Deployment

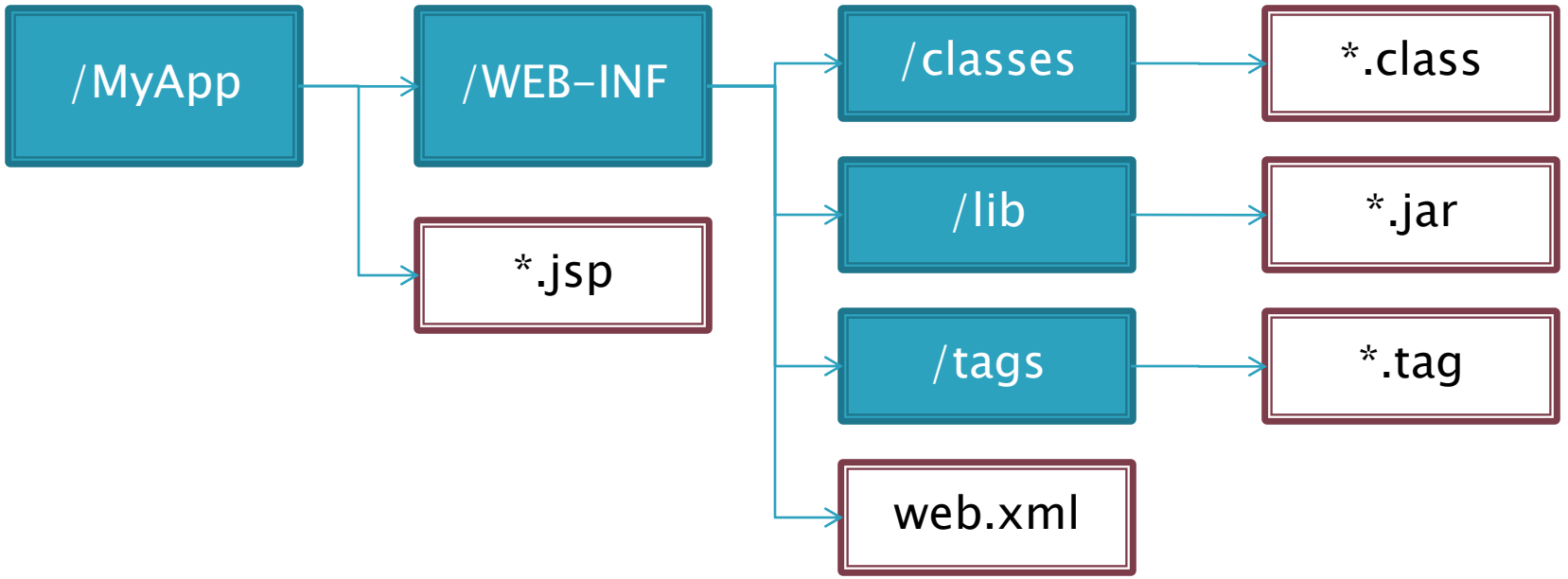
The complete web app

EHSDI

eBuzima

Directory structure

- All Java web apps have to have the same basic directory structure



Libraries

- Some libraries (`servlet-api.jar` etc) are part the JSP spec and thus part of the container distribution
  - This is why we can use the jar files in the Tomcat lib folder
  - We should not put additional copies of these jar files into our web app
- Other libraries need to placed in WEB-INF\lib – this is where the container expects to find them

WAR files

- A WAR file is essentially a JAR file containing a web app
- It has an additional directory called META-INF
- This contains the MANIFEST.MF file which can be used to specify library dependencies

WEB-INF and META-INF

- Both of these directories are not directly accessible – i.e. their content is hidden from the client
- This means we can protect JSPs from direct access by putting them inside one of those folders

Example

- MyServlet forwards to a JSP file in WEB-INF

```
request.getRequestDispatcher("WEB-INF/home.jsp")
  .forward(request, response);
```

- ... which works because it's being accessed through the servlet
- But a user cannot request it directly in their browser

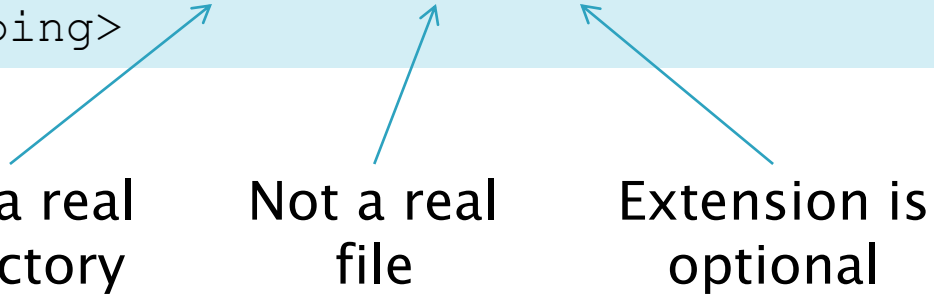
```
http://localhost:8080/mywebapp/WEB-INF/home.jsp
```

404 NOT FOUND

URL mappings

- The container can map any URL to any servlet, i.e. the URL doesn't have to correspond to a real file in a real folder

```
<servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/test/users/test.do</url-pattern>
</servlet-mapping>
```



Partial URL matches

- The URL pattern can include wildcards
- We can match all requests for items in a specific directory, e.g.

```
<url-pattern>/test/users/*</url-pattern>
```

- We can match all requests with a specific extension, e.g.

```
<url-pattern>*.htm</url-pattern>
```

Duplicate URL matches

- What if two URL mappings match a request?
- For example: if the user requests `/test/index.htm` which of the following mappings will be matched:

```
<url-pattern>*.htm</url-pattern>
```

```
<url-pattern>/test/*</url-pattern>
```

- **Answer**: `/test/*` because the container always chooses the longest mapping

Welcome files

- When a user tries to access a directory, the server can be configured to return a specific "welcome" file instead, e.g.

- We request http://localhost:8080/openmrs which is a directory (the root directory of the OpenMRS web app)
- The server actually returns http://localhost:8080/openmrs/index.htm

a welcome file

Welcome files

- The DD can be configured with multiple welcome pages, e.g.

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
```

- When you try to access a directory like "test", the container will now check for:
  1. Any matching URL mappings
  2. `test/index.html`
  3. `test/default.jsp`

names like **index** and **default** are just convention

Error pages

- We can also specify the default pages for when something goes wrong
- We can return a specific page for a specific exception, e.g.

```
<error-page>
  <exception-type>java.lang.NullPointerException</exception-type>
  <location>errorNull.jsp</location>
</error-page>
```

OpenMRS does this to catch authentication exceptions and send the user to the login page

Error pages

- The other use is to handle specific HTTP error codes

```
<error-page>
  <error-code>404</error-code>
  <location>errorNotFound.jsp</location>
</error-page>
```

```
<error-page>
  <error-code>403</error-code>
  <location>errorForbidden.jsp</location>
</error-page>
```

See http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Servlet initialization

- By default, servlets are initialized on the first request, i.e. the `init` method isn't called until a client makes a request
- This may mean that the first client has to wait a long time
- Sometimes its preferable to have a servlet initialize when it's deployed:

```
<servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-class>test.TestServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

References

- Books
  - Head First Servlets and JSP (O'Reilly)
- Websites
  - http://java.sun.com/javaee/reference/tutorials/