Generic Repository Pattern with ASP.NET MVC and EF

Md. Mahedee Hasan

Microsoft MVP | Trainer | Speaker

Software Architect

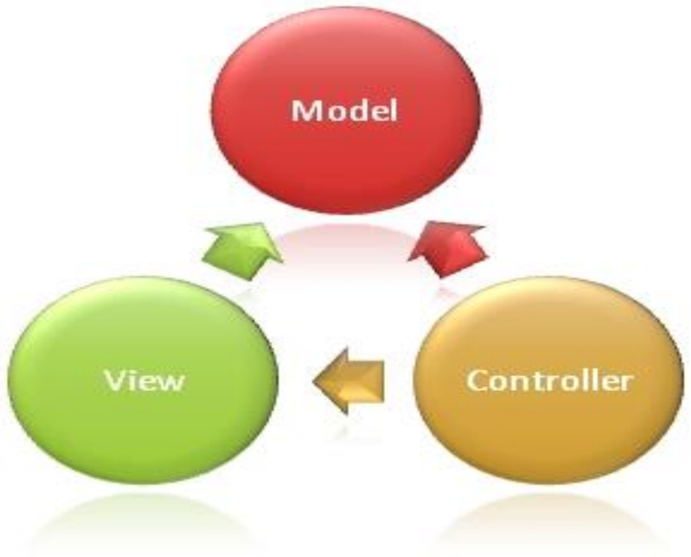
LeadSoft Bangladesh Limited

Linkedin: http://www.linkedin.com/in/mahedee

Blog: http://mahedee.net/

What is MVC?

- MVC Stands for **Model – View – Controller**
- It is **Software Architectural pattern**

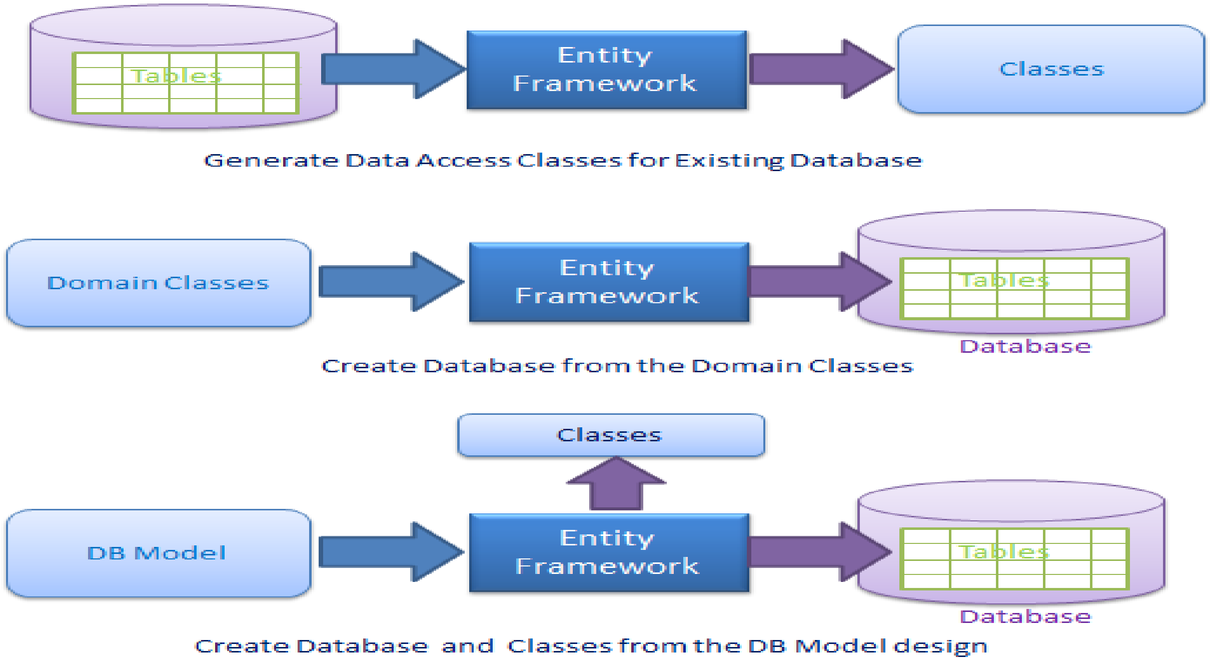


What is ASP.NET MVC?

- Is an **open source** web application **framework**
- It **implements** the model-view-controller **pattern**
- MVC design pattern aims to “**Separation Of concern**”

What is ASP.NET EF?

- Entity Framework is an **ORM**.
- Work with **relational data** using **domain-specific objects**
- **Eliminates** the need for most of the **data-access code**


Tables
Entity Framework
Classes
Generate Data Access Classes for Existing Database
Domain Classes
Entity Framework
Tables
Database
Create Database from the Domain Classes
Classes
DB Model
Entity Framework
Tables
Database
Create Database and Classes from the DB Model design

DRY?

Don’t repeat yourself

-Means don't write duplicate code

Use generic repository pattern to implement DRY

Software Design Pattern

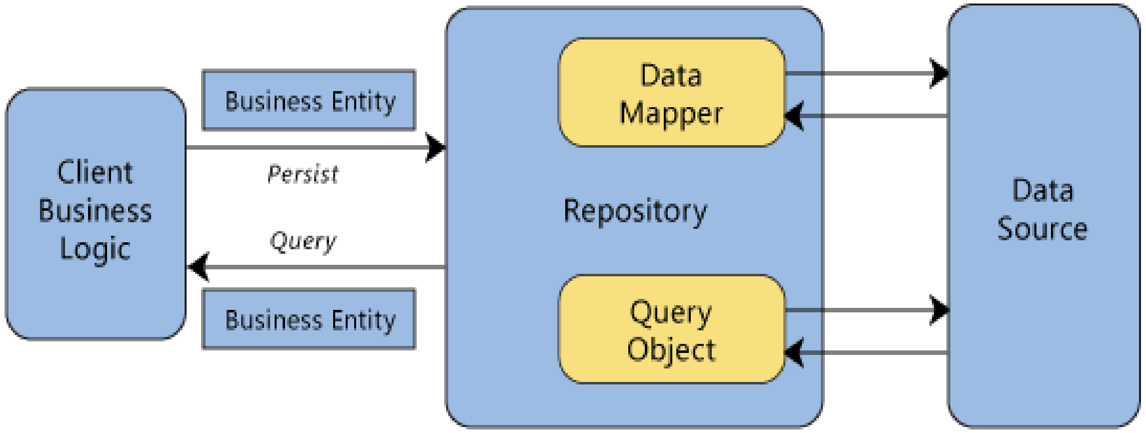
- What is **Design Pattern**?
  - Design pattern is a **solution** of **known problems**

- What is **Software Design Pattern**?
  - Is a **solution** of **known problems** in **Software**.

- **Strategies of solving** *commonly occurring problems*.
- A design pattern is not a **finish design.**
- It is like a **template** to **solve a problem**.

Repository Pattern

- **Mediator** between BLL and DAL(Data Source)
- It's a **separation layer** between **Data** and **Domain Layer**
  - **Separates** data and domain Layer



Benefits of Repository Pattern

- **Centralizes** data logic or service **logic**.
- Provides a substitution point for the **unit tests**
  - Both BLL and DAL
- Provides a **flexible architecture**
  - Can adopt **new change** easily
- **Domain driven** development is easier

What is Generic Repository Pattern?

- Generally **- one repository** for **one model** to **access data**.
  - Write **similar** code **again** and again

- What if Single Repository for all?

- **Single repository** for **data access** of all models.

Benefits of Generic Repository Pattern

- **Reduce redundancy** of code
- **Faster** development
- **Force** developer to work **same pattern**
  - Possibility of **less error or no error**
- Easy to maintain
  - **Centralize** data access logic

Implementation

- **Tools and Technology used**
  - Visual Studio 2013
  - Visual C#
  - ASP.NET MVC 5
  - Entity Framework 6
  - Razor view engine

Implementation…

Step 1: Create an ASP.NET **MVC 5 application** using Visual Studio 2013

Implementation …

Step 2: Configure **connection string** in **web.config**

Implementation …

Step 3: Create **Models**

Implementation …

Step 4: Create a **DbContext** in **Repository folder**.

Implementation …

Step 5: Create **IGenericRepository** and **GenericRepository** in Repository folder

Implementation …

- Step 6:
  - **Create controllers** of each models
  - Select template "**MVC5 Controller with views, using Entity Framework**"
  - Modify Controllers and **use Generic Repository in Controller**

Implementation …

- **Step 7**: Add **links to _Layout**

Implementation …

- Step 8: Write following command in package manager console
  - **PM> Enable-Migrations -ContextTypeName GenericRepoContext**
  - **PM> Add-Migration initialcreate**
  - **PM> Update-Database -Verbose -Force**

Implementation …

Step 9: **Run** Project

Thank you