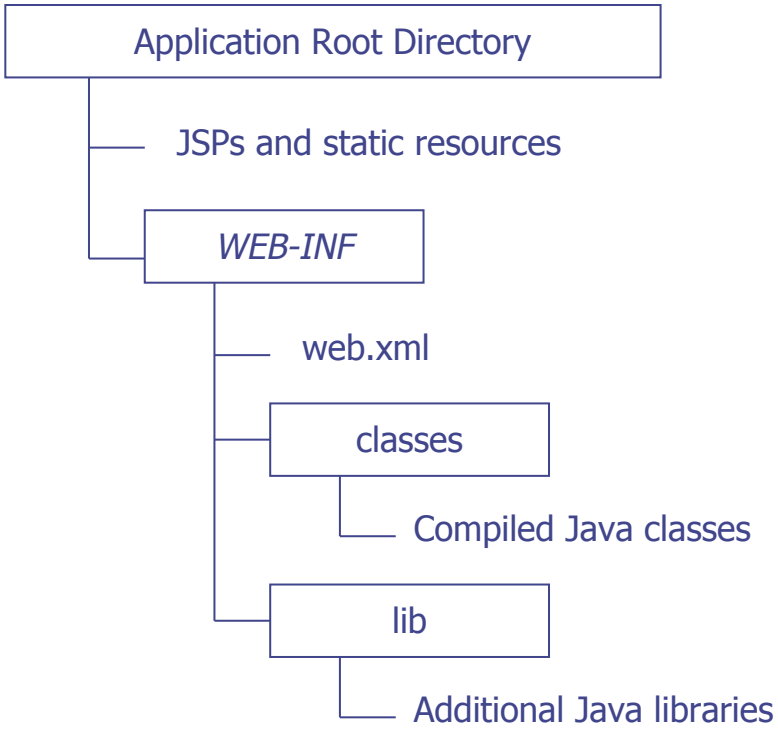
# Introduction to Java Servlets

ThS. Đặng Thị Kim Giao
Khoa CNTT - ĐH SPKT TP.HCM

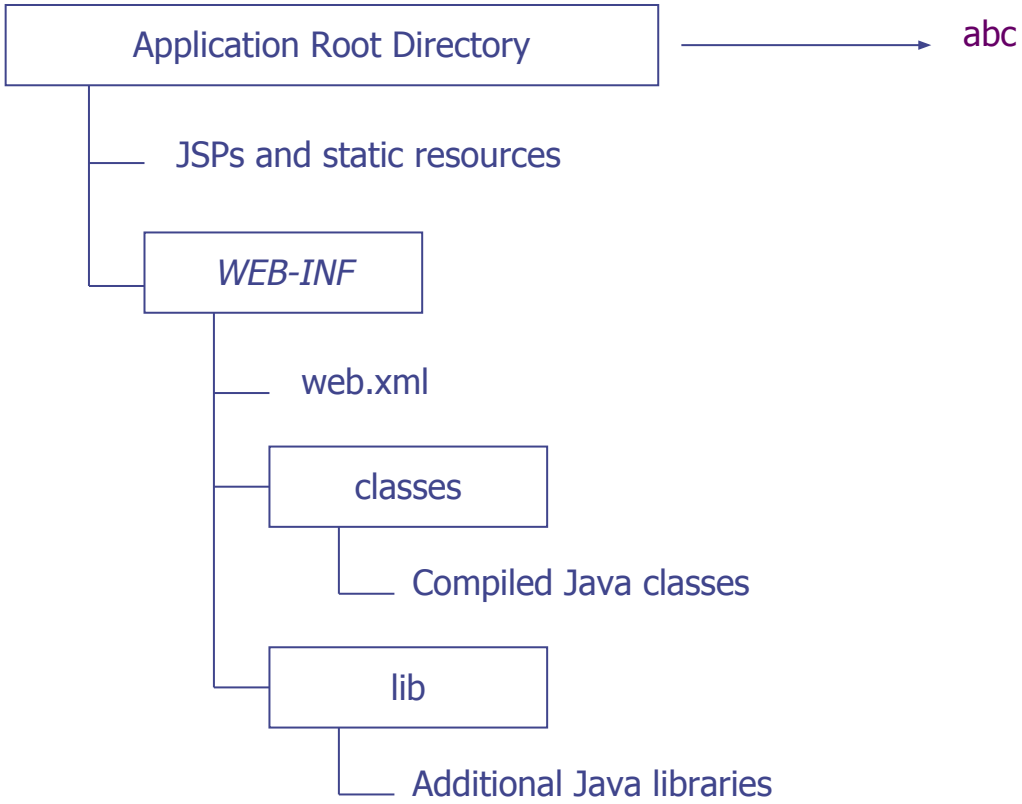
# Java Web Application Components

- Compiled Java classes (.class files)
- Servlets, beans, filters, ...
  - Bean: a data access object which is used to interact with the database. Java bean is a POJO (Plain Old Java Object), not a servlet.
  - Filter: a class implementing interface Filter and performing many different types of filtering functions on requests and/or responses. [see more]
- Additional Java libraries (.jar files)
- JavaServer Pages (JSPs)
- Static resources
  - HTML, CSS, images, ...
- Metadata files
  - web.xml, ...

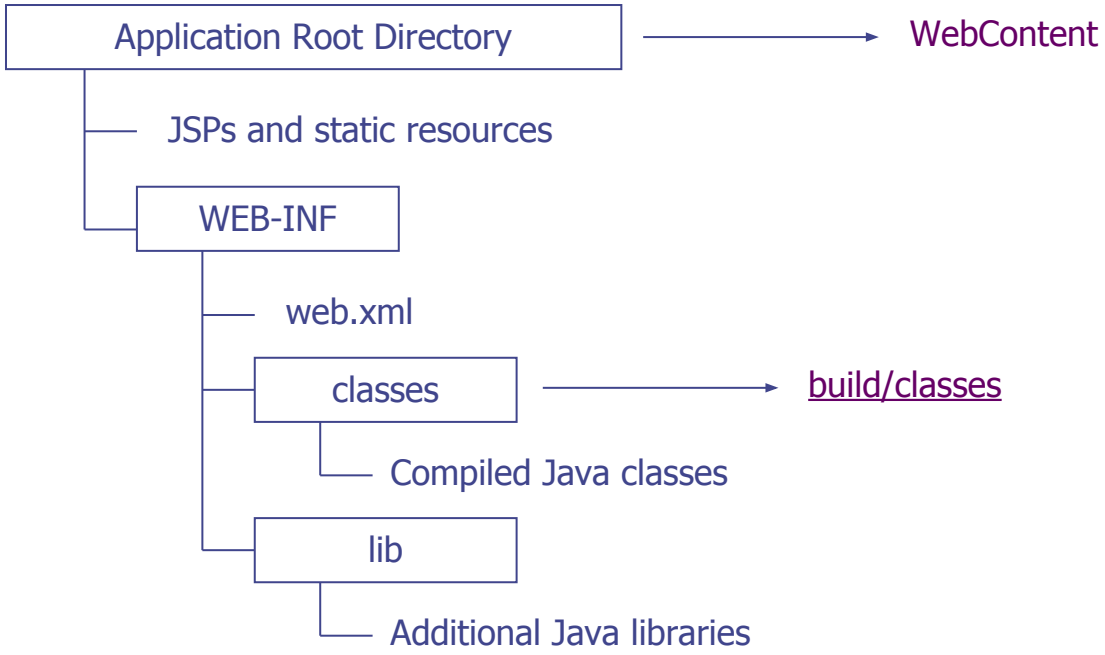
# Directory Structure of a Java Web Application



# Directory Structure on Sample Server



# Directory Structure of an Eclipse Dynamic Web Project



# 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 ServletException, IOException
    {
        PrintWriter out = response.getWriter();
        out.println( “Hello World” );
    }
}
```

# Some Simple Observations

- Servlet inherits from **HttpServlet**
  - 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

```
public void doGet( HttpServletRequest request,
                   HttpServletResponse response )
        throws ServletExceptoin, IOException
    {
        PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    out.println("<html><head><title>Hello</title></head>");
    out.println("<body><h2>Hello World!</h2></body></html>");
    }
```

# Generating HTML

- HttpServletResponse
- Set content type to “text/html”
  - `setContentType()`
  - Browser doesn't need to guess the content type
- Generate an HTML page
  - `getWriter().println()`
    - `<html>, <head>, <body>` …

# Servlet Mapping

- A java web application includes many servlets. Servlet mapping help knowing which servlet is used to handle a request.

- Servlet mapping takes the <path> component of the request url
  - `@WebServlet(<URL Pattern(s)>)`



- Look at following request url:

  `http://<host>/<app>/<path>`

  `http://localhost/jwd/HelloWorld`

# 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 (declare servlet url pattern)

```
@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
  - E.g. `@WebServlet(value={"/HelloServlet"})// this annotation has 1 //element: the value element`

- An element has a type (like a variable in Java)
  - element type can be some simple type like number, string, array
  - E.g. annotation `@WebServlet(value={"/HelloServlet"})` has string array

# About Annotations

- The default element is `value`
  - E.g. `@WebServlet({"/HelloServlet"})// element name 'value' is omitted.`

- `{}` can be omitted for array values if there's only one value in the array
  - E.g. `@WebServlet("/HelloServlet")//array braces can be omitted.`
  - =>why web servlet annotation look like this: **`@WebServlet("/HelloServlet")`**

# @WebServlet

- http://download.oracle.com/javaee/6/api/javax/servlet/annotation/WebServlet.html

| Optional Element Summary | |
|---|---|
| boolean | **asyncSupported**<br>Declares whether the servlet supports asynchronous operation mode. |
| java.lang.String | **description**<br>The description of the servlet |
| java.lang.String | **displayName**<br>The display name of the servlet |
| WebInitParam[] | **initParams**<br>The init parameters of the servlet |
| java.lang.String | **largeIcon**<br>The large-icon of the servlet |
| int | **loadOnStartup**<br>The load-on-startup order of the servlet |
| java.lang.String | **name**<br>The name of the servlet |
| java.lang.String | **smallIcon**<br>The small-icon of the servlet |
| java.lang.String[] | **urlPatterns**<br>The URL patterns of the servlet |
| java.lang.String[] | **value**<br>The URL patterns of the servlet |

# @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”, “/xx”, “/yy” })

@WebServlet( {“/HelloServlet”, “/member/*”} )
```

# @WebServlet Examples

```
@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`

# Be Careful with URL Patterns

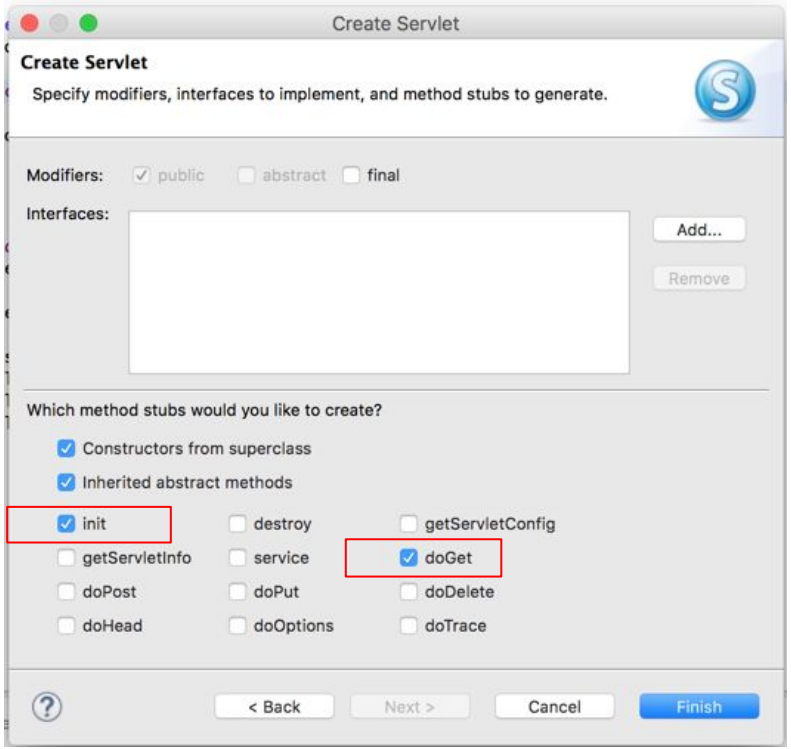
- Invalid patterns
  - E.g. `/member/*.html,` or `member/index.html`
- Conflicting patterns (>= servlets are mapped to same url)
  - E.g. two `/HelloServlet`
- Overlapping patterns
  - E.g. `*.html` and `/member/*`
    - how if adding one more pattern: `/member/a.html` (?)

# Example: RequestCounter

- Display the number of times a servlet is requested

# Example: RequestCounter

- Let's new a servlet
- Choose override:
  - **init**() method
  - **doGet**() method



# ... init() method: to initialize the counter value

- Let's look at the code
- Remember to call `super.init()`, otherwise,
  the `ServletContext` object
  cannot be initialized => **problem**

```
1  package jwd.servlet;
2
3  import java.io.IOException;
10
11
12 @WebServlet("/RequestCounter")
13 public class RequestCounter extends HttpServlet {
14
15     private static final long serialVersionUID = 1L;
16     int counter;
17
18     public RequestCounter() {
19         super();
20     }
21
22
23     public void init(ServletConfig config) throws ServletException {
24         super.init(config);
25         counter = 0;
26     }
27
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Servlet
29
30     }
31
32 }
```

# ... doGet() method

- Let's look at the code
-

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
    ++counter;
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>RequestCounter</title></head>
<body>");
    out.println("You are visitor #" + counter + ".");
    out.println("</body></html>");
}
```

# 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()`
  - `release resource used by the servlet`

# Why Use `init()` Instead of Constructor

- Historical reasons – see http://csns.calstatela.edu/wiki/content/cysun/notes/servlet_data_init
- `ServletContext` cannot be accessed in a constructor

# Example: SharedRequestCounter

- Use one servlet to count the number of requests, and another servlet to display the count
  - One servlet calculate the counter, one servlet displays the counter.

# … Can we refer to variable in another servlet?

- Can we refer to the counter variable in RequestCounter servlet?
  - No! Because:
  - 2 servlet are 2 different program!

The solution:

Application Scope.

```
13 @WebServlet("/RequestCounter")
14 public class RequestCounter extends HttpServlet {
15
16     private static final long serialVersionUID = 1L;
17     int counter;
18
19     public RequestCounter() {
20         super();
21
22     }
23     public void init(ServletConfig config) throws ServletException {
24
25         super.init(config);
26         counter = 0;
27     }
28     protected void doGet(HttpServletRequest request, HttpServletResponse response)
29             throws ServletException, IOException {
30         ++counter;
31         PrintWriter out = response.getWriter();
32         out.println("<html><head><title>RequestCounter</title></head><body>");
33         out.println("You are visitor #" + counter + ".");
34         out.println("</body></html>");
35     }
36 }
```

# Application Scope

- A “storage area” where data can stored and accessed
- Data in application scope will remain there as long as the application is running
- Data in application scope is shared by all servlets

# 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

# Code of RequestCounter Servlet

- 

```
public void init(ServletConfig config) throws ServletException {

    super.init(config);

    Integer counter = 0;
    ServletContext context = getServletContext();
    context.setAttribute("Counter", counter);

}
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    // get counter from servelet context
    ServletContext context = getServletContext();
    Integer counter = (Integer)context.getAttribute("Counter");

    // increase counter value
    counter++;

    // save the counter into Servlet Context
    context.setAttribute("Counter", counter);


    PrintWriter out = response.getWriter();
    out.println("<html><head><title>RequestCounter</title></head><body>");
    out.println("The counter has been increased .");
    out.println("</body></html>");

}
```

# Code of DisplayCounter Servlet

- We just edit doGet() of DisplayCounter servlet

```
protected void doGet( HttpServletRequest request,
    HttpServletResponse response ) throws ServletException, IOException
{
    ServletContext servletContext = getServletContext();

    // get the counter
    Integer counter = (Integer) servletContext.getAttribute( "Counter" );

    // display the message "the counter is incremented"
    PrintWriter out = response.getWriter();

    response.setContentType( "text/html" );
    out.println( "<html><head><title>Display Counter</title></head><body>" );
    out.println( "<p>The counter value is currently: " + counter + "</p>" );
    out.println( "</body></html>" );
}
```

# `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 (without any request)
- Pay attention to conflict `loadOnStartup` value (2 servlets have same value of `loadOnStartup`)

# About `web.xml`

- Web application deployment descriptor
  - `<web-app>`
    - Version
  - `<welcome-file-list>`
- More about `web.xml` in Java Servlet Specification

# Versions

| Servlet/JSP Spec | Tomcat | Java |
|---|---|---|
| 3.1/2.3 | 8.0.x | 1.7 |
| 3.0/2.2 | 7.0.x | 1.6 |
| 2.5/2.1 | 6.0.x | 1.5 |
| 2.4/2.0 | 5.5.x | 1.4 |

The `version` attribute of `<web-app>` in web.xml

# Debugging Servlets

- Read error message carefully
- Using the Eclipse debugger
  - Set break points
  - **`Debug As → Debug on Server`**
- View the source of the generated HTML
  - **`View Source`** in browser
    - Validation
      - http://validator.w3.org/
      - Use the Web Developer addon of Firefox

# `loadOnStartup` Example

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

`loadOnStartup` can specify an (optional) integer value.

If `loadOnStartup >=` 0, it indicates *an order for servlets to be loaded*, servlets with higher numbers get loaded after servlets with lower numbers.

# Exercise

-
-