Subject 1 - Service Architecture:

Distributed Applications

- Basic concepts of web programming

  - What is a server?

A server is a **computing device (a powerful computer)** that accepts and responds to requests made by another program, known as client.

Servers used to **manage network resources** and are useful for intense calculations.

They are always on because **constantly required**.

Metaphor: a customer (client) ordering a package then the mailman (server) delivering it to them or someone else.

  - What is a web server?

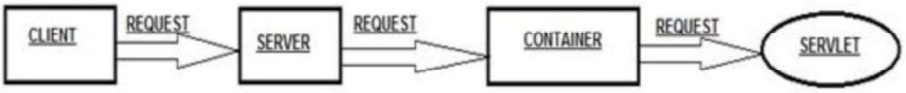
Computer or collection of computers used to **deliver web pages** and other content to multiple users. (Requests from clients to access web pages).

  - What is a web container?

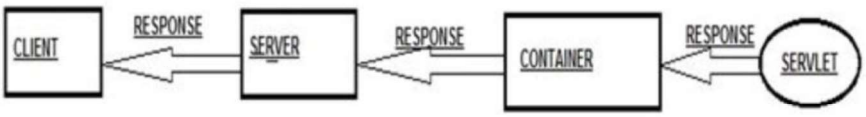
Web container is the **interface between web components and the web server**. A web component is a set of features that allow for the creation of reusable widgets or components in web documents and web applications (example: servlet).

The container:
- Manages component's **lifecycle**
- Dispatches **requests** to app components
- Provides **interfaces** to context data

1. Request made by client to server



2. Response received by client



- What is an application server?

An application server is a **software framework** that provides aids **to create web applications and a server environment** to run. It is a type of server designed to install, operate and host applications and services for users, IT services and organizations. It facilitates the hosting and delivery of applications.

• Different models of distributed architectures (1 tier, 2-tiers, 3-tiers, n-tiers)

The software architecture is divided in tiers. A tier can also be referred to as a "layer".

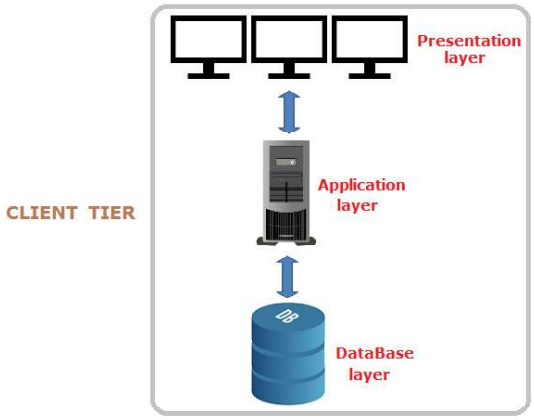
There are 3 layers:

- Presentation layer (=client layer): top most layer of an application**, layer seen when we use a Software**. With this layer we can access webpages. Main function = communicate with Application layer (keyboard actions…)
- Business layer (=application layer/logic layer): **interacts with database layer**, performing detailed processing
- Data layer: the **data is stored** in this layer

One-tier architecture = Standalone application

This architecture has **all the layers** (presentation + application + database) in a **single SW package**. Ex of app: MP3 player, MS Office…

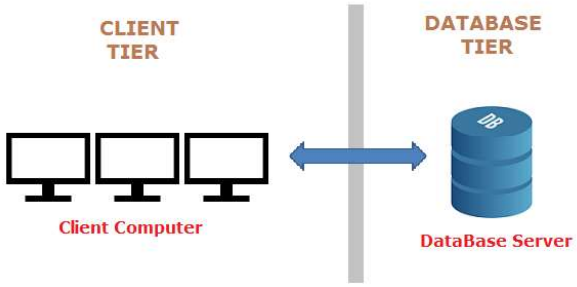
- **Client system** handles **Presentation + Application + Database layer**.



Two-tier architecture = Client-Server application

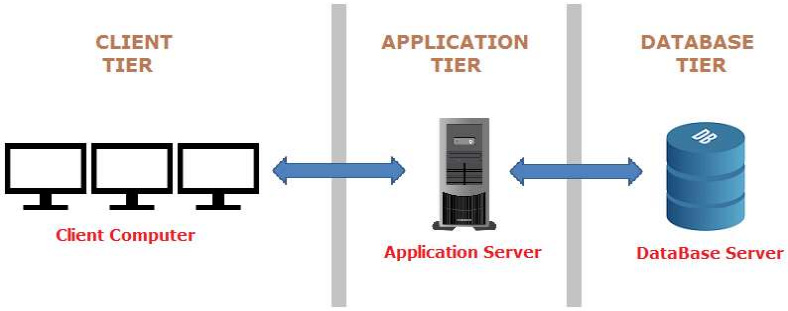
Just the client app and the data. Communication between the client and the server.

- **Client system** handles **Presentation + Application layer**.
- **Server system** handles **Database layer**



Three-tier architecture = Web Based application

- **Client system** handles **Presentation layer**.
- **Application server** handles **Application layer**
- **Server system** handles **Database layer**



N-tier architecture = Distributed application

It is **similar to the three-tier architecture,** but number of application servers are increased and represented in individual tiers. The business logic is distributed.

- Advantages and drawbacks of these models

One-tier architecture = Standalone application

(+) = easy to implement/optimize performance/no compatibilities errors/costless to develop

(–) = no distributed access for data resources/higher maintenance/high cost for central mainframe

Two-tier architecture = Client-Server application

(+) = applications easily developed/accurate and fast prototyping

(–) = dispersion of applications/change of database structure/min logic sharing/scalability/ security/control the SW version/environment changing rules rapidly

Three-tier architecture = Web Based application

(+) = hidden database structure/redundant server availability/reduced distribution/enhanced security/improve data integrity/enhanced re-usage/improve scalability

(–) = complex communication (clients to middle tier to server)/fewer tools

N-tier architecture = Distributed application

(+) = all advantages of the 3-tier model/increased performance

(–) = complex structure difficult to implement or maintain

- n-tiers architecture:
    - Java EE architecture (n-tiers architecture example)

It is a **set of specifications**, with specifications for enterprise features such as distributed computing and web services. Java EE applications are run on reference runtimes, that can be microservices or application servers, which handle transactions, security, scalability, concurrency and management. Java EE is defined by its specification. The specification defines **APIs** and their **interactions**.

- What is a JSP, Servlet, javabean session and javabean entity

- **JSP** = Java Server Page: technology for **controlling the content of appearance** of Web Pages through the use of servlets. It calls a **Java program executed by the web server**.
- **Servlet**: a small program that runs on a server. It responds to a network request. Basically, servlets are usually used to implement web applications.
- **Javabean**: reusable software programs that you can develop and assemble easily to create sophisticated applications (developers use Javabean to design and develop customized and reusable business logic)
- **Session bean**: business process objects that perform actions (action may be opening an account, transferring funds, performing a calculation)
- **Entity beans**: data objects that represent the real-life objects on which session beans perform actions.

- DAO (Data Access Object): what is it and what for?

An object that provides an **abstract interface** to some type of **database**. The DAO provides some **specific data operations** without exposing details of the database.