differentiating between session cookies and persistent cookies**
- **Simplifying cookie usage with utility classes**
- **Modifying cookie values**
- **Remembering user preferences**

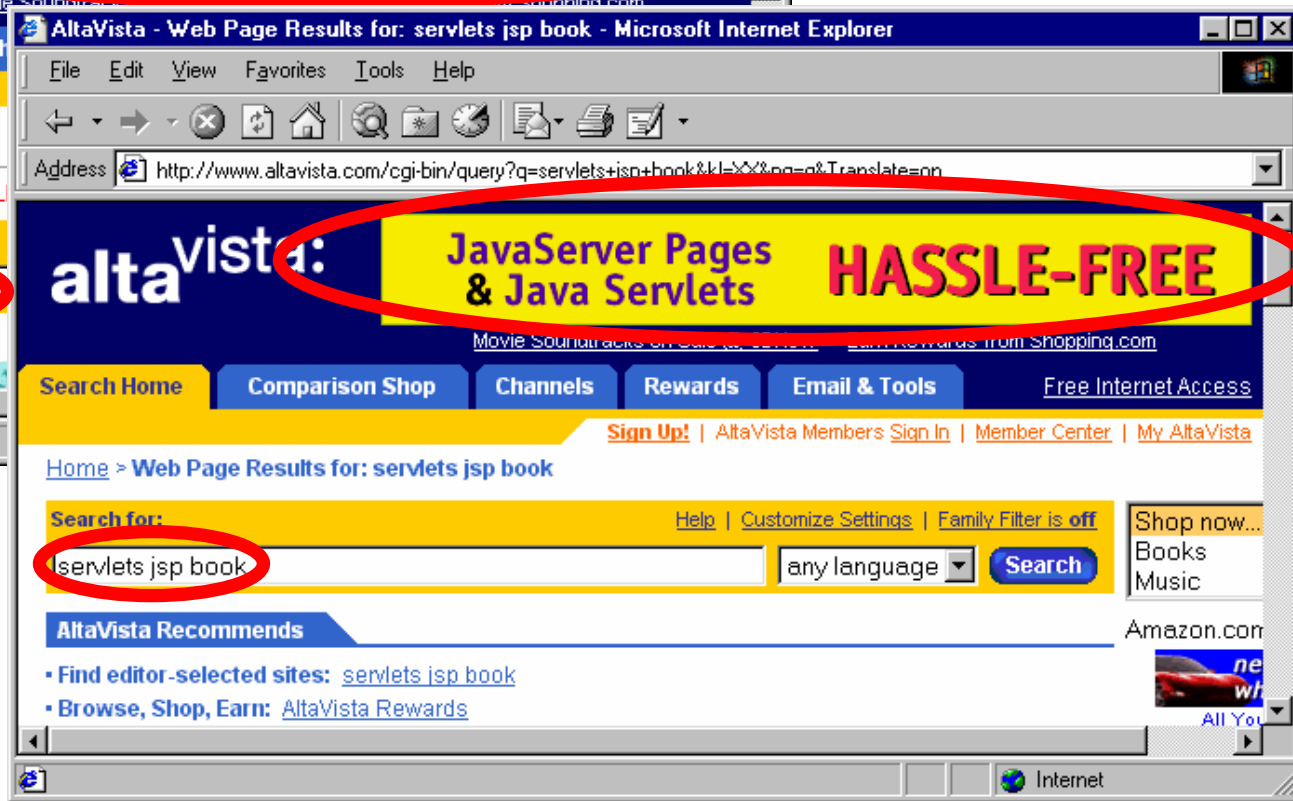
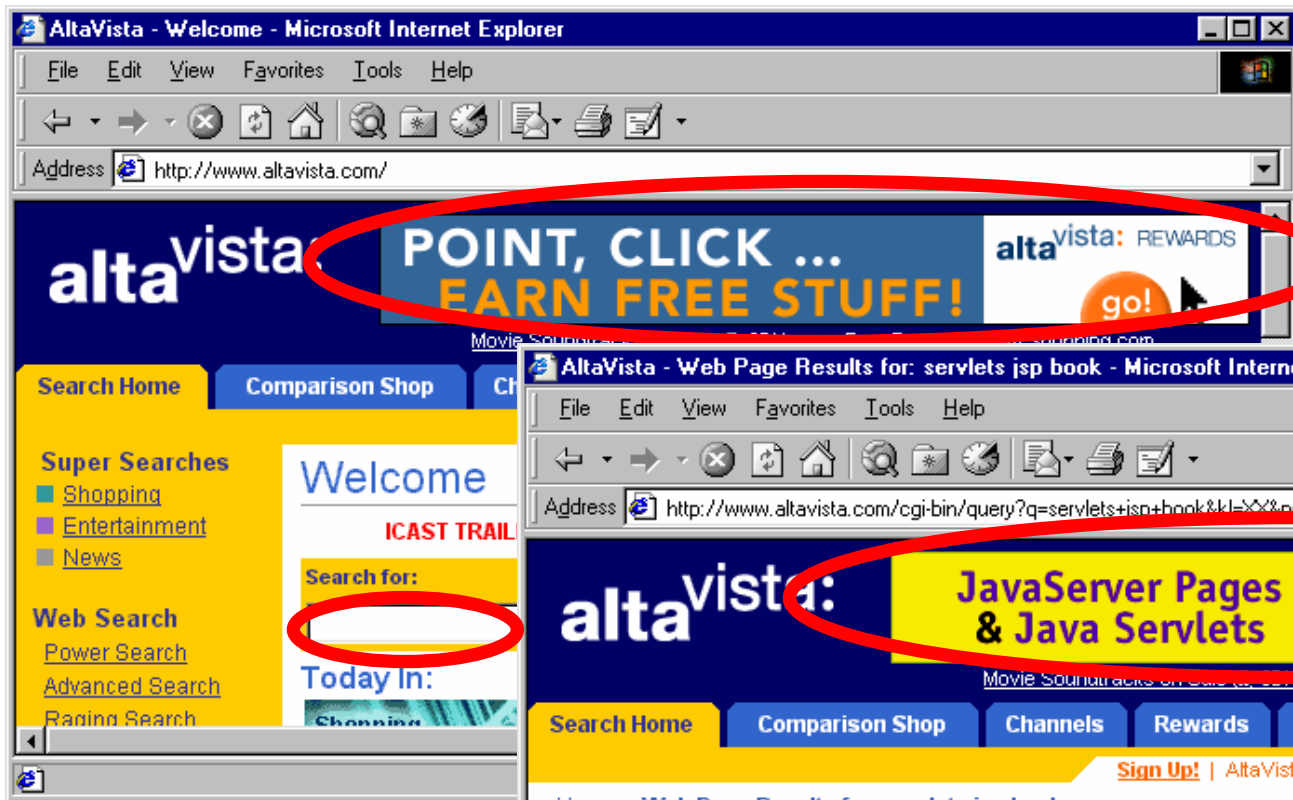
The Potential of Cookies

- **Cookies are small bits of textual information that a Web server sends to a browser and that the browser later returns unchanged when visiting the same Web site**
- **Typical Uses of Cookies**
 - Identifying a user during an e-commerce session and remembering items selected despite the fact that the HTTP connection is usually closed after each page is sent
 - Servlets have a higher-level API for session tracking
 - Remembering username and password on unshared computers
 - When a user registers at a site, a cookie containing a unique user ID is sent to him
 - When the client reconnects at a later date, the user ID is returned automatically, the server looks it up and determines it belongs to a registered user who prefers to user autologin.
 - Access is permitted without an explicit username and password

The Potential of Cookies

- Customizing a site based on user preferences
 - Select what you want to see on a website (weather, stocks, sports, etc.) and how and where it should be displayed
 - Settings could be stored in the cookie or in a server side database based on a unique client identifier
- Focusing advertising by remembering what interests a user
 - Advertisers are willing to pay more for advertisements that are shown to people who are interested in them
 - Cookies provide the ability to remember previous searches

Cookies and Focused Advertising



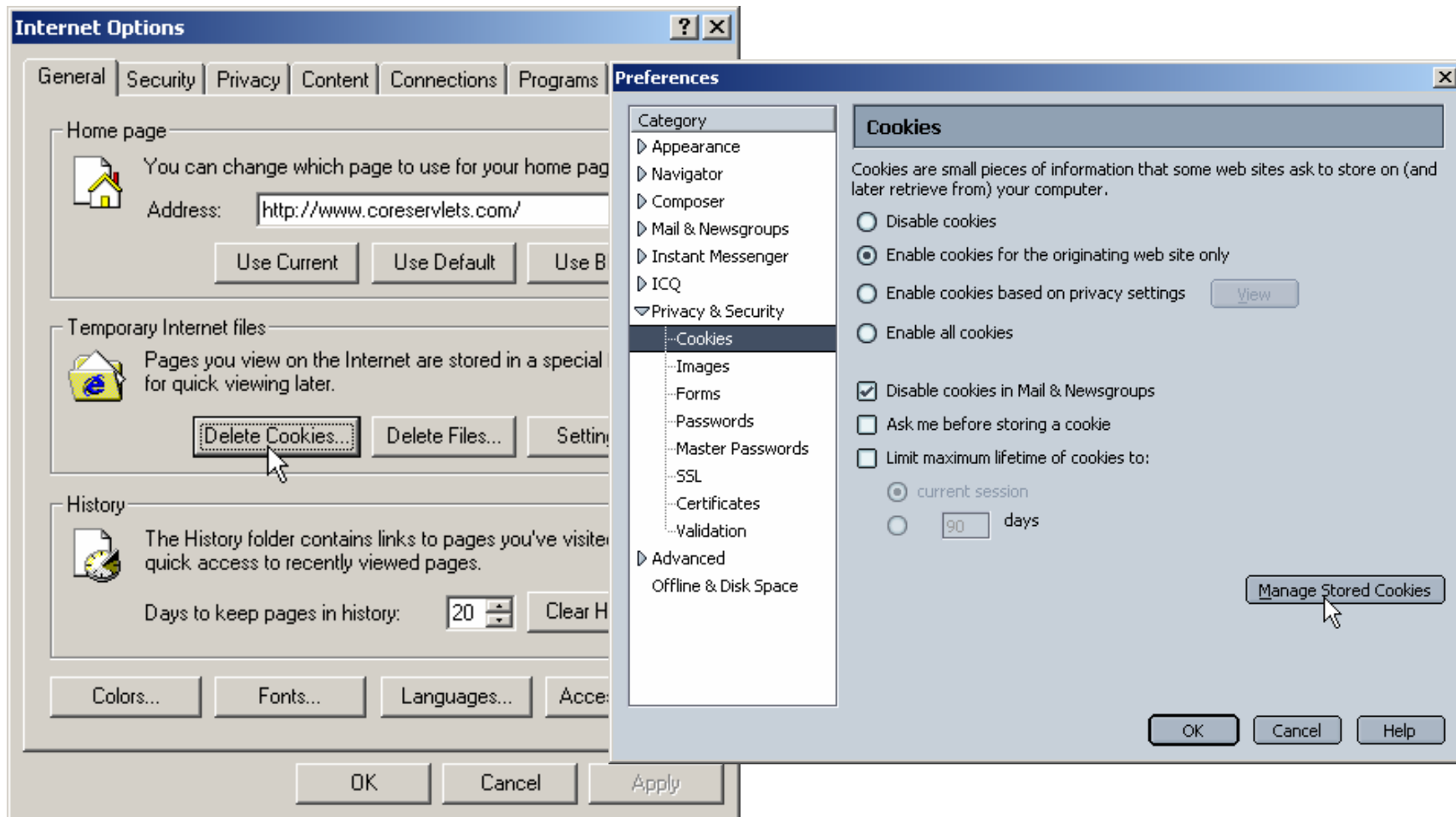
Cookies Not A Security Threat

- **Cookies can not be used to insert viruses or attack the computer**
 - Cookies are never interpreted or executed in any way so they
- **Cookies can not fill up a hard drive**
 - Browsers can limit how many cookies per site are accepted and how many total cookies it stores. Also can limit the size of a cookie

Privacy Is A Problem

- **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
 - An image (with an attached cookie) sent with an e-mail can identify you if you visit their website at a later time
 - Poorly designed sites store sensitive information like credit card numbers directly in cookie
- **Moral for servlet authors**
 - As some users turn off cookies, avoid servlets that totally fail when cookies are disabled if cookies are not critical to your task,
 - Don't put sensitive info in cookies

Manually Deleting Cookies (To Simplify Testing)



Sending Cookies to the Client

- **To send cookies to a client, a servlet should**
 - use the `Cookie` constructor to create one or more cookies with designated names and values
 - set any optional attributes
 - insert the cookies into the HTTP response headers with `response.addCookie`
- **To read incoming cookies, a servlet should**
 - Call `request.getCookies`
 - Loop through the array calling `getName` on each cookie until it finds the one it is looking for
 - Call `getValue` on that cookie to see the associated values

Sending Cookies to the Client

- **Create a Cookie object.**

- Call the Cookie constructor with a cookie name and a cookie value, both of which are strings.
- Special characters not allowed in either string

```
Cookie c = new Cookie("userID", "a1234");
```

- **Set the maximum age.**

- By default, a cookie is session-level - stored in the browser's memory and deleted when the user quits the browser
- To tell browser to store cookie on disk instead of just in memory, use `setMaxAge` (argument is in seconds)

```
c.setMaxAge(60*60*24*7); // One week
```

- **Place the Cookie into the HTTP response**

- Use `response.addCookie` before any other content is sent to the client
- If you forget this step, no cookie is sent to the browser!

```
response.addCookie(c);
```

Reading Cookies from the Client

- **Call request.getCookies**
 - This yields an array of **Cookie** objects.
- **Loop down the array, calling getName on each entry until you find the cookie of interest**
 - Use the value (getValue) in application-specific way.

```
String cookieName = "userID";
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for(int i=0; i<cookies.length; i++) {
        Cookie cookie = cookies[i];
        if (cookieName.equals(cookie.getName())) {
            doSomethingWith(cookie.getValue());
        }
    }
}
```

Using Cookies to Detect First-Time Visitors

```
public class RepeatVisitor extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        boolean newbie = true;
        Cookie[] cookies = request.getCookies();
        if (cookies != null) {
            for(int i=0; i<cookies.length; i++) {
                Cookie c = cookies[i];
                if ((c.getName().equals("repeatVisitor"))&&
                    (c.getValue().equals("yes"))) {
                    newbie = false;
                    break;
                }
            }
        }
    }
}
```

Using Cookies to Detect First-Time Visitors

- **A cookie enables you to differentiate between first-time users and repeat visitors and to display different information based on the type of user**
 - Check for the name of a uniquely named cookie
 - If present, user is repeat visitor
 - Not present, first-time user
- **You can check if the `Cookie[]` array is not null, but that does not necessarily tell you that the visitor is new**
 - It shows the client has been to your site but not that they have been to your servlet
 - Other servlets, JSP pages or non-Java Web applications can set cookies which can be returned to your browser

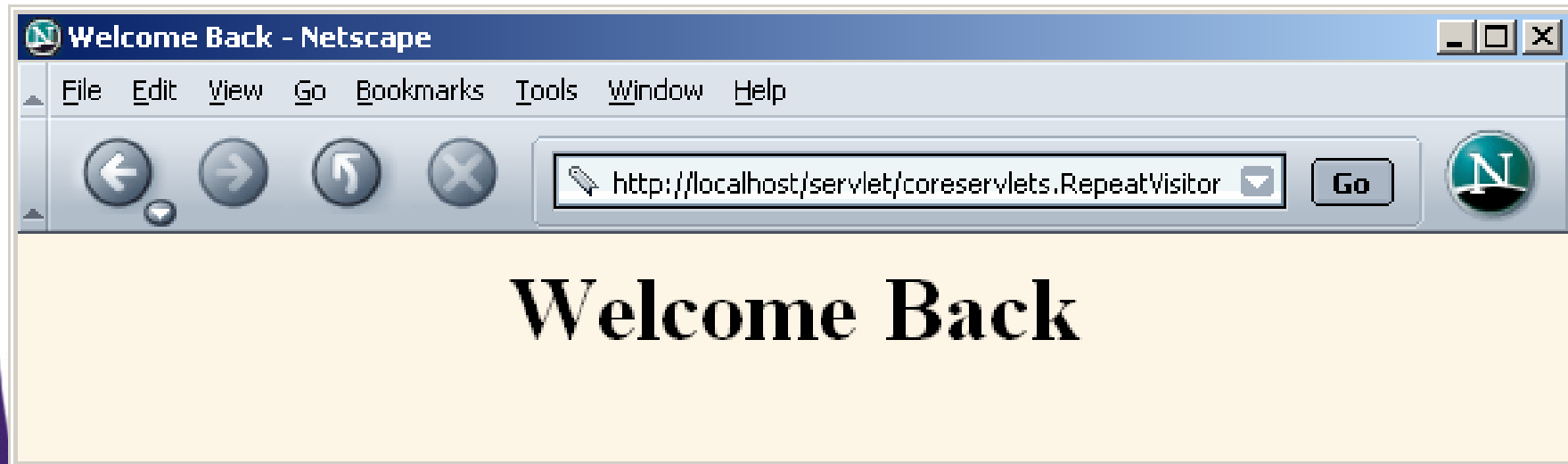
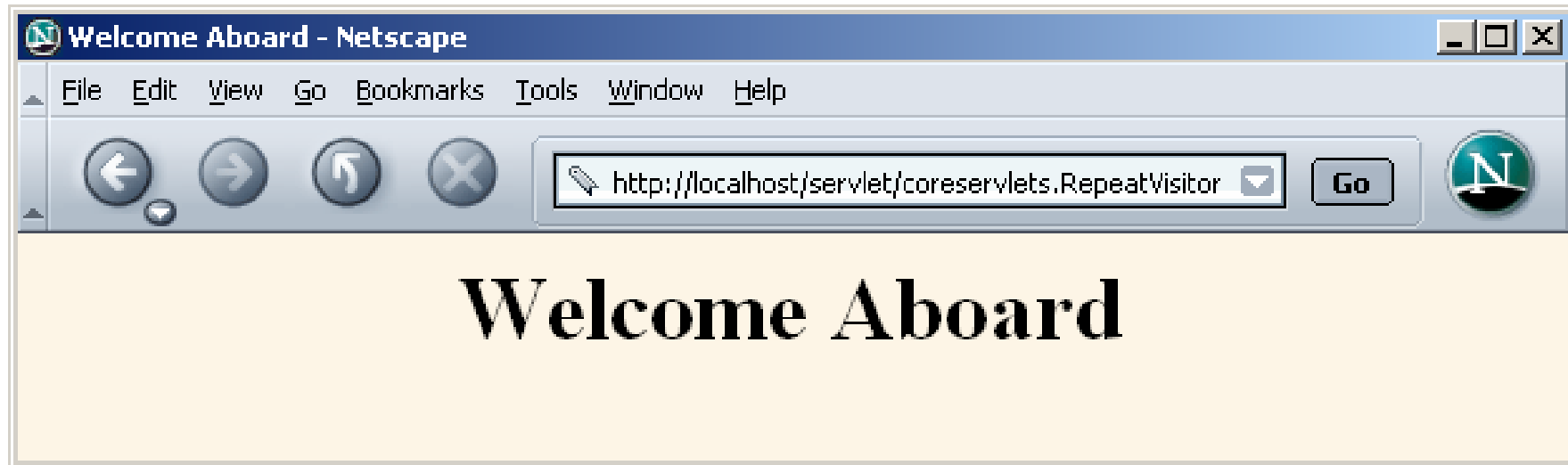
Using Cookies to Detect First-Time Visitors

```
boolean newbie = true;
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for(int i=0; i<cookies.length; i++) {
        Cookie c = cookies[i];
        // Could omit test and treat cookie name as a flag
        if ((c.getName().equals("repeatVisitor")) &&
            (c.getValue().equals("yes"))) {
            newbie = false;
            break;
        }
    }
}
```

Using Cookies to Detect First-Time Visitors

```
String title;
if (newbie) {
    Cookie returnVisitorCookie =
        new Cookie("repeatVisitor", "yes");
    returnVisitorCookie.setMaxAge(60*60*24*365);
    response.addCookie(returnVisitorCookie);
    title = "Welcome Aboard";
} else {
    title = "Welcome Back";
}
response.setContentType("text/html");
PrintWriter out = response.getWriter();
... // (Output page with above title)
```

Using Cookies to Detect First-Time Visitors (Results)



Using Cookie Attributes

- **Before adding the cookie to the outgoing header, you can set various properties of the cookie by using the setXxx methods**
- **Each SetXxx method has a corresponding getXxx method to retrieve the attribute value**
 - Note that the attributes are part of the header sent from the server to the browser, they are not part of the header returned by the browser to the servers
 - Except for name and value, the cookie attributes apply only to outgoing cookies from the server to the client; they are not set on cookies that come from the browser to the server
 - Each time you want to send a cookie you have to do addCookie; you won't find it in the incoming array on the next request even if you set the maxAge to a large value

Using Cookie Attributes

- **getDomain/setDomain**
 - Lets you specify domain to which cookie applies. Normally, the browser returns cookies only to the exact same hostname that sent the cookies. 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.
 - Negative value means the cookie will last only for the current browsing session (i.e., until the user quits the browser)
 - A value of zero instructs the browser to delete the cookie
- **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.

Using Cookie Attributes

- **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.
- To specify that a cookie apply to all URLs on your site, use `cookie.setPath("/")`

- **getSecure/setSecure**

- Gets/sets flag indicating whether cookie should apply only to SSL connections or to all connections; default is false

- **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`.

Differentiating Session Cookies from Persistent Cookies

- **The following example sets six outgoing cookies.**
 - Three have no explicit age and apply only in the current browsing session i.e., until the users restarts the browser
 - The other three use `setMaxAge` so that they will be written to disk and persist for the next hour regardless of whether or not the user restarts the browser or reboots the computer
 - Then the servlet uses `request.getCookies` to find all the incoming cookies and display their names and values in the browser window

Differentiating Session Cookies from Persistent Cookies

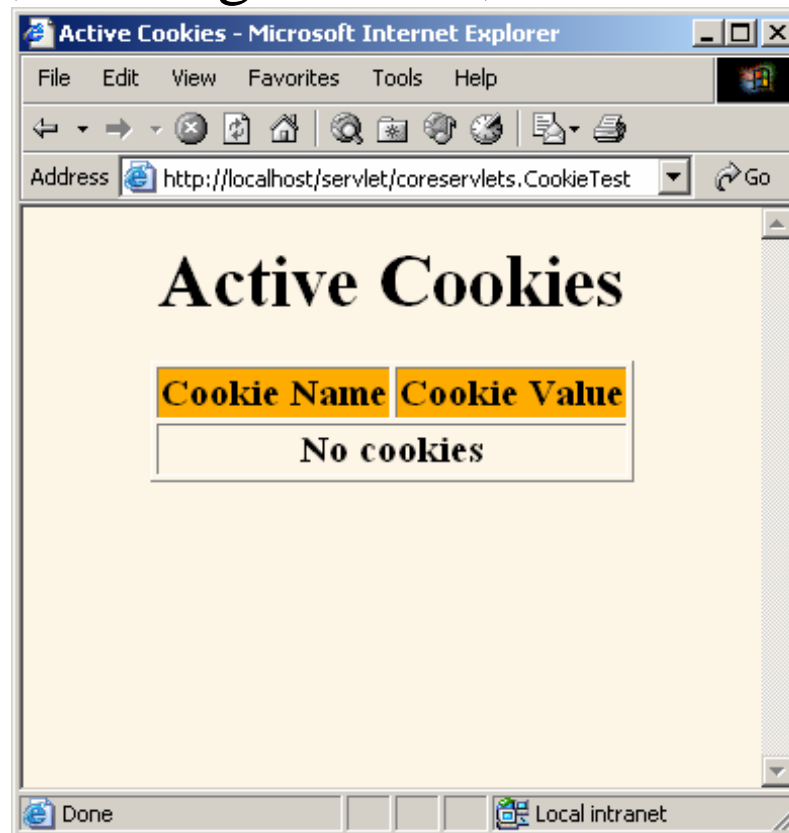
```
public class CookieTest extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        for(int i=0; i<3; i++) {
            Cookie cookie =
                new Cookie("Session-Cookie-" + i,
                          "Cookie-Value-S" + i);
            // No maxAge (ie maxAge = -1)
            response.addCookie(cookie);
            cookie = new Cookie("Persistent-Cookie-" + i,
                               "Cookie-Value-P" + i);
            cookie.setMaxAge(3600);
            response.addCookie(cookie);
        }
    }
}
```

Differentiating Session Cookies from Persistent Cookies (Cont)

```
... // Start an HTML table
Cookie[] cookies = request.getCookies();
if (cookies == null) {
    out.println("<TR><TH COLSPAN=2>No cookies");
} else {
    Cookie cookie;
    for(int i=0; i<cookies.length; i++) {
        cookie = cookies[i];
        out.println
            ("<TR>\n" +
             " <TD>" + cookie.getName() + "\n" +
             " <TD>" + cookie.getValue());
    }
}
```

Differentiating Session Cookies from Persistent Cookies

- **Result of initial visit to CookieTest servlet**
 - Same result as when visiting the servlet, quitting the browser, waiting an hour, and revisiting the servlet.



Differentiating Session Cookies from Persistent Cookies

- **Result of revisiting CookieTest within an hour of original visit (same browser session)**
 - i.e., browser stayed open between the original visit and the visit shown here



The screenshot shows a Microsoft Internet Explorer window titled "Active Cookies - Microsoft Internet Explorer". The address bar displays "http://localhost/servlet/coreservlets.CookieTest". The main content area shows the heading "Active Cookies" and a table with the following data:

Cookie Name	Cookie Value
Session-Cookie-0	Cookie-Value-S0
Persistent-Cookie-0	Cookie-Value-P0
Session-Cookie-1	Cookie-Value-S1
Persistent-Cookie-1	Cookie-Value-P1
Session-Cookie-2	Cookie-Value-S2
Persistent-Cookie-2	Cookie-Value-P2

Differentiating Session Cookies from Persistent Cookies

- **Result of revisiting CookieTest within an hour of original visit (different browser session)**
 - I.e., browser was restarted between the original visit and the visit shown here.



Utility: Finding Cookies with Specified Names

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

Cookie Utilities -getCookieValue

- Given the request object, a name, and a default value, this method tries to find the value of the cookie with the given name. If no cookie matches the name, the default value is returned.

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

Cookie Utilities -getCookie

- Given the request object and a name, this method tries to find and return the cookie that has the given name. If no cookie matches the name, null is returned.

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

Utility: Creating Long-Lived Cookies

```
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);  
    }  
}
```

Applying Utilities

- **The class RepeatVisitor2 redoes the RepeatVisitor servlet but**
 - Invokes `CookieUtilities.getCookieValue` and
 - Instantiates an object of the `LongLivedCookie` class

、

Applying Utilities: RepeatVisitor2

```
public class RepeatVisitor2 extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        boolean newbie = true;
        String value = CookieUtilities.getCookieValue
            (request, "repeatVisitor2", "no");
        if (value.equals("yes")) {
            newbie = false;
        }
        String title;
        if (newbie) {
            LongLivedCookie returnVisitorCookie =
                new LongLivedCookie("repeatVisitor2", "yes");
            response.addCookie(returnVisitorCookie);
            title = "Welcome Aboard";
        } else {
            title = "Welcome Back";
        }
        remainder of code to write HTML output to client
    }
}
```

Modifying Cookie Values

- **Replacing a cookie value**
 - Send the same cookie name with a different cookie value.
 - Reusing incoming Cookie objects.
 - Need to call `response.addCookie`; merely calling `setValue` is not sufficient.
 - Also need to reapply any relevant cookie attributes by calling `setMaxAge`, `setPath`, etc.—cookie attributes are not specified for incoming cookies.
 - Usually not worth the bother, so new Cookie object used
- **Instructing the browser to delete a cookie**
 - Use `setMaxAge` to assign a maximum age of 0.
- **The following example reuses a cookie to keep track of how many times each client has visited the site**

Tracking User Access Counts

```
public class ClientAccessCounts extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        String countString =
            CookieUtilities.getCookieValue(request,
                "accessCount", // name of cookie
                "1" ); // default value
        int count = 1;
        try {
            count = Integer.parseInt(countString);
        } catch(NumberFormatException nfe) { }
        LongLivedCookie c =
            new LongLivedCookie("accessCount",
                               String.valueOf(count+1));
        response.addCookie(c);
    }
}
```

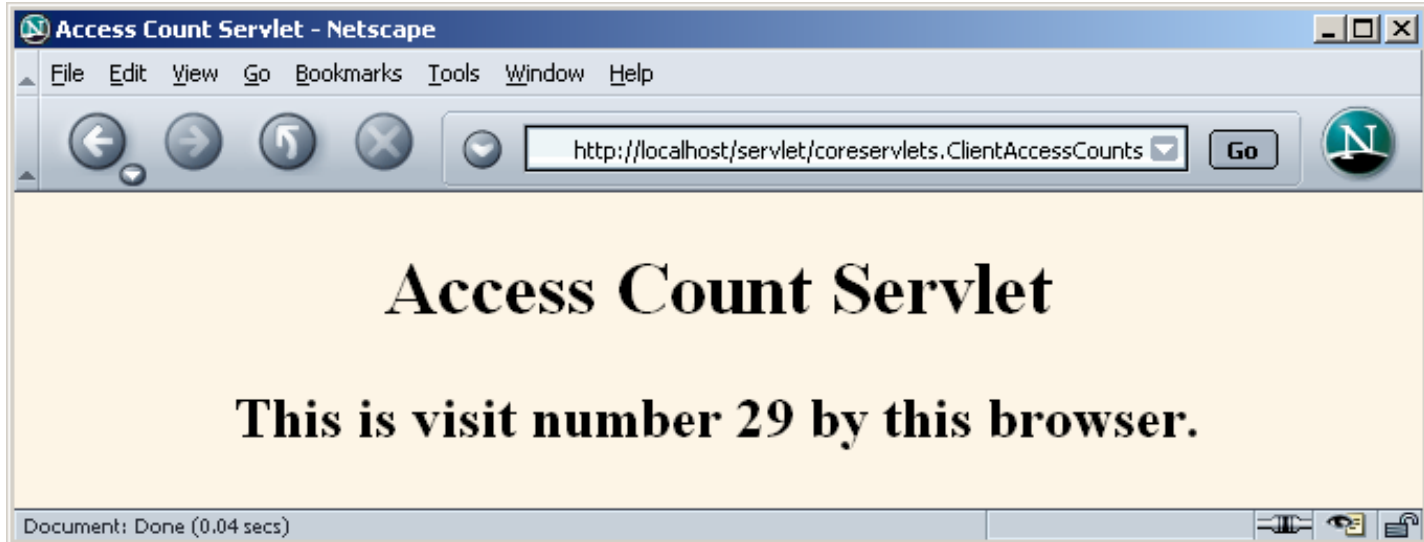
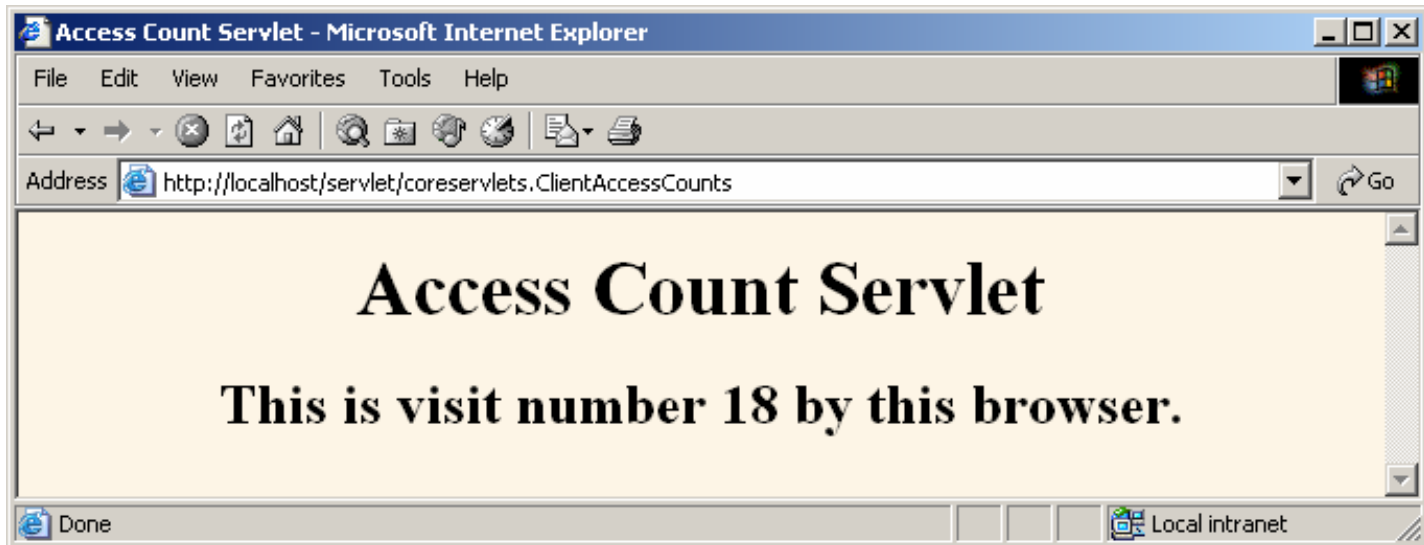
Tracking User Access Counts (Continued)

...

```
out.println(docType +
    "<HTML>\n" +
    "<HEAD><TITLE>" + title +
    "</TITLE></HEAD>\n" +
    "<BODY BGCOLOR=\"#FDF5E6\">\n" +
    "<CENTER>\n" +
    "<H1>" + title + "</H1>\n" +
    "<H2>This is visit number " +
    count + " by this browser.</H2>\n"+
    "</CENTER></BODY></HTML>");
}
```

```
}
```

Tracking User Access Counts (Results)



Using Cookies to Remember User Preferences

- **RegistrationForm servlet**
 - Uses cookie values to prepopulate form field values
 - Uses default values if no cookies are found
- **Registration servlet**
 - Creates cookies based on request parameters received
 - Displays values if all parameters are present
 - Redirects to form if any parameter is missing

RegistrationForm Servlet

```
public class RegistrationForm extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String actionURL =
            "/servlet/coreservlets.RegistrationServlet";
        String firstName =
            CookieUtilities.getCookieValue(request,
                                           "firstName", "");
        String lastName =
            CookieUtilities.getCookieValue(request,
                                           "lastName", "");
        String emailAddress =
            CookieUtilities.getCookieValue(request,
                                           "emailAddress",
                                           "");
    }
}
```

RegistrationForm Servlet (Continued)

```
out.println
  (docType +
   "<HTML>\n" +
   "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
   "<BODY BGCOLOR=\"#FDF5E6\">\n" +
   "<CENTER>\n" +
   "<H1>" + title + "</H1>\n" +
   "<FORM ACTION=\"" + actionURL + "\">\n" +
   "First Name:\n" +
   "  <INPUT TYPE=\"TEXT\" NAME=\"firstName\" " +
   "    \"VALUE=\"" + firstName + "\"><BR>\n" +
   "Last Name:\n" +
   "  <INPUT TYPE=\"TEXT\" NAME=\"lastName\" " +
   "    \"VALUE=\"" + lastName + "\"><BR>\n"+
   "Email Address: \n" +
   "  <INPUT TYPE=\"TEXT\" NAME=\"emailAddress\" " +
   "    \"VALUE=\"" + emailAddress + "\"><P>\n" +
   "<INPUT TYPE=\"SUBMIT\" VALUE=\"Register\">\n" +
   "</FORM></CENTER></BODY></HTML>");
}
```

Registration Servlet

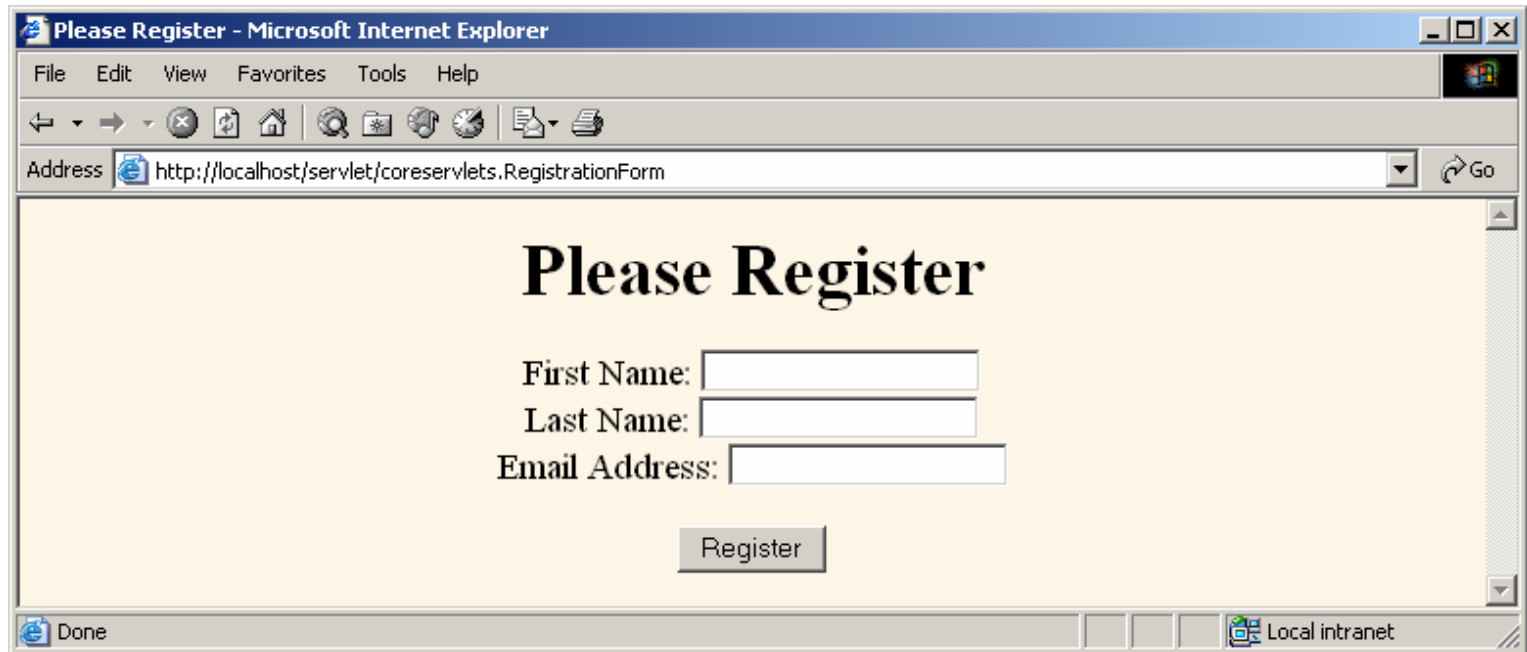
```
public class RegistrationServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        boolean isMissingValue = false;
        String firstName =
            request.getParameter("firstName");
        if (isMissing(firstName)) {
            firstName = "Missing first name";
            isMissingValue = true;
        }
        String lastName =
            request.getParameter("lastName");
        if (isMissing(lastName)) {
            lastName = "Missing last name";
            //above line causes error as cookie stores value with
            //double quotes around it. This causes an HTML error.
            //Change to "Missing_last_name";
            isMissingValue = true;
        }
        ...
    }
}
```

Registration Servlet (Continued)

```
Cookie c1 =
    new LongLivedCookie("firstName", firstName);
response.addCookie(c1);
Cookie c2 =
    new LongLivedCookie("lastName", lastName);
response.addCookie(c2);
Cookie c3 = new LongLivedCookie("emailAddress",
                                emailAddress);
response.addCookie(c3);
String formAddress =
    "/servlet/coreservlets.RegistrationForm";
if (isMissingValue) {
    response.sendRedirect(formAddress);
} else { ... }
```

```
// isMissing method needs to check for values of  
Missing_... as this should be considered missing  
as well
```


RegistrationForm (Initial Result)



The image shows a screenshot of a Microsoft Internet Explorer browser window. The title bar reads "Please Register - Microsoft Internet Explorer". The address bar contains the URL "http://localhost/servlet/coreservlets.RegistrationForm". The main content area displays a registration form with the following elements:

- Please Register** (Large heading)
- First Name:
- Last Name:
- Email Address:
-

The status bar at the bottom shows "Done" and "Local intranet".

RegistrationForm (Submitting Incomplete Form)



RegistrationForm (Submitting Complete Form)



RegistrationForm (Initial Result on Later Visit)



The screenshot shows a Microsoft Internet Explorer browser window titled "Please Register - Microsoft Internet Explorer". The address bar displays "http://localhost/servlet/coreservlets.RegistrationForm". The main content area features the heading "Please Register" in a large, bold, serif font. Below the heading are three text input fields: "First Name:" with the value "Marty", "Last Name:" with the value "Hall", and "Email Address:" with the value "hall@coreservlets.com". A "Register" button is positioned below the email field. The status bar at the bottom indicates "Done" and "Local intranet".

Please Register

First Name:

Last Name:

Email Address:

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