**Chapter-2**

**MVC(Model View Controller)**

MVC is a pattern for developing software applications which divide the representation of information from the user's interaction with it.MVC are well architected, testable and easy to maintain. MVC-based applications contain:

* **M**odels: Classes that represent the data of the application and that use validation logic to enforce business rules for that data.Itis the part of the application that handles the logic for the application data.Often model objects retrieve data (and store data) from a database.
* **V**iews: A view display data(database record)that dynamically generate HTML responses.Multiple views of the same data are possible. It is the parts of the application that handles the display of the data.Most often the views are created from the model data.
* **C**ontrollers: Classes that handle incoming browser requests(user interaction), retrieve model data, and then specify view templates that return a response to the browser.Typically controllers read data from a view, control user input, and send input data to the model.



The MVC model defines webapplications with 3 logic layers:

The business layer (Model logic),The display layer (View logic) and The input control (Controller logic).

**Client–server model**

Client/server architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or Internet connection. This system shares computing resources.

Client/server architecture may also be referred to as a **networking computing model** because all the requests and services are delivered over a network.

Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

The client–server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. The model assigns one of two roles to the computers in a network: Client or server. A server is a computer system that selectively shares its resources; a client is a computer or computer program that initiates contact with a server in order to make use of a resource. Data, CPUs, printers, and data storage devices are some examples of resources.

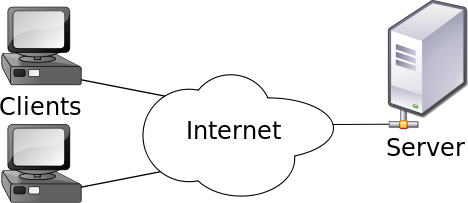
Clients and servers exchange messages in a request-response messaging pattern: The client sends a request, and the server returns a response. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. ***All client-server protocols operate in the application layer.***

In the client-server model, the server is a centralized system. The more simultaneous clients a server has, the more resources it needs.

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common.

Client-server architectures are sometimes called two-tier architectures.

Technology that separates computers and application software into two categories clients, and servers to better employ available computing resources and share data processing loads. A client computer provides the user interaction-facility (interface) and some or all application processing, while the a server computer might provide high-volume storage capacity, heavy data crunching, and/or high resolution graphics. Typically, several client computers are connected through a network (or networks) to a server which could be a large PC, minicomputer, or a mainframe computer. Every computer connected to a website acts as a client while the website's computer acts as a server. Also called client-server environment.



**Tier and Layer**

Tier usually means the physical deployment computer. Usually an individual running server is one tier.

Layer usually means logic software component group mainly by functionality; layer is used for software development purpose. Layer software implementation has many advantages and is a good way to achieve N-Tier architecture. Each layer may run in an individual tier. However, multiple layers may also be able to run in one tier.

### 2-Tier Architecture

Either presentation layer and application layer can only run in one computer, or application layer and data layer can only run in one computer. The whole application cannot run in more than 2 computers. The client that runs the application and the server that handles the database back-end.Server able to load many clients allowing more users to work on the system at the same time. 2-tier is also called thick client.

### C:\Users\generic\Desktop\2tier1.jpg

Client directly interact with the server. This type of architecture may have some security holes and performance problems. Internet Explorer and Web Server works on two tier architecture. Here security problems are resolved using Secure Socket Layer(SSL).

**Advantages**: simple and fast for a lower number of users due to fewer processes and fewer tiers; low cost for hardware, network, maintenance and deployment due to less hardware and network bandwidth needed.

**Disadvantages**: will have issues when the number of users gets big; has limitation to solve issues like security, scalability, fault tolerance and etc because it can be deployed in only 1 or 2 computes.

### 3-Tier 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.

The simplest of N-Tier architecture is 3-Tier which typically contains following software component layers listed from the top level to the low level: presentation layer, application layer and data layer.

A layer can access directly only the public components of its directly-below layer. For example, presentation layer can only access the public components in application layer, but not in data layer. Application layer can only access the public components in data layer, but not in presentation layer. Doing so can minimize the dependencies of one layer on other layers. This dependency minimization will bring benefits for layer development/maintenance, upgrading, scaling and etc. Doing so also makes the tier security enforcement possible. For example, the client layer cannot access the data layer directly but through the application layer, so data layer has a higher security guarding. Finally, doing so can also avoid cyclic dependencies among software components.

In order to claim a complete 3-Tier architecture, all three layers should able to run in separate computers. Practically, these three layers can also be deployed in one computer (3-Tier architecture, but deployed as 1-Tier). This may also called thin client.

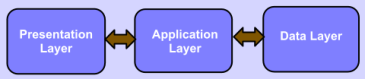
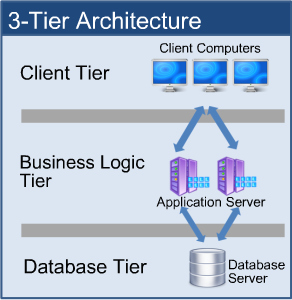


Fig 1: 3-Tier Architecture

**Presentation layer**: a layer that users can directly access , such as desktop UI, web page and etc. Also called client.

**Application layer**: this layer encapsulates the business logic (such as business rules and data validation), domain concept, data access logic and etc. Also called middle layer.

**Data layer**: the external data source to store the application data, such as database server. For N-Tier architecture, we need to use the non-embedded database server, such as SQL server, Oracle, DB2, MySQL or PostgreSQL. The non-embedded database server can be run in an individual computer. Whereas, the embedded type databases, such as Microsoft access, dbase and etc, cannot run in an individual computer, and then cannot be used as the data layer of the 3-Tier architecture.

****

**Comparisions of 2-tier and 3- tier**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2-Tier | 3-Tier |  |
| Benefits | Good security  More scalable  Faster execution | Exceptional securityFastest execution “Thin” client  Very scalable |  |
| Issues | More costly  More complex  “Thick” client  Less users | Very costly  Very complex  More users than 2 tier |  |

**N-Tier Architecture**

Some layers in 3-Tier can be broken further into more layers. These broken layers may be able to run in more tiers. For example, application layer can be broken into business layer, persistence layer or more. Presentation layer can be broken into client layer and client presenter layer. In order to claim a complete N-Tier architecture, client presenter layer, business layer and data layer should be able to run in three separate computers (tiers). Practically, all these layers can also be deployed in one compute (tier).



Fig12: N-Tier Architecture

***Client layer***: this layer is directly involved with users. There may be several different types of clients coexisting, such as WPF, Window form, HTML web page and etc.

***Client presenter layer***: contains the presentation logic needed by clients, such as ASP .NET MVC in IIS web server. Also it adapts different clients to the business layer.

***Business layer***: handles and encapsulates all of business domains and logics; also called domain layer.

***Persistence layer***: handles the read/write of the business data to the data layer, also called data access layer (DAL).

***Data layer***: the external data source, such as a database.

**N-Tier Architecture - Advantages**

* Scalable
* Better and finer security control to the whole system
* Better fault tolerance ability:
* Independent tier upgrading and changing without affecting other tiers
* Friendly and efficient for development:
* Friendly for maintenance:
* Friendly for new feature addition:
* Better reusability

**The Disadvantages of the N-Tier Deployment**

* Much more complicated to design and model.
* The performance of the whole application may be slow if the hardware and network bandwidth aren’t good enough because more networks, computers and processes are involved.
* More cost for hardware, network, maintenance and deployment because more hardware and better network bandwidth are needed.

**Thin Clients**

A thin client is designed to be especially small so that the bulk of the data processing occurs on the server. Although the term thin client often refers to software, it is increasingly used for the computers, such as network computers and Net PCs, that are designed to serve as the clients for client/server architectures. A thin client is a network computer without a hard disk drive. They act as a simple terminal to the server and require constant communication with the server as well.

Thin clients provide a desktop experience in environments where the end user has a well-defined and regular number of tasks for which the system is used. Thin clients can be found in medical offices, airline ticketing, schools, governments, manufacturing plants and even call centers. Along with being easy to install, thin clients also offer a lower total cost of ownership over thick clients.

a thin client is an Internet browser that allows you to see and provide information fast and almost with no delay. 3-tier or N-tier is also called thin client

**Thick Clients**

In contrast, a thick client (also called a fat client) is one that will perform the bulk of the processing in client/server applications. With thick clients, there is no need for continuous server communications as it is mainly communicating archival storage information to the server. As in the case of a thin client, the term is often used to refer to software, but again is also used to describe the networked computer itself. If your applications require multimedia components or that are bandwidth intensive, you'll also want to consider going with thick clients. One of the biggest advantages of thick clients rests in the nature of some operating systems and software being unable to run on thin clients. Thick clients can handle these as it has its own resources.

A thick client means that it requires heavy traffic with the server, thus making it difficult to use over slower network connections. sometimes 2-tier is also called thick client.

**Thick vs. Thin - A Quick Comparison**

|  |  |
| --- | --- |
| **Thin Clients** | **Thick Clients** |
| -Data is typically stored on servers.  Easy to deploy as they require no extra or specialized software installation  - Needs to validate with the server after data capture  - If the server goes down, data collection is halted as the client needs constant communication with the server  - Cannot be interfaced with other equipment (in plants or factory settings for example)  - Clients run only and exactly as specified by the server  - More downtime  -Portability in that all applications are on the server so any workstation can access  - Opportunity to use older, outdated PCs as clients  - Reduced security threat | -Data may be stored locally.  -More expensive to deploy and more work for IT to deploy  - Data verified by client not server (immediate validation)  - Robust technology provides better uptime  - Only needs intermittent communication with server - More expensive to deploy and more work for IT to deploy  - Require more resources but less servers  - Can store local files and applications  - Reduced server demands  - Increased security issues |