Knowledgebase (KB)

NeoSOFT® TECHNOLOGIES

All Categories

- HR Policies (40)
- Employee Performance Management System (0)
- Training (16)
- Utilities (2)
- Knowledge Sharing (15)
  - Mobile (0)
  - Others / Misc (6)
    - "Minute Of Meeting" - Guidelines
    - Difference Between Layers & Tiers
    - Email Writing - Guidelines
    - GPS
    - Prepared Statements / Parameterized Statements
    - Stand Up Meeting
  - Web (3)
  - Database Guidelines
  - Hardware - Software Performance Guidelines
  - Hosting Guidelines
  - HTML-CSS-JS Guidelines
  - Neosoft Secure Coding Practices
  - Security Guidelines

There are no categories to display in knowledge base.

Recently Viewed

- "Minute Of Meeting" - Guidelines
- Web Application - Architecture
- SQL Injection (SQLI)
- Different Types of Hacking Techniques
- Security Guidelines

Home » Group Categories » Others / Misc

Difference Between Layers & Tiers

Article Number: 52 | Rating: 5/5 from 1 votes | Last Updated: Wed, Nov 13, 2013 at 1:26 AM

What's the difference between “Layers” and “Tiers”?

Layers are merely a way of organizing your code. Layers are a means of logical separation, and are an architectural pattern to separate concerns. Typical layers are the storage / data layer, data access layer, business logic layer and presentation layer.

When we’re talking about layers, we’re only talking about logical organization of code. In no way, it is implied that these layers might run on different computers OR in different processes on a single computer OR even in a single process on a single computer. All we are doing is discussing a way of organizing a code into a set of layers defined by specific function.

Tiers however, are only about where the code runs. Specifically, tiers are places where layers are deployed and where layers run. In other words, tiers are the physical deployment of layers.

What we achieve by dividing application into various layers?

- Abstraction
- Isolation
- Manageability
- Performance
- Reusability
- Testability

What we achieve by dividing application into various tiers?

- **Maintainability:** Because each tier is independent of the other tiers, updates or changes can be carried out without affecting the application as a whole.
- **Scalability:** Because tiers are based on the deployment of layers, scaling out an application is reasonably straightforward.
- **Flexibility:** Because each tier can be managed or scaled independently, flexibility is increased.
- **Availability:** Applications can exploit the modular architecture of enabling systems using easily scalable components, which increases availability.

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client–server architecture in which presentation, application processing, and data management functions are logically separated.

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 is the three-tier architecture.

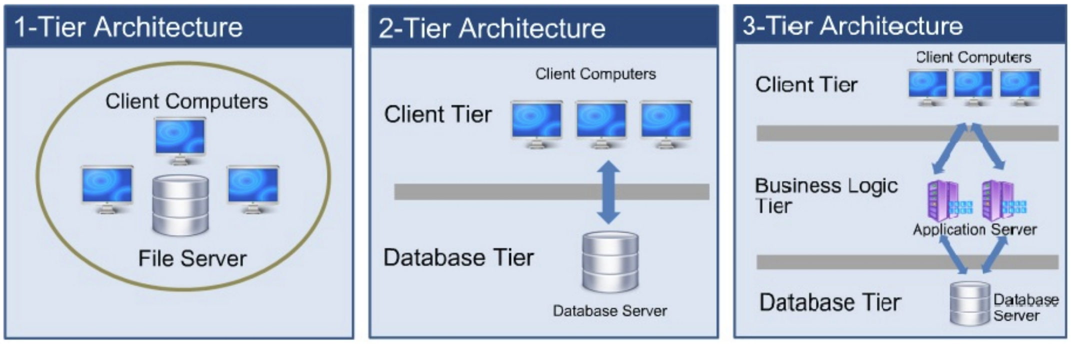
A Three-tier architecture is typically composed of a presentation tier, a business or data access tier, and a data tier.

While the concepts of layer and tier are often used interchangeably, one fairly common point of view is that there is indeed a difference. This view holds that a layer is a logical structuring mechanism for the elements that make up the software solution, while a tier is a physical structuring mechanism for the system infrastructure.

Presentation tier: Which users accesses directly such as a web pages(through browser) or windows forms(through Operating System GUI).

Application tier (business or data access tier): Which controls an application's functionality by performing detailed processing.

Data tier: Which consists of database servers. Here information is stored and retrieved.



Comparison with the MVC architecture

At first glance, the three tiers may seem similar to the model-view-controller (MVC) concept; however, topologically they are different. A fundamental rule in a three tier architecture is the client tier never communicates directly with the data tier; in a three-tier model all communication must pass through the middle tier. Conceptually the three-tier architecture is linear. However, the MVC architecture is triangular: the view sends updates to the controller, the controller updates the model, and the view gets updated directly from the model.

From a historical perspective the three-tier architecture concept emerged in the 1990s from observations of distributed systems (e.g., web applications) where the client, middle ware and data tiers ran on physically separate platforms. Whereas MVC comes from the previous decade (by work at Xerox PARC in the late 1970s and early 1980s) and is based on observations of applications that ran on a single graphical workstation; MVC was applied to distributed applications later in its history.

Posted by: Bhavesh Patel - Tue, Nov 12, 2013 at 11:59 PM This article has been viewed 312 times.
Filed Under: Others / Misc

Article Rating (1 Votes)

★★★★★

Rate this article

Bookmark Article (CTRL-D)

Select One ▾ Rate

Attachments

There are no attachments for this article.

Related Articles

- Prepared Statements / Parameterized Statements
  Viewed 231 times since Tue, Jul 29, 2014
- Stand Up Meeting
  Viewed 638 times since Tue, Feb 11, 2014
- "Minute Of Meeting" - Guidelines
  Viewed 557 times since Fri, Dec 19, 2014
- Email Writing - Guidelines
  Viewed 565 times since Mon, Sep 16, 2013
- GPS
  Viewed 269 times since Thu, Aug 29, 2013