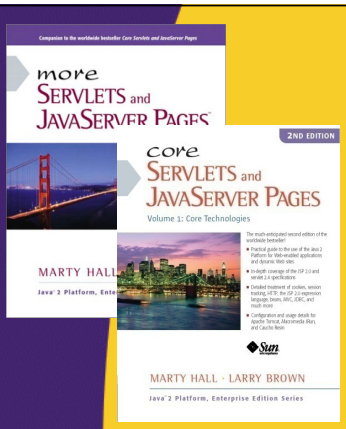
© 2012 Marty Hall

Handling Cookies

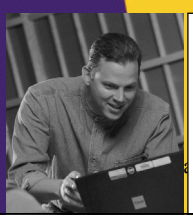
Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/csajsp2.html

Customized Java EE Training: http://courses.coreservlets.com/
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

© 2012 Marty Hall



For live Java EE training, please see training courses at http://courses.coreservlets.com/.
JSF 2, PrimeFaces, Servlets, JSP, Ajax (with jQuery), GWT, Android development, Java 6 and 7 programming, SOAP-based and RESTful Web Services, Spring, Hibernate/JPA, XML, Hadoop, and customized combinations of topics.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact hall@coreservlets.com for details.

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

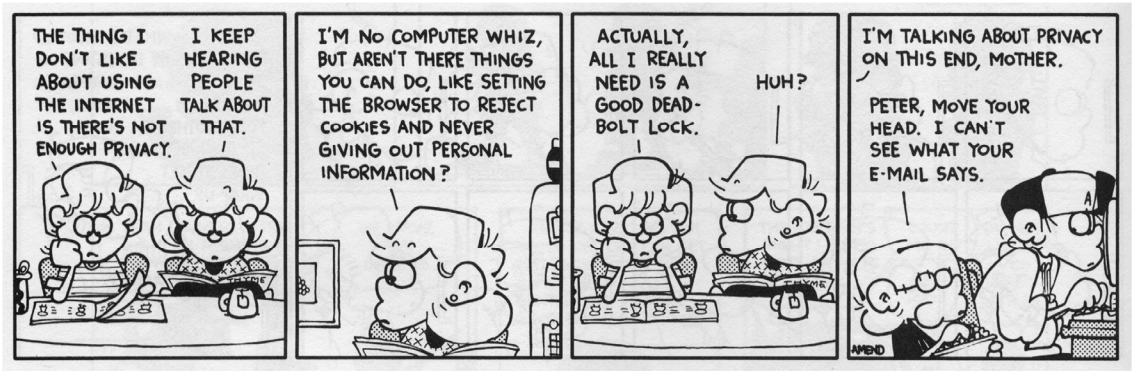
- **Idea**
  - Servlet sends a simple name and value to client.
  - Client returns same name and value when it connects to same site (or same domain, depending on cookie settings).
- **Typical Uses of Cookies**
  - Identifying a user during an e-commerce session
    - Servlets have a higher-level API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

Cookies and Focused Advertising

Amazon.com home page for new visitor or visitor with cookies disabled.

Amazon.com home page for repeat visitor. Books shown are based on prior history.

Cookies and Privacy

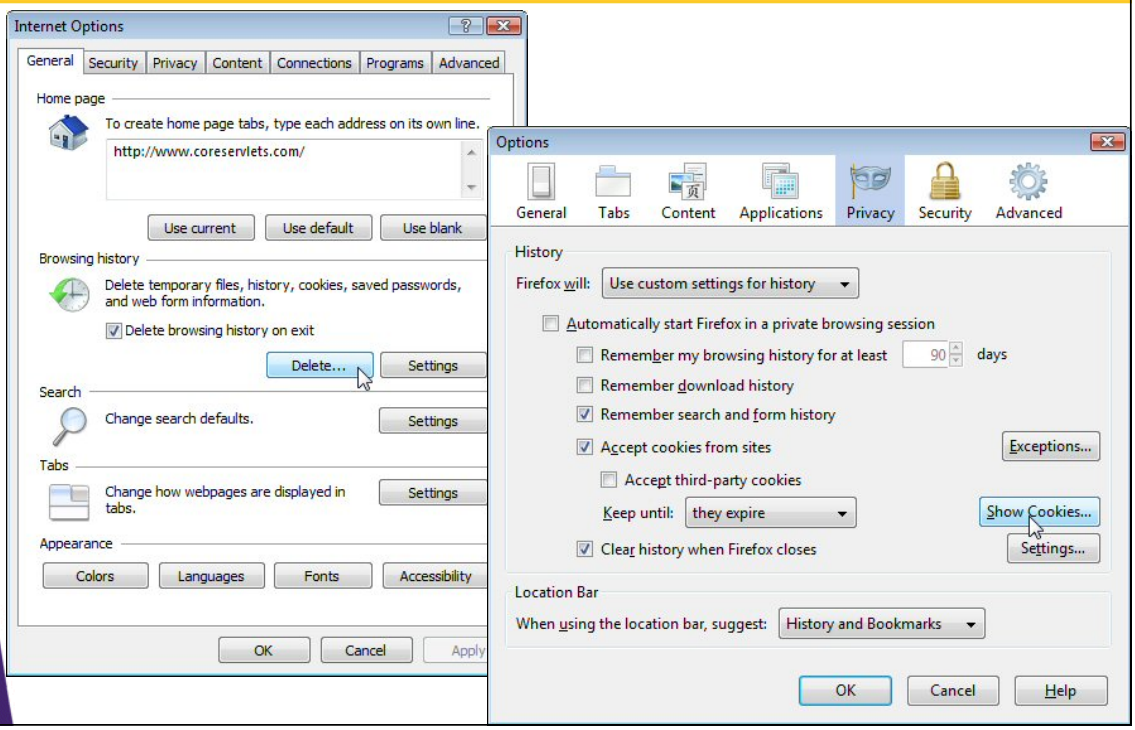


FoxTrot © 1998 Bill Amend. Reprinted with permission of Universal Press Syndicate. All rights reserved.

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

Manually Deleting Cookies (To Simplify Testing)



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.

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

- **Set the maximum age.**
  - 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.
  - 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(Cookie cookie: cookies) {
    if (cookieName.equals(cookie.getName())) {
      doSomethingWith(cookie.getValue());
    }
  }
}
```

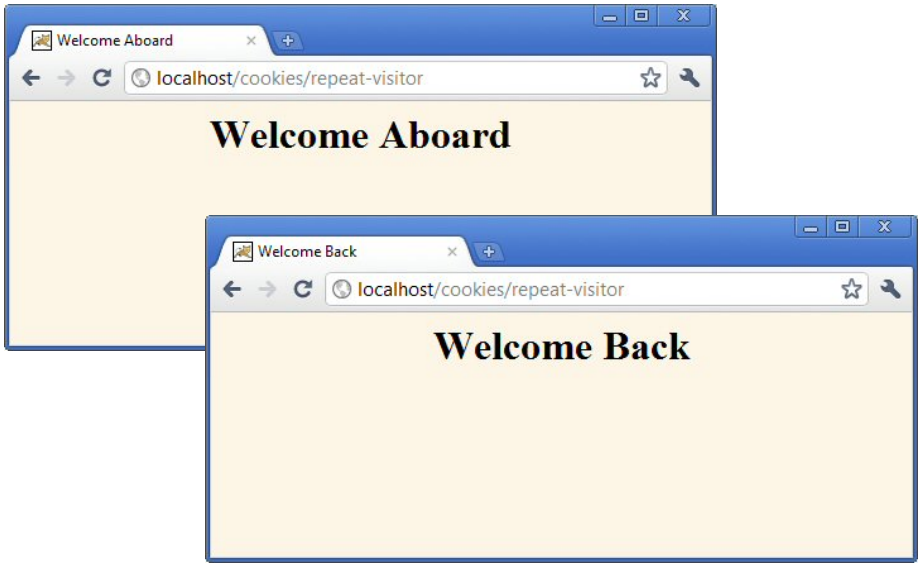
Using Cookies to Detect First-Time Visitors

```
@WebServlet("/repeat-visitor")
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(Cookie c: cookies) {
        if ((c.getName().equals("repeatVisitor")) &&
            (c.getValue().equals("yes"))) {
          newbie = false;
          break;
        }
      }
    }
```


Using Cookies to Detect First-Time Visitors (Continued)

```
    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

- **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.

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.
- **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.

Differentiating Session Cookies from Persistent Cookies

```
@WebServlet("/cookie-test")
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 {
      for(Cookie cookie: cookies) {
        out.println
          ("<TR>\n" +
           "  <TD>" + cookie.getName() + "\n" +
           "  <TD>" + cookie.getValue());
      }
    }
    out.println("</TABLE></BODY></HTML>");
  }
}
```


Differentiating Session Cookies from Persistent Cookies

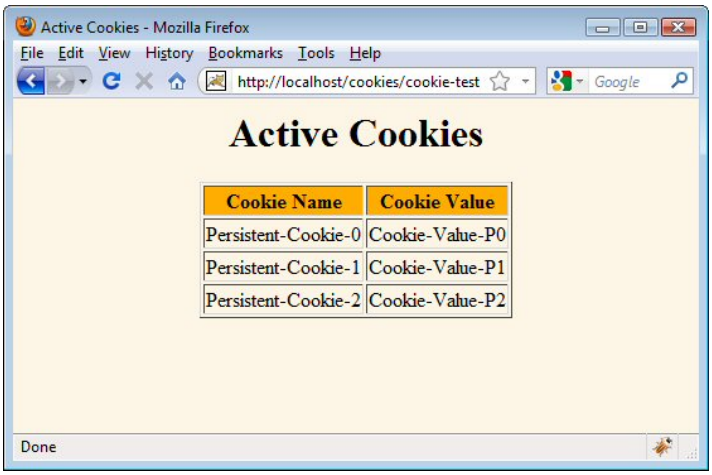
- **Result of initial visit to CookieTest servlet**
  - Same result as when visiting the servlet, quitting the browser, waiting an hour, and revisiting the servlet.

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

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(Cookie cookie: cookies) {
        if (cookieName.equals(cookie.getName())) {
          return(cookie.getValue());
        }
      }
    }
    return(defaultValue);
  }
  …
}
```


Utility: Creating Long-Lived Cookies

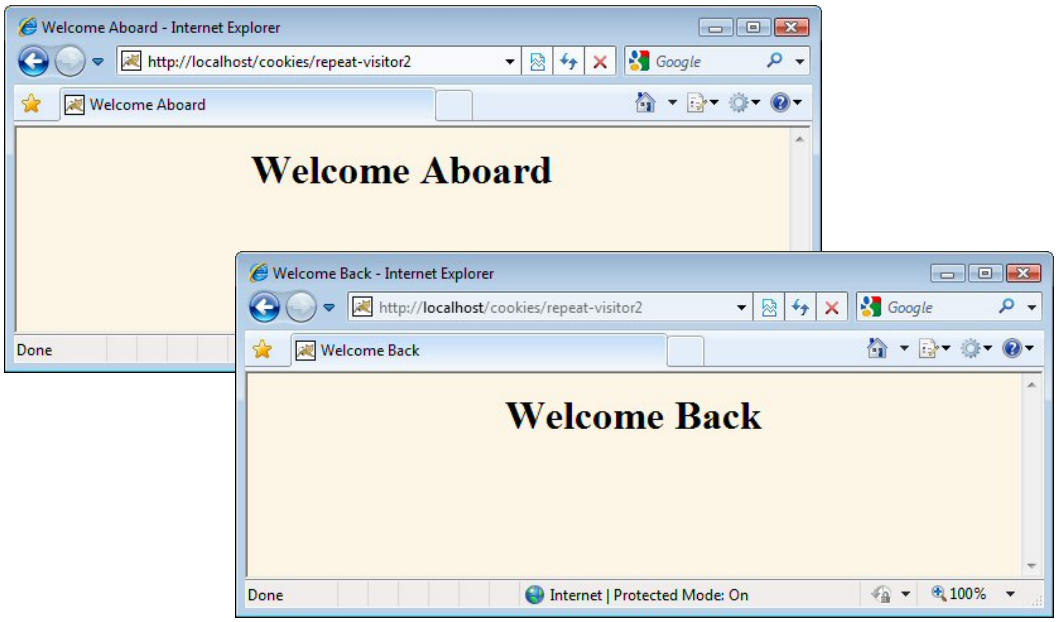
```
public class LongLivedCookie extends Cookie {
  public static final int SECONDS_PER_YEAR =
    60*60*24*365;

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

Applying Utilities: RepeatVisitor2

```
@WebServlet("/repeat-visitor2")
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";
    }
```


Applying Utilities: RepeatVisitor2



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**
  - Call setMaxAge(0)

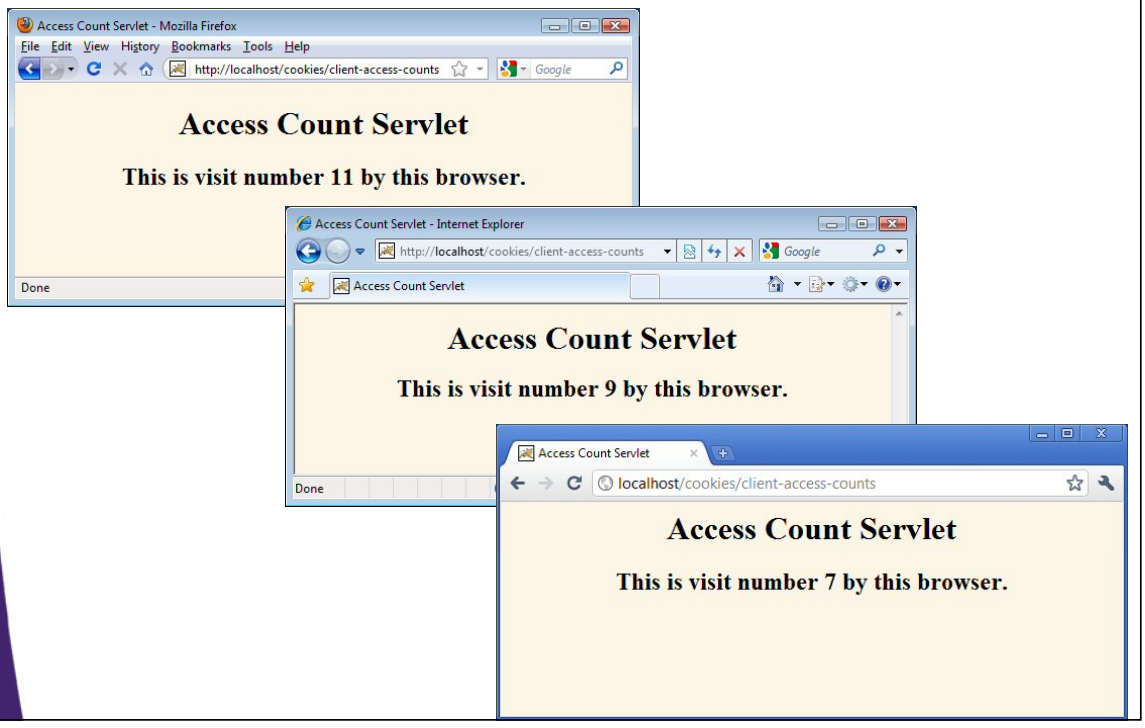
Tracking User Access Counts

```
@WebServlet("/client-access-counts")
public class ClientAccessCounts extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    String countString =
      CookieUtilities.getCookieValue(request,
                                     "accessCount",
                                     "1");
    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
  - Will be redone in JSP later in class
- **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

```
@WebServlet("/registration-form")
public class RegistrationForm extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    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=\"registration\">\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

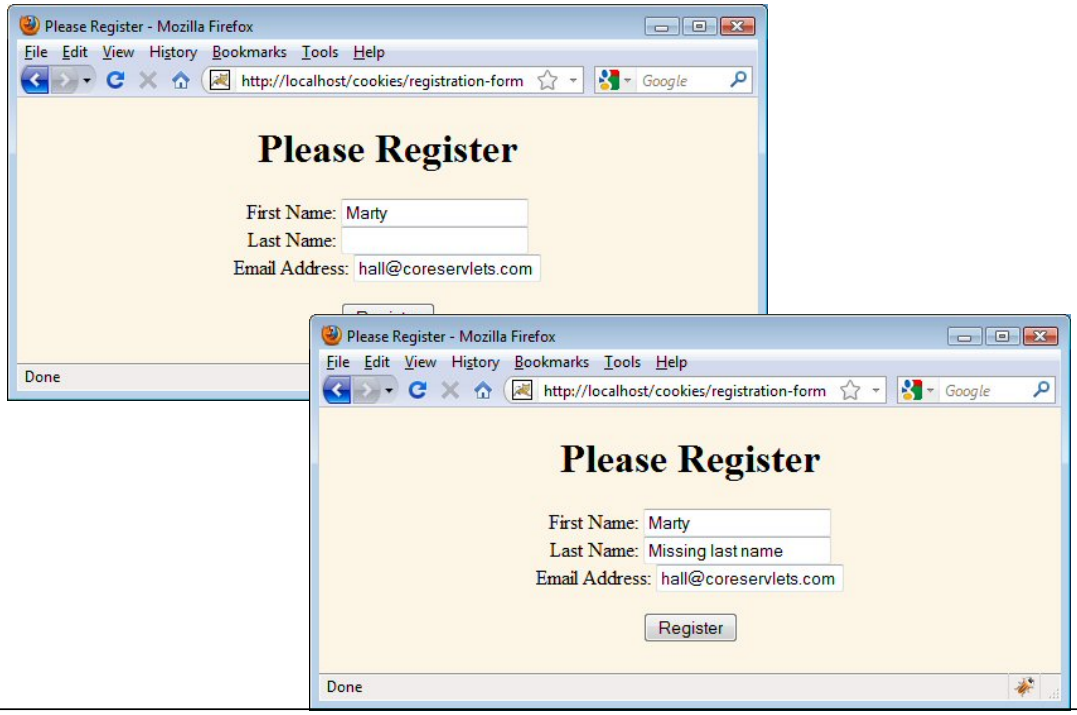
```
@WebServlet("/registration")
public class RegistrationServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    boolean isMissingValue = 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";
      isMissingValue = true;
    }
```

Registration Servlet (Continued)

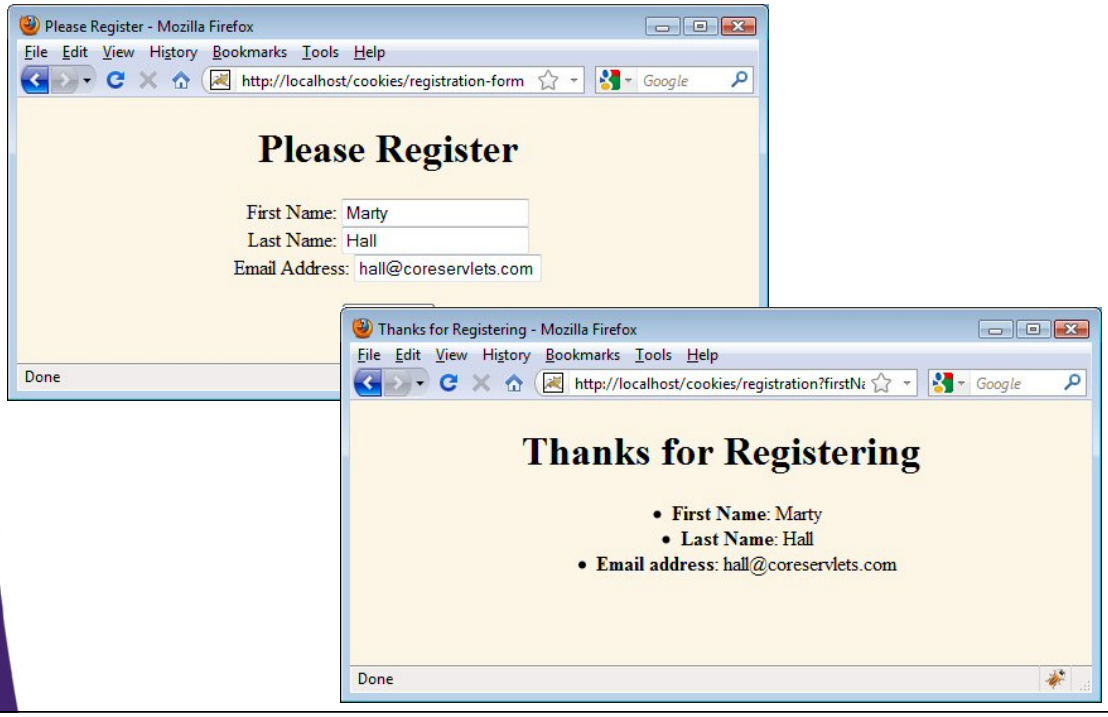
```
Cookie c1 =
  new LongLivedCookie("firstName", firstName);
response.addCookie(c1);
Cookie c2 =
  new LongLivedCookie("lastName", lastName);
response.addCookie(c2);
Cookie c3 = new LongLivedCookie("emailAddress",
                                emailAddress);
response.addCookie(c3);
if (isMissingValue) {
  response.sendRedirect("registration-form");
} else { … }
```


RegistrationForm (Initial Result)

RegistrationForm (Submitting Incomplete Form)



RegistrationForm (Submitting Complete Form)



RegistrationForm (Initial Result on Later Visit)

Summary

- **Basic functionality**
  - Cookies involve name/value pairs sent from server to browser and *automatically* returned when the same page (or possibly same site or domain) is visited later
- **Cookies let you**
  - Track sessions (use higher-level session-tracking 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

© 2012 Marty Hall

Questions?

JSF 2, PrimeFaces, Java 7, Ajax, jQuery, Hadoop, RESTful Web Services, Android, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training

Customized Java EE Training: http://courses.coreservlets.com/
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.