ASP.net

N-Tier Architecture. ASP.NET Example

Darius

Posted by Darius on August 12th, 2013.

Introduction

N-tier architecture is probably one of the most used architecture models in the industry. It is used so often because it's scalable, extensible, secure and maintainable over time. It also helps the developers with different roles to better do their job without interfering with each other.

In this article I will present a basic n-tier architecture that can be used for creating small up to medium asp.net applications either MVC or WebForms.

But before we start exemplifying the concept, I want first to mention some key aspects independent of this architecture that need to be considered so that the design of your application will remain viable over time:

- Make sure that your application layers are not dependent on each other. They need to remain independent so that new layers can be added or old technologies to be changed.
- Single Responsibility Principle. Make sure that every component or module is responsible only for a specific feature or functionality.
- Principle of Least Knowledge. A component or object should not know about internal details of other components.
- Don't Repeat Yourself(DRY). A specific functionality should exist only in one place. It should not be duplicated in any other component.
- Set up some coding standards for development. This assures code consistency and maintainability.
- Keep shared functionality such as exception logging or security in a place that be accessed from any application levels (i.e. presentation layer or business layer)

Architecture Schema

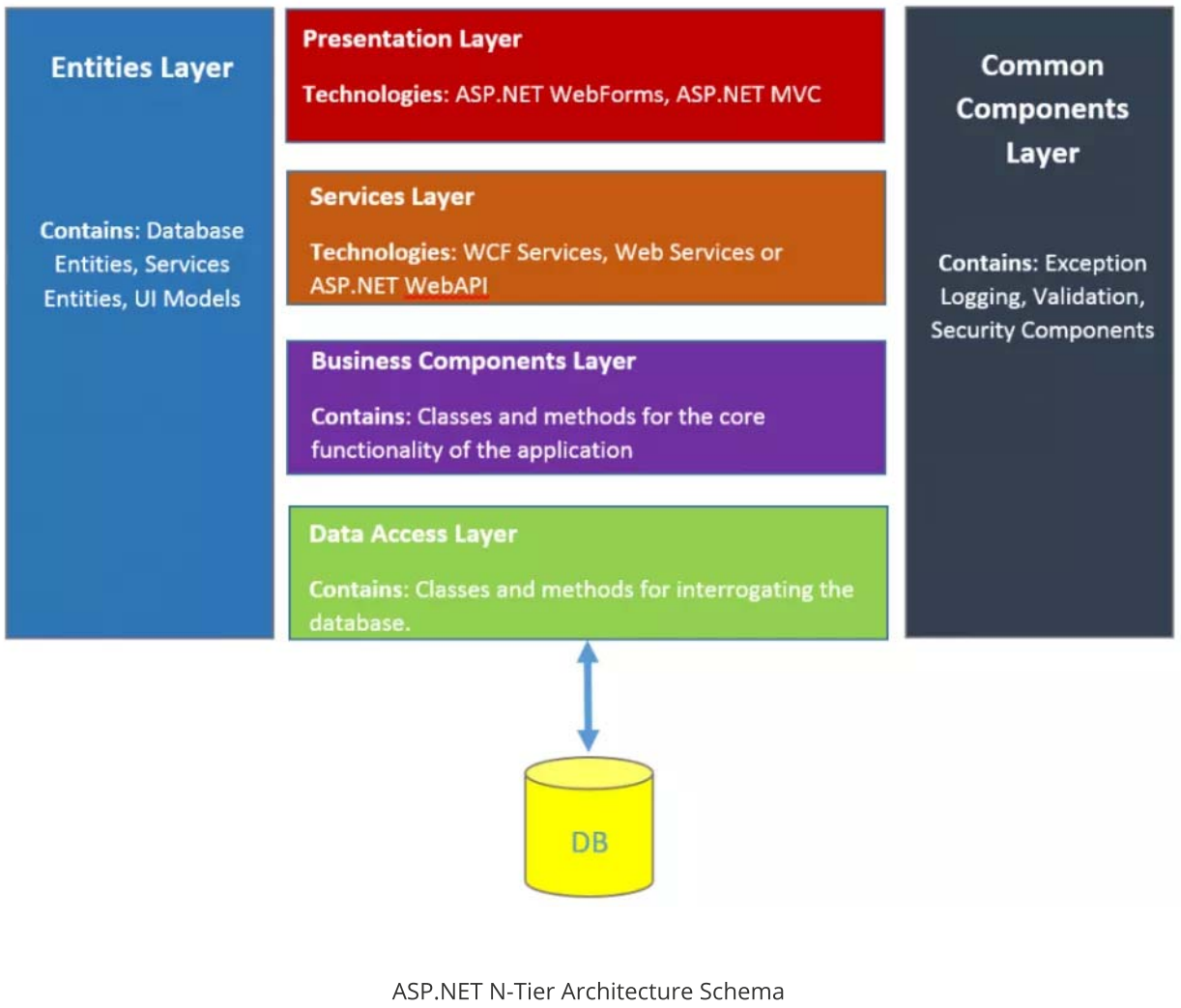
Below you have an image with how the architecture schema looks like:

Darius Dumitrescu is a Experience Senior CMS Consultant with in depth .NET knowledge, focused

LATEST POSTS

How to Use Adaptive Personalization with Sitecore
08th July, 2017

How to personalize based on search engine keywords with Sitecore
06th June, 2017



ASP.NET N-Tier Architecture Schema

A Short Description for each Layer

Entities Layer

The Entities layer contains all the entities that are used in all the other projects of the application such as tables mappings from database, DTOs (Data Transfer Objects) or ASP.NET MVC Models. Most of the classes in this layer are POCO (Plain Old CLR Objects).

Data Access Layer

The Data Access layer contains functionality for Creating, Returning, Updating, Deleting (CRUD) items into the database. In this layer you can use technologies like ADO.NET, nHibernate or Entity Framework.

The entities or the database table mappings classes needed for the above operations will be referenced from Entities layer described above.

This layer should only be shared by the Business Components layer. In Visual Studio, the layer will be represented by a “Class Library” project type.

Business Components Layer

Darius

Darius Dumitrescu is a creative Senior CMS Consultant with in depth .NET knowledge, focused

LATEST POSTS

How to Use Adaptive Personalization with Sitecore
08th July, 2017

How to personalize based on search engine keywords with Sitecore
10th June, 2017

he Business Components layer contains all the core functionality of the application. The purpose is to hold all the custom logic that is applied on the methods that are exposed from the Data Access Layer before they are referenced in the Presentation Layer or the Services Layer. Operations like converting arrays to lists, mathematical calculations or variable conversions are made in this layer.

The entities needed for the above operations will be referenced from Entities layer and the da operation methods will be referenced from Data Access layer.

This layer should be shared only by the Presentation Layer or the Services Layer. In Visual Studio the layer will be represented by a “Class Library” project type.

Services Layer

The Services Layer gives you the possibility to expose the Business Components Layer as an API to third party systems.

For example, if you want to share with a mobile application some data about the products that you have in your database, this layer should be responsible for that. However, this layer is not a mandatory one. If you consider that your application will not share data with another system, then this layer should not be created.

As I said earlier, the methods that are needed to be exposed in this layer are referenced from the Business Components Layer and this layer should only be shared by the Presentation Layer and other third party systems.

In this layer you can use technologies like WCF Services, Web Services or ASP.NET WebAPI.

Presentation Layer

The Presentation Layer is responsible of hosting the user interface of your application. As an example, in this layer you will have all the .aspx pages and .ascx user controls along with JavaScript and CSS files. The methods responsible for getting data from the database will be referenced only directly from Business Components layer or from Services layer.

In this layer you can use technologies like ASP.NET WebForms or ASP.NET MVC.

NOTE: Be sure that this layer is kept clean of database direct calls or direct access to Data Access layer.

Common Components Layer

The Common Components Layer contains all the common libraries or functionality that can be used in any of the above layers. For example: validation functionality, security libraries, encryption tools or exception logging classes can be added to this layer.

Darius

Darius Dumitrescu is a creative Senior CMS Consultant with in depth .NET knowledge, focused

LATEST POSTS

How to Use Adaptive Personalization with Sitecore

08th July, 2017

How to personalize based on search engine keywords with Sitecore

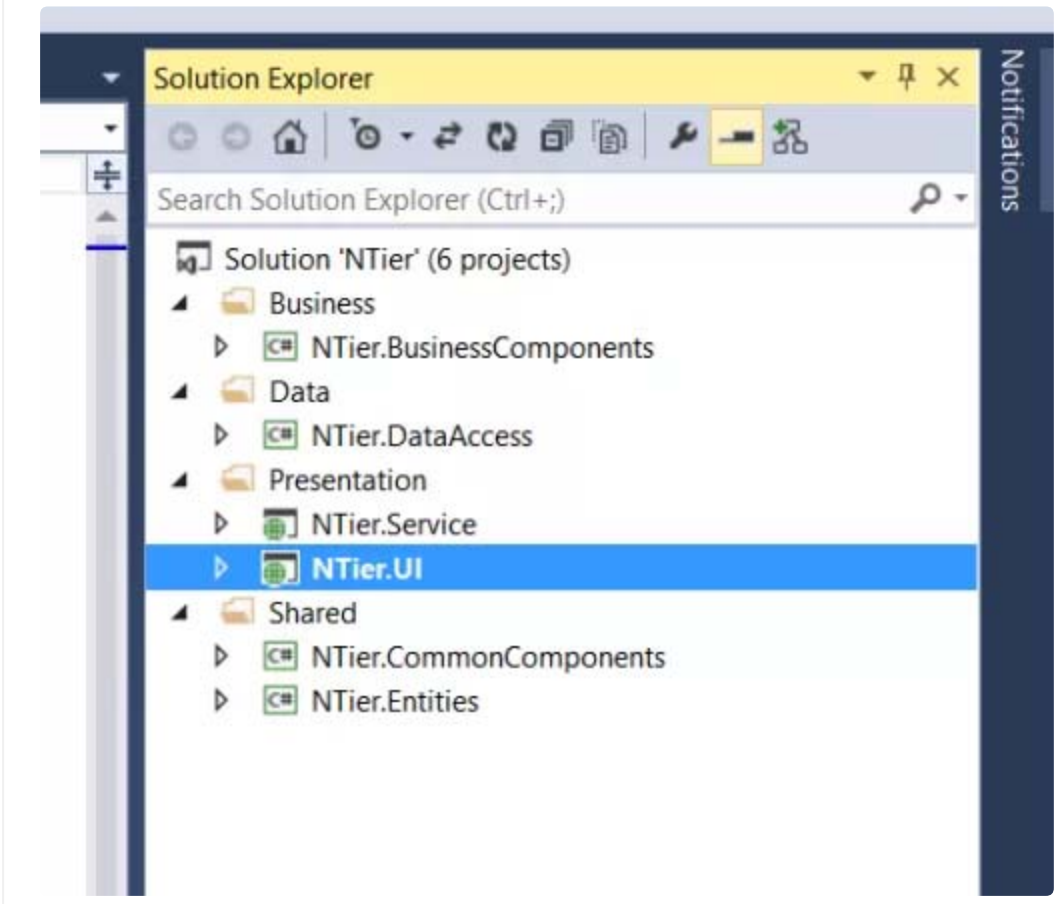
10th June, 2017

his is an optional layer and it is mandatory not to have any dependencies on the above described layers because this layer can be compiled and shared across multiple projects.

In Visual Studio, the layer will be represented by a “Class Library” project type.

Practical Example

In Visual Studio the described n-tier structure will look like this:



Visual Studio N-Tier Example

You can download a Visual Studio solution containing the layer described above from below

Attachments

- NTierArchitecture

  File size: 6 MB

Share this:

Related

Resources to learn ASP.NET WebForms, ADO.NET, HTML, CSS and SQL
September 9, 2013
In "ASP.net"

ASP.NET Core - How to Get and Write values to HTML Inputs outside the scope of a Model
May 8, 2017
In "ASP.net"

Batch Script for Cleaning ASP.NET solution
May 7, 2012
In "Tutorials"

Darius

Darius Dumitrescu is a creative Senior CMS Consultant with in depth .NET knowledge, focused

LATEST POSTS

How to Use Adaptive Personalization with Sitecore
08th July, 2017

How to personalize based on search engine keywords with Sitecore
10th June, 2017

Darius

Darius Dumitrescu is a creative Senior CMS Consultant with in depth .NET knowledge, f· ·sed on Web Development and Architecture Design.

View Comments (17) ...

Darius

Darius Dumitrescu is a creative Senior CMS Consultant with in depth .NET knowledge, focused

LATEST POSTS

How to Use Adaptive Personalization with Sitecore
08th July, 2017

How to personalize based on search engine keywords with Sitecore
10th June, 2017