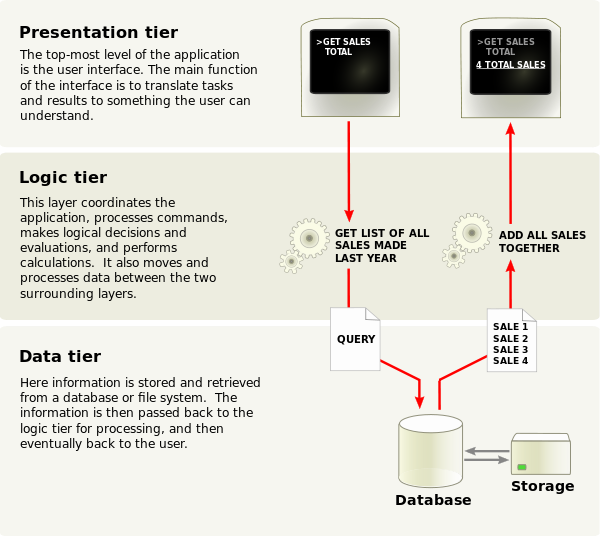
Architecture Patterns & Styles

# Architectural Patterns

### Three-Tier



**Presentation tier**

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. (In simple terms it is a layer which users can access directly such as a web page, or an operating systems GUI)

**Application tier (**[**business logic**](http://en.wikipedia.org/wiki/Business_logic)**, logic tier, data access tier, or middle tier)**

The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application’s functionality by performing detailed processing.

**Data tier**

This tier consists of database servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

In an MVC design, the view sends updates to the controller, the controller updates the model, and the model updates the view. In three-tier architecture, the client tier never communicates directly with the data tier; all communication must pass through the middle tier.

Most of architecture can claims that it is M-tirs, or M-layers, talking about M-tirs, usually we have hardware concerns involved.

N-tier just refers to the physical structure of an implementation. These two are sometimes confused because an MVC design is often implemented using an N-tier architecture.

### Model-view-controller

* A **controller** can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).
* A **model** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. In some cases an MVC implementation might instead be "passive," so that other components must [poll](http://en.wikipedia.org/wiki/Polling_(computer_science)) the model for updates rather than being notified.
* A **view** requests information from the model that it needs for generating an output representation to the user

# Architecture Styles

## Structure

### Component-based

Software engineers[[*who?*](http://en.wikipedia.org/wiki/Wikipedia:Avoid_weasel_words)] regard components as part of the starting platform for [service-orientation](http://en.wikipedia.org/wiki/Service-orientation). Components play this role, for example, in [web services](http://en.wikipedia.org/wiki/Web_service), and more recently, in [service-oriented architectures](http://en.wikipedia.org/wiki/Service-oriented_architecture) (SOA), whereby a component is converted by the web service into a *service* and subsequently inherits further characteristics beyond that of an ordinary component.

Components can produce or consume events and can be used for [event-driven architectures](http://en.wikipedia.org/wiki/Event-driven_architecture) (EDA).

(OOP) maintain that software should be written according to a mental model of the actual or imagined objects it represents.Component-based software engineering, by contrast, makes no such assumptions, and instead states that developers should construct software by gluing together prefabricated components many experts feel adaptable to evolving needs is more important than reuse, since 80% of software engineering deals with maintaining or releasing new versions.

### Monolithic application

### Layered

### Pipes & filters