2. Programming Architecture [Credit: 3 hrs]

Programming Architecture is both the process and the product of planning, designing and constructing programs or software. It's a design pattern that is followed in software development. **Software architecture** refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Object-oriented design patterns typically show relationships and Interactions between classes or objects, without specifying the final application classes or objects that are involved. One of these design patterns is Model-View-Controller (MVC). The programming language Smalltalk first defined the MVC concept it in the 1970's. Since that time, the MVC design idiom has become commonplace, especially in object oriented systems.

2.1. MVC

MVC is a software architecture pattern which allows us to split a software application into three interconnected parts. These parts are Model, View, and Controller, in short, they are often known as MVC. Most of the languages like Java, PHP, Python, C# etc use this pattern to develop the applications. In Java, it is known as Spring - MVC Framework, in PHP it is known as cake PHP and so on. By using the same concept, Microsoft introduced a framework called ASP.Net MVC.

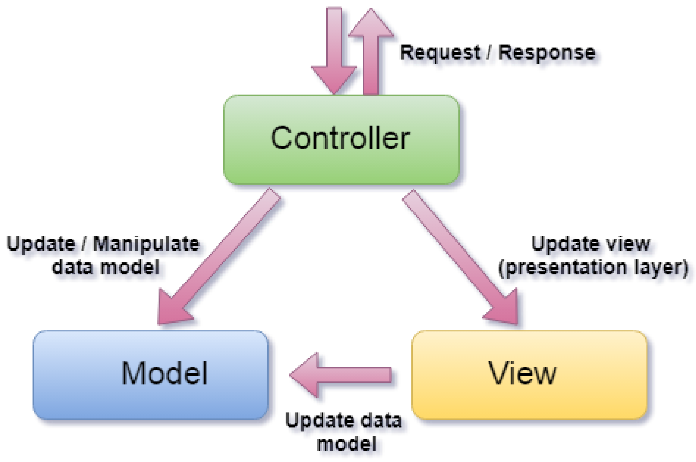

Request / Response
Controller
Update / Manipulate data model
Update view (presentation layer)
Model
View
Update data model

Figure: MVC architecture

Model:

Model is a data and business logic. Model represents shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database. When a View is designed generally we attach a Model to it. The view requests the data from Model and the Model responds to this request.

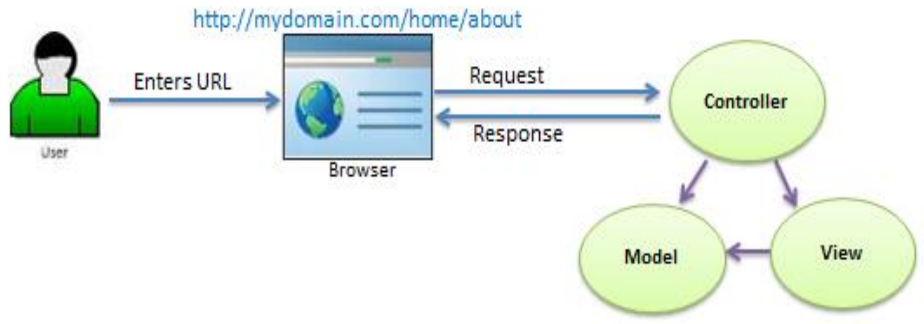
View:

View is a user interface. View display data using model to the user and also enables them to modify the data.

Controller:

Controller is a request handler. Controller handles the user request. Typically, user interact with View, which in-turn raises appropriate URL request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response.

Request/Response in MVC Architecture



As per the above figure, when the user enters a URL in the browser, it goes to the server and calls appropriate controller. Then, the Controller uses the appropriate View and Model and creates the response and sends it back to the user.

Advantages of MVC

1) **Faster development process:** MVC supports rapid and parallel development. With MVC, one programmer can work on the view while other can work on the controller to create business logic of the web application. The application developed using MVC can be three times faster than application developed using other development patterns.
2) **Ability to provide multiple views:** In the MVC Model, you can create multiple views for a model. Code duplication is very limited in MVC because it separates data and business logic from the display.
3) **Support for asynchronous technique:** MVC also supports asynchronous technique, which helps developers to develop an application that loads very fast.
4) **Modification does not affect the entire model:** Modification does not affect the entire model because model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture.
5) **MVC model returns the data without formatting:** MVC pattern returns data without applying any formatting so the same components can be used and called for use with any interface.
6) **SEO friendly Development platform**: Using this platform, it is very easy to develop SEO-friendly URLs to generate more visits from a specific application.

Disadvantages of MVC

1) Increased complexity
2) Inefficiency of data access in view
3) Difficulty of using MVC with modern user interface.
4) Need multiple programmers for parallel development
5) Knowledge on multiple technologies is required.
6) Developer must have knowledge of client side code and html code.

2.2. Client-server traditional model

Client-Server Architecture is a shared architecture system where loads of client-server are divided. The client-server architecture is a centralized resource system where server holds all the resources. The server receives numerous performances at its edge for sharing resources to its clients when requested. Client and server may be on the same or in a network. The server is profoundly stable and scalable to return answers to clients. This Architecture is Service Oriented which means client service will not be interrupted. Client-Server Architecture subdues network traffic by responding to the inquiries of the clients rather than a complete file transfer. It restores the file server with the database server.

*A **client-server architecture** divides an application into two parts, 'client' and 'server'. Such an application is implemented on a computer network, which connects the client to the server. The server part of that architecture provides the central functionality: i.e., any number of clients can connect to the server and request that it performs a task. The server accepts these requests, performs the required task and returns any results to the client, as appropriate.*

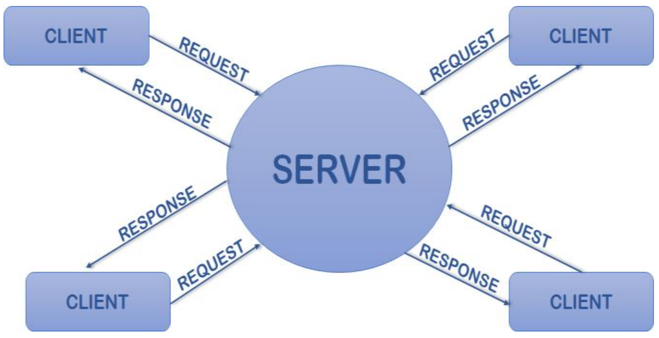


Figure: Client-Server model

Types of Server:

There are two types of servers: Iterative and concurrent.

1. **Iterative Server**

   This is the simplest form of server where a server process serves one client and after completing first request then it takes request from another client. Meanwhile another client keeps waiting.

2. **Concurrent server**

   This type of server runs multiple concurrent processes to serve many requests at a time. Because one process may take longer and another client cannot wait for so long.

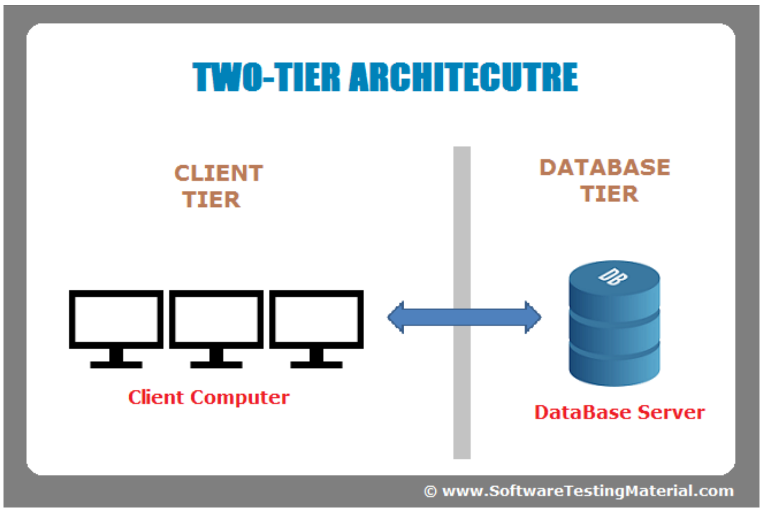
2.3. N-tier architecture

2.3.1) 2-tier Client Server Architecture

The Two-tier architecture is divided into two parts:

1. Client Application (Client Tier)
2. Database (Data Tier)

Client system handles both Presentation and Application layers and Server system handles Database layer. It is also known as client server application. The communication takes place between the Client and the Server. Client directly interacts with the server. Client system sends the request to the Server system and the Server system processes the request and sends back the data to the Client System. The server does not call on another application in order to provide part of the service.



Pros:

1. Easy to maintain and modification is bit easy.
2. Communication is faster.

Cons:

1. In 2-tier architecture, application performance will be degraded upon increasing the users.
2. Cost-ineffective

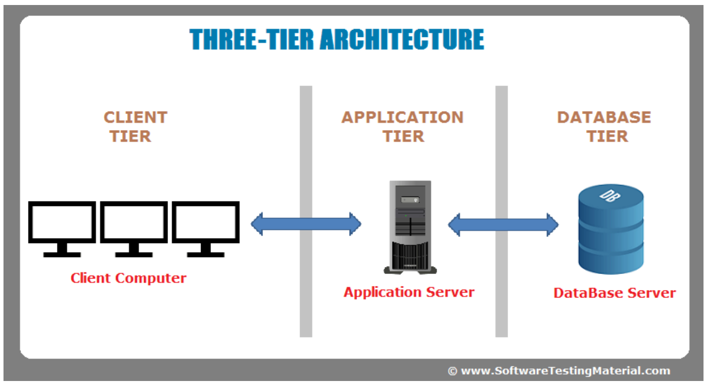
	Advantages	Disadvantage
Development Issues:	• Simple structure •Easy to setup and maintain	Complex application rules difficult to implement in database server - requires more code for the client Complex application rules difficult to implement in client and have poor performance Changes to business logic not automatically enforced by a server changes require new client side software to be distributed and installed
Performance:	Adequate performance for low to medium volume environments • Business logic and database are physically close, which provides higher performance	Inadequate performance for medium to high volume environments, since database server is required to perform business logic. This slows down database operations on database server.

2.3.1) 3-tier Architecture

The Three-tier architecture is divided into three parts:

1. Presentation layer (Client Tier)
2. Application layer (Business Tier)
3. Database layer (Data Tier)

Client system handles Presentation layer, Application server handles Application layer and Server system handles Database layer.



In this architecture, one more software sits in between client and server. This middle software is called **middleware**. Middleware are used to perform all the security checks and load balancing in case of heavy load. A middleware takes all requests from the client and after doing required authentication it passes that request to the server. Then server does required processing and sends response back to the middleware and finally middleware passes this response back to the client.

Middle tier is also called as business logic tier or service tired. In the simple client-server solution the client was handling the business logic and that makes the client "**thick**". A thick client means that it requires heavy traffic with the server, thus making it difficult to use over slower network connections like Internet and Wireless (LTE, 3G, or Wi-Fi).

By introducing the middle layer, the client is only handling presentation logic. This means that only little communication is needed between the client and the middle tier making the client "**thin**" or "thinner". An example of a thin client is an Internet browser that allows you to see and provide information fast and almost with no delay.

Pros:

1. High performance. Lightweight persistent objects.
2. Scability: each tier cab scale horizontally.
3. Performance: because the presentation tier can cache requests, network utilization is minimized, aand the load is reduced on the Application and Data tiers.
4. High degree of flexibility in deployment platform and configuration
5. Better re-use
6. Improve data integrity
7. Improved security: client cannot directly access to database.
8. Easy to maintain and modification is bit easy, won't affect other modules
9. Application performance is good.

Cons:

1. Increases complexity/effort

3-tier Pros and Cons:

Advantage	**Disadvantage**
Development Issues: • Complex application rules easy to implement in application server • Business logic off-loaded from database server and client, which improves performance • Changes to business logic automatically enforced by server – changes require only new application server software to be installed • Application server logic is portable to other database server platforms by virtue of the application software	*Development Issues:* • More complex structure • More difficult to setup and maintain.
Performance: • Superior performance for medium to high volume environments.	*Performance:* • The physical separation of application servers containing business logic functions and database servers containing databases may moderately affect performance.

N-tier Architecture:

Another layer is N-Tier application. N-Tier application AKA Distributed application. It is similar to three tier architecture but number of application servers are increased and represented in individual tiers in order to distribute the business logic so that the logic will be distributed.

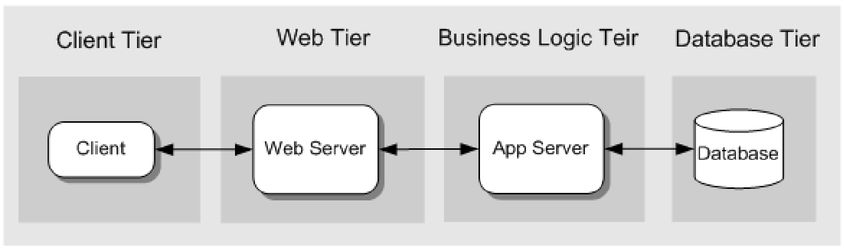


Figure: N-tier architecture

As more users access the system, a three-tier solution might not be sufficient, we might need to scale up. In N-tier architecture, we can add as many middle tiers (running on each own server) as needed to ensure good performance (N-tier or multiple-tier). Security is also the best in the N-tier architecture because the middle layer protects the database tier. There is one major drawback to the N-tier architecture and that is the additional tiers increase the complexity and cost of the installation.

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is client-server architecture in which, the **presentation**, the **application processing** and the **data management** are logically separate processes. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of "multi-tier architecture" refers to three-tier architecture.

An example of N-tier architecture is web-based application. A web-based application might consist of the following tiers.

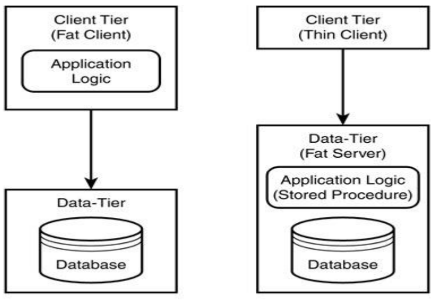
1. Tier 1: a client tier presentation implemented by a web browser.
2. Tier 2: a middle-tier distribution mechanism implemented by a web server.
3. Tier 3: a middle-tier service implemented by a set of server side scripts.
4. Tier 4: a data-tier storage mechanism implemented by a relational database.

Reference: https://www.softwaretestingmaterial.com/software-architecture/
http://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/

2.4. Comparison among 2-tier, 3-tier and N-tier architecture

	1-tier	2-tier	3-tier
Benefits	Very simple Inexpensive No Server needed	Good Security More Scalable Faster execution	Exceptional security Faster execution Thin client Very scalable
Issues	Poor security Multi user issues	More costly More complex Thin client	Very costly Very complex
Users	Usually 1(or few)	2-100	50-2000(+)

2.5. Thin and Thick clients



2-Tier Architecture
(Client/Server)

Fat-Client/Fat-Server:

Client does most of the processing. User Interface and all data processing / Business logic are done on client. The Data Service Layer is the actual Database Server, which handles the storage or services the data.

Advantages of Thick clients:

- Lower server requirements
- Working offline
- Better multimedia performance
- More flexibility
- Using existing infrastructure
- Higher server capacity

Thin-Client/Fat-Server:

Client has less processing, usually UI & UI-Centric *Business Objects*. The *Data-Centric Business Objects,* which handles the data access code, resides on the Data Service Layer or Database Server

Advantages:

- Cost saving and less work load
- Simplified management
- Enhanced security

References:

https://www.profolus.com/topics/thick-client-vs-thin-client-advantages-and-disadvantages/
https://www.webopedia.com/DidYouKnow/Hardware_Software/thin_client.asp

Exam questions

1. What is tier in software? Explain about MVC and 3-tier. [2018 Spring]
2. Illustrate MVC with diagram? What are the differences between 2-tier and 3-tier architecture of software development? [2018 Fall]
3. Differentiate between: [2017 fall]
   a) Thin and Thick client
   b) 2-tier, 3-tier and N-tier architecture.
4. What is the difference between 3-layer architecture and MVC architecture? [2017 spring]
5. What do you mean by Thick and Thin client? Explain N-tier architecture. [2016 fall]
6. What is MVC architecture? Explain pros and cons of 2-tier and 3-tier architecture. [2016 spring]
7. Write merits and demerits of MVC. What is the difference between 2-tier and 3-tier architecture? [2015 spring]
8. Write merits and demerits of MVC. Why is MVC a three tier architecture? [2015 fall]
9. What is MVC architecture? How MVC processes the request and response? [2014 spring]
10. What is client-server model? Describe the relationship between Model, view and controller in MVC architecture. [2014 fall]
11. What is MVC architecture? Explain pros and cons of 2-tie, 3-tier and N-tier architecture. [2013 spring]