COMP 3800SEF /3820SEF /S380F /S380W Lecture 1: Overview of Web Applications

Dr. Flora ZHANG

School of Science and Technology

Hong Kong Metropolitan University

Teaching Team

- **Lecture**:
  - ✓ Dr. ZHANG Jingyu, Flora (fzhang@hkmu.edu.hk, 2768 6887 (office))
  - ✓ Code: L21 (COMP S380F) / L01 (COMP 3800SEF) / L01 (COMP 3820SEF)
  - ✓ Time: 16:00 – 17:50 PM (Tuesday)
  - ✓ Venue: IOH / F0201
- **Lab:**
  - **COMP S380F**
    - ✓ P21 (17:00 PM – 17:50 PM, Monday, C0411): Yihan (email TBC)
    - ✓ P22 (12:00 PM – 12:50 PM, Monday, C0411): Yihan (email TBC)
    - ✓ P23 (17:00 PM – 17:50 PM, Thursday, C0411): Winnie (wguo@hkmu.edu.hk)
  - **COMP 3800SEF**
    - ✓ P01 (10:00 AM – 10:50 AM, Friday, C0412): Jiahui (s1365639@live.hkmu.edu.hk)
    - ✓ P02 (9:00 AM – 9:50 AM, Friday, C0412): Yihan (email TBC)
    - ✓ P03 (12:00 PM – 12:50 PM, Wednesday, C0411): Winnie (wguo@hkmu.edu.hk)
  - **COMP 3820SEF**
    - ✓ P01 (12:00 PM – 12:50 PM, Thursday, C0411): Jiahui (s1365639@live.hkmu.edu.hk)
    - ✓ P02 (11:00 AM – 11:50 AM, Thursday, C0411): Jiahui (s1365639@live.hkmu.edu.hk)

Assessment (3800SEF/3820SEF/S380F)

- **Group Project (OCAS) 25%:**
  1. Implement a web application (*details to be released later*)
  2. Submission deadline: April 13, 2026 (Week 14) (tentative)
     - ✓ Gradle Web Application
     - ✓ Submission Form
     - ✓ Demo Video
  3. At most 4 members **from the same course**.

- **Mid-term Test (OCAS) 25%:**
  1. Time: March 10, 2026 (Tues) 16:15 – 17:30 (Week 9) (tentative)
  2. Venue: **IOH / F0201**

- **Final Exam (OES) 50%.**

- To obtain a Pass grade, you must pass both **OCAS (40%)** and **OES (40%)**.

Week	Lecture	Tutorial
Week 1	L1: Overview of Web App	Lab 1: HTML5, CSS, and JavaScript
Week 2	L2: Servlet	Lab 2: Servlet
Week 3	L3: JSP, JavaBean	Lab 3: Servlet: Parameters and Attributes
Week 4	L4: Session	Lab 4: Servlet: Request Parameters
Week 5	L5: EL, JSTL, Custom tag	Lab 5 & 6: JSP, Session activity tracking
Week 6	No lecture! Lunar New Year Holiday	Holiday (S380F-P21/22/23, 3800SEF-P03, 3820SEF-P01/02); Face-to-face Q&A (Friday 3800SEF-P01/02)
Week 7	L6: MVC Model 1 & Model 2, Spring MVC Web Framework	Lab 7: Session – Shopping cart
Week 8	L7: Spring Boot, More on Spring MVC	Lab 8: EL, JSTL
Week 9	Mid-term test	Lab 9: Spring MVC Web Framework
Week 10	L8: Data Access Object, Hibernate, Spring Data JPA	Lab 10: Spring boot, Spring MVC
Week 11	L9: Spring Security, Spring Profiles	Lab 11: Spring Data JPA with Hibernate
Week 12	L10: Dependency Injection, Aspect-Oriented Programming	Lab 12: Spring Security; Holiday: Friday 3800SEF-P01/02 -> Lab 12 delivered on Week 13
Week 13	No lecture! Easter & Ching Ming Holiday	Face-to-face Q&A Lab 12: Friday 3800SEF-P01/02
Week 14	Study break	

Overview of this lecture

- Internet and World Wide Web
- Web browser
- HTTP, HTML, URL
- Static and Dynamic web pages
- Web application
- Server
  - Web Server
  - Application Server
  - Jakarta EE Server
  - Web container
- Structure of a Web Application and its archive (WAR)
- Framework-based development

Internet and Web

- Internet (Infrastructure): It is a massive network of networks that connects computers all over the world. By itself, the Internet only provides connectivity, i.e., it does not define how information is presented or accessed.
- World Wide Web (WWW, Web): The Web is a service built on top of the Internet that allows users to access and share information, mainly through web pages.
  - TCP/IP: reliable data transmission between devices.
  - IP addresses: uniquely identify devices on the Internet.
  - Domain Name System (DNS): translates human-readable domain names into IP addresses.
- Web Communication Protocol: HTTP protocol is used by the Web to transmit data between browser and server.

Web Browser

- Web browser (Client-side application)
  - Display content using mark-up language (e.g., HTML, XML, XHTML)

	Purpose	Syntax Strictness	Custom tags
HTML	Display web pages	Flexible	No
XML	Store/transport data	Very strict	Yes
XHTML	Display web pages	Very strict	No

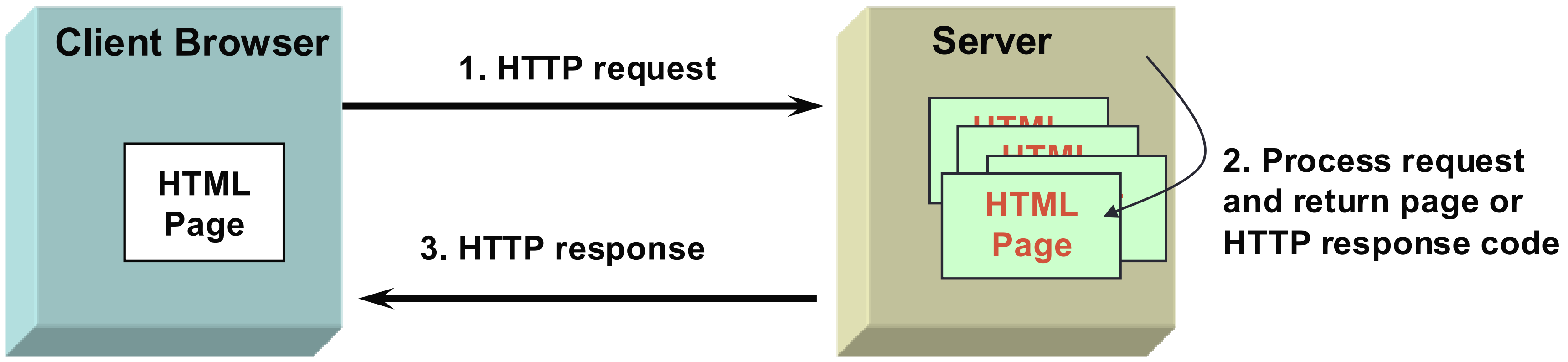
  - Run embedded client-side applications like JavaScript, Ajax, VBScript, Java applet, Flash
- Your favorite browser?
  - Safari, Firefox, Chrome, Opera, IE, Edge?

HyperText Transfer Protocol (HTTP)

- HTTP is a ‘request-response’ protocol.
- Clients (usually browser software) send a request to a web server.
- The server handles the request and provides a response, usually in the form of an HTML page.

HTTP Request and Response

- Clients (browsers) send HTTP requests and web servers send HTTP responses (HTML pages)
  - HTTP request can be issued with different request method, e.g., GET, POST



HTTP Request Methods

- Each HTTP request contains a method attribute that identifies its purpose.

HTTP method	Action to be performed
GET	retrieve a resource
HEAD	get only response headers (for GET request)
PUT	replace a resource with the request body
DELETE	delete the specified resource
POST	submit data to be processed
CONNECT	create a TCP/IP tunnel
OPTIONS	get a list of supported methods
TRACE	echo the request

https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods

HTTP Response Codes

- Each HTTP response contains a response code that indicates the general outcome.

Response code categories	Examples
1xx: Information	*100 continue*
2xx: Success	*200 OK*
3xx: Redirect	*301 Moved Permanently*
4xx: Client Error	*404 Not Found*
5xx: Server Error	*500 Internal Server Error*

HTTP Headers

- Each request and response message begins with header lines that provide meta-information

Headers
Data
Request
Headers
Data
Response

- Request header data examples:
  - method, resource, protocol version, host
- Response header data examples:
  - protocol version, response code, content type, content length, date

HTTP Headers Example

HTTP Request Message

```
GET  /hello.html  HTTP/1.1
Host: www.hkmu.edu.hk
↵
```

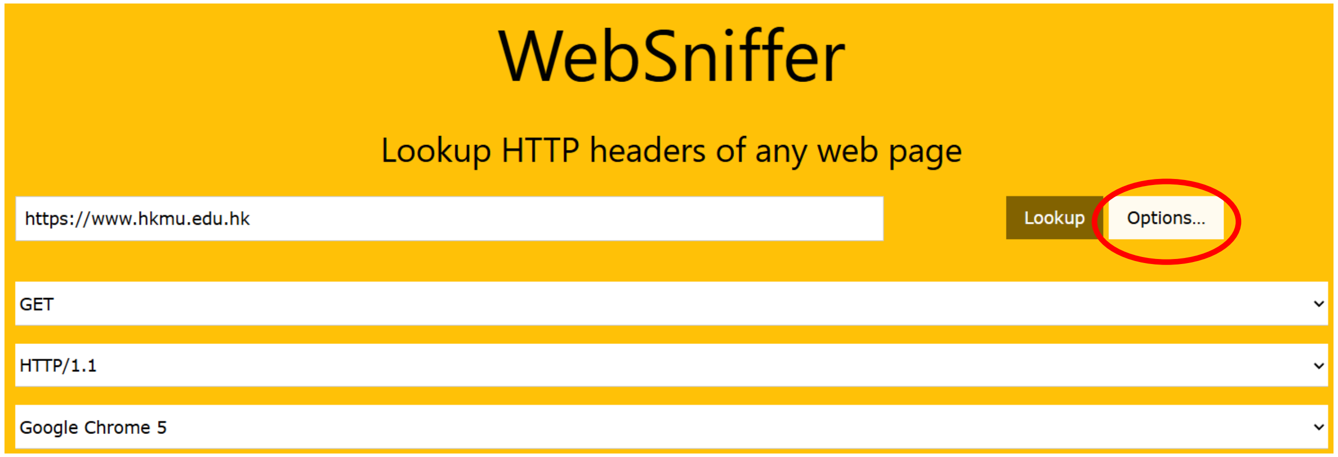
HTTP Response Message

```
HTTP/1.1   200   OK
Server: Apache-Coyote/1.1
Content-Type: text/html
Content-Length: 37
Date: Fri, 07 Sep 2023 16:13:28 GMT
↵
<html>
<body>
Hello!
</body>
</html>
```

A blank line separates message headers from message body

Checking HTTP Headers

- You may what to check out or test more on https://websniffer.com
- You can submit an HTTP request by a given URL. e.g. https://www.hkmu.edu.hk
  - You can see the HTTP request sent and response received.


WebSniffer

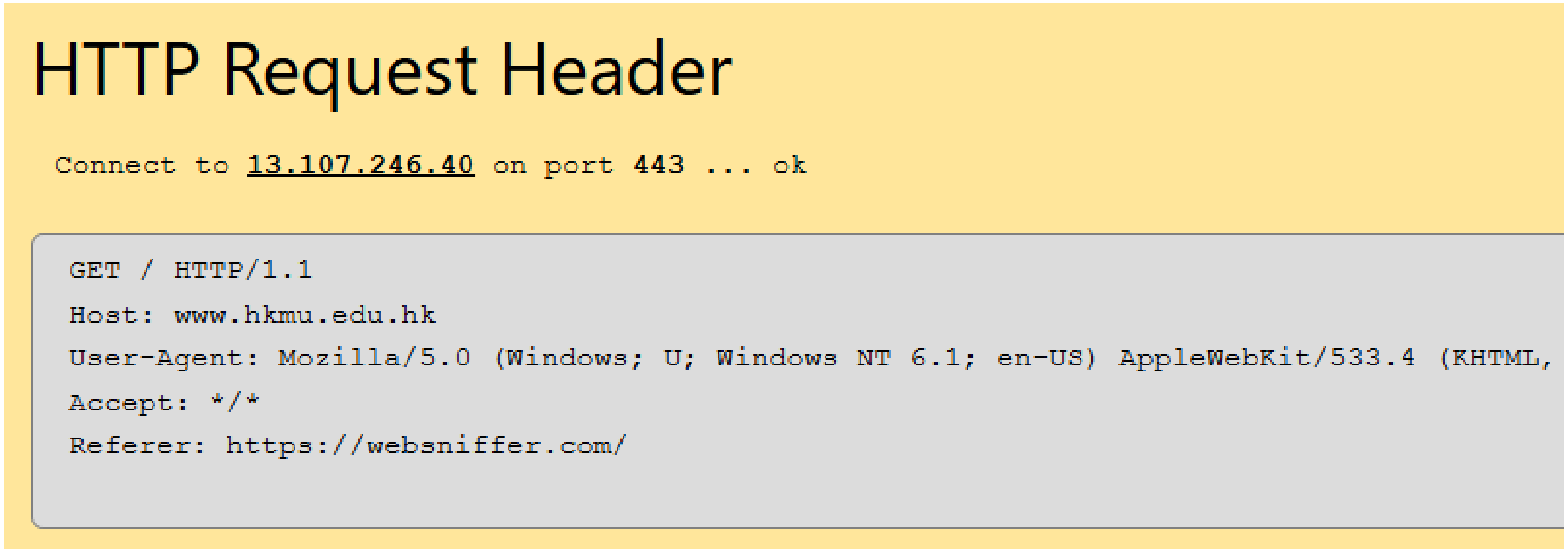
Lookup HTTP headers of any web page

https://www.hkmu.edu.hk

GET

HTTP/1.1

Google Chrome 5

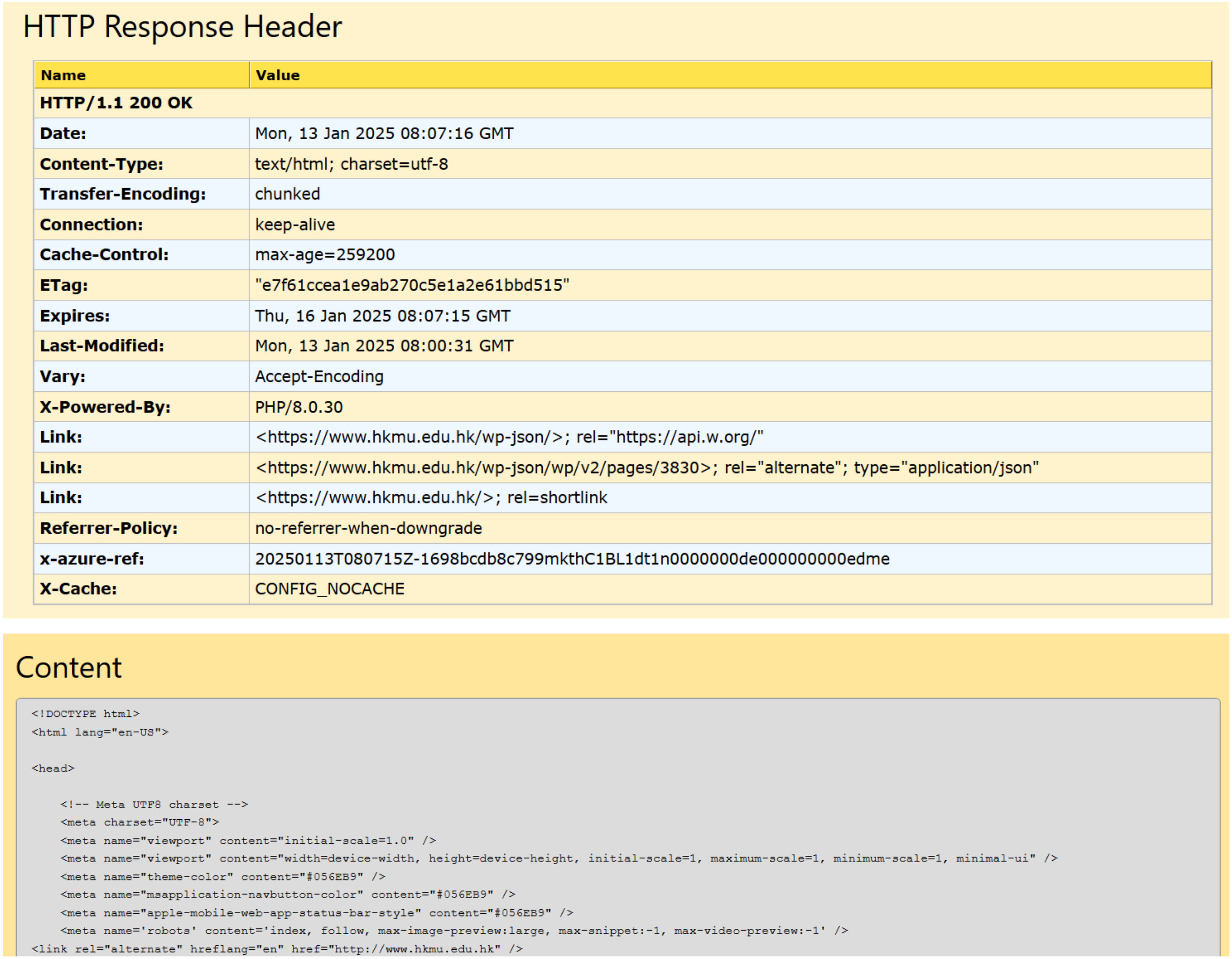


HTTP Request Header

```
Connect to 13.107.246.40 on port 443 ... ok
```

```
GET / HTTP/1.1
Host: www.hkmu.edu.hk
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/533.4 (KHTML,
Accept: */*
Referer: https://websniffer.com/
```

DNS: convert domain name to IP address
User-Agent: identify the client software (e.g., browser type, operating system)
Accept: tell the server what types of content the client can handle. (e.g., */* any type is ok)



HTTP Response Header

Name	Value
HTTP/1.1 200 OK	
Date:	Mon, 13 Jan 2025 08:07:16 GMT
Content-Type:	text/html; charset=utf-8
Transfer-Encoding:	chunked
Connection:	keep-alive
Cache-Control:	max-age=259200
ETag:	"e7f61ccea1e9ab270c5e1a2e61bbd515"
Expires:	Thu, 16 Jan 2025 08:07:15 GMT
Last-Modified:	Mon, 13 Jan 2025 08:00:31 GMT
Vary:	Accept-Encoding
X-Powered-By:	PHP/8.0.30
Link:	<https://www.hkmu.edu.hk/wp-json/>; rel="https://api.w.org/"
Link:	<https://www.hkmu.edu.hk/wp-json/wp/v2/pages/3830>; rel="alternate"; type="application/json"
Link:	<https://www.hkmu.edu.hk/>; rel=shortlink
Referrer-Policy:	no-referrer-when-downgrade
x-azure-ref:	20250113T080715Z-1698bcdb8c799mkthC1BL1dt1n0000000de000000000edme
X-Cache:	CONFIG_NOCACHE

Content

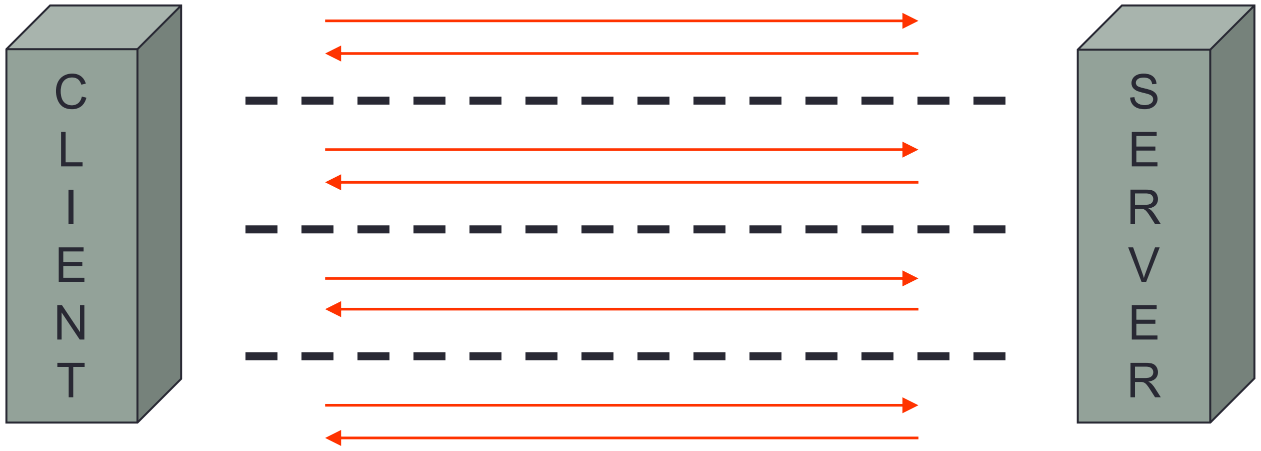
```
<!DOCTYPE html>
<html lang="en-US">

<head>

    <!-- Meta UTF8 charset -->
    <meta charset="UTF-8">
    <meta name="viewport" content="initial-scale=1.0" />
    <meta name="viewport" content="width=device-width, height=device-height, initial-scale=1, maximum-scale=1, minimum-scale=1, minimal-ui" />
    <meta name="theme-color" content="#056EB9" />
    <meta name="msapplication-navbutton-color" content="#056EB9" />
    <meta name="apple-mobile-web-app-status-bar-style" content="#056EB9" />
    <meta name='robots' content='index, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1' />
<link rel="alternate" hreflang="en" href="http://www.hkmu.edu.hk" />
```

HTTP is stateless

- There is no memory (preservation of state) between HTTP transactions.



- Each HTTP transaction is independent of the one before it and the one after it.
    - Statelessness is a scalability property.
    - To customize content of a website for a user, we can use cookies, sessions, hidden variables in a web form…

HTML

- The information on the web is mainly in the form of HTML (HyperText Markup Language) pages.
  - HTML pages are text documents that contain special mark-up tags telling the browser what type of information they contain.
- It is up to the browser to format the page and manage its content.
  - The same page can look different in different browsers.

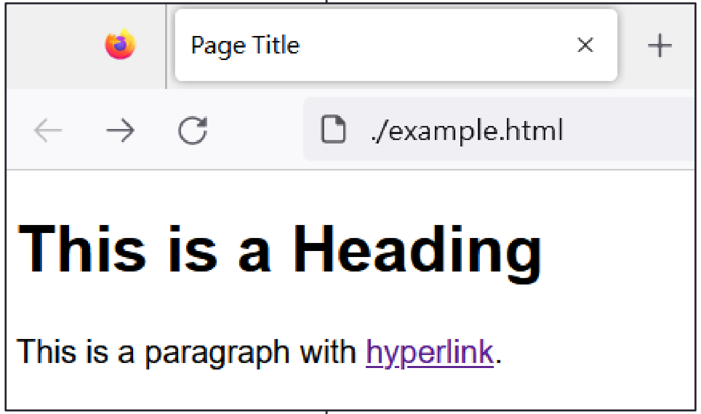
```
<html>
 <head>
  <title>My Page</title>
 </head>
 <body>…
```

HTML: Example

- Below is an HTML5 page; contents are marked up by HTML tags.
- A tag may have attributes, e.g., href in <a href = "URL">. The HTML attribute values must be enclosed in double quotes.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Page Title</title>
</head>
<body>
  <h1>This is a Heading</h1>
  <p>This is a paragraph with
  <a href="http://www.hkmu.edu.hk">hyperlink</a>.
  </p>
</body>
</html>
```

example.html



This is a Heading

This is a paragraph with hyperlink.

- HTML tutorial: https://www.w3schools.com/html/

URL (Uniform Resource Locator)

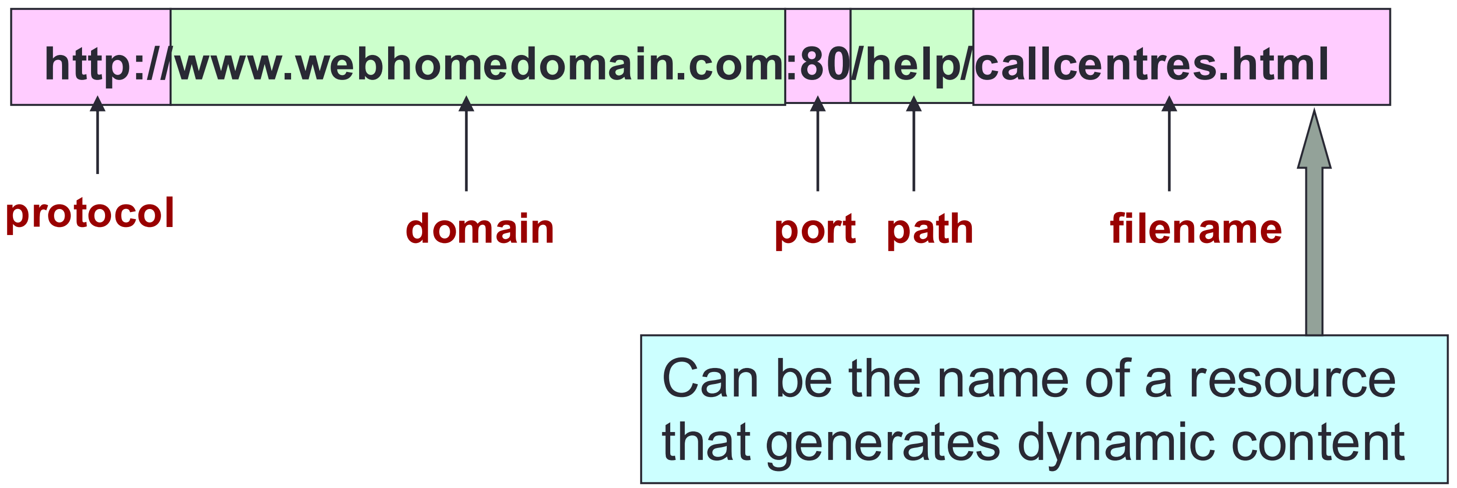
- A URL is the complete location of an Internet resource, comprising:
  - The protocol of the request (usually http://)
  - The server's domain name or IP address
  - The port number (http is default to port 80, https 443)
  - The subdirectory path (if applicable)
  - The name of the resource (e.g., default file index.html, index.jsp, non-files: https://www.google.com/search)

- Failed requests have specific HTTP responses (e.g. 404 – file not found)

Example URL

- General form for a URL:
  <scheme><domain name><port><path><filename>

- For example,
  http://www.mywebsite.net:80


http://www.webhomedomain.com:80/help/callcentres.html
protocol
domain
port
path
filename
Can be the name of a resource that generates dynamic content

URL Protocols

Feature	HTTP (HyperText Transfer Protocol)	HTTPS (HTTP Secure)
Purpose	Web content	Secure web content
Encryption	✗	✓
Default port	80	443
Stateless	Yes	Yes
Browser support	Yes	Yes
Security level	Low	High

There are many other protocols: FTP, FTPS, SFTP,…
https://en.wikipedia.org/wiki/Lists_of_network_protocols

Static and Dynamic Web Pages

- Static web pages are of limited use to the users.
- Dynamic web pages are desirable as it can provide a live, dynamic, or interactive user experience.
- Dynamic web pages can be made using
    - Client-side scripting: The web page is processed using HTML scripting running in the browser when it loads, e.g., JavaScript.
    - Server-side scripting: The web page is generated by an application server which processes server-side scripts, before the web page is sent to the client.

Pros and Cons

Client-side scripting (e.g., JavaScript in the browser)

Pros	Cons
Fast response (immediate feedback)	**Not secure** (code is visible)
Reduced server load	Limited access to server resources
(Partial) offline support	Browser compatibility issues

Server-side scripting (e.g., HTML, JSP, Servlet, PHP)

Pros	Cons
More secure (code not visible to user)	**Slower** interaction
Access to server resources	Higher server load

Server Pages

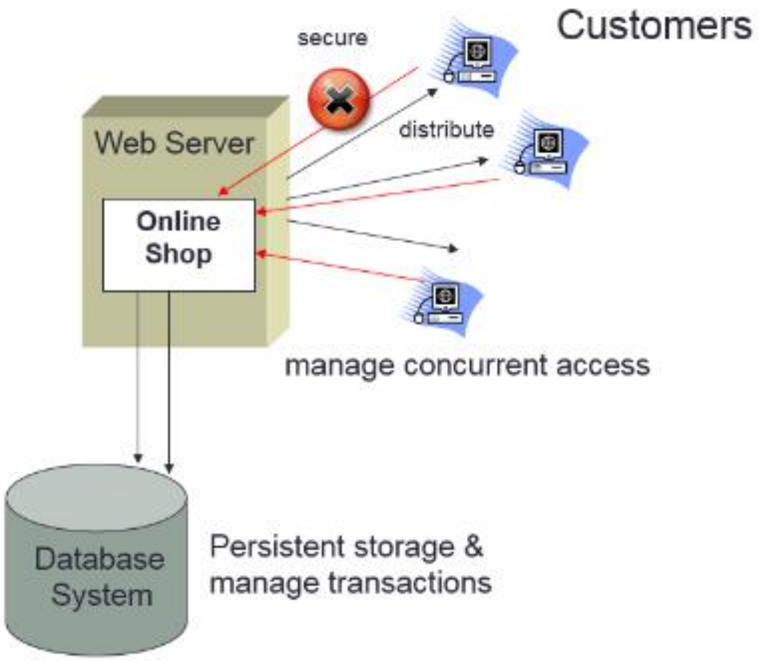
- Server pages are a technology for generating **dynamic** web pages on the server before sending them to the client as HTML
- They are programs that run on the server
    - We will use server pages written in Java (JSP)
    - But they can be written using other languages, e.g., PHP, ASP, Python Server Pages.

Web Application and its features

- A Web application is any application that uses a Web browser as a client.
  - Accessed via a URL; Runs over HTTP/HTTPS; Provides dynamic functionality.
  - Examples: online shopping site, student portal

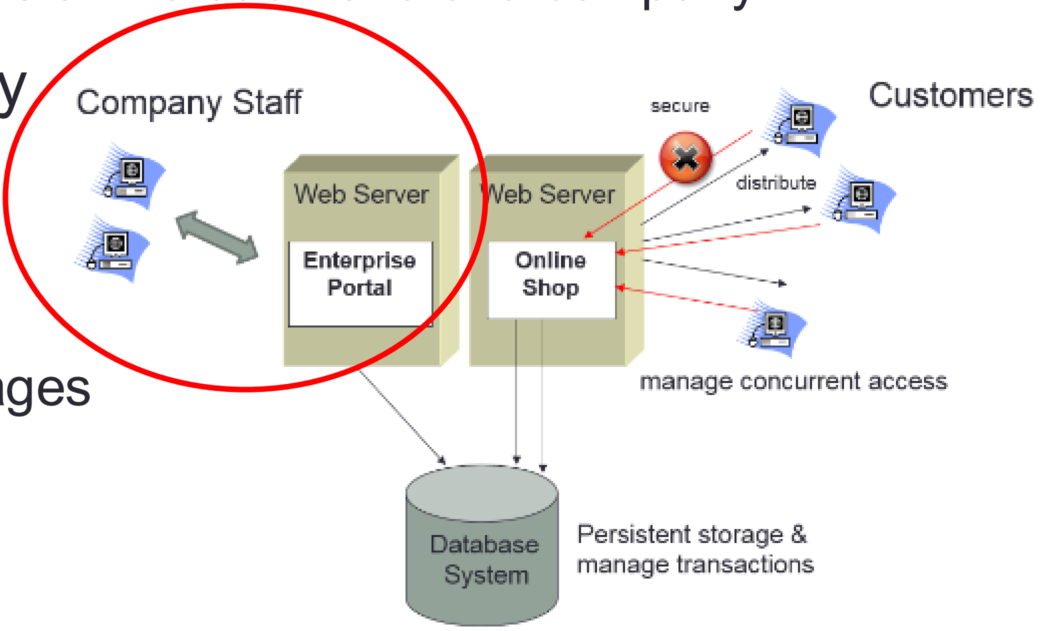
Features of web applications

- Distribute information over WWW
  - New announcement or promotion
- Manage concurrency access from many users
  - Both new or old customers



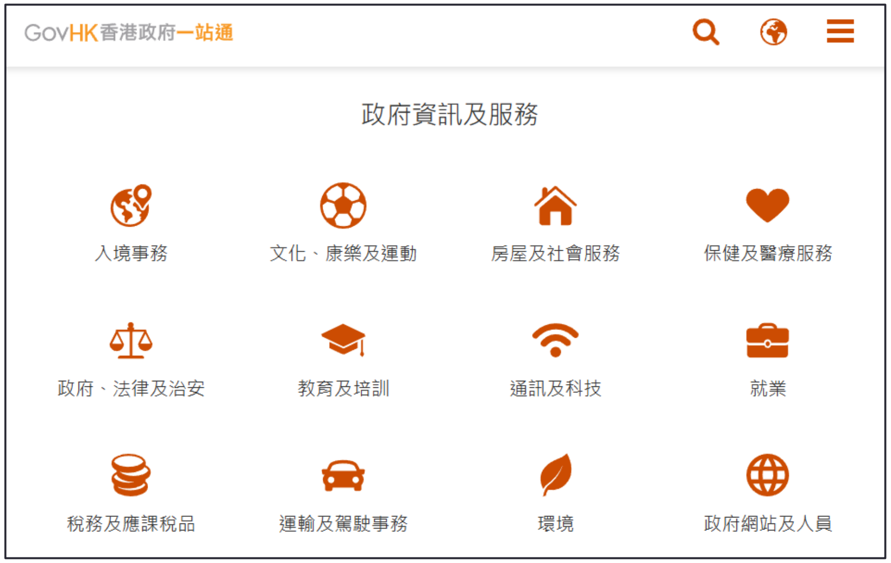
Web Application and its features (cont’d)

- Generate dynamic content based on user’s need
  - Respond to user’s search of particular product
- Utilize a database for permanent storage and transaction handling
  - Give a linkage to the database at the back-end of a company
- Include role-based security and access rights
  - Certain customer is allowed to access limited pages only, while staff may modify the pages



Example: Portals

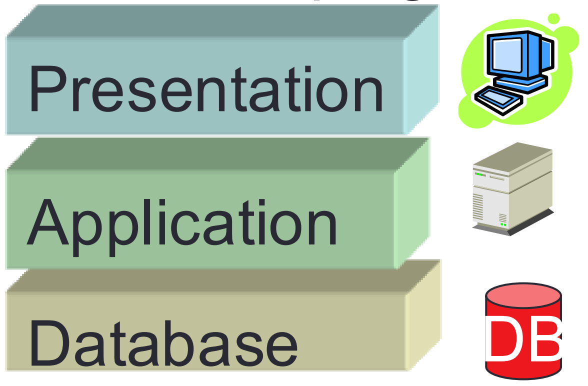
- Portals are web applications which provide a single point of access to online information
- Gateways into other applications
- In order to facilitate access to large amount of information, portals usually include search and navigation capabilities.
- Personalised / customisable
- See Yahoo!, GovHK, etc.



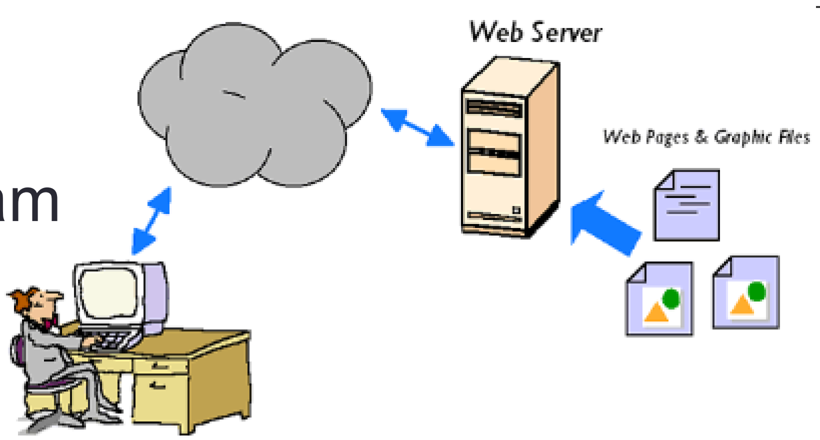
Architectures of a Web Application

- Web application is a software that is created with web technologies and accessed via a web browser.
- It involves different layers or levels of development.
  - Similar or even more complicated than developing software
- Web Applications are multi-tier:

Layer	Responsibility	Technologies
Presentation Layer	UI, displaying web pages	HTML, CSS, JavaScript, etc.
Application Layer	Process user requests	Servlets, JSP, Controllers.
Database Layer	Storing & managing data	SQL, NoSQL



Web (HTTP) servers



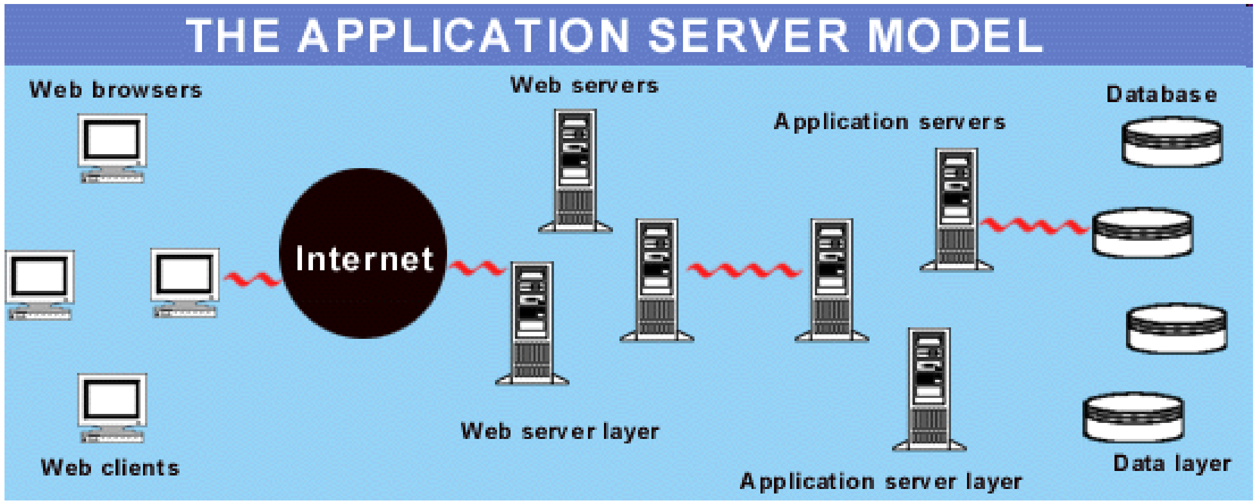
- **Web server** is a computer program that is responsible for accepting HTTP requests from clients and serving them HTTP responses.
- To process an HTTP request, a Web server may
  - Respond with a ***static*** HTML page or image
  - Send a redirect (response code 3xx)
  - Delegate the dynamic response generation to some other program, such as
    - CGI scripts
    - Servlets or JSPs (Jakarta Server Pages / JavaServer Pages)
    - some other server-side technology

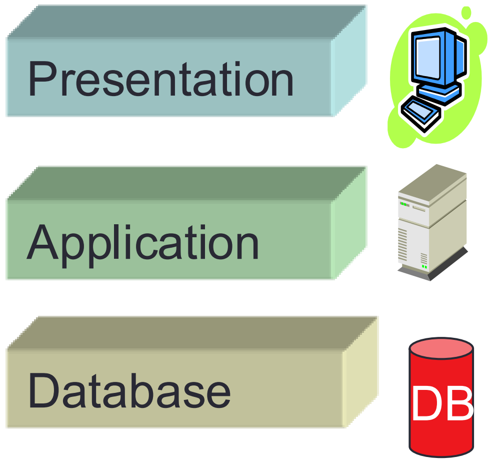
Web server features

- In practice, many web servers implement the following features:
  - Authentication
  - Handling of static content
  - Secure communication: HTTPS support (by SSL or TLS)
  - Content compression (e.g., compress responses by gzip)
  - Virtual hosting (multiple domains on a server)
  - Large file support
  - Bandwidth throttling
- Examples of Web servers:
  - Apache
  - Nginx
  - Microsoft IIS

Application servers

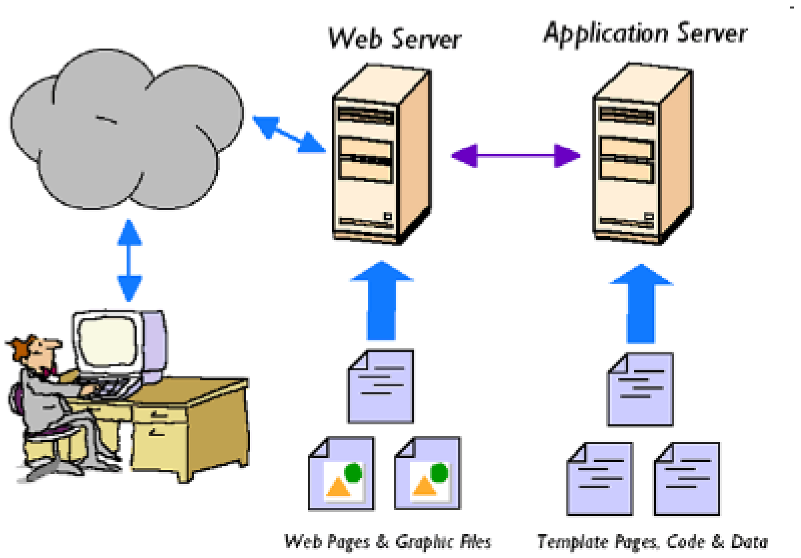
- Application server is responsible for handling the business logic of the system.
- Separating business (application) logic from the presentation logic and database logic: **3-tier architecture**





Application server features

- Application servers extend web servers to support dynamic content using server-side scripting.
- The application server manages its own resources and may provide features such as:
  - Security
  - Transaction management
  - Database connection pooling
  - Clustering support
  - Messaging



- Most application servers also contain a Web server.

Web server vs Application server

	Web server	Application server
Primary Function	Serves static content like HTML, CSS, and JavaScript.	Serves dynamic content by executing server-side code and providing application services.
Content Handling	Handles HTTP requests and responses for static content.	Serve static content, generate dynamic content, handle business logic, and also provide access to server-side logic (server application)
Use Case	Ideal for serving static websites or acting as a reverse proxy.	Ideal for hosting complex web applications requiring server-side logic and business processes
Scope	May be used alone, or as a component in an application server.	Have components and features to support application-level services such as connection pooling, object pooling, transaction support, messaging services, etc.
Security Features	Basic authentication and SSL/TLS support.	Advanced security features like authentication, authorization, and data encryption.

Jakarta EE / Java EE

- **Jakarta Enterprise Edition (Jakarta EE)** (formerly, Java EE) is a comprehensive platform for multi-user, enterprise-wide applications.

= Core parts of **Java SE** (Java Standard Edition)
   + Many **<u>additional APIs</u>** for writing **enterprise-level** software
   - ➢ E.g., distribution, security, transactions, persistence
- Support **web application** development

- Jakarta EE system includes
  - ➢ Servlets
  - ➢ Jakarta Server Pages (JSPs) (formerly, JavaServer Pages)
  - ➢ Jakarta Enterprise Beans (EJB) (formerly, Enterprise JavaBeans)
  - ➢ + many others

Reference: https://jakarta.ee/specifications/platform/10/

Java EE versions (History)

- Java EE **was** maintained by Oracle until 2017.
- Eclipse Foundation has taken over from Oracle.

Java EE version	Release time	Java SE support
J2EE 1.2 – 1.4	1999 – 2003	J2SE 1.2 – 1.4
Java EE 5	2006	Java SE 5
Java EE 6	2009	Java SE 6
Java EE 7	2013	Java SE 7
Java EE 8	2017	Java SE 8

Jakarta EE versions (History)

- As Oracle owns the trademark for “Java”, the Eclipse Foundation renamed Java EE to Jakarta EE.
- Jakarta (雅加達) was the old capital of Indonesia (印尼), and also the largest city on the island of Java (爪哇島).

Jakarta EE version	Release time	Java SE support	Remark
Jakarta EE 8	2019	Java SE 8	Fully compatible with Java EE 8
Jakarta EE 9	2020	Java SE 8	API namespace move from javax.* to jakarta.*
Jakarta EE 9.1	2021	Java SE 8 Java SE 11	
Jakarta EE 10	2022	Java SE 11 Java SE 17	In this course, we use **Jakarta EE 10** with **Java SE 17 (Liberica JDK 17)**.

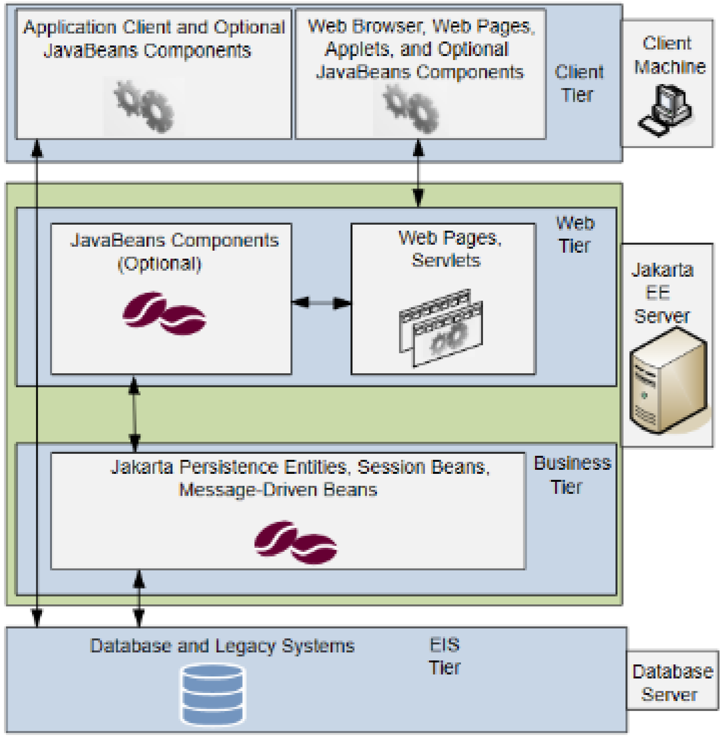
Jakarta EE 10 technologies

- Web Application Technologies
  - Servlet 6.0
  - Server Pages 3.1
  - Faces 4.0
- Web Services Technologies
  - JAX-RS 3.0.0, JAX-WS 4.0, JAXB 4.0, …
- Enterprise Application Technologies
  - EJB 4.0, JMS 3.1.0, JPA 3.1, JTA 2.0.0, Jakarta Mail 2.1, …

References:
https://jakarta.ee/release/10/
https://projects.eclipse.org/releases/jakarta-10

Jakarta EE Server

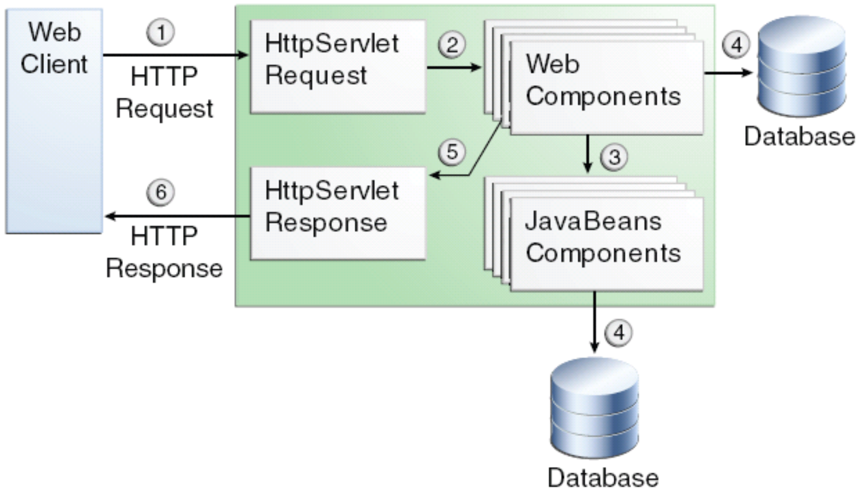
- Jakarta EE Server (formerly, Java EE server) is an **application server**, whose core set of API and features are defined by Jakarta EE.
- Jakarta EE defines an architecture for implementing services through the use of a Jakarta EE server as multi-tier applications that deliver the scalability, accessibility, and manageability needed by enterprise-level applications.
  - **Business and presentation logic**: to be implemented by developer
  - **Standard system services**: provided by the Jakarta EE platform


Application Client and Optional JavaBeans Components
Web Browser, Web Pages, Applets, and Optional JavaBeans Components
Client Tier
Client Machine
JavaBeans Components (Optional)
Web Pages, Servlets
Web Tier
Jakarta EE Server
Jakarta Persistence Entities, Session Beans, Message-Driven Beans
Business Tier
Database and Legacy Systems
EIS Tier
Database Server

Reference (Jakarta EE tutorial): https://eclipse-ee4j.github.io/jakartaee-tutorial/

Web Applications & Components

- **Web application** is a dynamic extension of a web server or an application server.
  - Presentation-oriented (HTML, XML pages)
  - Service-oriented (Web services)
- **Web components** provide the dynamic extension capabilities for a web server:
  - Servlets
  - JSP pages
  - Web service endpoints



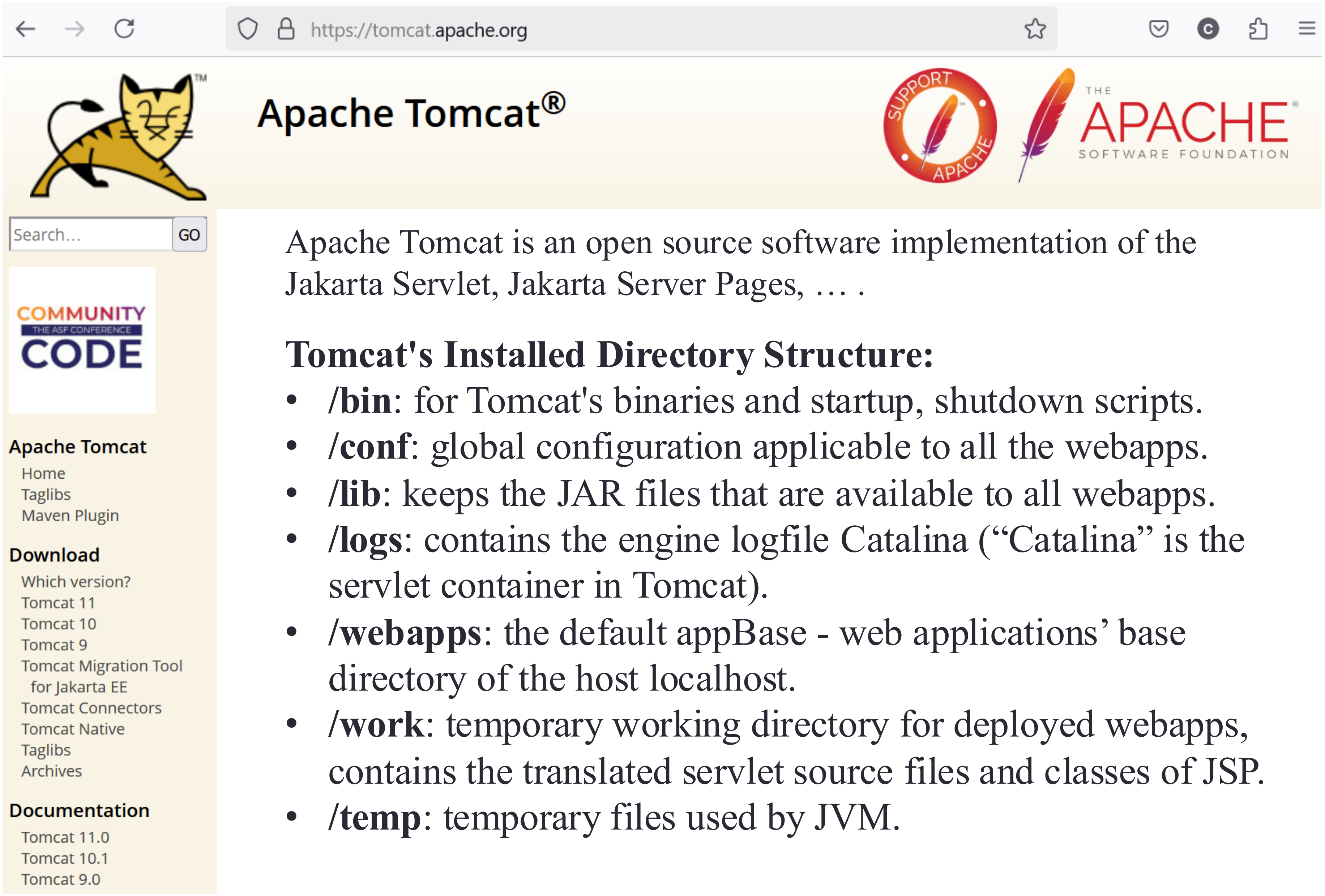
Web Containers

- Web components are supported by the services of a runtime platform called a **web container** (also known as **Servlet containers**).
- Most **web containers** implement only the Servlet, JSP and JSTL specifications.
- **Jakarta EE Application Server** implements the entire Jakarta EE specification.
- Every application server contains a web container, which is responsible for
    - Managing the life cycle of Servlets
    - Mapping request URLs to Servlet code
    - Accepting and responding to HTTP requests
    - Concurrency
    - Security
    - Naming, transactions, email APIs

Examples of Application Servers & Web Containers

- Application Servers
  - Eclipse GlassFish (Jakarta EE server)
  - WildFly
  - Oracle WebLogic Server
  - IBM WebSphere
- Web Containers (lightweight)
  - Apache Tomcat
  - Jetty
  - Tiny Java Web and App server (TJWS)

Apache Tomcat


https://tomcat.apache.org
Apache Tomcat®
SUPPORT APACHE
THE APACHE SOFTWARE FOUNDATION
Search... GO
COMMUNITY THE ASF CONFERENCE CODE

Apache Tomcat
- Home
- Taglibs
- Maven Plugin

Download
- Which version?
- Tomcat 11
- Tomcat 10
- Tomcat 9
- Tomcat Migration Tool for Jakarta EE
- Tomcat Connectors
- Tomcat Native
- Taglibs
- Archives

Documentation
- Tomcat 11.0
- Tomcat 10.1
- Tomcat 9.0

Apache Tomcat is an open source software implementation of the Jakarta Servlet, Jakarta Server Pages, … .

Tomcat's Installed Directory Structure:

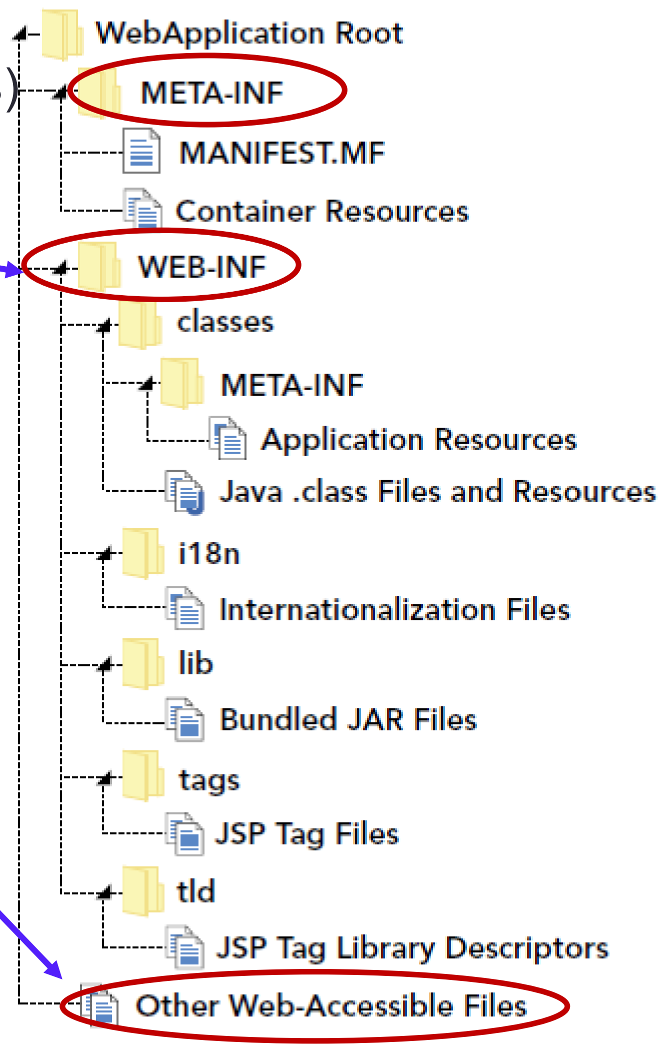
- **/bin**: for Tomcat's binaries and startup, shutdown scripts.
- **/conf**: global configuration applicable to all the webapps.
- **/lib**: keeps the JAR files that are available to all webapps.
- **/logs**: contains the engine logfile Catalina ("Catalina" is the servlet container in Tomcat).
- **/webapps**: the default appBase - web applications' base directory of the host localhost.
- **/work**: temporary working directory for deployed webapps, contains the translated servlet source files and classes of JSP.
- **/temp**: temporary files used by JVM.

Deployment

- Web components have to be installed or **deployed** to the web container
- Aspects of web application behaviour can be configured during application **deployment**
- The configuration information is maintained in an XML file called a web application **deployment descriptor**
  - Its filename is ***web.xml***

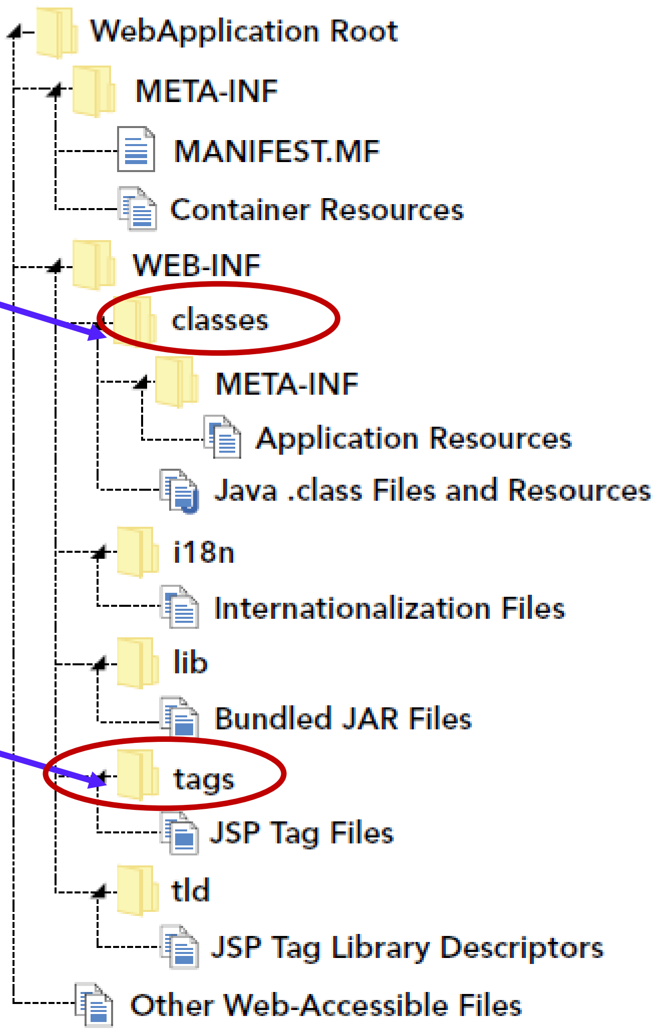
Web Application Structure

- **Web Application Root** (URL visits this)
- **Private resources**:
  - WEB-INF
  - META-INF
- **Public resources**:
  - JSP pages
  - client-side classes
  - client-side archives
  - static web resources



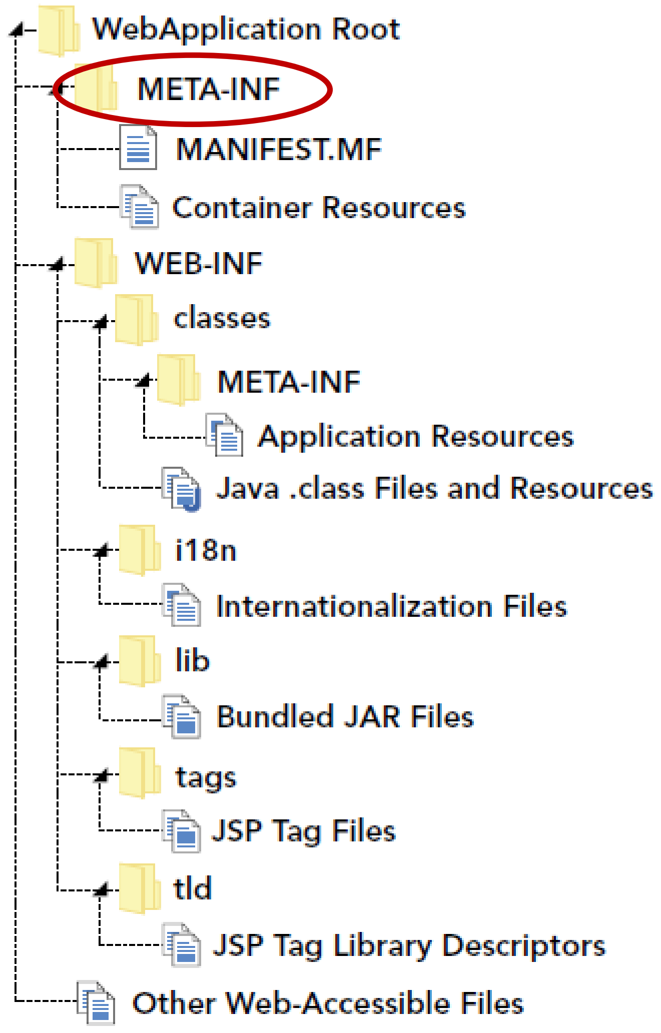
Web Application Structure

- **WEB-INF/**
  - **classes**: server-side classes
    - servlets
    - utility classes
    - JavaBeans components
  - **tags**: JSP tag files, which are implementations of tag libraries



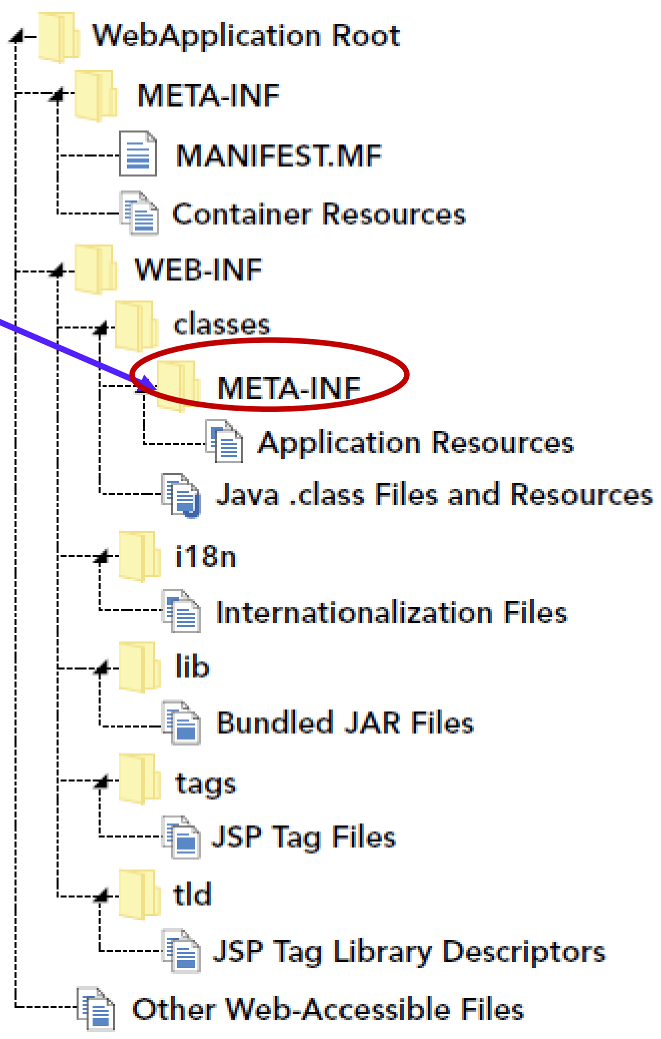
Web Application Structure

- **META-INF/**
  - Contain application manifest file
  - E.g., Tomcat looks for and uses **context.xml** file in this directory to help customize how the application is deployed in Tomcat.
  - NOT on application classpath.



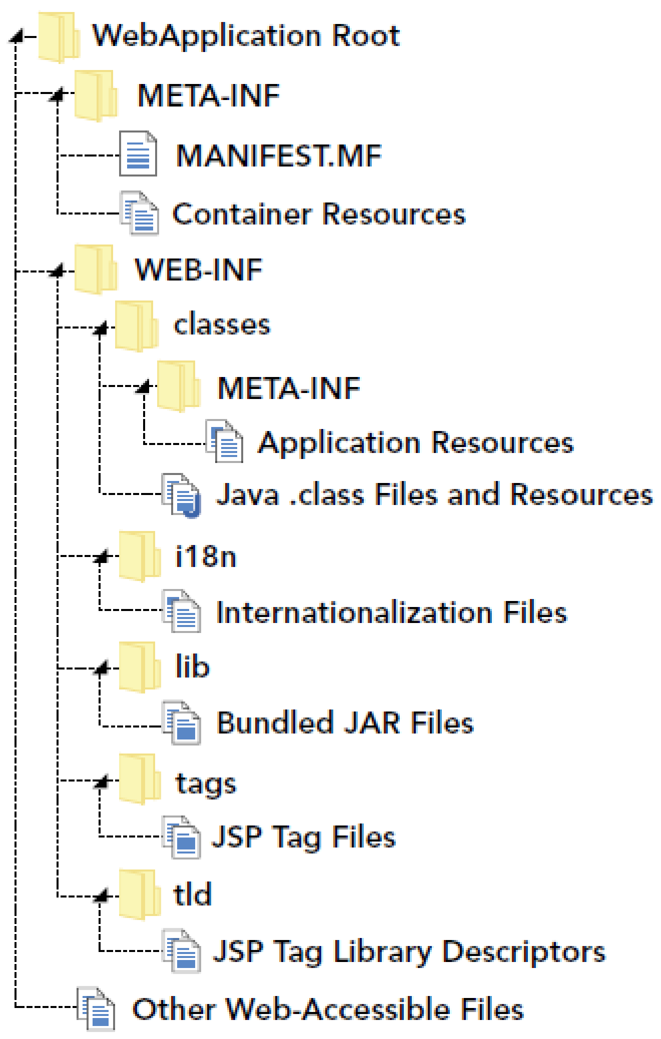
Web Application Structure

- /WEB-INF/classes/**META-INF/**
  - On the application classpath
  - Some Java EE components require files in this directory.
  - E.g., **Java Persistence API**:
    - persistence.xml
    - orm.xml



Web Application Structure

- Files in /META-INF/ and /WEB-INF/ are protected resources that are **not accessible via URL**.
- We may place files that we do not want browsers to access directly into /WEB-INF/
  - E.g., We may put some JSP files into the directory /WEB-INF/jsp/



Web Application Archive (WAR)

- A web application can be deployed as an unpacked (or “exploded”) file structure or can be packaged in a JAR file known as a Web application archive (WAR).
    - Any ZIP archive application can create it.

The structure of a Web Application Archive (.war):

```
simple.war\
     index.html
     WEB-INF\
          lib
          classes\myFirstServlet.class
          web.xml
```

To access the Web app:
http://localhost:8080/simple/index.html

Framework-based Development

- Common for many mature application development environments
  - Provide a standard structure or design that allows the developer to create an application, without having to learn or understand complex low-level APIs.

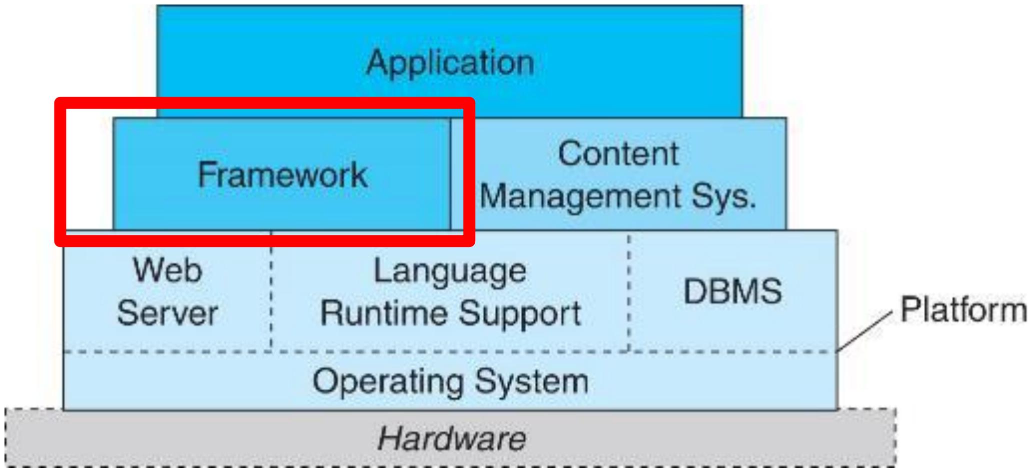
- An example of framework model is MVC (Model-View-Controller; more details will be given later in the course).

Web Application Framework

- A web app framework is a set of tools that support web app development with:
  - A standard design model (e.g., MVC)
  - User interface toolkit
  - Reusable components for common functions (authentication, e-commerce, etc.)
  - Database support
  - Support for distributed system integration

Web Application Framework

- Frameworks give application developers more powerful building blocks to work with.



- Some existing web application frameworks include
  - **Java** : JavaServer Faces (JSF), Struts, **Spring, Spring Boot**
  - JavaScript: React, Angular, Vue.js, Express for Node.js
  - PHP: Laravel, CodeIgniter
  - Python: Django, Flask