Entity Vs Model Vs ViewModel Vs DataModel

Published by Gaurav Gahlot on July 6, 2017

Last updated on April 1, 2019

Different people define **Entity**, **Model**, **ViewModel** and **DataModel** in different ways. However, these terms may sometimes differ from their actual meaning, based upon the context. In this post I would like to share my understanding of these terms.

Entity

An **entity** is the tabular representation of your domain class/object in the database and has an identity. In fact, an entity represents a single instance of your domain object saved into the database as a record. It has some attributes that we represent as columns in our tables.

For instance, in order to represent a Customer as an entity, it must have some attributes like CustomerId, CustomerName, Address and many more depending on the context. Consequently, these attributes form the columns of our Customer table. And therefore, each customer data that you save in the table, represents a **customer entity**.

Model

People often confuse **entity** with **model**. However, these two are quite different. A model typically represents a real world object that is related to the problem or domain space.

In programming, we create classes to represent them. These classes, known as models, have some properties and methods (defining objects behavior) in a particular domain space. For instance, in any customer oriented problem, we may have a customer class that has some properties and methods.

In some ORM (Object Relational Mapper) frameworks, a model is tightly bound to an entity. This means that every scalar property maps to an entity attribute. However, if you are familiar with Entity Framework, you must know that, not all properties in your domain class map to an entity column. And that's one way in which an entity is different from a model.

ViewModel

The term **ViewModel** originates from the MVVM design pattern. A view has the responsibility of rendering data typically coming from an object. However, there are instances in which the data for the view comes from two different objects.

In such scenarios, we create a model class which consists of all properties required by the view. Different domain model instances then initialize this object. It's not a

domain model but a **ViewModel** because, a specific view uses it. Also, it doesn't represent a real world object.

DataModel

The models in a particular domain space represent the real world objects. In order to solve a problem, these objects interact with each other. Some objects share a relationship among them and consequently, form a **data model** that represents the objects and the relationship between them. Click here to know more about object relationships.

In an application managing customer orders, for instance, if we have a customer and order object then these objects share a many to many relationship between them. The **data model** is eventually dependent on the way our objects interact with each other. In a database, we see the data model as a network of tables referring to some other tables.

We can find different definitions of these terms at different places. However, through this post I just shared my opinion about the same. It would be great to receive any feedback on this topic. Please do share your opinion through comments.

Published in CS Fundamentals

Previous Post
Understanding Relationship Between Objects

Next Post
Anonymous Types in C#