Session 20

Data Sharing
&
Cookies

1

Reading & Reference

- Reading
  - Shared scopes
    - Java EE 7 Tutorial - Section 17.3
- Reference
  - http state management
  www.ietf.org/rfc/rfc2965.txt
  - Cookies
  en.wikipedia.org/wiki/HTTP_cookie

2

Lecture Objectives

- Understand the mechanisms to share data on the server
- Know how to use server shared objects to store state information
- Understand how the Web Container uses cookies to store server data so that it is available to separate server requests
- Understand the differences among shared scopes
- Understand how the Web container uses threads to support simultaneous access to server resources

When Do You Need to Share Data?

- Among objects cooperating on an application
- Among separate accesses from a single user (e.g., shopping cart)
  - Usually on the same workstation and browser

Remember that a Cloud application usually involves multiple simultaneous users in which some data is shared and some data is kept private from other users' access

How Do You Share Information?

- Using private helper objects (e.g., JavaBeans)
- Using attributes of a shared scope
- Using a DB (or a serialized file)
- Invoking Web resources

MVC Architecture / JavaBeans

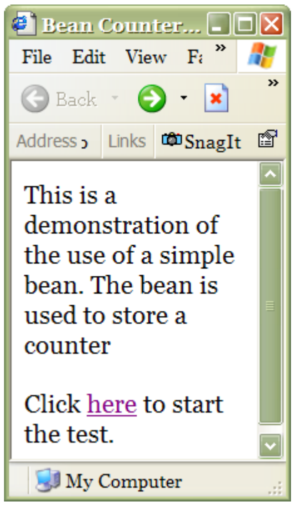
- Model, View, Controller
- Model manages data, logic, and rules of the application
- Java beans are in the model layer, and provide shared access to data
- A bean is an object that you can easily access within your server
- You can share a bean with other server objects
- You can get and set properties in the bean
- Bean data can be persistent

What Makes a Bean a Bean?

- A bean is an instance of a Java class that:
  - Must have a zero argument constructor
  - Should have no public instance variables
  - Should have (properly named) get and set methods for any instance variables that are to be accessed (setter argument type and getter return type must be identical)
  - Must support persistence (the bean is serializable)
- A bean usually supports events either by firing events when some properties change or listening for events (although we usually do not use this feature)

Example: Counter

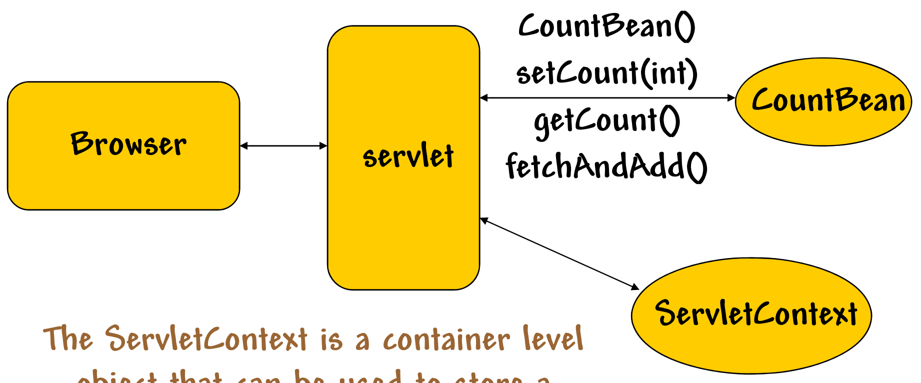
- The counter value is stored in a bean - along with methods to increment, get, and set the counter



This is a demonstration of the use of a simple bean. The bean is used to store a counter

Click here to start the test.

Bean Counter



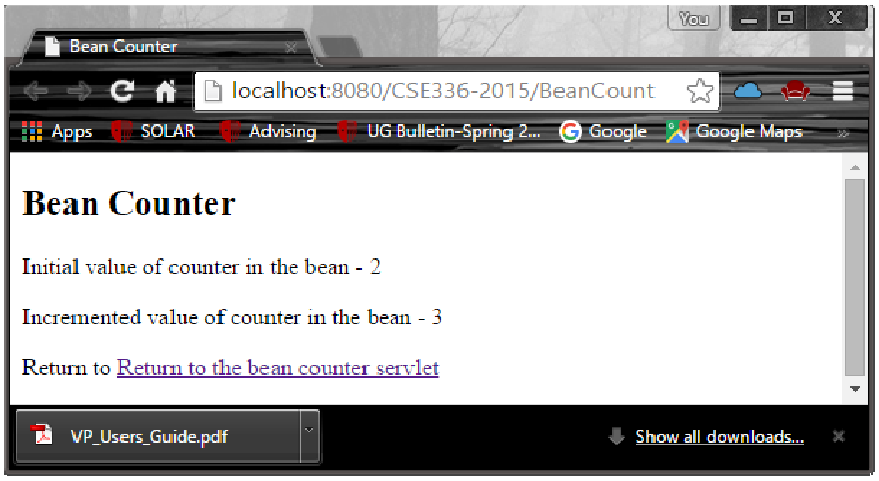
The ServletContext is a container level object that can be used to store a handle to the CountBean

Visibility

- Your Java Bean will have a life that extends beyond a single request/response
- The part of the server handling a request will need to have a handle to the bean
- You can make the bean visible by storing a handle to the bean in a shared context (e.g., Session)

We store a handle to the bean in a ServletContext object for now, but later we will store it in a Session object

BeanCounter Generated Page



Bean Counter

Initial value of counter in the bean - 2

Incremented value of counter in the bean - 3

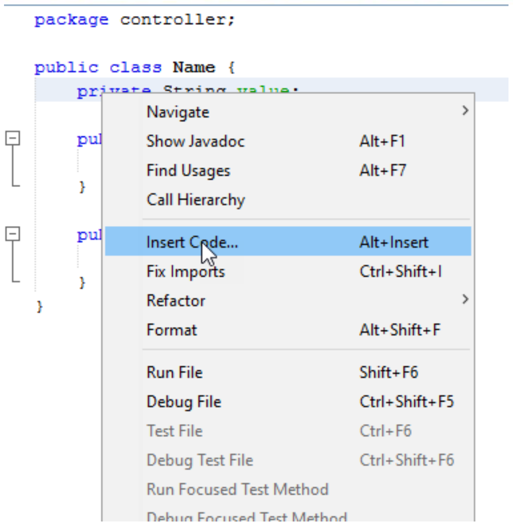
Return to Return to the bean counter servlet

© Robert Kelly, 2017-2018

11

NetBeans Help in Bean Generation

- Adding getter and setter methods can be tedious
- NetBeans can generate these automatically (almost)
  - Right click on the property and select Insert Code from the drop-down
  - Select getter and setter



```
package controller;

public class Name {
    private String value;
```

© Robert Kelly, 2017-2018

12

BeanCounter Servlet ...

```
@WebServlet(name = "BeanCount", urlPatterns = {"/BeanCount"})
public class BeanCount extends HttpServlet {
...
 protected void processRequest(HttpServletRequest request,
  HttpServletResponse response)
            throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-8");
  int bCount = 0;
  try (PrintWriter out = response.getWriter()) {
      out.println("<!DOCTYPE html>");
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Bean Counter</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h2>Bean Counter</h2>");
...
```

© Robert Kelly, 2017-2018

13

... BeanCounter Servlet

```
  ServletContext sc = this.getServletContext();
  CountBean b = (CountBean) sc.getAttribute("b");
  if (b == null) {
      b = new CountBean();
      sc.setAttribute("b", b);
  }
  bCount = b.fetchAndAdd();
  out.println("<p>Initial value of counter in the bean - ");
  out.println(bCount + "</p>");
  bCount = b.getCount();
  out.println("<p>Incremented value of counter in the bean - ");
  out.println(bCount + "</p>");
  out.println("<p>Return to");
  out.println(
  "<a href=\"http://localhost:8080/CSE336-2015/BeanCount\">");
  out.println("Return to the bean counter servlet</a>");
  out.println("</body>");     out.println("</html>");
  } }
```

The servlet gets the value of the counter from the CountBean bean

Shows that a bean can have methods other than getters and setters

© Robert Kelly, 2017-2018

14

CountBean

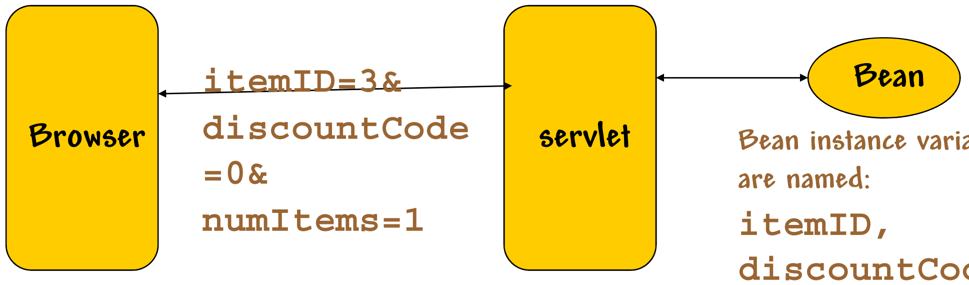
Notice the setter and getter naming conventions

```
public class CountBean implements Serializable {
  private int count = 0;

  public int getCount() {
    return (count);   }

  public int fetchAndAdd() {
    int temp=count;
    count++;
    return (temp);   }

  public void setCount(int newCount) {
    this.count = newCount;   }
}
```

Notice that fetchAndAdd returns the pre-incremented value of the counter

Notice that the bean is a standard Java class, but has the features of a bean (constructor, persistence, private instance variable, and properly named methods)

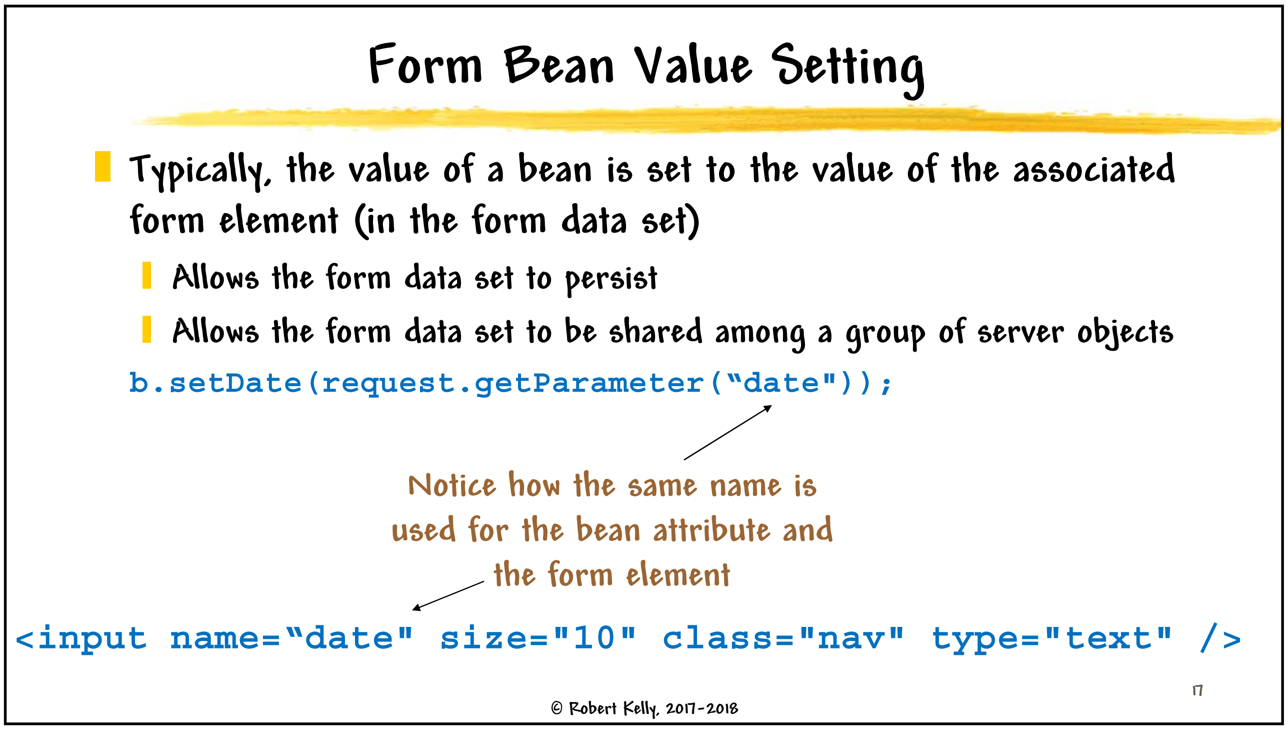
Setting all Bean Values From the Form



Bean instance variables are named: itemID, discountCode, and numItems

- A Web module (e.g., servlet) will usually read the form data set and set the values of the form in a bean so that they can be used by other Web modules

Frameworks will usually automate this part of the process



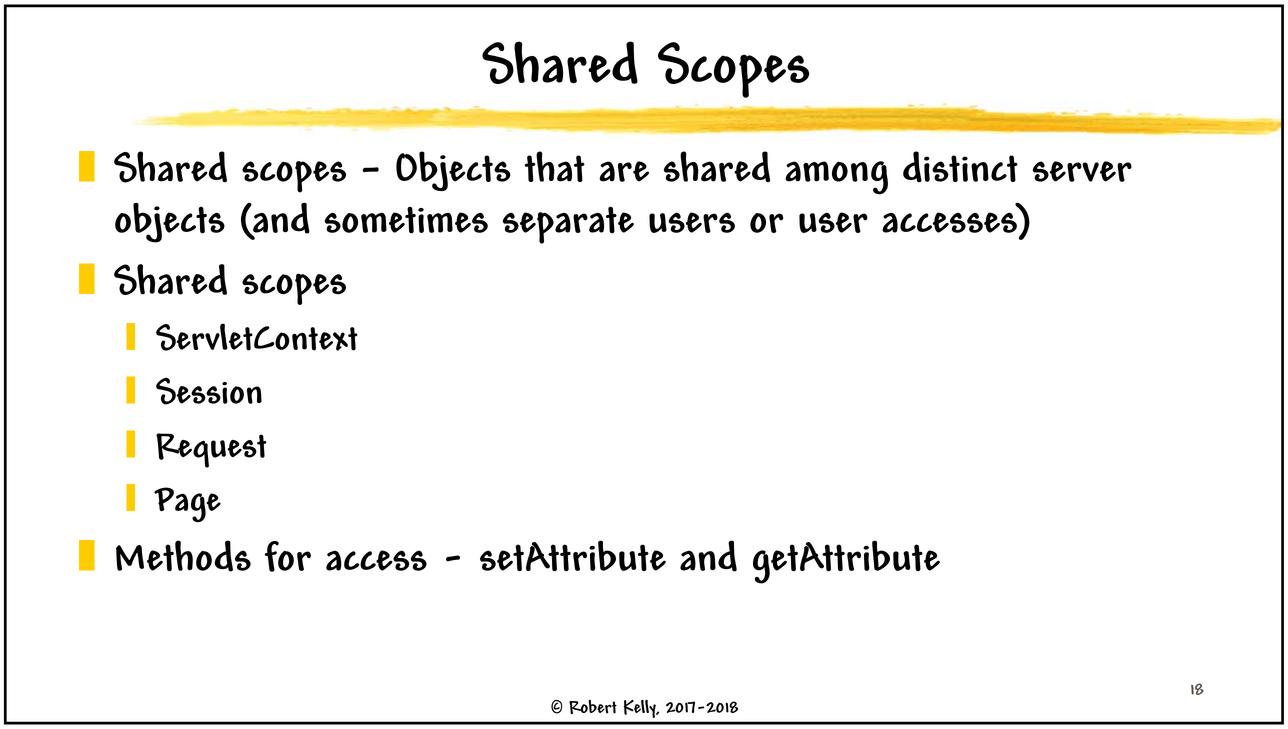
Form Bean Value Setting

- Typically, the value of a bean is set to the value of the associated form element (in the form data set)
  - Allows the form data set to persist
  - Allows the form data set to be shared among a group of server objects

```
b.setDate(request.getParameter("date"));
```

Notice how the same name is used for the bean attribute and the form element

```
<input name="date" size="10" class="nav" type="text" />
```



Shared Scopes

- Shared scopes - Objects that are shared among distinct server objects (and sometimes separate users or user accesses)
- Shared scopes
  - ServletContext
  - Session
  - Request
  - Page
- Methods for access - setAttribute and getAttribute

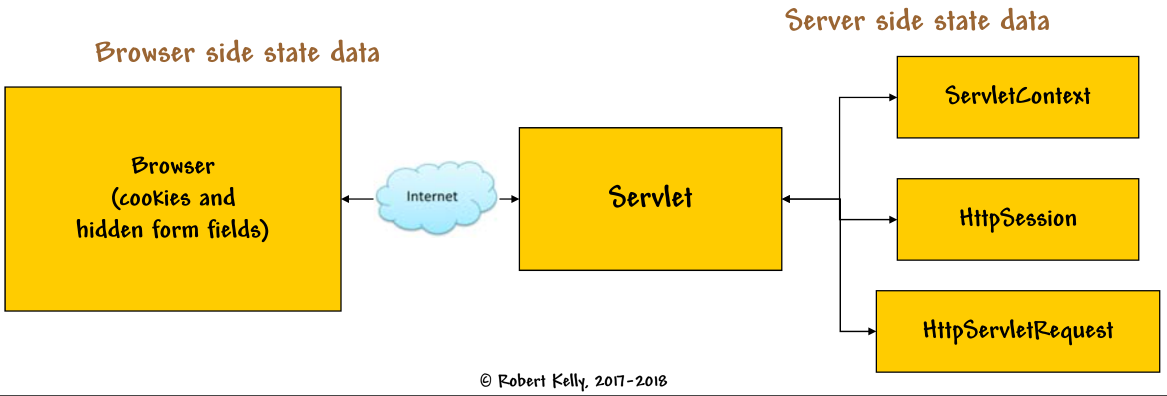
How Do The Shared Scopes Differ?

- Visibility
  - Different browsers
  - Different computers
- Lifetime
  - ServletContext - life of the container
  - Request - Duration of the request
  - Session - until timeout or destroy
  - Page - life of the servlet invocation

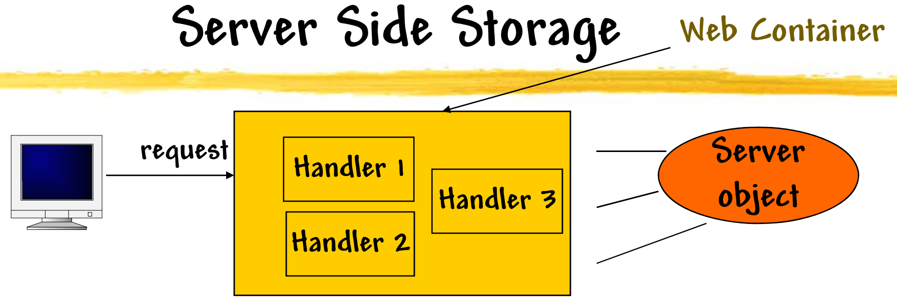
Visibility and lifetime define the scope of the object

Server Data Sharing

- The Http protocol is stateless, so your handler only responds to a single request
- Approaches: browser side and server side

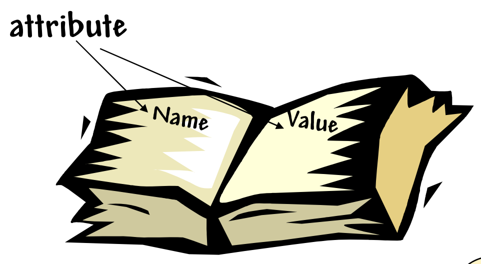


Server Side Storage



- Data stored on the server is usually contained in an object visible to the process handling the request
- To access the shared object, you need to obtain a reference (handle) to the object
- Objects for sharing
  - HttpServletRequest
  - ServletContext
  - Session
  - Other predefined and private objects

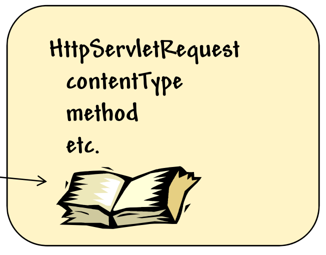
Shared Objects



The shared objects are referred to as "scopes"

The shared scopes are contained in other objects

For example, the request object contains the request scope



Why Do We Need a Session?

- The ServletContext object allows you to store data beyond a single request/response, but:
  - The life of the ServletContext object is too long for a user transaction
  - You probably want to limit the sharing to one user
- For example, data for a shopping application (a shopping cart) has a life that is only as long as the user is shopping - and you want the shopping cart to only be visible to servlet executions for that user

© Robert Kelly, 2017-2018

23

Session Object

- The Web container provides (and manages) session objects

Note that there are many session objects, but only one associated with a single computer/browser

- You can store information in a session object using name-value pairs, but the session object only exists for the "life of the session"
- A session usually corresponds to one user, who may visit a site many times where the interval between visits is "small"

How does the Web Container identify a user?

© Robert Kelly, 2017-2018

24

Session Tracking

- You get a handle to the session with a call to request.getSession()
- You access the session data through the session tracking parts of the Session API

Session
getAttribute(String)
setAttribute(String, Object)
getAttributeNames()
removeAttribute(String)

Returns an enumeration

Notice that the name/value pair is of type String/Object

Session Life Cycle API

- You can set the duration of a session (e.g., 20 minutes)
- Or you can invalidate the session when you are finished (e.g., when the user logs out)

Session
invalidate()
isNew()
getCreationTime()
getLastAccessedTime()
setMaxInactiveInterval(int)

Steps in Session Management

- Request a session object. This can be either:
  - A session object that was previously created and may contain data inserted by another servlet
  - A new session object when there is no existing session object matching this user
- Store information in the session object
- Invalidate the session - or allow the session to time out when maxInactiveInterval (time in seconds) is exceeded

```
setMaxInactiveInterval(int interval)
```

- Objects attached to the session can receive notification when they are unbound - through a listener interface

Obtaining a Session

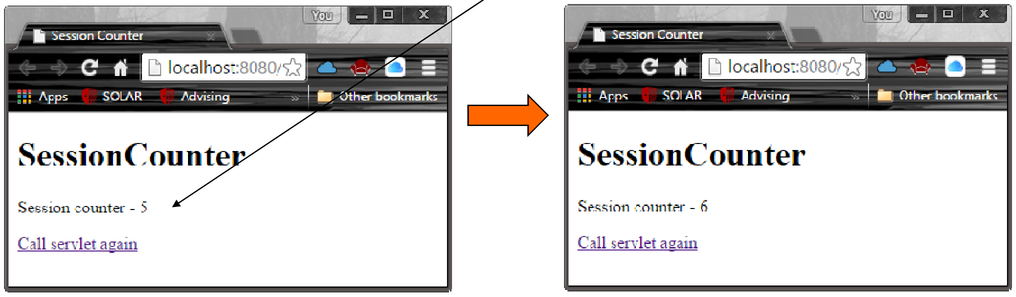
- Use the getSession method of HttpServletRequest
  - Returns an HttpSession object
- When the parameter of getSession is true or there is no parameter, a new session object is created, if it does not already exist
- getSession(false) will return null if there is no session

```
HttpSession session = request.getSession(true);
```

A good example of the factory design pattern

Example - Counter with Session

- The servlet uses the Session to store the access count



SessionCounter

Session counter - 5

Call servlet again

SessionCounter

Session counter - 6

Call servlet again

© Robert Kelly, 2017-2018

29

Session Counter

```
HttpSession hs = request.getSession(true);
Integer c = (Integer) hs.getAttribute("counter");
int hsCount;
if (c != null) {
  hsCount = c;
} else {
  hsCount = 0;
}
hsCount++;
hs.setAttribute("counter", new Integer(hsCount));
out.println("<p>Session counter - ");
out.println(hsCount + "</p>");
out.println("<p><a href='http://localhost:8080/CSE336-
  2017/SessionCounter'>Call servlet again</a></p>");
```

getAttribute returns an Object, which we cast to Integer

© Robert Kelly, 2017-2018

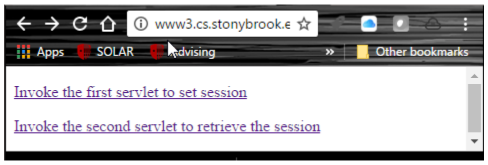
30

Are We on Track?

- Download TrackSessions
  - www3.cs.stonybrook.edu/~cse336/TrackSessions.htm
- Write 2 servlets to be invoked from the page
- Servlet 1
  - Instantiate a Date object
  - Store the Date object in the Session object
- Servlet 2
  - Access the Session object
  - Retrieve the Date object
  - Display the minutes component of the Date object

You will need to modify the html to access your servlets

You can use the deprecated methods of the Date object



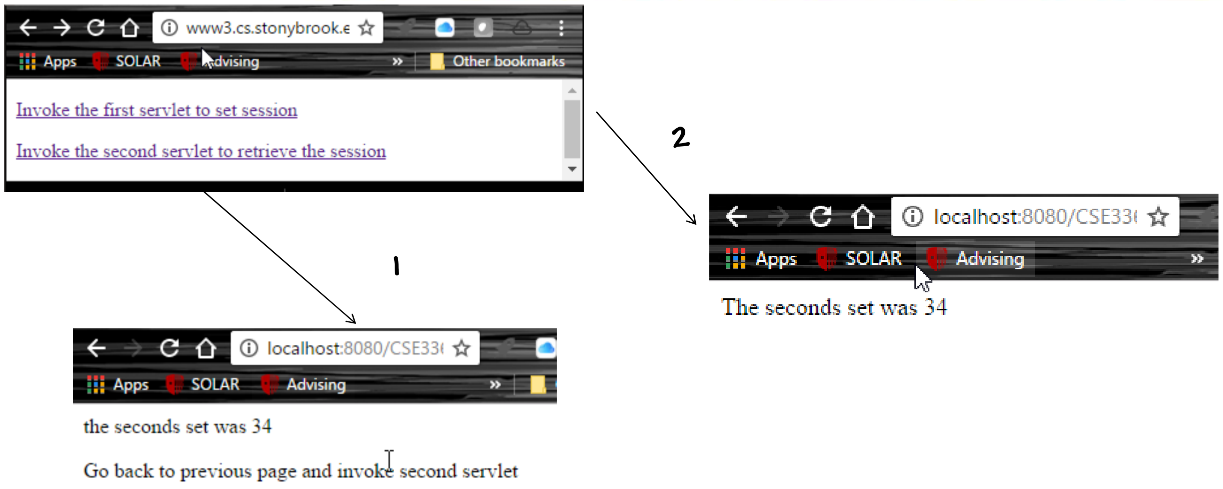
Invoke the first servlet to set session

Invoke the second servlet to retrieve the session

© Robert Kelly, 2017-2018

31

Are We on Track?



Invoke the first servlet to set session

Invoke the second servlet to retrieve the session

The seconds set was 34

the seconds set was 34

Go back to previous page and invoke second servlet

© Robert Kelly, 2017-2018

32

Were We on Track?

- Servlet 1

```
 HttpSession s = request.getSession(true);
 Date d = new Date();
 s.setAttribute("today", d);
...
 out.println("the seconds set was" + d.getSeconds());
 out.println("<p>Go back to previous page and invoke second servlet </p>");
```

- Servlet 2

```
HttpSession s2 = request.getSession();
Date d2 = (Date) s2.getAttribute("today");
...
out.println("<p>The seconds set was " + d2.getSeconds() + "</p>");
```

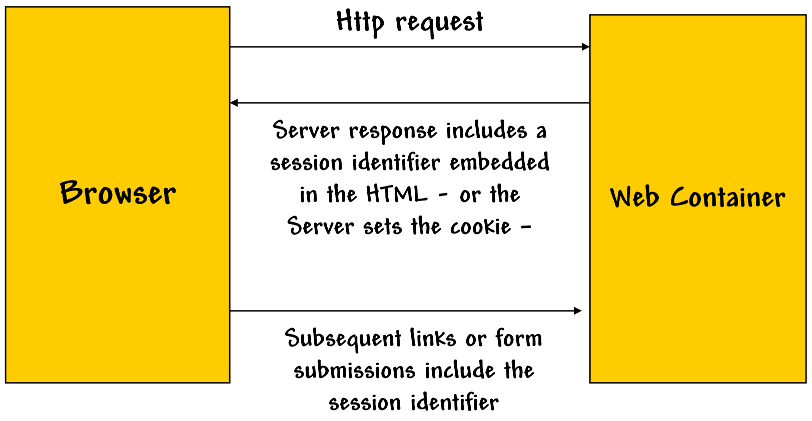
Server Session Recognition

- Session object is managed by the Web Container
- Implementation technique depends on the Web container implementation (and browser settings), and includes:
  - Hidden form fields
  - URL Rewriting
  - Cookies

Session data access and storage is usually implemented by the Web Container, but you should understand what is done

Used most often

Session Identifier Strategies

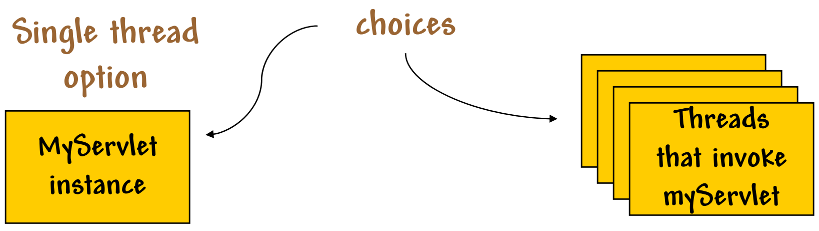

Browser
Http request
Web Container
Server response includes a session identifier embedded in the HTML - or the Server sets the cookie -
Subsequent links or form submissions include the session identifier

© Robert Kelly, 2017-2018

35

Servlet Execution

- How does the Web Container handle simultaneous requests to a servlet?

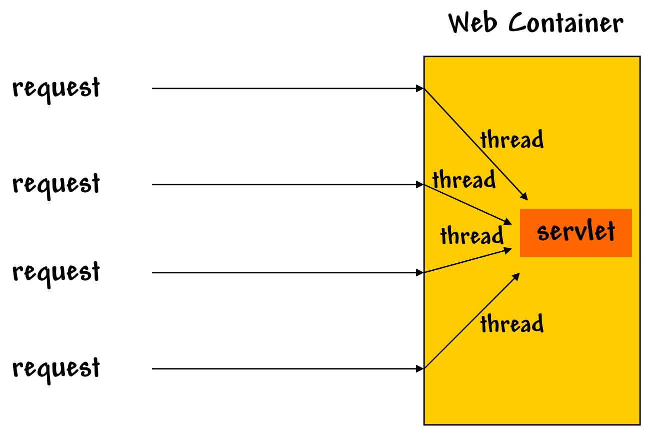


Is it safe for multiple threads to invoke myServlet?

© Robert Kelly, 2017-2018

36

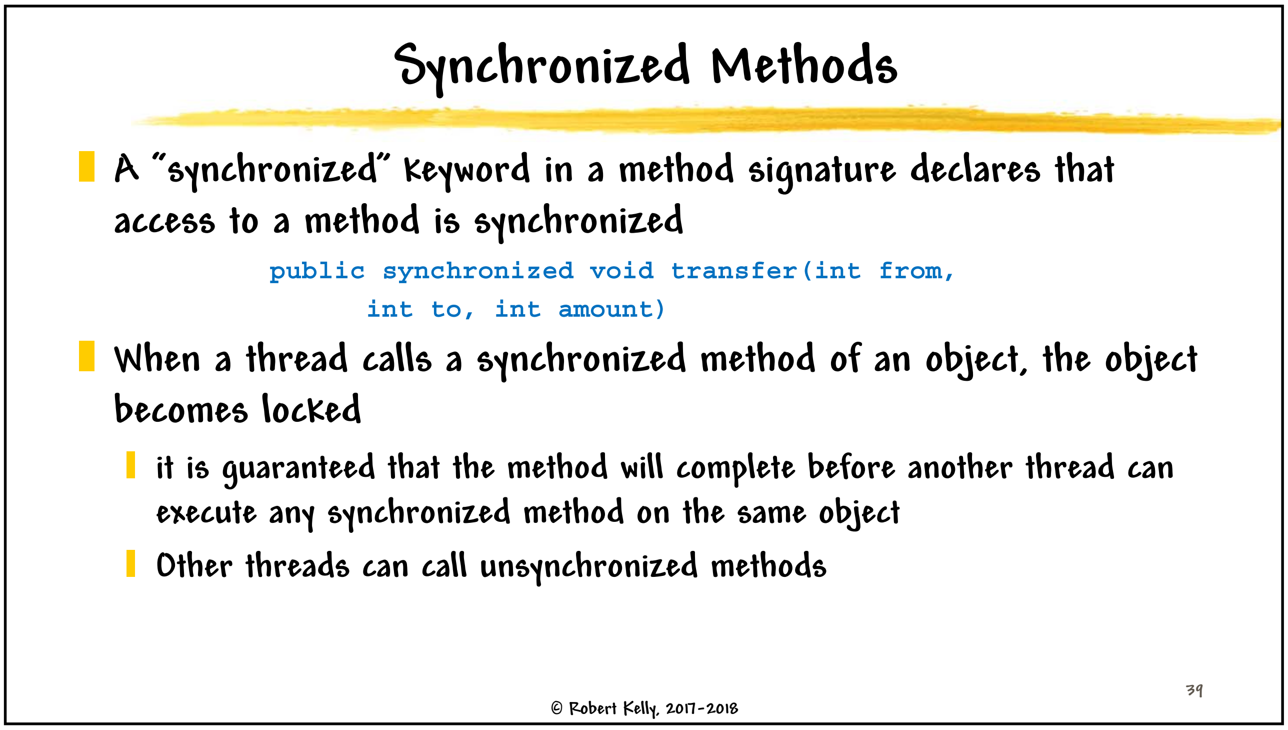
Multi-threaded Servlet Access



Synchronization

- It is possible for 2 or more threads to have access to the same object (or primitive)
- Most operations are not indivisible (modifications require multiple machine instructions), so corruption can result (called a race condition)
- To avoid simultaneous access to a shared object, you synchronize access to the object
  - Synchronized method
  - Synchronized block
  - Single Thread Model

Remember: a servlet local variable is not shared



Synchronized Methods

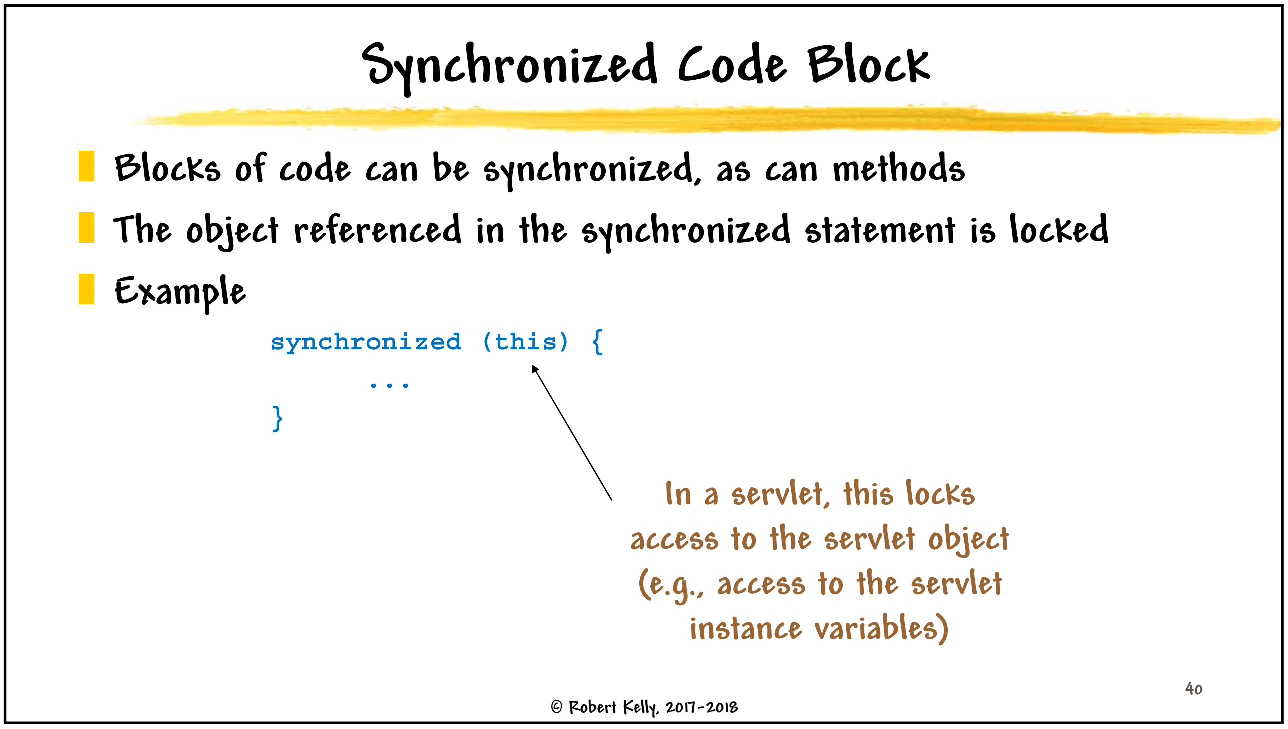
- A "synchronized" keyword in a method signature declares that access to a method is synchronized

```
public synchronized void transfer(int from,
        int to, int amount)
```

- When a thread calls a synchronized method of an object, the object becomes locked
  - it is guaranteed that the method will complete before another thread can execute any synchronized method on the same object
  - Other threads can call unsynchronized methods

© Robert Kelly, 2017-2018

39



Synchronized Code Block

- Blocks of code can be synchronized, as can methods
- The object referenced in the synchronized statement is locked
- Example

```
synchronized (this) {
        ...
}
```

In a servlet, this locks access to the servlet object (e.g., access to the servlet instance variables)

© Robert Kelly, 2017-2018

40

Single Thread Model

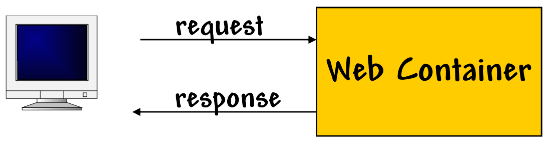
- Your servlet can implement the (empty) SingleThreadModel interface
- The server guarantees that "no two threads will execute concurrently in the servlet's service method"
- Much better to synchronize access than to use the SingleThreadModel

© Robert Kelly, 2017-2018

41

Browser Side Storage

You will rarely use browser side storage



- Data stored on the browser is included in the response object and returned to the servlet through the request object
- What data is usually transmitted through http?
  - Form data set
  - Cookies

© Robert Kelly, 2017-2018

42

Cookies

- A cookie is a small amount of information sent by the server to the browser that can later be read back from the browser
- Usually contained in a Cookies folder

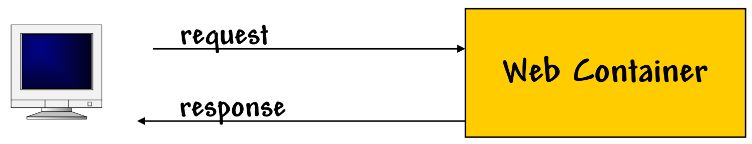
Typical cookie

```
Adc1
11780|NY56|078|@NY|ISP|ISP
accuweather.com/
0
3337461760
29399690
101711582429393656
*
Adc2
5|1|40.88|-73.16|SAINT JAMES
...
```

© Robert Kelly, 2017-2018

43

Cookie Process



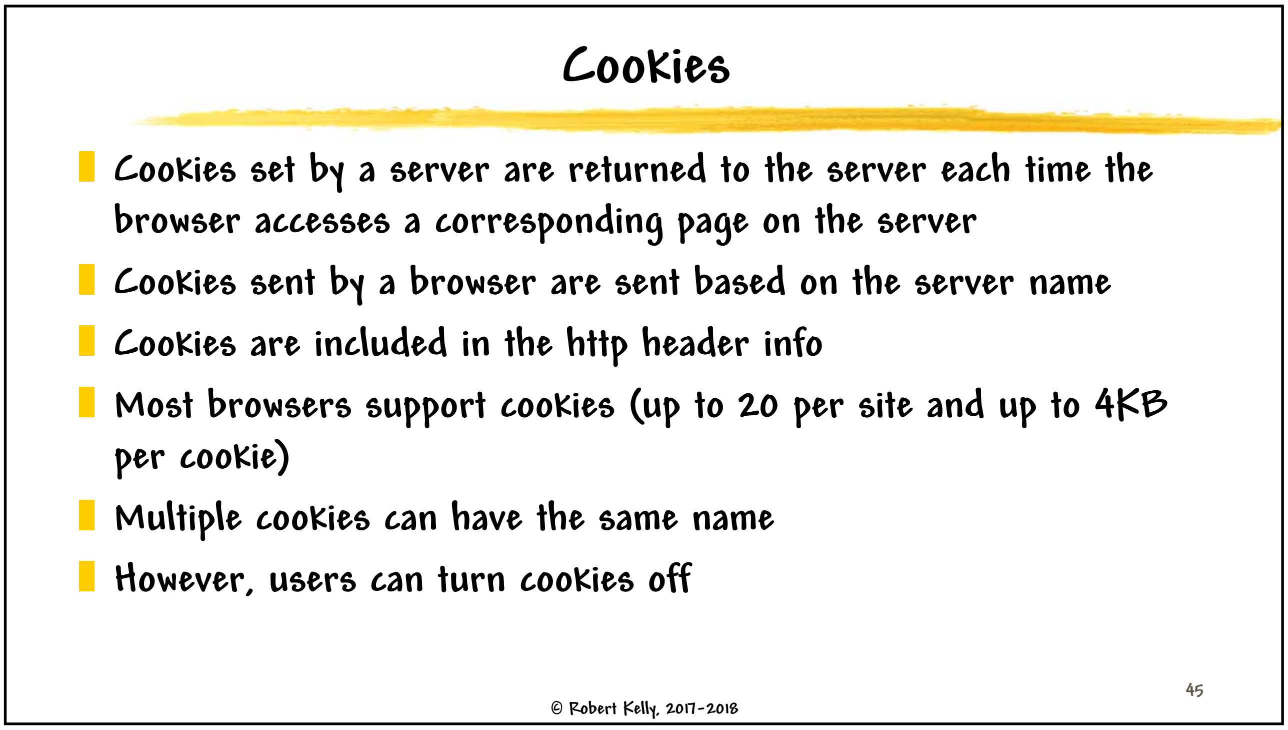
1. Your servlet "sets a cookie" by including it in the response

2. Your browser stores the cookie In your cookies directory on your hard disk

3. Your browser sends the cookie every time a request is made to a server "in your domain"

© Robert Kelly, 2017-2018

44

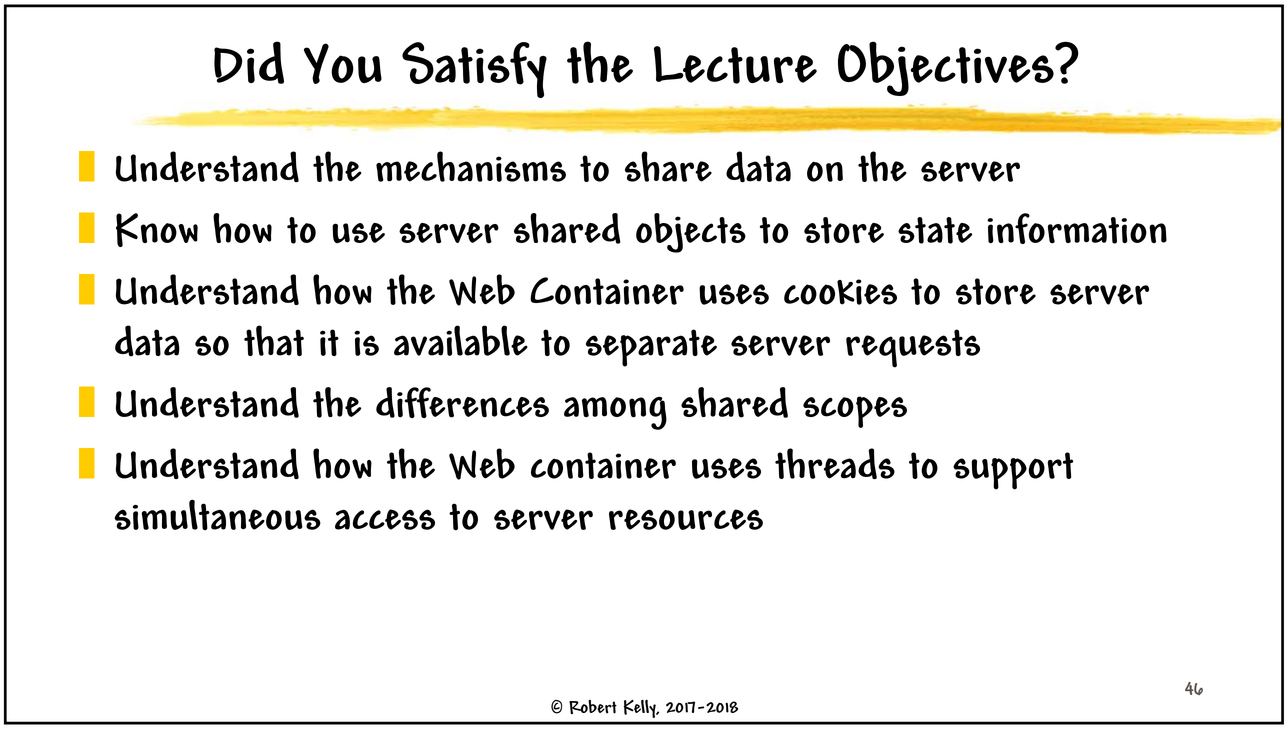


Cookies

- Cookies set by a server are returned to the server each time the browser accesses a corresponding page on the server
- Cookies sent by a browser are sent based on the server name
- Cookies are included in the http header info
- Most browsers support cookies (up to 20 per site and up to 4KB per cookie)
- Multiple cookies can have the same name
- However, users can turn cookies off

© Robert Kelly, 2017-2018

45



Did You Satisfy the Lecture Objectives?

- Understand the mechanisms to share data on the server
- Know how to use server shared objects to store state information
- Understand how the Web Container uses cookies to store server data so that it is available to separate server requests
- Understand the differences among shared scopes
- Understand how the Web container uses threads to support simultaneous access to server resources

© Robert Kelly, 2017-2018

46