CS3220 Web and Internet Programming

Introduction to Java Servlets

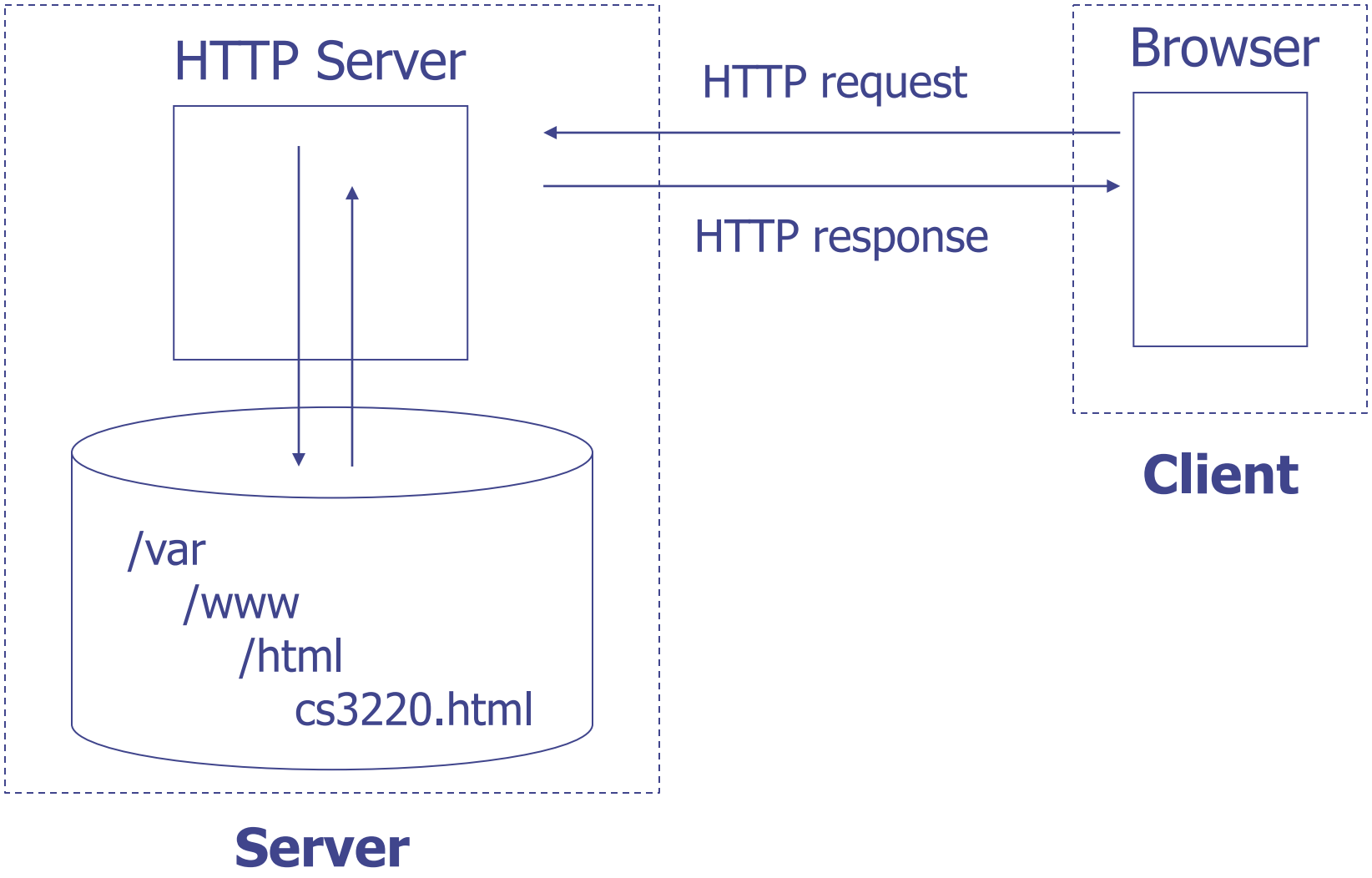
Chengyu Sun

California State University, Los Angeles

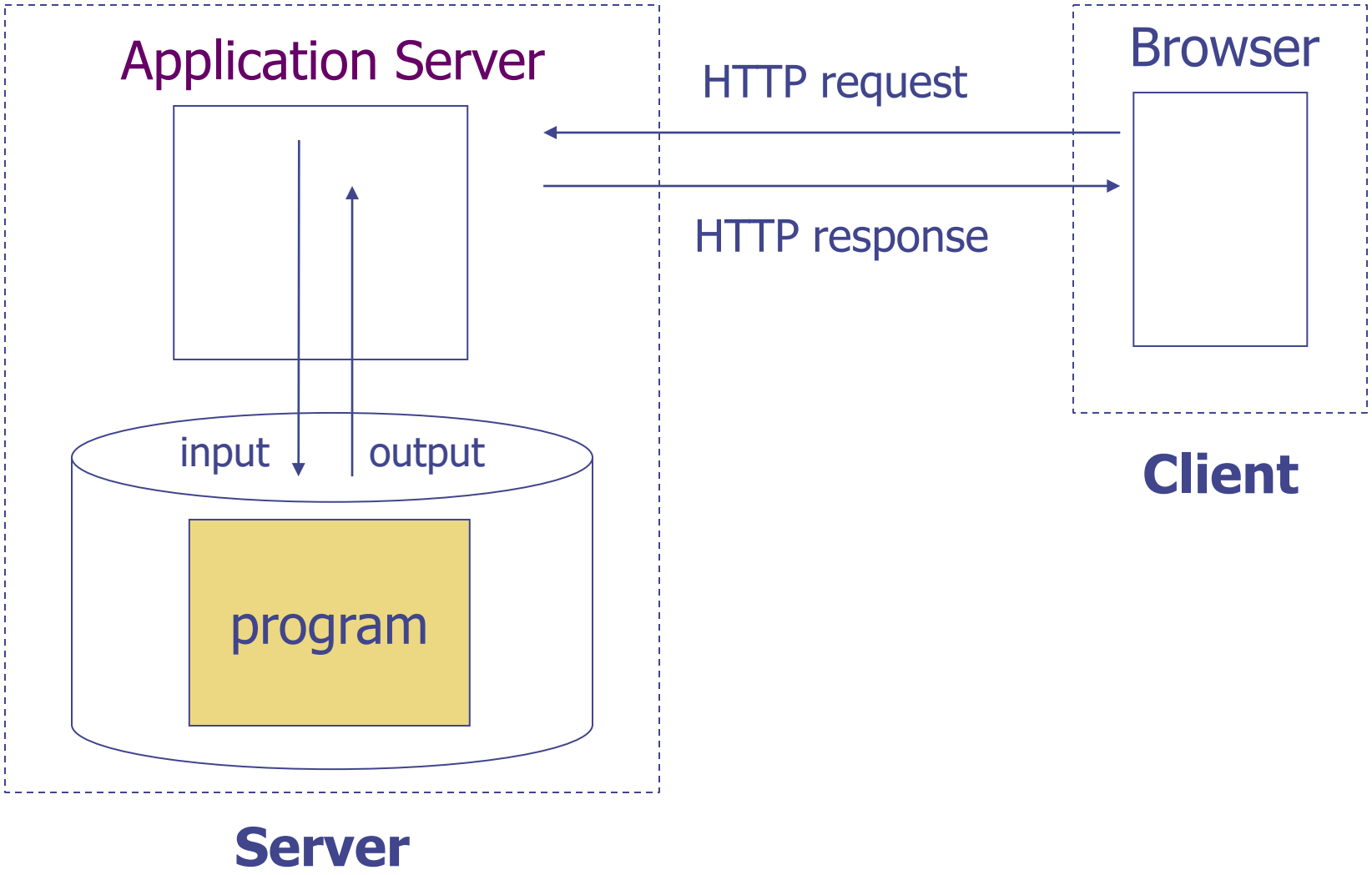
Set Up Development Environment

- *Web Development with Eclipse and Tomcat* on Canvas
- Accompanied video on YouTube

Static Web


HTTP Server
HTTP request
Browser
HTTP response
/var
/www
/html
cs3220.html
Client
Server

Create Dynamic Content



Servlet HelloWorld

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet( "/HelloWorld" )
public class HelloWorld extends HttpServlet {
    public void doGet( HttpServletRequest request,
                       HttpServletResponse response )
        throws ServletExceptoin, IOException
    {
        PrintWriter out = response.getWriter();
        out.println( "Hello World" );
    }
}
```

Some Simple Observations

- Inherits from HttpServlet
  - http://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html
  - There's no `main()` method
- doGet()
  - Input: HttpServletRequest
  - Output: HttpServletResponse → sent back to the client browser

Example: HelloWorld in HTML

- Modify the `HelloWorld` servlet to output in HTML
  - Check the source of the generated HTML in a browser

Generating HTML

- HttpServletResponse
- Set content type to "text/html"
  - setContentType()
  - Common MIME types
- Generate an HTML page
  - getWriter().println()
    - <html>, <head>, <body> ...

Servlet Mapping

- @WebServlet(<URL Pattern(s)>)

Java Annotations

- Available since JDK 1.5 (Java 5)
- Data about a program that is not part of the program itself
- Can be used by compiler, VM, and other software tools for *various purposes*

Annotation Examples ...

- Error detection

```
@Override
protected void doGet()
```

- Suppress warning

```
@SuppressWarnings("unchecked")
public List<User> getAllUsers()
{
    return (List<User>) new ArrayList();
}
```

… Annotation Examples

- Servlet mapping in Sevelet 3.x Specification

```
@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet
```

- Web service

```
@WebService
public class HashService {

    @WebMethod
    public String md5( String text )
}
```

About Annotations

- An annotation may have *elements* (like attributes of HTML tags)
- The default element is `value`
- An element has a type (like a variable in Java)
- `{ }` can be omitted for array values if there's only one value in the array

@WebServlet

- http://docs.oracle.com/javaee/7/api/javax/servlet/annotation/WebServlet.html

@WebServlet Elements for URL Patterns

- `value`
  - URL pattern(s) of the servlet
  - The default element
- `urlPatterns`
  - Same purpose as `value`
  - Usually used when more than one element is specified
  - Only one of `value` and `urlPatterns` can be specified

@WebServlet Examples

```
@WebServlet( "/HelloServlet" )

@WebServlet( {"/HelloServlet", "/member/*"} )

@WebServlet( name="Hello", urlPatterns={"/HelloServlet", "/*.html"} )

@WebServlet(
        urlPatterns="/MyPattern",
        initParams={@WebInitParam(name="ccc", value="333")}
)
```

Wildcard in Servlet Mapping

- A string beginning with a **/** and ending with a **/***
  - E.g. `/*`, `/content/*`
- A string beginning with a ***.**
  - E.g. `*.html`, `*.do`

See Servlet Specification 3.0, Section 12

Be Careful with URL Patterns

- Invalid patterns
  - E.g. `/member/*.html`, or `member/index.html`
- Conflicting patterns
  - E.g. two `/HelloServlet`
- Overlapping patterns
  - E.g. `*.html` and `/member/*`

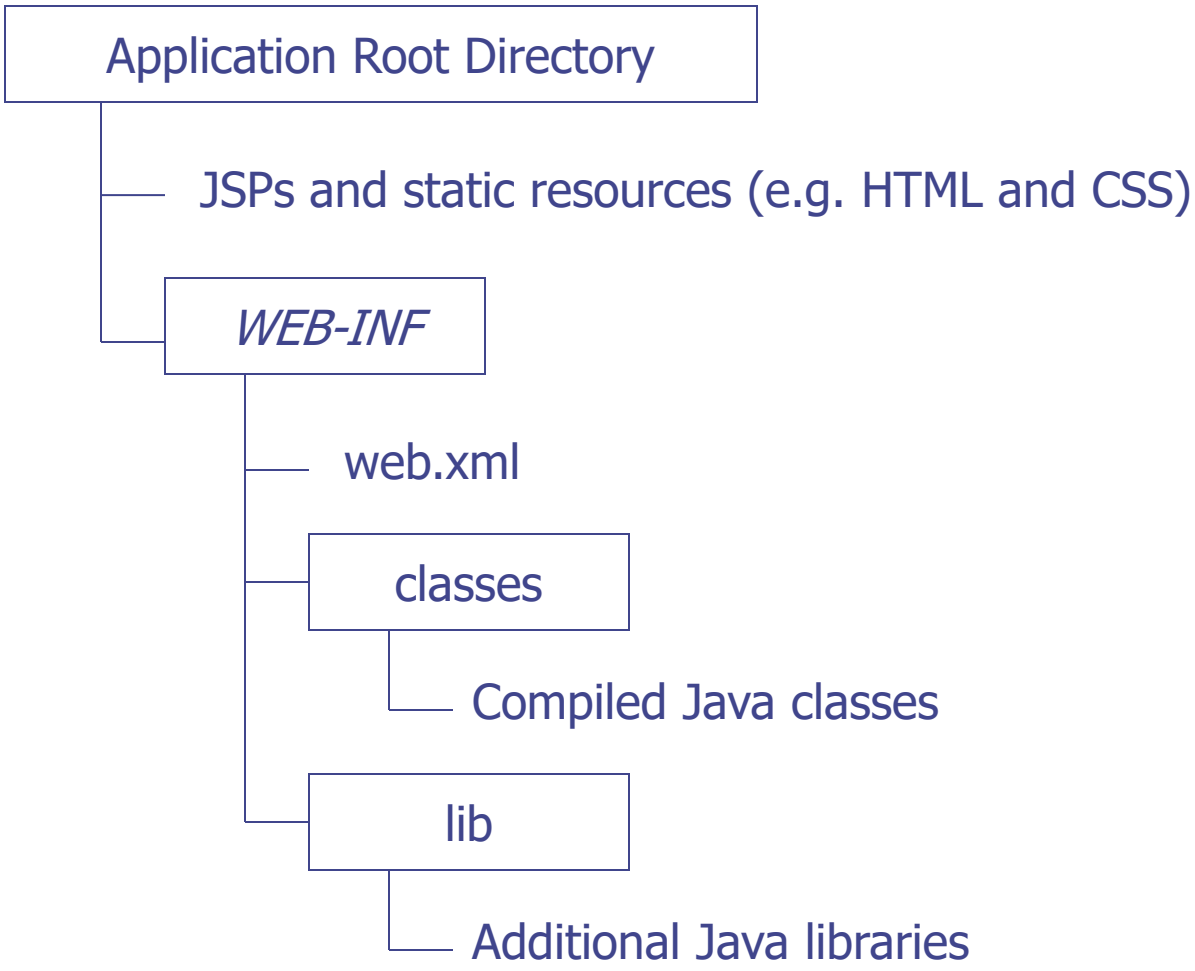
Deploy to a Server

- Understand the directory structure
- Transfer files to the right folder

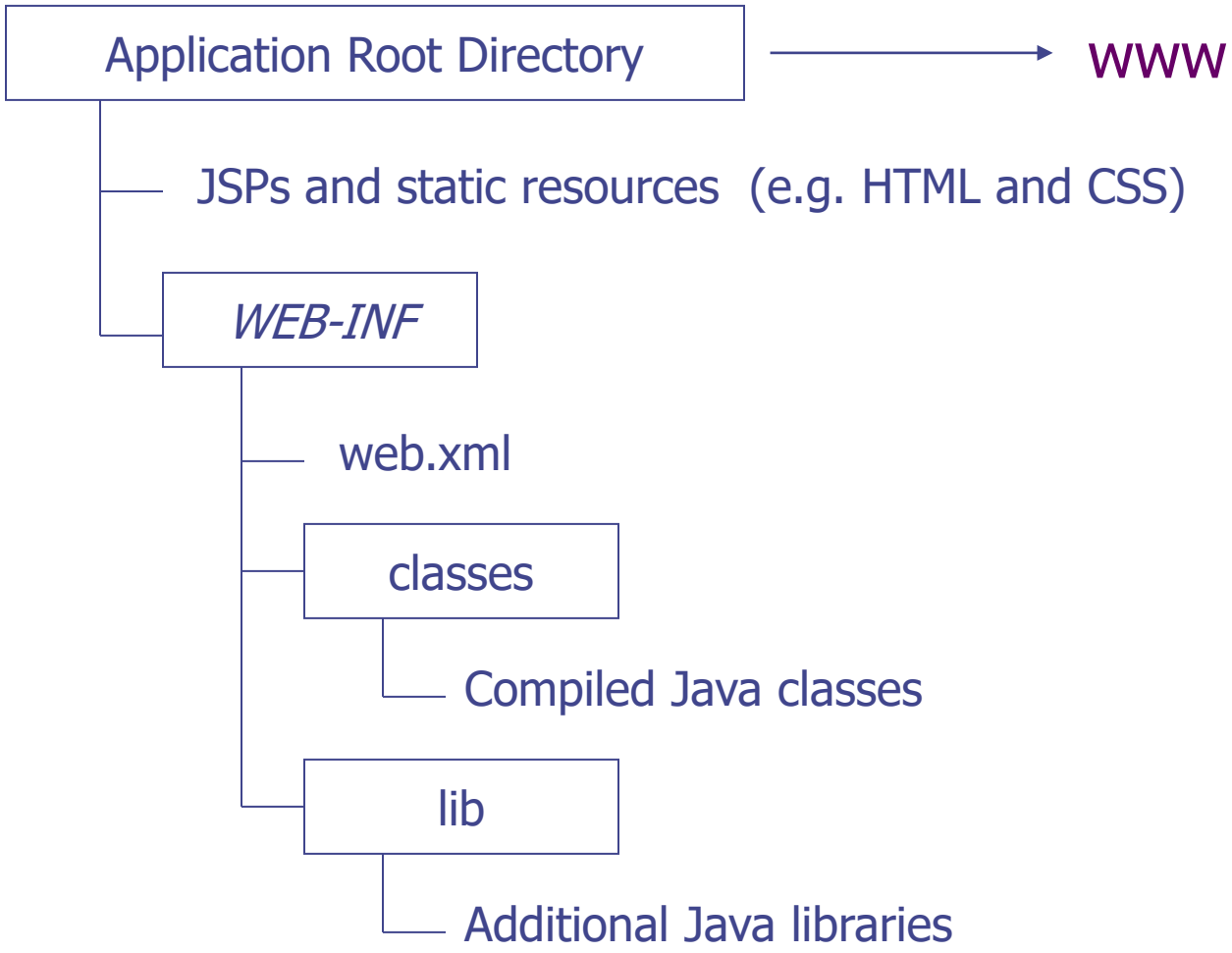
Java Web Application Components

- Compiled Java classes (.class files)
  - Servlets, beans, filters, ...
- Addtional Java libraries (.jar files)
- JavaServer Pages (JSPs)
- Static resources
  - HTML, CSS, images, ...
- Metadata files
  - web.xml, ...

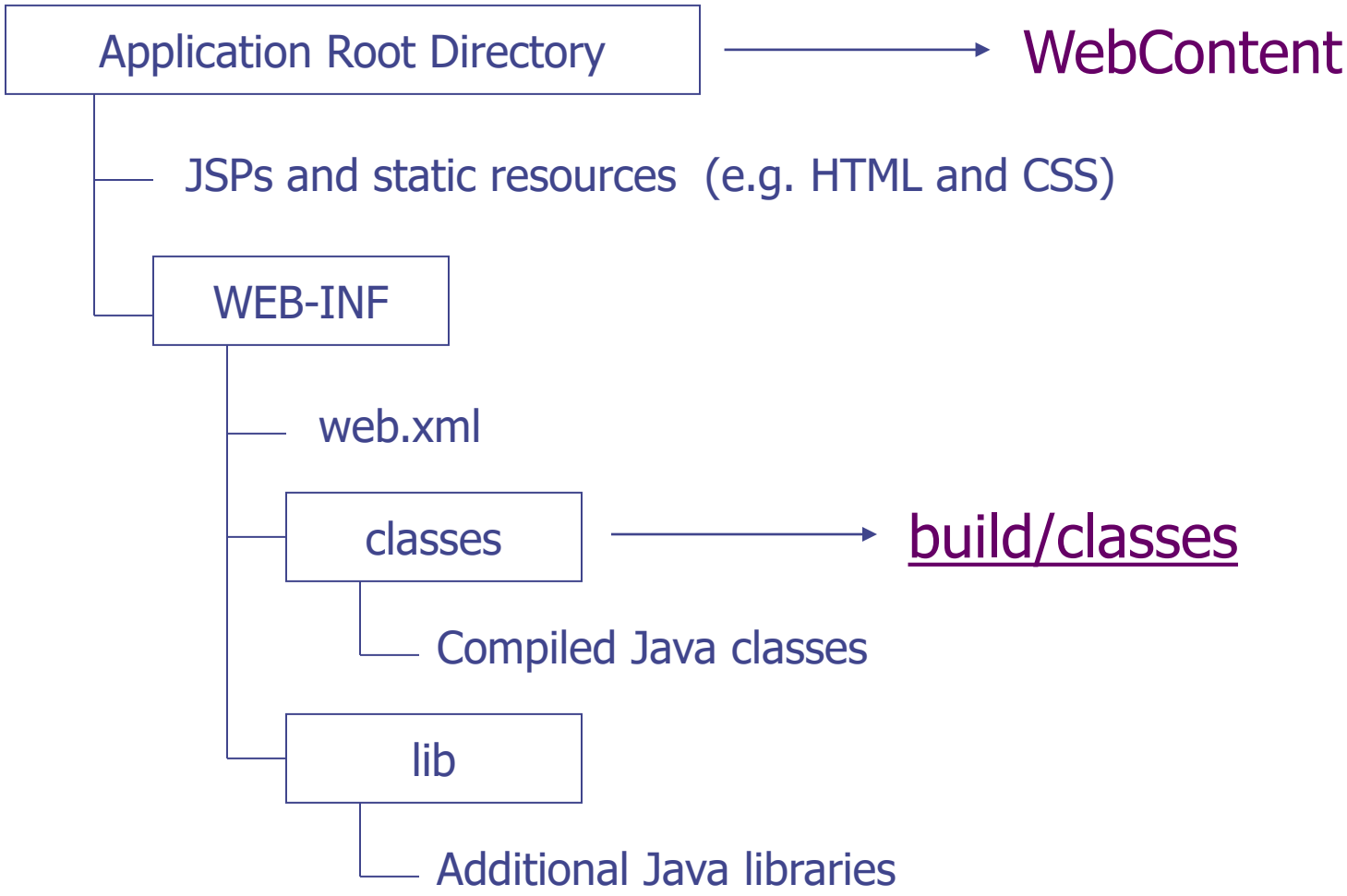
Directory Structure of a Java Web Application



Directory Structure on CS3



Directory Structure of an Eclipse Dynamic Web Project



About web.xml

- Web application deployment descriptor
- Not required after Servlet 3.0
- Useful for "notifying" Tomcat on CS3 that your project has been updated
  - Reload web.xml or simply updating its timestamp using the "touch" command

What's Next

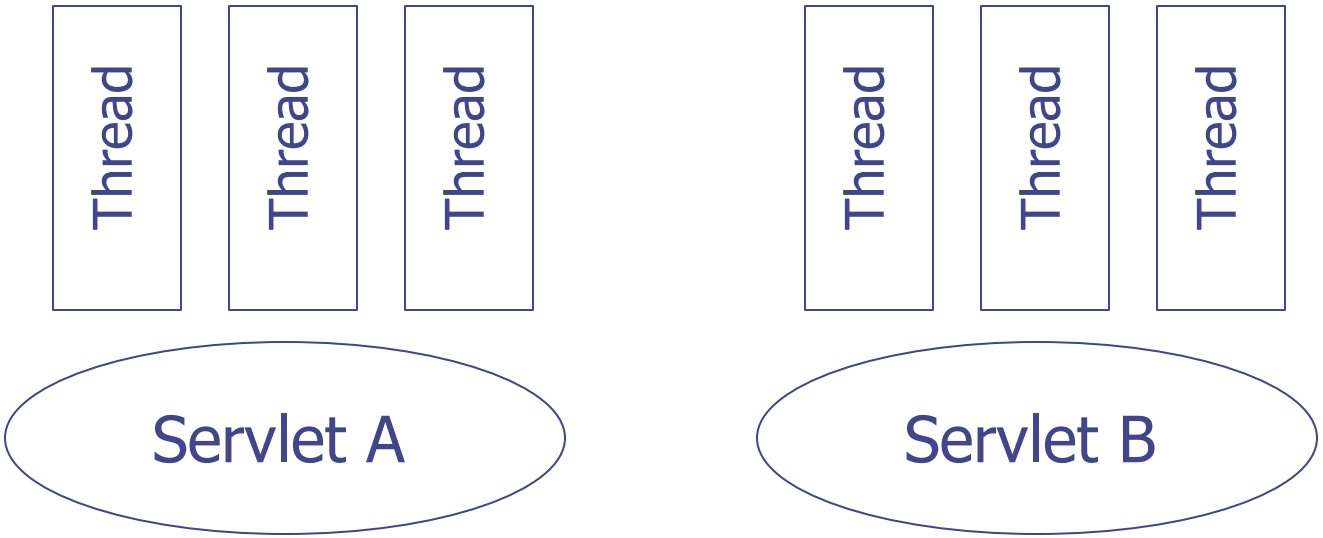
- Keeping data in a servlet
- Sharing data among servlets

Example: RequestCounter

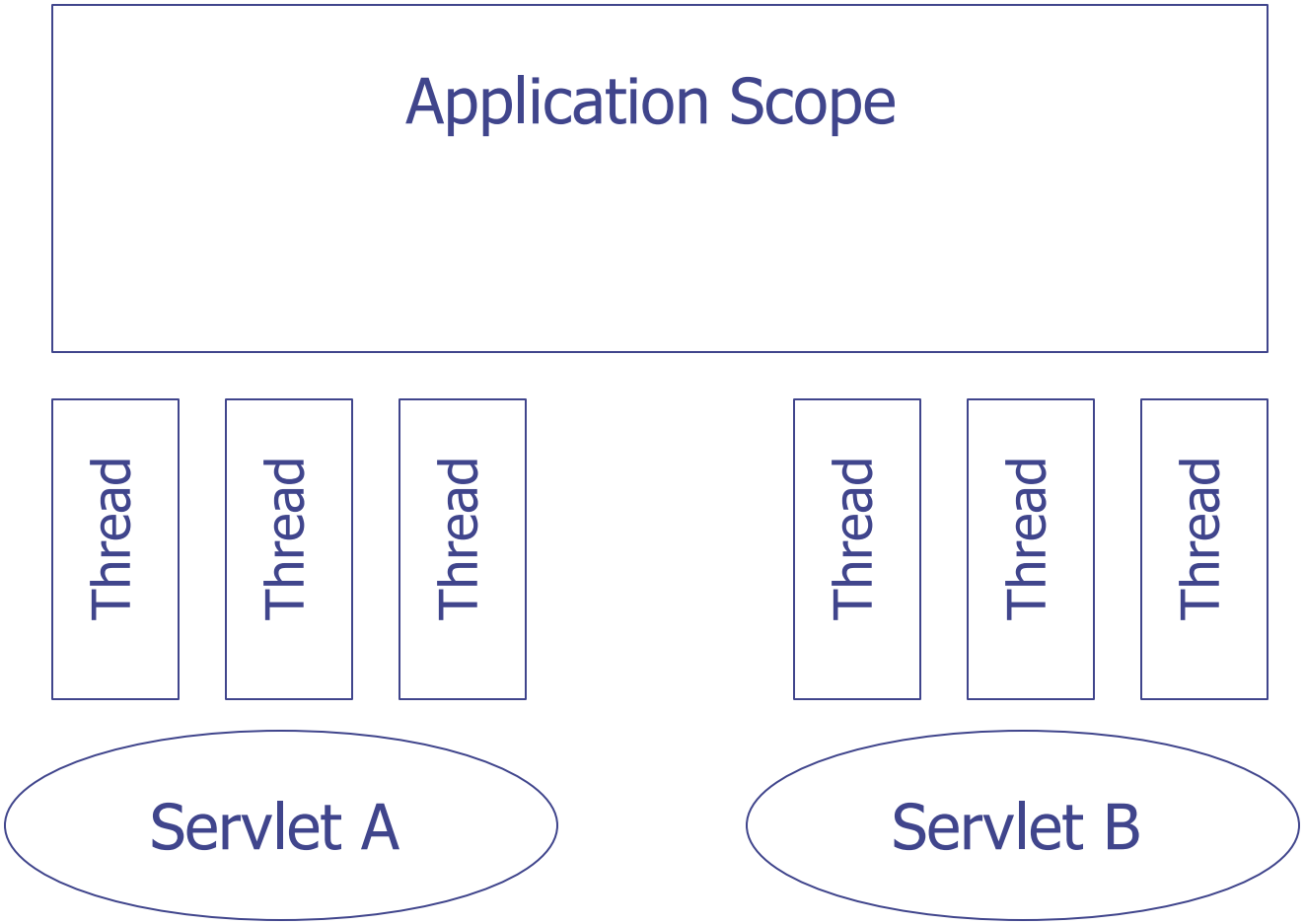
- Display the number of times a servlet is requested
  - Keep the count as an instance variable

Problem with Instance Variables

- Most web applications have multiple servlets working on the same data
- Instance variables of a servlet cannot be shared with other servlets



Application Scope ...



... Application Scope

- A "storage area" for sharing data among all servlets
- Data in application scope will remain there as long as the application is running

Access Application Scope

- HttpServlet
  - getServletContext()
- HttpServletContext
  - setAttribute(String name, Object value)
    - Give any object a name and save it to application scope
  - getAttribute(String name)
    - Retrieve the object from application scope

Example: SharedCounter

- Keep the request counter in application scope

Servlet Life Cycle

- When the servlet is loaded – `init()`
  - Executed only once
  - Don't forget `super.init(config)`
- Per request – `service()`
  - dispatch to `doXxx()`
- When the servlet is unloaded – `destroy()`

Why Use init() Instead of Constructor

- Because `ServletContext` is not available in constructor

Example: DisplayCounter

- Use one servlet to count the number of requests, and another servlet to display the count

loadOnStartup

- By default, a servlet is not created until it is accessed for the first time
  - Could cause problem if one servlet must run before another servlet
- Use the `loadOnStartup` element of `@WebServlet` to have a servlet created during application startup

loadOnStartup Example

```
@WebServlet(
        name="Hello",
        urlPatterns={"/HelloServlet", "/*.html"},
        loadOnStartup=1
)
```

The value for `loadOnStartup` is the order in which the application server will start the servlets.

Debugging Servlets

- 404 Errors: check URL and URL mapping
- Display Problems: check the source of the generated HTML
  - `View Source` in browser, or
  - Use an HTML Validator
- Logical errors: use the Eclipse debugger
  - Set *break points*
  - `Debug As` → `Debug on Server`