Web Applications Intro

BWA Slides #2

Web Application Basic Setup

It's a client/server <u>stateless pull</u> architecture

Client (the browser) sends HTTP-requests and handles HTTP-responses

- Text based protocol (binary payload possible, images, ...)

Server can handle HTTP-requests and send HTTP-responses (and more ...)

Payload in responses: **Hyper text markup language** (HTML), **Cascading Style sheets** (CSS), **JavaScript**, ...

Payload Responsibilities

HTML, ("the page") the structure. Description of a tree, the **Document Object Model** (HTML DOM) -tree.

- Nodes in DOM-tree are objects i.e. have API

CSS, the rendering of the DOM-tree (a 2D view)

JavaScript, to manipulate the DOM-tree (and more)

- Beside core language, many API's
- Most famous (herostratic): JavaScript DOM API

The Browser

Programmers view of browser;

An in memory DOM-tree (specified by the HTML)

A layout engine (processing the CSS)

A JavaScript engine (interpreting the JavaScript code)

Browsers and Engines

Browser	**Layout engine**	**JavaScript engine**
Chrome	WebKit	V8
Internet Explorer	Trident	Chakra (IE 9)
Firefox	Gecko	Spider Monkey/Trace Monkey/Jäger Monkey
Safari	WebKit	SquirrelFish (marked as Nitro)

Browsers and Standards

Browser should follow web standards

- ...but not all do (getting better, ... worst ever IE 6)
- No guarantees that everything works in all browsers

Code samples tested on Chrome (and to some extent Firefox) http://html5test.com/

The Standards

- HTTP 1.1, RFC 2616 (HTTP 2.0 round the corner..?)
- HTML 4.01, http://www.w3.org/TR/html401/
- HTML 5 (5.1) drafts http://www.w3.org/html/wg/drafts/html/master/
- DOM http://www.w3.org/TR/domcore/
- CSS .... puhiiii.... http://www.w3.org/standards/techs/css
- ECMA Script ed. 5.1 (a formalization of JavaScript core, no browser objects) http://www.ecma-international.org/publications/standards/Ecma-262.htm

Standards in Course

We're in a transition from HTML 4.01/CSS 2.1 to HTML 5/CSS 3, we'll try to use: **HTML 5/CSS 3**

- In fact we prefer **XHTML5** upcoming...

We use **ECMA Script 5.1** (JavaScript also has versions 1.3, 1.4... 1.6, confusing)

- Supported by all modern browsers

Also a lot of other specifications in background, ...

The Server Side

Client side is any browser (hopefully adhering to the standards)

Server side

- If just serving **static content** (HTML, CSS, JavaScript, ...) no problems, "any" web server will do ...
- But we're not! We need to serve **dynamic content**, i.e. need **server side processing.** Typically content from database

Need to decide on **platform** and possible **framework**...(senseless to implements everything from scratch)

Platforms and Frameworks

A platform is a (huge) "ecosystem" for building/ running software

- Win/.NET/
- Android/Linux/Java
- LAMP: Linux/Apache/MySql/PHP
- iOS/iPhone

A framework is a generic structure for a special type of application. Frameworks are built on top of platforms (some call an API for a framework)

- Game frameworks
- Web app frameworks. Which to choose (can't test them all...)?????

http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

Selecting a Platform

In reality, great many aspects of selecting platform/ framework, see Web

For this course

- Should be Java based (student educational background)
- Should be platform independent (students have different machines)
- Should be open source/free (students have no money, ...)
- Preferably some standard... .. boils down to

Platform **Java Enterprise Edition 6 (JEE6)**

- Framework: None or possible 0,5 (too much time to learn)

Java EE 6 vs Java EE 7

Right now Java EE is in a transition from version 6 to 7

This course uses Java EE 6 (probably Java EE 7 next year)
A lot on the web about Java EE 7, watch out ...

Java Enterprise Edition 6

JEE 6 is an umbrella specification for web/enterprise applications

- Specifies a number of API's (each in turn specified by ...)
- **Java Specification Requests, JSR's**
    - JSR's are numbered and have named reference implementations: **Tomcat, Metro, Mojarra, Jersey, Weld**, ...
- Specifies runtime environment for the server side part of application, a **container**
- A JEE container implements the runtime environment and supports the API's
- Huge platform
    - Division into profiles **"Web profile"** (subset of API's)

Java SE is a subset (but can't use everything, IO, Thread, ...)

JEE is a Specification

JEE is a specification

- Your application should run on any product adhering to the specification
- Possible to choose from many vendors
- If not satisfied, choose another. Consumer rule!
- Low risk for vendor lock in!
- ..but also confusing, Many vendors to choose from

Some Links

JEE Specification

http://www.oracle.com/technetwork/java/javaee/tech/index.html

All JSR's

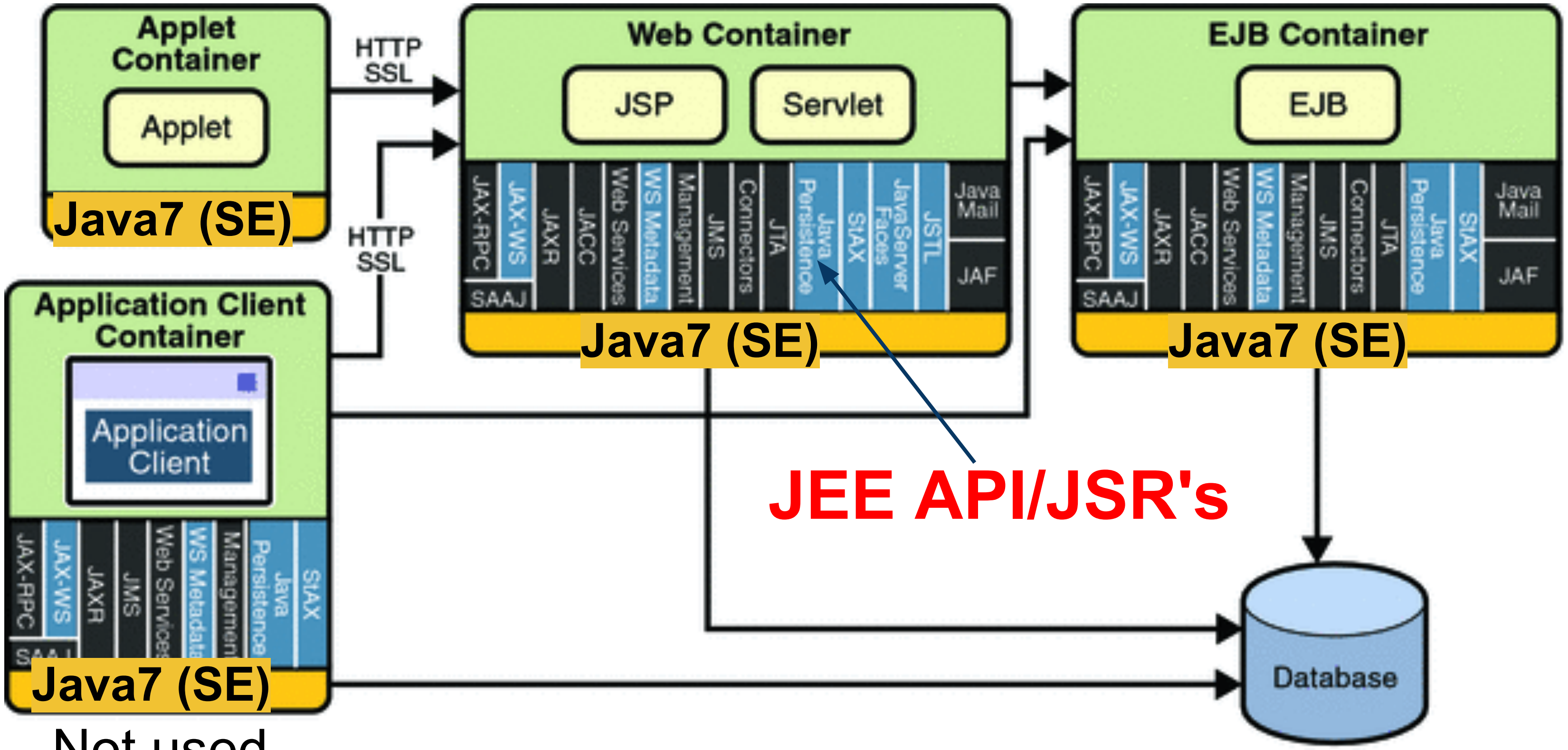
http://jcp.org/en/jsr/all

JEE 6 Tutorial

http://docs.oracle.com/javaee/6/tutorial/doc/

JEE Containers and API's

Not used

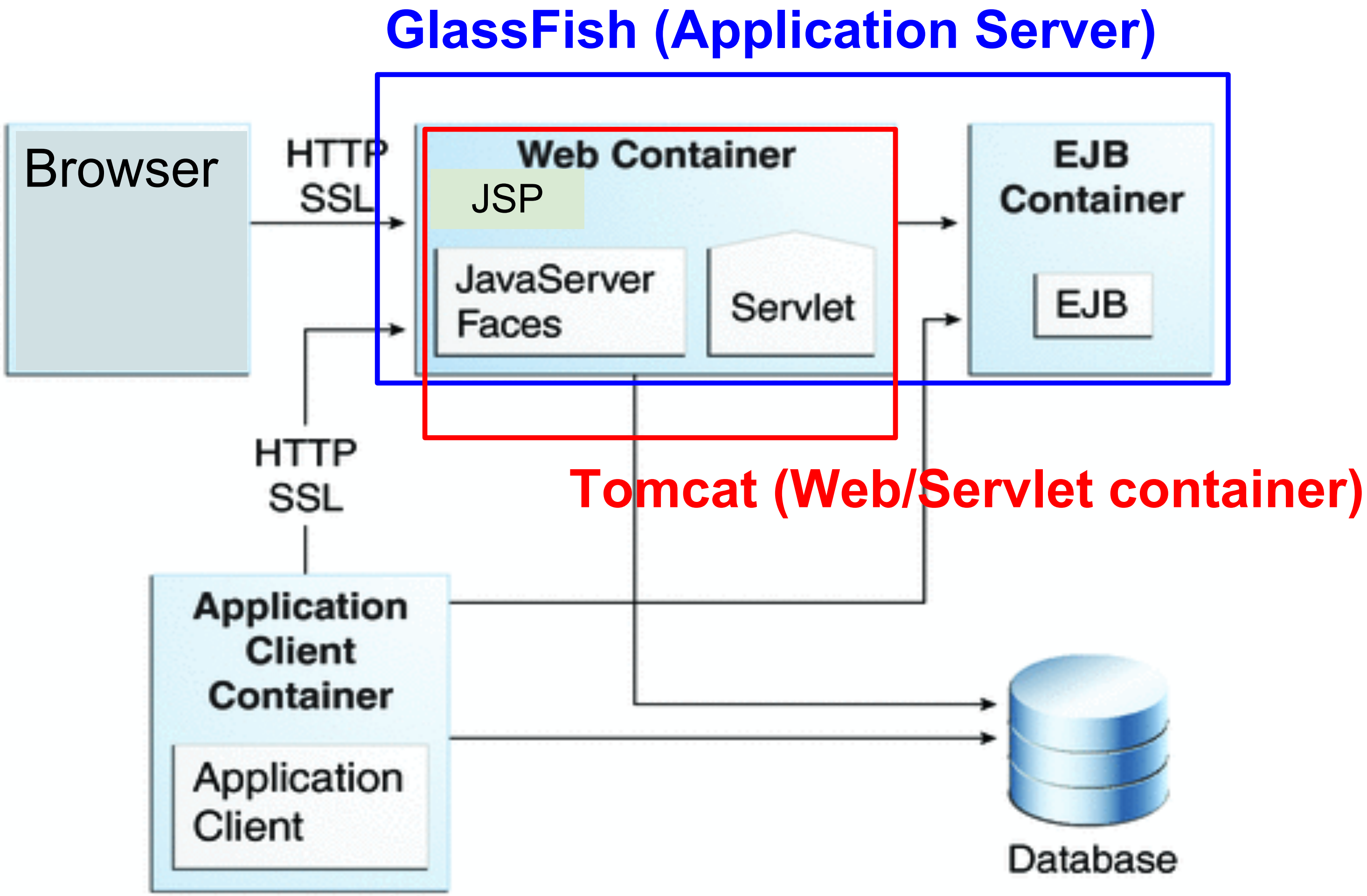


JEE API/JSR's

Not used

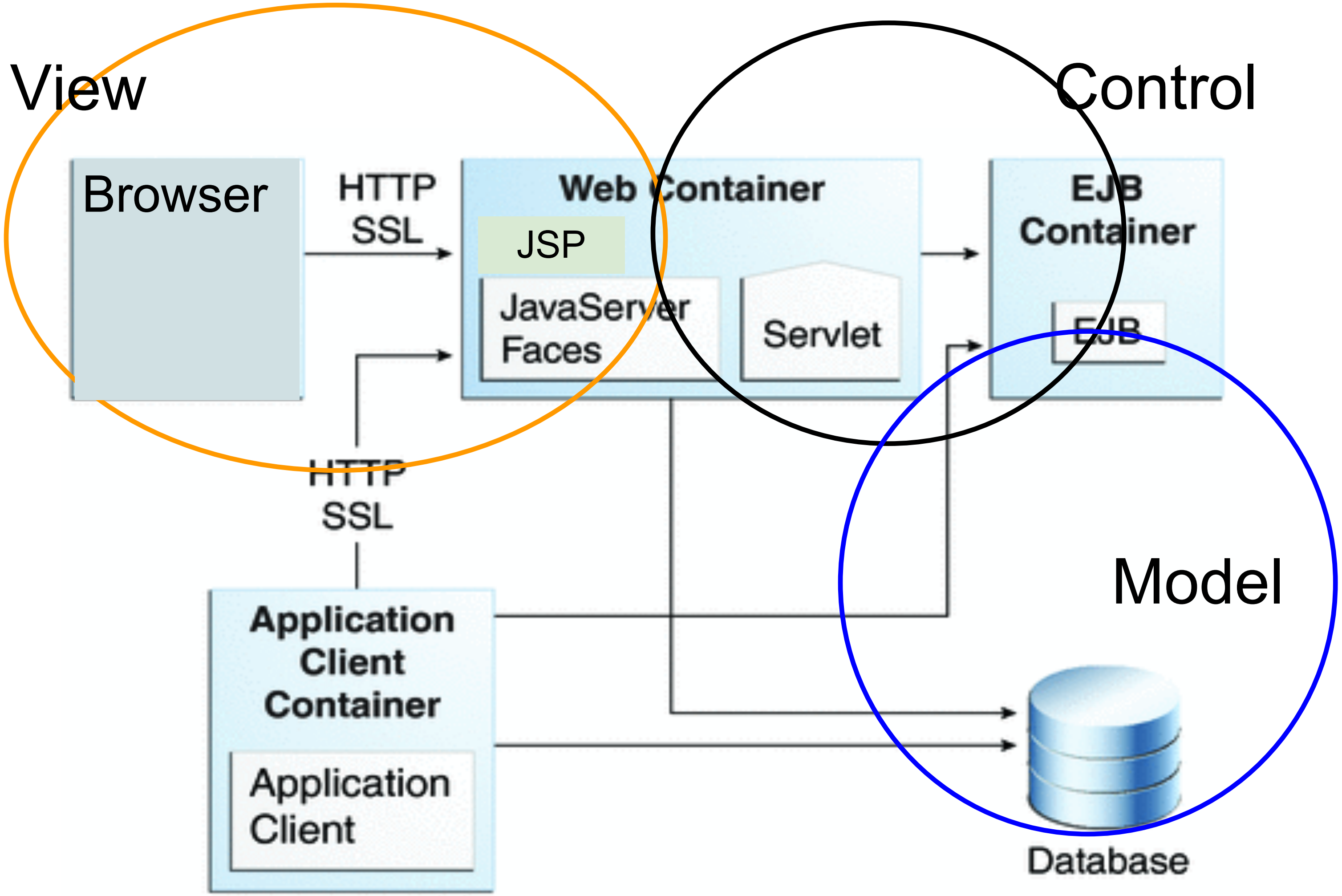
Not all API/JSR's shown

JEE Containers In Course

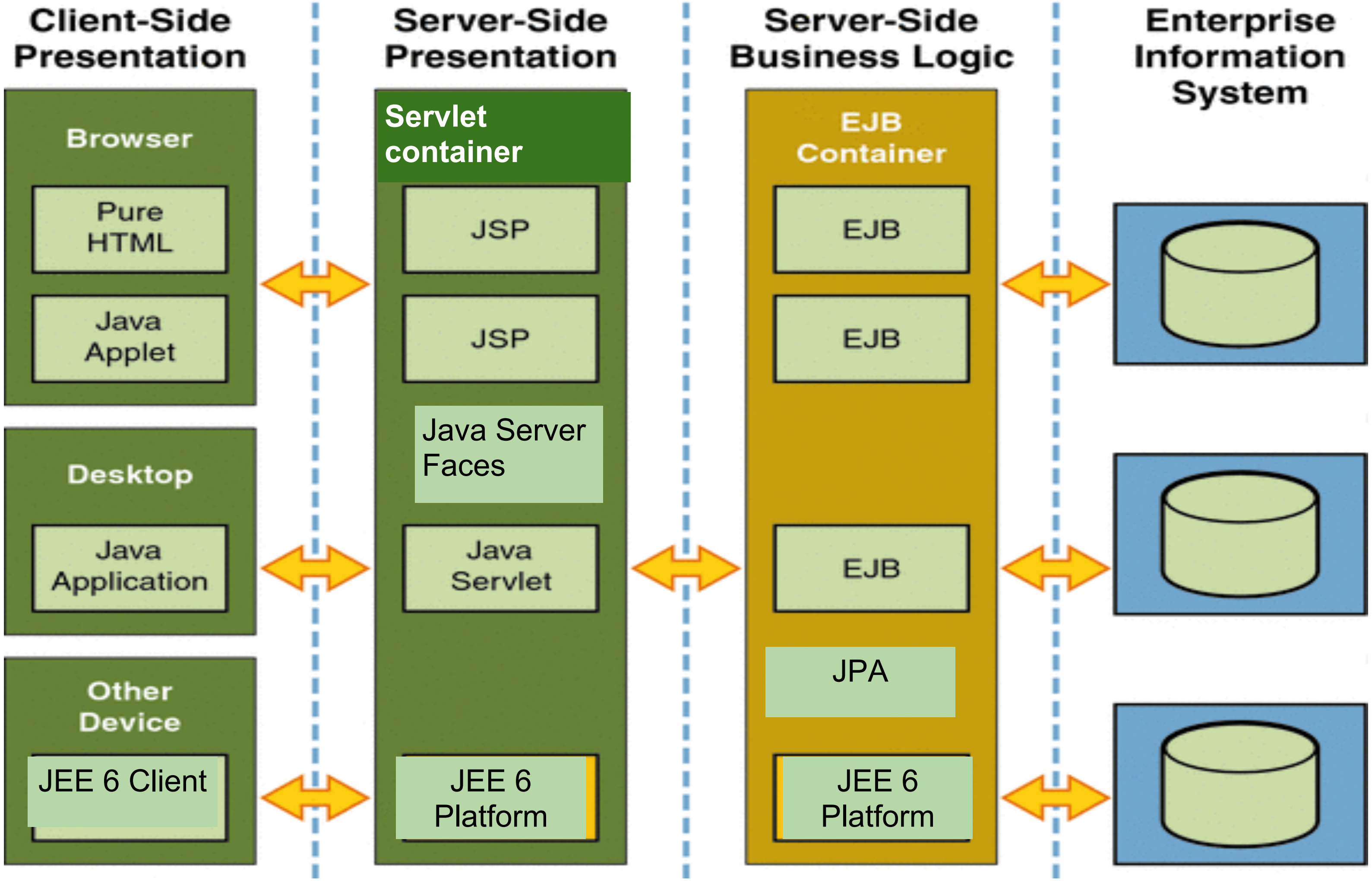


There are many more containers, see web

JEE Containers and MVC



JEE Tiers



Client machine

Each tier possible on separate machine(s) or in cloud

JEE Container Runtime Support

Container "runs" the application

- No `public static void main(...)`

Many objects handled by container

- Objects created, initiated, handled,..., destroyed by container
- Container handles input/output to application

Container uses **dependency injection**

- To connect the application (connecting objects)

Concurrency, security, transactions, ... handled, a bit confusing

- **Container managed** or...
- ...**application managed** (application run in container but handle the issue, often used for customization)

JEE Web application

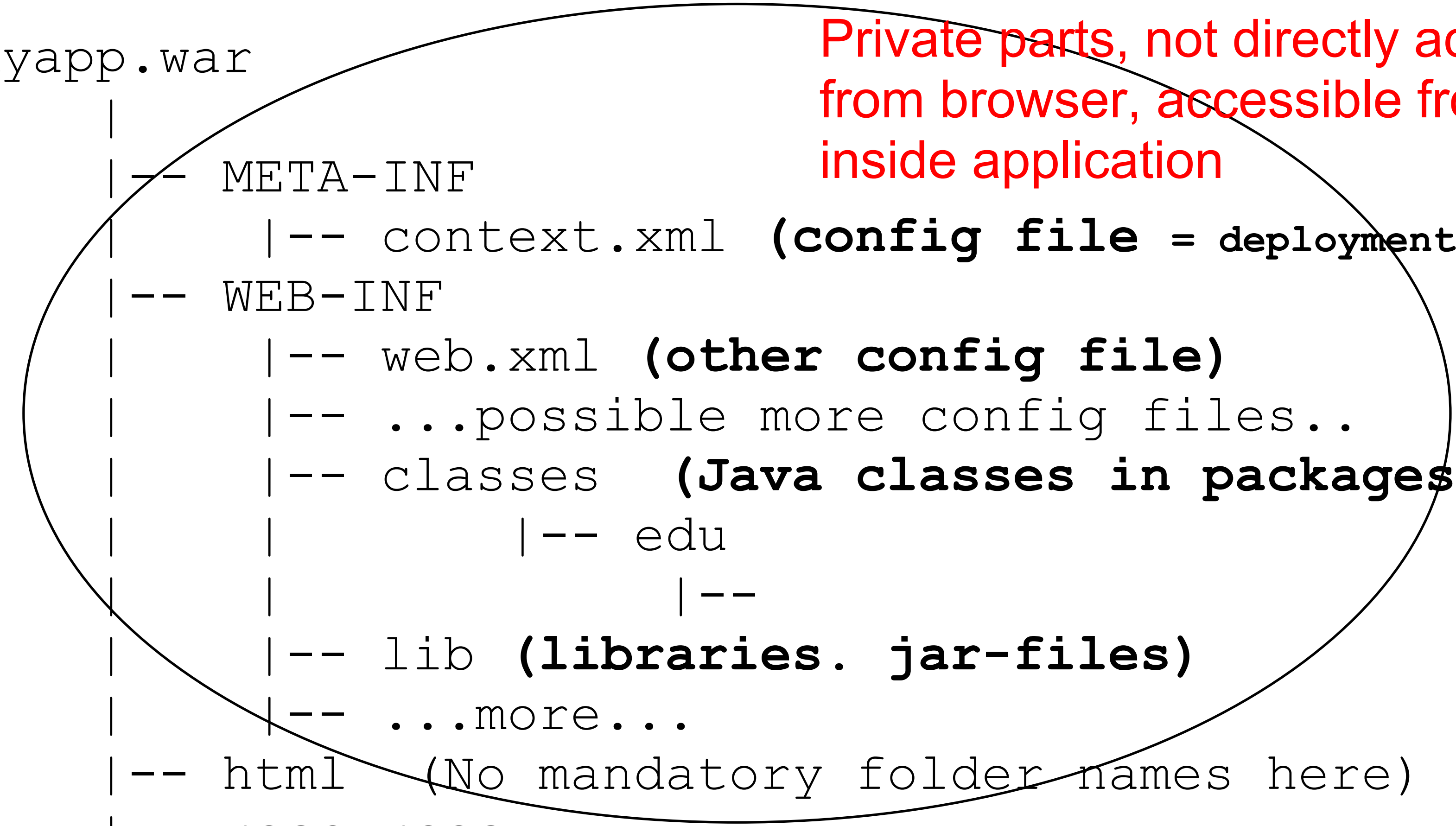
To make it possible for a container to run the application it must be packaged as a "Web-application"

- A *.**war-file**, a packed directory structure with some fixed directory names and a few configuration files (**deployment descriptors**)

<u>Not</u> the same structure as Maven!

- NetBeans will transform the Maven structure to a Web application structure during build

War-file Structure



```
myapp.war
    |
    |-- META-INF
    |    |-- context.xml (config file = deployment descriptor)
    |-- WEB-INF
    |    |-- web.xml (other config file)
    |    |-- ...possible more config files..
    |    |-- classes  (Java classes in packages)
    |    |       |-- edu
    |    |            |--
    |    |-- lib (libraries. jar-files)
    |    |-- ...more...
    |-- html  (No mandatory folder names here)
    |-- resources
         |-- css
         |-- img
```

Private parts, not directly accessible from browser, accessible from inside application

Deployment of Web Applications

Deployment: Install application (war-file) on server (no need in course, NetBeans runs "in place")

New <u>important phase</u> (besides coding, compiling)

Application **verified** during deployment

- Deployment descriptors
- War-structure
- … many more…

If application erroneous, will not be installed (i.e. not run)

- <u>Successful compilation doesn't guarantee successful deployment</u>
- Watch output in NetBeans for **deployment errors**

Configuring Web Applications

JEE has adopted "configuration by exception"
- Famous example: Ruby on Rails framework

Principle: All configuration should have reasonable default values

Many (most) things will work out of the box
- Sometimes a bit confusing, ... when does it not?

If not satisfied modify
- "Everything" in JEE is configurable
- "Everything" in JEE is customizable