CNT 4714: Enterprise Computing
Fall 2015

Introduction to Servlet Technology– Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
Office Hours: M, T, W, & Th 3:00-4:30pm

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

More HTML Document Details

- Let's look a bit closer at what happens in our servlet as it executes. (See the servlet code on page 4.)

```
protected void doGet( HttpServletRequest request,
                      HttpServletResponse response )
        throws ServletException, IOException {
```

  - This line begins the overridden method `doGet` to respond to the get requests. In this case, the `HttpServletRequest` object parameter represents the client's request and the `HttpServletResponse` object parameter represents the server's response to the client.
  - If method `doGet` is unable to handle a client's request, it throws an exception of type `javax.servlet.ServletException`. If `doGet` encounters an error during stream processing (when reading from the client or writing to the client), it throws a `java.io.IOException`.

More HTML Document Details (cont.)

```
response.setContentType( "text/html" );
      PrintWriter out = response.getWriter();
```

- The first line above uses the response object's `setContentType` method to specify the content type of the document to be sent as the response to the client. This enables the client browser to understand and handle the content it receives from the server. The content type is also referred to as the MIME (Multipurpose Internet Mail Extension) type of the data. In this servlet, the content type is `text/html` to indicate to the browser that the response is an HTML document.
- The second line above uses the response object's getWriter method to obtain a reference to the PrintWriter object that enables the servlet to send content to the client. If the response is binary data, like an image, method getOutputStream would be used to obtain a reference to a ServletOutputStream object.

```java
// WelcomeServlet2.java
// Processing HTTP get requests containing data.

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

public class WelcomeServlet2 extends HttpServlet {
   // process "get" request from client
   protected void doGet( HttpServletRequest request,
      HttpServletResponse response )
         throws ServletException, IOException
   {
      String clientName = request.getParameter( "clientname" );

      response.setContentType( "text/html" );
      PrintWriter out = response.getWriter();

      // send HTML document to client
      // start HTML document
      out.println( "<!DOCTYPE html>" );
      out.println( "<html lang='en'>" );
      out.println ("<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=blue >");
      out.println ("<body style='tab-interval:.5in'>");
      // head section of document
      out.println( "<head>" );
      out.println( "<title>Processing get requests with data</title>" );
      out.println( "</head>" );
      // body section of document
      out.println( "<body>" );
      out.println( "<h1>Hello " + clientName + ",<br />" );
      out.println();
      out.println( "Welcome to the Exciting World of Servlet Technology!</h1>" );
      out.println( "</body>" );
      // end HTML document
      out.println( "</html>" );
      out.close();  // close stream to complete the page
   }
}
```

Java source file | length : 1464 lines : 41 | Ln : 35 Col : 14 Sel : 0 | 0 | Dos\Windows | UTF-8 w/o BOM | INS

Deploying a Web Application

- Servlets, JSPs and their supporting files are deployed as part of a Web application.
- Web applications are deployed in the `webapps` subdirectory of Tomcat.
- A Web application has a well-known directory structure in which all the files that are part of the application reside.
- This directory structure is created by the server administrator in the `webapps` directory, or the entire directory structure can be archived in a Web application archive file known as a `WAR file` (ending with a `.war` file extension – `war` stands for `web application archive`) which is placed in the `webapps` directory.

Deploying a Web Application (cont.)

- The Web application directory structure contains a context root, which is the top-level directory for an entire Web application along with several subdirectories as shown below:

context root – The root directory for the Web application. All the JSPs, HTML documents, servlets and supporting files such as images and class files reside in this directory or one of the subdirectories. The name of this directory is specified by the Web application creator. To provide structure in a Web application, subdirectories can be placed in the context root. It is common to see an images subdirectory, for example.

WEB-INF – This subdirectory contains the Web application deployment descriptor web.xml.

WEB-INF/classes – This subdirectory contains the servlet class files and other supporting class files used in a Web application. If the classes are part of a package, the complete package directory structure would begin here.

WEB-INF/lib – This subdirectory contains Java archive (JAR) files. The JAR files can contain servlet class files and other supporting class files.

Deploying a Web Application (cont.)

- As we mentioned in the previous section of notes, Tomcat will default to a welcome page which is specified in the `web.xml` file. The standard default values were shown on page 50 in the previous set of notes.
- If you do not create one of these files, the default page for a web application is not very appealing.

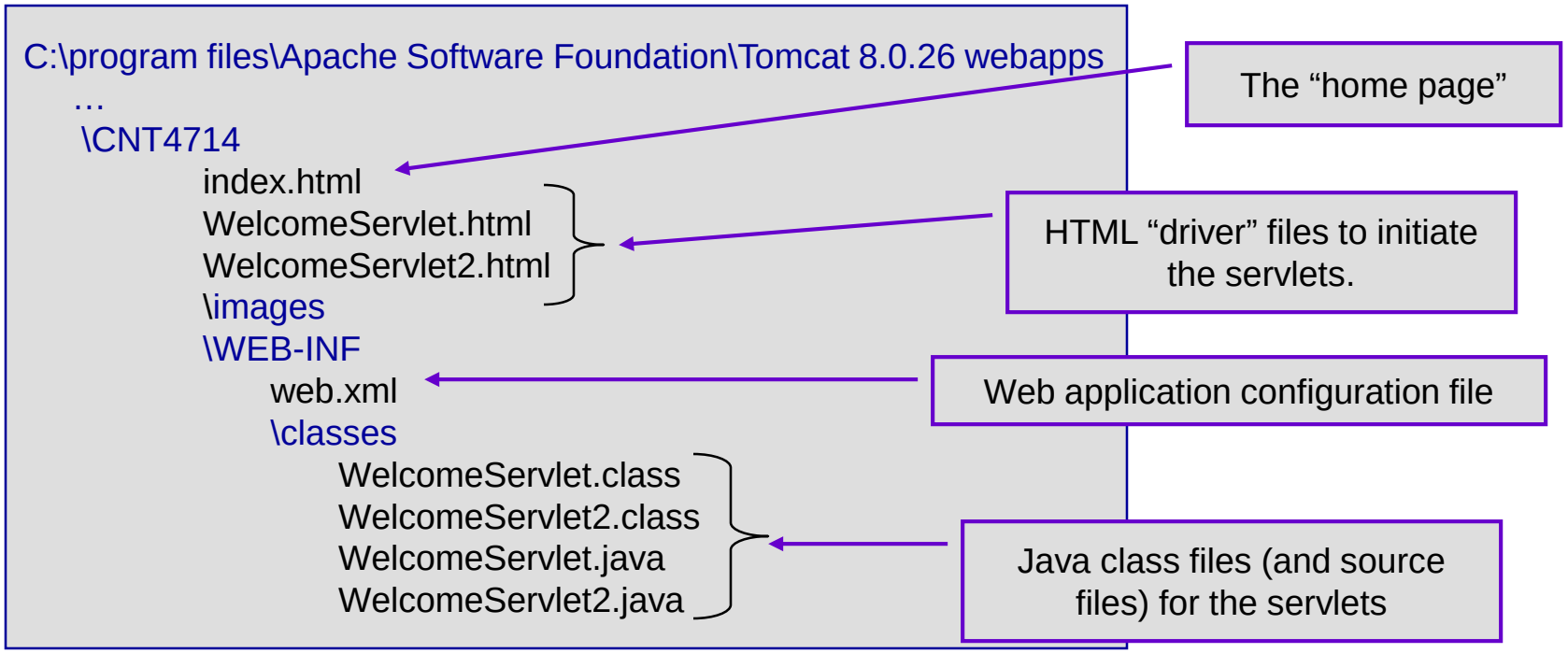
Deploying a Web Application (cont.)

- Since we would like our clients to see something more appropriate that the default web application page, you should create your own web application welcome page.
- This page is simply an HTML page and I've created one for the web applications we create from this point forward. I've simply modeled the page using our course web page as a template. The code for this page is included on the course code page if you want to use it, but feel free to design your own.
- I'll utilize this page as a home page for all of our servlets and JSPs that we'll see later in the course.
- I've also created a new web application named CNT4714 that we'll use for our future servlets and JSPs.
- Now, when the client enters the URL, http://localhost:8080/CNT4714 they will see the home page shown in the next slide.

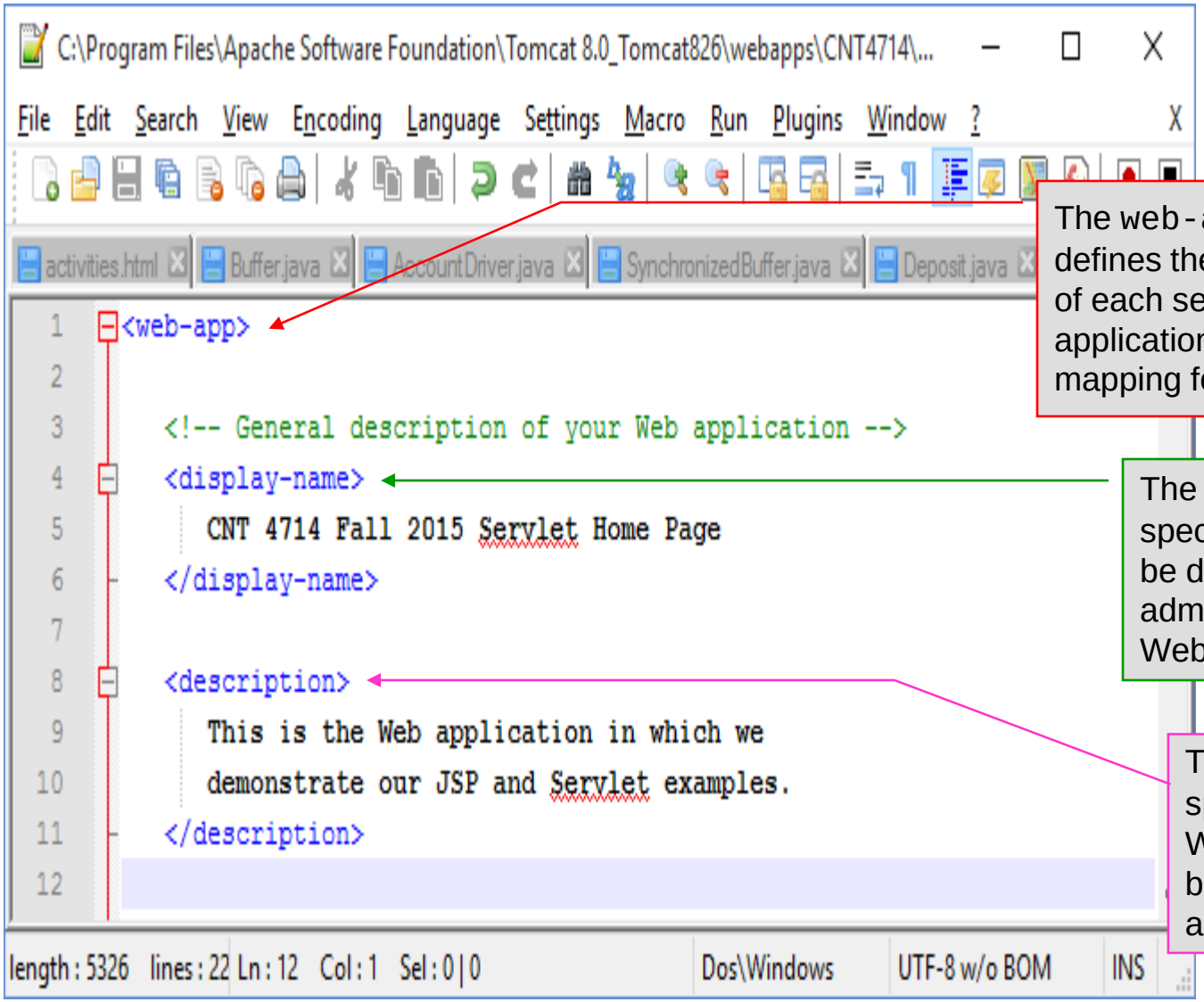

Opera
/manager
CNT 4714 - Fall 2015
localhost:8080/CNT4714/
UCF
CNT 4714 - Enterprise Computing Fall 2015
Instructor: Dr. Mark Llewellyn
HEC 236
(407) 823-2790
Email to: markl@cs.ucf.edu
Welcome To The CNT 4714 Servlet Home Page
SERVLET HOME PAGE
Click any of the links below to directly invoke the corresponding servlet
Click here to invoke a simple Welcome Servlet
Click here to invoke a more personal Welcome Servlet
Click here to run the Redirection Servlet
Click here to see the current date and time information
Click here to run a Cookie Demonstration Servlet

Deploying a Web Application (cont.)

- The Web application directory structure that I set up for the CNT4714 web application looks like the following:



A Closer Look at the web.xml File

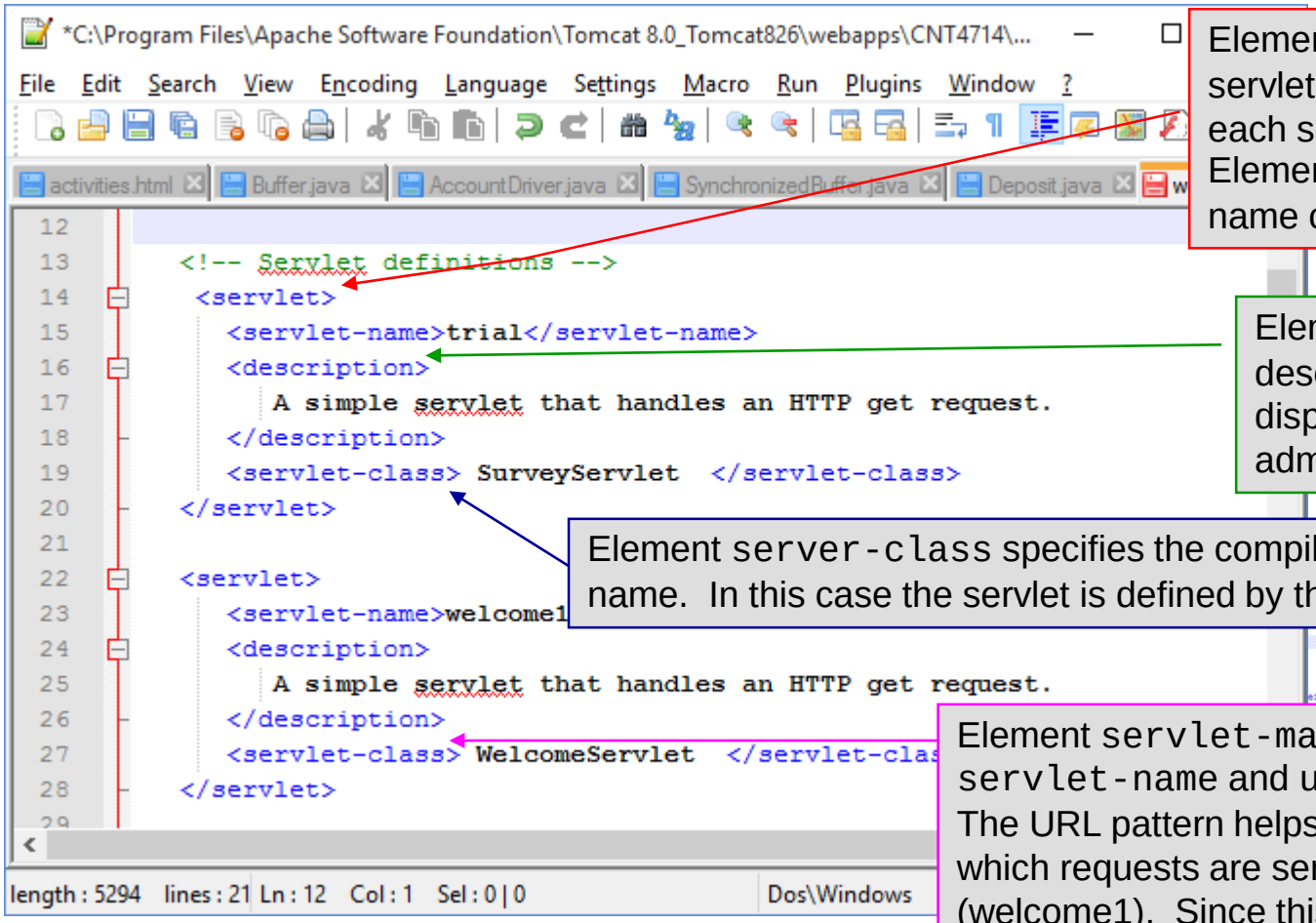


The `web-app` element defines the configuration of each servlet in the Web application and the servlet mapping for each servlet.

The `display-name` element specifies a name which can be displayed to the server administrator on which the Web application is installed.

The `description` element specifies a description of the Web application that can also be displayed to the server administrator.

A Closer Look at the web.xml File



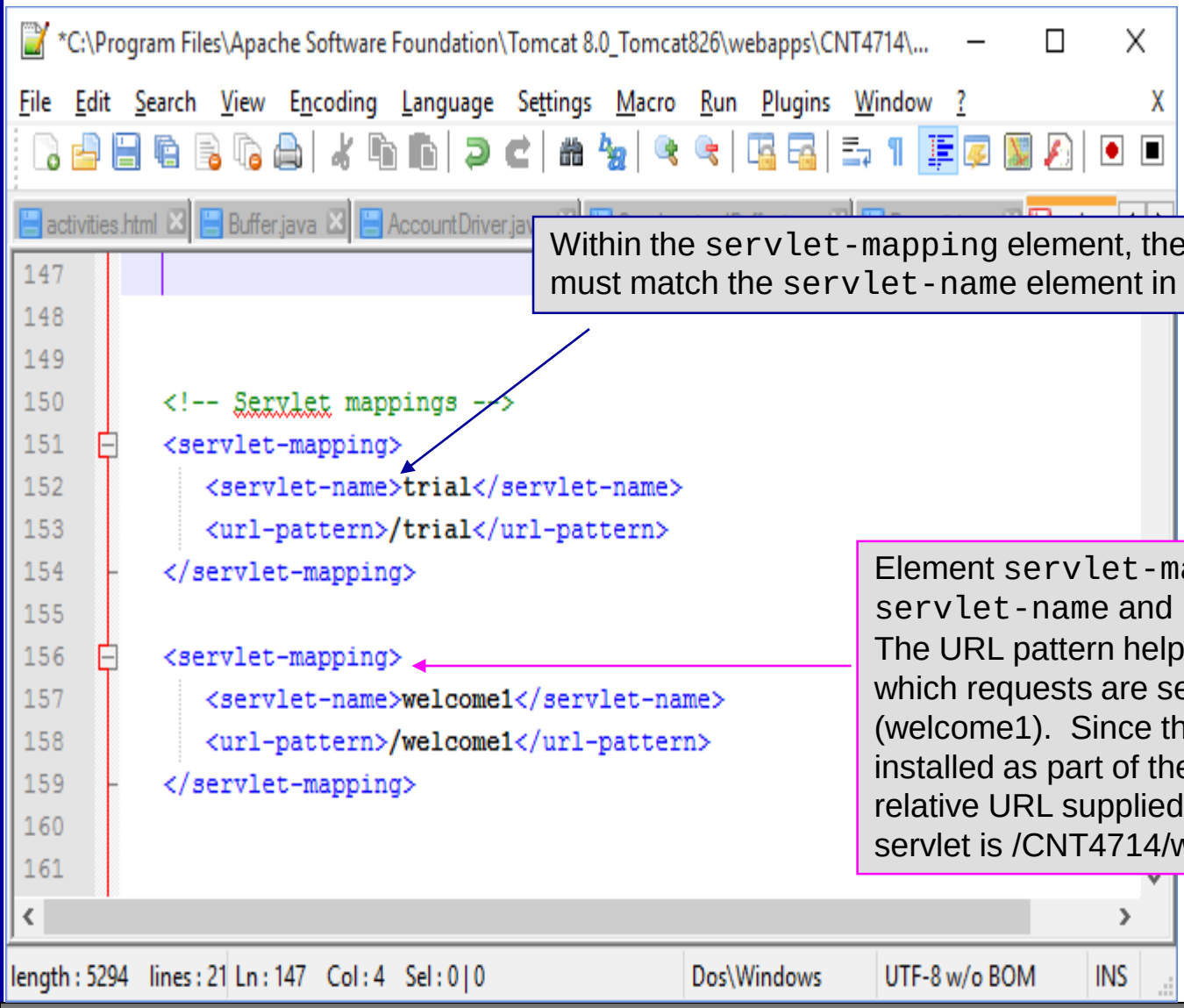
Element `servlet` describes a servlet. There is one of these for each servlet in the Web application. Element `servlet-name` is the name chosen for the servlet.

Element `description` describes the servlet and can be displayed to the server administrator.

Element `server-class` specifies the compiled servlet's fully qualified path name. In this case the servlet is defined by the class WelcomeServlet.

Element `servlet-mapping` specifies the `servlet-name` and `url-pattern` elements. The URL pattern helps the server determine which requests are sent to the servlet (welcome1). Since this web application will be installed as part of the CNT4714 context root, the relative URL supplied to the browser to invoke the servlet is /CNT4714/welcome1.

A Closer Look at the web.xml File



Within the `servlet-mapping` element, the element `servlet-name` value must match the `servlet-name` element in the `servlet` element.

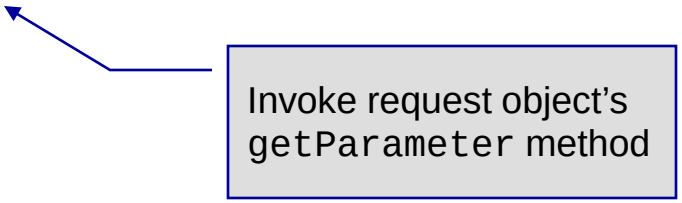
Element `servlet-mapping` specifies the `servlet-name` and `url-pattern` elements. The URL pattern helps the server determine which requests are sent to the servlet (welcome1). Since this web application will be installed as part of the CNT4714 context root, the relative URL supplied to the browser to invoke the servlet is /CNT4714/welcome1.

Handling HTTP `get` Requests Containing Data

- When a requesting a document or resource from a Web server, it is often the case that data needs to be supplied as part of the request. The second servlet example in the previous set of notes responds to an HTTP `get` request that contains the name entered by the user. The servlet uses this name as part of the response to the client.

- Parameters are passes as name-value pairs in a `get` request. Within the source code for the second `WelcomeServlet2` you will find the following line (see next page):

```
String clientName = request.getParameter( “clientname" );
```



C:\Program Files\Apache Software Foundation\Tomcat 8.0_Tomcat826\webapps\CNT4714\WEB-INF\classes\WelcomeServlet2.java - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

AccountDriver.java | SynchronizedBuffer.java | Deposit.java | web.xml | WelcomeServlet2.java

```
1  // WelcomeServlet2.java
2  // Processing HTTP get requests containing data.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet2 extends HttpServlet {
9     // process "get" request from client
10    protected void doGet( HttpServletRequest request,
11       HttpServletResponse response )
12          throws ServletException, IOException
13    {
14       String clientName = request.getParameter( "clientname" );
15
16       response.setContentType( "text/html" );
17       PrintWriter out = response.getWriter();
18
19       // send HTML document to client
20       // start HTML document
21       out.println( "<!DOCTYPE html>" );
22       out.println( "<html lang='en'>" );
23       out.println ("<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=blue >");
24       out.println ("<body style='tab-interval:.5in'>");
25       // head section of document
26       out.println( "<head>" );
27       out.println( "<title>Processing get requests with data</title>" );
28       out.println( "</head>" );
29       // body section of document
30       out.println( "<body>" );
31       out.println( "<h1>Hello " + clientName + ",<br />" );
32       out.println();
33       out.println( "Welcome to the Exciting World of Servlet Technology!</h1>" );
34       out.println( "</body>" );
35       // end HTML document
36       out.println( "</html>" );
37       out.close();  // close stream to complete the page
38    }
39 }
```

Invoke request object's `getParameter` method

Java source file | length : 1464 lines : 41 | Ln : 1 Col : 1 Sel : 0 | 0 | Dos\Windows | UTF-8 w/o BOM | INS

Handling HTTP `get` Requests Containing Data (cont.)

- The `WelcomeServlet2.html` document provides a `form` in which the user can input their name into the text `input` element `clientname` and click the `Submit` button to invoke the servlet.
- When the user clicks the `Submit` bitton, the values of the `input` elements are placed in name-value pairs as part of the request to the server.
- Notice in the screen shot on the next page that the Tomcat server has appended `?clientname=Mark` to the end of the `action` URL. The ? separates the query string (i.e., the data passed as part of the `get` request) from the rest of the URL in a `get` request. The name-value pairs are passed with the name and value separated by an = sign. If there is more than one name-value pair, each pair is separated by an &.

C:\Program Files\Apache Software Foundation\Tomcat 8.0_Tomcat826\webapps\CNT4714\WelcomeServlet2.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

AccountDriver.java | SynchronizedBuffer.java | Deposit.java | web.xml | WelcomeServlet2.java | WelcomeServlet2.html

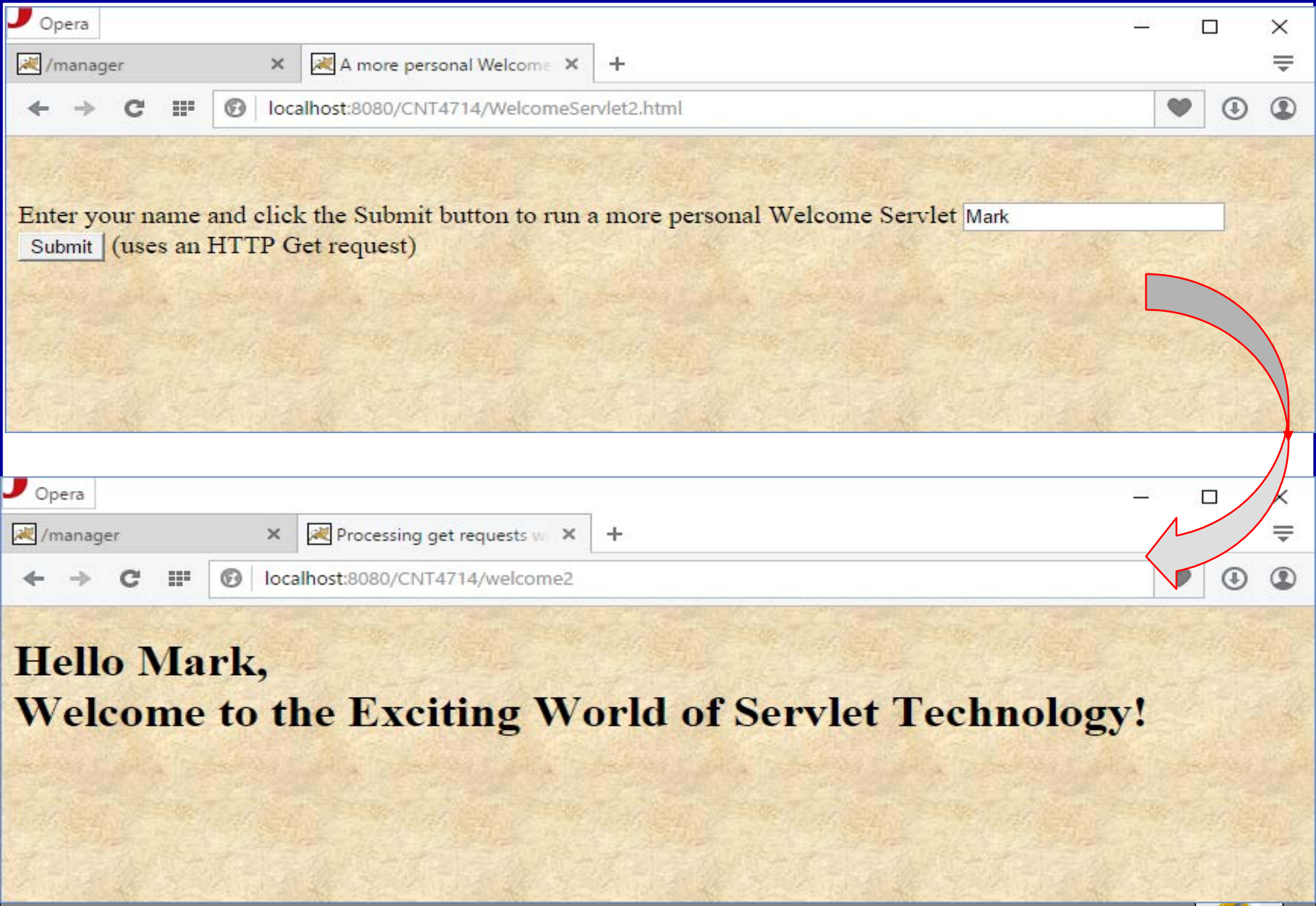
```
1  <!DOCTYPE html>
2  <!-- WelcomeServlet2.html -->
3  <html lang="en">
4  <head>
5     <title>A more personal Welcome Servlet - contains input
6  </head>
7  <body>
8     <font size = 4>
9     <body bgcolor=white background=images/background.jpg lang=EN-US
10       link=blue vlink=blue  style='tab-interval:.5in'>
11     <br>
12    <form action = "/CNT4714/welcome2" method = "get">
13       <p><label>
14          Enter your name and click the Submit button to run a more personal Welcome Servlet
15          <input type = "text" name = "clientname" />
16          <input type = "submit" value = "Submit" /><label> (uses an HTTP Get request)
17       </p></label>
18    </form>
19 </body>
20 </html>
```

Context root is `/CNT4714`

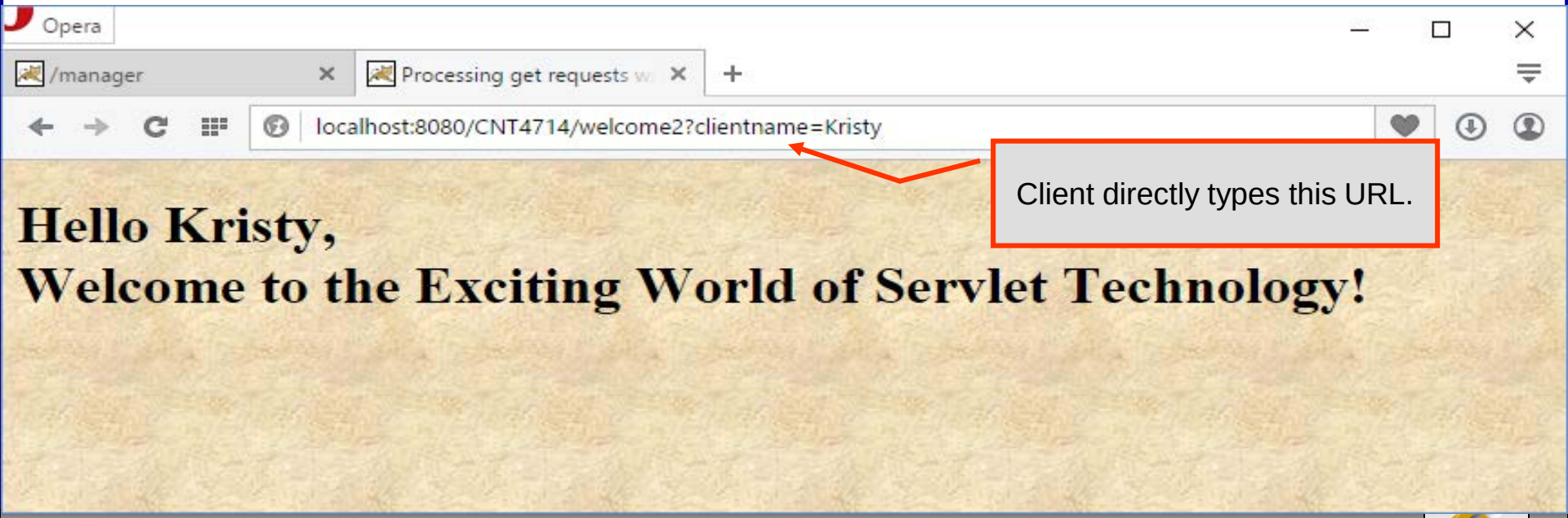
Servlet alias is `welcome2`

Form in `WelcomeServlet2.html` that specifies an input whose type is “text” and whose name is “`clientname`”

Hyper Text Markup length : 660 lines : 24 | Ln : 1 Col : 1 Sel : 0 | 0 | Macintosh | UTF-8 w/o BOM | INS


Opera
/manager
A more personal Welcome
localhost:8080/CNT4714/WelcomeServlet2.html
Enter your name and click the Submit button to run a more personal Welcome Servlet
Mark
Submit
(uses an HTTP Get request)
Opera
/manager
Processing get requests w
localhost:8080/CNT4714/welcome2
Hello Mark,
Welcome to the Exciting World of Servlet Technology!

Note: The same servlet could have been invoked directly by typing in directly to the browsers Address or Location field. This is shown in the overlay below



Handling HTTP `post` Requests

- An HTTP `post` request is typically used to send data from an HTML form to a server-side form handler that processes the data. For example, when you respond to a Web-based survey, a `post` request normally supplies the information you entered into the form to the Web server.
- If you were to replace the `doGet` method in WelcomeServlet2 with a `doPost` method, nothing would change in the apparent execution of the servlet with the exception that the values passed to the server are not appended to the request URL.
- This is illustrated by WelcomeServlet3 which is exactly the same as WelcomeServlet2 except that it uses the `doPost` method. Notice how the URL differs between the two versions.

WelcomeServlet2 uses the get method to supply the data to the form whereas WelcomeServlet3 uses the post method to do the same. Notice that the data is appended to the URL when the get method is used but it is not appended to the URL when the post method is used.

C:\Program Files\Apache Software Foundation\Tomcat 8.0_Tomcat826\webapps\CNT4714\WelcomeServlet3.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

Deposit.java | web.xml | WelcomeServlet2.java | WelcomeServlet2.html | WelcomeServlet3.html

```
1  <!-- WelcomeServlet3.html -->
2  <html>
3  <head>
4     <title>Handling an HTTP Post Request with Data</title>
5  </head>
6  <body>
7     <font size = 4>
8     <body bgcolor=white background=images/background.jpg lang=EN-US
9        link=blue vlink=blue  style='tab-interval:.5in'>
10     <br>
11    <form action = "/CNT4714/welcome3" method = "post">
12       <p><label>
13          Type your first name and press the Submit button
14          <input type = "text" name = "clientname" />
15          <input type = "submit" value = "Submit" /></label> (uses an HTTP Post request)
16       </label></p>
17    </form>
18 </body>
19 </html>
20
21
```

`WelcomeServlet3` uses a post method.

Hyper Text Markup length : 589 lines : 23 | Ln : 20 Col : 1 Sel : 0 | 0 | Macintosh | UTF-8 w/o BOM | INS

Java - Servlets/src/WelcomeServlet3.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

SQLGUIClient.java | ResultSetTableModelK.java | WelcomeServlet.java | *WelcomeServlet3.java

```
10 public class WelcomeServlet3 extends HttpServlet
11 {
12    // process "post" request from client
13    protected void doPost( HttpServletRequest request,
14       HttpServletResponse response )
15          throws ServletException, IOException
16    {
17       String clientName = request.getParameter( "clientname" );
18
19       response.setContentType( "text/html" );
20       PrintWriter out = response.getWriter();
21
22       // send HTML page to client
23       // start HTML document
24       out.println( "<html>" );
25       out.println( "<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=blue>");
26       out.println(" <body style='tab-interval:.5in' >");
27       // head section of document
28       out.println( "<head>" );
29       out.println(
30          "<title>Processing post requests with data</title>" );
31       out.println( "</head>" );
32       // body section of document
33       out.println( "<body>" );
34       out.println( "<h1>Hello " + clientName + ",<br />" );
35       out.println( "Welcome to Servlets!</h1>" );
36       out.println( "</body>" );
37
38       // end HTML document
39       out.println( "</html>" );
40       out.close(); // close stream to complete the page
41    } // end method doPost
42 } // end class WelcomeServlet3
43
44
```

doPost() method is used in this case.

Writable | Smart Insert | 44 : 1

Modifications Necessary to `web.xml` File For Handling Additional Servlets

- In addition to modifying our `index.html` (homepage) file to include descriptors for launching the additional WelcomeServlet2 and WelcomeServlet3 servlets, we also need to modify the `web.xml` configuration file to register these servlets with Tomcat.
- We will need to include servlet definitions and servlet mappings for both WelcomeServlet2 and WelcomeServlet3.
- The additional statements that must be included in this file are shown on the next slide.
- You must also include the Java class files for these servlets in the `classes` folder.

```
<servlet>
    <servlet-name>welcome2</servlet-name>

            <description>
              A more personal welcome servlet
            </description>

            <servlet-class>
              WelcomeServlet2
            </servlet-class>
  </servlet>

  <servlet>
    <servlet-name>welcome3</servlet-name>

            <description>
              A more personal welcome serlvet - using a post action
            </description>

            <servlet-class>
              WelcomeServlet3
            </servlet-class>
  </servlet>
```

Servlet descriptions

Servlet mappings

```
<servlet-mapping>
            <servlet-name>welcome2</servlet-name>
            <url-pattern>/welcome2</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
            <servlet-name>welcome3</servlet-name>
            <url-pattern>/welcome3</url-pattern>
  </servlet-mapping>
```


Redirecting Requests to Other Resources

- Sometimes it is useful to redirect a request to a different resource. For example, a servlet's job might be to determine the type of the client's browser and redirect the request to a Web page that was designed specifically for that browser.
- The same technique is used when redirecting browsers to an error page when the handling of a request fails.
- Shown on the next two pages is the source code for a `ReDirectionServlet` (available on the course website) which redirects the client to another resource selected from a list of resources.

ReDirectionServlet.java

Java - Servlets/src/ReDirectionServlet.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

SQLGUIClient.java | ResultSetTableModelK.java | WelcomeServlet.java | *WelcomeServlet3.java | ReDirectionServlet.java

```
1  // Redirecting a client to a different Web page.
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4  import java.io.*;
5
6  public class ReDirectionServlet extends HttpServlet {
7
8     // process "get" request from client
9     protected void doGet( HttpServletRequest request,
10       HttpServletResponse response )
11          throws ServletException, IOException
12    {
13       String location = request.getParameter( "page" );
14
15       if ( location != null )
16
17          if ( location.equals( "CNT4714" ) )
18             response.sendRedirect( "https://idp-prod.cc.ucf.edu/idp/Authn/UserPassword" );
19          else
20             if ( location.equals( "welcome1" ) )
21                response.sendRedirect( "welcome1" );
22                 else
23                     if ( location.equals ("cyclingnews") )
24                         response.sendRedirect( "http://www.cyclingnews.com" );
25                      else
26                        if ( location.equals ( "error" ) )
27                            response.sendRedirect( "error"  );
28
29       // code that executes only if this servlet does not redirect the user to another page
30
31       response.setContentType( "text/html" );
32       PrintWriter out = response.getWriter();
```

sendRedirect is a method within the HTTPServletResponse Interface. The string parameter is utilized as the URL to which the client's request is redirected.

Writable | Smart Insert | 53 : 1

ReDirectionServlet.java
continued

Java - Servlets/src/ReDirectionServlet.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

SQLGUIClient.java | ResultSetTableModelK.java | WelcomeServlet.java | *WelcomeServlet3.java | ReDirectionServlet.java

```
                else
                    if ( location.equals ("cyclingnews") )
                        response.sendRedirect( "http://www.cyclingnews.com" );
                     else
                       if ( location.equals ( "error" ) )
                           response.sendRedirect( "error"  );

        // code that executes only if this servlet does not redirect the user to another page

        response.setContentType( "text/html" );
        PrintWriter out = response.getWriter();

        // start HTML document
        out.println( "<!DOCTYPE html\">" );
        // head section of document
        out.println( "<head>" );
        out.println( "<title>Invalid page</title>" );
        out.println( "</head>" );
        // body section of document
        out.println( "<body>" );
        out.println( "<h1>Invalid page requested</h1>" );
        out.println( "<p><a href = " + "RedirectionServlet.html\">" );
        out.println( "Click here for more details</a></p>" );
        out.println( "</body>" );
        // end HTML document
        out.println( "</html>" );
        out.close();  // close stream to complete the page
    }
}
```

Writable | Smart Insert | 4 : 18

C:\Program Files\Apache Software Foundation\Tomcat 8.0_Tomcat826\webapps\CNT4714\ReDirectionServlet.html - Notepad...

File Edit Search View Encoding Language Settings Macro

ReDirectionServlet.html

Deposit.java | web.xml | WelcomeServlet2.java | WelcomeServlet2.html | WelcomeServlet3.html | ReDirectionServlet.html

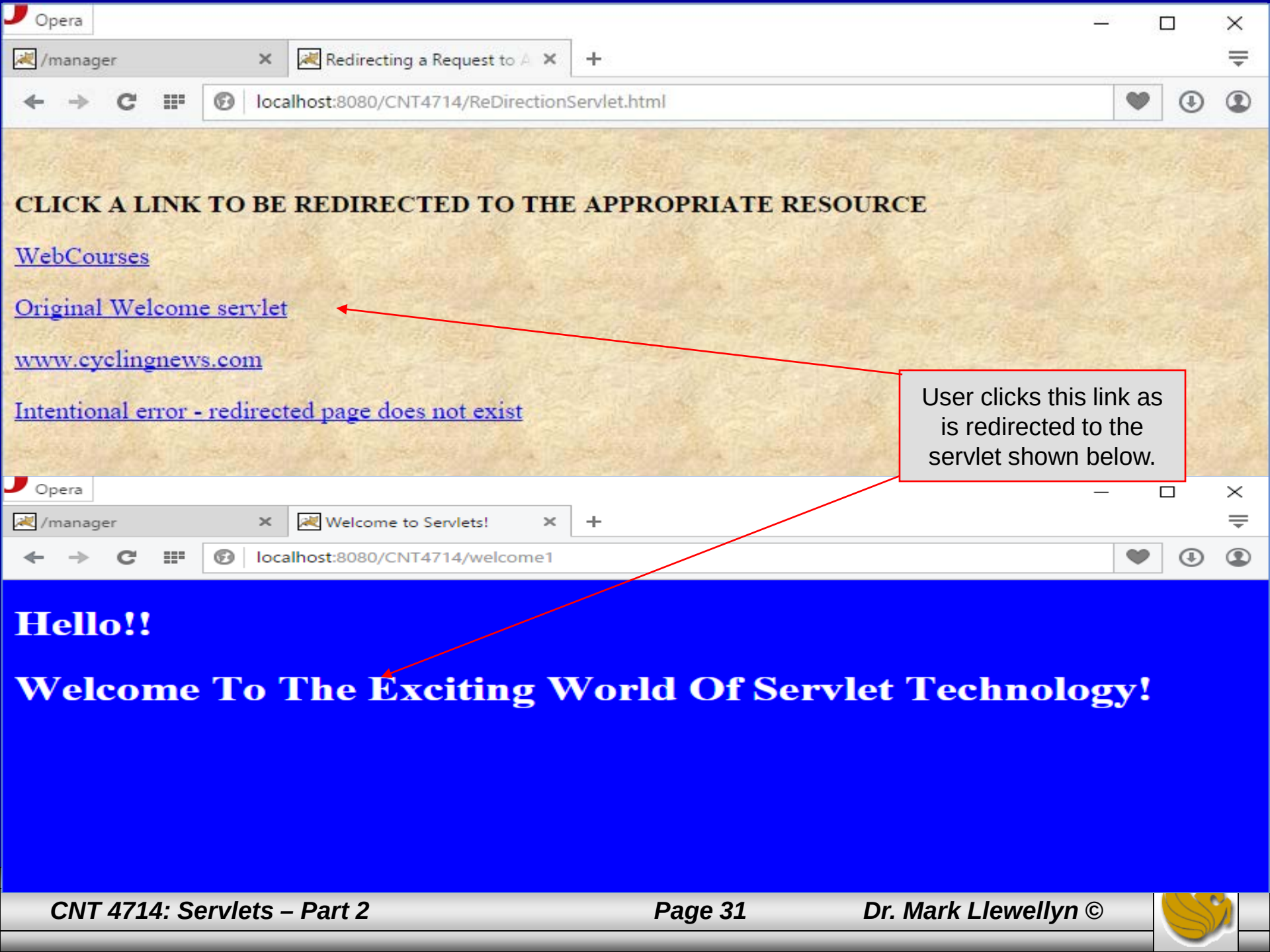
```
1  <!DOCTYPE html>
2
3  <!-- ReDirectionServlet.html -->
4  <html lang="en">
5  <head>
6     <title>Redirecting a Request to Another Site</title>
7     <meta charset="utf-8" />
8  </head>
9
10 <body>
11    <font size = 4><body bgcolor=white background=images/background.jpg lang=EN-US
12       link=blue vlink=blue  style='tab-interval:.5in'>
13    <br>
14    <p><b>CLICK A LINK TO BE REDIRECTED TO THE APPROPRIATE RESOURCE</b></p>
15    <p>
16    <a href = "/CNT4714/redirect?page=CNT4714">WebCourses</a><p>
17    <a href = "/CNT4714/redirect?page=welcome1">Original Welcome servlet</a><p>
18    <a href = "/CNT4714/redirect?page=cyclingnews">www.cyclingnews.com</a><p>
19    <a href="/CNT4714/redirect?page=error">Intentional error - redirected page does not exist</a
20    </p>
21 </body>
22
23 </html>
```

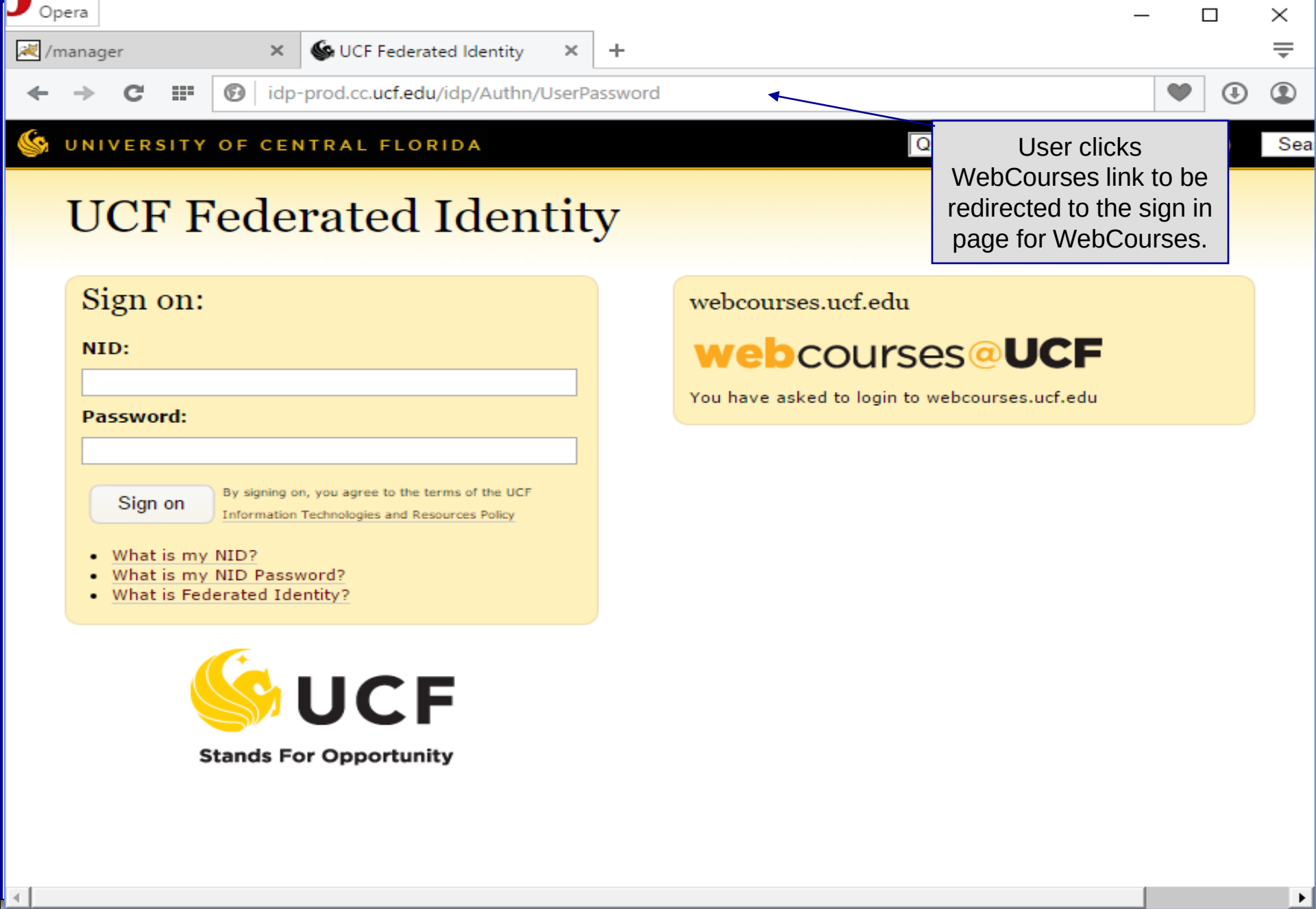
Hyper Text Markup | length : 751 | lines : 27 | Ln : 1 | Col : 1 | Sel : 0 | 0 | Macintosh | UTF-8 w/o BOM | INS

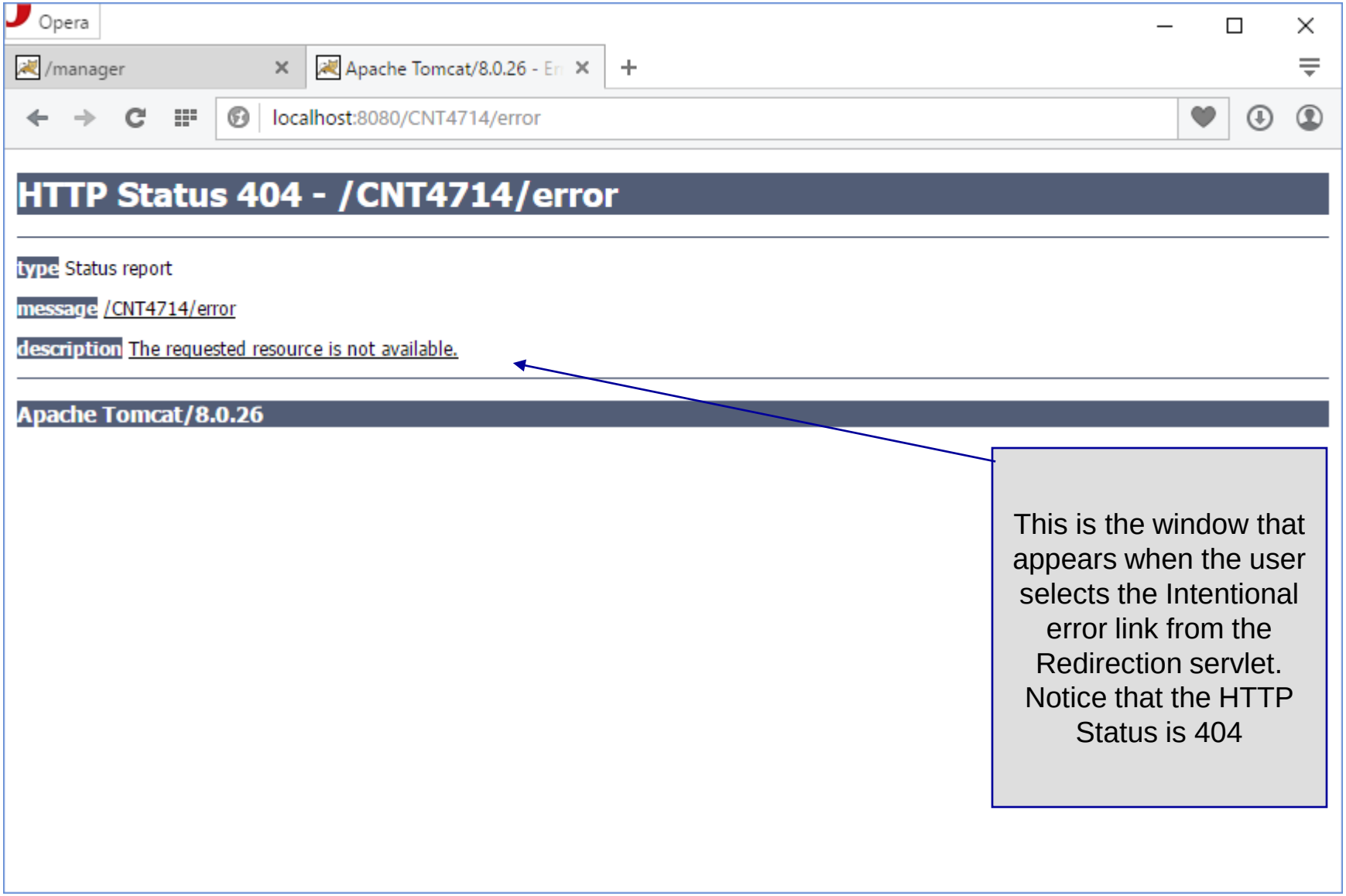
The servlet and servlet-mapping Portions Of web.xml Modified To Handle The ReDirectionServlet

```
<servlet>
       <servlet-name>redirect</servlet-name>
       <description>
          A redirection servlet.
       </description>
       <servlet-class>
          ReDirectionServlet
       </servlet-class>
   </servlet>


 <servlet-mapping>
      <servlet-name>redirect</servlet-name>
      <url-pattern>/redirect</url-pattern>
   </servlet-mapping>
```


Opera
/manager
Redirecting a Request to A
localhost:8080/CNT4714/ReDirectionServlet.html
CLICK A LINK TO BE REDIRECTED TO THE APPROPRIATE RESOURCE
WebCourses
Original Welcome servlet
www.cyclingnews.com
Intentional error - redirected page does not exist
User clicks this link as is redirected to the servlet shown below.
Opera
/manager
Welcome to Servlets!
localhost:8080/CNT4714/welcome1
Hello!!
Welcome To The Exciting World Of Servlet Technology!


Opera
/manager
UCF Federated Identity
idp-prod.cc.ucf.edu/idp/Authn/UserPassword
UNIVERSITY OF CENTRAL FLORIDA
User clicks WebCourses link to be redirected to the sign in page for WebCourses.
UCF Federated Identity
Sign on:
NID:
Password:
Sign on
By signing on, you agree to the terms of the UCF Information Technologies and Resources Policy
What is my NID?
What is my NID Password?
What is Federated Identity?
webcourses.ucf.edu
webcourses@UCF
You have asked to login to webcourses.ucf.edu
UCF
Stands For Opportunity



This is the window that appears when the user selects the Intentional error link from the Redirection servlet. Notice that the HTTP Status is 404

Session Tracking and Servlets

- Many e-businesses personalize users' browsing experiences, tailoring web pages to their users' individual preferences and allowing them to bypass irrelevant content.
- This is typically done by tracking the user's movement through the Internet and combining that data with information provided by the users themselves, such as billing information, interests and hobbies, among other things.
- Personalization of the Internet has become rather commonplace today with many sites even allowing their clients the ability to customize their homepage to fit individual user likes/needs (see MSN.com, CNN.com or numerous other sites).
- This increase in personalization of the Internet has also given rise to the problems of privacy invasion. What happens when the e-business to which you give your personal data sells or gives that data to another organization without your knowledge?

Session Tracking and Servlets (cont.)

- As we have discussed before, the request/response mechanism of the Internet is based on HTTP.
- Unfortunately, HTTP is a stateless protocol – it does not support persistent information that could help a web server determine that a request is from a particular client.
- As far as a web server is concerned, every request could be from the same client or every request could be from a different client. Thus, sites like MSN.com and CNN.com need a mechanism to identify individual clients.
- To help the server distinguish between clients, each client must identify itself to the server. There are a number of popular techniques for distinguishing between clients.
- Two common techniques are cookies and session tracking we'll look at both of these mechanisms. Two other techniques are hidden forms and URL-rewriting.

Cookies

- Cookies are a popular technique for customizing web pages. Browsers can store cookies on the user's computer for retrieval later in the same browsing session or in future browsing sessions.
- For example, cookies are used in on-line shopping applications to store unique identifiers for the users. When users add items to their on-line shopping carts or perform other tasks resulting in a request to the web server, the server receives cookies containing unique identifiers for each user. The server then uses the unique identifier to locate the shopping carts and perform any necessary processing.
- Cookies are also used to indicate the client's shopping preferences. When the servlet receives the client's nest communication, the servlet can examine the cookie(s) it sent to the client in a previous communication, identify the client's preferences and immediately display products of interest to that particular client.

Cookies (cont.)

- Cookies are text-based data that are sent by servlets (or other similar server-side technologies like JSPs and PHP that we will see later) as part of responses to clients.
- Every HTTP-based interaction between a client and a server includes a header containing information about the request (when the communication is from the client to the server) or information about the response (when the communication is from the server to the client).
- When an `HTTPServlet` receives a request the header includes information such as the request type (e.g., `get` or `post`) and the cookies that are sent by the server to be stored on the client machine. When the server prepares its response, the header information includes any cookies the server wants to store on the client computer and other information such as the MIME type of the response.

Cookies (cont.)

- Depending on the maximum age of a cookie, the web browser either maintains the cookie for the duration of the browsing session (i.e., until the user closes the web browser) or stores the cookie on the client computer for future use.
- When a browser requests a resource from a server, cookies that were previously sent to the client by that server are returned to the server as part of the request formulated by the browser.
- Cookies are deleted automatically when they expire (i.e., reach their maximum age).
- Browsers that support cookies must be able to store a minimum of 20 cookies per web site and 300 cookies per user. Browsers may limit the cookie size to 4K.
- Each cookie stored on the client contains a domain. The browser sends a cookie only to the domain stored in the cookie.

Cookies (cont.)

- The next example shows how a cookie can be used to differentiate between first-time and repeat visitors to our servlet. To do this our servlet needs to check for the existence of a uniquely named cookie; if it is there, the client is a repeat visitor. If the cookie is not there, the visitor is a newcomer.

- This example, will use a cookie for this purpose. The cookie will be named "RepeatVisitor".

- Recall that by default, a cookie exists only during the current browsing session. The cookie in this example is a persistent cookie with a lifetime of 1 minute (see the code on the next page).

Java - Servlets/src/RepeatVisitor.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

SQLGUIClient.j... | ResultSetTable... | WelcomeServlet... | *WelcomeServle... | ReDirectionSer... | RepeatVisitor.j...

```
1  //package coreservlets;
2
3  import java.io.*;
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6
7  // Servlet that says "Welcome aboard" to first-time
8  //  visitors and "Welcome back" to repeat visitors.
9
10 public class RepeatVisitor extends HttpServlet {
11   public void doGet(HttpServletRequest request,
12                     HttpServletResponse response)
13       throws ServletException, IOException {
14     boolean newbie = true;
15     Cookie[] cookies = request.getCookies();
16     if (cookies != null) {
17       for(int i=0; i<cookies.length; i++) {
18         Cookie c = cookies[i];
19         if ((c.getName().equals("repeatVisitor")) &&
20             // Could omit test and treat cookie name as a flag
21             (c.getValue().equals("yes"))) {
22           newbie = false;
23           break;
24         }
25       }
26     }
```

If the cookie name is "RepeatVisitor" and the value of the cookie is "yes" then the visitor has been here before.

Writable | Smart Insert | 25 : 8

Java - Servlets/src/RepeatVisitor.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

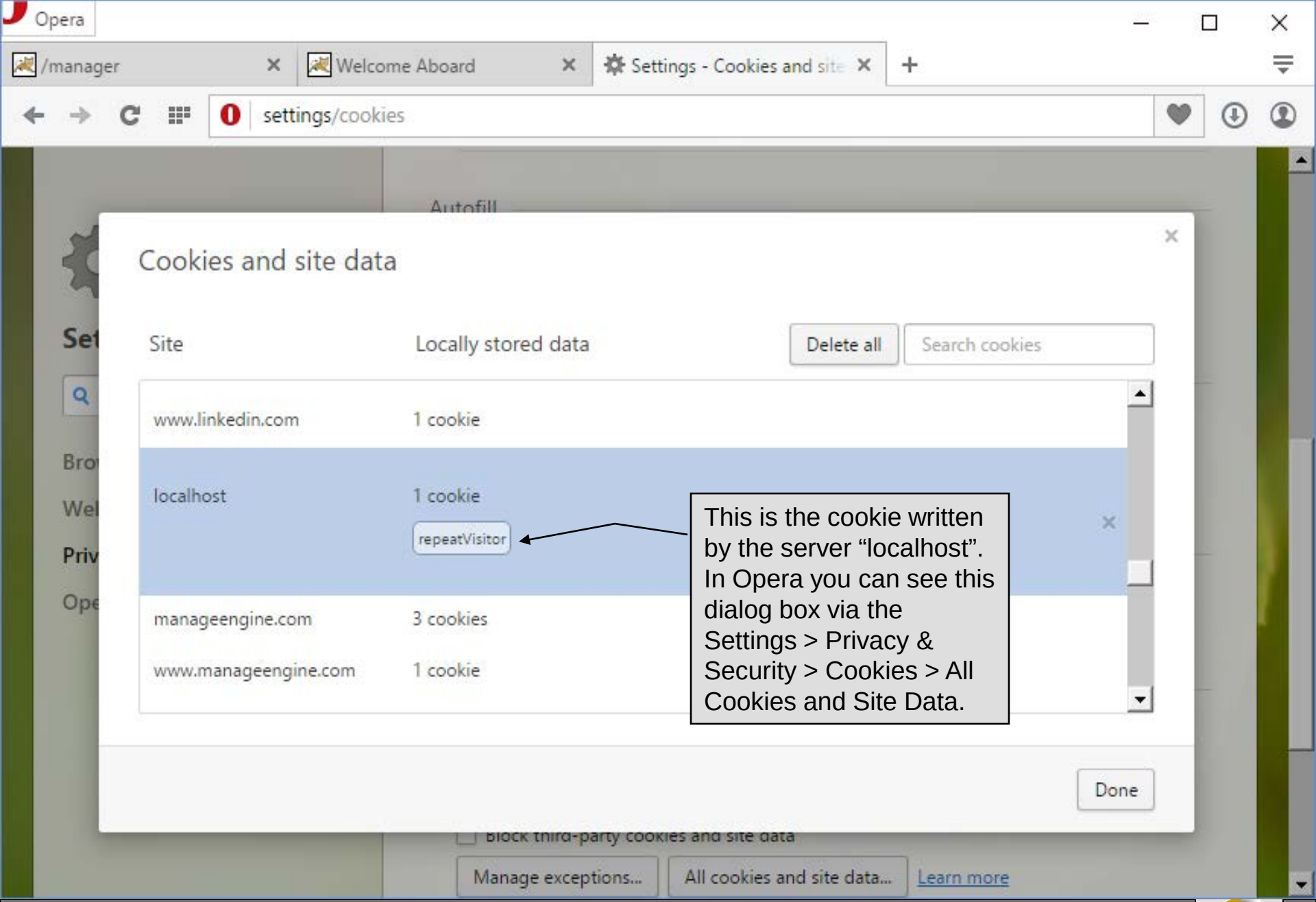
Quick Access | Java

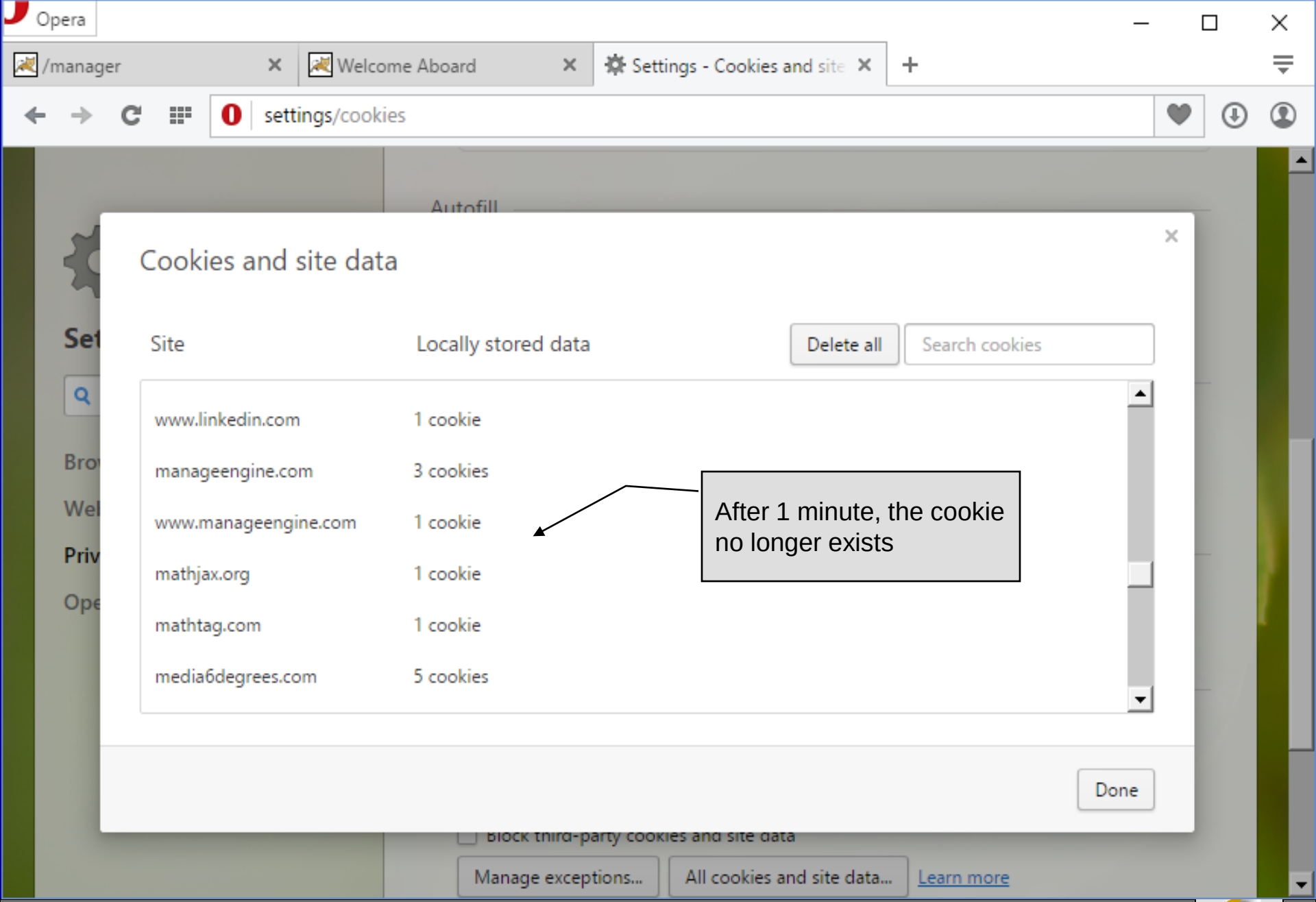
SQLGUIClient.j... | ResultSetTable... | WelcomeServlet... | *WelcomeServle... | ReDirectionSer... | RepeatVisitor.j...

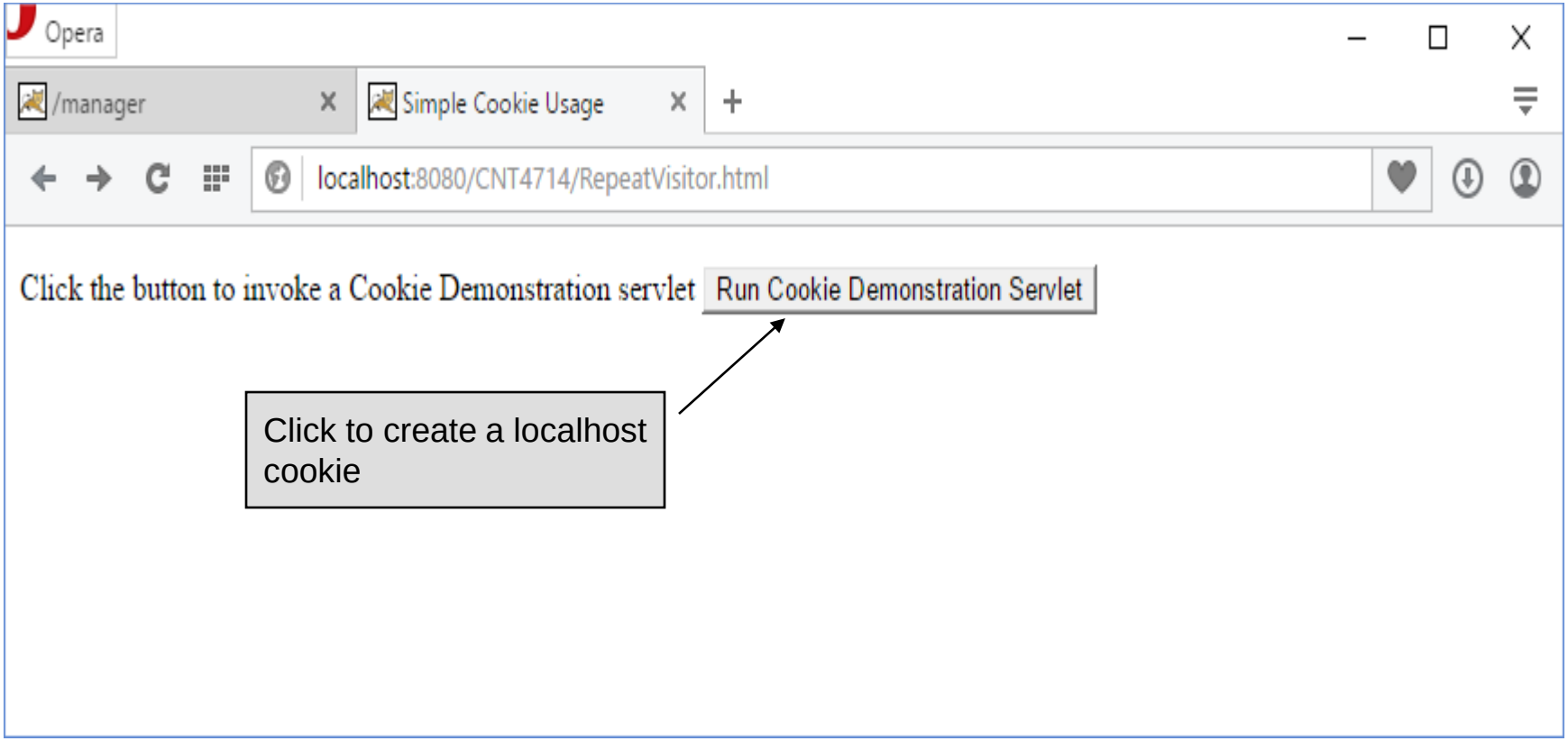
```
26      }
27      String title;
28      if (newbie) {
29        Cookie returnVisitorCookie =
30          new Cookie("repeatVisitor", "yes");
31       // returnVisitorCookie.setMaxAge(60*60*24*365); // 1 year
32        returnVisitorCookie.setMaxAge(60); //cookie expires in 1 minute
33        response.addCookie(returnVisitorCookie);
34        title = "Welcome Aboard";
35      } else {
36        title = "Welcome Back";
37      }
38      response.setContentType("text/html");
39      PrintWriter out = response.getWriter();
40      String docType =
41        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
42        "Transitional//EN\">\n";
43           out.println ("<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=
44      out.println ("<body style='tab-interval:.5in'>");
45       out.println ("<font size = 5>");
46       out.println ("<br>");
47
48      out.println(docType +
49                  "<HTML>\n" +
50                  "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
51                  "<BODY BGCOLOR=\"#FDF5E6\">\n" +
52                  "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
53                  "</BODY></HTML>");
54    }
55  }
```

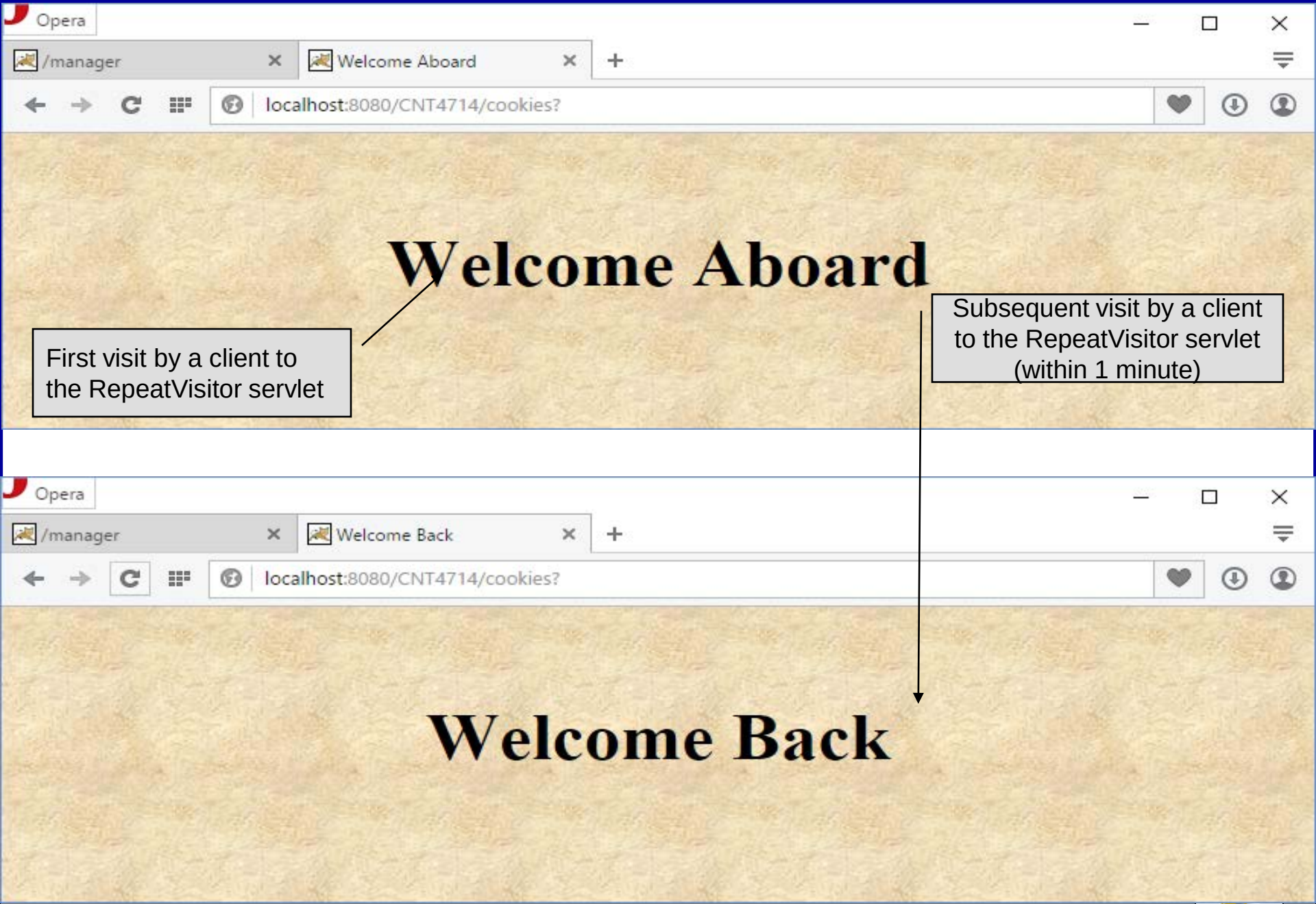
Set cookie expiration date and send it to the client.

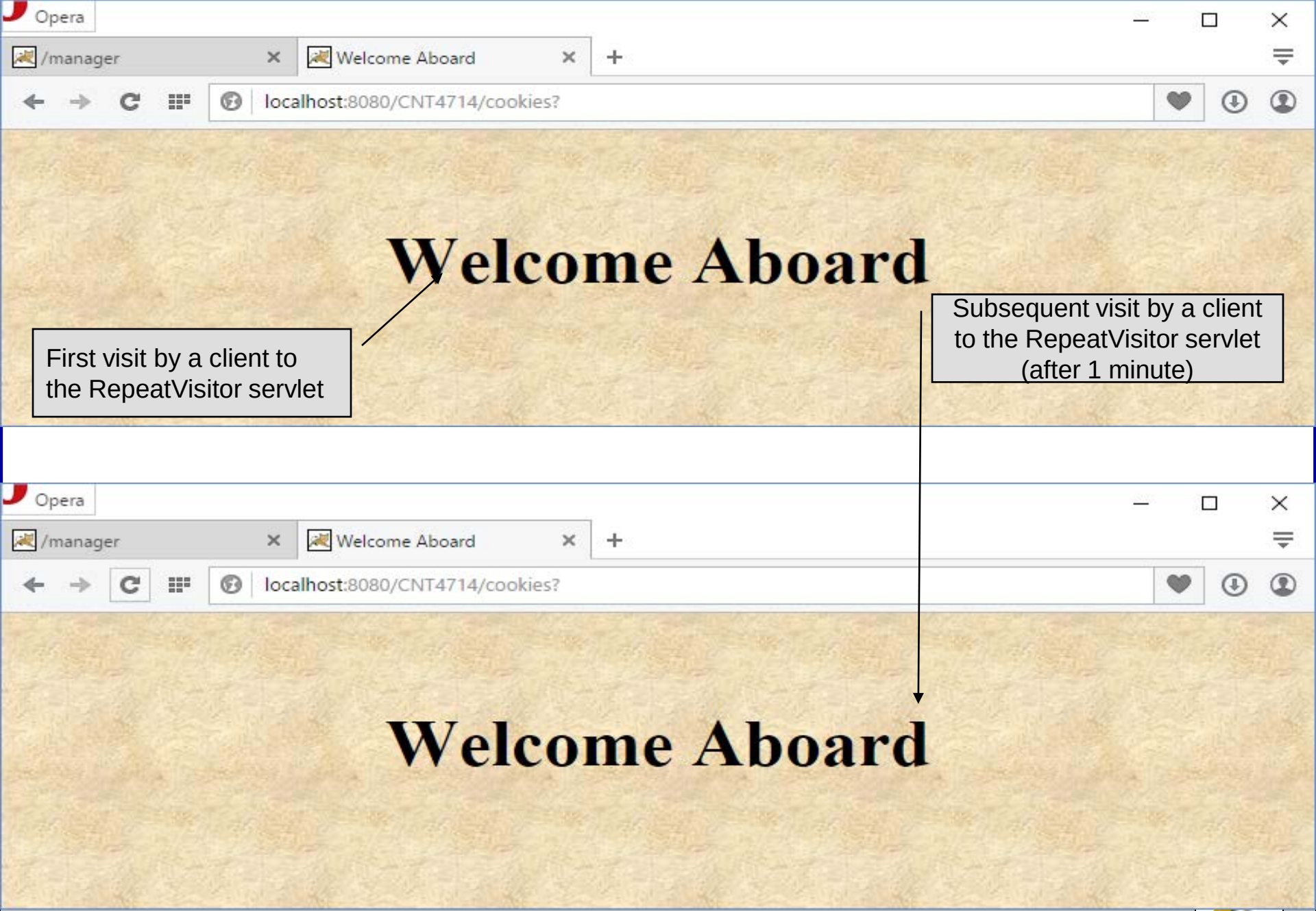
Writable | Smart Insert | 23 : 17


Opera
/manager
Welcome Aboard
Settings - Cookies and site
settings/cookies
Cookies and site data
Site
Locally stored data
Delete all
Search cookies
www.linkedin.com
1 cookie
localhost
1 cookie
repeatVisitor
manageengine.com
3 cookies
www.manageengine.com
1 cookie
This is the cookie written by the server “localhost”. In Opera you can see this dialog box via the Settings > Privacy & Security > Cookies > All Cookies and Site Data.
Done
Manage exceptions...
All cookies and site data...
Learn more


Opera
/manager
Welcome Aboard
Settings - Cookies and site
settings/cookies
Autofill
Cookies and site data
Site
Locally stored data
Delete all
Search cookies
www.linkedin.com
1 cookie
manageengine.com
3 cookies
www.manageengine.com
1 cookie
mathjax.org
1 cookie
mathtag.com
1 cookie
media6degrees.com
5 cookies
After 1 minute, the cookie no longer exists
Done
Block third-party cookies and site data
Manage exceptions...
All cookies and site data...
Learn more


Opera
/manager
Simple Cookie Usage
localhost:8080/CNT4714/RepeatVisitor.html
Click the button to invoke a Cookie Demonstration servlet
Run Cookie Demonstration Servlet
Click to create a localhost cookie


Opera
/manager
Welcome Aboard
localhost:8080/CNT4714/cookies?
Welcome Aboard
First visit by a client to the RepeatVisitor servlet
Subsequent visit by a client to the RepeatVisitor servlet (within 1 minute)
Opera
/manager
Welcome Back
localhost:8080/CNT4714/cookies?
Welcome Back


Opera
/manager
Welcome Aboard
localhost:8080/CNT4714/cookies?
Welcome Aboard
First visit by a client to the RepeatVisitor servlet
Subsequent visit by a client to the RepeatVisitor servlet (after 1 minute)
Opera
/manager
Welcome Aboard
localhost:8080/CNT4714/cookies?
Welcome Aboard

Using Cookies Attributes

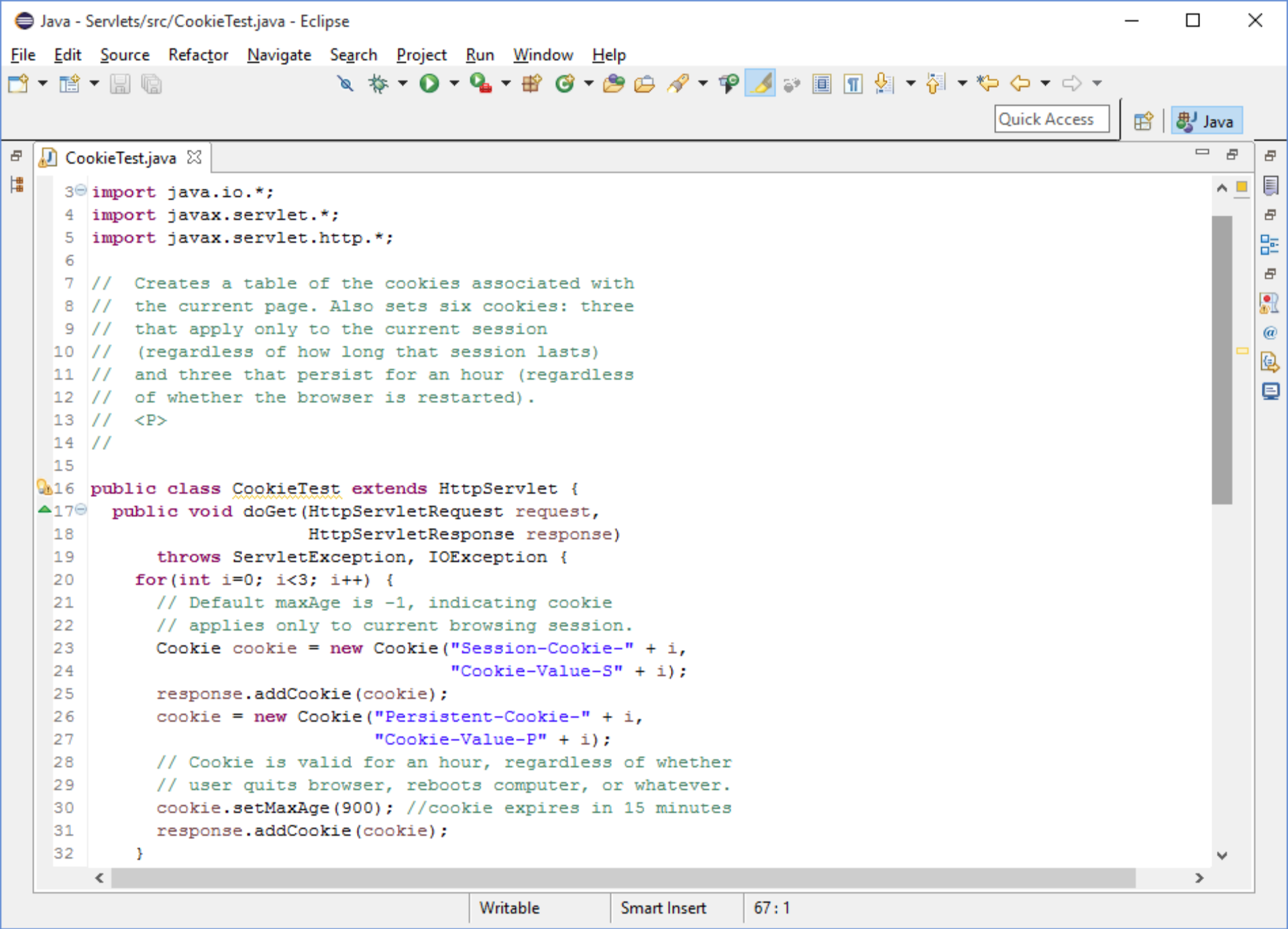
- Before adding the cookie to the outgoing headers, you can set various characteristics of the cookie by using the following `set` methods.
- Although each `set` method has a corresponding `get` method to retrieve the attribute value, note that the attributes are part of the header sent from the server to the browser; they are **not** part of the header returned by the browser to the server.
- 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. This means that these attributes are not available in the cookies that you get by means of `request.getCookies`.
- A brief description of the methods for setting and getting cookie attribute values are shown on the next page.

setComment(string) getComment()	Specify or look up a comment associated with the cookie.
setDomain(string) getDomain()	Set or retrieve the domain to which the cookie applies. Normally, the browser returns cookies only to the exact same hostname that sent the cookies.
setMaxAge(int) getMaxAge()	These methods tell how much time (in seconds) should elapse before the cookie expires. A negative value, which is the default, indicates that the cookie will last only for the current browsing session and will not be stored on disk. Specifying a value of 0 instructs the browser to delete the cookie.
getName()	Retrieves the name of the cookie. The name and the value are the two pieces of information which are the most important.
setPath(string) getPath()	Sets or retrieves the path to which the cookie applies. If you do not specify a path, the browser returns the cookie only to URLs in or below the directory containing the page that sent the cookie.
setSecure(boolean) getSecure()	Sets or retrieves the boolean value which indicates whether the cookie should only be sent over encrypted connections. The default is false.
setValue(string) getValue()	Sets or retrieves the value associated with the cookie.

Differentiating Session Cookies From Persistent Cookies

- The next example illustrates the use of cookie attributes by contrasting the behavior of cookies with and without a maximum age.
- This servlet called CookieTest, performs two tasks

1. First the servlet sets six outgoing cookies. Three have no explicit age (i.e., they have a negative value by default), meaning that they will apply only to the current browsing session – until the client restarts the browser. The other three cookies use `setMaxAge` to stipulate that the browser should write them to disk and that they should persist for the next 15 minutes (15 * 60 = 900 seconds), regardless of whether the client restarts the browser or not.
2. Second, the servlet uses request.getCookies to find all the incoming cookies and display their names and values in an HTML table.


Java - Servlets/src/CookieTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
CookieTest.java
3 import java.io.*;
4 import javax.servlet.*;
5 import javax.servlet.http.*;
6
7 // Creates a table of the cookies associated with
8 // the current page. Also sets six cookies: three
9 // that apply only to the current session
10 // (regardless of how long that session lasts)
11 // and three that persist for an hour (regardless
12 // of whether the browser is restarted).
13 // <P>
14 //
15
16 public class CookieTest extends HttpServlet {
17 public void doGet(HttpServletRequest request,
18 HttpServletResponse response)
19 throws ServletException, IOException {
20 for(int i=0; i<3; i++) {
21 // Default maxAge is -1, indicating cookie
22 // applies only to current browsing session.
23 Cookie cookie = new Cookie("Session-Cookie-" + i,
24 "Cookie-Value-S" + i);
25 response.addCookie(cookie);
26 cookie = new Cookie("Persistent-Cookie-" + i,
27 "Cookie-Value-P" + i);
28 // Cookie is valid for an hour, regardless of whether
29 // user quits browser, reboots computer, or whatever.
30 cookie.setMaxAge(900); //cookie expires in 15 minutes
31 response.addCookie(cookie);
32 }
Writable
Smart Insert
67 : 1

Java - Servlets/src/CookieTest.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

CookieTest.java

```
32      }
33      response.setContentType("text/html");
34      PrintWriter out = response.getWriter();
35      String docType =
36        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
37        "Transitional//EN\">\n";
38      String title = "Active Cookies";
39       out.println ("<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=blue
40      out.println ("<body style='tab-interval:.5in'>");
41       out.println ("<font size = 5>");
42       out.println ("<br>");
43      out.println(docType +
44                  "<HTML>\n" +
45                  "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
46                  "<BODY BGCOLOR=\"#FDF5E6\">\n" +
47                  "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
48                  "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
49                  "<TR BGCOLOR=\"#FFAD00\">\n" +
50                  "  <TH>Cookie Name\n" +
51                  "  <TH>Cookie Value");
52      Cookie[] cookies = request.getCookies();
53      if (cookies == null) {
54        out.println("<TR><TH COLSPAN=2>No cookies");
55      } else {
56        Cookie cookie;
57        for(int i=0; i<cookies.length; i++) {
58          cookie = cookies[i];
59          out.println("<TR>\n" +
60                      "  <TD>" + cookie.getName() + "\n" +
61                      "  <TD>" + cookie.getValue());
62        }
63      }
64      out.println("</TABLE></BODY></HTML>");
65    }
66  }
```

Writable | Smart Insert | 4 : 24


Opera
/manager
Session and Persistent Coo
Settings - Cookies and site
localhost:8080/CNT4714/CookieTest.html
Click the button to invoke a Cookie Test servlet
Run Cookie Test Servlet
Client clicks the Run Cookie Test Servlet button from HomePage
Opera
/manager
Active Cookies
Settings - Cookies and site
localhost:8080/CNT4714/cookietest?
Active Cookies
Cookie Name
Cookie Value
No cookies
Results of initial visit to the CookieTest servlet.
CookieTest Servlet response to client indicates that initially there are no active cookies.
Same results would occur if you waited more than 15 minutes and then revisited the CookieTest servlet in a new browser session.

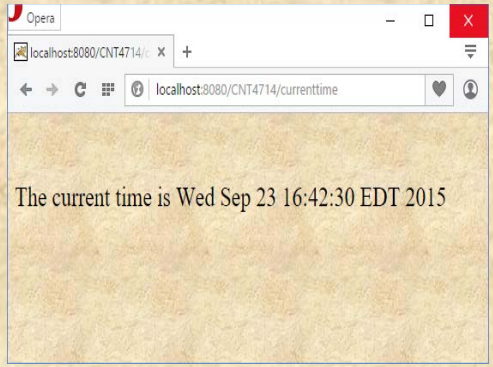
Opera

/manager | Active Cookies | Settings - Cookies and site

localhost:8080/CNT4714/cookietest?

Active Cookies

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

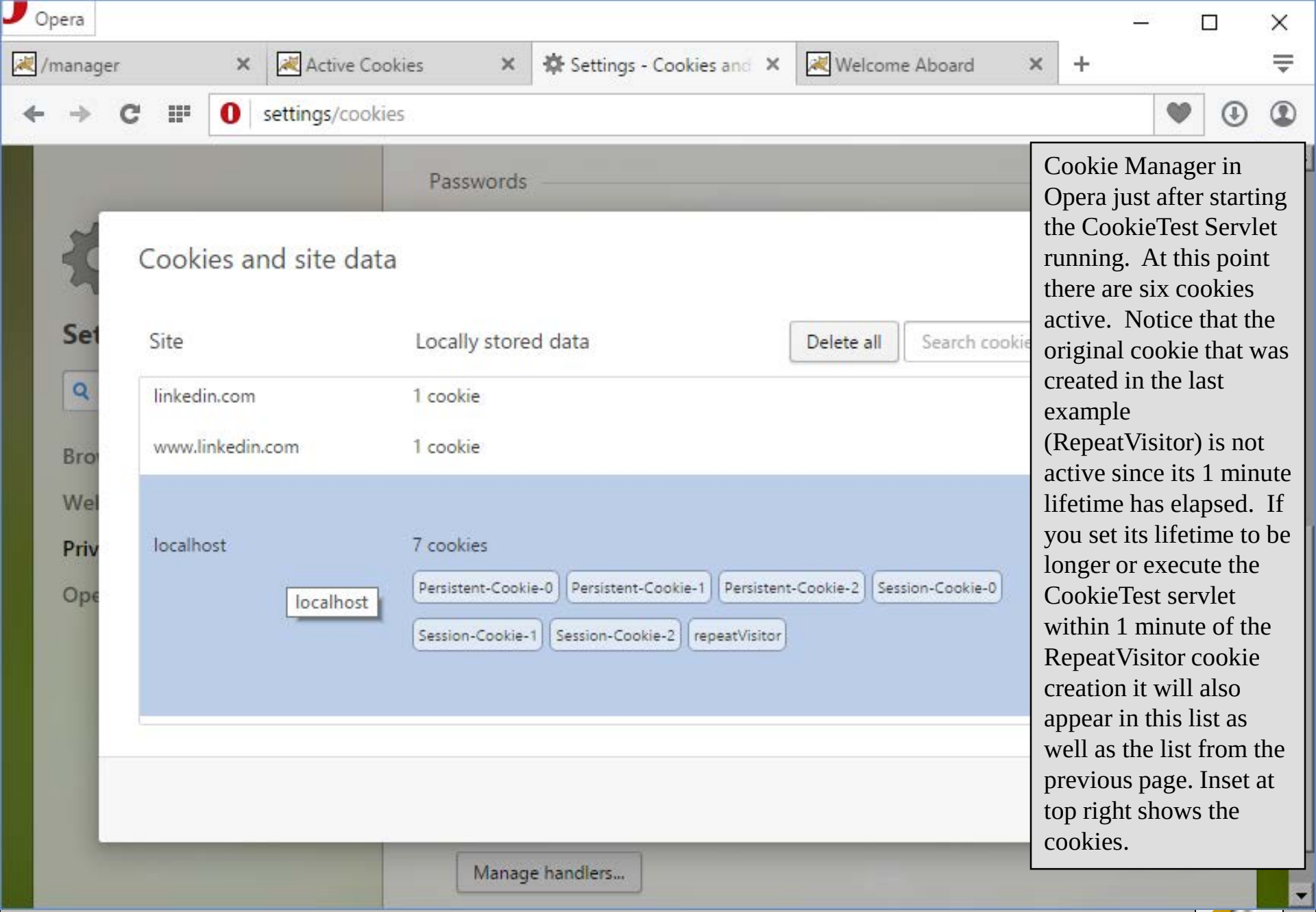


Results of revisiting the CookieTest servlet in the same browser session within 15 minutes of the very first visit.

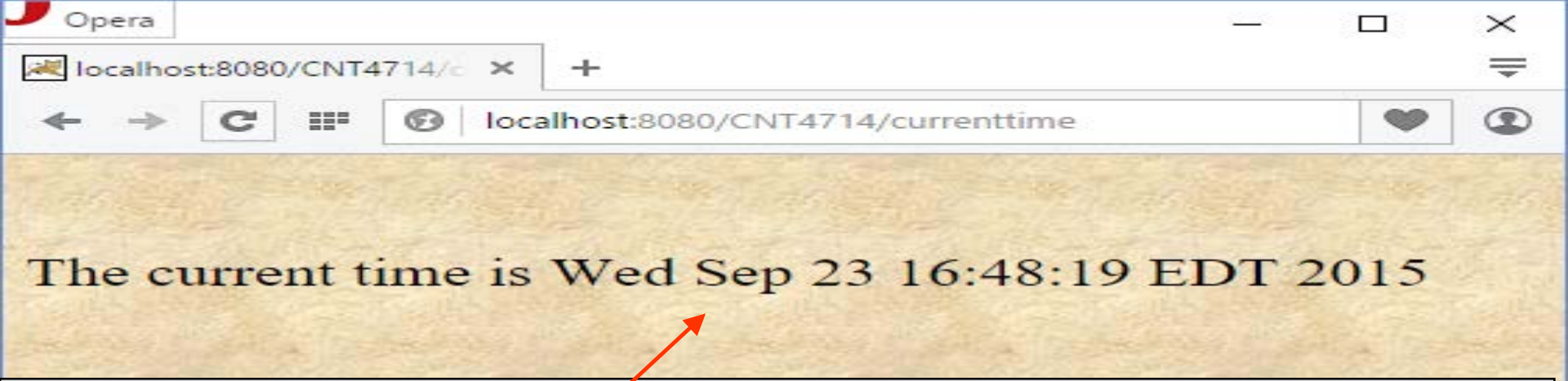
After client runs the servlet the first time 3 persistent cookies and 3 session cookies are created. At this point there are six cookies active. Notice that the original cookie that was created in the last example (RepeatVisitor) is not active since its 1 minute lifetime has elapsed. If you set its lifetime to be longer or execute the CookieTest servlet within 1 minute of the RepeatVisitor cookie creation it will also appear in this list as well as the list from the previous page (see next page).

Notice the time of day.


Opera
/manager
Active Cookies
Settings - Cookies and
Welcome Aboard
localhost:8080/CNT4714/cookietest?
Active Cookies
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
repeatVisitor
yes



Cookie Manager in Opera just after starting the CookieTest Servlet running. At this point there are six cookies active. Notice that the original cookie that was created in the last example (RepeatVisitor) is not active since its 1 minute lifetime has elapsed. If you set its lifetime to be longer or execute the CookieTest servlet within 1 minute of the RepeatVisitor cookie creation it will also appear in this list as well as the list from the previous page. Inset at top right shows the cookies.

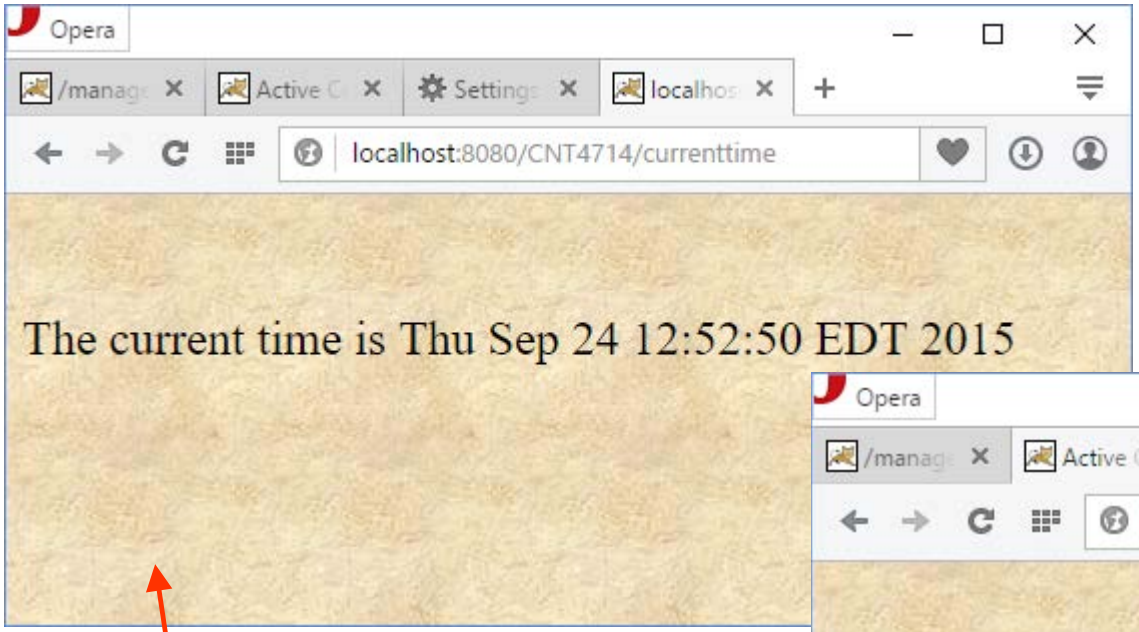


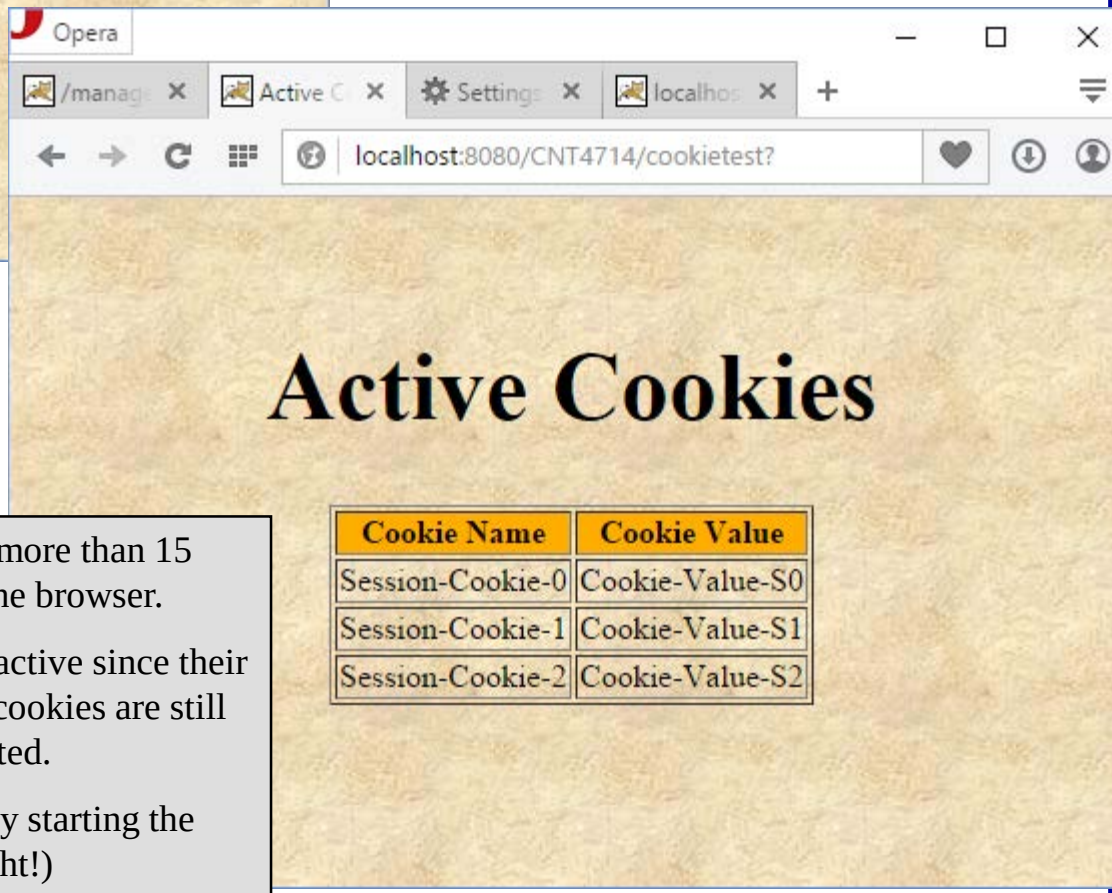
Results of revisiting the CookieTest servlet using a new browser session within 15 minutes of the first visit (in the earlier browser session. Notice that the only cookies which are now active are the persistent cookies (the ones with the 15 minute lifetime).

Notice the time of day is less than 15 minutes after starting servlet.

Active Cookies

Cookie Name	Cookie Value
Persistent-Cookie-0	Cookie-Value-P0
Persistent-Cookie-1	Cookie-Value-P1
Persistent-Cookie-2	Cookie-Value-P2

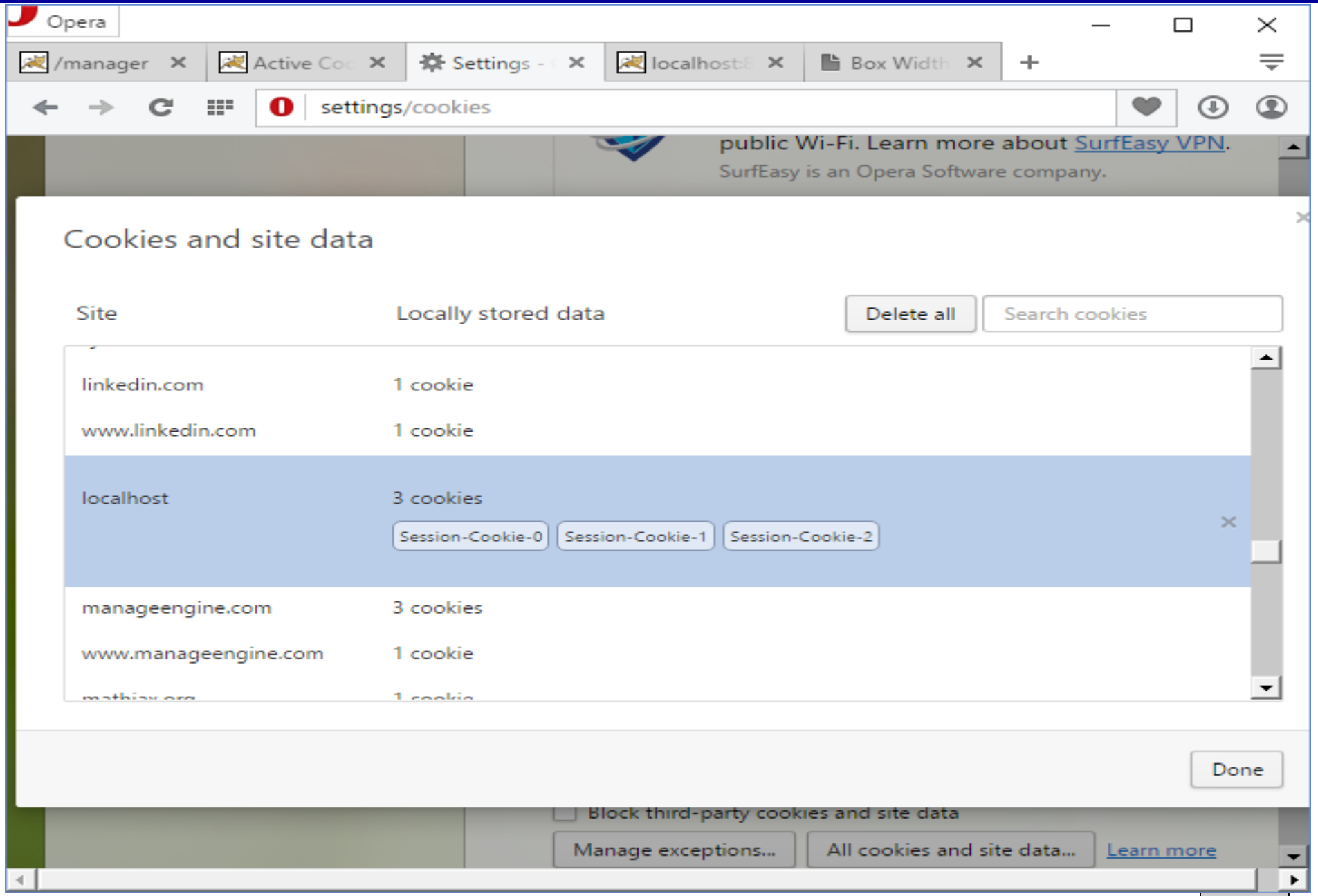




Result of revisiting the CookieTest servlet after more than 15 minutes since the last visit but without closing the browser.

In this case the persistent cookies are no longer active since their maximum age has been exceeded. The session cookies are still active since the browser session has not terminated.

Note the time of day is >15 minutes after initially starting the servlet. (I actually let the session persist overnight!)


Opera
/manager
Active Coc
Settings -
localhost:8
Box Width
settings/cookies
public Wi-Fi. Learn more about SurfEasy VPN.
SurfEasy is an Opera Software company.
Cookies and site data
Site
Locally stored data
Delete all
Search cookies
linkedin.com
1 cookie
www.linkedin.com
1 cookie
localhost
3 cookies
Session-Cookie-0
Session-Cookie-1
Session-Cookie-2
manageengine.com
3 cookies
www.manageengine.com
1 cookie
Done
Block third-party cookies and site data
Manage exceptions...
All cookies and site data...
Learn more

Modifying Cookie Values

- In the previous examples, we've sent a cookie to the user only on the first visit to the servlet. Once the cookie had a value, we never changed it.
- This approach of a single cookie value is quite common since cookies frequently contain nothing but unique user identifiers: all the real user data is stored in a database – the user identifier is merely the database key.
- But what if you would like to periodically change the value of a cookie?
- To replace a previous cookie value, send the same cookie name with a different cookie value. If you actually use the incoming `Cookie` objects, don't forget to do `response.addCookie`; merely calling `setValue` is not sufficient.

Modifying Cookie Values (cont.)

- You also need to reapply any relevant cookie attributes by calling `setMaxAge`, `setPath`, etc. – remember that cookie attributes are not specified for incoming cookies.
- Reapplying these attributes means that reusing the incoming `Cookie` object saves you very little, so many developers don't bother to use the incoming `Cookie` object.
- The next example illustrates modifying a cookie value by maintaining a count of the number of times your web browser visits a servlet named `ClientAccessCounts`.
- The code for ClientAccessCounts is shown on the next page with some results shown on the following pages.

Java - Servlets/src/ClientAccessCounts.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

ClientAccessCounts.java | CookieUtilities.java

```
1  // Servlet that prints per-client access counts.
2  // Note: this class requires the CookieUtilities class
3
4  import java.io.*;
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7
8
9  public class ClientAccessCounts extends HttpServlet {
10   public void doGet(HttpServletRequest request,
11                     HttpServletResponse response)
12       throws ServletException, IOException {
13     String countString =
14       CookieUtilities.getCookieValue(request, "accessCount", "1");
15     int count = 1;
16     try {
17       count = Integer.parseInt(countString);
18     } catch(NumberFormatException nfe) { }
19     Cookie c =
20       new Cookie("accessCount",
21                  String.valueOf(count+1));
22     c.setMaxAge(900);
23     response.addCookie(c);
24     response.setContentType("text/html");
25     PrintWriter out = response.getWriter();
26     String title = "Access Count Servlet";
27     String docType =
```

Read the current cookie value.

Create a new cookie with value 1 greater than current cookie value.

Writable | Smart Insert | 3 : 1

Java - Servlets/src/ClientAccessCounts.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

ClientAccessCounts.java | CookieUtilities.java

```
20        new Cookie("accessCount",
21                   String.valueOf(count+1));
22      c.setMaxAge(900);
23      response.addCookie(c);
24      response.setContentType("text/html");
25      PrintWriter out = response.getWriter();
26      String title = "Access Count Servlet";
27      String docType =
28        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
29        "Transitional//EN\">\n";
30       out.println ("<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=blue
31      out.println ("<body style='tab-interval:.5in'>");
32       out.println ("<font size = 5>");
33       out.println ("<br>");
34      out.println(docType +
35                  "<HTML>\n" +
36                  "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
37                  "<BODY BGCOLOR=\"#FDF5E6\">\n" +
38                  "<CENTER>\n" +
39                  "<H1>" + title + "</H1>\n" +
40                  "<H2>This is visit number " +
41                  count + " by this browser.</H2>\n" +
42                  "</CENTER></BODY></HTML>");
43    }
44  }
45
46
```

Writable | Smart Insert | 5 : 24

Java - Servlets/src/CookieUtilities.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

ClientAccessCounts.java | CookieUtilities.java

```
1  import javax.servlet.*;
2  import javax.servlet.http.*;
3  // Two static methods for use in cookie handling.
4
5  public class CookieUtilities {
6    /** Given the request object, a name, and a default value,
7     *  this method tries to find the value of the cookie with
8     *  the given name. If no cookie matches the name,
9     *  the default value is returned.
10    */
11   public static String getCookieValue
12           (HttpServletRequest request, String cookieName, String defaultValue) {
13     Cookie[] cookies = request.getCookies();
14     if (cookies != null) {
15       for(int i=0; i<cookies.length; i++) {
16         Cookie cookie = cookies[i];
17         if (cookieName.equals(cookie.getName())) {
18           return(cookie.getValue());
19         }
20       }
21     }
22     return(defaultValue);
23   }
24
25   /** Given the request object and a name, this method tries
26    *  to find and return the cookie that has the given name.
27    *  If no cookie matches the name, null is returned.
28    */
29   public static Cookie getCookie(HttpServletRequest request, String cookieName) {
30     Cookie[] cookies = request.getCookies();
31     if (cookies != null) {
32       for(int i=0; i<cookies.length; i++) {
33         Cookie cookie = cookies[i];
34         if (cookieName.equals(cookie.getName())) {
35           return(cookie);
36         }
37       }
38     }
39     return(null);
40   }
41 }
42
```

Writable | Smart Insert | 41 : 2

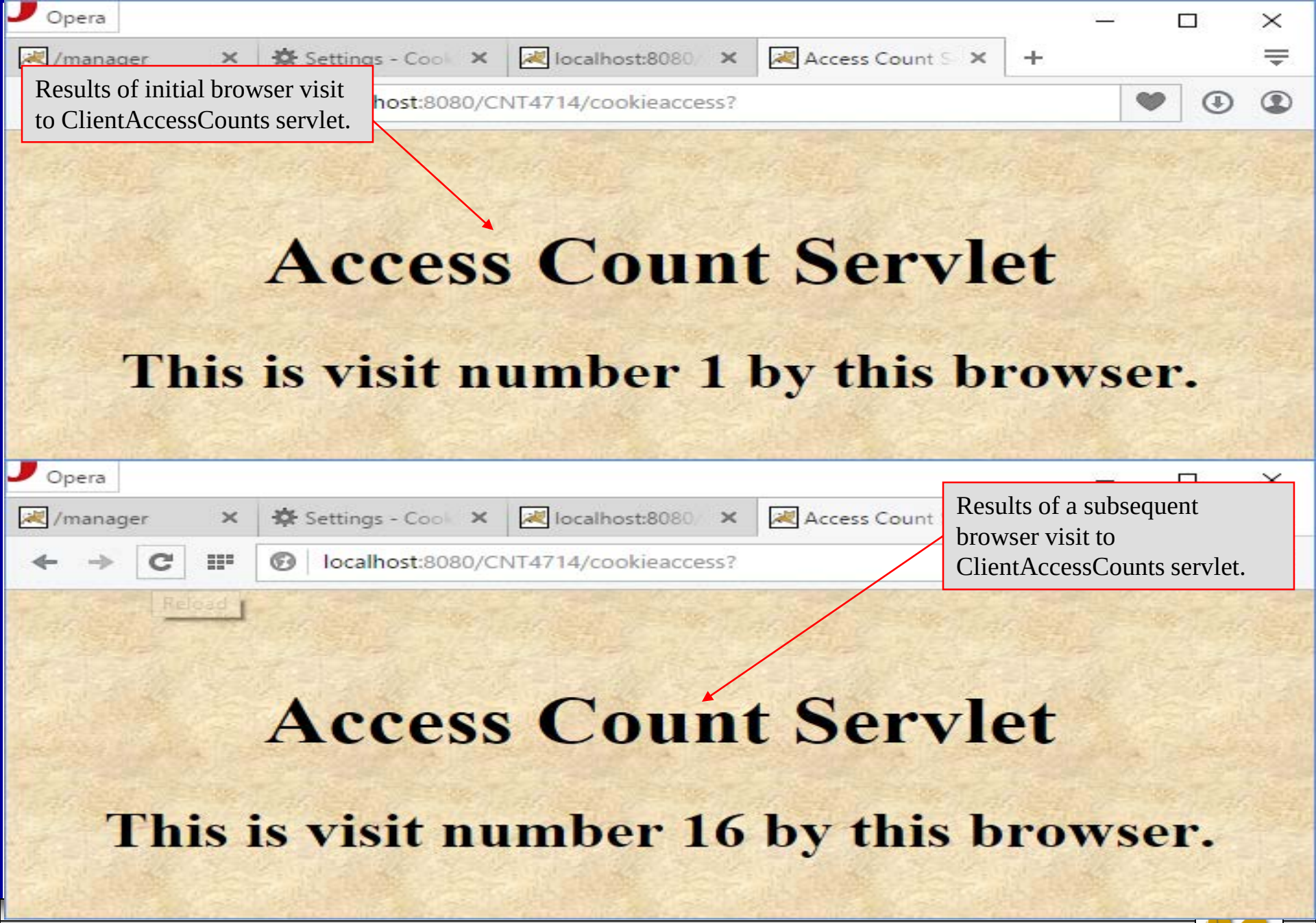
C:\Program Files\Apache Software Foundation\Tomcat 8.0_Tomcat826\webapps\CNT4714\ClientAccessCounts.h...

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

web.xml | ClientAccessCounts.html

```
1  <!-- CookieTest.html -->
2  <html>
3  <head>
4     <title>Client Access Counts</title>
5  </head>
6  <body>
7     <form action = "/CNT4714/cookieaccess" method = "get">
8        <p><label>Click the button to invoke a Client Access Counter servlet
9           <input type = "submit" value = "Run Client Access Counter Servlet" />
10       </label></p>
11    </form>
12 </body>
13 </html>
```

Hyper length : 364 lines : 13 | Ln : 13 Col : 8 Sel : 0 | 0 | Dos\Windows | UTF-8 w/o BOM | INS


Opera
/manager
Settings - Cool
localhost:8080
Access Count S
host:8080/CNT4714/cookieaccess?
Results of initial browser visit to ClientAccessCounts servlet.
Access Count Servlet
This is visit number 1 by this browser.
Opera
/manager
Settings - Cool
localhost:8080
Access Count
localhost:8080/CNT4714/cookieaccess?
Reload
Results of a subsequent browser visit to ClientAccessCounts servlet.
Access Count Servlet
This is visit number 16 by this browser.

Session Tracking

- As we mentioned before, HTTP is a "stateless" protocol: each time a client retrieves a web page, the client opens a separate connection to the web server and the server does not automatically maintain contextual information about the client.
- Even with servers that support persistent (keep-alive) HTTP connections and keep sockets open for multiple client requests that occur in rapid succession, there is no built-in support for maintaining contextual information.
- This lack of context causes a number of difficulties. For example, when clients at an online store add an item to their shopping carts, how does the server know what's already in the carts? Similarly, when clients decide to proceed to checkout, how can the server determine which of the previously created shopping carts are theirs?
- Servlets provide an outstanding session tracking solution: the `HttpSession` API. This high-level interface is built on top of cookies (and URL rewriting). All servers are required to support session tracking with cookies.

Session Tracking (cont.)

- Using sessions in servlets is straightforward and involves four basic steps:

1. Accessing the session object associated with the current request. Call `request.getSession` to get an `HttpSession` object, which is a simple hash table for storing user-specific data.
2. Looking up information associated with a session. Call `getAttribute` on the `HttpSession` object, cast the return value to the appropriate type, and check whether the result is null.
3. Storing information in a session. Use `setAttribute` with a key and a value.
4. Discarding session data. Call `removeAttribute` to discard a specific value. Call `invalidate` to discard an entire session. Call `logout` to log the client out of the web server and invalidate all sessions associated with that user.

Browser Sessions Vs. Server Sessions

- By default, session-tracking is based on cookies that are stored in the browser's memory, not written to disk. Thus, unless the servlet explicitly reads the incoming `JSESSIONID` cookie, sets the maximum age and path, and sends it back out, quitting the browser, results in the session being broken: the client will not be able to access the session again.
- The problem, however, is that the server does not know that the browser was closed and thus the server must maintain the session in memory until the inactive interval has been exceeded.
- To understand this problem consider the following scenario:

Browser Sessions Vs. Server Sessions

- Consider a physical shopping trip to your favorite store. You browse around and put some items into a physical shopping cart, then leave that shopping cart at the end of an aisle while you look for another item. A clerk walks up and sees the shopping cart. Can they reshelve the items in it?
- No – you are probably still shopping and will come back for the cart soon.
- What if you realize that you left your wallet at home – so you go back home to get it. Can the clerk reshelve the items in the cart now?
- Again, no – the clerk presumably does not know that you have left the store.
- So, what can the clerk do? They can keep an eye on the cart, and if nobody claims the cart for some period of time, they can conclude that it is abandon and remove the items in it for reshelving.
- The only exception would be if you explicitly brought the cart to the clerk and told them that you left your wallet at home are have to leave.

Browser Sessions Vs. Server Sessions (cont.)

- The analogous situation in the servlet world is one in which the server is trying to decided if it can throw away your `HttpSession` object.
- Just because you are not currently using the session does not mean the server can throw it away. Maybe you will be back (submit a new request) soon.
- If you quit your browser, thus causing the browser-session-level cookies to be lost, the session is effectively broken. But as with the physical case of getting in your car and leaving, the server does not know that you quit your browser. So the server must still wait for a period of time to see if the session has been abandoned.
- Sessions automatically become inactive when the amount of time between client accesses exceeds the interval specified by `getMaxInactiveInterval`. When this happens, objects stored in the `HttpSession` object are removed.
- The one exception to the "server waits until the sessions time out" rule is if invalidate or logout is called. This is akin to your explicitly telling the clerk that you are leaving, so the server can immediately remove all the items from the session and destroy the session object.

A Servlet That Shows Per-Client Access Counts

- The last example in this section of notes is a servlet that shows basic information about the client's session.
- When the client connects, the servlet uses `request.getSession` either to retrieve the existing session or, if there is no session, to create a new one.
- The servlet then looks for an attribute called `accessCount` of type `Integer`. If it cannot find such an attribute, it uses the value of 0 as the number of previous accesses by the client. This value is then incremented and associated with the session by `setAttribute`.
- Finally, the servlet prints a small HTML table showing information about the session.
- Note that `Integer` is an immutable data structure: once built, it cannot be changed. That means that you have to allocate a new `Integer` object on each request, then use `setAttribute` to replace the old object.

Java - Servlets/src/ShowSession.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

ClientAccessCounts.java | CookieUtilities.java | ShowSession.java

```
1  // Servlet that uses session-tracking to keep per-client
2  //  access counts. Also shows other info about the session.
3
4  import java.io.*;
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7  import java.util.*;
8
9  public class ShowSession extends HttpServlet {
10   public void doGet(HttpServletRequest request,
11                     HttpServletResponse response)
12       throws ServletException, IOException {
13     response.setContentType("text/html");
14     HttpSession session = request.getSession();
15     String heading;
16     Integer accessCount =
17       (Integer)session.getAttribute("accessCount");
18     if (accessCount == null) {
19       accessCount = new Integer(0);
20       heading = "Welcome, Newcomer";
21     } else {
22       heading = "Welcome Back";
23       accessCount = new Integer(accessCount.intValue() + 1);
24     }
25     // Integer is an immutable data structure. So, you
26     // cannot modify the old one in-place. Instead, you
27     // have to allocate a new one and redo setAttribute.
28     session.setAttribute("accessCount", accessCount);
29     PrintWriter out = response.getWriter();
30     String title = "Session Tracking Example";
```

Writable | Smart Insert | 66 : 1

Java - Servlets/src/ShowSession.java - Eclipse

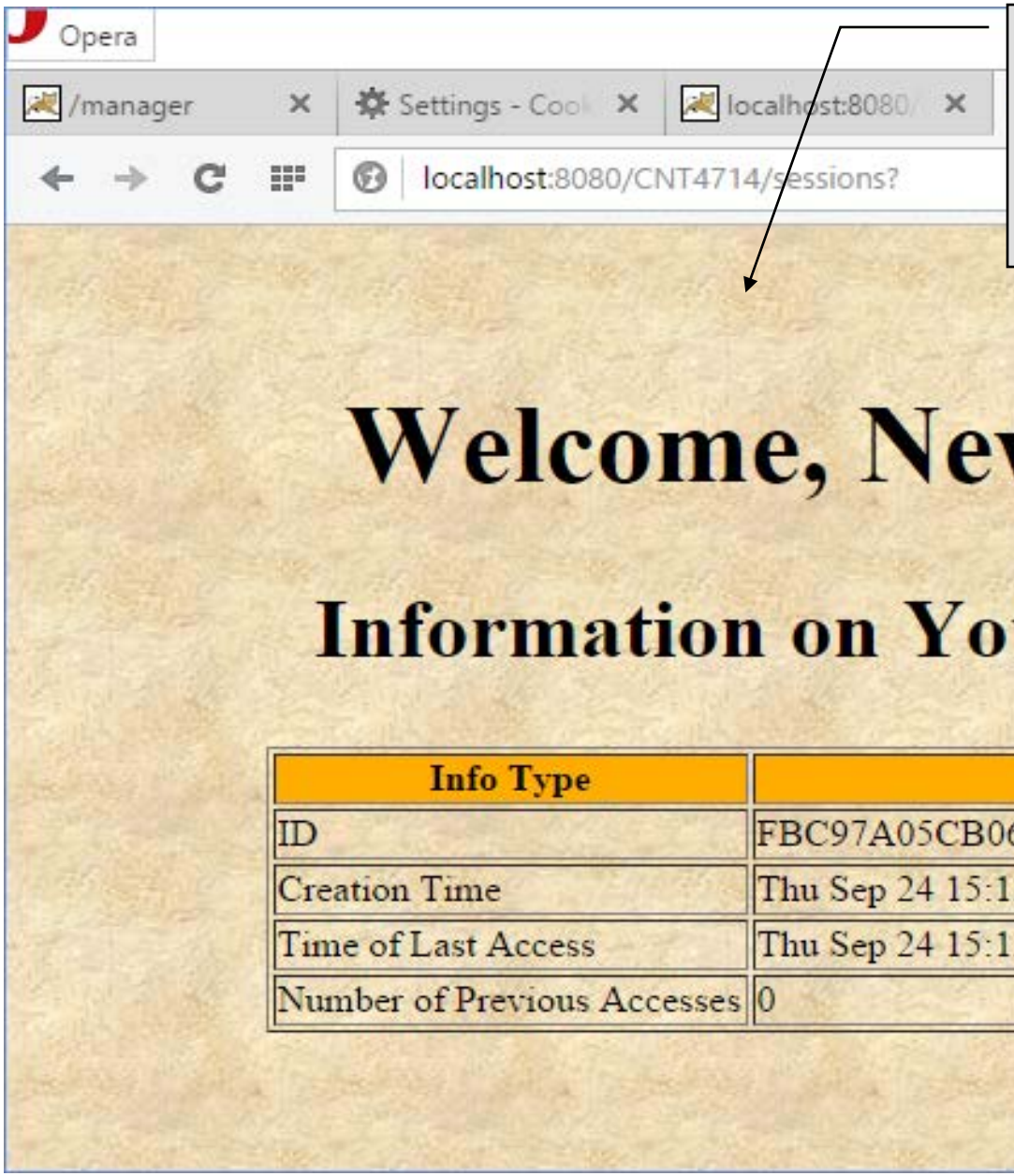
File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java

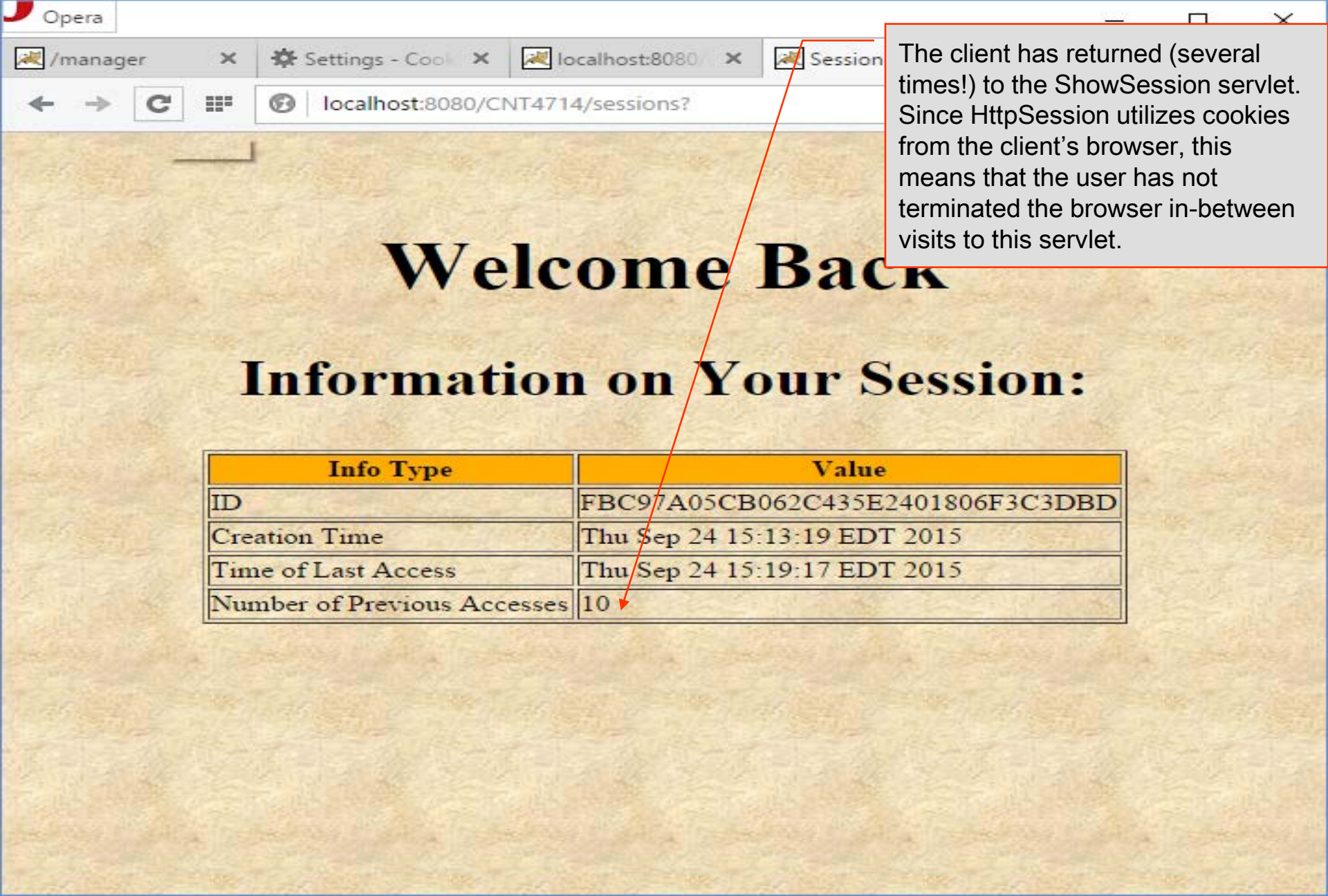
ClientAccessCounts.java | CookieUtilities.java | ShowSession.java

```
30      String title = "Session Tracking Example";
31      String docType =
32        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
33        "Transitional//EN\">\n";
34       out.println ("<body bgcolor=white background=images/background.jpg lang=EN-US link=blue vlink=blue
35      out.println ("<body style='tab-interval:.5in'>");
36       out.println ("<font size = 5>");
37       out.println ("<br>");
38      out.println(docType +
39                  "<HTML>\n" +
40                  "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
41                  "<BODY BGCOLOR=\"#FDF5E6\">\n" +
42                  "<CENTER>\n" +
43                  "<H1>" + heading + "</H1>\n" +
44                  "<H2>Information on Your Session:</H2>\n" +
45                  "<TABLE BORDER=1>\n" +
46                  "<TR BGCOLOR=\"#FFAD00\">\n" +
47                  "  <TH>Info Type<TH>Value\n" +
48                  "<TR>\n" +
49                  "  <TD>ID\n" +
50                  "  <TD>" + session.getId() + "\n" +
51                  "<TR>\n" +
52                  "  <TD>Creation Time\n" +
53                  "  <TD>" +
54                  new Date(session.getCreationTime()) + "\n" +
55                  "<TR>\n" +
56                  "  <TD>Time of Last Access\n" +
57                  "  <TD>" +
58                  new Date(session.getLastAccessedTime()) + "\n" +
59                  "<TR>\n" +
60                  "  <TD>Number of Previous Accesses\n" +
61                  "  <TD>" + accessCount + "\n" +
62                  "</TABLE>\n" +
63                  "</CENTER></BODY></HTML>");
64    }
65  }
66
```

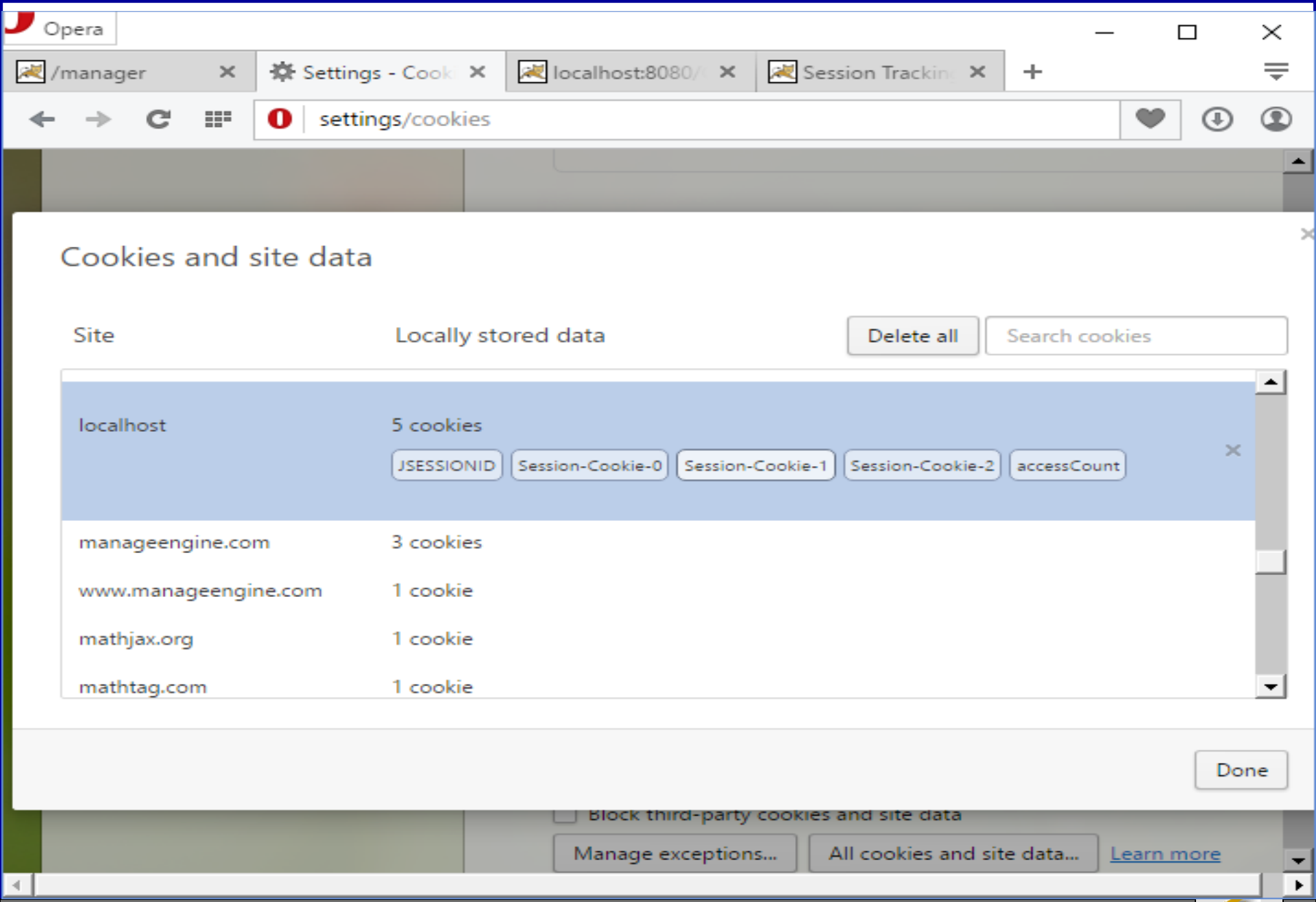
Writable | Smart Insert | 13 : 32

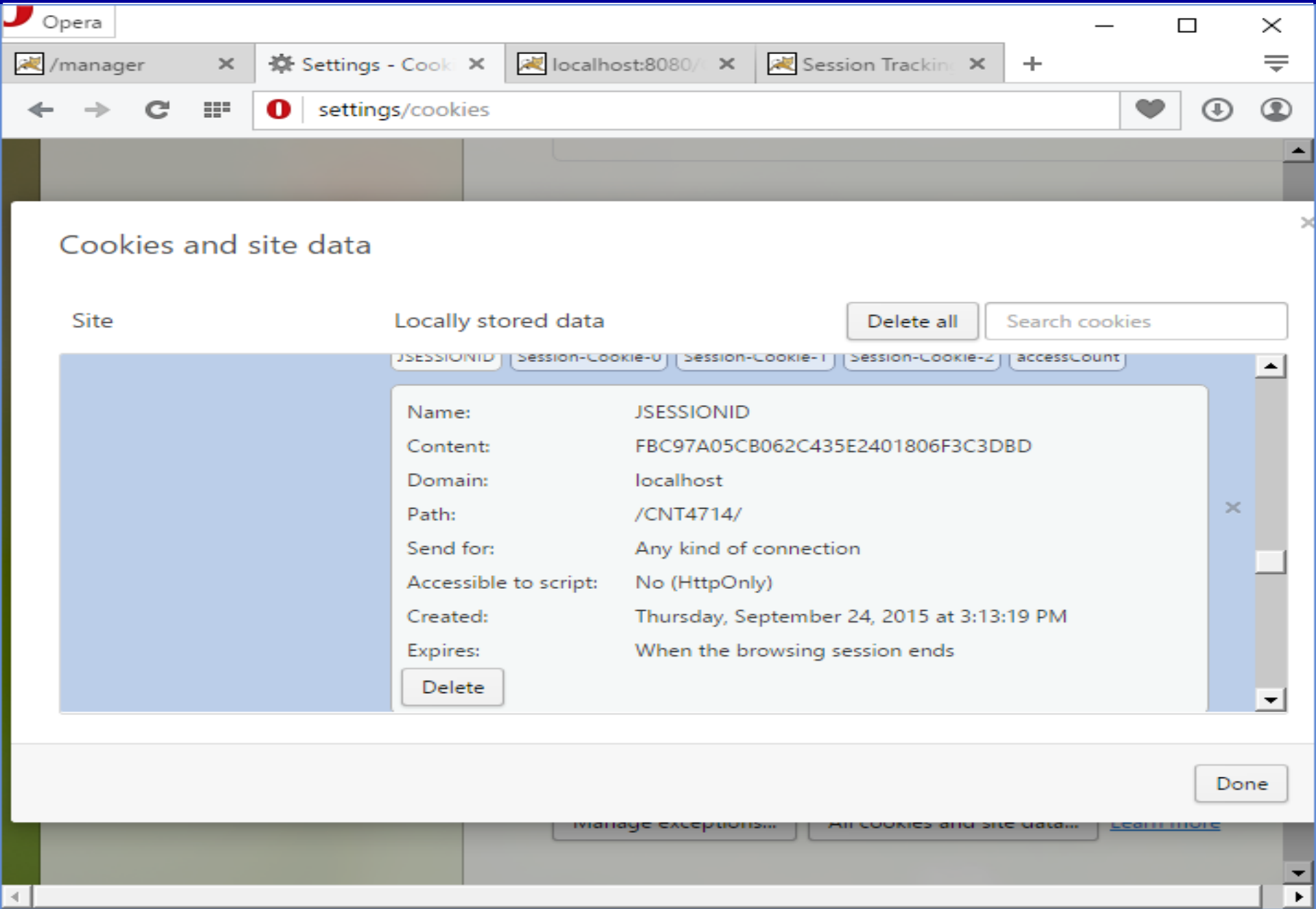


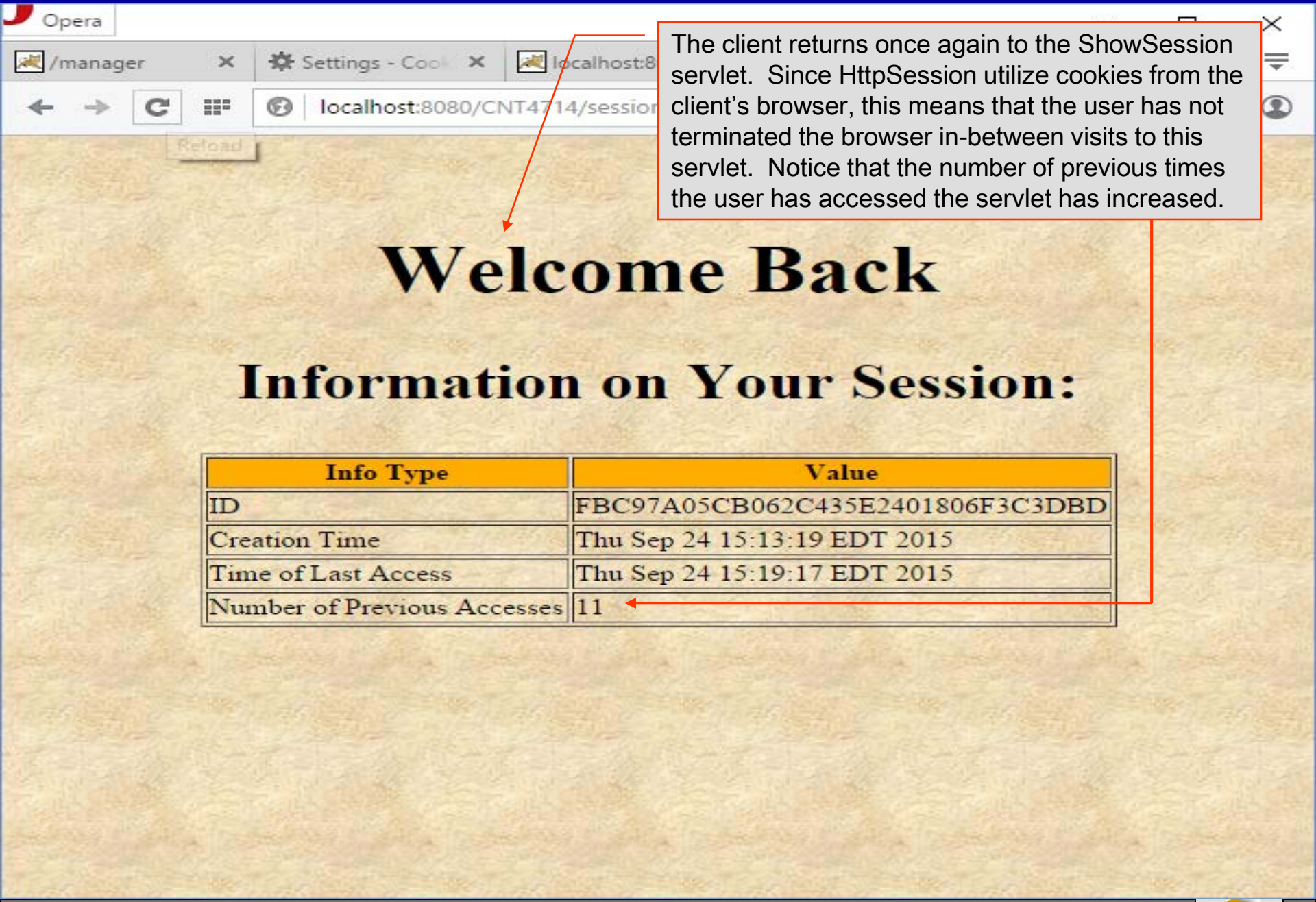
Client makes their first visit to the ShowSession servlet. Since no session exists for this client, one is created. Some of the details about this session are shown by the servlet. Note time of creation and time of last access are initially the same.

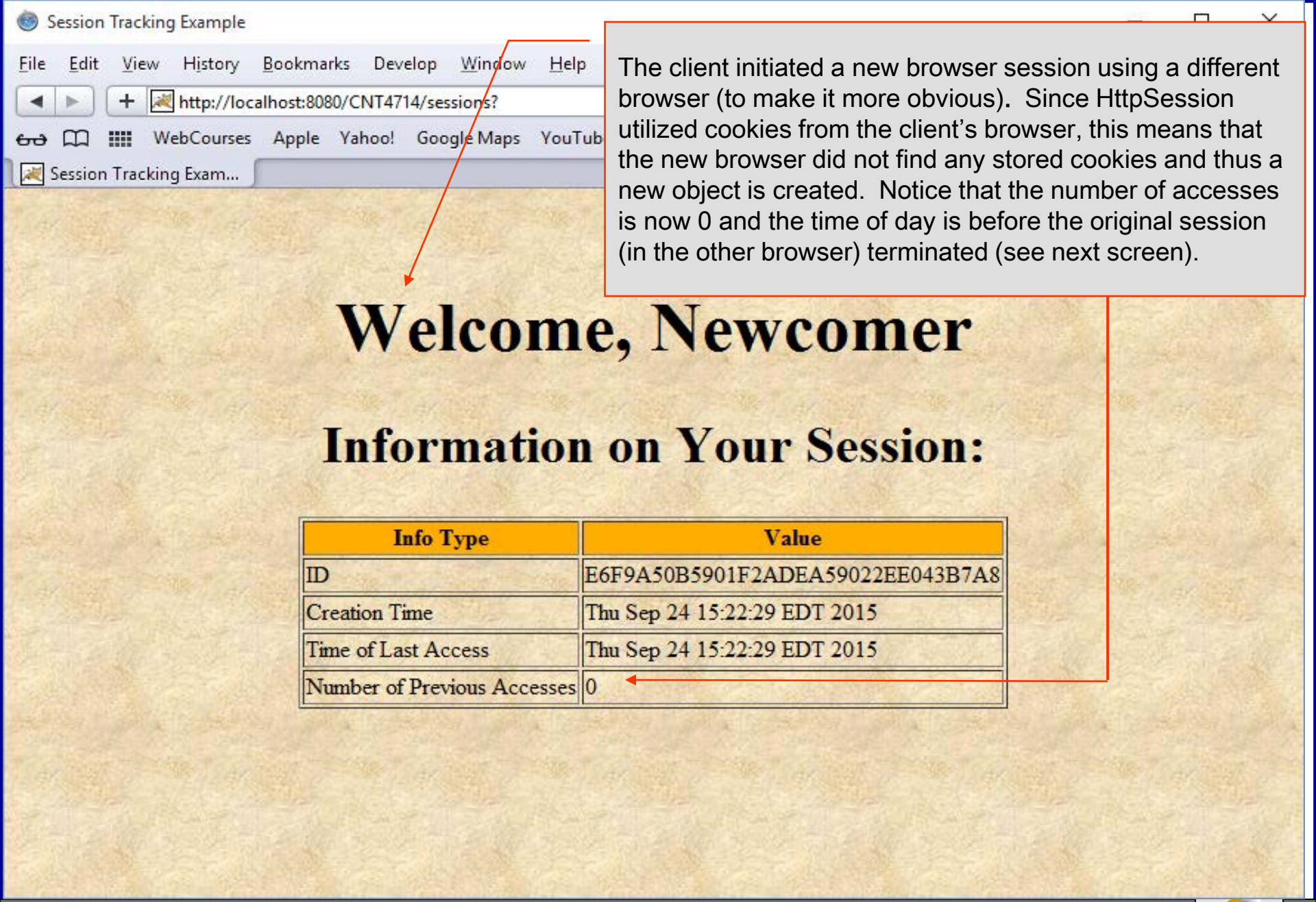


The client has returned (several times!) to the ShowSession servlet. Since HttpSession utilizes cookies from the client's browser, this means that the user has not terminated the browser in-between visits to this servlet.

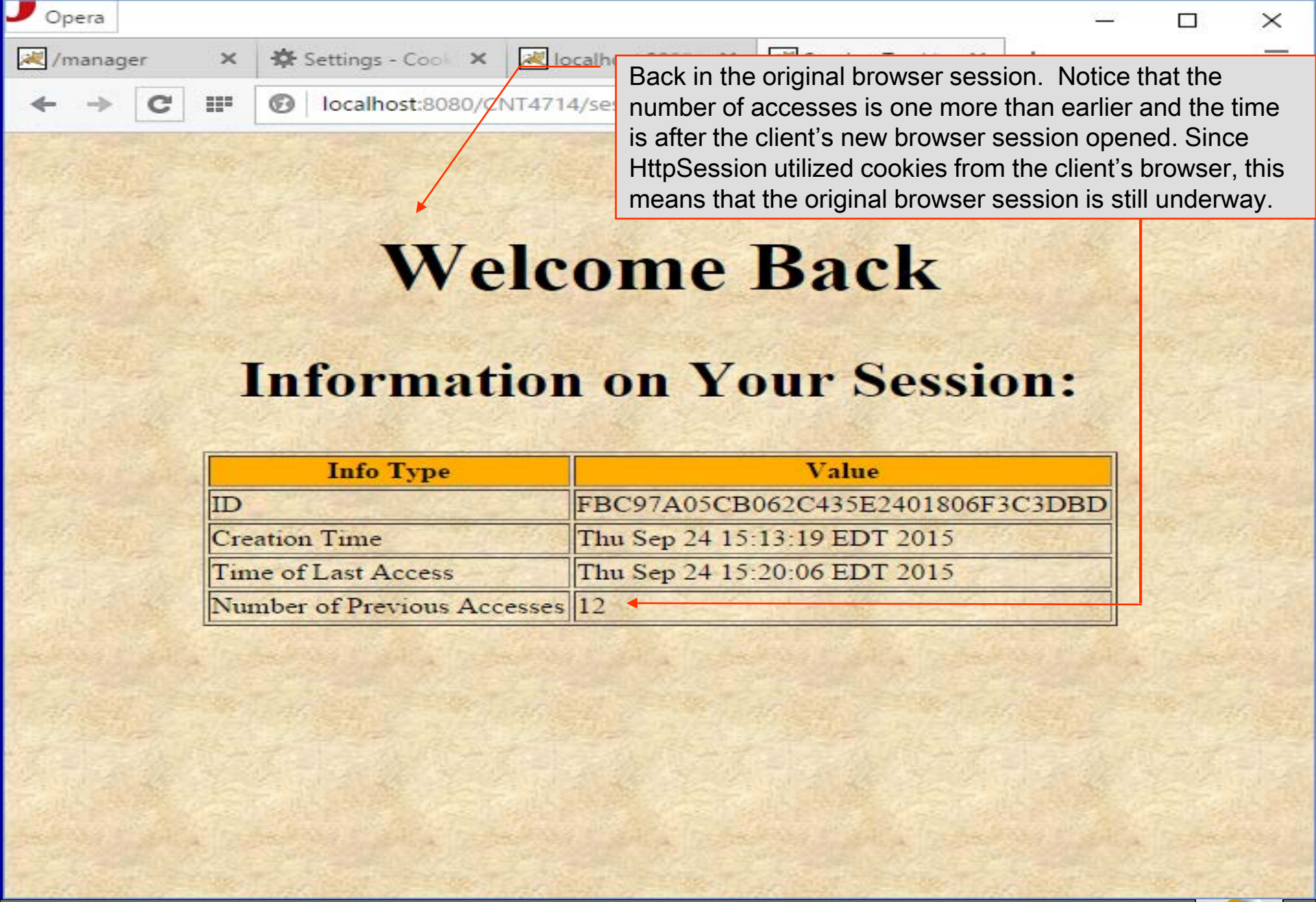

Opera
/manager
Settings - Cook
localhost:8080/
Session Tracking
settings/cookies
Cookies and site data
Site
Locally stored data
Delete all
Search cookies
localhost
5 cookies
JSESSIONID
Session-Cookie-0
Session-Cookie-1
Session-Cookie-2
accessCount
manageengine.com
3 cookies
www.manageengine.com
1 cookie
mathjax.org
1 cookie
mathtag.com
1 cookie
Done
Block third-party cookies and site data
Manage exceptions...
All cookies and site data...
Learn more

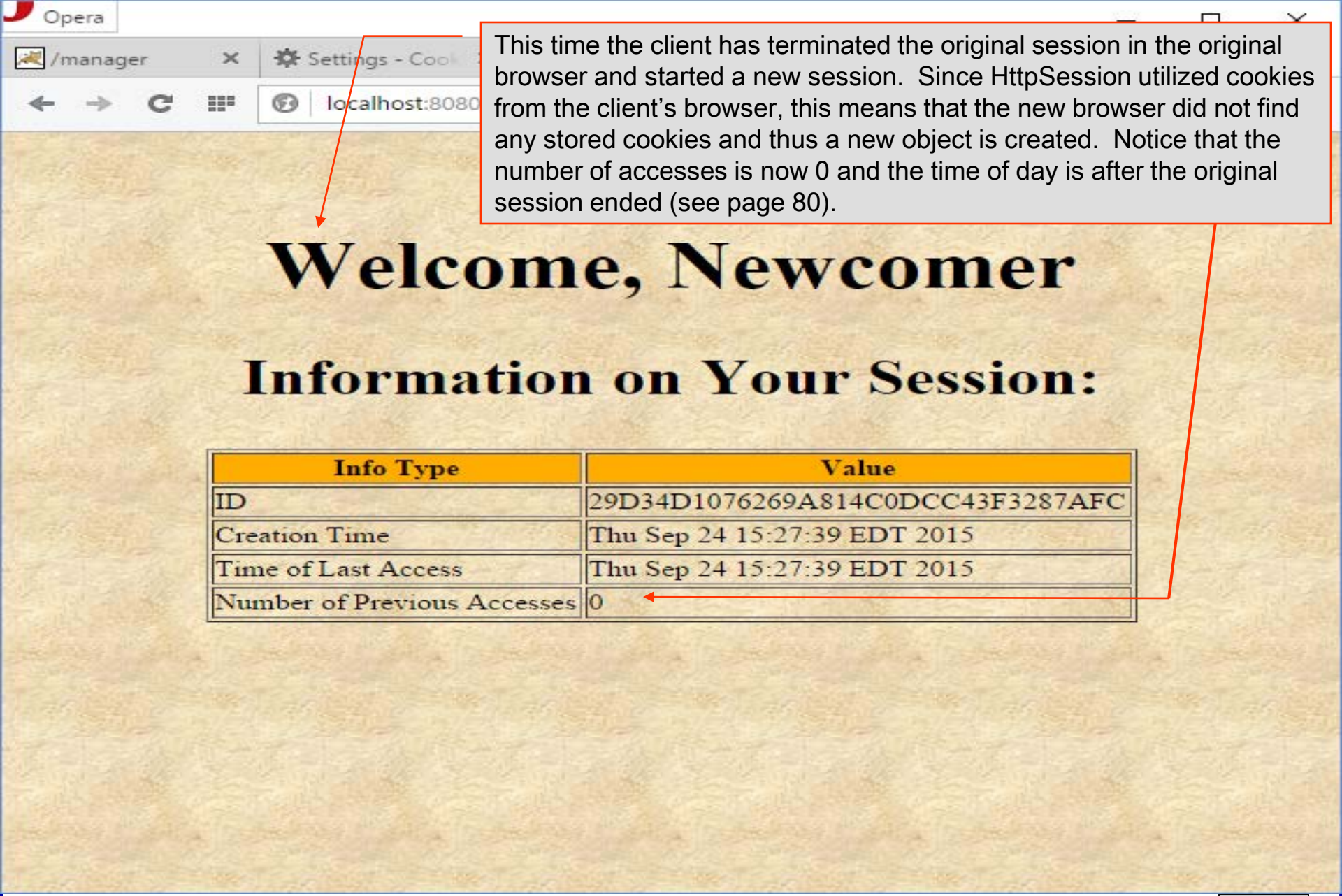

Opera
/manager
Settings - Cook
localhost:8080/
Session Tracking
settings/cookies
Cookies and site data
Site
Locally stored data
Delete all
Search cookies
JSESSIONID
Session-Cookie-0
Session-Cookie-1
Session-Cookie-2
accessCount
Name: JSESSIONID
Content: FBC97A05CB062C435E2401806F3C3DBD
Domain: localhost
Path: /CNT4714/
Send for: Any kind of connection
Accessible to script: No (HttpOnly)
Created: Thursday, September 24, 2015 at 3:13:19 PM
Expires: When the browsing session ends
Delete
Done


Opera
/manager
Settings - Coo
localhost:8
localhost:8080/CNT4714/sessio
Reload
The client returns once again to the ShowSession servlet. Since HttpSession utilize cookies from the client's browser, this means that the user has not terminated the browser in-between visits to this servlet. Notice that the number of previous times the user has accessed the servlet has increased.
Welcome Back
Information on Your Session:
Info Type
Value
ID
FBC97A05CB062C435E2401806F3C3DBD
Creation Time
Thu Sep 24 15:13:19 EDT 2015
Time of Last Access
Thu Sep 24 15:19:17 EDT 2015
Number of Previous Accesses
11



The client initiated a new browser session using a different browser (to make it more obvious). Since HttpSession utilized cookies from the client's browser, this means that the new browser did not find any stored cookies and thus a new object is created. Notice that the number of accesses is now 0 and the time of day is before the original session (in the other browser) terminated (see next screen).


Opera
/manager
Settings - Cook
localhost:8080/CNT4714/ses
Back in the original browser session. Notice that the number of accesses is one more than earlier and the time is after the client's new browser session opened. Since HttpSession utilized cookies from the client's browser, this means that the original browser session is still underway.
Welcome Back
Information on Your Session:
Info Type
Value
ID
FBC97A05CB062C435E2401806F3C3DBD
Creation Time
Thu Sep 24 15:13:19 EDT 2015
Time of Last Access
Thu Sep 24 15:20:06 EDT 2015
Number of Previous Accesses
12


Opera
/manager
Settings - Cook
localhost:8080
This time the client has terminated the original session in the original browser and started a new session. Since HttpSession utilized cookies from the client's browser, this means that the new browser did not find any stored cookies and thus a new object is created. Notice that the number of accesses is now 0 and the time of day is after the original session ended (see page 80).
Welcome, Newcomer
Information on Your Session:
Info Type
Value
ID
29D34D1076269A814C0DCC43F3287AFC
Creation Time
Thu Sep 24 15:27:39 EDT 2015
Time of Last Access
Thu Sep 24 15:27:39 EDT 2015
Number of Previous Accesses
0