COMP S380F Lecture 1: Overview of Web Applications

Dr. Keith Lee

School of Science and Technology

The Open University of Hong Kong

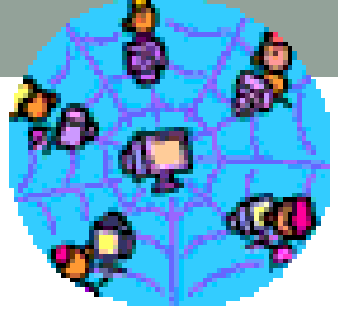
How important is this course?

- Many company depends on the Web to do their business.
  - Newspapers, bookstores, supermarkets, etc.
- Many job titles require the understanding of Web technologies nowadays.
- No matter you work on system development or support
- Many jargons you should have seen in job advertisements
    - JSP, Java EE, Spring framework

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 & Java EE Server
  - Web container
- Structure of a Web Application and its archive (WAR)
- Framework-based development

Internet and Web



- The Internet is a massive network of networks, a networking infrastructure.
- World Wide Web (WWW or Web) is a way of accessing information over the medium of the Internet. It is built on top of the internet infrastructure, which include
  - TCP/IP
  - IP Addresses
  - Domain Name System (DNS)
- The Web uses the HTTP protocol to transmit data over the Internet.

Web Browser

- Web browser
  - Display mark-up language like HTML, XML, XHTML
  - Run embedded client-side applications like Java applet, Flash, Shockwave, Silverlight
- Your favorite browser?
  - Safari, Firefox, Chrome, Opera, IE, Edge?

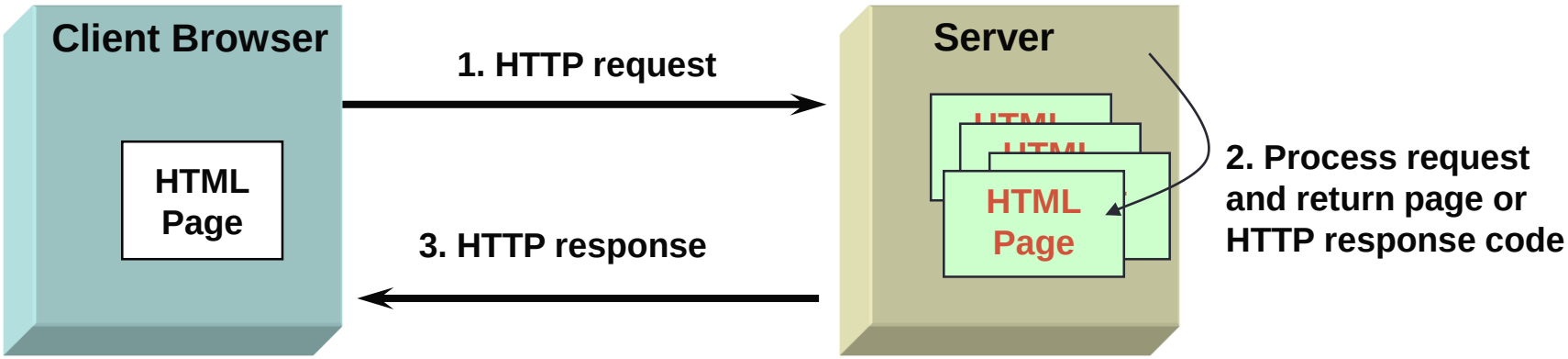



HyperText Transfer Protocol (HTTP)

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

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

- 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.ouhk.edu.hk
↵
```

A blank line separates message headers from message body

HTTP Response Message

```
HTTP/1.1   200   OK
Server: Apache-Coyote/1.1
Content-Type: text/html
Content-Length: 37
Date: Fri, 07 Sep 2007 16:13:28 GMT
↵
<html>
<body>
Hello!
</body>
</html>
```

Checking HTTP Headers

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

HTTP Request Header

Connect to **202.40.220.3** on port **80** ... ok

```
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKi
Host: www.ouhk.edu.hk
Accept: */*
Referer: https://websniffer.cc/
Connection: Close
```

HTTP Response Header

Name	Value
HTTP/1.1 200 OK	
Date:	Wed, 23 Jan 2019 04:58:31 GMT
Server:	Apache/2.2.12 (Linux/SUSE)
Last-Modified:	Tue, 10 Mar 2015 07:02:59 GMT
ETag:	"26a-510e9bda9cec0"
Accept-Ranges:	bytes
Content-Length:	618
Content-Type:	text/html
Connection:	close

Content

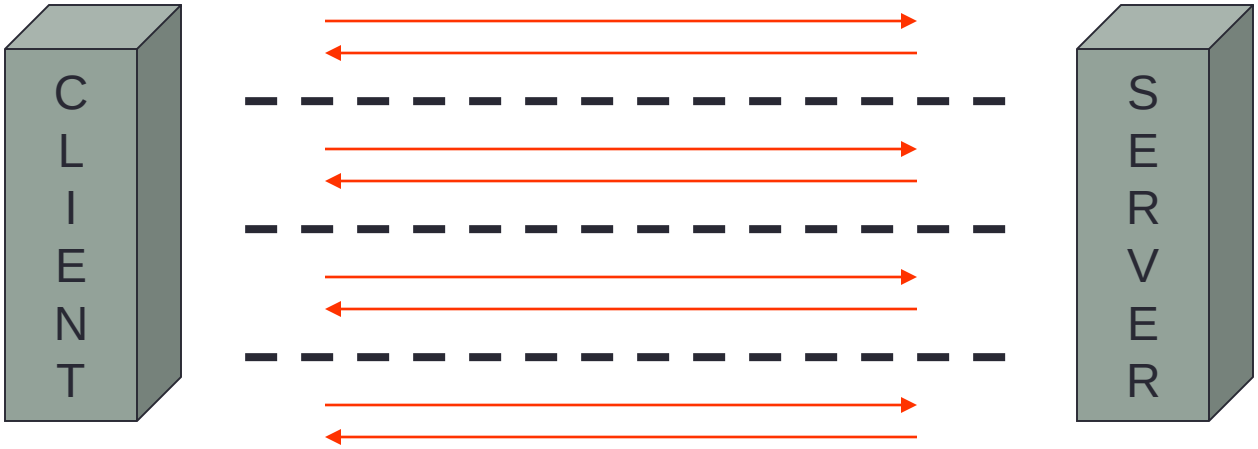
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML lang="zh">
<HEAD>
<title>The Open University of Hong Kong</title>
<HEAD>
<meta http-equiv="refresh" content="0;URL=http://www.ouhk.edu.hk/WCM/?FUELAP_TEMPLATENAME=tcSingPage&lang=eng">
<META NAME="Author" CONTENT="The Open University of Hong Kong">
<META NAME="Keywords" CONTENT="The Open University of Hong Kong">
<META NAME="Description" CONTENT="This is a redirect page">
<META HTTP-EQUIV="Pragma" content="no-cache">
<META HTTP-EQUIV="Cache-Control" content="no-cache">
<META HTTP-EQUIV="Expires" content="-1">
</HEAD>

<BODY>

</BODY>
</HTML>
```

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>…
```

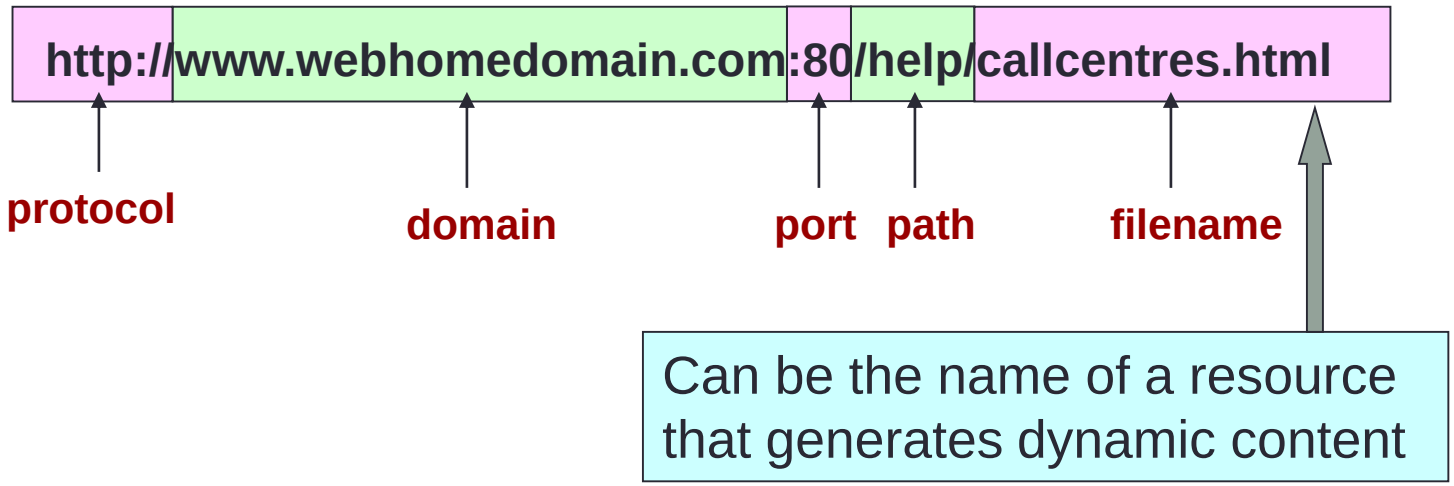
URL

- 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 (though there may be a default)
- 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



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.

Server Pages

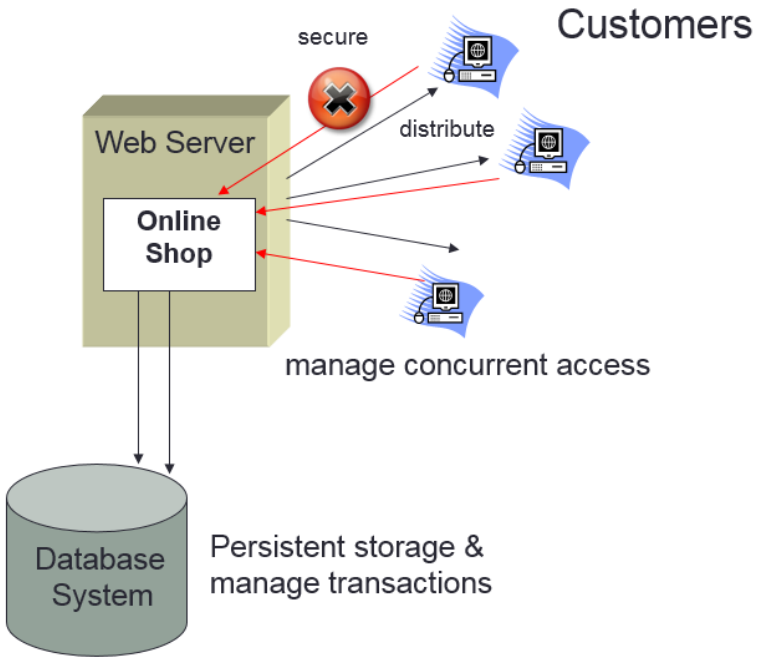
- Server pages are a technology for generating dynamic web pages on the server before sending them to the client as HTML
- They are programs that run on the server
  - We will use server pages written in Java (JSP)
  - But they can be written using other languages, e.g., PHP, ASP.

Web Application and its features

- A Web application is any application that uses a Web browser as a client.
  - It can be as simple as a message board or a guest sign-in book on a website or as complex as a word processor or a spreadsheet.

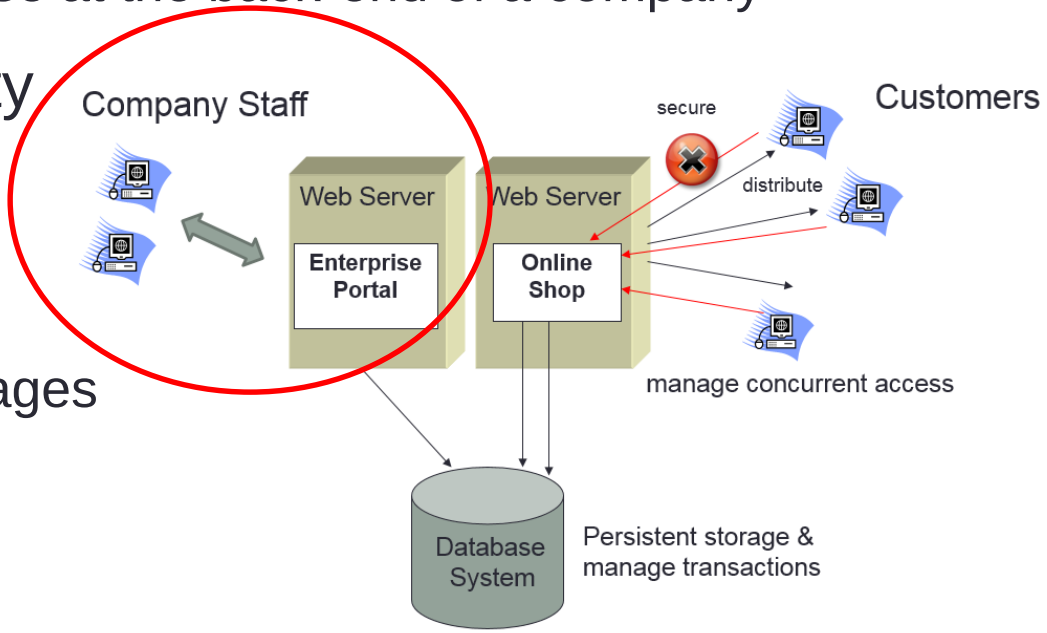
Features of web applications

- 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')

- 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

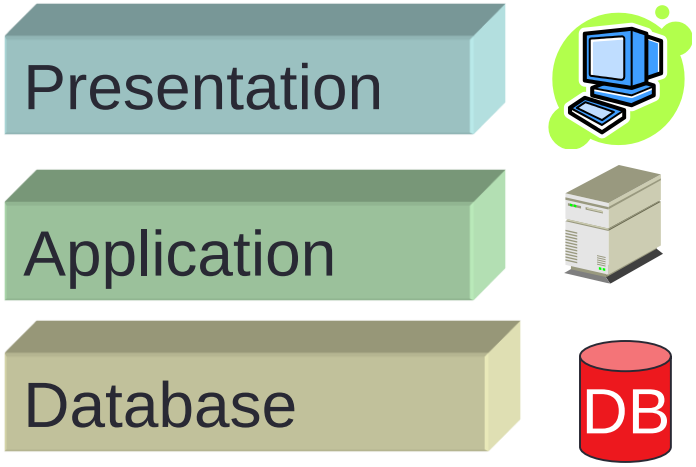


Example: Portals

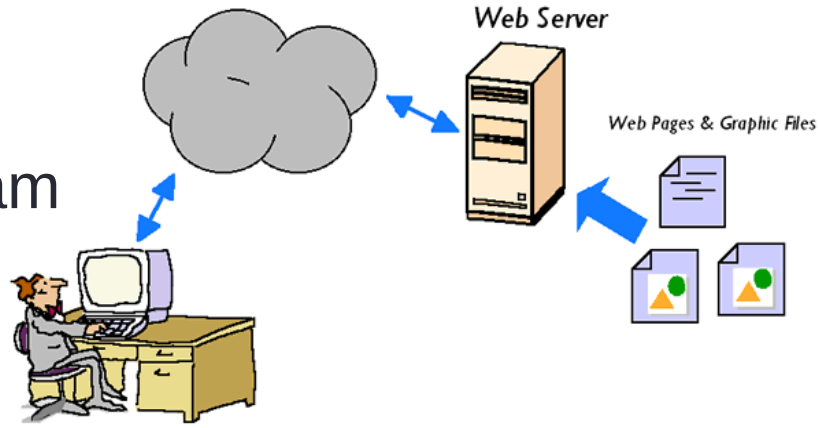
- Portals are web applications which provide a single point of access to online information
- Gateways into other applications
- In order to facilitate access to large amount of information, portals usually include search and navigation capabilities.
- Personalised / customisable
- See Yahoo!, GovHK,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

Architectures of a Web Application

- A web application is **NOT** a single web page.
- It involves different layers or levels of development.
  - Similar or even more complicated than developing software
- Web Applications are multi-tiers.
  - Layers distributed on different hardware or locations (More details will be given later in the course.)



Web (HTTP) servers

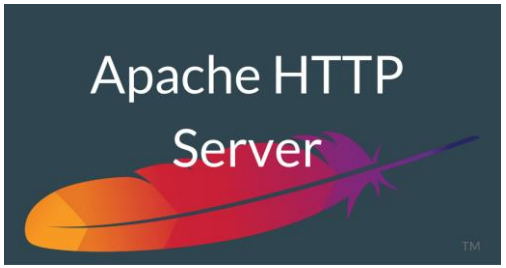


- **Web server** is a computer program that is responsible for accepting HTTP requests from clients and serving them HTTP responses.
- To process a HTTP request, a Web server may
  - respond with a ***static*** HTML page or image
  - send a redirect
  - delegate the dynamic response generation to some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

Web server features

In practice, many web servers implement the following features:

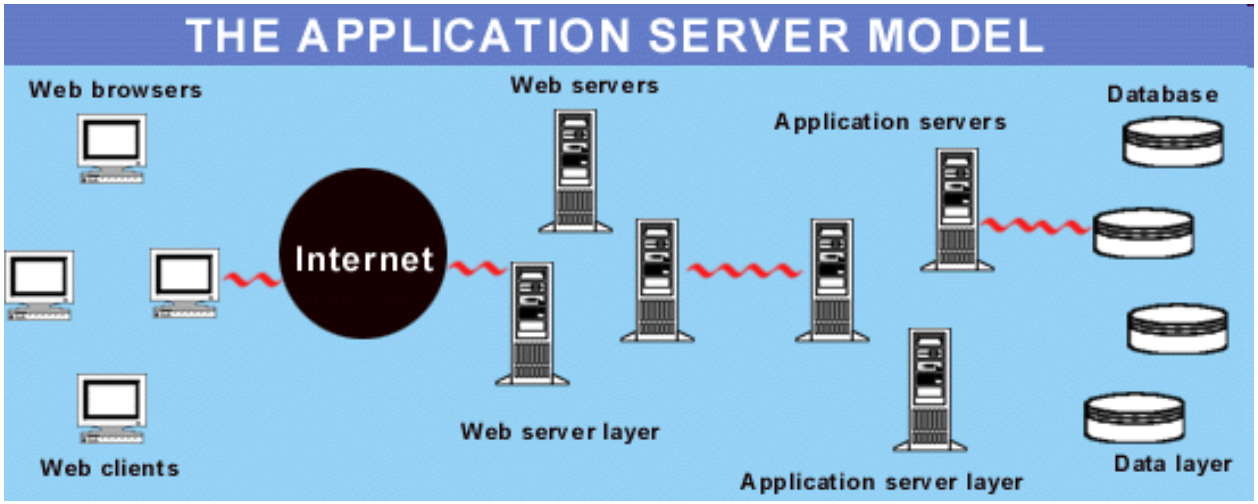
- Authentication
- Handling of static content
- HTTPS support (by SSL or TLS)
- Content compression (i.e. by gzip encoding)
- Virtual hosting (multiple domains on a server)
- Large file support
- Bandwidth throttling

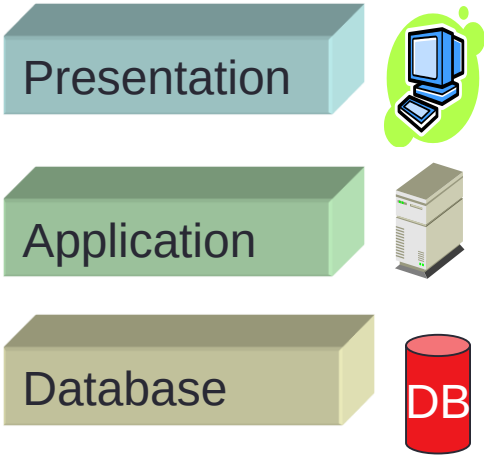


- Examples of Web server:
  - Apache
  - Microsoft IIS

Application servers

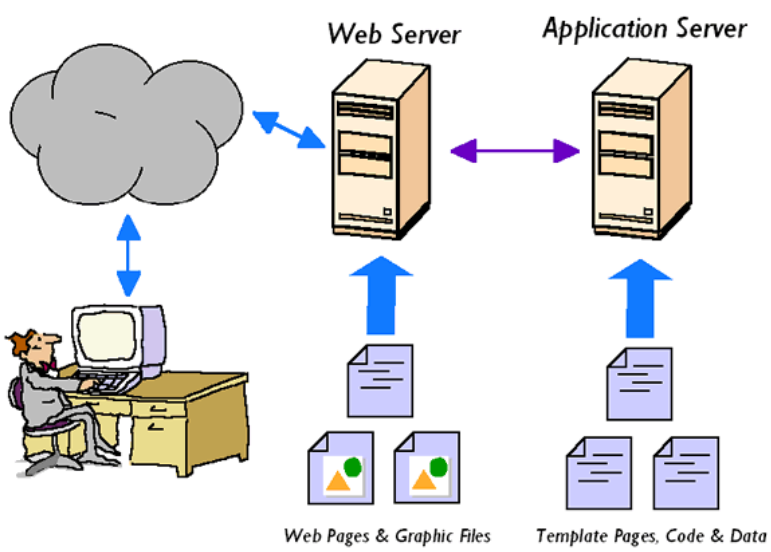
- Application server is responsible for handling the business logic of the system.
- Separating business (application) logic from the presentation logic and database logic: **3-tier architecture**





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
General	Limited to handling HTTP requests.	Execute programs, routines, or scripts that support application construction.
Content	Limited to serve static HTML content.	Serve static content, generate dynamic content, and also provide access to server-side logic (server application)
Visual Display	Adding content extensions is technically possible, but time-consuming, difficult to use, and maintain.	**Include web server** within a complete integrated application server framework.
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.

Java Enterprise Edition (Java EE)

- Java EE is a comprehensive platform for multi-user, enterprise-wide applications.

= Core parts of **Java SE** (Java Standard Edition)
  + Many **additional APIs** for writing **enterprise level** software
    - E.g., distribution, security, transactions, persistence
- Support **web application** development
- Java EE system includes
  - Servlets
  - JavaServer Pages (JSPs)
  - Enterprise JavaBeans (EJB)
  - + many others

Reference: https://docs.oracle.com/javaee/7/tutorial/index.html

Java EE versions

- J2EE 1.2 (Dec 12, 1999)
- J2EE 1.3 (Sep 24, 2001)
- J2EE 1.4 (Nov 11, 2003)
- Java EE 5 (May 11, 2006)
- Java EE 6 (Dec 10, 2009)
- **Java EE 7 (May 28, 2013)**
- Java EE 8 (Aug 31, 2017) (Incompatible with NetBeans)
- Jakarta EE (Feb 26, 2018)

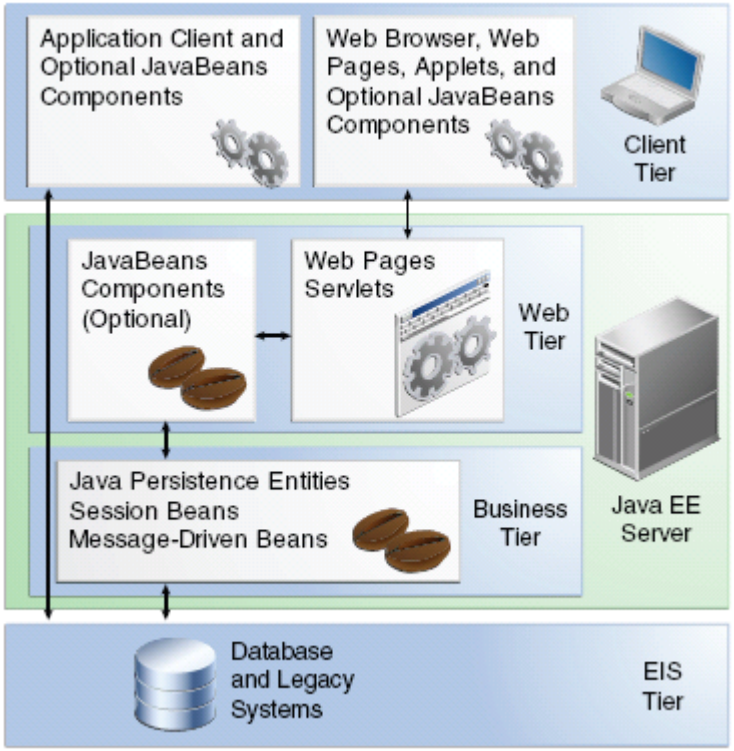
Java EE 7 technologies

- Web Application Technologies
  - Java Servlet 3.1
  - JavaServer Pages 2.3
  - JavaServer Faces 2.2
- Web Services Technologies
  - JAX-RS 2.0, JAX-WS 2.2, JAXB 2.2, StAX …
- Enterprise Application Technologies
  - EJB 3.2, JMS 2.0, JPA 2.1, JTA 1.2, JavaMail 1.5…

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

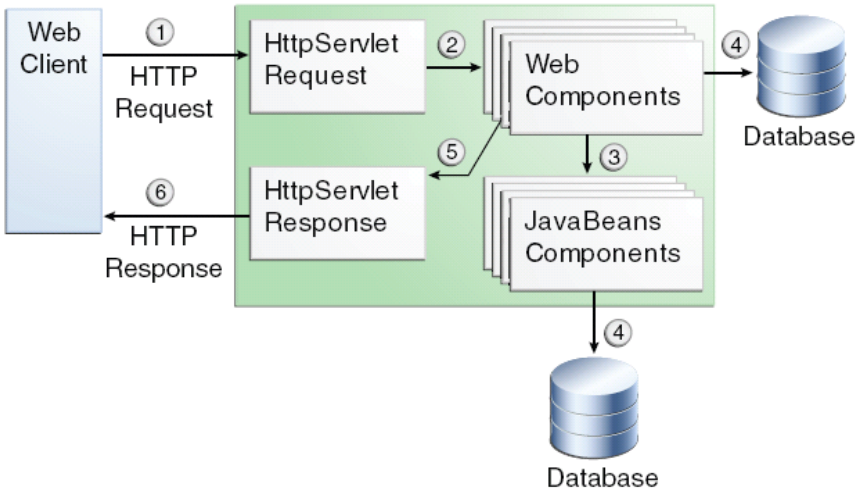
Java EE Server

- Java EE Server is an **application server**, whose core set of API and features are defined by Java EE.
- Java EE defines an architecture for implementing services through the use of a Java 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 Java EE platform



Web Applications & Components

- **Web application** is a dynamic extension of a web or application server.
  - Presentation-oriented (HTML, XML pages)
  - Service-oriented (Web services)
- **Web components** provide the dynamic extension capabilities for a web server:
  - Java 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.
- **Java EE Application Server** implements the entire Java 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
  - GlassFish
  - WildFly and Jboss
  - Oracle WebLogic
  - IBM WebSphere

JBoss Application Server 7

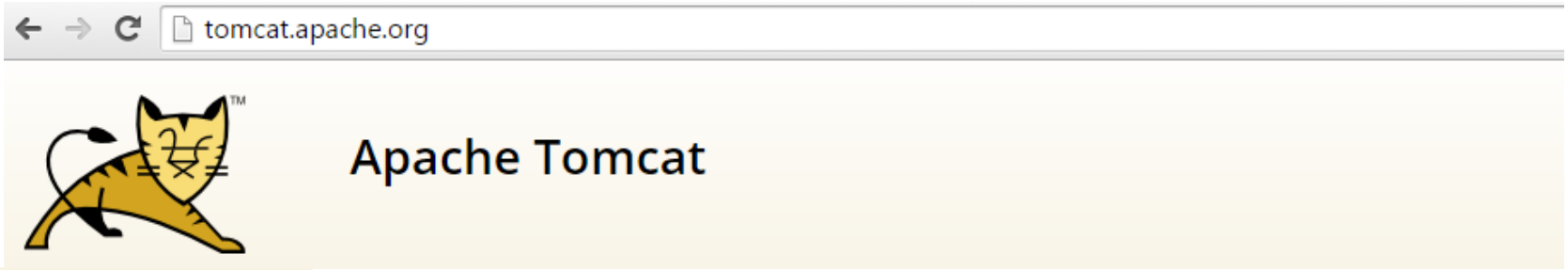
ORACLE
WebLogic

IBM
WebSphere

- Web Containers
  - Apache Tomcat
  - Jetty
  - Tiny Java Web Server (TJWS)

jetty://

Apache Tomcat



Apache Tomcat™ is an open source software implementation of the Java Servlet, JavaServer Pages, … .

Tomcat's Installed Directory Structure:

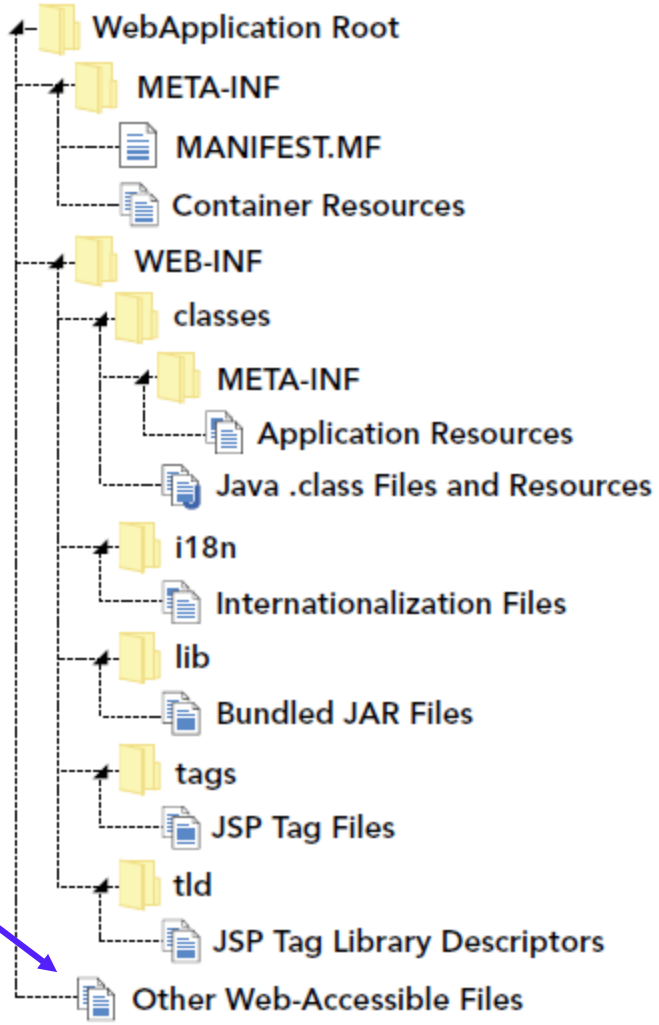
- **bin**: for Tomcat's binaries and startup scripts.
- **conf**: global configuration applicable to all the webapps.
- **lib**: Keeps the JAR-file that are available to all webapps.
- **logs**: contains the engine logfile Catalina ("Catalina" is servlet container in Tomcat).
- **webapps**: the default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

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 a XML file called a web application **deployment descriptor**
  - Its filename is ***web.xml***

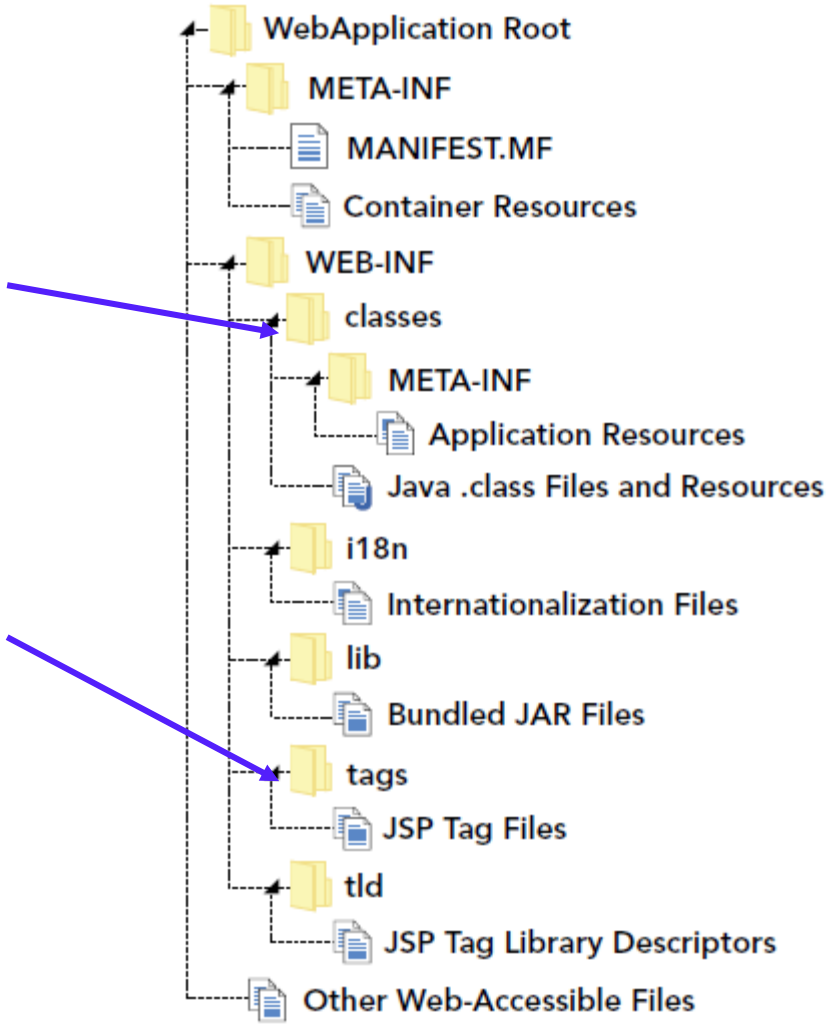
Web Application Structure

- The top-level directory of a web application is the *document root* of the application
- The document root contains:
  - JSP pages
  - client-side classes
  - client-side archives
  - static web resources



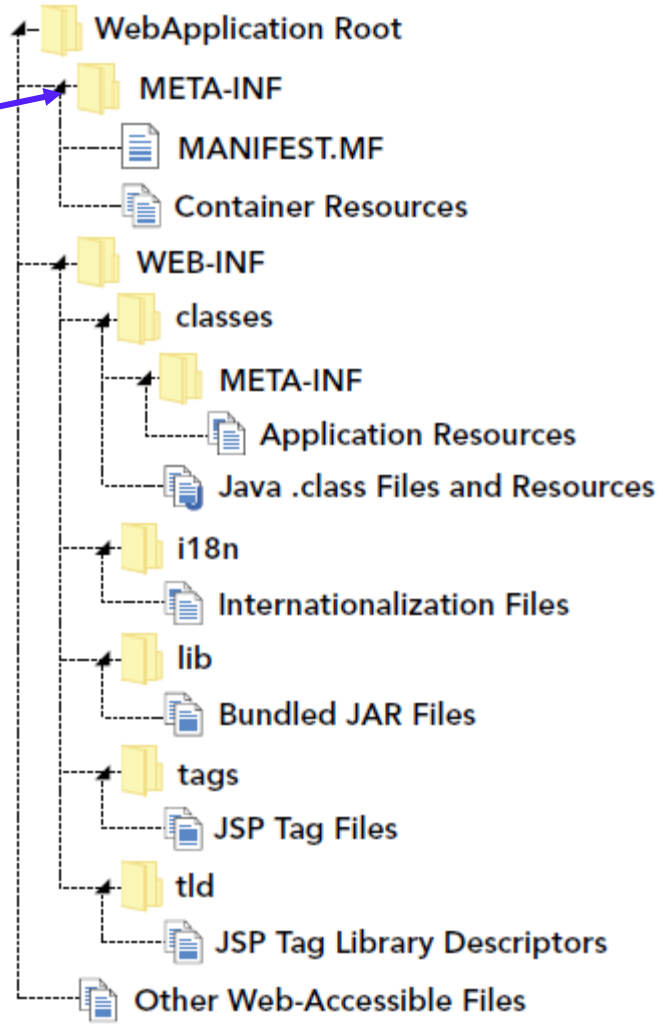
Web Application Structure

- The document root contains a subdirectory /WEB-INF/
- **classes**: server-side classes
  - servlets
  - utility classes
  - JavaBeans components
- **tags**: JSP tag files, which are implementations of tag libraries



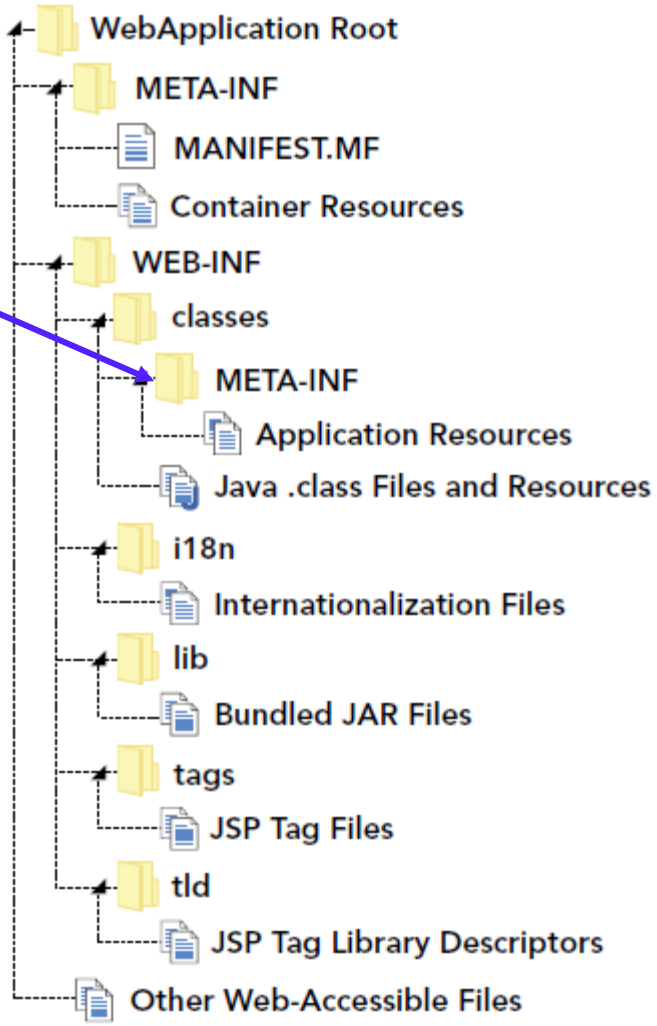
Web Application Structure

- The document root contains a subdirectory /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

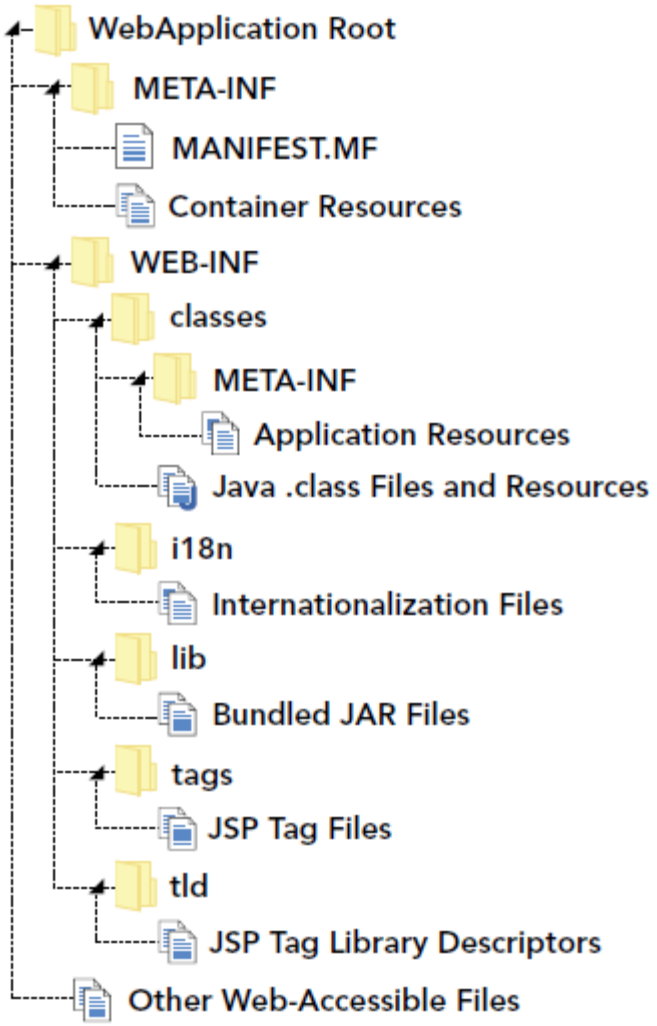
- The directory /WEB-INF/classes/ also contains ***another*** /META-INF/
  - On the application classpath
  - Some Java EE components require files in this directory.
  - E.g., **Java Persistence API**:
    - persistence.xml
    - orm.xml



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.

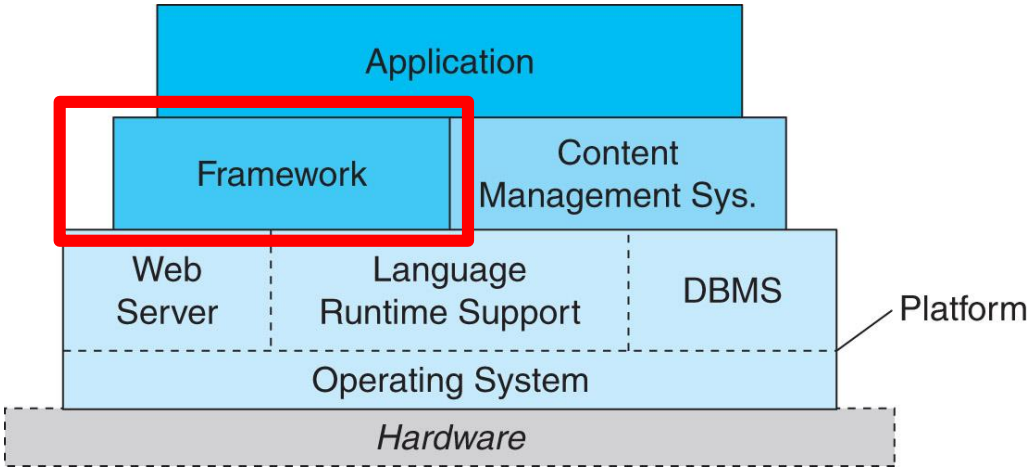
- An example of framework model is MVC (Model-View-Controller; more details will be given later in the course).

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

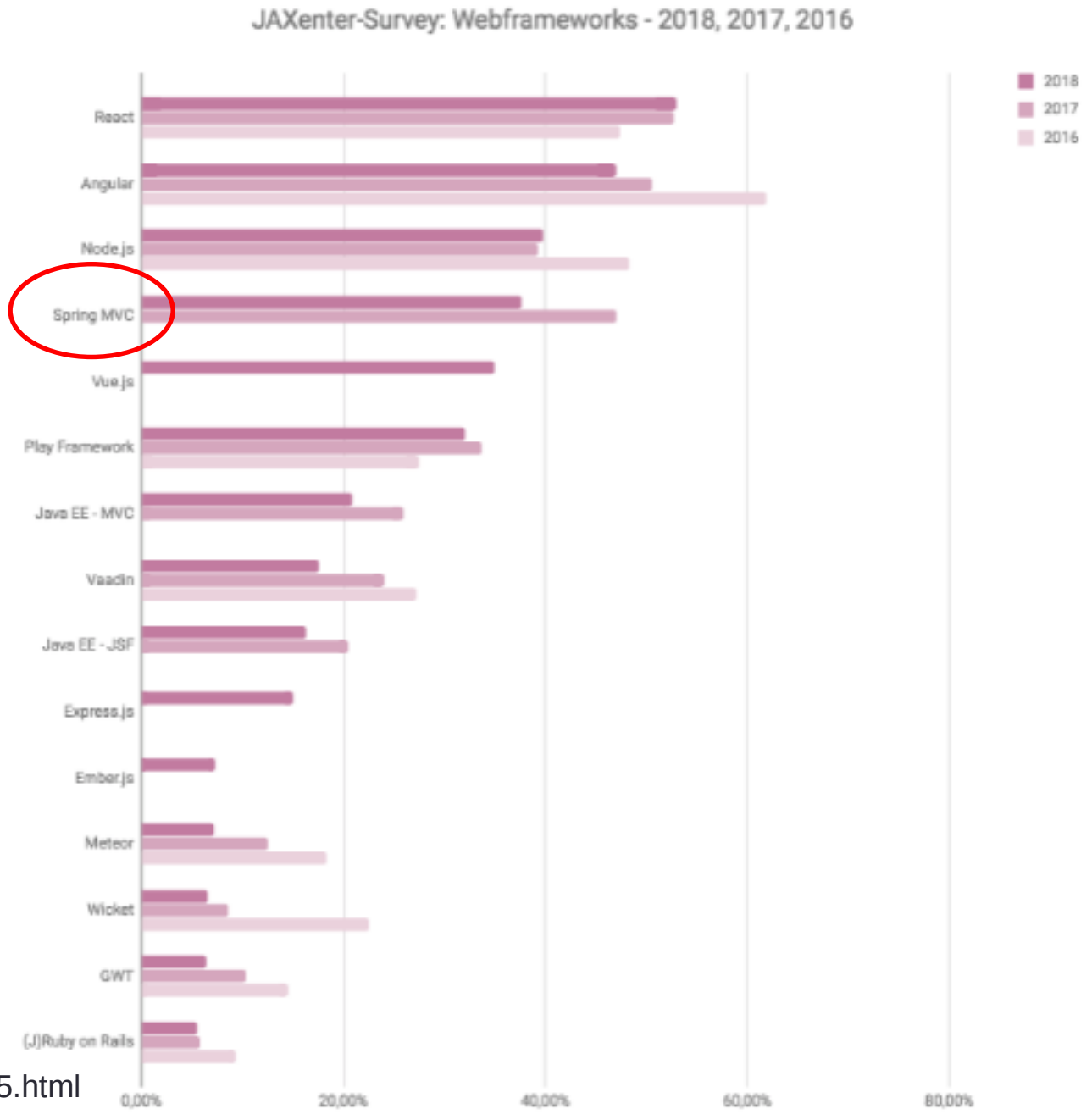
Web Application Framework

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



- Some existing web application frameworks include
  - Java : JavaServer Faces (JSF), Struts, Spring
  - JavaScript: React, Angular, Node.js
  - PHP: Laravel, CodeIngiter

Popular Web Frameworks



https://jaxenter.com/technology-trends-2018-frameworks-144575.html