

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

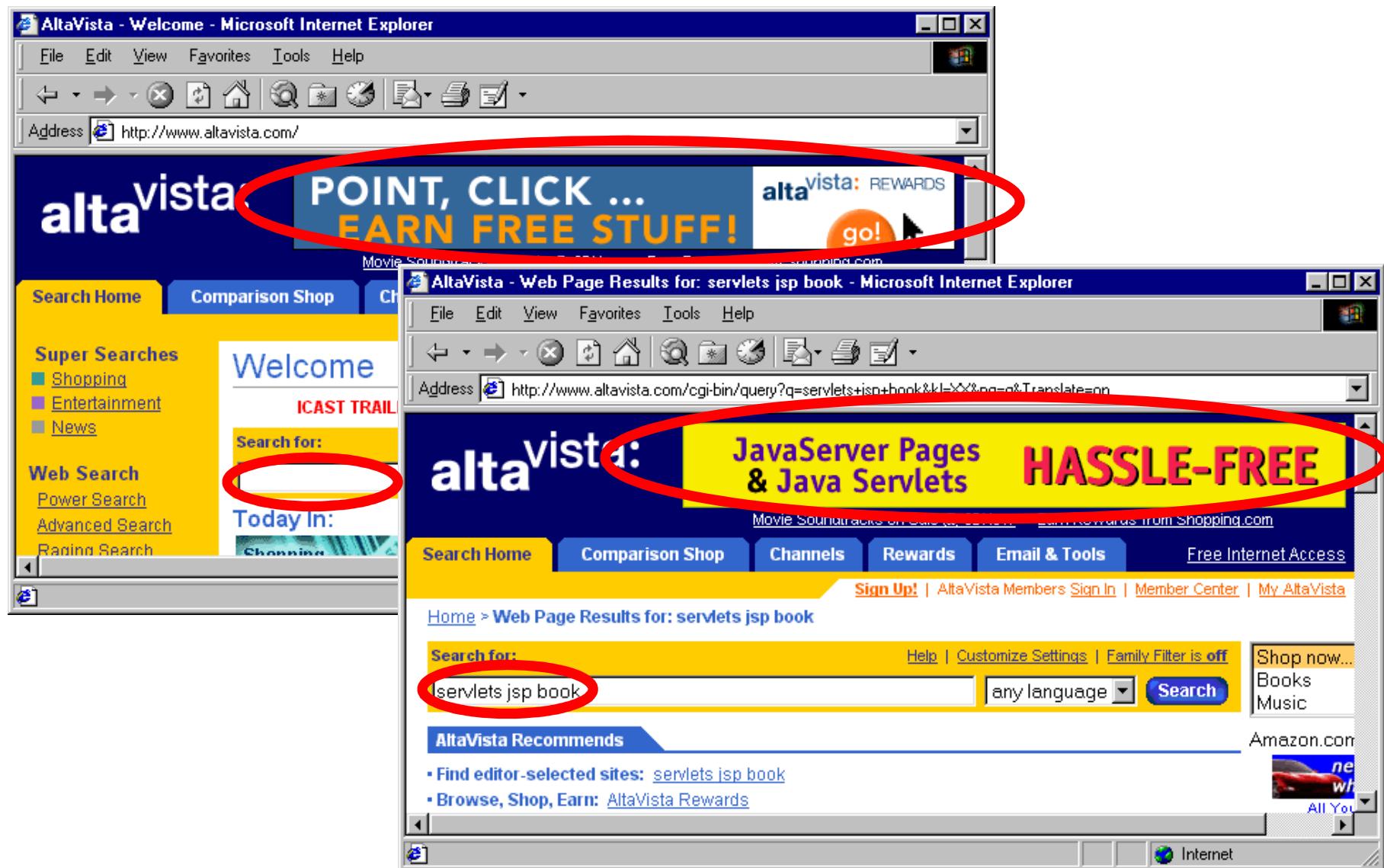
Cookies

- 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



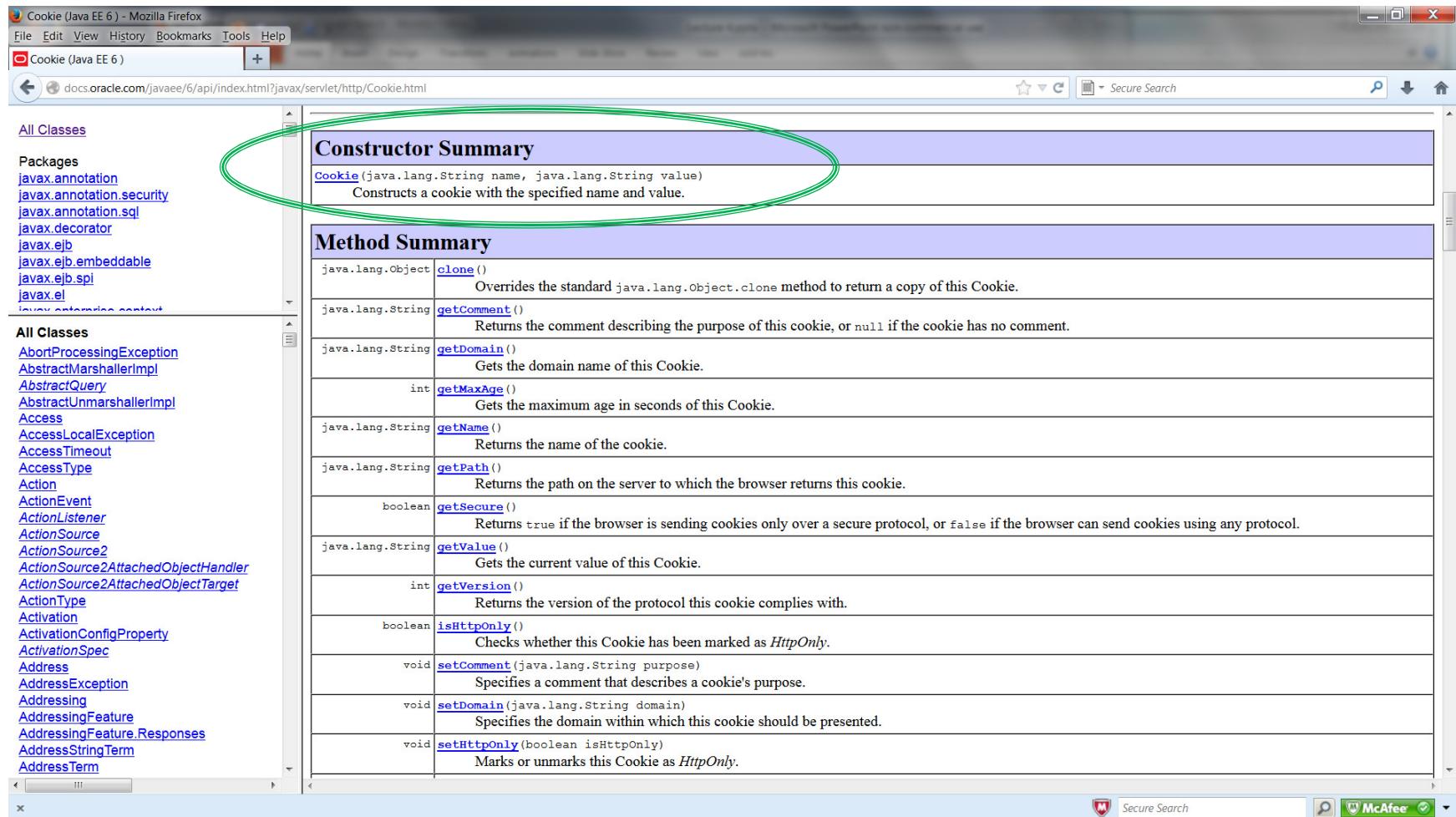
Problems with Cookies

- **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

The screenshot shows a Mozilla Firefox window displaying the Java EE 6 API documentation for the `Cookie` class. The URL in the address bar is `docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/Cookie.html`. The page title is "Cookie (Java EE 6)". The left sidebar contains a navigation tree with categories like "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", and "javax.servlet.jsp.jstl.tlv". Under "javax.servlet.http", there are links for "Interfaces" (including `HttpServletRequest`, `HttpServletResponse`, `HttpSession`, `HttpSessionActivationListener`, `HttpSessionAttributeListener`, `HttpSessionBindingListener`, `HttpSessionContext`, `HttpSessionListener`, `Part`) and "Classes" (including `Cookie`, `HttpServlet`, `HttpServletRequestWrapper`, `HttpServletResponseWrapper`, `HttpSessionBindingEvent`, `HttpSessionEvent`, `HttpUtils`). The main content area shows the `Class Cookie` documentation. It includes the package (`javax.servlet.http`), inheritance (`java.lang.Object`), implemented interfaces (`java.io.Serializable`, `java.lang.Cloneable`), and the class definition (`public class Cookie extends java.lang.Object implements java.lang.Cloneable, java.io.Serializable`). A detailed description follows, mentioning session management, cookie attributes, and compatibility with HTTP 1.1. The page also includes sections for "All Implemented Interfaces", "Author" (Various), and links for "PREV CLASS", "NEXT CLASS", "SUMMARY", "FIELD", "CONSTR", "METHOD", "FRAMES", "NO FRAMES", and "DETAIL". The bottom of the browser window shows the McAfee security status.

Cookie API



The screenshot shows a Mozilla Firefox browser window displaying the Java EE 6 API documentation for the `Cookie` class. The URL in the address bar is `docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/Cookie.html`. The page content is organized into several sections:

- Left Sidebar:** Contains links to "All Classes" and "Packages". Under "Packages", there is a list of javax.annotation sub-packages: annotation, annotation_security, annotation_servlet, decorator, ejb, ejb_embeddable, ejb_spi, and ejb_el.
- Constructor Summary:** This section is highlighted with a green oval. It contains one entry:

```
Cookie(java.lang.String name, java.lang.String value)
Constructs a cookie with the specified name and value.
```
- Method Summary:** This section lists various methods of the `Cookie` class, each with its return type, name, and a brief description. The methods are:

Return Type	Method Name	Description
java.lang.Object	<code>clone()</code>	Overrides the standard <code>java.lang.Object.clone</code> method to return a copy of this Cookie.
java.lang.String	<code>getComment()</code>	Returns the comment describing the purpose of this cookie, or <code>null</code> if the cookie has no comment.
java.lang.String	<code>getDomain()</code>	Gets the domain name of this Cookie.
int	<code>getMaxAge()</code>	Gets the maximum age in seconds of this Cookie.
java.lang.String	<code>getName()</code>	Returns the name of the cookie.
java.lang.String	<code>getPath()</code>	Returns the path on the server to which the browser returns this cookie.
boolean	<code>getSecure()</code>	Returns <code>true</code> if the browser is sending cookies only over a secure protocol, or <code>false</code> if the browser can send cookies using any protocol.
java.lang.String	<code>getValue()</code>	Gets the current value of this Cookie.
int	<code>getVersion()</code>	Returns the version of the protocol this cookie complies with.
boolean	<code>isHttpOnly()</code>	Checks whether this Cookie has been marked as <code>HttpOnly</code> .
void	<code>setComment(java.lang.String purpose)</code>	Specifies a comment that describes a cookie's purpose.
void	<code>setDomain(java.lang.String domain)</code>	Specifies the domain within which this cookie should be presented.
void	<code>setHttpOnly(boolean isHttpOnly)</code>	Marks or unmarks this Cookie as <code>HttpOnly</code> .
- Bottom Navigation:** Includes standard browser controls like back, forward, and search, along with McAfee security branding.

Cookie API

The screenshot shows the Java EE 6 API documentation for the `Cookie` class in Mozilla Firefox. The page lists various methods of the `Cookie` class, each with a brief description. A green oval highlights several of these methods:

- `getComment()`: Returns the comment describing the purpose of this cookie, or `null` if the cookie has no comment.
- `getDomain()`: Gets the domain name of this Cookie.
- `getMaxAge()`: Gets the maximum age in seconds of this Cookie.
- `getName()`: Returns the name of the cookie.
- `getPath()`: Returns the path on the server to which the browser returns this cookie.
- `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.
- `getValue()`: Gets the current value of this Cookie.
- `getVersion()`: Returns the version of the protocol this cookie complies with.
- `isHttpOnly()`: Checks whether this Cookie has been marked as `HttpOnly`.
- `setComment(java.lang.String purpose)`: Specifies a comment that describes a cookie's purpose.
- `setDomain(java.lang.String domain)`: Specifies the domain within which this cookie should be presented.
- `setHttpOnly(boolean isHttpOnly)`: Marks or unmarks this Cookie as `HttpOnly`.
- `setMaxAge(int expiry)`: Sets the maximum age in seconds for this Cookie.
- `setPath(java.lang.String uri)`: Specifies a path for the cookie to which the client should return the cookie.
- `setSecure(boolean flag)`: Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.
- `setValue(java.lang.String newValue)`: Assigns a new value to this Cookie.
- `setVersion(int v)`: Sets the version of the cookie protocol that this Cookie complies with.

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.

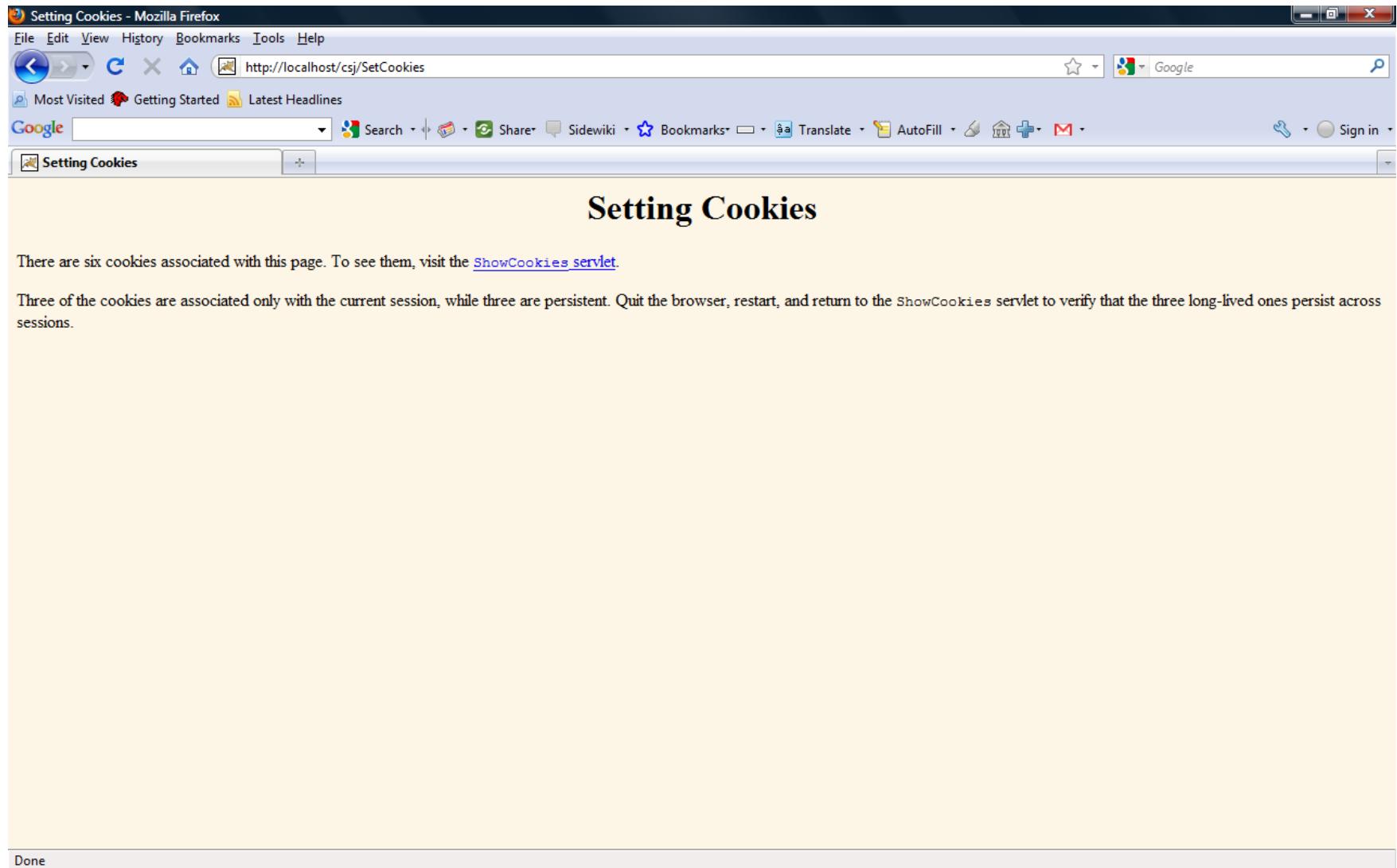
Cookie API (cont.)

- 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)`

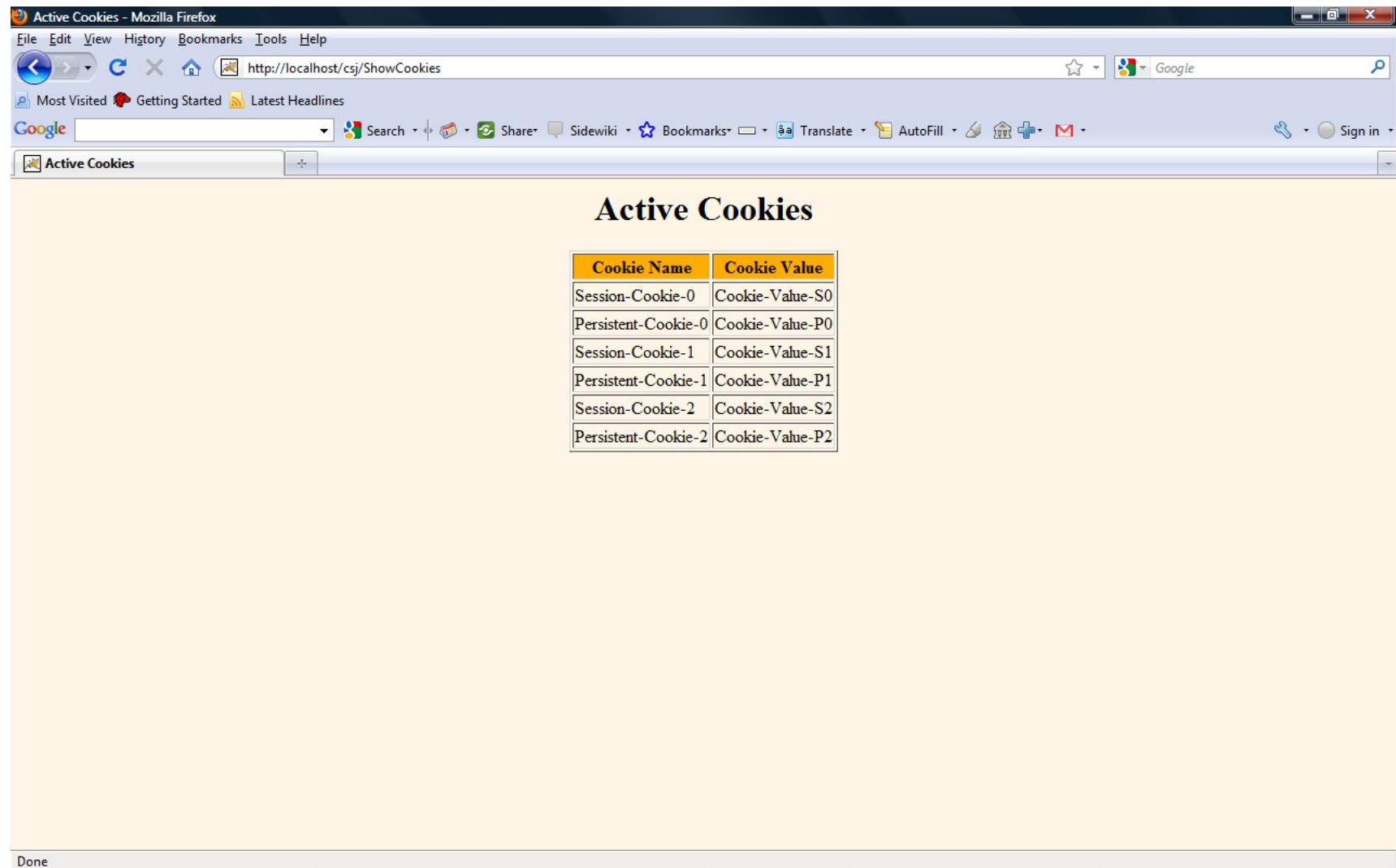
How to run and test session cookie and persistent cookie?

- 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?



A screenshot of a Mozilla Firefox browser window titled "Active Cookies - Mozilla Firefox". The address bar shows the URL <http://localhost/csj>ShowCookies>. The main content area displays a table titled "Active Cookies" with the following data:

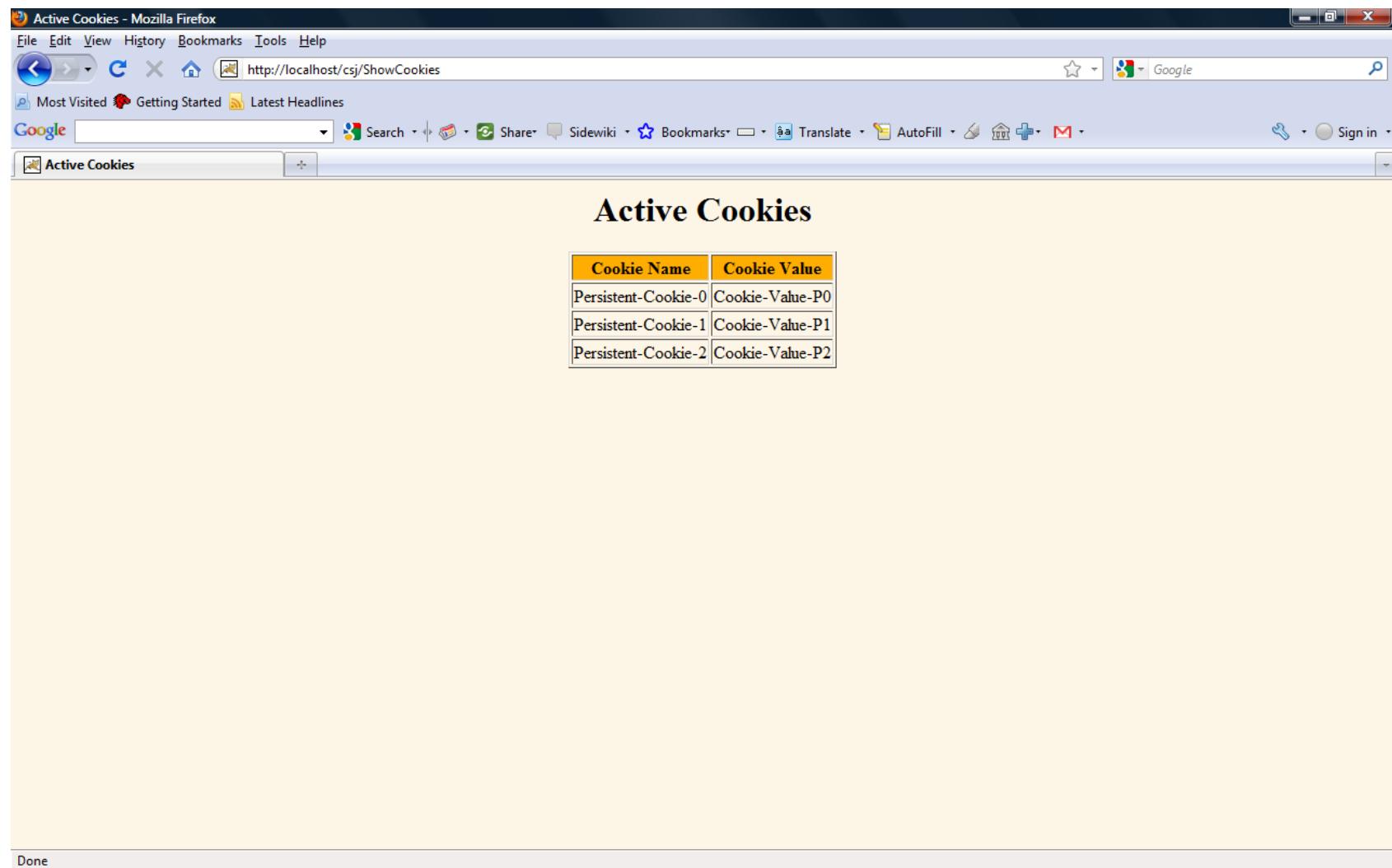
Cookie Name	Cookie Value
Session-Cookie-0	Cookie-Value-S0
Persistent-Cookie-0	Cookie-Value-P0
Session-Cookie-1	Cookie-Value-S1
Persistent-Cookie-1	Cookie-Value-P1
Session-Cookie-2	Cookie-Value-S2
Persistent-Cookie-2	Cookie-Value-P2

The Firefox interface includes standard toolbar icons, a search bar, and a bookmarks bar. The status bar at the bottom left shows the word "Done".

How to run and test session cookie and persistent cookie?

- Now, kill the browser and start it again by accessing directly the **ShowCookies** without 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?



A screenshot of a Mozilla Firefox browser window titled "Active Cookies - Mozilla Firefox". The address bar shows the URL "http://localhost/csj>ShowCookies". The main content area displays a table titled "Active Cookies" with three rows of data. The table has two columns: "Cookie Name" and "Cookie Value". The data is as follows:

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

At the bottom left of the browser window, there is a "Done" button.

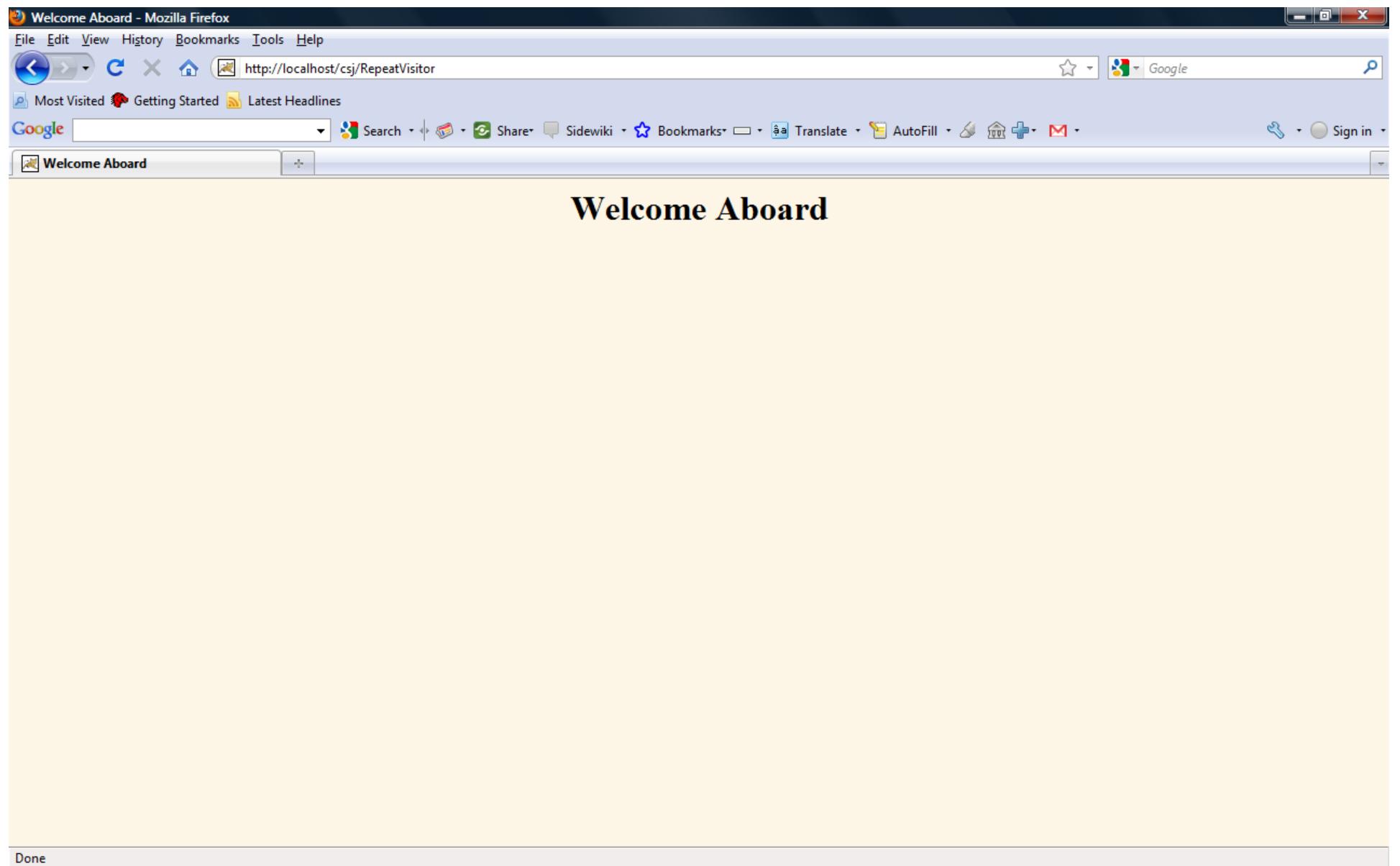
SetCookies and ShowCookies

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

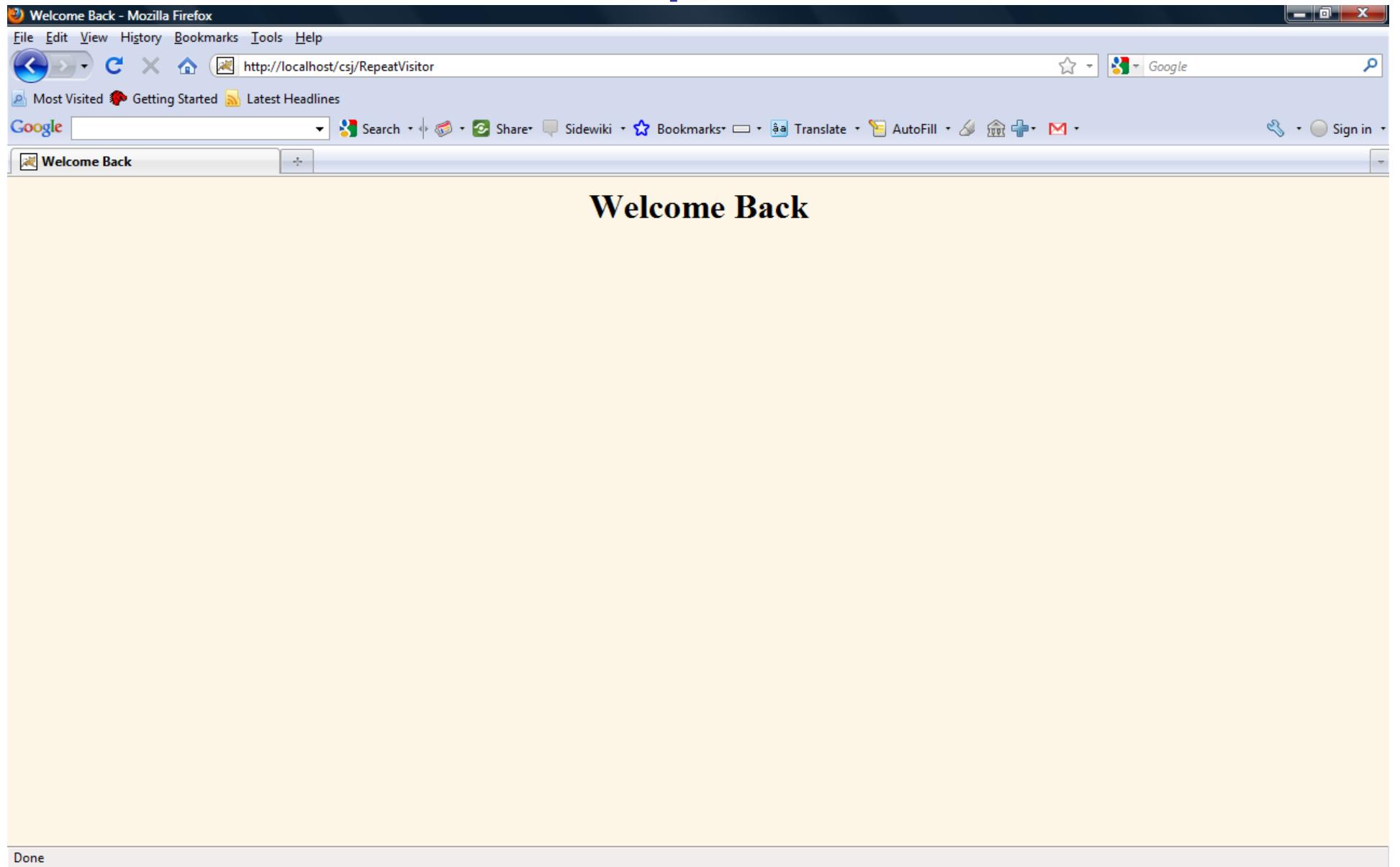
Cookie for Repeat Visitor

- 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



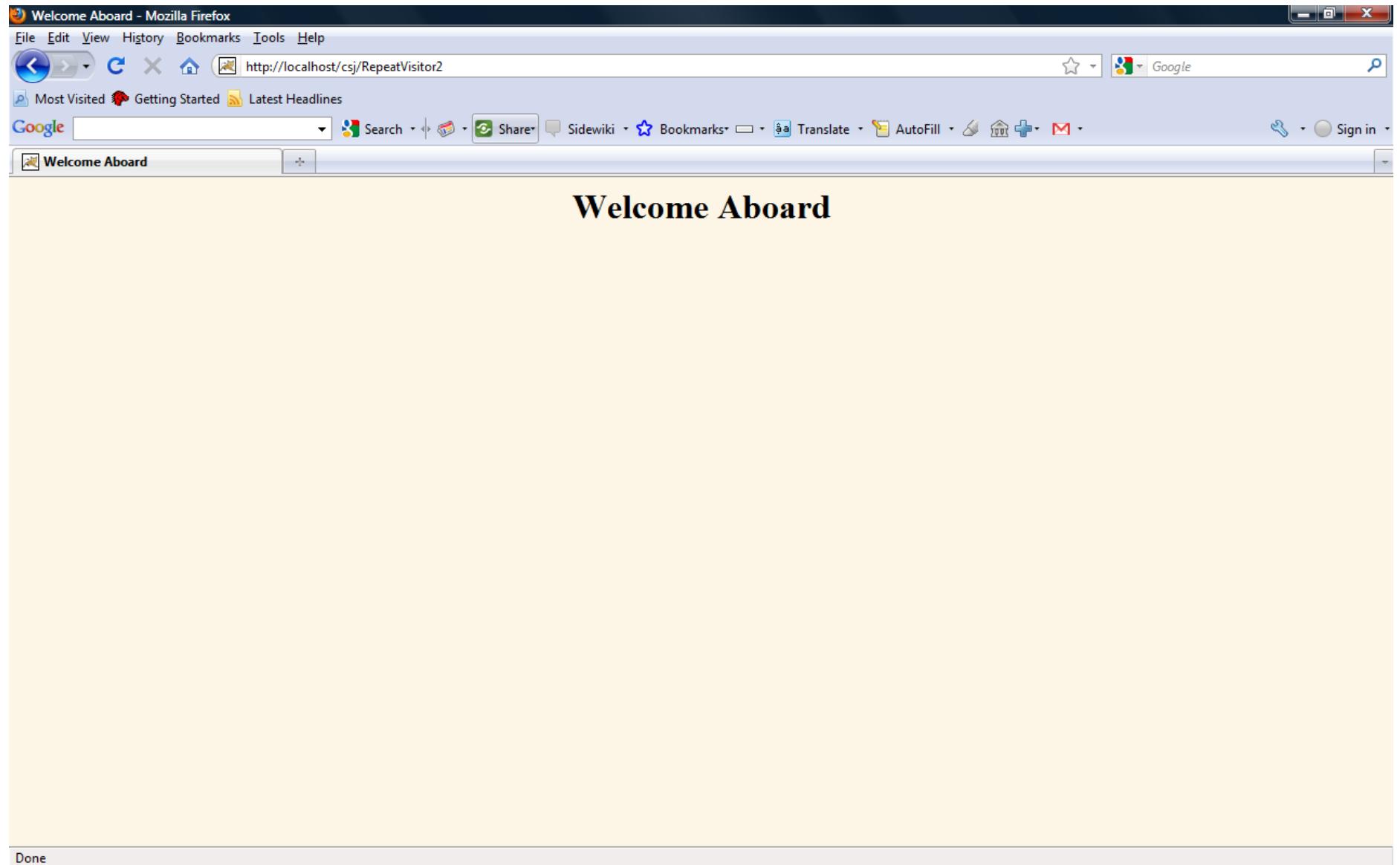
RepeatVisitor

- Lets review the source code for
 - RepeatVisitor.java

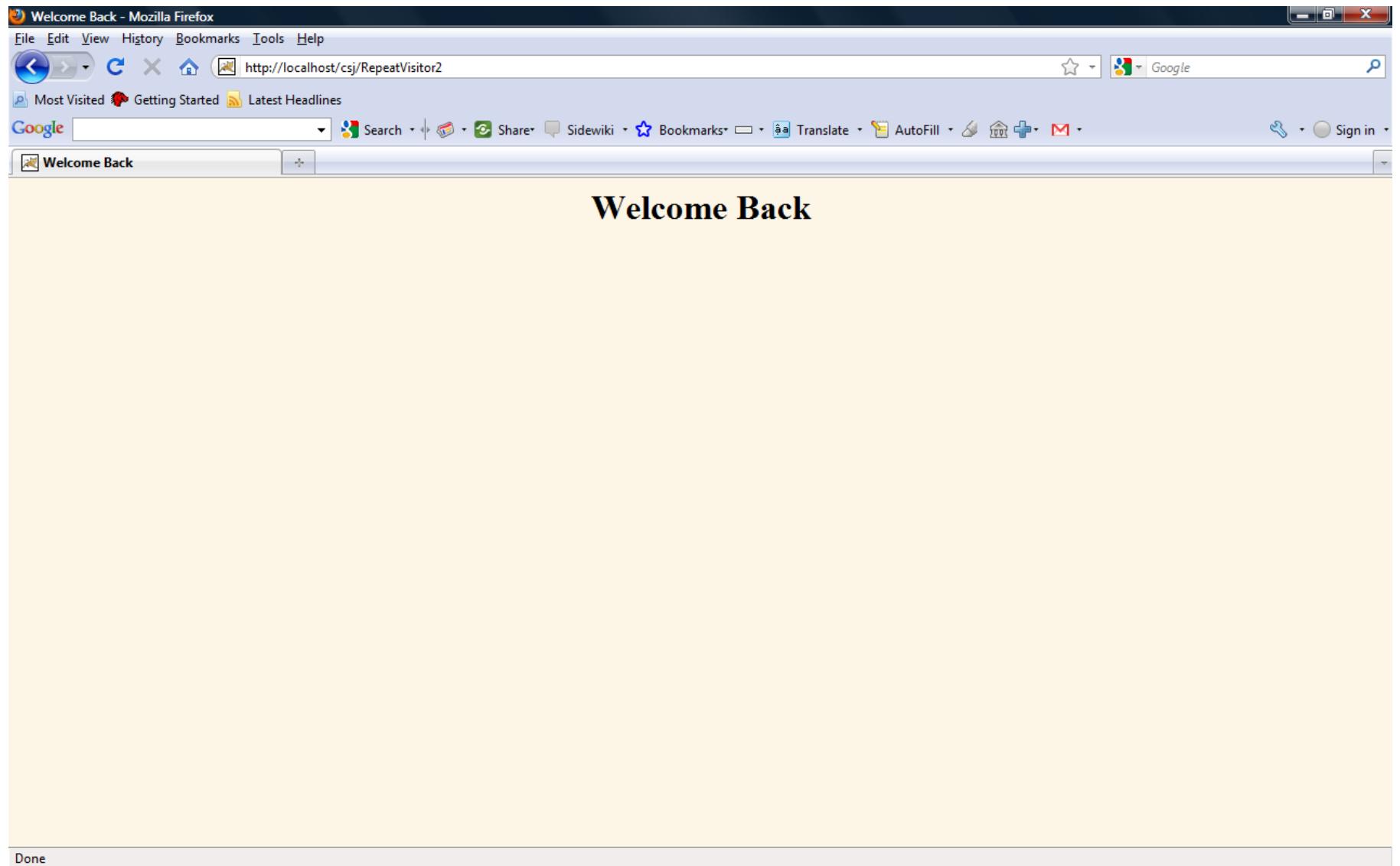
Repeat Visitor 2

- 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



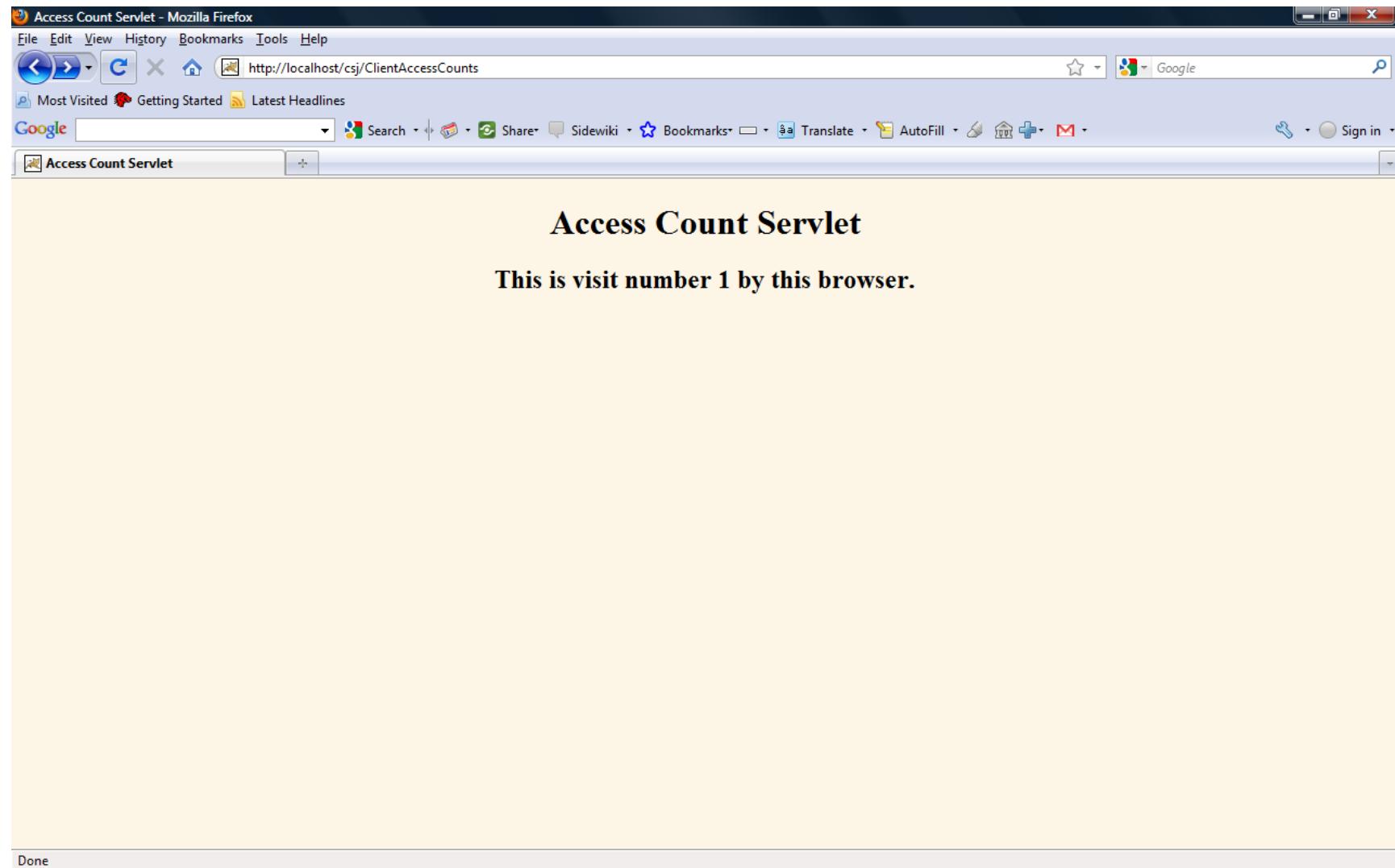
RepeatVisitor2

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

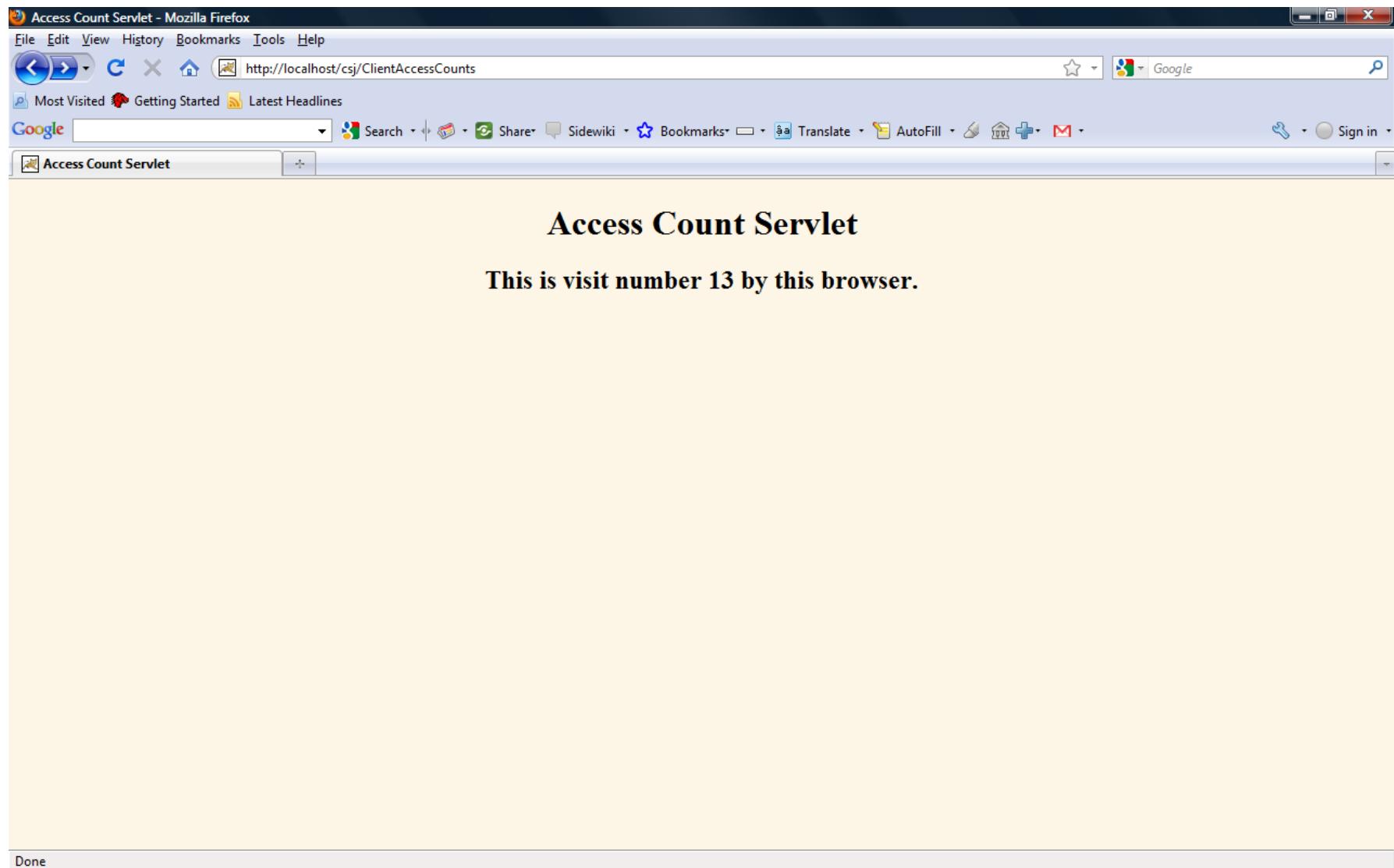
How to repeatedly change cookie values?

- 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

The screenshot shows a Mozilla Firefox browser window with the title "HttpSession (Java EE 6) - Mozilla Firefox". The address bar contains the URL "docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/HttpSession.html". The page content is the HttpSession API documentation.

Overview Package Class Tree Deprecated Index Help

javaservlet.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

The screenshot shows a Mozilla Firefox browser window displaying the Java EE 6 HttpSession API documentation on docs.oracle.com. The title bar reads "HttpSession (Java EE 6) - Mozilla Firefox". The left sidebar contains navigation links for "All Classes" and "Packages", listing various Java annotations and interfaces. The main content area is titled "Method Summary" and lists the following methods:

Type	Method Name	Description
java.lang.Object	getAttribute(java.lang.String name)	Returns the object bound with the specified name in this session, or <code>null</code> 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()	<i>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.</i>
java.lang.Object	getValue(java.lang.String name)	<i>Deprecated. As of Version 2.2, this method is replaced by <code>getAttribute(java.lang.String)</code>.</i>
java.lang.String[]	getValueNames()	<i>Deprecated. As of Version 2.2, this method is replaced by <code>getAttributeNames()</code>.</i>
void	invalidate()	Invalidates this session then unbinds any objects bound to it.
boolean	isNew()	Returns <code>true</code> 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)	<i>Deprecated. As of Version 2.2, this method is replaced by <code>setAttribute(java.lang.String, java.lang.Object)</code>.</i>
void	removeAttribute(java.lang.String name)	Removes the object bound with the specified name from this session.
void	removeValue(java.lang.String name)	<i>Deprecated. As of Version 2.2, this method is replaced by <code>removeAttribute(java.lang.String)</code>.</i>

Session API

The screenshot shows a Mozilla Firefox browser window displaying the HttpSession API documentation from the Oracle Java EE 6 API index. The URL in the address bar is docs.oracle.com/javaee/6/api/index.html?javax/servlet/http/HttpSession.html. The page lists various methods of the HttpSession interface, each with its return type, name, and a brief description.

		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() <i>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.</i>
	java.lang.Object	getValue(java.lang.String name) <i>Deprecated. As of Version 2.2, this method is replaced by getAttribute(java.lang.String).</i>
	java.lang.String[]	getValueNames() <i>Deprecated. As of Version 2.2, this method is replaced by getAttributeNames().</i>
	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) <i>Deprecated. As of Version 2.2, this method is replaced by setAttribute(java.lang.String, java.lang.Object).</i>
	void	removeAttribute(java.lang.String name) Removes the object bound with the specified name from this session.
	void	removeValue(java.lang.String name) <i>Deprecated. As of Version 2.2, this method is replaced by removeAttribute(java.lang.String).</i>
	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.

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>

A screenshot of a Mozilla Firefox browser window titled "Session Tracking Example - Mozilla Firefox". The address bar shows the URL "http://localhost/csj>ShowSession". The page content displays a welcome message "Welcome, Newcomer" and a table titled "Information on Your Session:".

Welcome, Newcomer

Information on Your Session:

Info Type	Value
ID	CFBB771944FD0E98ED7447ED72CE91A7
Creation Time	Tue Oct 12 20:27:06 CDT 2010
Time of Last Access	Tue Oct 12 20:27:06 CDT 2010
Number of Previous Accesses	0

A green callout bubble with the text "1st Visit" points to the "Number of Previous Accesses" value of 0.

ShowSession.java

<http://localhost/csj>ShowSession>

The screenshot shows a Mozilla Firefox browser window titled "Session Tracking Example - Mozilla Firefox". The address bar contains the URL "http://localhost/csj>ShowSession". The main content area displays a "Welcome Back" message and a table titled "Information on Your Session:". The table has two columns: "Info Type" and "Value". The rows show session details: ID (CFBB771944FD0E98ED7447ED72CE91A7), Creation Time (Tue Oct 12 20:27:06 CDT 2010), Time of Last Access (Tue Oct 12 20:28:19 CDT 2010), and Number of Previous Accesses (12). A large green callout bubble points from the number "12" in the table to the text "12th Visit" located in the bottom right corner of the slide.

Info Type	Value
ID	CFBB771944FD0E98ED7447ED72CE91A7
Creation Time	Tue Oct 12 20:27:06 CDT 2010
Time of Last Access	Tue Oct 12 20:28:19 CDT 2010
Number of Previous Accesses	12

12th Visit

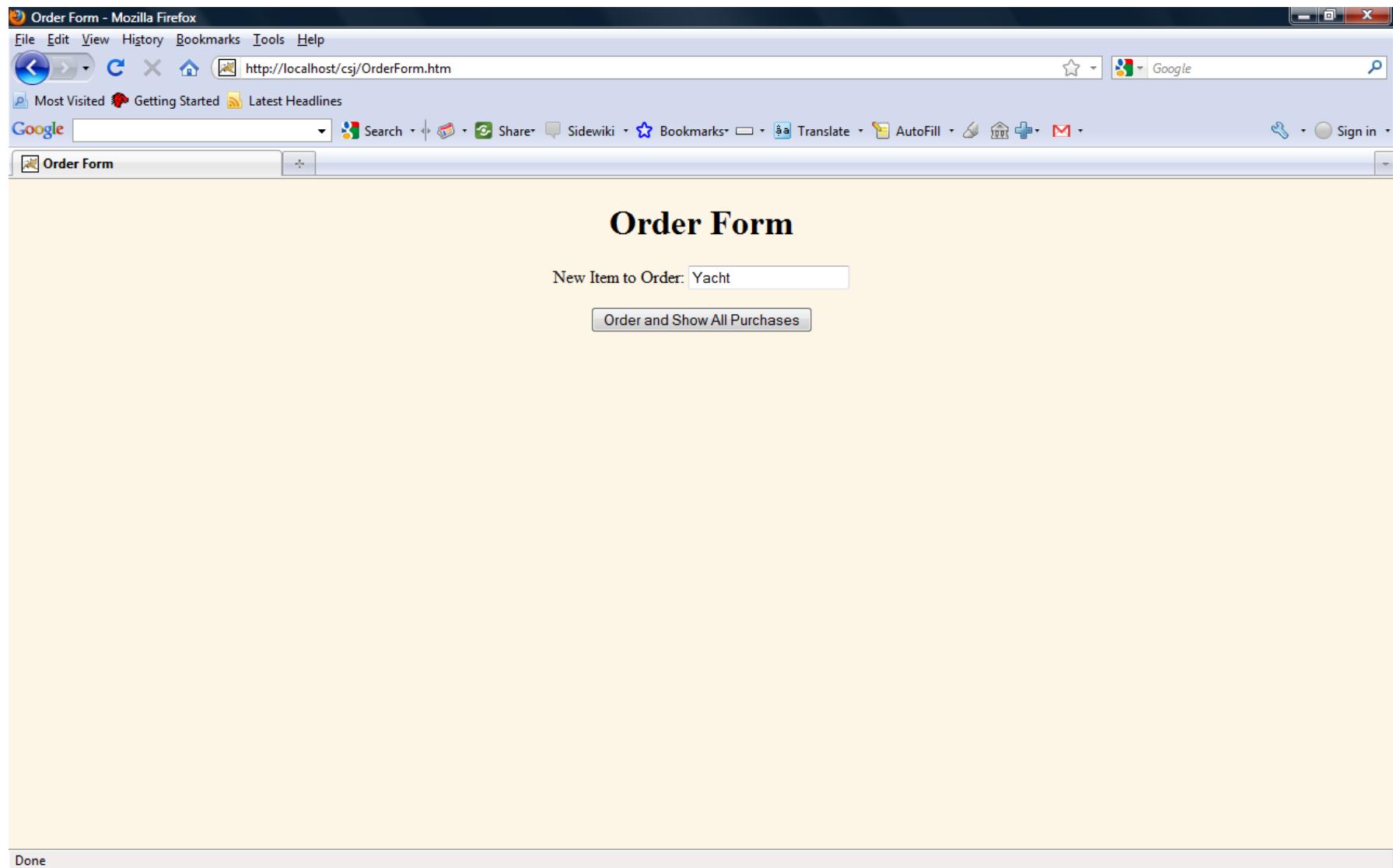
Accumulating a List of User Data

- 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.

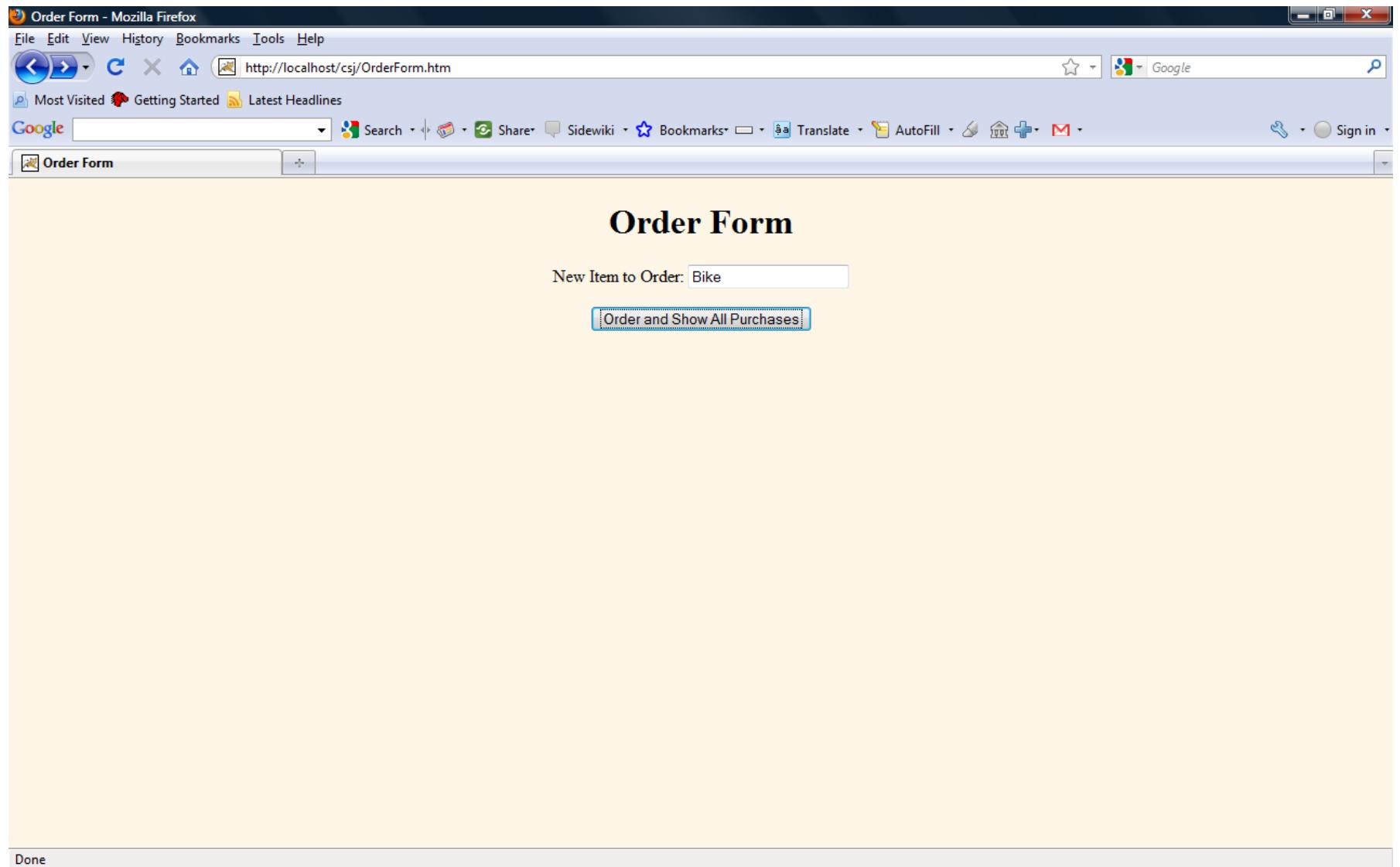
Accumulating a List of User Data

- 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.

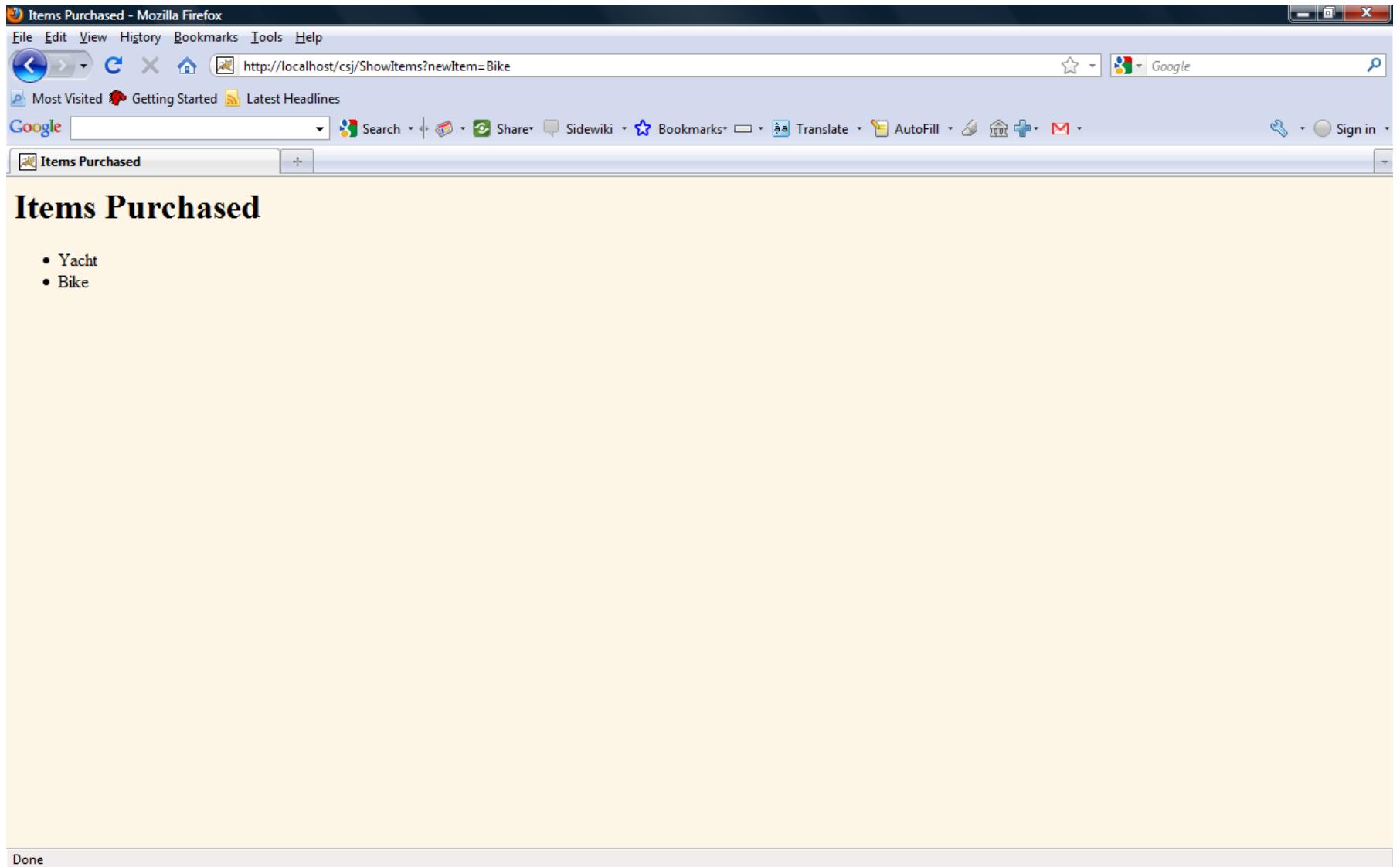
Accumulating a List of User Data



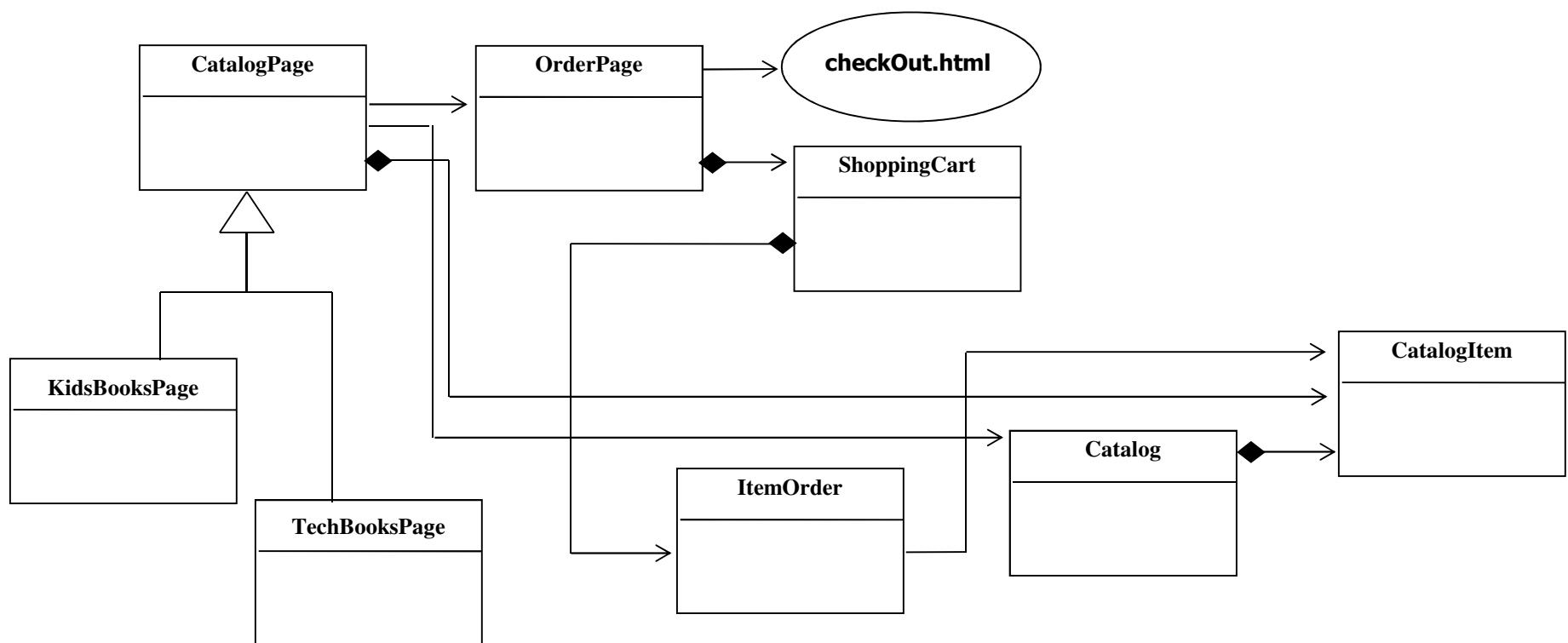
Accumulating a List of User Data



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 screenshot shows a Mozilla Firefox browser window displaying the Java EE 5 SDK API documentation for the `encodeURL` method of `HttpServletResponse`. The URL in the address bar is `docs.oracle.com/javaee/5/api/index.html?javax/servlet/http/HttpServletResponse.html`.

The left sidebar contains a navigation tree with categories like `javax.resource.spi.security`, `javax.resource.spi.work`, `javax.security.jacc`, `javax.servlet`, `javax.servlet.http`, `javax.servlet.jsp`, `javax.servlet.jsp.el`, `javax.servlet.jsp.tagext`, `javax.servlet.http`, `Interfaces`, `HttpServletRequest`, `HttpServletResponse`, `HttpSession`, `HttpSessionActivationListener`, `HttpSessionAttributeListener`, `HttpSessionBindingListener`, `HttpSessionContext`, and `HttpSessionListener`. Below these are `Classes` sections for `Cookie`, `HttpServlet`, `HttpServletRequestWrapper`, `HttpServletResponseWrapper`, `HttpSessionBindingEvent`, `HttpSessionEvent`, and `HttpUtils`.

The main content area shows the `encodeURL` method documentation. It includes parameters (`name` - the header name), returns (`true` if the named response header has already been set; `false` otherwise), and a detailed description: "Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged. The implementation of this method includes the logic to determine whether the session ID needs to be encoded in the URL. For example, if the browser supports cookies, or session tracking is turned off, URL encoding is unnecessary." A callout box highlights a note: "For robust session tracking, all URLs emitted by a servlet should be run through this method. Otherwise, URL rewriting cannot be used with browsers which do not support cookies." The `Parameters:` section specifies `url` - the url to be encoded, and the `Returns:` section specifies the encoded URL if encoding is needed; the unchanged URL otherwise.

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

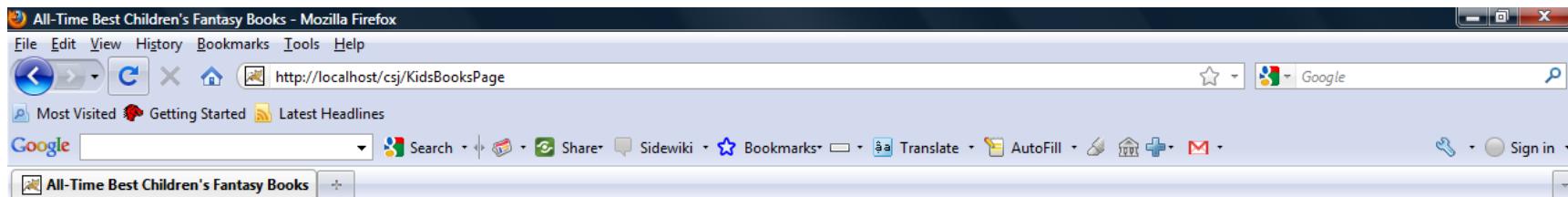
- [TechBooksPage.java](#). A specialization of the [CatalogPage](#) servlet that displays a page selling two famous computer books. Orders are sent to the [OrderPage](#) servlet.
- [OrderPage.java](#). Servlet that records new item orders (if any) and then displays all items in the shopping cart.
- [Checkout.html](#). 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>



A screenshot of a Mozilla Firefox browser window. The title bar reads "All-Time Best Children's Fantasy Books - Mozilla Firefox". The address bar shows the URL "http://localhost/csj/KidsBooksPage". The toolbar includes standard buttons for back, forward, stop, and search, along with links for "Most Visited", "Getting Started", and "Latest Headlines". The menu bar has options like File, Edit, View, History, Bookmarks, Tools, and Help. The main content area displays the "All-Time Best Children's Fantasy Books" page.

All-Time Best Children's Fantasy Books

The Chronicles of Narnia by C.S. Lewis (\$19.95)

The classic children's adventure pitting Aslan the Great Lion and his followers against the White Witch and the forces of evil. Dragons, magicians, quests, and talking animals wound around a deep spiritual allegory. Series includes *The Magician's Nephew*, *The Lion, the Witch and the Wardrobe*, *The Horse and His Boy*, *Prince Caspian*, *The Voyage of the Dawn Treader*, *The Silver Chair*, and *The Last Battle*.

The Prydain Series by Lloyd Alexander (\$19.95)

Humble pig-keeper Taran joins mighty Lord Gwydion in his battle against Arawn the Lord of Annuvon. Joined by his loyal friends the beautiful princess Eilonwy, wannabe bard Fflewddur Fflam, and fury half-man Gurgi, Taran discovers courage, nobility, and other values along the way. Series includes *The Book of Three*, *The Black Cauldron*, *The Castle of Llyr*, *Taran Wanderer*, and *The High King*.

The Harry Potter Series by J.K. Rowling (\$59.95)

The first five of the popular stories about wizard-in-training Harry Potter topped both the adult and children's best-seller lists. Series includes *Harry Potter and the Sorcerer's Stone*, *Harry Potter and the Chamber of Secrets*, *Harry Potter and the Prisoner of Azkaban*, *Harry Potter and the Goblet of Fire*, and *Harry Potter and the Order of the Phoenix*.

Done

The Shopping Cart

<http://localhost/csj/TechBooksPage>

All-Time Best Computer Books - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/csj/TechBooksPage

Most Visited Getting Started Latest Headlines

Google Search Share Sidewiki Bookmarks Translate AutoFill

All-Time Best Computer Books

All-Time Best Computer Books

Core Servlets and JavaServer Pages 2nd Edition (Volume 1) by Marty Hall and Larry Brown (\$39.95)

The definitive reference on servlets and JSP from Prentice Hall and Sun Microsystems Press.

Nominated for the Nobel Prize in Literature.

Core Web Programming, 2nd Edition by Marty Hall and Larry Brown (\$49.99)

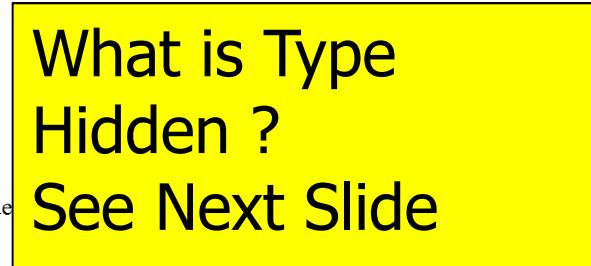
One stop shopping for the Web programmer. Topics include

- Thorough coverage of Java 2; including Threads, Networking, Swing, Java 2D, RMI, JDBC, and Collections
- A fast introduction to HTML 4.01, including frames, style sheets, and layers.
- A fast introduction to HTTP 1.1, servlets, and JavaServer Pages.
- A quick overview of JavaScript 1.2

Done

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>");
    }
}
```

Ln 106, Col 3

The Shopping Cart

Input Type = Hidden

HTML input type Attribute - Mozilla Firefox

File Edit View History Bookmarks Tools Help

www.w3schools.com/tags/att_input_type_hidden.asp

Input type: hidden

Example

Define a hidden field (not visible to a user). A hidden field often store a default value, or can have its value changed by a JavaScript:

```
<input type="hidden" name="country" value="Norway">
```

Try it yourself »

Input type: image

Example

Define an image as a submit button:

```
<input type="image" src="img_submit.gif" alt="Submit">
```

Try it yourself »

Input type: month

hidden

Highlight All Match

Secure Search McAfee McAfee

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 screenshot shows a Mozilla Firefox browser window titled "Status of Your Order". The address bar displays the URL: "http://localhost/csj/OrderPage?itemID=hall001&numItems=2". The main content area is titled "Status of Your Order" and contains a table of four items. Each item row includes an "Update Order" button and a numeric input field for quantity.

Item ID	Description	Unit Cost	Number	Total Cost
lewis001	<i>The Chronicles of Narnia</i> by C.S. Lewis	\$19.95	1	<input type="button" value="Update Order"/> \$19.95
rowling001	<i>The Harry Potter Series</i> by J.K. Rowling	\$59.95	1	<input type="button" value="Update Order"/> \$59.95
hall001	<i>Core Servlets and JavaServer Pages 2nd Edition</i> (Volume 1) by Marty Hall and Larry Brown	\$39.95	2	<input type="button" value="Update Order"/> \$79.90
hall002	<i>Core Web Programming, 2nd Edition</i> by Marty Hall and Larry Brown	\$49.99	1	<input type="button" value="Update Order"/> \$49.99

Done

The Shopping Cart

<http://localhost/csj/Checkout.htm>

A screenshot of a Mozilla Firefox browser window titled "Checking Out - Mozilla Firefox". The address bar shows the URL <http://localhost/csj/Checkout.htm>. The page content is centered with the heading "Checking Out" in a large, bold, black serif font. Below the heading, a message reads: "We are sorry, but our electronic credit-card-processing system is currently out of order. Please send a check to:" followed by the contact information for Marty Hall. At the bottom of the page, another message states: "Since we have not yet calculated shipping charges, please sign the check but do not fill in the amount. We will generously do that for you." The browser interface includes standard toolbar icons, a search bar, and a sidebar with links like "Most Visited", "Getting Started", and "Latest Headlines".

Checking Out

We are sorry, but our electronic credit-card-processing system is currently out of order. Please send a check to:

Marty Hall
coreservlets.com, Inc.
6 MeadowSweet Ct., Suite B1
Reisterstown, MD 21136-6020
410-429-5535
hall@coreservlets.com

Since we have not yet calculated shipping charges, please sign the check but do not fill in the amount. We will generously do that for you.

Done

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

```
Command Prompt
c:\csj\S4\SerializedDataStore>javac *.java
Note: GameSpeedSerializedDataStore.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

c:\csj\S4\SerializedDataStore>
c:\csj\S4\SerializedDataStore>dir
 Volume in drive C is Windows8_OS
 Volume Serial Number is 12AF-F3E4

Directory of c:\csj\S4\SerializedDataStore

04/18/2016  10:54 AM    <DIR>          .
04/18/2016  10:54 AM    <DIR>          ..
04/18/2016  10:54 AM           1,355 Accessory.class
04/15/2016  11:20 PM           1,301 Accessory.java
04/18/2016  10:54 AM           2,014 Console.class
04/15/2016  11:20 PM           1,656 Console.java
04/18/2016  10:47 AM           859 GameSpeedDataStore
04/18/2016  10:54 AM           4,279 GameSpeedSerializedDataStore.class
04/16/2016  12:49 AM           3,938 GameSpeedSerializedDataStore.java
04/18/2016  10:54 AM           344 TestDrive.class
04/16/2016  12:42 AM           176 TestDrive.java
04/16/2016  12:42 AM           9 File(s)   15,922 bytes
                           2 Dir(s)  172,313,710,592 bytes free

c:\csj\S4\SerializedDataStore>java TestDrive
xboxone
    Name : Xbox One
    Price : 399.0
    Accessories :
        xboxone_wc
            Name : Controller
            Price : 40.0
        xboxone_hs
            Name : Turtle Beach Headset
            Price : 51.0
xbox360
    Name : Xbox 360
    Price : 299.0
    Accessories :
        xbox360_sw
            Name : Speeding Wheel
            Price : 42.0
        xbox360_wa
            Name : Wireless Adapter
            Price : 53.0

c:\csj\S4\SerializedDataStore>
```

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

```
Command Prompt
c:\csj\S4\SAX Parser>javac *.java
Note: SaxParser4GameSpeedXMLdatastore.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

c:\csj\S4\SAX Parser>
c:\csj\S4\SAX Parser>dir
'Volume in drive C is Windows8_OS
Volume Serial Number is 12AF-F3E4

Directory of c:\csj\S4\SAX Parser

04/18/2016  09:37 AM    <DIR>      .
04/18/2016  09:37 AM    <DIR>      ..
04/18/2016  10:01 AM      1,000 Console.class
04/18/2016  01:29 AM        732 Console.java
04/18/2016  12:31 AM      3,531 MySaxParser.class
04/18/2016  01:15 AM        773 ProductCatalog.xml
04/18/2016  10:01 AM      3,460 SaxParser4GameSpeedXMLdatastore.class
04/18/2016  09:17 AM      5,001 SaxParser4GameSpeedXMLdatastore.java
               6 File(s)   14,497 bytes
               2 Dir(s)  172,561,600,512 bytes free

c:\csj\S4\SAX Parser>
c:\csj\S4\SAX Parser>java SaxParser4GameSpeedXMLdataStore
console #001:
    retailer: GameSpeed
    name: XBOX1
    accessory: XBOX controller
    accessory: Turtle Beach Headset
console #002:
    retailer: GameSpeed
    name: XBOX360
    accessory: Speeding Wheel
    accessory: Wireless Adapter

c:\csj\S4\SAX Parser>
```