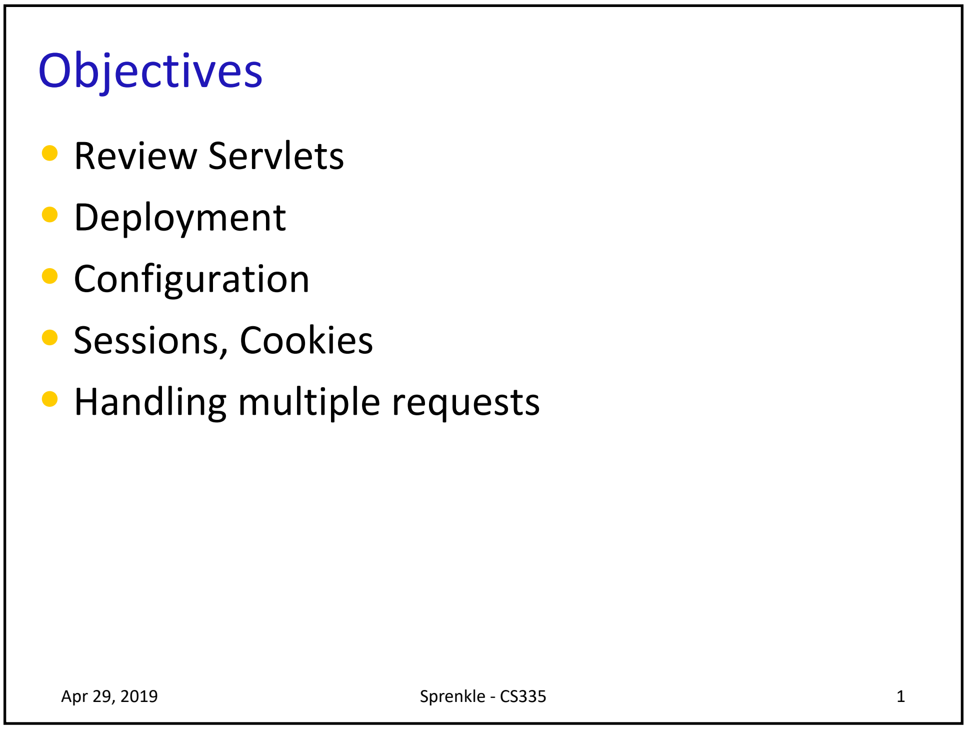
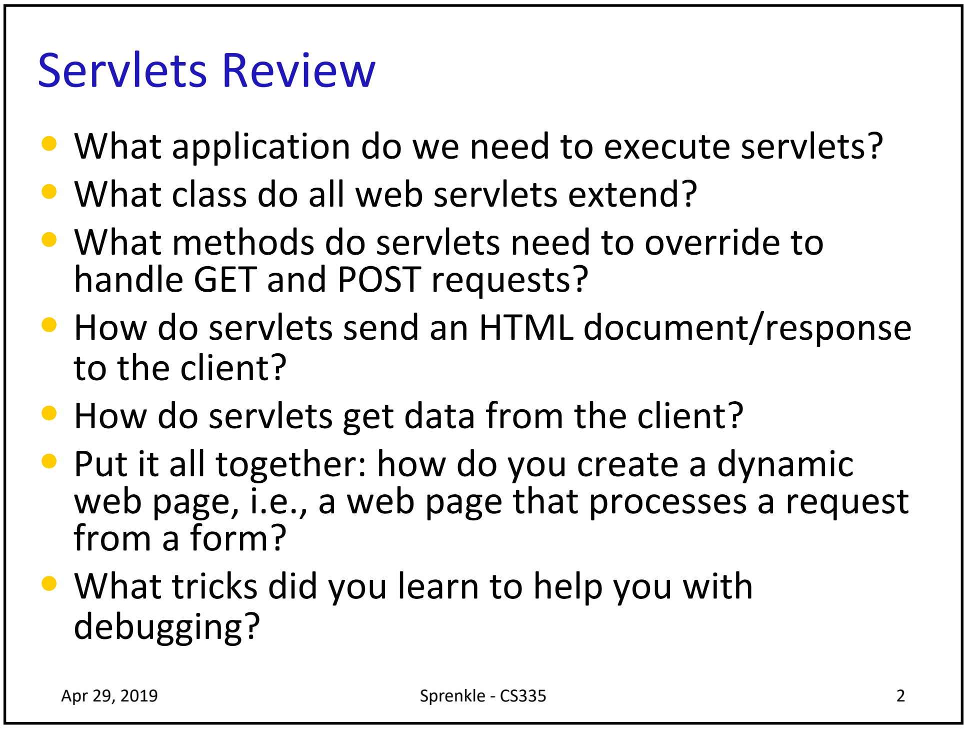


Objectives

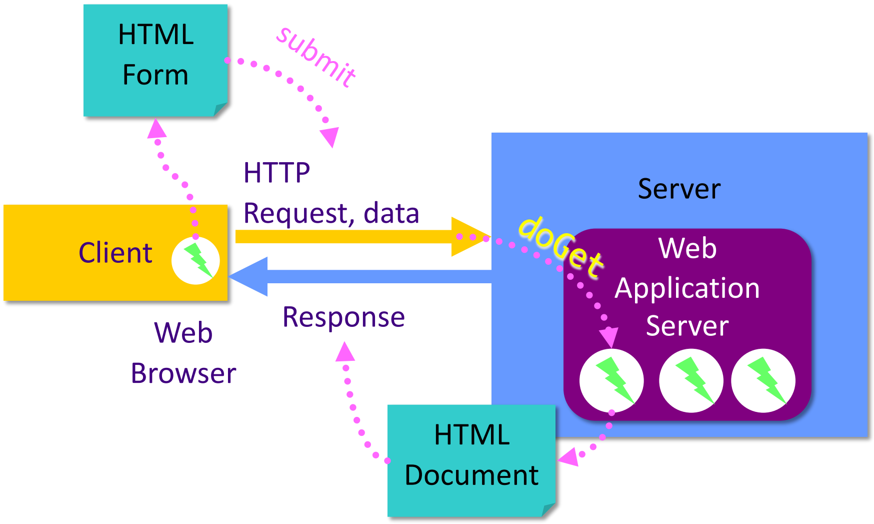
- Review Servlets
- Deployment
- Configuration
- Sessions, Cookies
- Handling multiple requests



Servlets Review

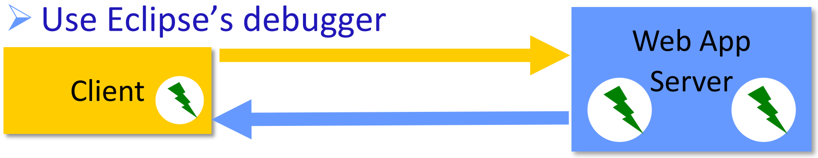
- What application do we need to execute servlets?
- What class do all web servlets extend?
- What methods do servlets need to override to handle GET and POST requests?
- How do servlets send an HTML document/response to the client?
- How do servlets get data from the client?
- Put it all together: how do you create a dynamic web page, i.e., a web page that processes a request from a form?
- What tricks did you learn to help you with debugging?

Example Servlet Flow



Servlet Development Discussion

- Distributed applications are difficult to debug and test
  - Multiple components: Client code? Server code?
- Suggestions
  - Use Eclipse to help you find errors in HTML
  - Check response’s HTML source code
    - Shows you what was written to output
    - Location of error
  - Use Eclipse’s debugger



More on Java-based Web Applications

- Structure
- Other classes
- Initialization, customization
- Synchronization

Web App Directory Structure

- Different from Eclipse code organization
- When Eclipse deploys the web application, it organizes it this way.

- `projectname/`
  - HTML, CSS, and JSP files
- `projectname/WEB-INF`
  - Other resources, e.g., `web.xml`
- `projectname/WEB-INF/classes`
  - Servlet and utility (data structures, etc)
  - Why we put our servlets in `servlets` package
- `projectname/WEB-INF/lib`
  - Jar files that application depends on

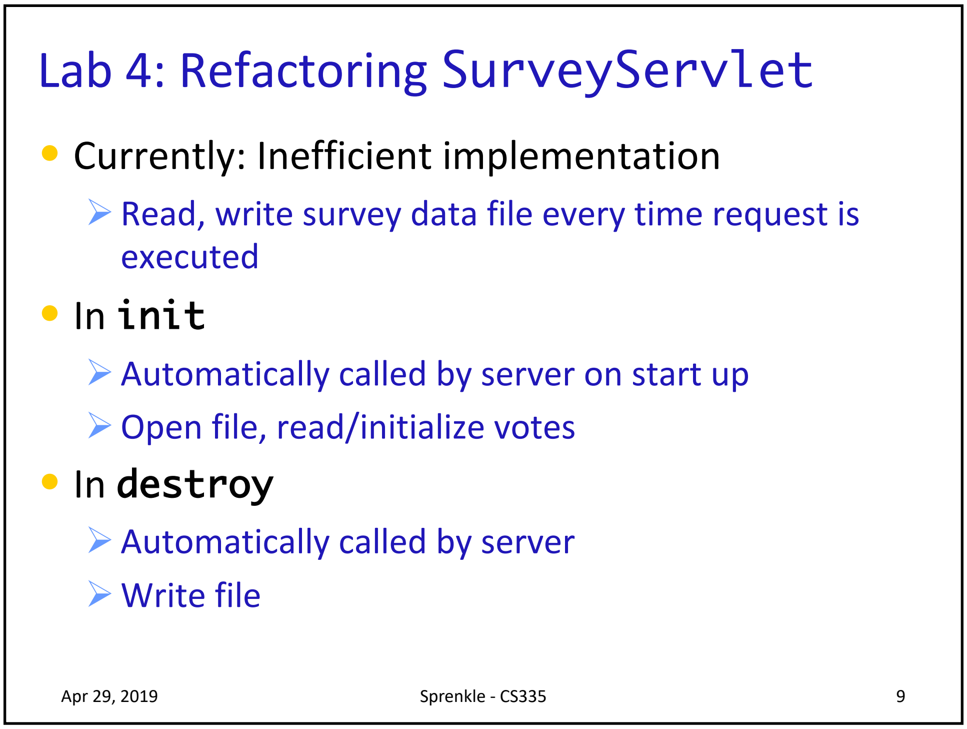
Servlet Interface Methods

- `init(ServletConfig config)`
  - Web app server calls once to initialize the servlet
  - Typically opening DB connection, files
- `ServletConfig getServletConfig()`
  - Returns a reference to a `ServletConfig`
- `void service(ServletRequest, ServletResponse)`
  - Called to respond to a client request
- `String getServletInfo()`
  - Returns a String that describes the servlet (name, version, etc.)
- `void destroy()`
  - Called by the server to terminate a servlet
  - Should close open files, close DB connections, etc.

Servlet Life Cycle in Web Application Server

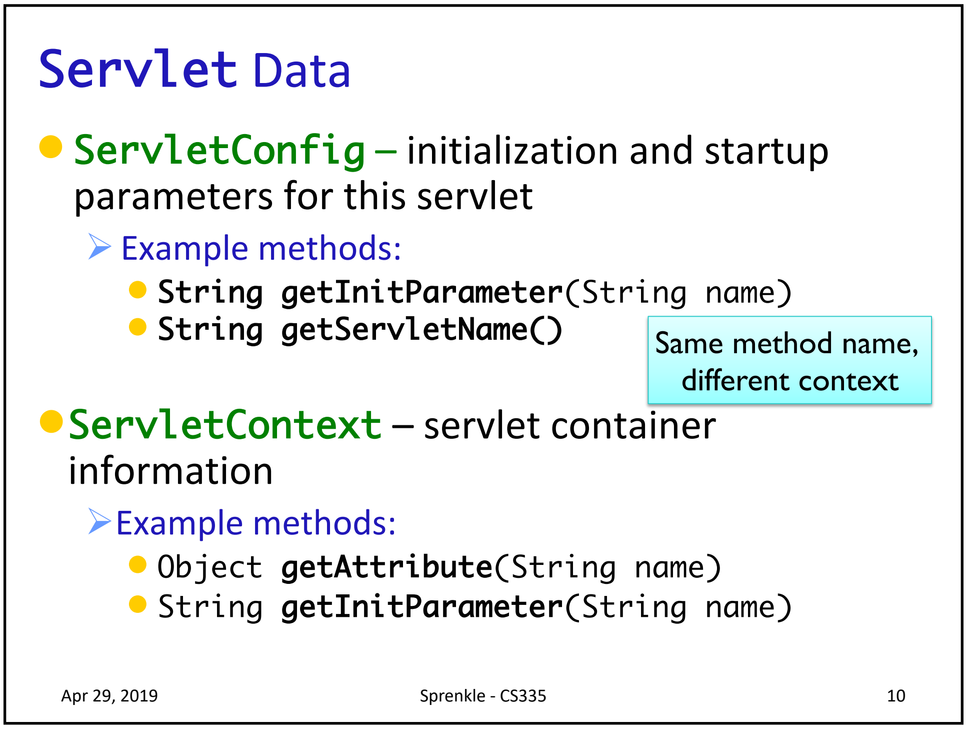
SurveyServlet
Parameter Servlet
Web Application Server

- Web application server creates **one** instance of servlet
  - Calls **`init`** method of servlet created
- As requests come in, WAS calls **`service`** method of appropriate servlet
  - In turn, servlet calls appropriate **doMethod**
- When web application server shuts down, calls **`destroy`** method of each servlet



Lab 4: Refactoring SurveyServlet

- Currently: Inefficient implementation
  - Read, write survey data file every time request is executed
- In **init**
  - Automatically called by server on start up
  - Open file, read/initialize votes
- In **destroy**
  - Automatically called by server
  - Write file



Servlet Data

- `ServletConfig` – initialization and startup parameters for this servlet
  - Example methods:
    - `String getInitParameter(String name)`
    - `String getServletName()`

Same method name, different context

- `ServletContext` – servlet container information
  - Example methods:
    - `Object getAttribute(String name)`
    - `String getInitParameter(String name)`

ServletContext

- One `ServletContext` per web application per JVM
  - If you have both Lab3 and FirstServlets running on Tomcat, they will each have their own ServletContext
- Share state among multiple clients
  - Allow multiple users to interact in, e.g., chat rooms, online meeting, reservation systems
- Info about servlet's environment
  - E.g., server's name
- `log()`: method to write to a log file
- Context attributes
  - `getAttribute, setAttribute, removeAttribute`

`web.xml` File

- Describes how to deploy the web application
- XML file
  - Used for data
  - Marked up with elements
  - Same rules as XHTML: close most recently opened tag, attributes in quotes

```
<tag attr="value">
     Content
</tag>
```

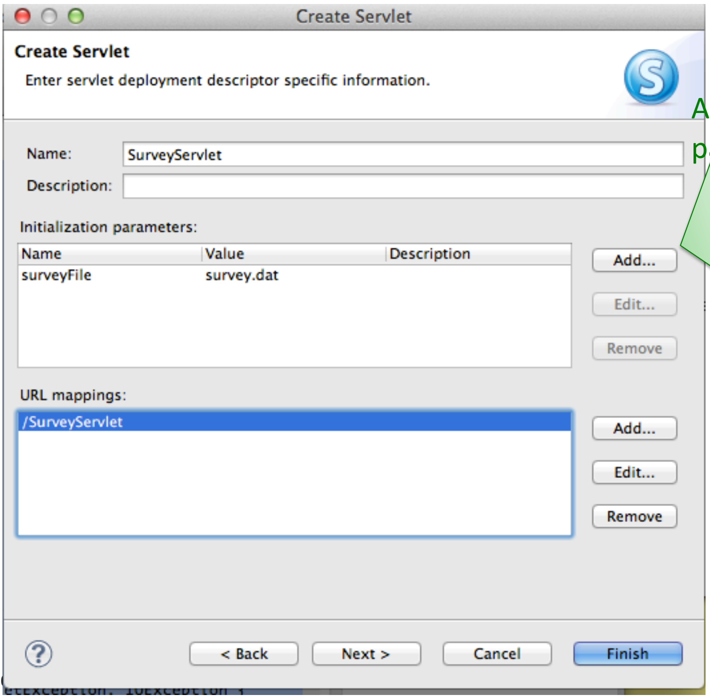
- DTD: Document Type Definition
  - Define elements that can be in a particular XML document
  - Includes specification of attributes, nesting

Annotations

- In Servlets 3.x, we can easily configure a web application using ***annotations***
  - Don’t need to directly update web.xml
  - Provide defaults, can be overridden in web.xml
- Example:

```
@WebServlet("/SurveyServlet")
public class SurveyServlet extends HttpServlet {
```

  - Means the URL pattern “/SurveyServlet” maps to this servlet (servlets.SurveyServlet)



Create Servlet

Create Servlet

Enter servlet deployment descriptor specific information.

Name: SurveyServlet

Description:

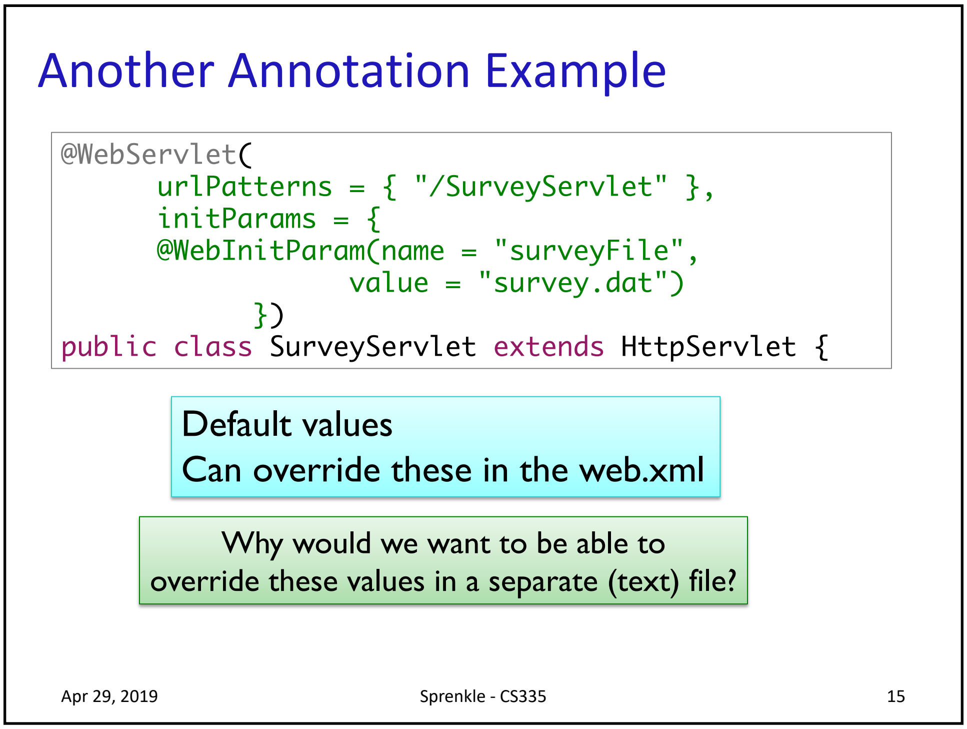
Initialization parameters:

Name	Value	Description
surveyFile	survey.dat	

URL mappings:

/SurveyServlet

Add init parameters

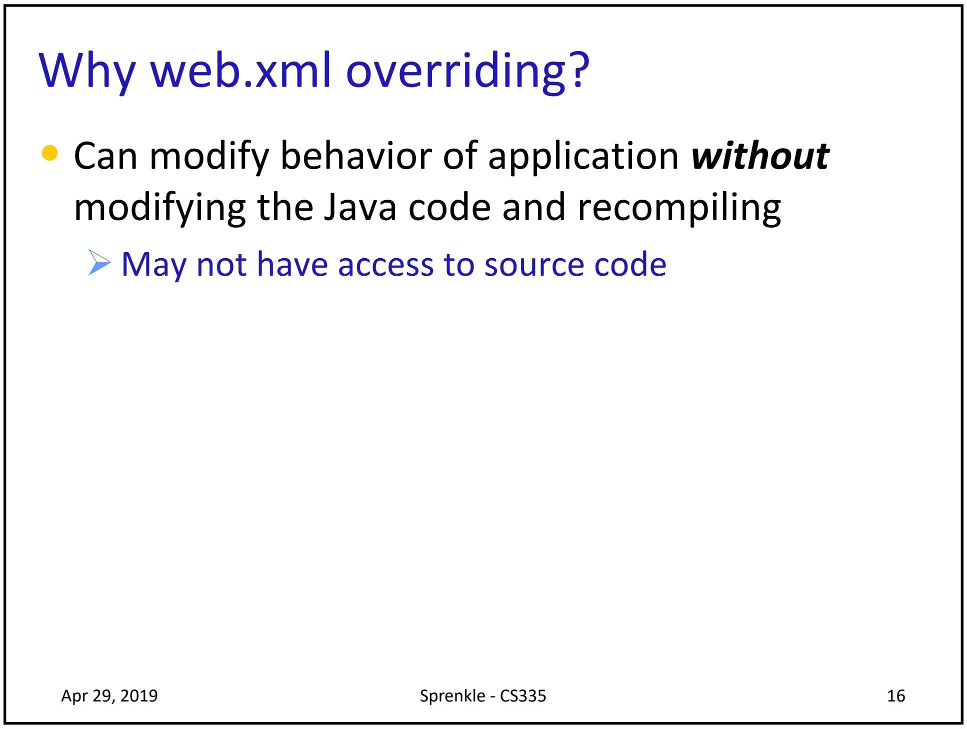


Another Annotation Example

```
@WebServlet(
      urlPatterns = { "/SurveyServlet" },
      initParams = {
      @WebInitParam(name = "surveyFile",
                  value = "survey.dat")
            })
public class SurveyServlet extends HttpServlet {
```

Default values
Can override these in the web.xml

Why would we want to be able to override these values in a separate (text) file?



Why web.xml overriding?

- Can modify behavior of application ***without*** modifying the Java code and recompiling
  - May not have access to source code

web.xml File

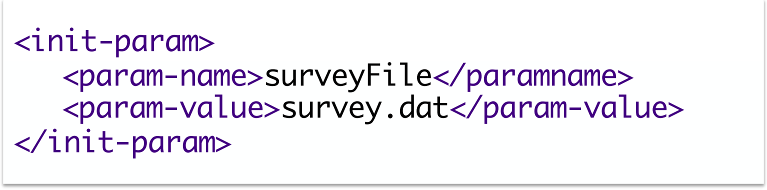
- Top-level: <webapp>
- <servlet> element describes a servlet
- <servlet-mapping> element maps URLs to servlets
  - May want to have shorthands, aliases
  - Restrict users' direct access to servlets

web.xml File: Subelements of <servlet>

<servlet-name>	canonical name of the deployed servlet
<servlet-class>	fully qualified class name of the servlet
<init-param>	optional parameter containing a name-value pair that is passed to the servlet on initialization. Contains elements, <param-name> and <param-value>, which contain the name and value, respectively, to be passed to the servlet.

Example of Configuring web.xml

- Configure `SurveyServlet` to use a given file
- Add the following to web.xml file:



```
<init-param>
   <param-name>surveyFile</paramname>
   <param-value>survey.dat</param-value>
</init-param>
```

- Note that `<init-param>` is a child of `<servlet>`, which means your web.xml file would look like what?

Note about init-params in web.xml

- If you set init-param in web.xml, you need to *annotate* the servlet with its name
- You can have multiple configurations for the same [servlet] class
  - Using the name lets the application server know that the annotations and the web.xml configurations are both part of the same configuration

Using Init Parameter

- Configure `SurveyServlet` to use a given file
  - Either in annotation or web.xml
- Modify `init` method to call **`HttpServlet`**’s **`getInitParameter`** method

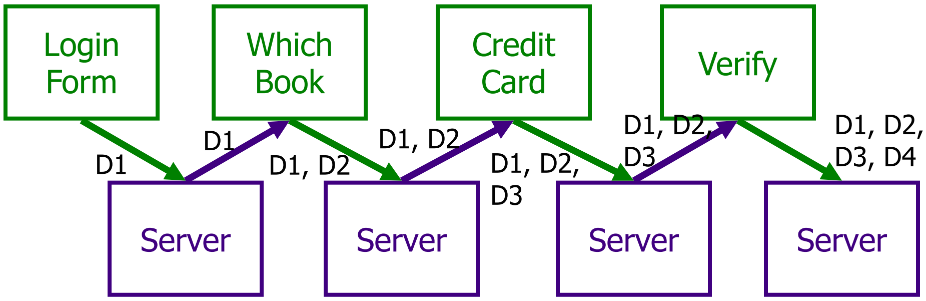
```
// calls HttpServlet method, i.e., this’s method
filename = getInitParameter("surveyFile");

// open file …
```

MAINTAINING STATE ACROSS REQUESTS

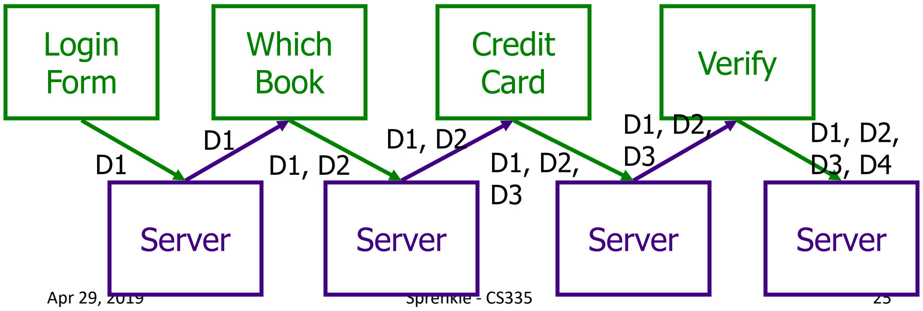
Maintaining State

- If you have multiple pages, how can you save or accumulate data?
  - Example scenario: buying a book



Maintaining State

- If you have multiple pages, how can you save or accumulate data?
  - Hidden fields (**`type=hidden`**)
  - Cookies
  - Sessions



Hidden Fields

```
<input type="hidden" name="userid" value="superfly"/>
```

- Data is coming from client
- Users can see the hidden fields
  - View HTML Source
- Users can change the data

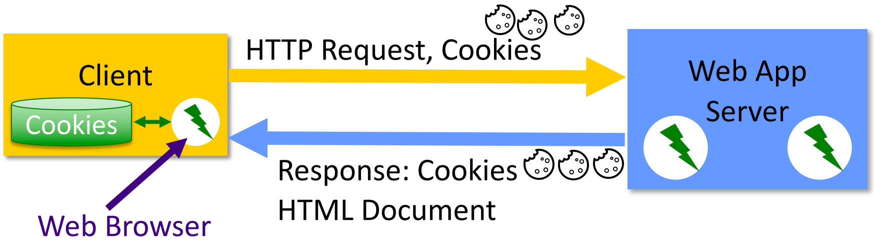
➡ Useful in limited situations

COOKIES

Cookies

- Cookies are initially sent from the webapp to the client to store application-specific information on the client
- Part of an HTTP header in response to a client
  - Every HTTP transaction includes HTTP headers
  - Not part of the HTML content
- Client includes cookies in HTTP headers in subsequent requests
  - Provides way to do behavior tracking

Process with Cookies



- Cookies
  - Associated with server name
  - Part of HTTP Headers
- Example: Amazon.com
  - Cookie stores your name, login information
  - Example: Not Sara?

Cookies in Java

- Cookies have a name and value
- Create a Cookie object using its constructor
  - Part of `javax.servlet.http.Cookie`
- Example: store a user's preferred language on the client
  - App only has to ask for this information once

```
String cookie_name = "pref_language";
String cookie_value = "English";
Cookie new_cookie = new Cookie(cookie_name, cookie_value);
```

Sending the Cookie to the Client

- HTTP header is sent first
- Cookie(s) must be added to the response object ***before*** you start writing to the client
- Call **`addCookie`**() on **`HttpServletResponse`** object before you call the **`getWriter`**() method
- Inside of **`doGet`** or **`doPost`** method:

```
Cookie c = new Cookie( "pref_language", "English" );
c.setMaxAge(60*60*24*365);  // max age of cookie
response.addCookie(c);
…
output = response.getWriter();
```

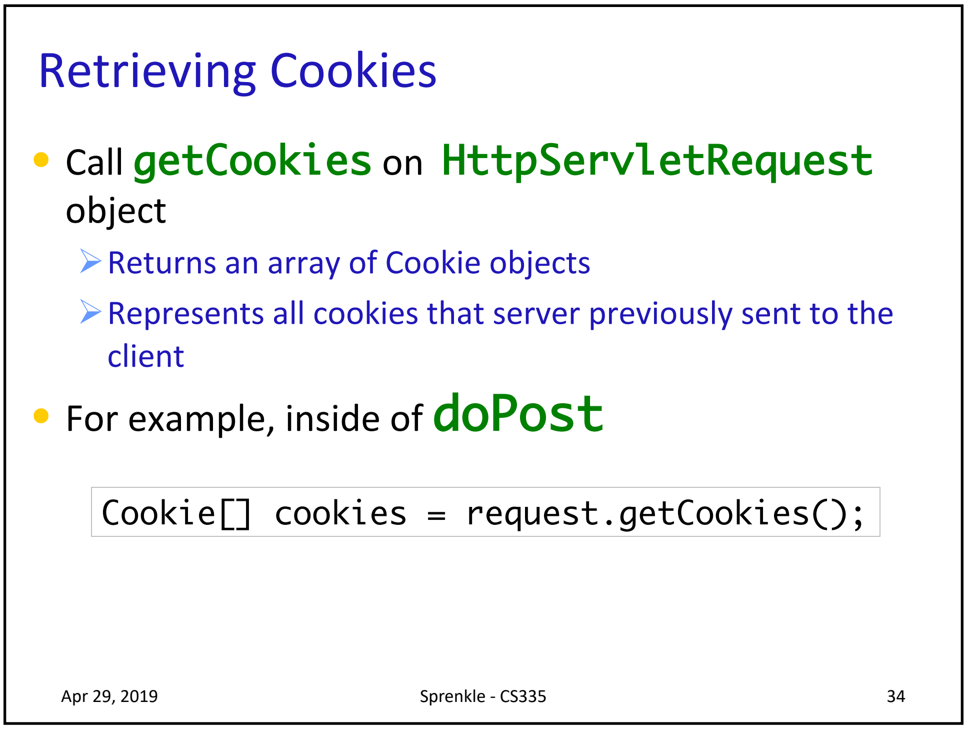
HttpServletResponse Method

- void addCookie(Cookie)
  - Add a Cookie to the header in the response to the client
  - The cookie will be stored on the client, depending on the max-life and if the client allows cookies

Cookies: Maximum Ages

```
c.setMaxAge(60*60*24*365);  // max age of cookie
```

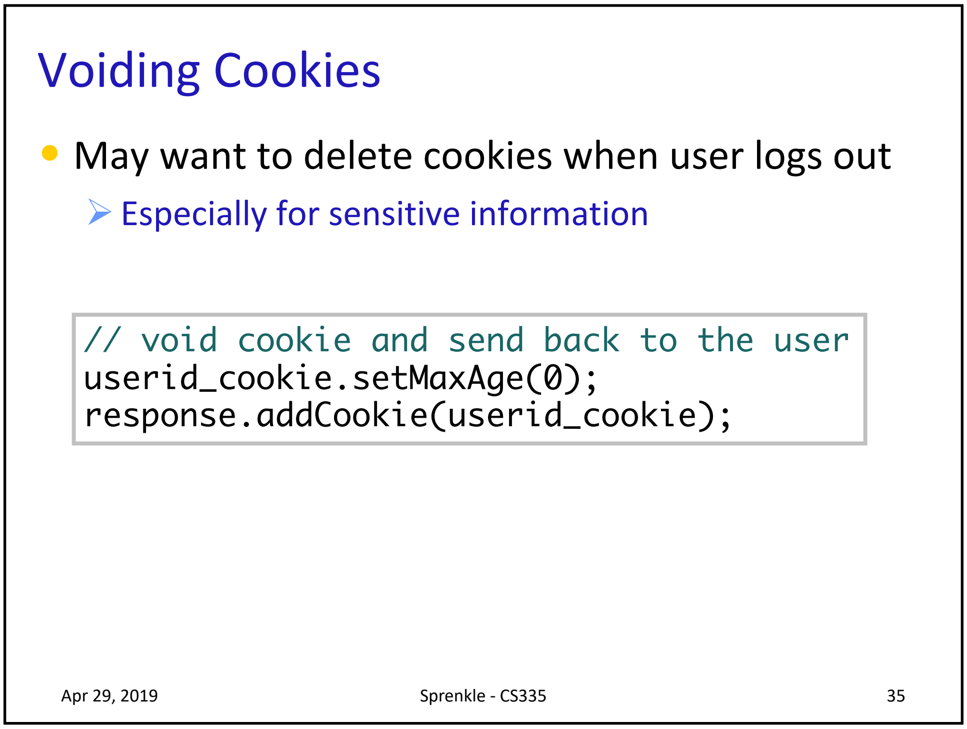
- The maximum age of the cookie is how long the cookie can live on the client, in seconds
- When a cookie reaches its maximum age, client deletes it
- -1 means persists until browser exits



Retrieving Cookies

- Call `getCookies` on `HttpServletRequest` object
  - Returns an array of Cookie objects
  - Represents all cookies that server previously sent to the client
- For example, inside of `doPost`

```
Cookie[] cookies = request.getCookies();
```



Voiding Cookies

- May want to delete cookies when user logs out
  - Especially for sensitive information

```
// void cookie and send back to the user
userid_cookie.setMaxAge(0);
response.addCookie(userid_cookie);
```

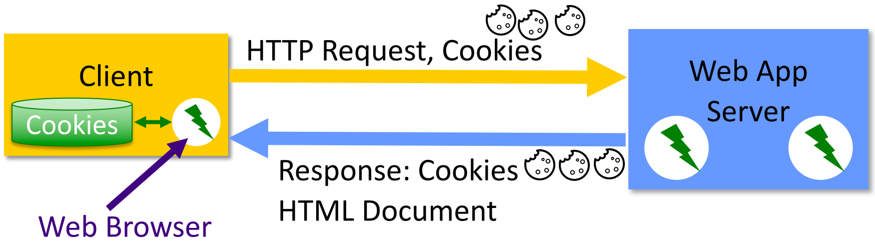
Why Are They “Cookies”?

- Http Cookie, Source: Wikipedia
  - The term "cookie" derives from "magic cookie", which is a packet of data a program receives and sends out again unchanged.
- Magic Cookie, Source: Wikipedia
  - The name "cookie" comes from a comparison to an unopened fortune cookie, because of the hidden information inside.

What are challenges with using cookies?

What are challenges with using cookies?

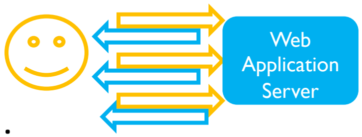
- They are saved on the client machine
  - ➢ Clients can delete or modify them



- Increase the sizes of your network packets
  - ➢ Send cookies on each request

SESSION STATE

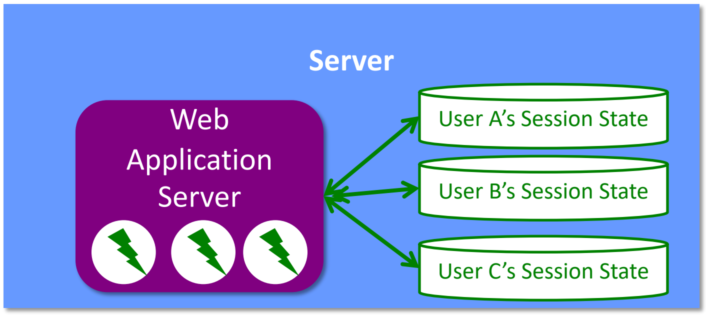
Session



- One user’s visit to an application
- Can be made up of many requests
- Server maintains a session with a particular client
  - Can maintain ***state*** within that session
- Duration of a session:
  - If no requests from client for specified period of time (the timeout), user’s session ends
  - Timeout: typically 30 minutes

Benefits of Using Session State

- Simpler for developer
- Reduces network traffic
  - Don’t need to keep passing data between client and server

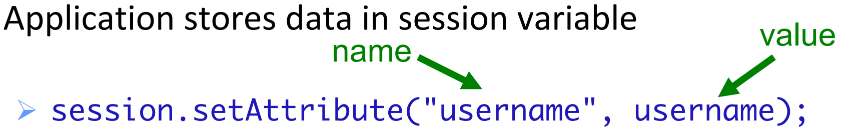


Session State in Java

- `HttpSession` stores session data
- Data is known as ***session attributes***
  - Have names and values
- Store, access, and remove attributes:
  - Like a HashMap
  - `void setAttribute(String name, Object value)`
    - Values no longer need to be strings
    - Cookies and Parameters had to be strings
  - `Object getAttribute(String name)`
  - `void removeAttribute(String name)`

Example Session Variables

- User gives application data
- Application stores data in session variable



```
session.setAttribute("username", username);
```

- Application can use later in session, without user having to give information again

```
String username = (String)
                  session.getAttribute("username");
```

- More examples:
  - Server computes information once, caches in session
  - Shopping carts

Getting a Session

- HttpServletRequest's getSession(boolean create) method
  - Returns the current HttpSession object
  - Boolean parameter specifies if a new session should be created if one does not already exist

Other Useful Session Methods

- setMaxInactiveInterval(), getCreationTime(), getLastAccessedTime()
  - If want shorter than server's timeout
- invalidate()
  - Invalidates session, unbinds objects bound to it

Lab 4: Add Session Variable

- `LoginServlet` will add a session variable with name “authenticated”

Eclipse Development Hints

- Safe bet: restart server whenever change to a servlet
  - Can modify Server’s configuration
    - Under Publishing
- Typical programming
  - Write a few lines of code/make small changes
  - Run, test
  - Repeat

HANDLING MULTIPLE REQUESTS

Multiple Clients

- Web server handles multiple requests at a time by executing multiple *threads*

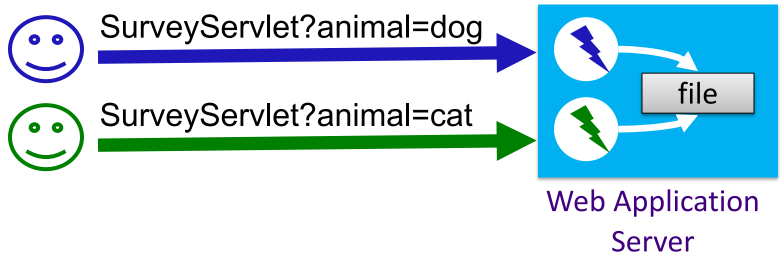
- Approximately 1 thread/request

⇒Need to make sure that threads overlap in ways that do not break the application

Example Scenario

- Original **SurveyServlet** stores the results of the survey in a file on the server
- When >1 client connects to the server at one time, server handles both clients **concurrently**
  - >1 can execute **SurveyServlet**
  - >1 thread can read/modify file at one time
  - Can lead to inconsistent data!

SurveyServlet Implementation

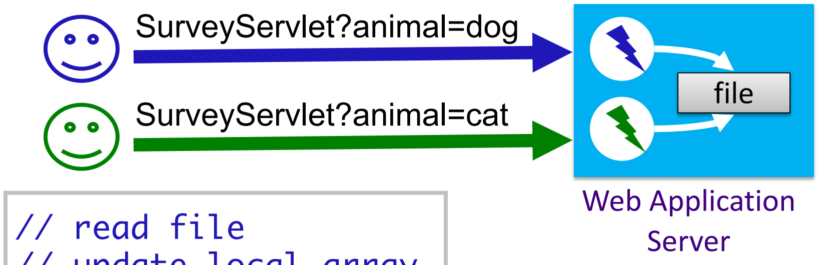


- Operations can overlap

```
// read file
// update local array
// write file
// print results
```

```
// read file
// update local array
// write file
// print results
```

Bad Interleaving



```
// read file
// update local array
// read file
// update local array
// write file
// print results
// write file
// print results
```

What happens in this case?

Loses blue’s vote

Critical Section

- Sections of code that have to happen uninterrupted or **atomically**
  - Only one thread can execute at a time

- What is the critical section in this code?

```
// read file
// update local array
// write file
// print results
```

Critical Section

- Sections of code that have to happen uninterrupted or **atomically**
  - Only one thread can execute at a time
- What is the critical section in this code?
  - The shared file must be read and written atomically
- **Writes** cause trouble

```
// read file
// update local array
// write file
// print results
```

210 in 335

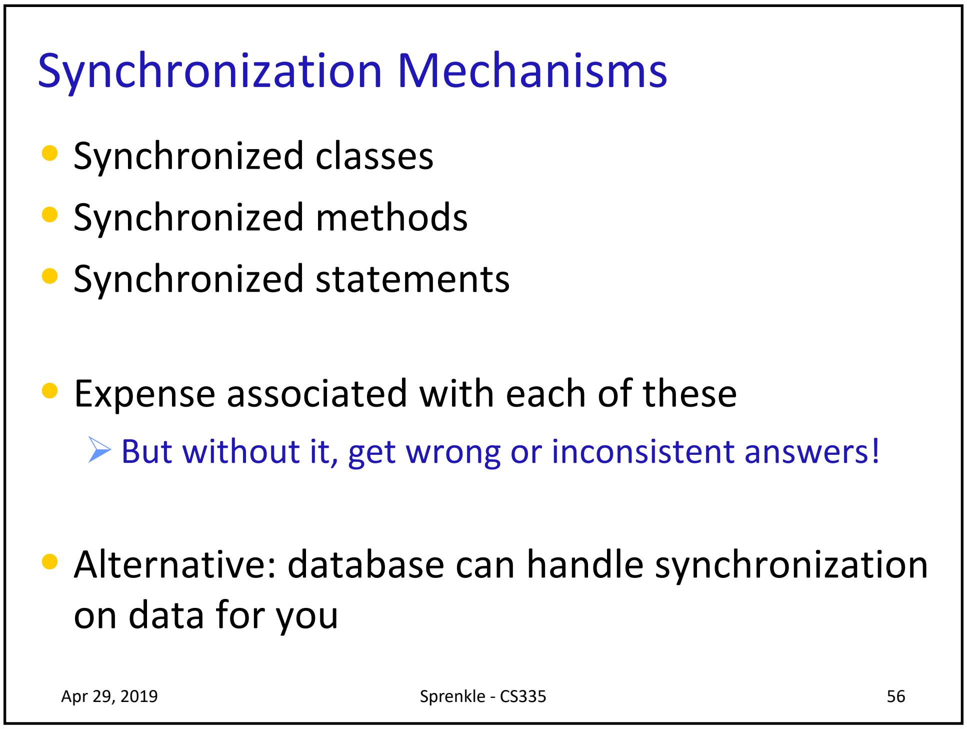
- Even if only one Java statement in critical section, synchronize it!
- One high-level Programming Language statement probably translates into multiple VM language statements
  - Prevent interruption at low level

High-level:

```
count++;
```

Virtual Machine level:

```
Retrieve count
Add 1 to count
Store count
```

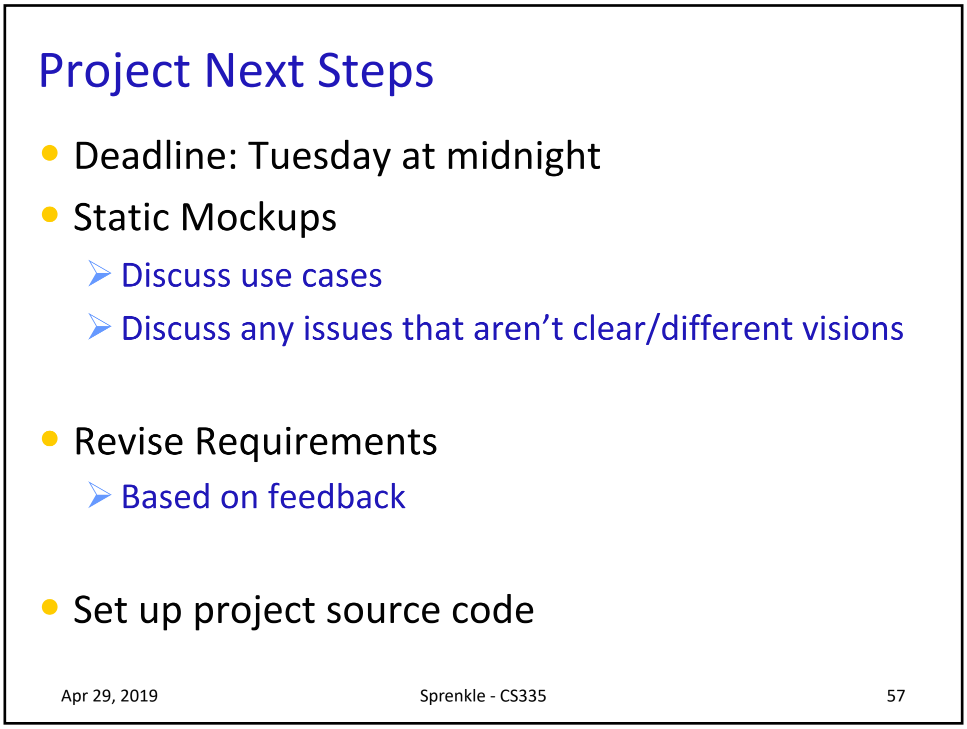


Synchronization Mechanisms

- Synchronized classes
- Synchronized methods
- Synchronized statements

- Expense associated with each of these
  - ➢ But without it, get wrong or inconsistent answers!

- Alternative: database can handle synchronization on data for you



Project Next Steps

- Deadline: Tuesday at midnight
- Static Mockups
  - ➢ Discuss use cases
  - ➢ Discuss any issues that aren’t clear/different visions

- Revise Requirements
  - ➢ Based on feedback

- Set up project source code

TODO

- Lab 4: Servlet Configuration and Session State
  - Create a GitHub account and email to me
  - Init, destroy methods
  - Configuration parameters
  - Session state
- Tomorrow: Static Mockups
- Read/Summarize Quality Attributes paper by Wed, midnight
  - See Sakai description for details about contents