Agenda for this lecture:

- Sessions and Cookies
- Shopping cart examples
- Handling the Data Store/Data Model
- Midterm Exam

Maintaining State

- HTTP is a stateless protocol
- Each request is completely independent of the previous
- Most applications need to understand the current state of the application
  - The pervasive "shopping cart"
  - Personalization
  - Workflow

How can you maintain state?

- State must either be completely part of the request, or maintained on the server
  - Cookies - simple data
  - Sessions - data for a current session (across pages)
  - Databases - persisting data across sessions

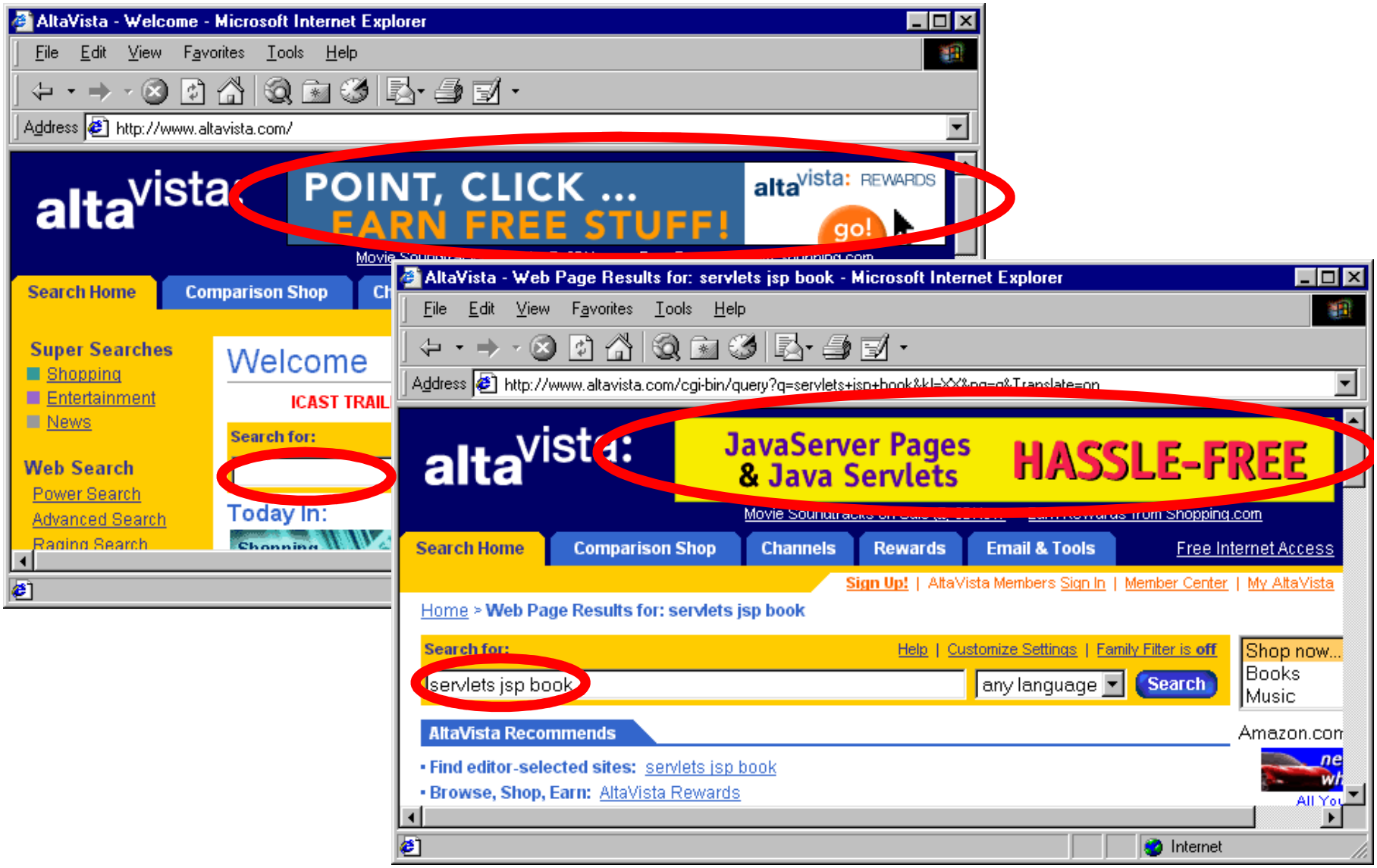
Cookies

- Defined in RFC 2109
- **A cookie is basically a small piece of information stored by a web browser on the local client.**
- Browsers are expected to only support 20 cookies per Web server, 300 total, and can be limited to 4kb each
- Cookies are returned to the server as HTTP request headers

Cookie uses

- Cookies are used on the web for a number of purposes
  - Identifying a user during an e-commerce session
  - Tracking Sessions
  - Focusing advertising
  - Tracking usage
  - Personalization
- If used properly, not a security risk

Cookies and Focused Advertising



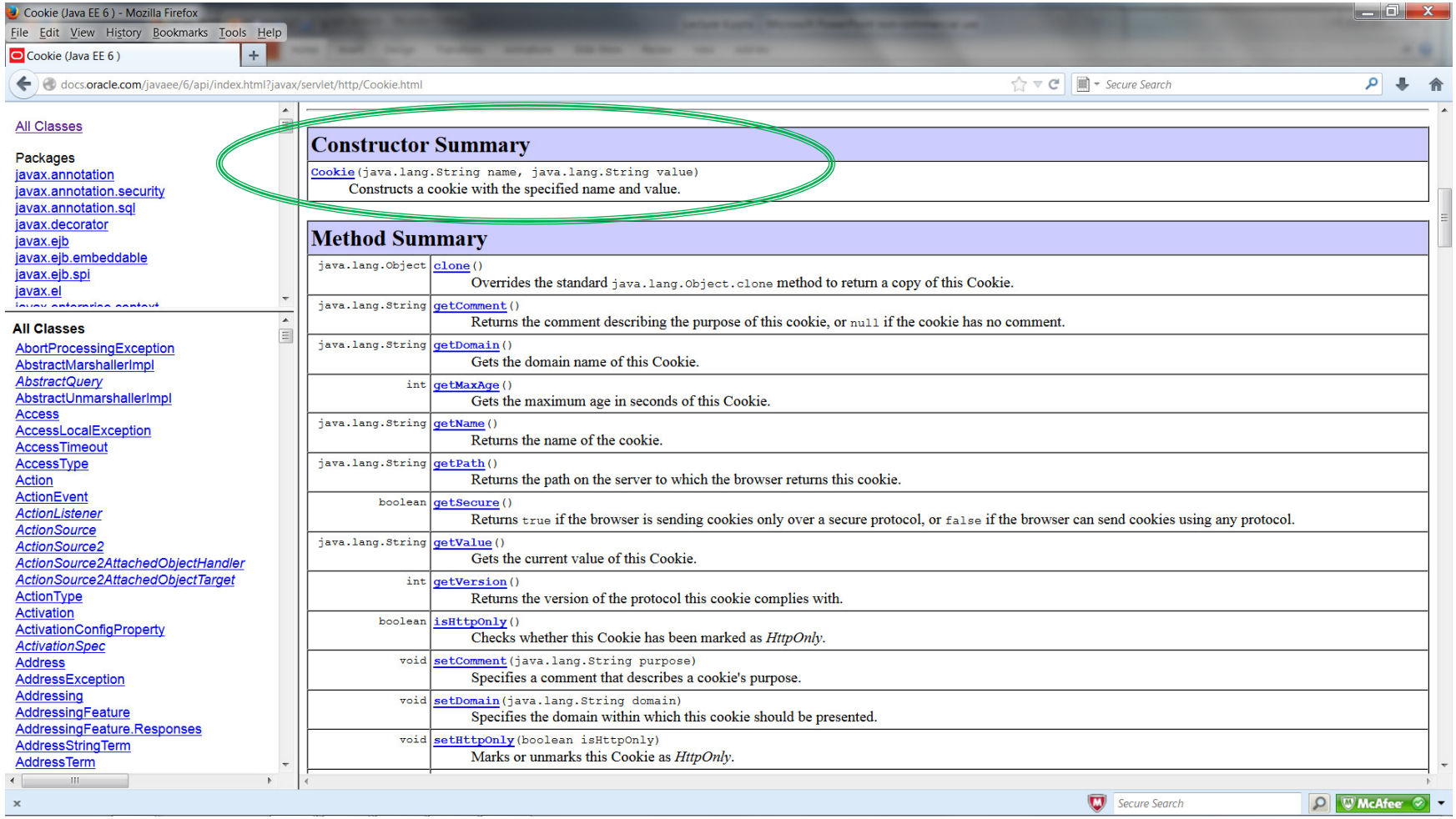
Problems with Cookies

- **Some may argue that the problem with the cookies is privacy NOT security!**
  - A user can turn them off
  - They can violate privacy
  - If used improperly, a security risk. For example, Poorly designed sites store sensitive information like credit card numbers directly in cookie
- **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

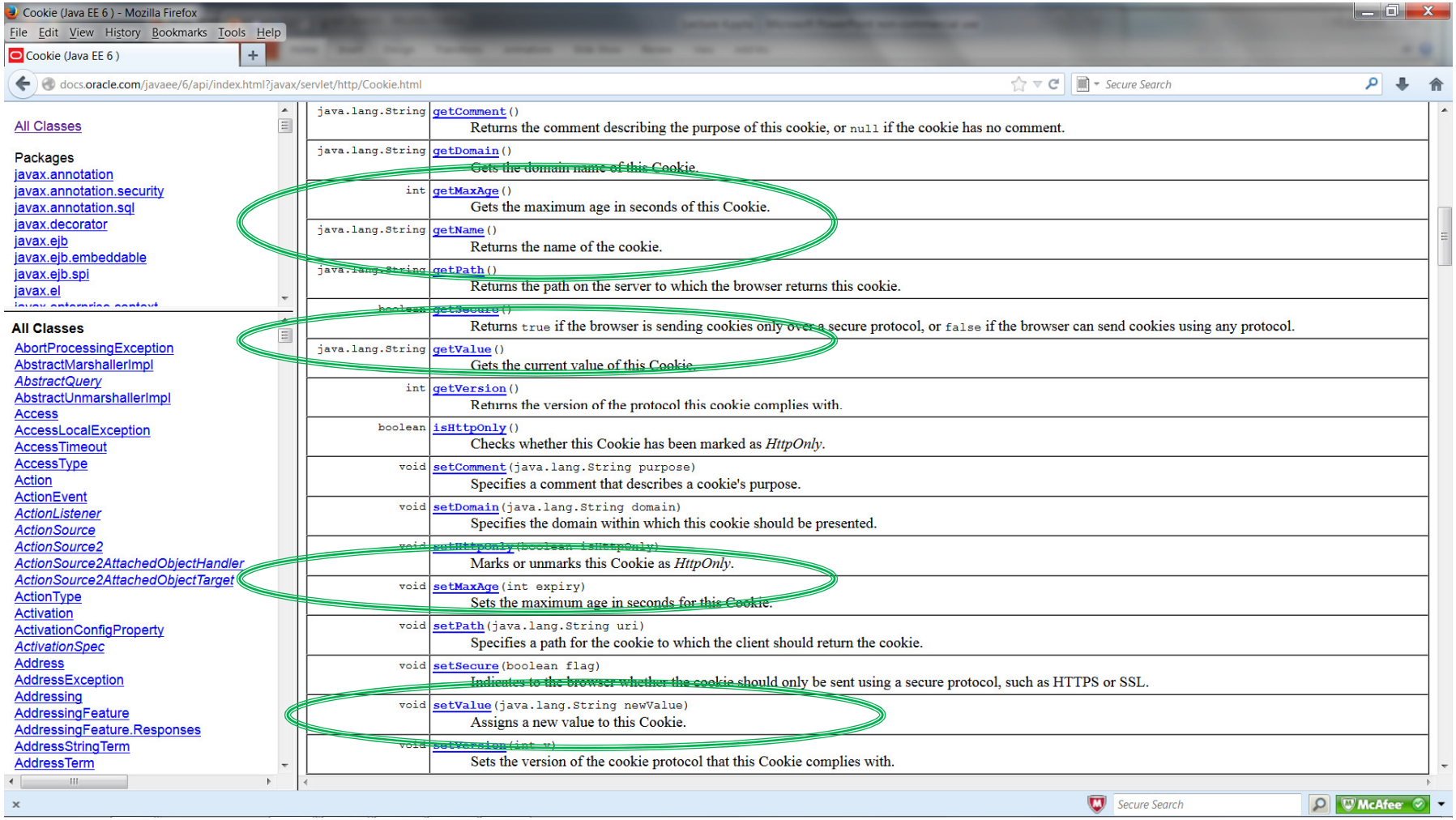
Cookie API


Cookie (Java EE 6) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Cookie (Java EE 6)
docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/Cookie.html
Secure Search
javax.security.auth.message.module
javax.security.jacc
javax.servlet
javax.servlet.annotation
javax.servlet.descriptor
javax.servlet.http
javax.servlet.jsp
javax.servlet.jsp.el
javax.servlet.jsp.jstl.core
javax.servlet.jsp.jstl.fmt
javax.servlet.jsp.jstl.sql
javax.servlet.jsp.jstl.tlv
javax.servlet.http
Interfaces
HttpServletRequest
HttpServletResponse
HttpSession
HttpSessionActivationListener
HttpSessionAttributeListener
HttpSessionBindingListener
HttpSessionContext
HttpSessionListener
Part
Classes
Cookie
HttpServlet
HttpServletRequestWrapper
HttpServletResponseWrapper
HttpSessionBindingEvent
HttpSessionEvent
HttpUtils
Overview Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
javax.servlet.http
Class Cookie
java.lang.Object
└ javax.servlet.http.Cookie
All Implemented Interfaces:
java.io.Serializable, java.lang.Cloneable
public class Cookie
extends java.lang.Object
implements java.lang.Cloneable, java.io.Serializable
Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management.
A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number. Some Web browsers have bugs in how they handle the optional attributes, so use them sparingly to improve the interoperability of your servlets.
The servlet sends cookies to the browser by using the HttpServletResponse#addCookie method, which adds fields to HTTP response headers to send cookies to the browser, one at a time. The browser is expected to support 20 cookies for each Web server, 300 cookies total, and may limit cookie size to 4 KB each.
The browser returns cookies to the servlet by adding fields to HTTP request headers. Cookies can be retrieved from a request by using the HttpServletRequest#getCookies method. Several cookies might have the same name but different path attributes.
Cookies affect the caching of the Web pages that use them. HTTP 1.0 does not cache pages that use cookies created with this class. This class does not support the cache control defined with HTTP 1.1.
This class supports both the Version 0 (by Netscape) and Version 1 (by RFC 2109) cookie specifications. By default, cookies are created using Version 0 to ensure the best interoperability.
Author:
Various
Secure Search
McAfee

Cookie API


Cookie (Java EE 6) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Cookie (Java EE 6)
docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/Cookie.html
Secure Search
All Classes
Packages
javax.annotation
javax.annotation.security
javax.annotation.sql
javax.decorator
javax.ejb
javax.ejb.embeddable
javax.ejb.spi
javax.el
All Classes
AbortProcessingException
AbstractMarshallerImpl
AbstractQuery
AbstractUnmarshallerImpl
Access
AccessLocalException
AccessTimeout
AccessType
Action
ActionEvent
ActionListener
ActionSource
ActionSource2
ActionSource2AttachedObjectHandler
ActionSource2AttachedObjectTarget
ActionType
Activation
ActivationConfigProperty
ActivationSpec
Address
AddressException
Addressing
AddressingFeature
AddressingFeature.Responses
AddressStringTerm
AddressTerm
Constructor Summary
Cookie(java.lang.String name, java.lang.String value)
Constructs a cookie with the specified name and value.
Method Summary
java.lang.Object clone()
Overrides the standard java.lang.Object.clone method to return a copy of this Cookie.
java.lang.String getComment()
Returns the comment describing the purpose of this cookie, or null if the cookie has no comment.
java.lang.String getDomain()
Gets the domain name of this Cookie.
int getMaxAge()
Gets the maximum age in seconds of this Cookie.
java.lang.String getName()
Returns the name of the cookie.
java.lang.String getPath()
Returns the path on the server to which the browser returns this cookie.
boolean getSecure()
Returns true if the browser is sending cookies only over a secure protocol, or false if the browser can send cookies using any protocol.
java.lang.String getValue()
Gets the current value of this Cookie.
int getVersion()
Returns the version of the protocol this cookie complies with.
boolean isHttpOnly()
Checks whether this Cookie has been marked as HttpOnly.
void setComment(java.lang.String purpose)
Specifies a comment that describes a cookie's purpose.
void setDomain(java.lang.String domain)
Specifies the domain within which this cookie should be presented.
void setHttpOnly(boolean isHttpOnly)
Marks or unmarks this Cookie as HttpOnly.
Secure Search
McAfee

Cookie API


Cookie (Java EE 6) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Cookie (Java EE 6)
docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/Cookie.html
Secure Search
All Classes
Packages
javax.annotation
javax.annotation.security
javax.annotation.sql
javax.decorator
javax.ejb
javax.ejb.embeddable
javax.ejb.spi
javax.el
All Classes
AbortProcessingException
AbstractMarshallerImpl
AbstractQuery
AbstractUnmarshallerImpl
Access
AccessLocalException
AccessTimeout
AccessType
Action
ActionEvent
ActionListener
ActionSource
ActionSource2
ActionSource2AttachedObjectHandler
ActionSource2AttachedObjectTarget
ActionType
Activation
ActivationConfigProperty
ActivationSpec
Address
AddressException
Addressing
AddressingFeature
AddressingFeature.Responses
AddressStringTerm
AddressTerm
java.lang.String getComment()
Returns the comment describing the purpose of this cookie, or null if the cookie has no comment.
java.lang.String getDomain()
Gets the domain name of this Cookie.
int getMaxAge()
Gets the maximum age in seconds of this Cookie.
java.lang.String getName()
Returns the name of the cookie.
java.lang.String getPath()
Returns the path on the server to which the browser returns this cookie.
boolean getSecure()
Returns true if the browser is sending cookies only over a secure protocol, or false if the browser can send cookies using any protocol.
java.lang.String getValue()
Gets the current value of this Cookie.
int getVersion()
Returns the version of the protocol this cookie complies with.
boolean isHttpOnly()
Checks whether this Cookie has been marked as HttpOnly.
void setComment(java.lang.String purpose)
Specifies a comment that describes a cookie's purpose.
void setDomain(java.lang.String domain)
Specifies the domain within which this cookie should be presented.
void setHttpOnly(boolean isHttpOnly)
Marks or unmarks this Cookie as HttpOnly.
void setMaxAge(int expiry)
Sets the maximum age in seconds for this Cookie.
void setPath(java.lang.String uri)
Specifies a path for the cookie to which the client should return the cookie.
void setSecure(boolean flag)
Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.
void setValue(java.lang.String newValue)
Assigns a new value to this Cookie.
void setVersion(int v)
Sets the version of the cookie protocol that this Cookie complies with.
Secure Search
McAfee

Cookie API

- `public` **`Cookie`**`(java.lang.String name, java.lang.String value)`
  - Creates a Cookie
- `public void` **`setMaxAge`**`(int expiry)`
  - Set to age in seconds when Cookie expires
  - 0 deletes it
  - negative value deletes it when browser exits
- getters/setters for Name, Value, Path, Domain, etc.

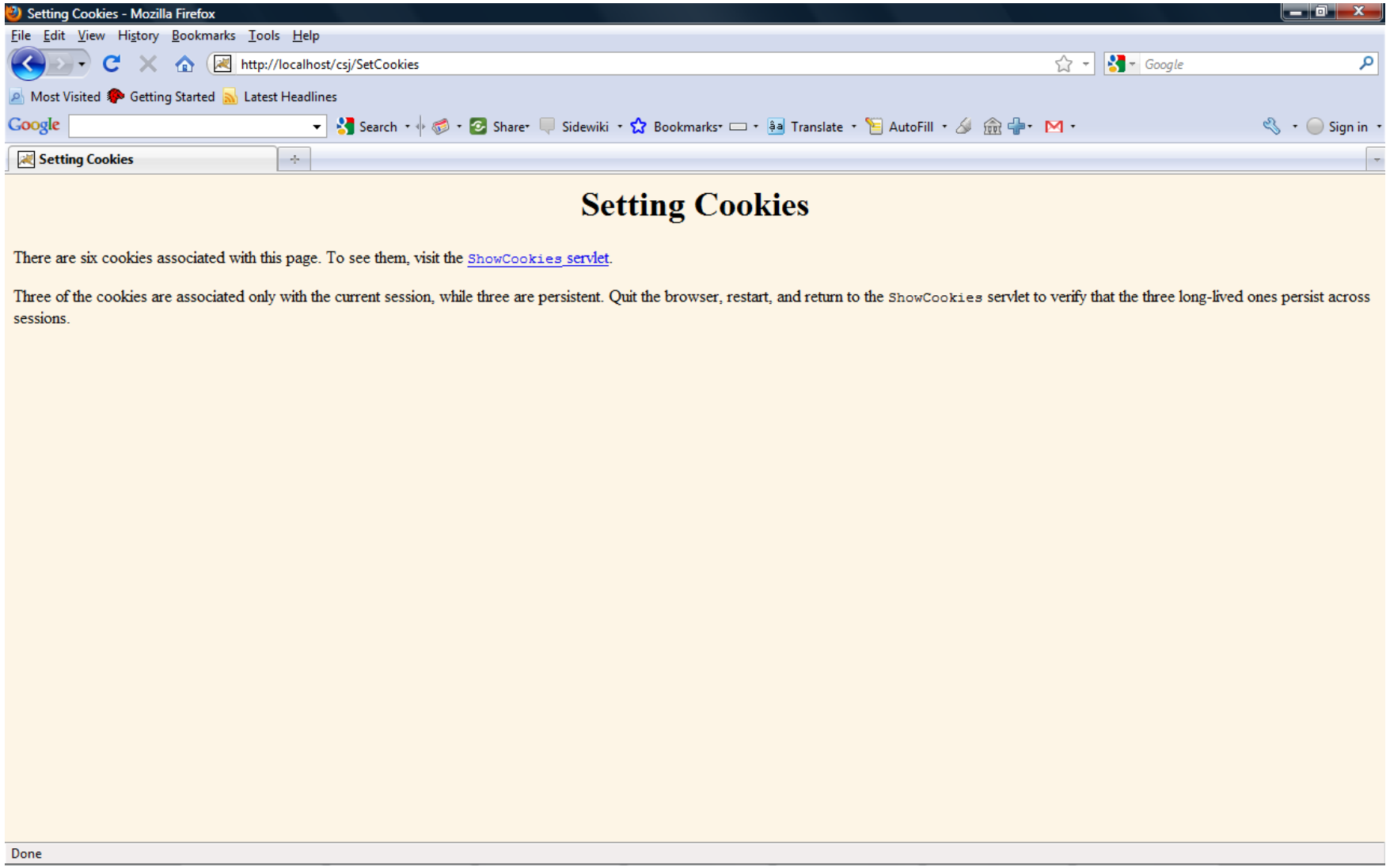
Cookie API (cont.)

- Cookies are returned by the browser to the server if the Domain and Path match the server
- Getting Cookies from the request
  - `public Cookie[] getCookies()`
- Adding Cookies to the response
  - `public void addCookie(Cookie cookie)`

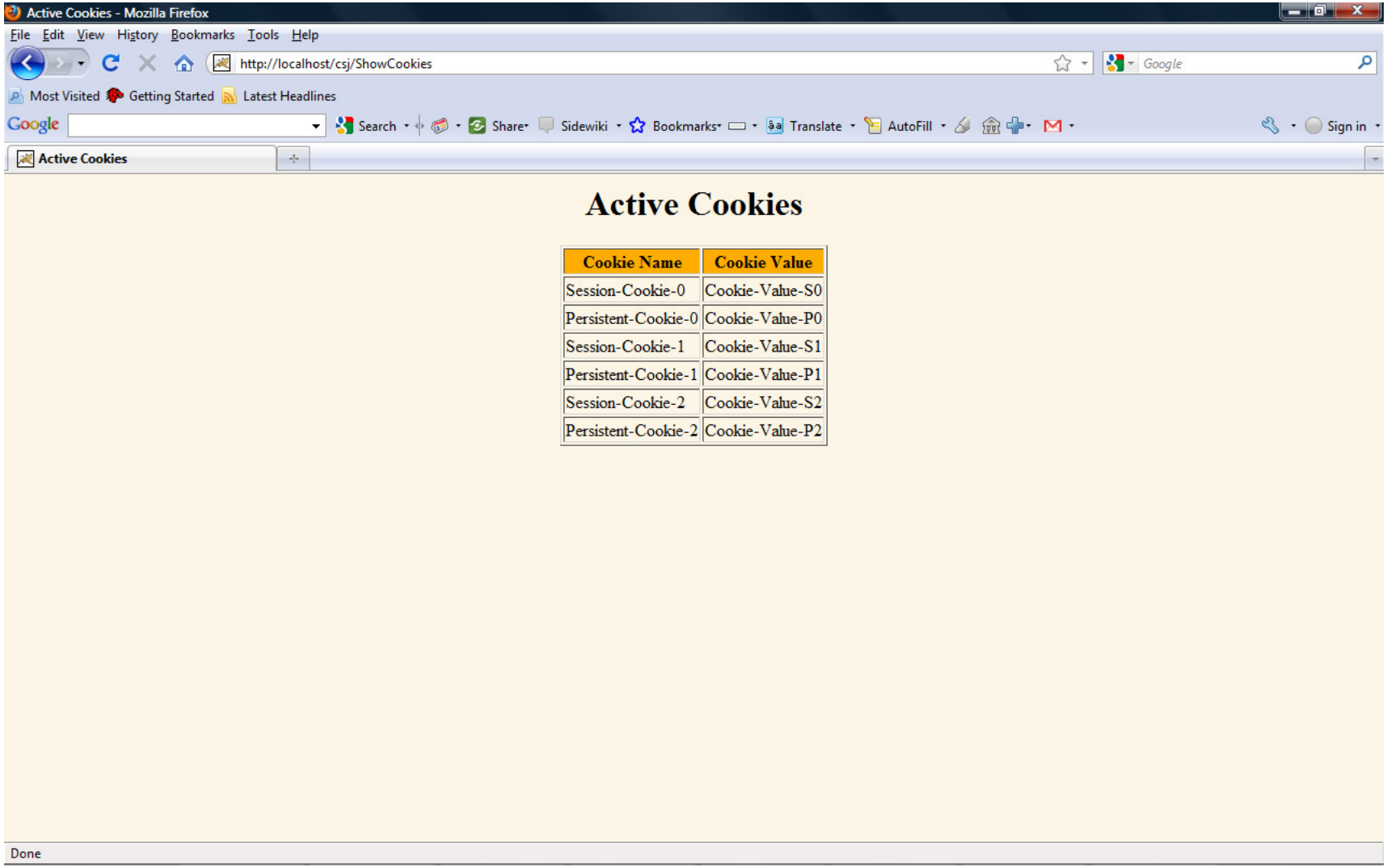
How to run and test session cookie and persistent cookie?

- From the Browser type:
  - http://localhost/csj/SetCookies
    - `You will see 3 session cookies and 3 persistent cookie`

How to run and test session cookie and persistent cookie?



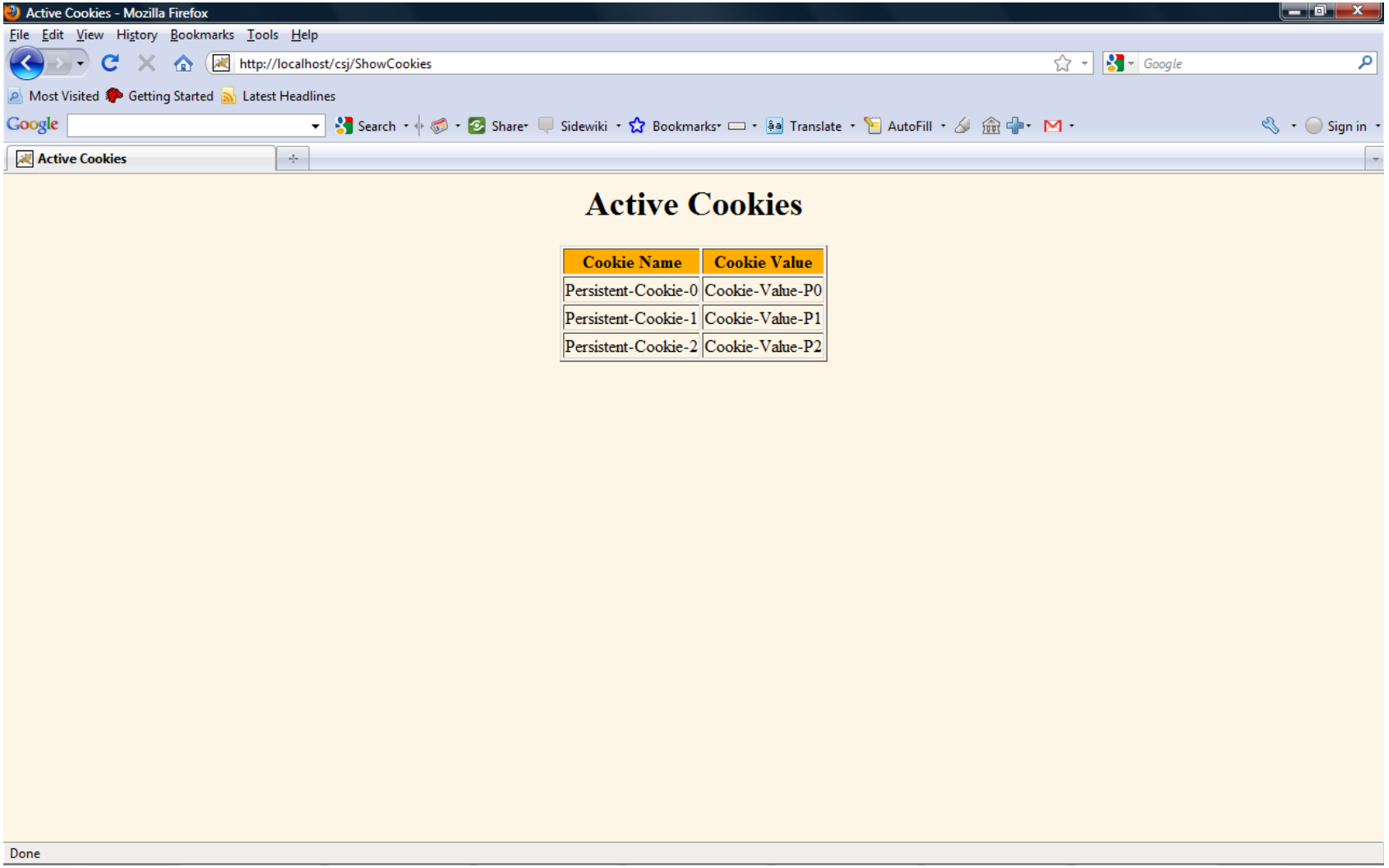
How to run and test session cookie and persistent cookie?



How to run and test session cookie and persistent cookie?

- **Now, kill the browser and start it again by accessing directly the ShowCookies witout calling SetCookies:**
  - **http://localhost/csj/ShowCookies**
    - You will see that there are 3 cookies that are still alive.

How to run and test session cookie and persistent cookie?



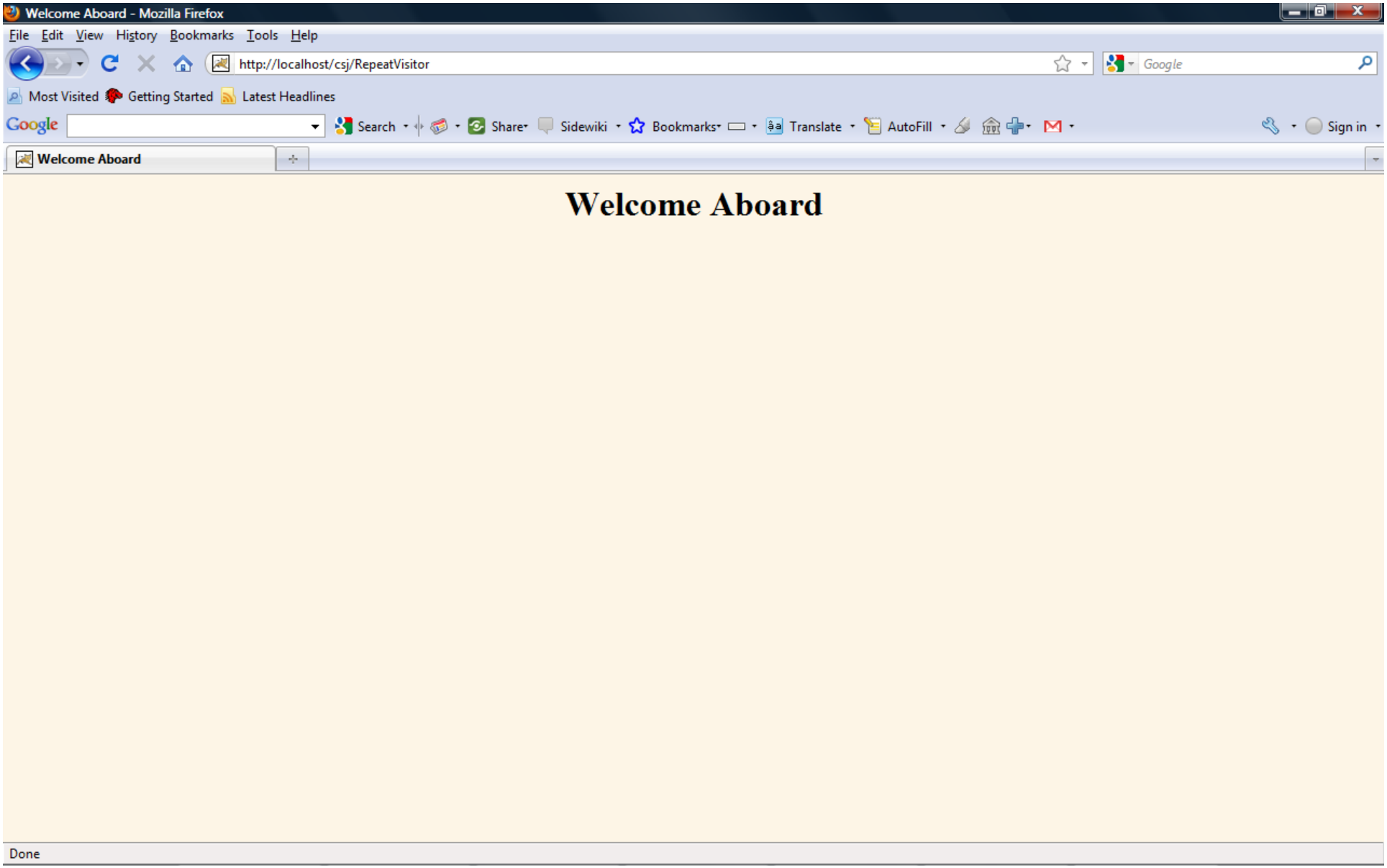
SetCookies and ShowCookies

- Lets review the source code for
  - SetCookies.java
  - ShowCookies.java

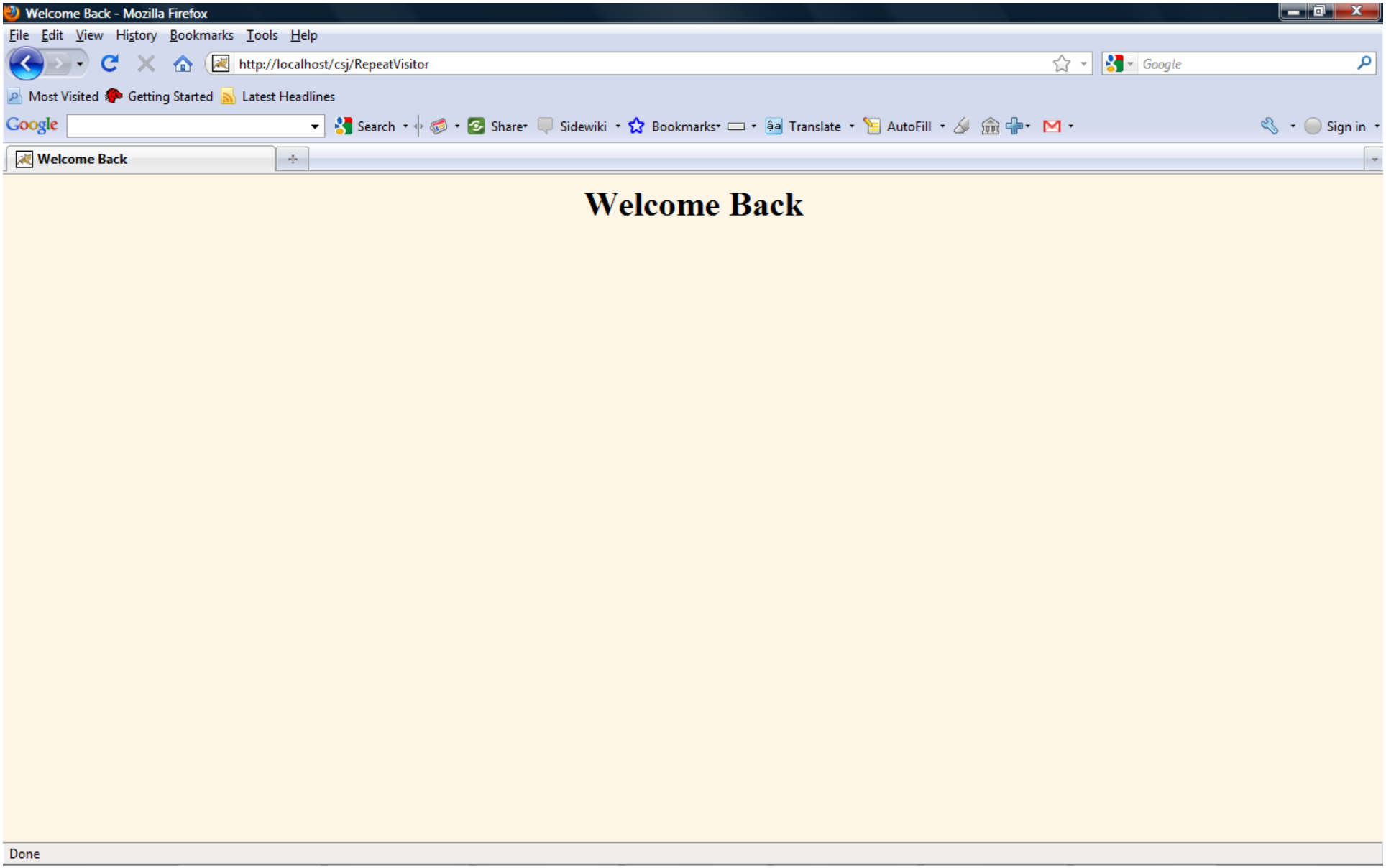
Cookie for Repeat Visitor

- Lets see an example for a servlet that says "Welcome aboard" to first-time visitors and "Welcome back" to repeat visitors
  - http://localhost/csj/RepeatVisitor
  - Refresh the browser after the first visit

Cookie for Repeat Visitor



Cookie for Repeat Visitor



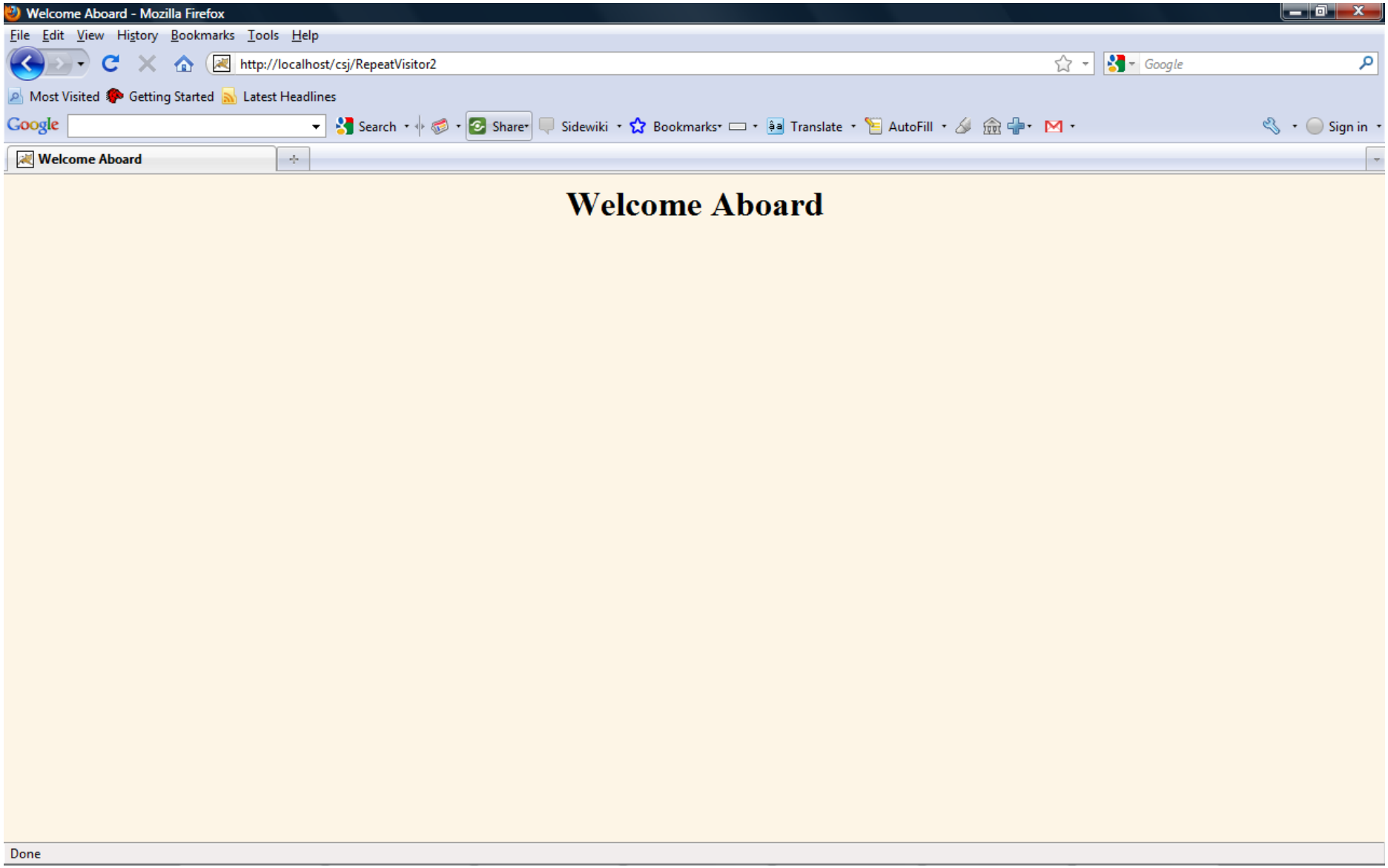
RepeatVisitor

- Lets review the source code for
  - RepeatVisitor.java

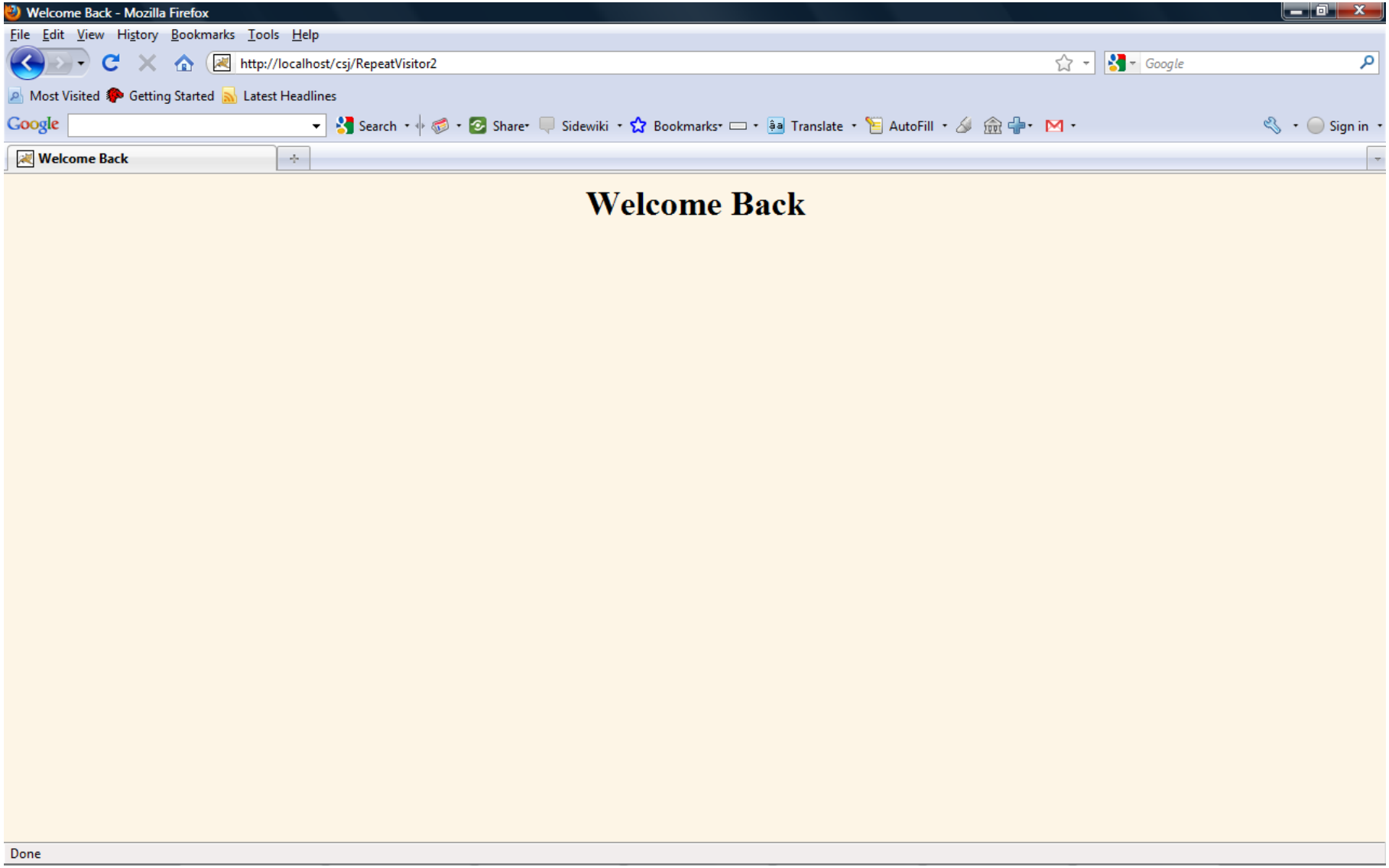
Repeat Visitor 2

- Better code for the previous RepeatVisitor example as follows:
  - RepeatVisitor2.java servlet uses the CookieUtilities.java and LongLivedCookie.java classes to simplify the code

Repeat Visitor 2



Repeat Visitor 2



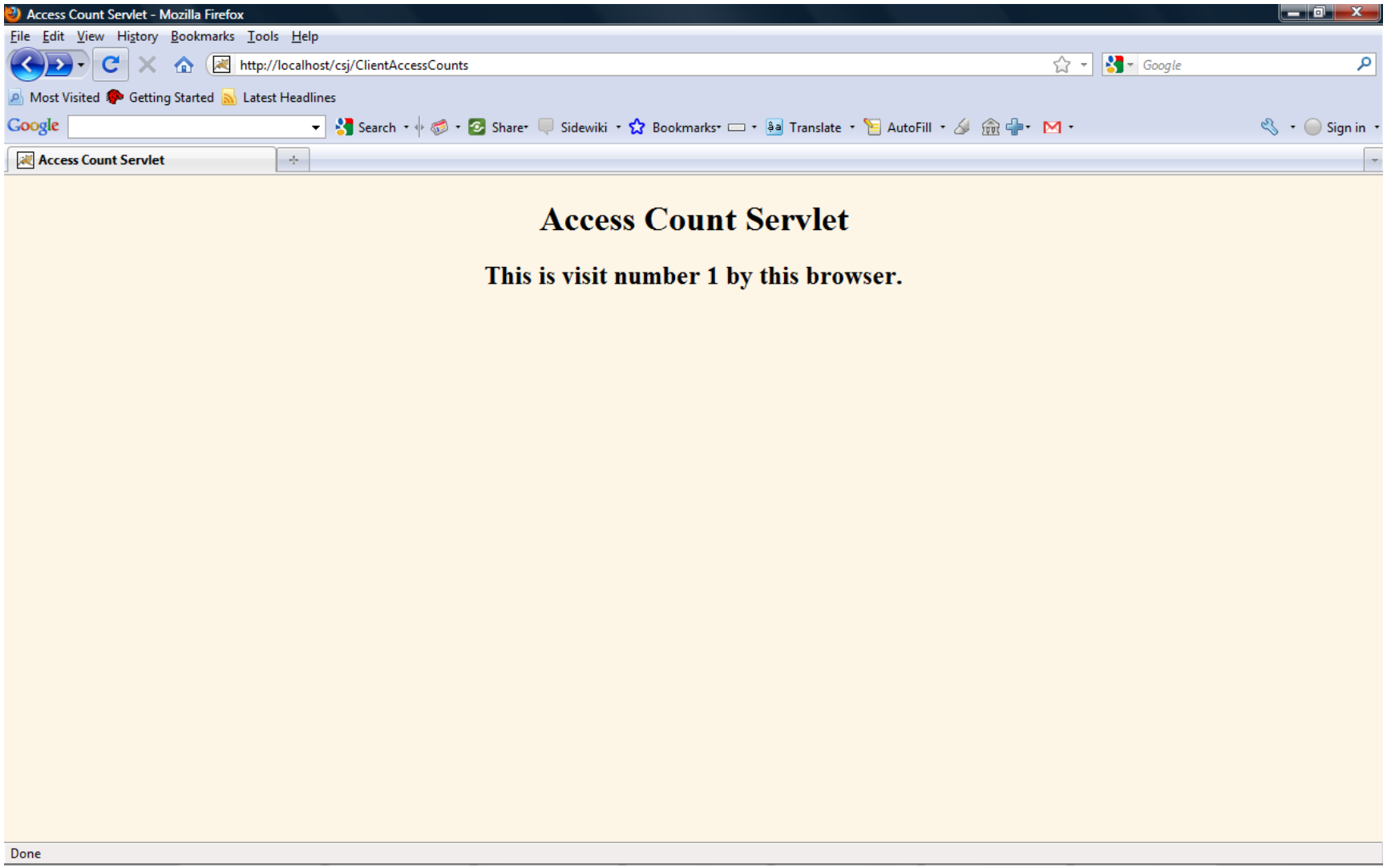
RepeatVisitor2

- Lets review the source code for
  - RepeatVisitor2.java
  - CookieUtilities.java
  - LongLivedCookie.java

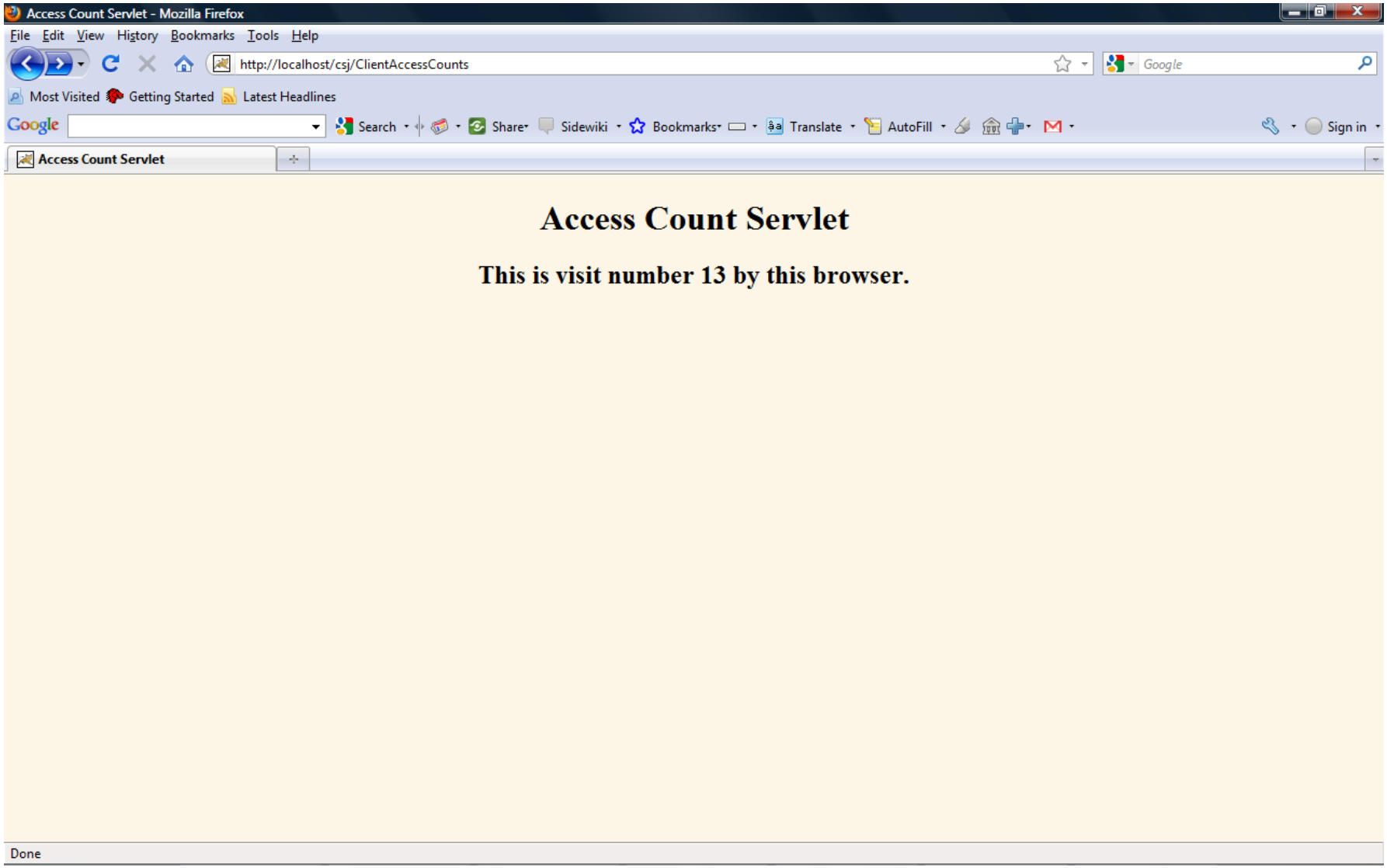
How to repeatedly change cookie values?

- 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. You also need to reapply any relevant cookie attributes by calling setMaxAge, setPath, etc
- The following example illustrates how to repeatedly change cookie values
  - ClientAccessCounts.java. Servlet that prints per-client access counts.

How to repeatedly change cookie values?



How to repeatedly change cookie values?



Session Tracking

- Why session tracking?
  - A **cookie** has a name and a **single value**. It is **stored on the client side**
  - When clients at on-line store add item to their shopping cart, how does server know what's already in cart?
  - When clients decide to proceed to checkout, how can server determine which previously created cart is theirs?

Session Tracking

- In web applications, the concept of a session is used to put consecutive requests and responses in context
- Implicit in a *session* are the following concepts:
  - start and end
  - may be terminated by either party
  - state information is present
- The concept of *session tracking* allows stateful information to be stored on the server and associated with a specific client

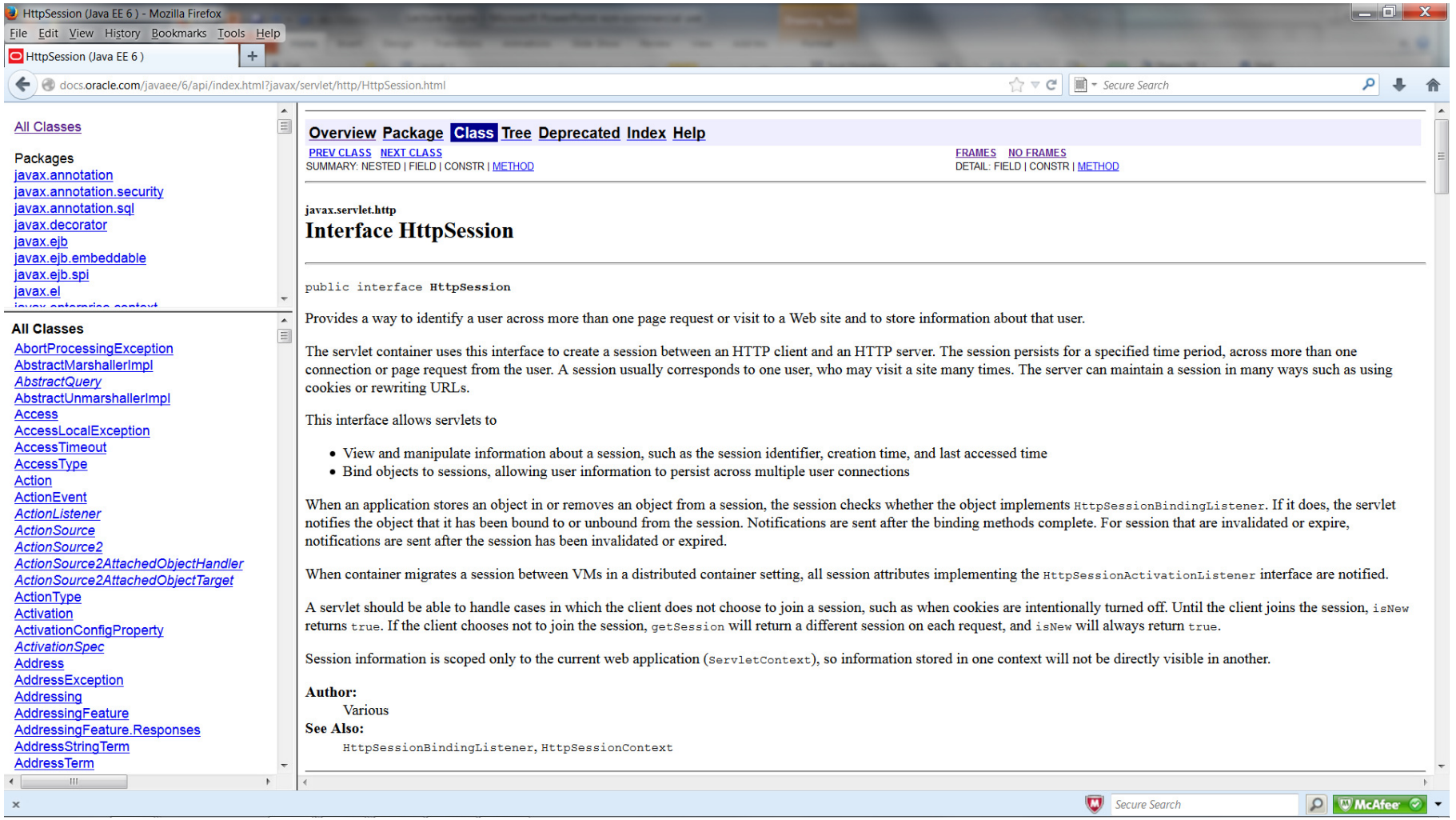
Session Tracking API

- The Java Servlets API includes support for session tracking
- The interface `HttpSession` encapsulates the session information
- The session is stored on the server
- Tracking is performed using Cookies or URL-rewriting and is transparent to the servlet

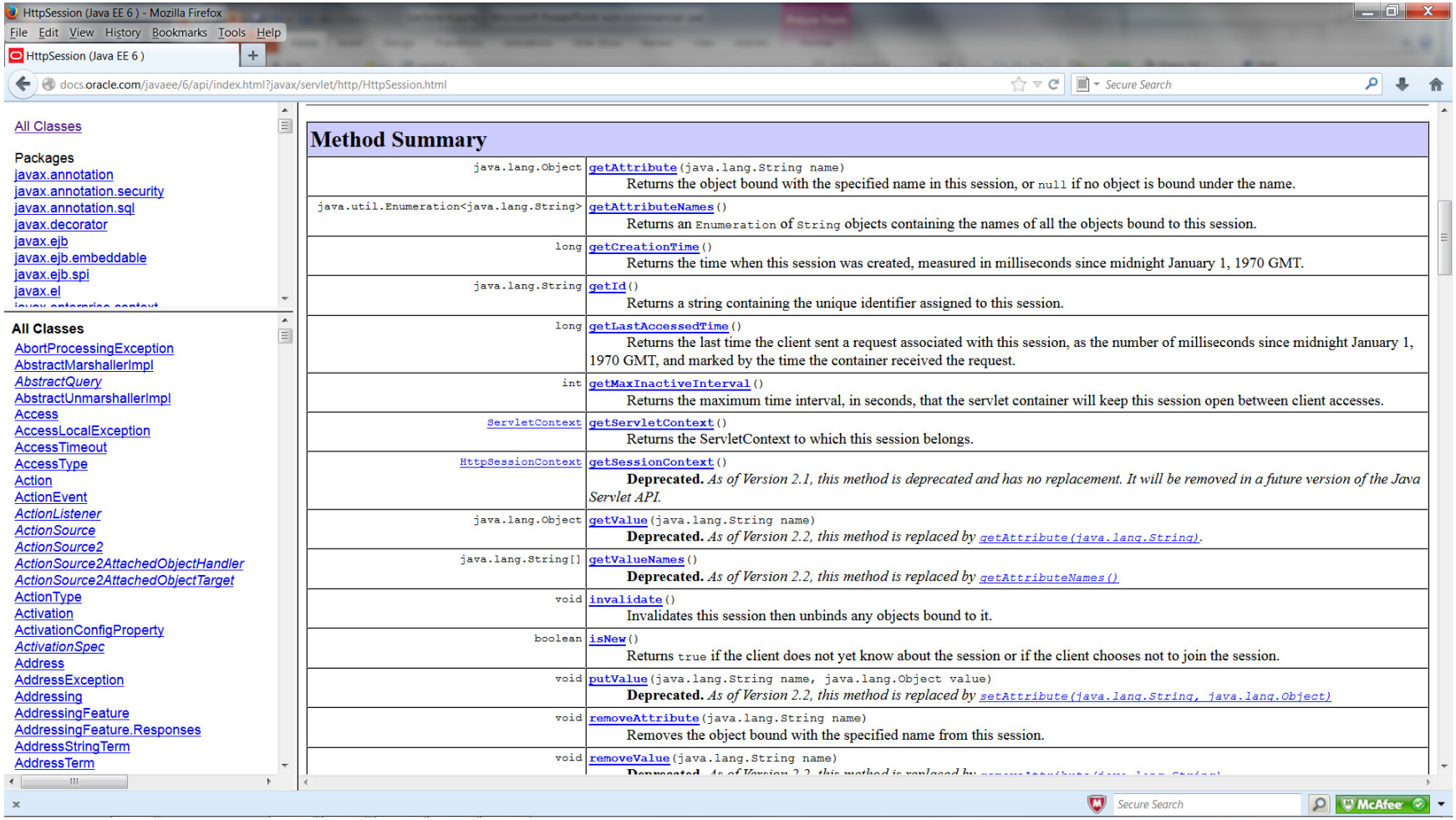
Using the Session

- The scope of an HTTPSession is the entire webapp
- The session persists until it expires on the server, or it is expired in code
- Programming issues
  - All session objects have to be Serializable (implement `java.io.Serializable`)
  - The HTTPSession can be accessed from more than one thread at a time

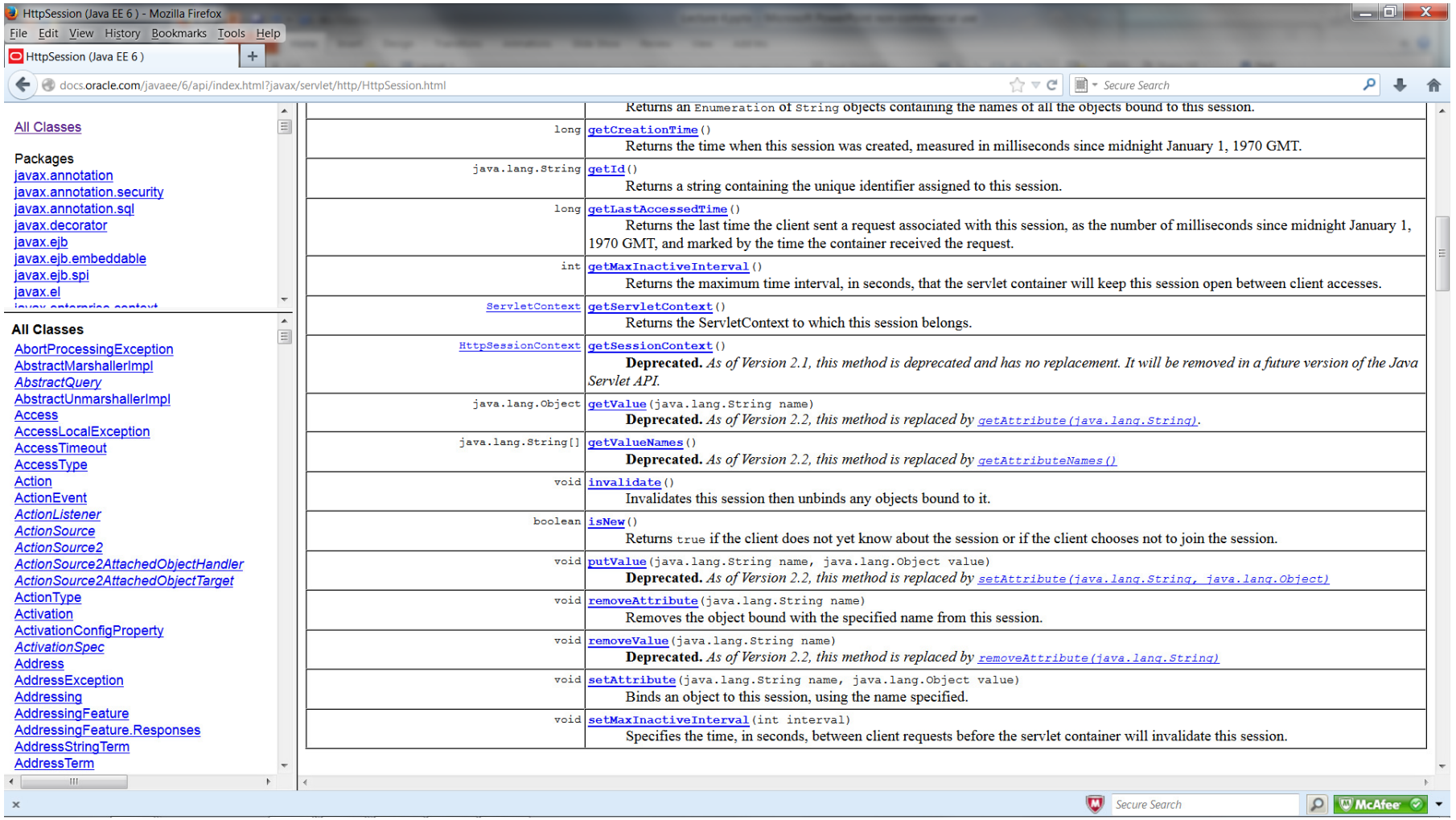
Session API


HttpSession (Java EE 6) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
HttpSession (Java EE 6)
docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/HttpSession.html
Secure Search
All Classes
Packages
javax.annotation
javax.annotation.security
javax.annotation.sql
javax.decorator
javax.ejb
javax.ejb.embeddable
javax.ejb.spi
javax.el
All Classes
AbortProcessingException
AbstractMarshallerImpl
AbstractQuery
AbstractUnmarshallerImpl
Access
AccessLocalException
AccessTimeout
AccessType
Action
ActionEvent
ActionListener
ActionSource
ActionSource2
ActionSource2AttachedObjectHandler
ActionSource2AttachedObjectTarget
ActionType
Activation
ActivationConfigProperty
ActivationSpec
Address
AddressException
Addressing
AddressingFeature
AddressingFeature.Responses
AddressStringTerm
AddressTerm
Overview Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
javax.servlet.http
Interface HttpSession
public interface HttpSession
Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user. A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using cookies or rewriting URLs.
This interface allows servlets to
View and manipulate information about a session, such as the session identifier, creation time, and last accessed time
Bind objects to sessions, allowing user information to persist across multiple user connections
When an application stores an object in or removes an object from a session, the session checks whether the object implements HttpSessionBindingListener. If it does, the servlet notifies the object that it has been bound to or unbound from the session. Notifications are sent after the binding methods complete. For session that are invalidated or expire, notifications are sent after the session has been invalidated or expired.
When container migrates a session between VMs in a distributed container setting, all session attributes implementing the HttpSessionActivationListener interface are notified.
A servlet should be able to handle cases in which the client does not choose to join a session, such as when cookies are intentionally turned off. Until the client joins the session, isNew returns true. If the client chooses not to join the session, getSession will return a different session on each request, and isNew will always return true.
Session information is scoped only to the current web application (ServletContext), so information stored in one context will not be directly visible in another.
Author:
Various
See Also:
HttpSessionBindingListener, HttpSessionContext
Secure Search
McAfee

Session API


HttpSession (Java EE 6) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
HttpSession (Java EE 6)
docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/HttpSession.html
Secure Search
All Classes
Packages
javax.annotation
javax.annotation.security
javax.annotation.sql
javax.decorator
javax.ejb
javax.ejb.embeddable
javax.ejb.spi
javax.el
All Classes
AbortProcessingException
AbstractMarshallerImpl
AbstractQuery
AbstractUnmarshallerImpl
Access
AccessLocalException
AccessTimeout
AccessType
Action
ActionEvent
ActionListener
ActionSource
ActionSource2
ActionSource2AttachedObjectHandler
ActionSource2AttachedObjectTarget
ActionType
Activation
ActivationConfigProperty
ActivationSpec
Address
AddressException
Addressing
AddressingFeature
AddressingFeature.Responses
AddressStringTerm
AddressTerm
Method Summary
java.lang.Object getAttribute(java.lang.String name)
Returns the object bound with the specified name in this session, or null if no object is bound under the name.
java.util.Enumeration<java.lang.String> getAttributeNames()
Returns an Enumeration of String objects containing the names of all the objects bound to this session.
long getCreationTime()
Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
java.lang.String getId()
Returns a string containing the unique identifier assigned to this session.
long getLastAccessedTime()
Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container received the request.
int getMaxInactiveInterval()
Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
ServletContext getServletContext()
Returns the ServletContext to which this session belongs.
HttpSessionContext getSessionContext()
Deprecated. As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.
java.lang.Object getValue(java.lang.String name)
Deprecated. As of Version 2.2, this method is replaced by getAttribute(java.lang.String).
java.lang.String[] getValueNames()
Deprecated. As of Version 2.2, this method is replaced by getAttributeNames()
void invalidate()
Invalidates this session then unbinds any objects bound to it.
boolean isNew()
Returns true if the client does not yet know about the session or if the client chooses not to join the session.
void putValue(java.lang.String name, java.lang.Object value)
Deprecated. As of Version 2.2, this method is replaced by setAttribute(java.lang.String, java.lang.Object)
void removeAttribute(java.lang.String name)
Removes the object bound with the specified name from this session.
void removeValue(java.lang.String name)
Secure Search
McAfee

Session API


HttpSession (Java EE 6) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
HttpSession (Java EE 6)
docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/HttpSession.html
Secure Search
All Classes
Packages
javax.annotation
javax.annotation.security
javax.annotation.sql
javax.decorator
javax.ejb
javax.ejb.embeddable
javax.ejb.spi
javax.el
All Classes
AbortProcessingException
AbstractMarshallerImpl
AbstractQuery
AbstractUnmarshallerImpl
Access
AccessLocalException
AccessTimeout
AccessType
Action
ActionEvent
ActionListener
ActionSource
ActionSource2
ActionSource2AttachedObjectHandler
ActionSource2AttachedObjectTarget
ActionType
Activation
ActivationConfigProperty
ActivationSpec
Address
AddressException
Addressing
AddressingFeature
AddressingFeature.Responses
AddressStringTerm
AddressTerm
Returns an Enumeration of String objects containing the names of all the objects bound to this session.
long getCreationTime()
Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
java.lang.String getId()
Returns a string containing the unique identifier assigned to this session.
long getLastAccessedTime()
Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container received the request.
int getMaxInactiveInterval()
Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
ServletContext getServletContext()
Returns the ServletContext to which this session belongs.
HttpSessionContext getSessionContext()
Deprecated. As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.
java.lang.Object getValue(java.lang.String name)
Deprecated. As of Version 2.2, this method is replaced by getAttribute(java.lang.String).
java.lang.String[] getValueNames()
Deprecated. As of Version 2.2, this method is replaced by getAttributeNames()
void invalidate()
Invalidates this session then unbinds any objects bound to it.
boolean isNew()
Returns true if the client does not yet know about the session or if the client chooses not to join the session.
void putValue(java.lang.String name, java.lang.Object value)
Deprecated. As of Version 2.2, this method is replaced by setAttribute(java.lang.String, java.lang.Object)
void removeAttribute(java.lang.String name)
Removes the object bound with the specified name from this session.
void removeValue(java.lang.String name)
Deprecated. As of Version 2.2, this method is replaced by removeAttribute(java.lang.String)
void setAttribute(java.lang.String name, java.lang.Object value)
Binds an object to this session, using the name specified.
void setMaxInactiveInterval(int interval)
Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.
Secure Search
McAfee

Session API

- In the `HttpServletRequest` class:
  - `HttpSession getSession()`
  - `HttpSession getSession( boolean create)`
  - Returns the current `HttpSession` associated with this request, or If there is no current session and `create` is `true`, returns a new session.
  - Default: `create` is `true`

- `String getRequestedSessionId()`
  - Returns the session ID.

Session API (cont)

- The `HttpSession` class:

`void setAttribute( String name,Object value)`

  - Binds an object to the specified name in this session.

`Object getAttribute( String name)`

  - Returns the object bound to the specified name in this session, or
  - Returns `null` if no object is bound to the name.

`void removeAttribute( String name)`

  - Removes the binding with the specified name and the object from this session.
  - Can do the same by calling `setAttribute(name, null);`

Session API (cont)

`String getId()`

- Returns the session ID.

`Enumeration getAttributeNames()`

- Returns an Enumeration of string objects containing the names of all the objects bound to this session.

`long getCreationTime()`

- Returns the time when this session was created in ms since the epoch

`long getLastAccessedTime()`

- Returns the last time the client sent a request associated with this session.

Session API (cont)

```
int getMaxInactiveInterval()
```

- Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
- Tomcat default: 30 minutes

```
void setMaxInactiveInterval( int interval)
```

- Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.
- A negative time indicates the session should never expire.

Session API (cont)

`void invalidate()`

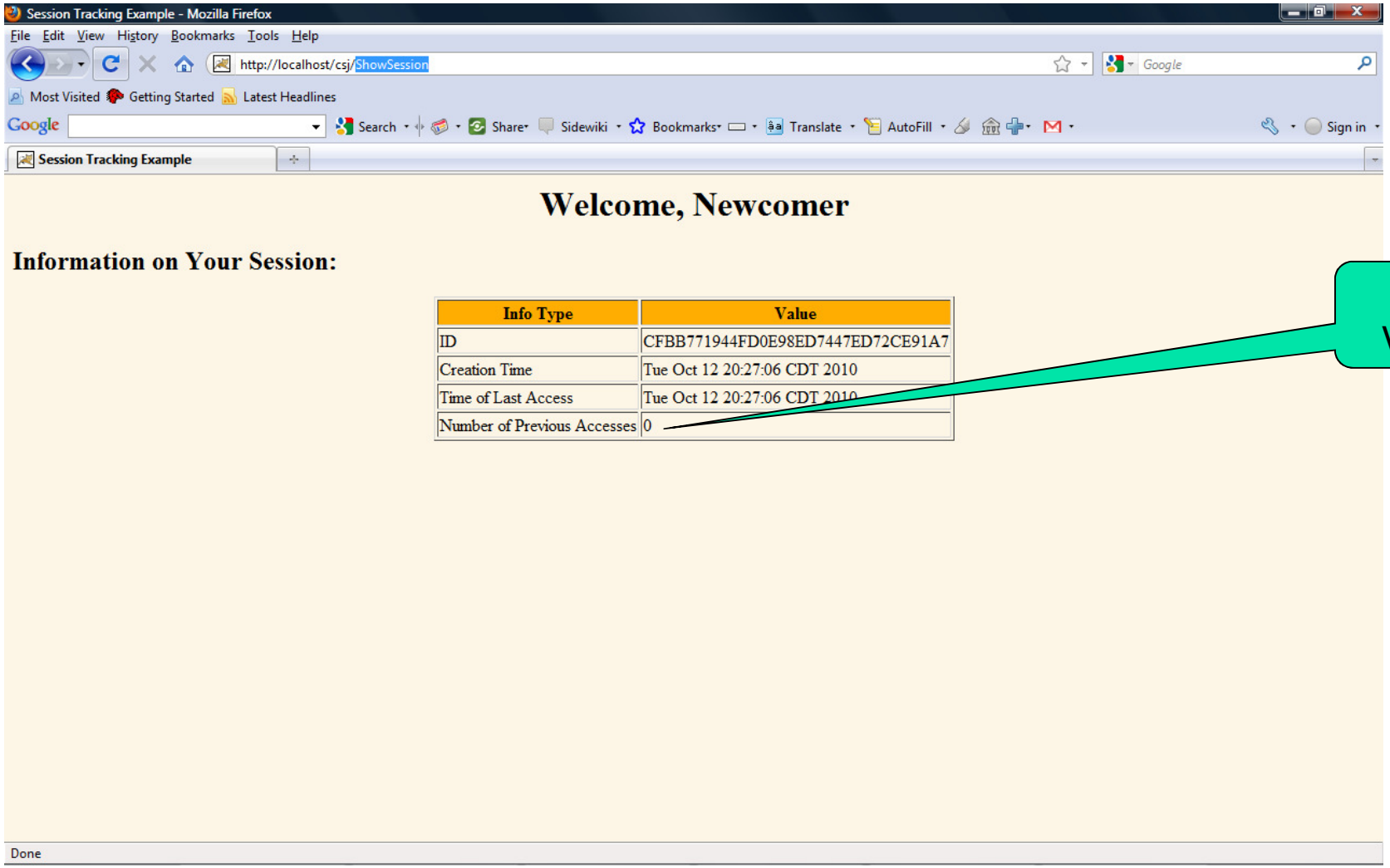
- Invalidates this session and unbinds any objects in this session.

`boolean isNew()`

- Returns `true`
  - if the client does not yet know about the session, or
  - if the client chooses not to join the session (i.e. cookies are disabled).

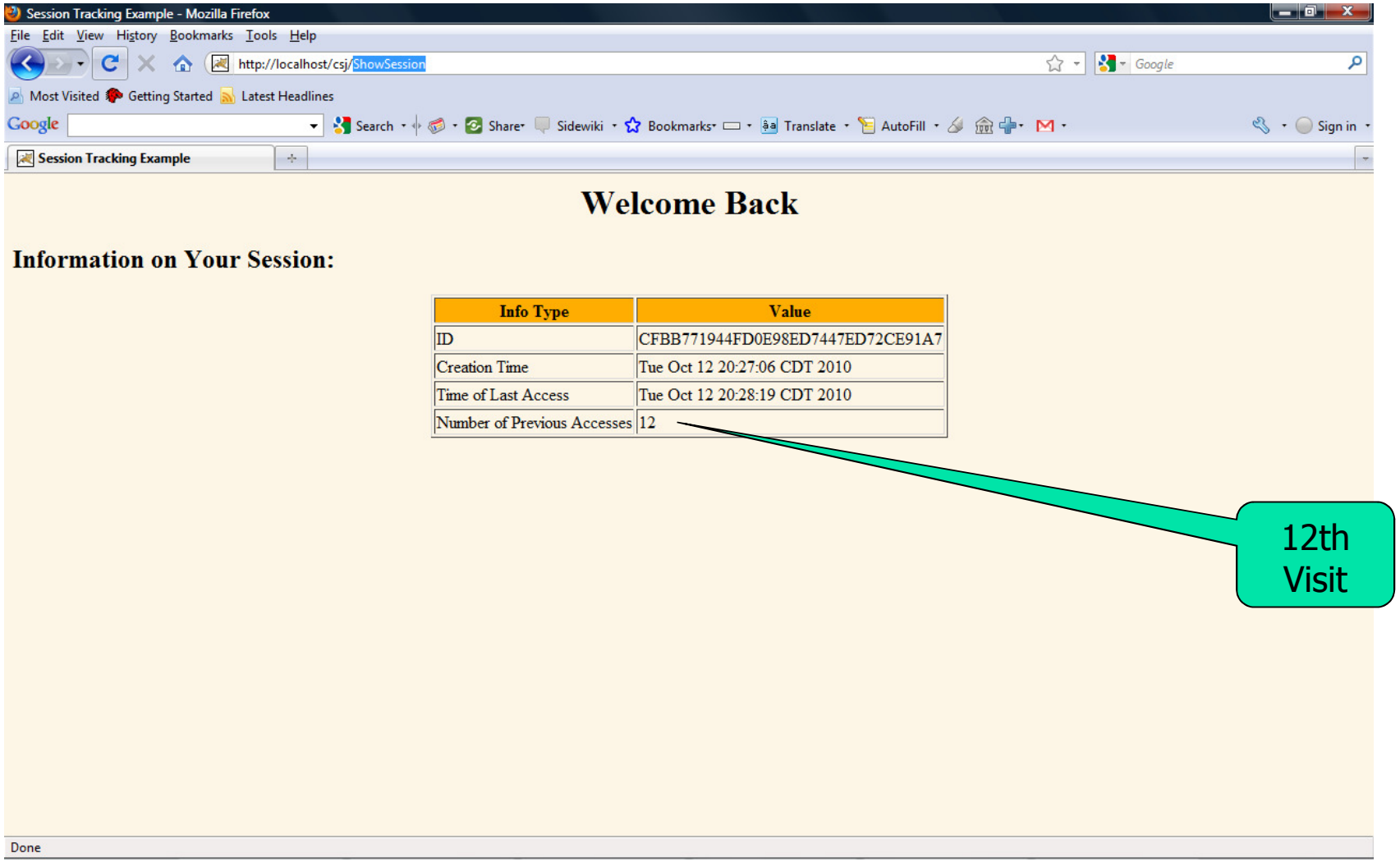
The ShowSession.java Run

http://localhost/csj/ShowSession



ShowSession.java

http://localhost/csj/ShowSession



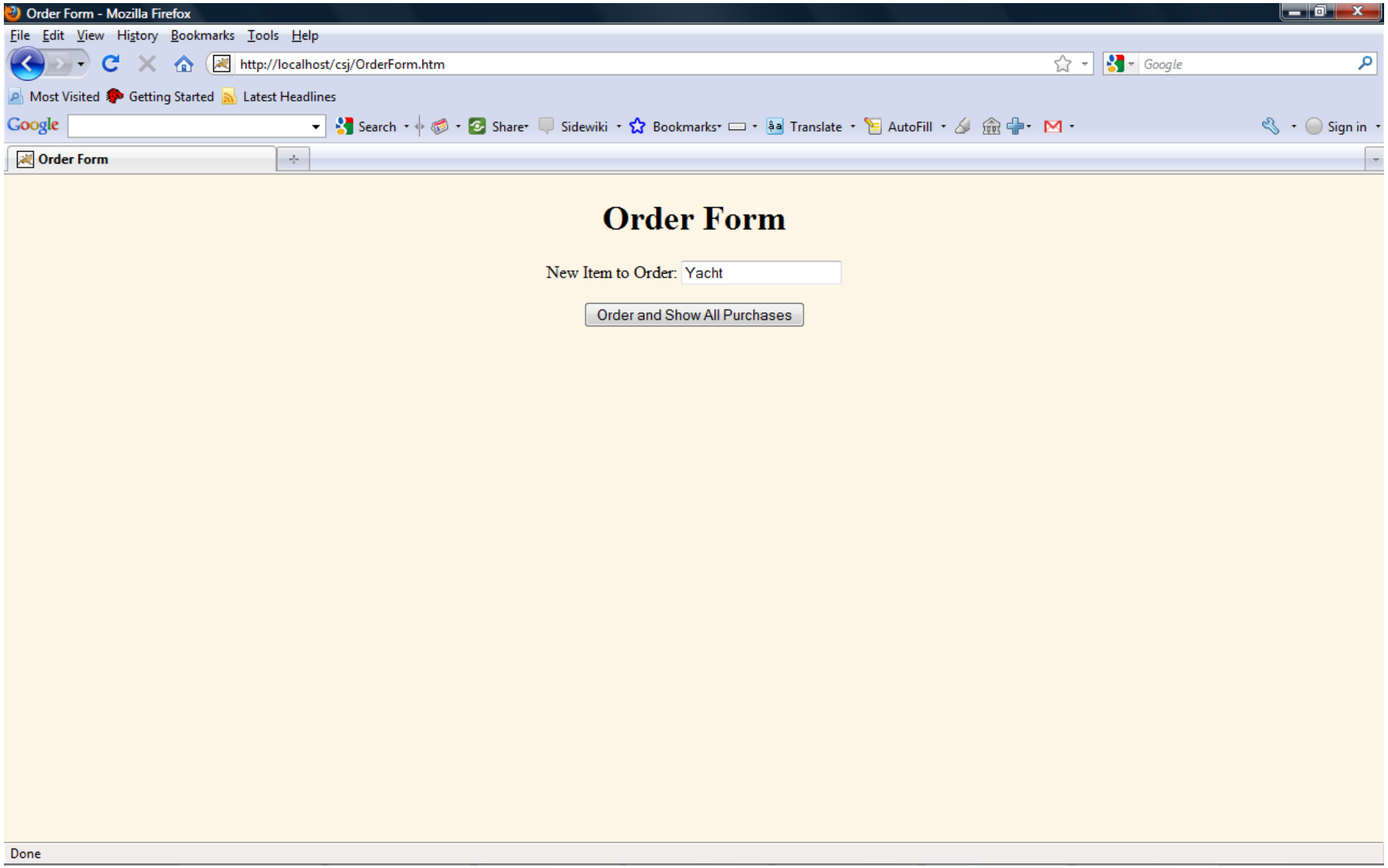
Accumulating a List of User Data

- Mutable data structures (*mutable* data structure such as an array,List, Map) are often used to maintain a set of data associated with the user.
- In the next example, we present a simplified example in which we maintain a basic list of items that each user has purchased.

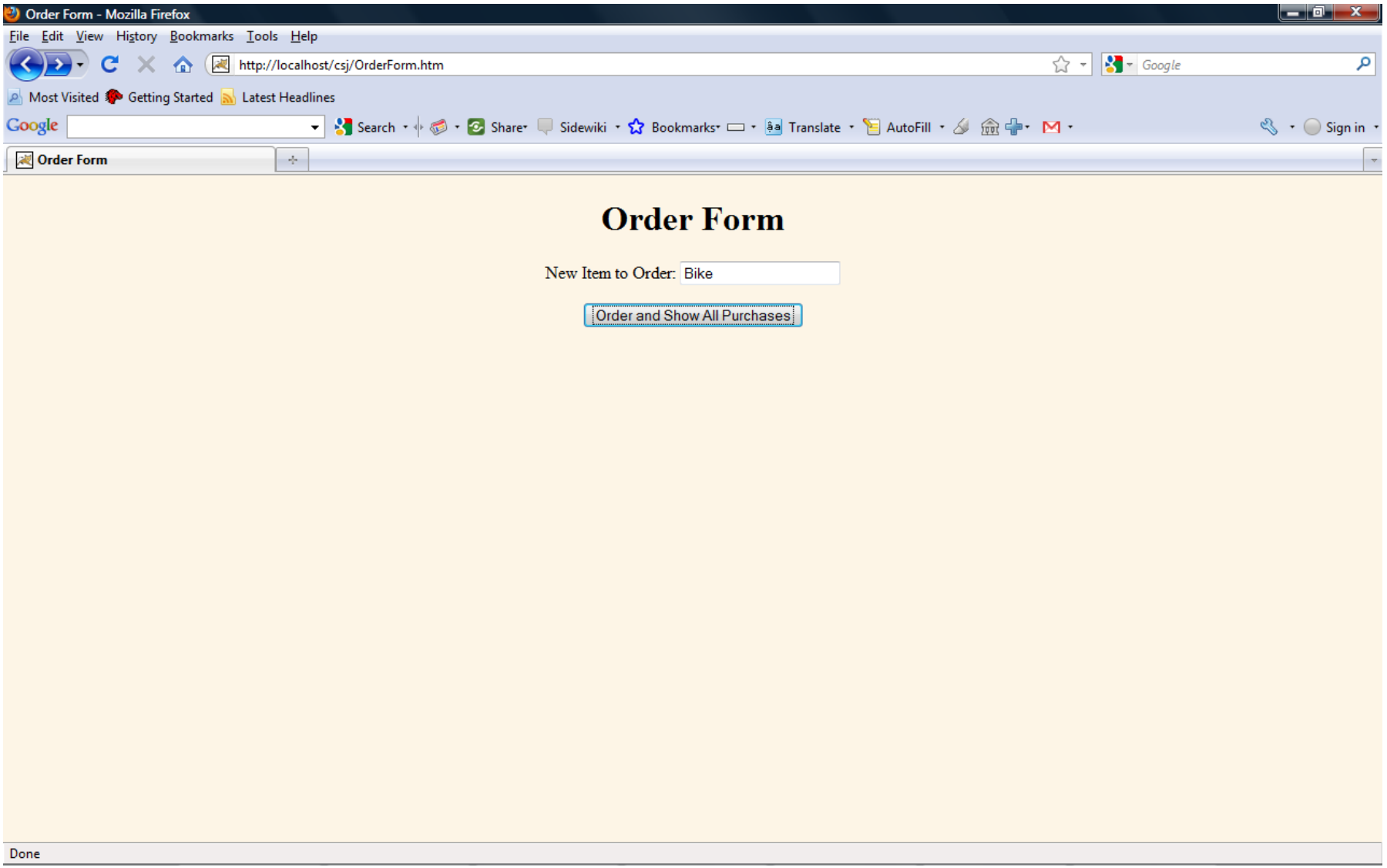
Accumulating a List of User Data

- ShowItems.java. Servlet that displays a list of items being ordered. Accumulates them in an ArrayList with no attempt at detecting repeated items. Uses OrderForm.html to collect data.
- OrderForm.html. Front end to the ShowItems servlet.

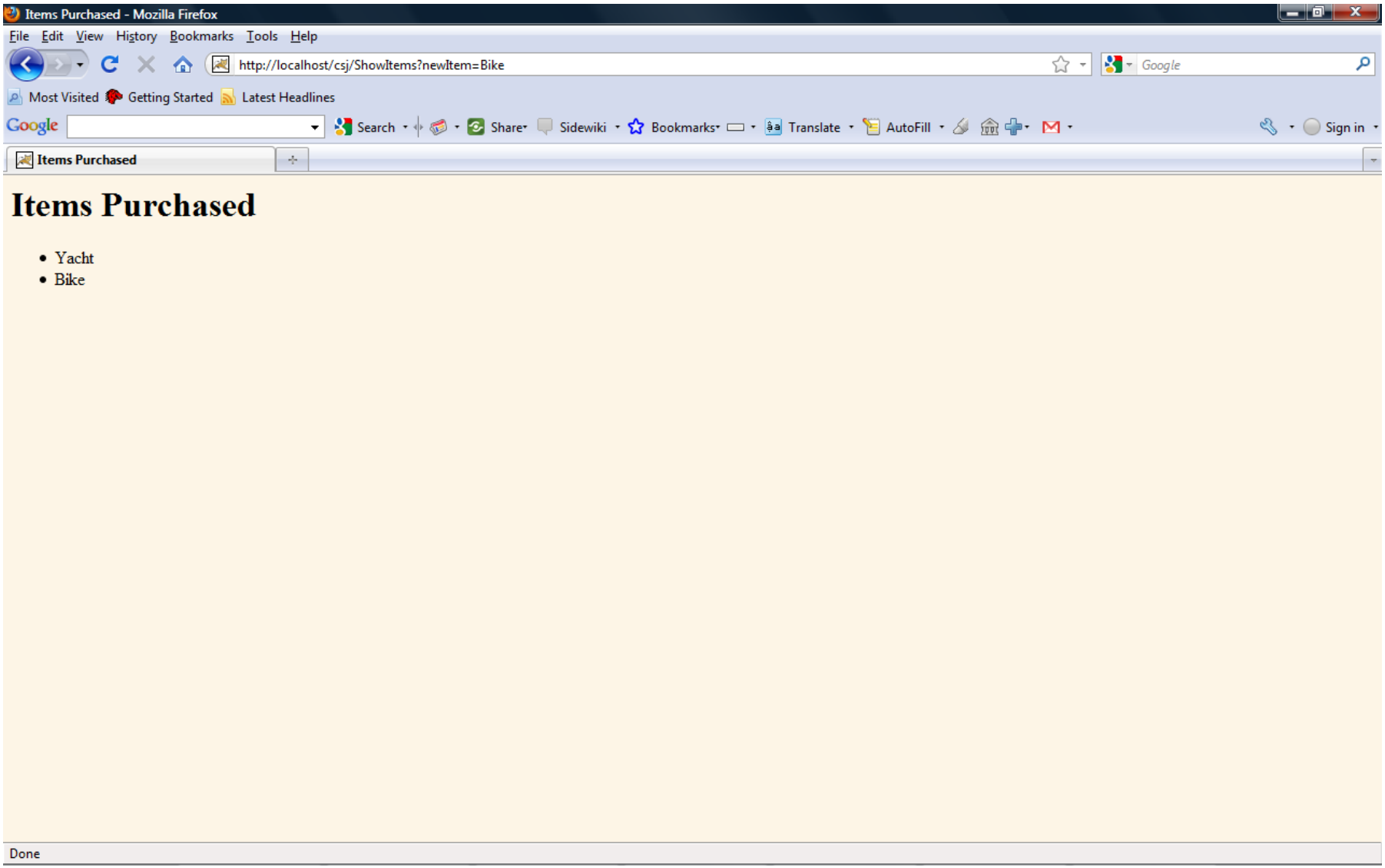
Accumulating a List of User Data



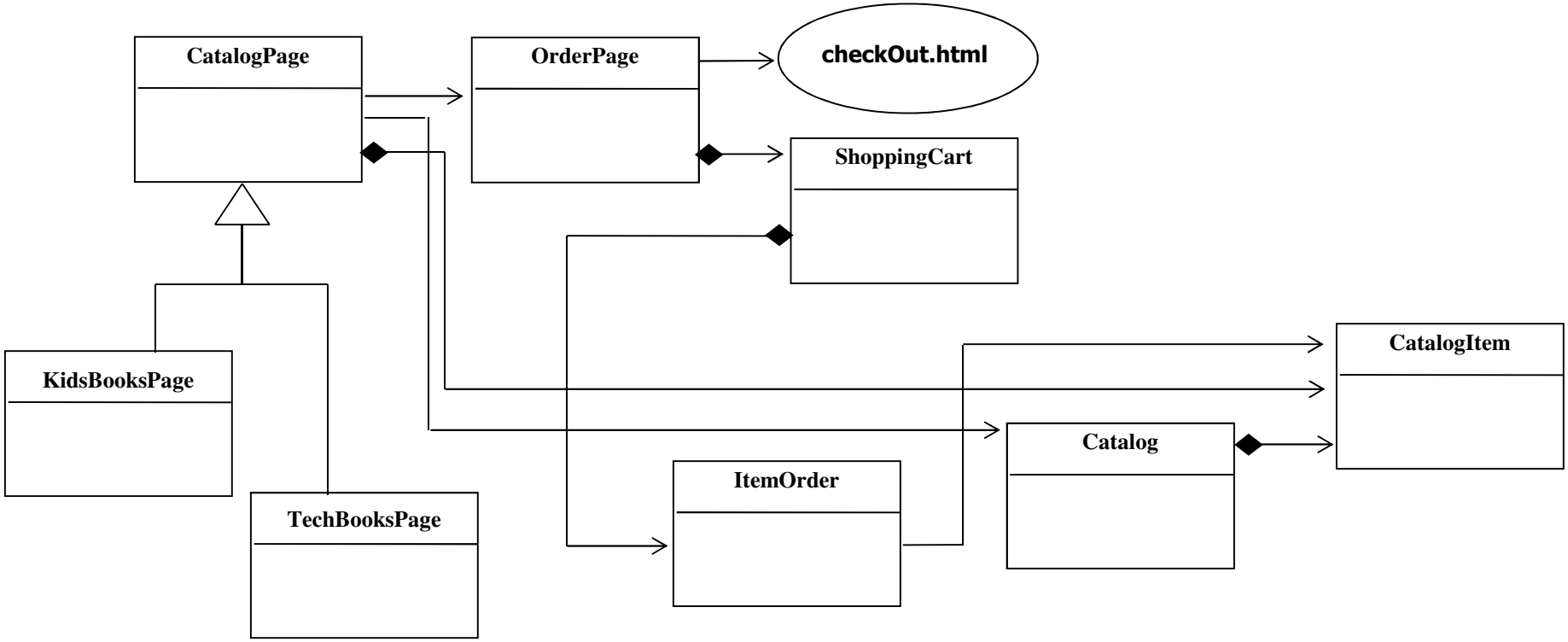
Accumulating a List of User Data



Accumulating a List of User Data

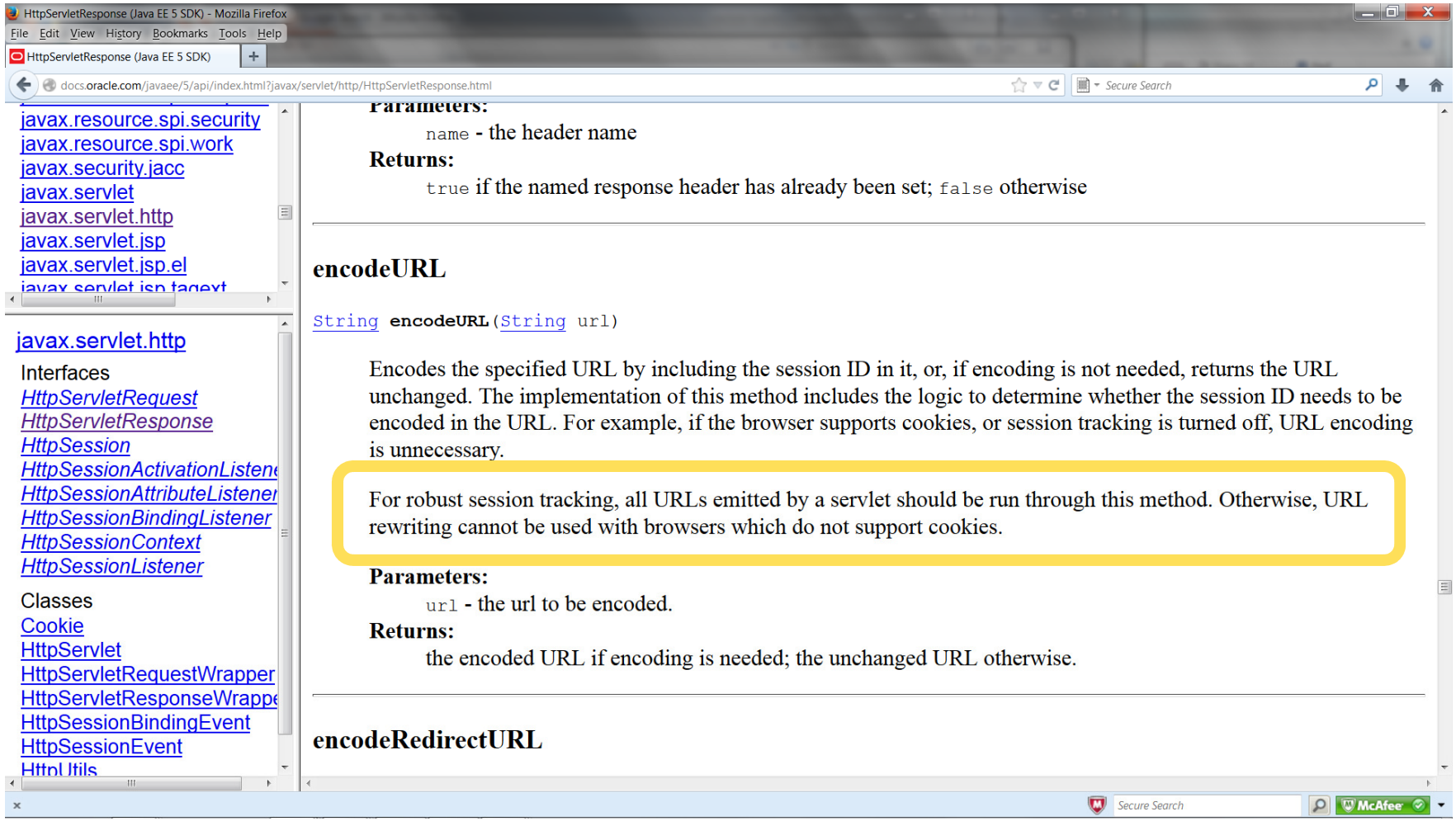


The Shopping Cart Example



The Shopping Cart Example

- For this example we will use **encodeURL** method from **HttpServletResponse , why?** We want to include the sessionID for session tracking between the servlets



The Shopping Cart Example

- CatalogPage.java. Base class for pages showing catalog entries. Servlets that extend this class must specify the catalog entries that they are selling and the page title.
- KidsBooksPage.java. A specialization of the CatalogPage servlet that displays a page selling three famous kids-book series. Orders are sent to the OrderPage servlet.

The Shopping Cart Example

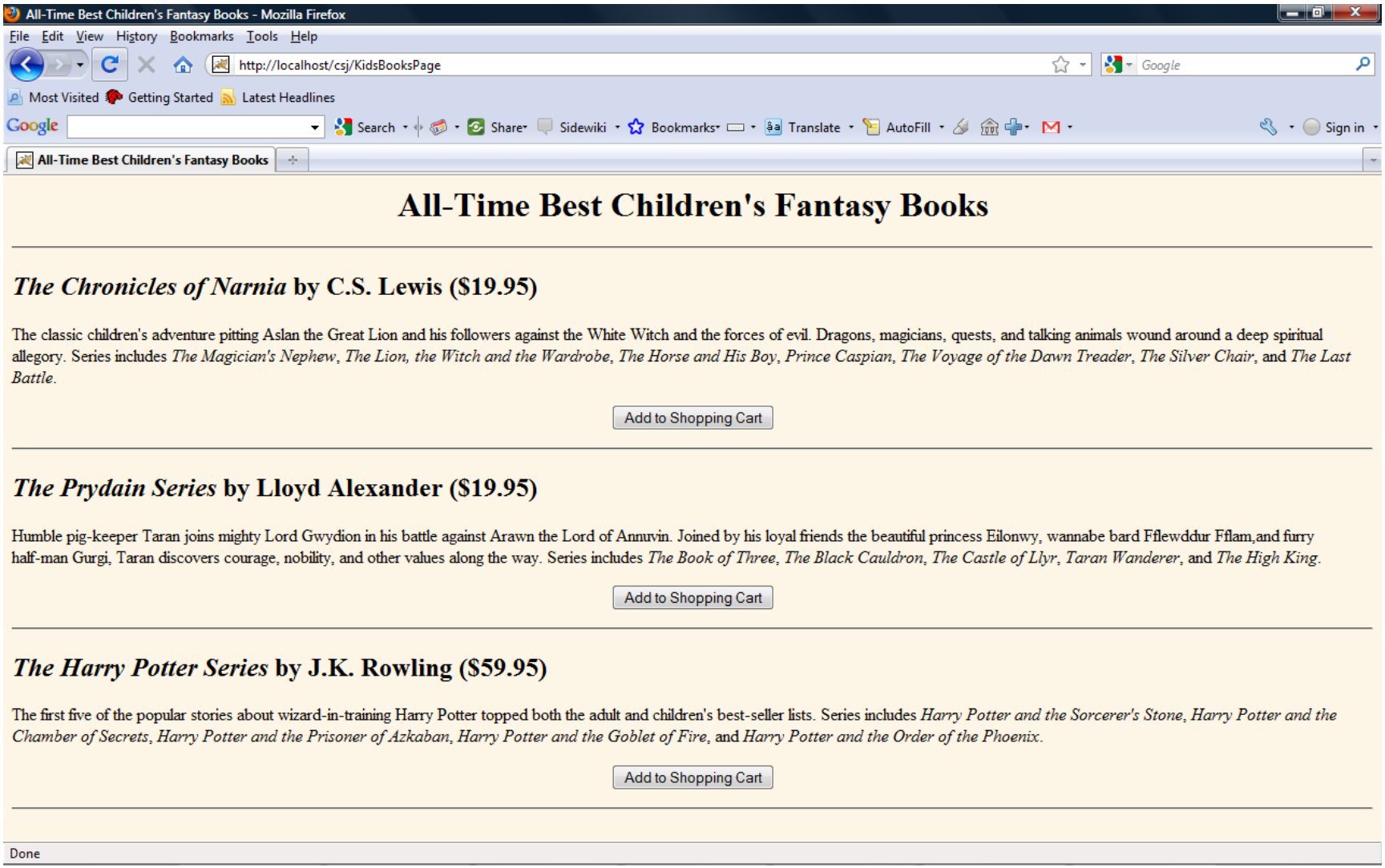
- TechBooksPage.java. A specialization of the CatalogPage servlet that displays a page selling two famous computer books. Orders are sent to the OrderPage servlet.
- OrderPage.java. Servlet that records new item orders (if any) and then displays all items in the shopping cart.
- Checkout.html. Page that handles the final checkout.
- ShoppingCart.java. A shopping cart data structure used to track orders. The OrderPage servlet associates one of these carts with each user session.

The Shopping Cart Example

- CatalogItem.java. Describes a catalog item for an online store. Elements of this class are *indirectly* stored in the ShoppingCart (after they are embedded within an ItemOrder).
- ItemOrder.java. Class that associates a catalog Item with a specific order by keeping track of the number ordered and the total price. Elements of this class are *directly* stored in the ShoppingCart (after each is populated with a CatalogItem).
- Catalog.java. A catalog that lists the items available in inventory.

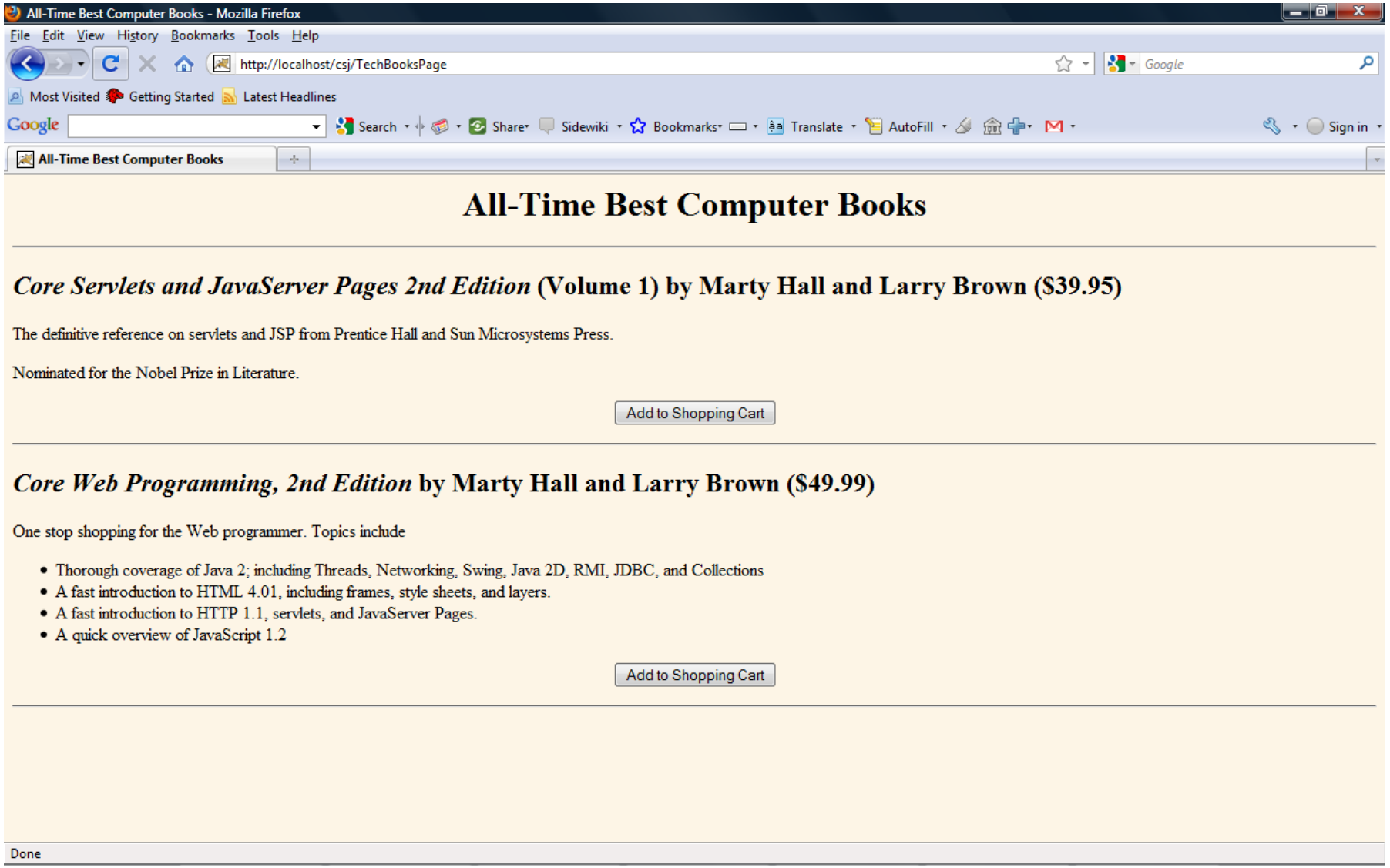
The Shopping Cart

http://localhost/csj/KidsBooksPage



The Shopping Cart

http://localhost/csj/TechBooksPage



The Shopping Cart Example

- **Handling the Orders:**
  - KidsBooksPage and TechBooksPage are of supertype Servlet CatalogPage

CatalogPage - Notepad

File Edit Format View Help

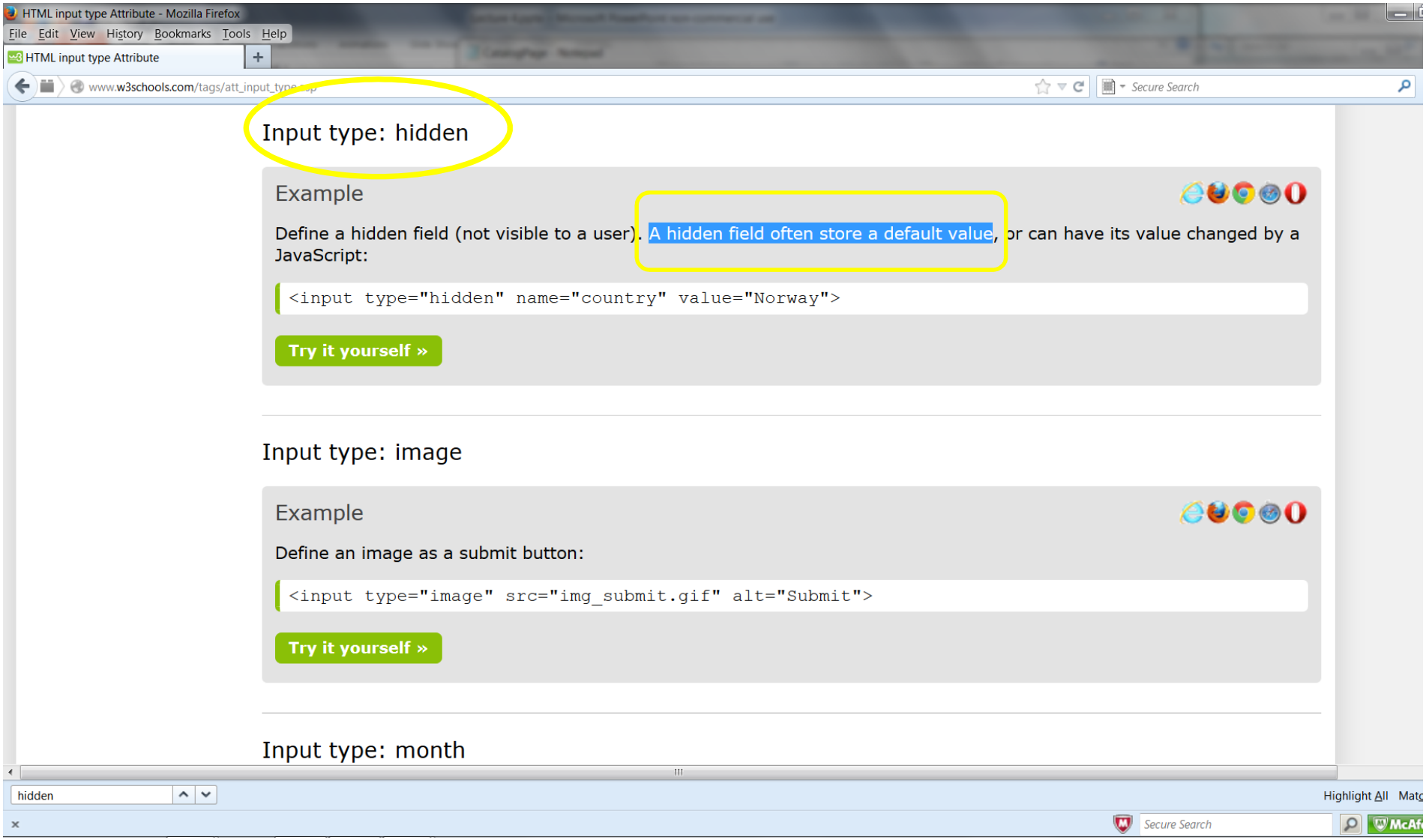
```
            "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + title + "</H1>");
  CatalogItem item;
  for(int i=0; i<items.length; i++) {
   out.println("<HR>");
   item = items[i];
   // Show error message if subclass lists item ID
   // that's not in the catalog.
   if (item == null) {
    out.println("<FONT COLOR=\"RED\">" +
                "Unknown item ID " + itemIDs[i] +
                "</FONT>");
   } else {
    out.println();
    String formURL =
      "/csj/OrderPage";
    // Pass URLs that reference own site through encode
    formURL = response.encodeURL(formURL);
    out.println
     ("<FORM ACTION=\"" + formURL + "\">\n" +
      "<INPUT TYPE=\"HIDDEN\" NAME=\"itemID\" " +
      "       VALUE=\"" + item.getItemID() + "\">\n" +
      "<H2>" + item.getShortDescription() +
      " ($" + item.getCost() + ")</H2>\n" +
      item.getLongDescription() + "\n" +
      "<P>\n<CENTER>\n" +
      "<INPUT TYPE=\"SUBMIT\" " +
      "VALUE=\"Add to Shopping Cart\">\n" +
      "</CENTER>\n<P>\n</FORM>");
  }
```

What is Type Hidden ?
See Next Slide

Ln 106, Col 3

The Shopping Cart

Input Type = Hidden



The Shopping Cart Example

- **Handling the Orders:**
  - OrderPage.java is a servlet that handles the orders coming from the various catalog pages. It uses session tracking to associate a shopping cart with each user.
  - Since each user has a separate session, it is unlikely that multiple threads will be accessing the same shopping cart simultaneously.
  - However, a few circumstances in which concurrent access could occur, such as when a single user has multiple browser windows open and sends updates from more than one in quick succession.

The Shopping Cart Example

- **Handling the Orders:**
  - So, what is the remedy?
    - Your code should synchronize access based upon the session object.
    - This synchronization prevents other threads that use the same session from accessing the data concurrently, while still allowing simultaneous requests from different users to proceed.

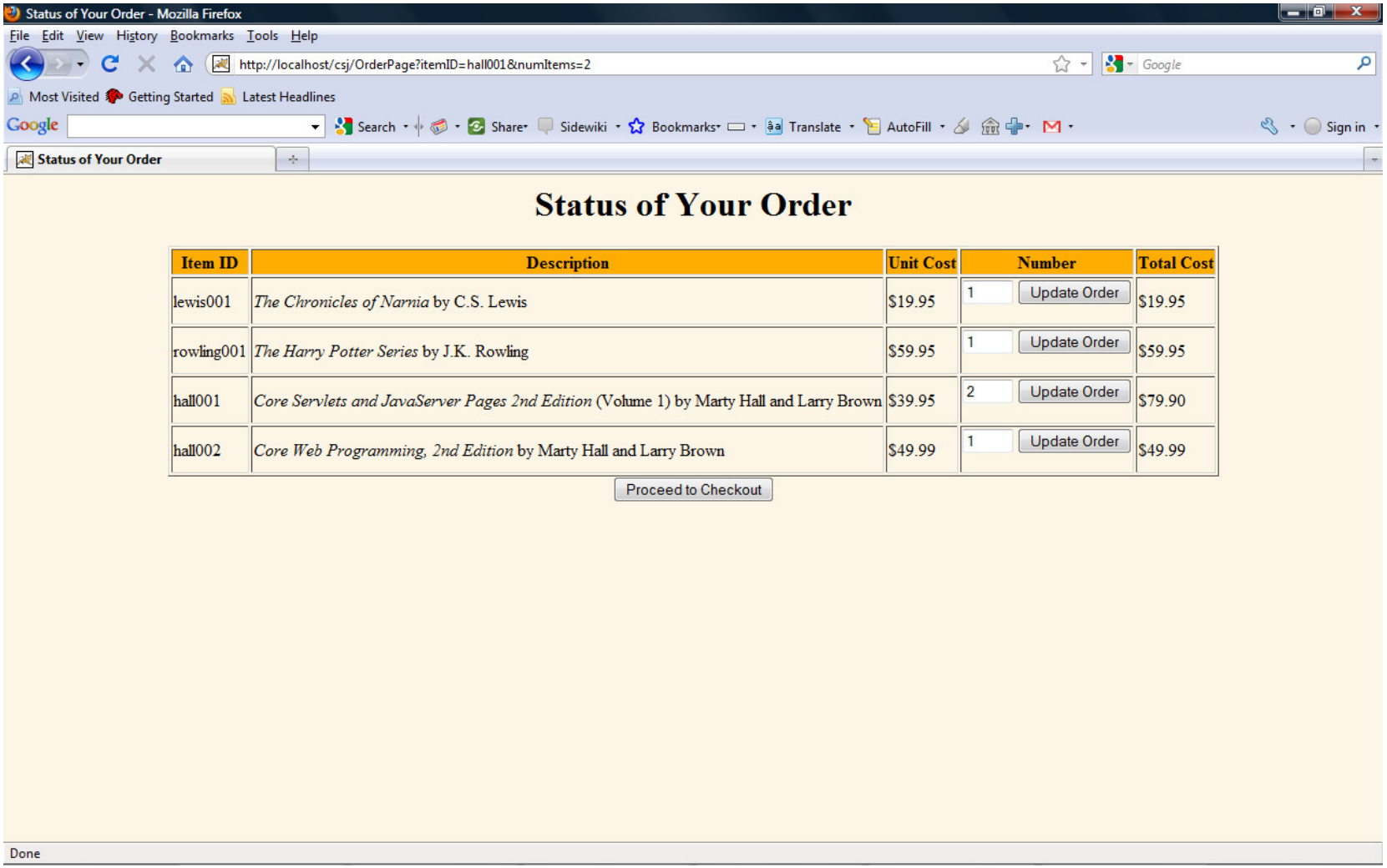
The Shopping Cart Example

- **See below how to use Synchronized on the session object**

```
public class OrderPage extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession();
    ShoppingCart cart;
    synchronized(session) {
      cart = (ShoppingCart)session.getAttribute("shoppingCart");
      // New visitors get a fresh shopping cart.
      // Previous visitors keep using their existing cart.
      if (cart == null) {
        cart = new ShoppingCart();
        session.setAttribute("shoppingCart", cart);
      }
```

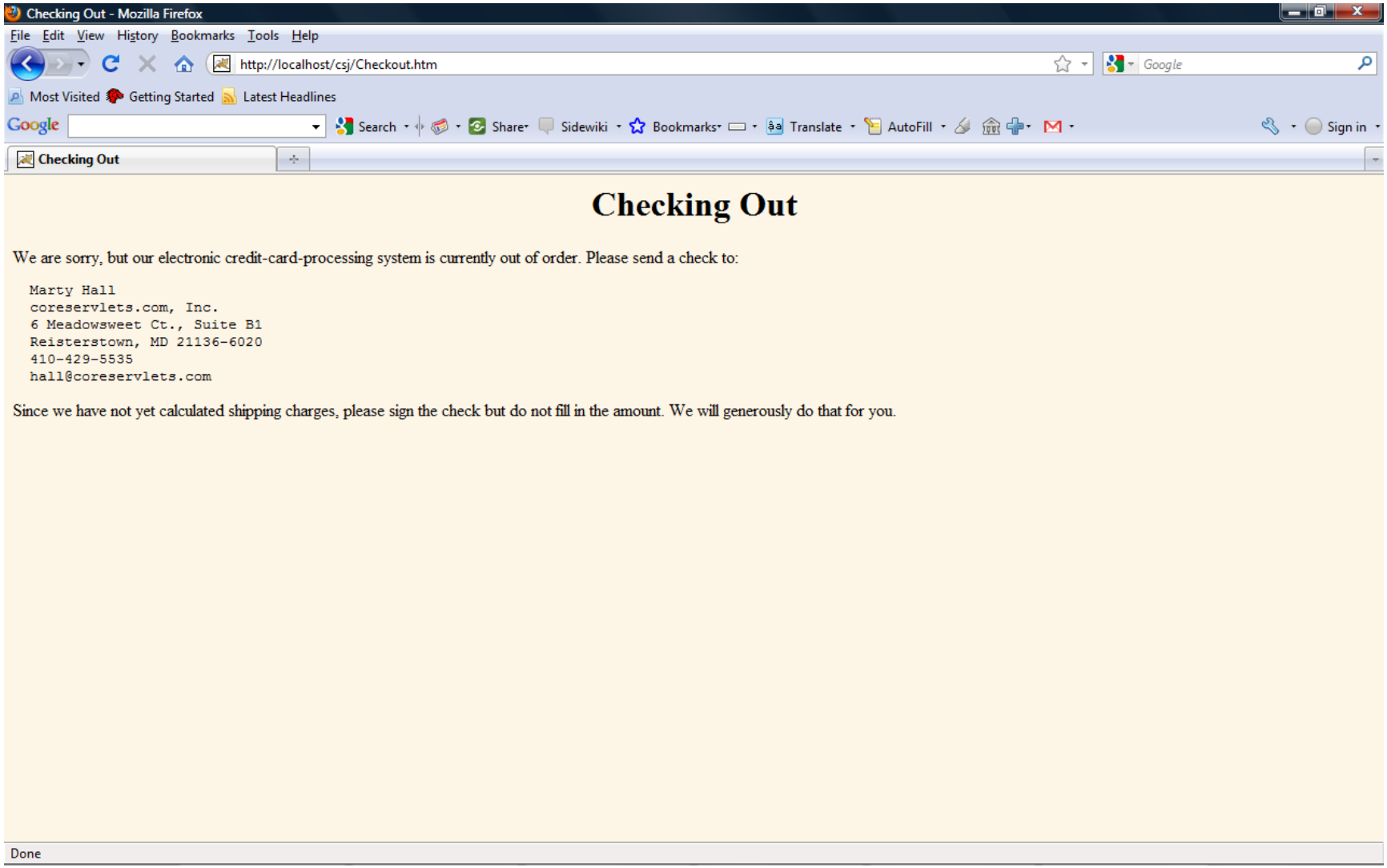
The Shopping Cart

http://localhost/csj/OrderPage?itemID=hall001&numItems=2



The Shopping Cart

http://localhost/csj/Checkout.htm

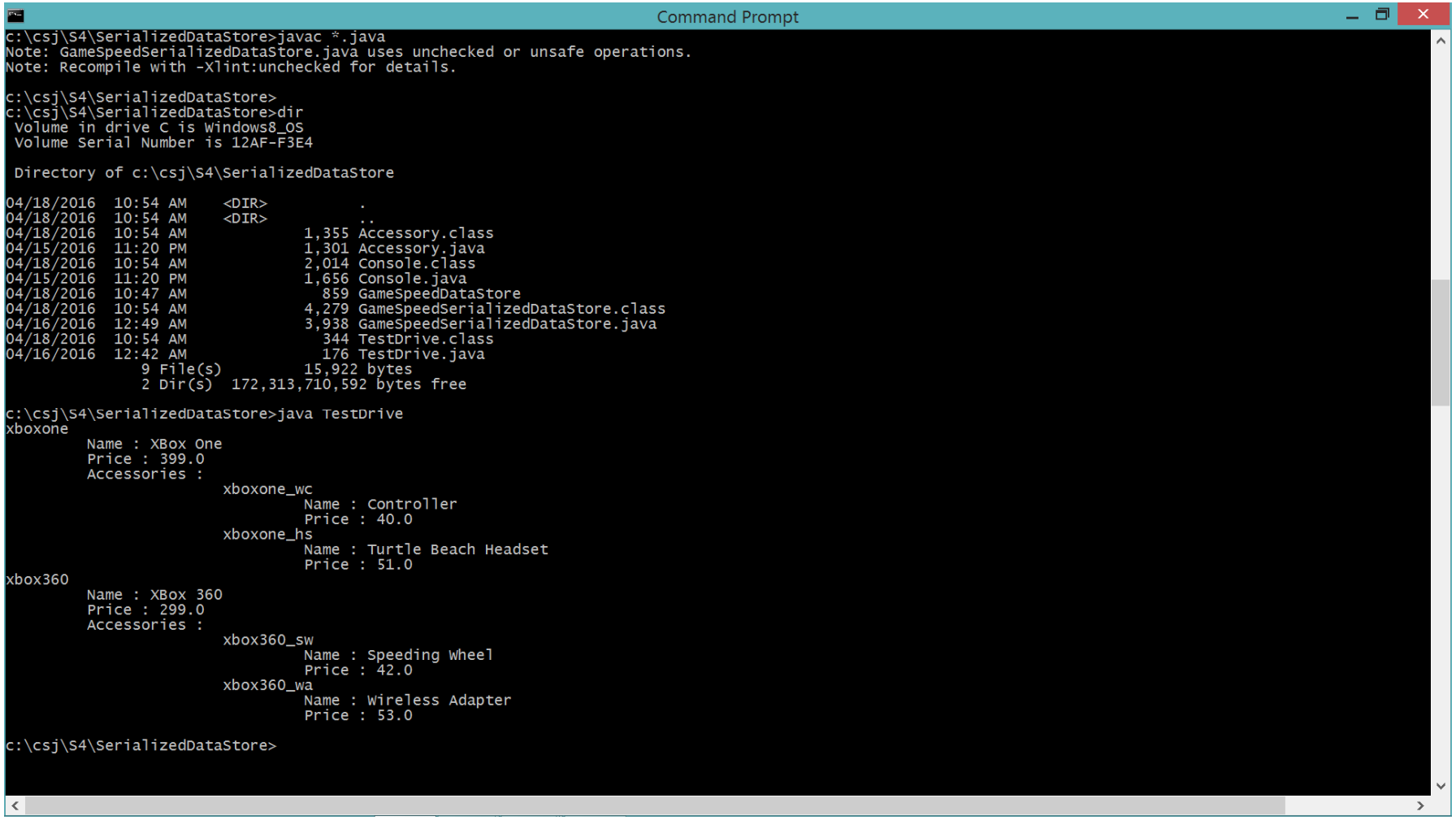


Handling the Data Store/Data Model

Handling the Data Store/Data Model

- **Four possible ways:**
  - **Use static data in class and serialized objects/hashmaps and store serialized objects in a file**
    - **(the Servlet loads the data from the serialized file)**
  - **Use XML file to specify data and use SAX parser to load the data**
    - **(the Servlet loads the data from the product catalog XML file)**
  - **Use web.xml to specify the data**
    - **(the Servlet loads the data from the deployment descriptor web XML file)**
  - **Use plaintext file to store plaintext data**
    - **(the Servlet loads the data from the plaintext file)**

Serialized Objects/HashMaps
for the Data Store/Data Model



SAX parser to load the data
for the Data Store/Data Model