CSBP461 - Internet Computing

Web Application Concepts

Elarbi Badidi

College of information Technology, UAE University

Spring 2025

Objectives

- Describe the evolution of Web applications
- Introduce the basics of HTTP protocol
- Introduce the Web application architectures
- Introduce the basics of Java Web technologies and applications

History

- The Internet is a global network of computer networks that join together millions of government, university, and private computers.
- This network provides a mechanism for communication where any type of data (text, images, video, etc.) can be exchanged between linked computers.
- These computers can be physically located on opposite ends of the globe, yet the data can be exchanged in a matter of seconds.

History, cont.

- Although often used interchangeably, the terms **Internet** and **WWW** are not the same thing.
- The Internet is the worldwide network of computers (and other devices like cell phones).
- The WWW refers to all of the information sources that a Web browser can access, which includes all of the global publicly available Web sites plus FTP (File Transfer Protocol) sites, USENET newsgroups, etc.
- **Email** is not considered to be part of the WWW, but is a technology that is made possible by the Internet.

History, cont.

- **1962** – Joseph Carl Robnett Licklider at the Massachusetts Institute of Technology (MIT) first proposed a global network of computers. Later that year he started working at the Defense Advanced Research Projects Agency (DARPA), then called the Advanced Research Projects Agency (ARPA), to develop his idea.
- **1961** through 1964 – Leonard Kleinrock, while working on a Ph.D. thesis at MIT, and later while working at the University of California at Los Angeles (UCLA), developed the concept of packet switching, which is the basis for Internet communications today.
- **1965** – While at MIT, Lawrence Roberts and Thomas Merrill used Kleinrock's packet switching theory to successfully connect a computer in Massachusetts with a computer in California over dial-up telephone lines (WAN)

History, cont.

- **1966** – Roberts started working at DARPA on plans for the first large-scale computer network called ARPANET, at which time he became aware of work done by Donald Davies and Roger Scantlebury of National Physical Laboratory (NPL) and Paul Baran of RAND Corporation that coincided with the packet switching concept developed by Kleinrock at MIT.
- DARPA awarded the contract for bringing ARPANET online to BBN Technologies of Massachusetts. Bob Kahn headed the work at BBN, which, in 1969, brought ARPANET (later called the Internet, in 1974) online at 50 kilobits per second (kbps), connecting four major computers at universities in the southwestern United States – UCLA, Stanford Research Institute, University of California at Santa Barbara, and University of Utah.

History, cont.

- **1970** – First host-to-host protocol for ARPANET was developed called **Network Control Protocol (NCP)**.
- **1973** – Vinton Cerf of Stanford and Bob Kahn of DARPA began to develop a replacement for NCP, which was later called **Transmission Control Protocol/Internet Protocol (TCP/IP)**.
- **1983** – ARPANET was transitioned to TCP/IP. TCP/IP still used today as the Internet's underlying protocol for connecting computers and transmitting data between them over the network.

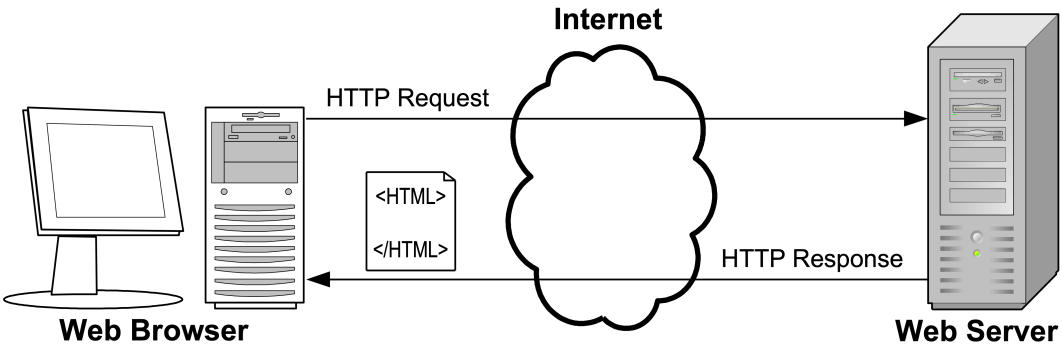
History, cont.

- **1991** – University of Minnesota developed the first user-friendly interface to the Internet called Gopher. Gopher became very popular because it allowed non-computer scientist types to easily use the Internet.
- **1989** – **Tim Berners-Lee** and others at the European Laboratory for Particle Physics (CERN) in Switzerland proposed a new protocol for information distribution on the Internet, which was based on hypertext, a system of embedding links in text to link to other text. This system was invented before Gopher but took longer to develop. Berners-Lee eventually created the **Hypertext Transfer Protocol (HTTP)** and the **HyperText Markup Language (HTML)**, coined the term "**World Wide Web**", developed the first Web browser and Web server, and went on to help found the **World Wide Web Consortium (W3C)**.

Web Architecture

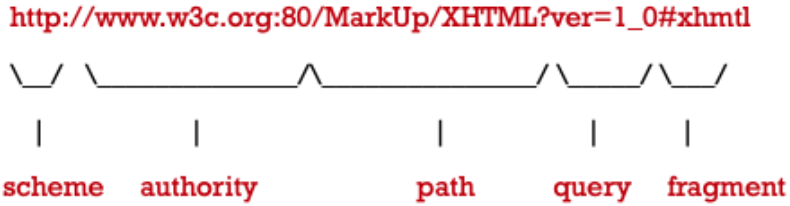
- **Web browser** is a user interface that knows how to send HTTP messages to, and receive HTTP messages from, a remote Web server.
- The Web browser establishes a TCP/IP connection with the Web server and sends it an **HTTP request message**.
- The Web server knows how to handle HTTP request messages to get data (text, images, movies, etc.) from the server and send it back to the Web browser, or process data that is submitted to the Web server from the Web browser (e.g., a username and password required for login).
- Internet users typically use Web browsers to get Web pages from the Web server in the form of **HTML documents**. The Web browser knows how to process the HTML document that it receives from the Web server, and display the results to the user via a graphical interface.

Web Architecture, cont.



URI / URL

- Web browsers always initiate TCP/IP connections with the Web server, never vice-versa.
- The Web browser identifies which Web server to make a connection with and what is being requested of the Web server with a **Uniform Resource Locator (URL)**.
- A URL is a classification of **Uniform Resource Identifier (URI)** that identifies a resource by its location. A URI is a more general term that encompasses all types of Web identifier schemes.



URI / URL, cont.

- **scheme** – Identifies the application-level protocol. Examples are **http, ftp, news, mailto, file, telnet**. The **://** following the scheme separates the scheme from the authority.
- **authority** – The host name or IP address of the Web server and an optional port number. The standard port for HTTP is 80, which most all computers already know, so it can be omitted. If the Web server is listening for connections to a different port, like 8080, then the port will need to be specified.
- **path** – A directory path to the resource. The concept of directory used here is the same as used with file systems. The ? (question mark) after the path is used to separate the query from the rest of URI and is not necessary if there is no query.

URI / URL, cont.

- **query** – The optional query is information that is to be interpreted by the Web server. It is used to provide additional information that is not included in the path or to submit text data to the Web server. The query can contain multiple **name=value** pairs separated by an & (ampersand symbol). Each name is separated from its associated value by an = (equal symbol).
- **fragment** – The optional fragment is used to identify a location within a document. This part is actually used by the Web browser, not the Web server, to bring you to a specific location in a document. The # (pound symbol) is used to separate the fragment from the rest of the URI.

Hypertext Transfer Protocol (HTTP)

- **HTTP** is a stateless protocol that supports **requests** followed by **responses** (request-response message exchange pattern)
- The HTTP protocol does not require the use of a Web browser; it simply describes how data can be exchanged over a network that uses TCP/IP (e.g., the Internet).
- By default, HTTP uses TCP/IP connections on port 80 of a computer, but other ports can, and often are used.
- An HTTP transaction begins with a request from the client and ends with a response from the server.
- An **HTTP request message** consists of three parts:
  1. A line defining the HTTP method, the URI requested, and HTTP version used;
  2. A list of HTTP request headers;
  3. The entity body.

Hypertext Transfer Protocol (HTTP), cont.

```
POST /catalog/prices HTTP/1.1[CRLF]
Host: www.somesite.com[CRLF]
Connection: close[CRLF]
Accept-Encoding: gzip[CRLF]
Accept: text/plain; text/html[CRLF]
Accept-Language: en-us,en[CRLF]
Accept-Charset: ISO-8859-1,utf-8[CRLF]
User-Agent: Mozilla/5.0 Gecko/20041107 Firefox/1.0[CRLF]
Referer: http://web-sniffer.net/[CRLF]
Content-Length: 16[CRLF]
Content-Type: application/x-www-form-urlencoded[CRLF]
[CRLF]
productId=ABC123[CRLF]
```

HTTP requests headers

Hypertext Transfer Protocol (HTTP), cont.

- CRLF tags in the previous slide represent the carriage return/linefeed (CRLF) characters
- Normally would not see them, however they are significant in an HTTP message so they are displayed here.
- CRLF characters are used to separate each line of the header and the header from the entity body.
- The message header includes every line before the first blank line (the line above with only a CRLF). The first blank line defines where the message header ends and the entity body begins.

Hypertext Transfer Protocol (HTTP), cont.

- Each line of the HTTP request message that occurs after the first line and before the blank line is called an **HTTP request header**.
- HTTP request headers contain useful information about the client environment and the entity body, such as:
  - the type of Web browser being used,
  - languages the browser is configured for, and
  - the length of the entity body.
- The first line of the HTTP request message contains the **HTTP method** (GET), the **URI** (/catalog/prices), and the **protocol/version** (HTTP/1.1).
- The HTTP method tells the Web server something about how the message is structured and what the client expects the Web server to do.
  - **GET** – Retrieves the data identified by the URL
  - **POST** – Submits data to the Web server in the entity body

Hypertext Transfer Protocol (HTTP), cont.

- An HTTP response message also contains three parts, like the request message:
  1. A line defining the version of the protocol used, a status code to identify if the request was successful, and a description;
  2. A list of HTTP response headers;
  3. The entity body.
- The message header is separated from the entity body by a blank line. Every line after the first line and before the first blank line is called an HTTP response header.
- The HTTP response headers contain useful information such as:
  - the length and type of data in the entity body,
  - the type of server that processed the request, and
  - information that can be used by the Web browser to determine how long it can cache the data.
- The entity body of the message may contain both text and binary data.

Hypertext Transfer Protocol (HTTP), cont.

```
HTTP/1.1 200 OK[CRLF]
Date: Sun, 13 Mar 2005 22:07:43 GMT[CRLF]
Server: Apache/2.0.49[CRLF]
Last-Modified: Sun, 17 Oct 2004 00:26:16 GMT[CRLF]
Content-Length: 70[CRLF]
Keep-Alive: timeout=15, max=99[CRLF]
Connection: Keep-Alive[CRLF]
Content-Type: text/html;charset=UTF-8[CRLF]
[CRLF]
<html>[CRLF]
<head>[CRLF]
<title>Example</title>[CRLF]
</head>[CRLF]
<body>[CRLF]
Hello World[CRLF]
</body>[CRLF]
```

HTTP response headers

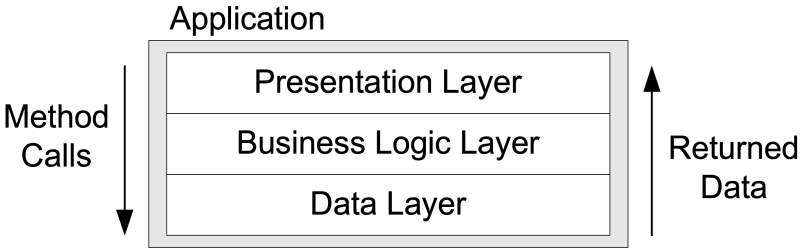
Entity body

Web Application Architecture

- **Web applications** are programs that execute business processes in response to Web requests.
- Most software systems can be partitioned into separate physical tiers and logical layers, where each part handles a different responsibly.
- The words **tier** and **layer** are often used interchangeably when discussing software architectures.
- We use the word tier to mean a physical partition. In other words, different tiers reside on different computers.
- We will use the word layer to mean a logical partition. Logical layers may or may not reside on different tiers.
- It is considered good practice to design and develop software applications with different logical layers.

Web Application Architecture, cont.

- Each layer only interacts with its neighboring layers.



- Presentation layer only interacts with the business logic layer; Business logic layer interacts with both the presentation layer and the data layer; Data layer only interacts with the business logic layer.
- Best practice is to let upper layers make calls to lower layers, but not vice-versa. This reduces coupling because each layer only knows (is coupled to) one other layer – the layer below. Method calls flow down; Only data flows up.

Web Application Architecture, cont.

Advantages to separating your application into layers:

- They are easier to understand, easier to maintain, scale easier, and provide the opportunity for the application code to be distributed on separate computers and networked together.
- When the application layers are well separated and loosely coupled, you can modify or maybe even completely rewrite one of the layers without incurring changes to the others.
- Additionally, this separation reduces the complexity for the developer because he can concentrate his focus on one responsibility of the application without having to think much about the other responsibilities.

Web Application Architecture, cont.

Disadvantages to separating your application into layers:

- Complexity and latency from having to pass commands across the network.
- If all of the code is executed on one computer, you don't have to write or use network code, integrate various computers, maintain various computers, etc. You can also more easily take steps to optimize the code to improve performance.
- The advantages of monolithic applications dissipate when you want to increase the users of the application, and you are also required to maintain the application on the user's computer.
- Every time a new user is added you have to install and maintain the application on that user's computer.

Web Application Architecture, cont.

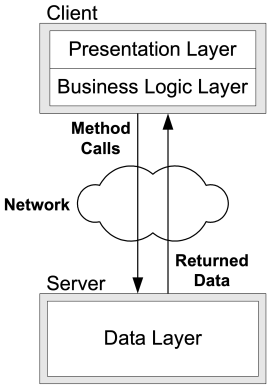
- Web applications are by nature distributed applications. They have at least two physical tiers, the client computer with the Web browser, and the computer hosting the Web server.
- Tiers and layers typically fall into one of three categories **presentation**, **business logic**, and **data**.
  - The presentation tier/layer handles presenting data to the user and accepting user inputs (often a graphical user interface).
  - The business logic tier/layer handles the business processing
  - The data tier/layer is used to permanently persist state. A typical data tier of today is a relational database, which typically resides and therefore executes on a separate computer from the rest of the system.

Web Application Architecture, cont.

- As Web applications and other distributed software applications have become larger, more complex, and asked to handle significantly more processing, distributing the layers to separate computers to improve performance has become more important.
- Distributing layers to different tiers can certainly hurt performance if it isn't done right, and it definitely increases complexity, but it is sometimes demanded by usage.
- The layers of an application can be distributed to different computers in a number of ways, which fall into one of essentially three different types of architectures – **two-tier architecture**, **three-tier architecture**, and **n-tier architecture**.
- Each tier is usually a different computer, but sometimes it is only a different process on the same computer.

Two-Tier Architecture

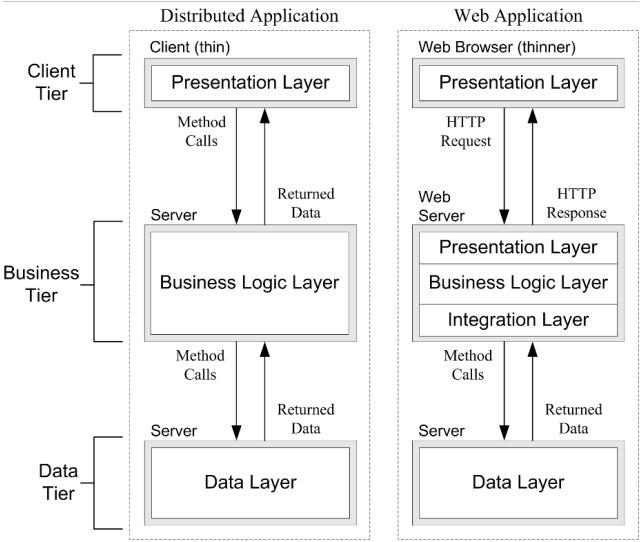
- A two-tier architecture is the simplest of the three, only consisting of two computers – a client and a server.
- The client is commonly referred to as a "**fat client**" because it handles a significant portion of the processing.



Disadvantages of Two-Tier Architecture

- The client computers must be powerful enough to handle the processing required by both the presentation layer and the business layer.
- The business layer usually needs to make a large amount of calls to the data layer, which resides on a different computer. As the amount of clients on the network increases, the amount of network traffic significantly increases because all of those clients are making lots of network calls to the data tier.
- Upgrading the clients to new versions of the software can become labor intensive. The business layer usually incurs the most changes over the lifetime of system. Since the business layer is on the client and there can be any number of clients, system administrators are dealt a significant burden.

Three-Tier Architecture



Three-Tier Architecture, cont.

- Consists of three computers – a **client**, a **business server**, and a **database server**. In general, the client handles the presentation layer, the business server handles the business layer, and the database server handles the data layer.
- However, the presentation layer is often split between the client tier and the business tier. Additionally, the business tier usually has an integration layer where part of the data layer resides on the client to provide an interface to database.
- In a typical Web application, the presentation (HTML and Cascading Style Sheets) is essentially constructed on a Web server and the Web browser just displays it to the user.
- The Web browser is considered a "thin client" because it does very little of the system processing.

Advantage of the Three-Tier Architecture

- Three-tier architectures solve many of the problems of two-tier architectures by separating the presentation layer from the business layer. The business layer can now reside on one or a few computers that are strictly maintained and controlled by the system administrators and developers.
- Only one or a few business servers are communicating with one or a few database servers, reducing the network traffic. Since the presentation layer doesn't have to make many calls to the business layer, the amount of clients can grow to a large extent without significantly impacting network traffic, and thus performance.
- Additionally, the business server and database server can be upgraded without affecting the clients or drastically increasing the cost of the system.

n-Tier Architecture

- n-tier architectures break down the application into even more tiers.
- Additional separation typically occurs within the presentation layer and the business layer, but can also happen in the data layer.
- Additional computers can be added as demands increase and development and maintenance responsibilities can be divided among business units.
- The increased division of responsibilities has lead to the concept of **service-oriented architectures** where different functional units of a business process are actually hosted on different Web sites (located anywhere in the world).

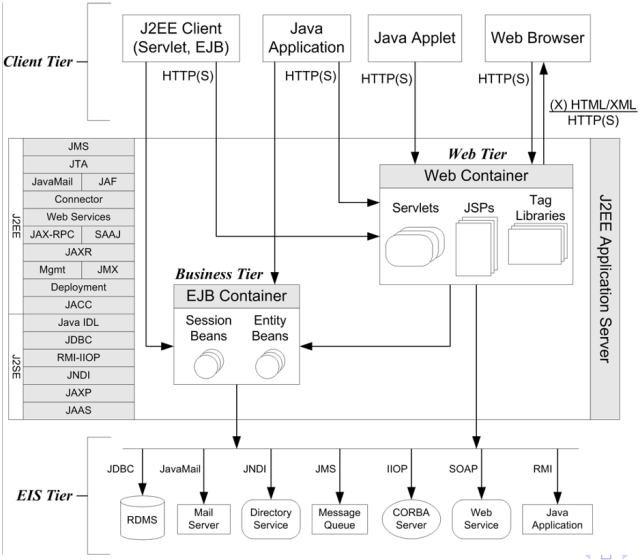
Java Servlets

- The first Java solution to dynamic Web applications was **Java Servlets**. A Java Servlet is a server-side program that services HTTP requests and returns HTTP responses.
- Java Servlets use a threading model to process multiple requests, which allows them to require fewer resources than independent processes, thereby improving performance.
- Java Servlets are completely object-oriented, independent of the underlying platform, less prone to memory leaks, and they run in a virtual machine that imposes security constraints to help minimize programs wreaking havoc on your system.

Java 2 Platform, Enterprise Edition (J2EE)

- Java provides a full suite of technologies for enterprise Web development called **J2EE** (APIs that require impl.).
- J2EE is managed by Sun Microsystems and developed under the Java Community Process (JCP), which includes participation from a variety of leading industry developers, like IBM, Oracle, SAP, BEA, and Macromedia.
- J2EE is a standard for component-based design, development, assembly, and deployment of n-tier distributed applications.
- It offers a reusable component model, a security model, transaction control, and support for Web services. The applications you develop can be moved to a different vendor implementation with little or no changes. You can choose a vendor based upon things like pricing, performance, hardware requirements, support services, etc.

J2EE Architecture



J2EE Architecture, cont.

- J2EE uses a distributed multi-tiered application model that consists of four distinct tiers – the **client tier**, the **Web tier**, the **business tier**, and the **enterprise information systems (EIS) tier**.
- Java Servlets, JSPs, and tag libraries execute on the Web tier in a runtime environment called the Web Container.
- EJBs execute on the business tier in a runtime environment called the EJB Container.
- These containers provide lifecycle management for the components, various services like security and transaction management, and a suite of useful APIs.
- A J2EE application server is the combination of a Web container and an EJB container.