

Objectives

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

May 2, 2016

Sprengle - CS335

1

Servlets Review

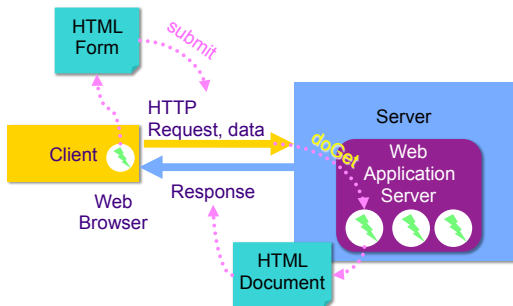
- What application do we need to execute servlets?
- What class do all web servlets extend?
- What methods do servlets need to override to handle GET and POST requests?
- How do servlets send an HTML document/response to the client?
- How do servlets get data from the client?

May 2, 2016

Sprengle - CS335

2

The Example Servlet Flow



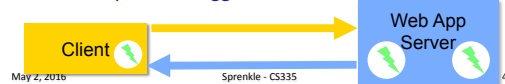
May 2, 2016

Sprengle - CS335

3

Servlet Development Discussion

- Distributed applications are difficult to debug and test
 - Many 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



May 2, 2016

Sprengle - CS335

4

More on Java-based Web Applications

- Structure
- Other classes
- Initialization, customization
- Synchronization

May 2, 2016

Sprengle - CS335

5

Web App Directory Structure

- **projectname/** Different from Eclipse code organization
 - 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

May 2, 2016

Sprengle - CS335

6

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.

May 2, 2016

Sprengle - CS335

7

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

May 2, 2016

Sprengle - CS335

8

Servlet Data

- **ServletConfig** – initialization and startup parameters for this servlet
 - Example methods:
 - **String getInitParameter(String name)**
 - **String getServletName()**
- **ServletContext** – servlet container information
 - Example methods:
 - Object **getAttribute(String name)**
 - String **getInitParameter(String name)**

Same method name,
different context

May 2, 2016

Sprengle - CS335

9

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**

May 2, 2016

Sprengle - CS335

10

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

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

May 2, 2016

Sprengle - CS335

11

web.xml File

Show 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

May 2, 2016

Sprengle - CS335

12

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.

May 2, 2016

Sprengle - CS335

13

Configure SurveyServlet

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

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

May 2, 2016

Sprengle - CS335

14

Configure SurveyServlet

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

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

- Modify init method to call HttpServlet's **getInitParameter** method

```
// calls HttpServlet method
filename = getInitParameter("surveyFile");
// open file ...
```

May 2, 2016

Sprengle - CS335

15

Annotations

- In Servlets 3.x, we can easily configure a web application **annotations**

- Don't need to directly update web.xml
- Provide defaults, can be overridden in web.xml

- Example:

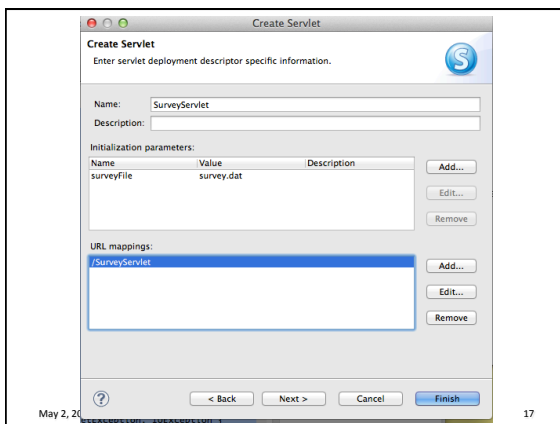
```
@WebServlet("/SurveyServlet")
public class SurveyServlet extends HttpServlet {
```

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

May 2, 2016

Sprengle - CS335

16



May 2, 2016

17

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?

May 2, 2016

Sprengle - CS335

18

Why web.xml overriding?

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

May 2, 2016

Sprengle - CS335

19

MAINTAINING STATE ACROSS REQUESTS

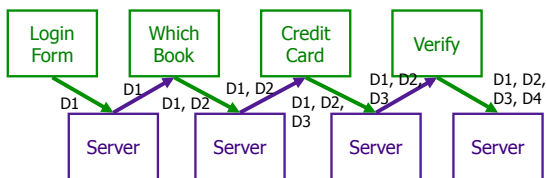
May 2, 2016

Sprengle - CS335

20

Maintaining State

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



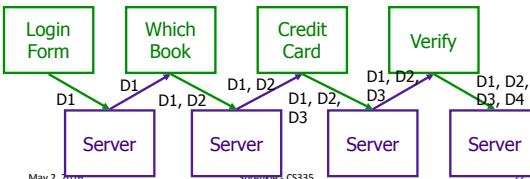
May 2, 2016

Sprengle - CS335

21

Maintaining State

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



May 2, 2016

Sprengle - CS335

22

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

May 2, 2016

Sprengle - CS335

23

COOKIES

May 2, 2016

Sprengle - CS335

24

Cookies

Do you see any issues with cookies?

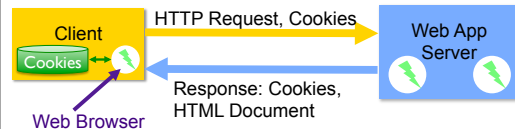
- Cookies are initially sent from the webapp to the client to store application-specific information on the client
- Part of an HTTP header in response to a client
 - Every HTTP transaction includes HTTP headers
 - Not part of the HTML content
- Client includes cookies in HTTP headers in subsequent requests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

May 2, 2016

Sprengle - CS335

25

Process with Cookies



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

May 2, 2016

Sprengle - CS335

26

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);
```

May 2, 2016

Sprengle - CS335

27

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();
```

May 2, 2016

Sprengle - CS335

28

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

May 2, 2016

Sprengle - CS335

29

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

May 2, 2016

Sprengle - CS335

30

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();
```

May 2, 2016

Sprenkle - CS335

31

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);
```

May 2, 2016

Sprenkle - CS335

32

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.

May 2, 2016

Sprenkle - CS335

33

SESSION STATE

May 2, 2016

Sprenkle - CS335

34

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

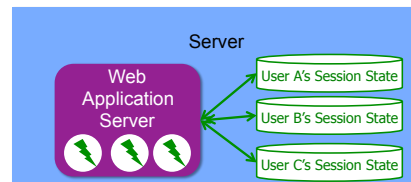
May 2, 2016

Sprenkle - CS335

35

Benefits of Using Session State

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



May 2, 2016

Sprenkle - CS335

36

Session State in Java

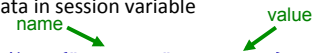
- `HttpSession` stores session data
- Data is known as **session attributes**
 - Have names and values
- Store, access, and remove attributes:
 - Like a `HashMap`
 - `void setAttribute(String name, Object value)`
 - Values no longer need to be strings
 - Cookies and Parameters had to be strings
 - `Object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

May 2, 2016

Sprengle - CS335

37

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 = session.getAttribute("username");`
- More examples:
 - Server computes information once, caches in session
 - Shopping carts

May 2, 2016

Sprengle - CS335

38

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

May 2, 2016

Sprengle - CS335

39

Other Useful Session Methods

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

May 2, 2016

Sprengle - CS335

40

Lab 4: Add Session Variable

- `LoginServlet` will add a session variable with name "authenticated"

May 2, 2016

Sprengle - CS335

41

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

May 2, 2016

Sprengle - CS335

42

HANDLING MULTIPLE REQUESTS

May 2, 2016

Sprengle - CS335

43

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

May 2, 2016

Sprengle - CS335

44

Example Scenario

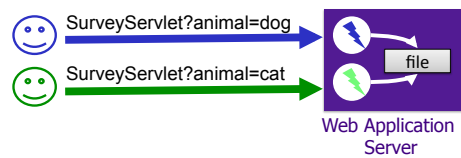
- Original **SurveyServlet** stores the results of the survey in a file on the server
- When >1 client connects to the server at one time, server handles both clients **concurrently**
 - >1 can execute **SurveyServlet**
 - >1 thread can read/modify file at one time
 - Can lead to inconsistent data!

May 2, 2016

Sprengle - CS335

45

SurveyServlet Implementation



- Operations can overlap

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

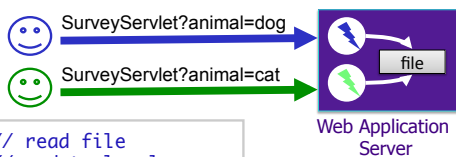
```
// read file  
// update local array  
// write file  
// print results
```

May 2, 2016

Sprengle - CS335

46

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

May 2, 2016

Sprengle - CS335

47

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
```

May 2, 2016

Sprengle - CS335

48

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
```

May 2, 2016

Sprengle - CS335

49

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
```

May 2, 2016

Sprengle - CS335

50

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 for you

May 2, 2016

Sprengle - CS335

51

Project Demos – Tuesday p.m.

- Static Mockups
 - Pick someone's machine for static mockup demo
 - Discuss through use cases
 - Discuss any issues that aren't clear/different visions
- Discussion of Requirements
 - Changes from original
 - JIRA
- Set up with project source code

May 2, 2016

Sprengle - CS335

52

TODO

- Lab 4: Servlets – Application and Session State
 - Init, destroy methods
 - Configuration parameters
 - Session state
- Tomorrow p.m.: Static Mockups demos
 - Discussion of requirements
 - JIRA
- Read/Summarize Quality Attributes paper by Wed, midnight

May 2, 2016

Sprengle - CS335

53