CookBook (Recipe.Core)

This post will describe how to start using **cookbook** instantly.

# REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Date** | **Editor** | **Description** |
| 1 | 01/05/2017 | Ovais Bawany | Initiate the Document with all the details about **CookBook** and its usage |
|  |  |  |  |

# Cookbook About

Cookbook contains Generic base classes for controller, service, repository, Entity and DTO.

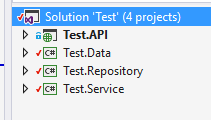
Key point is to inherit those classes respectively in our project and reuse their built in functionalities which helps in quick development of API and easy to maintain code structure and architecture.

# Cookbook Prerequisites

Cookbook has been designed mainly focusing Web API solutions built using **Code First Approach** provided by **.Net framework**.

Recommended code structure for using Cookbook should contain

* Web.API project (Web.API template)
* Web.Data project (Class Library)
* Web.Repository (Class Library)
* Web.Service (Class Library)



# Install Nuget Package (Recipe.Core)

Install cookbook from package manager console by using command **install-package Recipe.Core** on all the project in the code structure.

# CookBook Details and Steps for Using It

## Create Request info

Create a class in API project with the name **RequestInfo** and inherit the interface **IRequestInfo**. Since this class is defined in the API project, so it can use the http request object. This class should contain common properties and methods which are required in the service or repository to get object data from http request and token provided by the client.

## Create Unit of Work

**IUnitOfWork** and **UnitOfWork** is defined and implemented in the cookbook which contains the common properties required like **DBContext** and common methods like **SaveChanges**.

It is required to create an interface **IWebUnitofWork** and inherit **IUnitOfWork** and create a class **WebUnitofWork** and inherit **UnitofWork** and **IWebUnitofWork** in the **Web.Repository** project.

Unit of Work should contains the objects of all the repositories, so it can be injected in any service and that service can access all the repositories via **UnitOfWork** object.

**Example:**

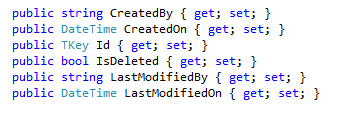




## Create Entity

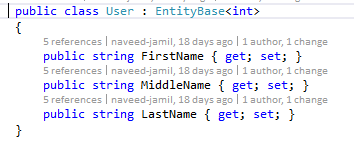
There is a base class named **EntityBase** which can be inherited in the entity classes in the project.

This base class takes generic parameter **Key Type** for that entity and inherit some properties in the derived entity which are



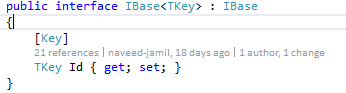
**Id** will be of the type provided at the time of inheritance.

**Example:**

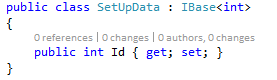


Normally entities required auditing and the deletion is soft delete.

In case when there is no need for auditing like in setup data, derived class can inherit interface named “IBase” and implement the Id property of the interface.



**Example:**



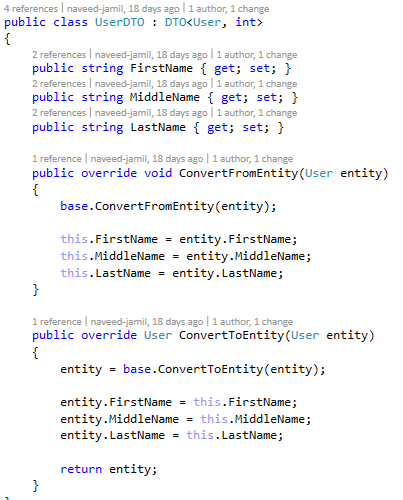
## Create DTO

There is a base class named **DTO** which can be inherited in the **DTO** classes in the project.

This base class takes generic parameters **Entity Type** for which DTO is created and **Key Type** for that DTO and inherit some common methods of conversion in the derived DTO.

All the conversion methods are virtual so the derived class can override them.

**Example:**



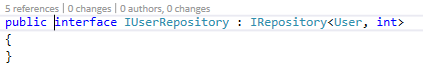
## Create Repository

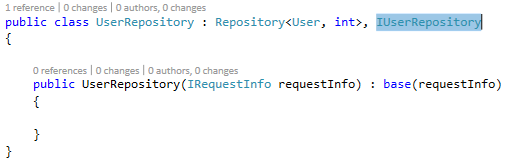
There is a base interface **IRepository** and a base class **Repository** which contains methods of **CRUD** operations and getters.

Create an interface **IWebRepository** which inherits **IRepository** interface and create a class **WebRepository** which inherits **Repository** class and created **IWebRepository** interface in the “Web.Repository” project.

**Repository** class and **IRepository** interface takes generic parameters **Entity Type** for which the repository is being created and **Key Type** of that entity.

**Example:**





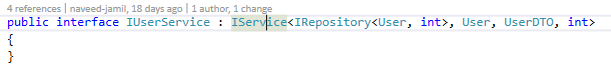
## Create Service

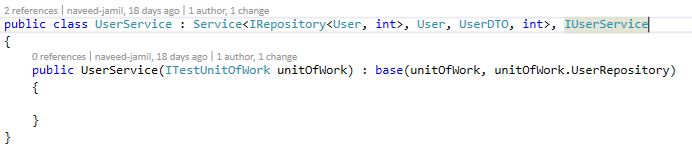
There is a base interface **IService** and a base class **Service** which contains methods of **CRUD** operations and getters.

Create an interface **IWebService** which inherits **IService** interface and create a class **WebService** which inherits **Service** class and created interface **IWebService** in the “Web. Service” project.

**Service** class and **IService** interface takes generic parameters **Repository Type**, **Entity Type**, **DTO Type** and **Key Type**.

**Example:**





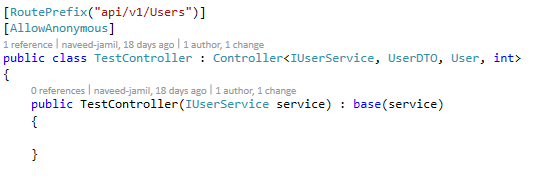
## Create Controller

There is an abstract class **Controller** which itself inherited from **ApiController** and contains all the common methods of **CRUD** operations and getters.

Create a new controller which inherits **Controller** class in the “Web. API” project.

**Controller** class takes generic parameters **Service Type**, **DTO Type**, **Entity Type** and **Key Type**.

**Example:**



# Conclusion

Using CookBook (Recipe.Core) package, all it is required to create only classes while inheriting respective ones and the exposed controller is ready with the five common API end points (Create, Update, Delete, Get and GetAll).