Application Design Using Java

Lecture 19

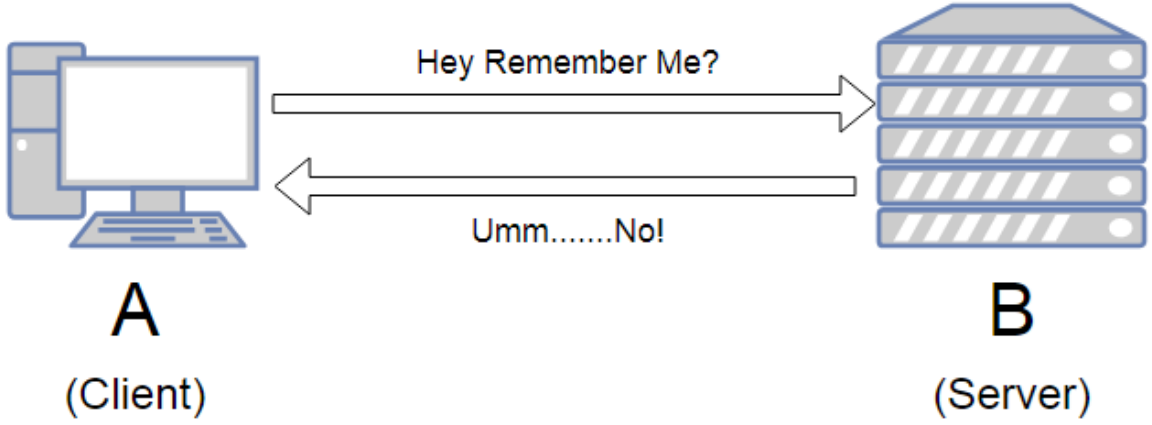
Accessing Form Data from Servlets

- GET
  - request.getQueryString()
  - Then parse the string
- GET and POST

Method	Description
getParameter()	Gets the value of a form parameter
getParameterValues()	Use if the parameter appears more than once and returns multiple values, for example checkbox
getParameterNames()	Gets a complete list of all parameters in the current request

State

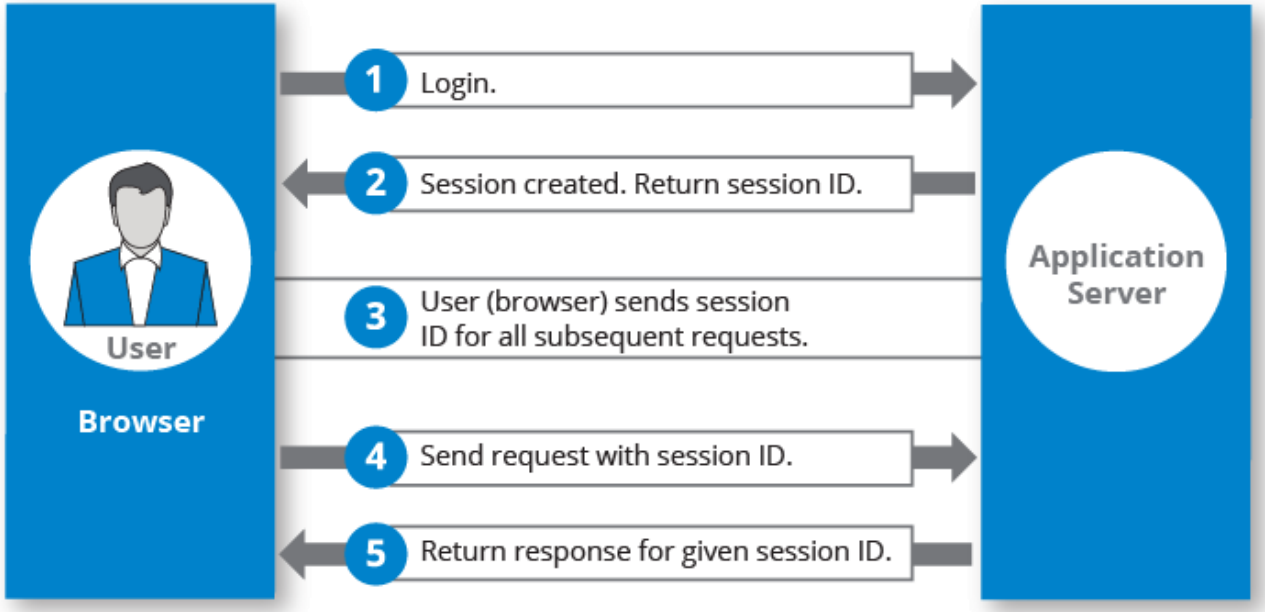
- HTTP is stateless
- Maintaining state requires special efforts
  - Reading/writing files on the server
  - Form data
  - Sessions
  - Cookies
  - Database



Form Data

- Hidden field inside the form
  - `<input type="hidden" name="`*`name`*`" value="`*`value`*`"></input>`
- Hidden from the user
- Read the values of hidden fields from the request object
- Write new state into the values of hidden fields when streaming the output of the response object

Session



- HttpSession object
- Obtained from the request
  `request.getSession()`
- When a user enters the Web site for the first time, they are given a unique ID to identify the session
- Unique ID can be stored into a cookie or in a request parameter
- HttpSession stays alive until it has not been used for more than the timeout value
  - Specified in the tag in the deployment descriptor file (`web.xml`)
  - Default timeout value is 30 minutes
- Store and retrieve state in a session object programmatically
  - `getAttribute()`
  - `setAttribute()`

Cookies

- Small pieces of state data
- Stored in the client's browser
- Sent back to the server for all the subsequent requests while the cookie is valid
- Sent in message headers
- Types
  - Session cookies
    - Do not have expiration time
    - Live in the browser memory
    - As soon as the web browser is closed this cookie gets destroyed
  - Persistent Cookies
    - Have expiration time
    - Stored on the client's hard drive (files or a database)
    - Get destroyed based on the expiry time

Java ARchive (JAR)

- A single archive file which you can deliver to your customers instead of a directory structure filled with class files
- Contain
  - Classes
  - Images
  - Audio/video
  - Configuration files
  - Other *resources*
- Uses well-known ZIP compression format
- Syntax
  - Creating: `jar cvf` *`JARFileName File1 File2`*
  - Listing contents: `jar tvf` *`jar-file`*

Resources

Can you guess what A11Y stands for?

- Hardcoding any data that might change is poor style
- Configuration parameters
  - Property maps
  - Preferences API
- Data required by the application itself
  - Images
  - Strings for UI elements

Localization – "**adaptation** of a product, application or document content to meet the language, cultural and other requirements of a specific target market (a locale)".

Internationalization – "design and development of a product, application or document content that **enables** easy localization for target audiences that vary in culture, region, or language".

Quotes are from https://www.w3.org/International/questions/qa-i18n

Locating Resources

- Resources are associated with classes
- Class loader searches for the associated resource
- In the same directory where the class was found, if no directory name specified
- In a subdirectory of the directory where the class was found, if relative path is specified
- In a subdirectory of the class path inside the JAR file, if absolute path is specified

Preparing and Accessing Resources

- Get the Class object of the class that has a resource
- If the resource is an image or audio file
  - Call `getResource(filename)` to get the resource location as a URL
  - Read it with the `getImage` or `getAudioClip` method
- Otherwise
  - Use the `getResourceAsStream` method to read the data in the file

Manifest

- Describes special features of the archive
- Called MANIFEST.MF
- Placed in a special META-INF subdirectory
- Starts with
  - `Manifest-Version: 1.0`
- Entries are grouped into sections
- Sections
  - Main (first section in the file) applies to the whole JAR archive
  - Other sections
    - Must start with a `Name` entry
    - Specify properties of named entities such as individual files, packages, or URLs
  - Sections are separated by blank lines
- To include a manifest in the JAR file
  - `jar cvfm` *`JARFileName.jar ManifestFileName File1 File2`*

Creating and Deploying a J2SE Application

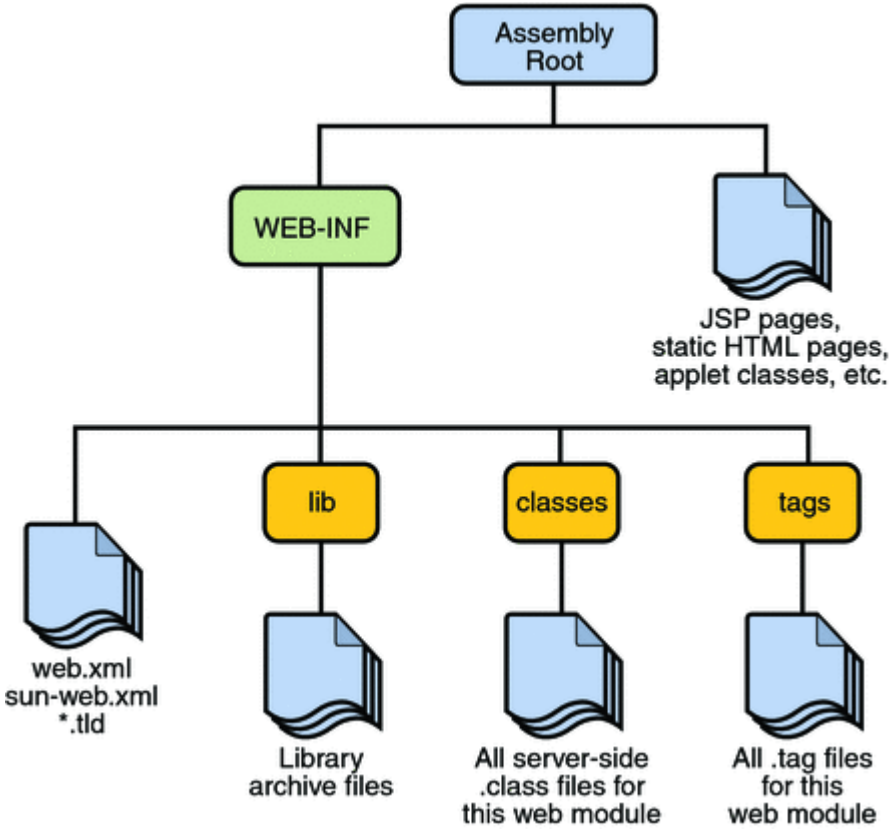
- Write your code (*.java)
- Prepare all other resources required by the application
  - *.txt
  - *.xml with data
  - *.csv
  - *.png, *.gif, *.jpg, etc.
  - *.mp3, *.mp4, etc.
  - *.xml with exported preferences
  - *. properties or other extension for a property map file
- Compile the source code into byte code (*.class) and move *.class files to appropriate directories
- Create a manifest file
- Pack all files into a JAR file
  - `jar cvfe JARFileName.jar MainClassName File1 File2`
- Deliver the JAR file to the customer
- Run the application
  - Double click on the JAR file icon (most systems)
  - `java -jar JARFileName.jar`

Web Application Resource or Web application ARchive (WAR)

- A more elegant way for deploying your application
- Created using the jar program
- Contains every file in an application
- The name of the WAR file is typically the same as the application name
- Tomcat automatically unpacks WAR files and deploys the application

Directory Structure of a WAR File

- `WEB-INF`
  - Meta information directory
  - Located just below the Web app root directory
  - Files stored here are not supposed to be accessible from a browser
- `web.xml` file contains information about the Web application
- `classes` directory contains all compiled Java classes that are part of your web application
- `lib` directory contains all JAR files used by the Web application



Creating and Deploying a Web Application

- Create all necessary files and place them in the appropriate directories
  - Servlets
  - JSP files
  - Any resources used by your application
  - Compiled classes in `classes`
  - JAR libraries in `lib`
  - web.xml directly under `WEB-INF`
- Test your application on the development system
- Pack all contents using the `jar` command
  - `jar cvf` *`WARFileName.war`* `*`
- Place *`WARFileName.war`* directly under `webapps` in the Tomcat directory
- If Tomcat is running, it will pick up the new application automatically
- Call your Web application from a Web browser

//TODO before next lecture:

- Homework 4 due on 4/16 at 11:59 pm EDT. Must be submitted on Submitty.