Introduction to Dapper and Repository Pattern Framework in MVC

1. Foreword

The framework is based on the underlying framework of the dapper.net repository and follows the SOLID design principles. At the same time, it supports multiple database instances of different types of databases (mssql, mysql, oracle ...) and multiple databases coexisting with the same type of database. Simply modify the configuration file to move it at will. Using the currently popular tools (dependency injection: autofac, cache: redis, etc.), the underlying package encapsulates the basic CURD operations.

First add a new database called DapperRepositoryDb and execute the sql script you need (mssql or mysql).

1. Project stratification

reference from nopcommerce：[https://www.nopcommerce.com](https://www.nopcommerce.com/)

Relevant descriptions and current status of each layer：

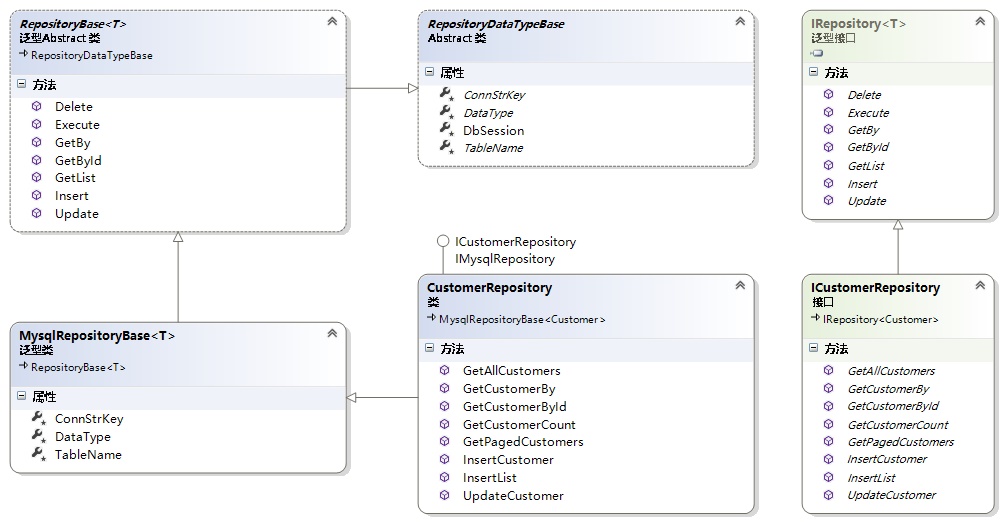
DapperRepository.Core: Placing related data interfaces and entity objects, caches, and public classes

DapperRepository.Data: Data manipulation layer, implementing specific Repository and database connection and access

DapperRepository.Services: Business logic layer, processing related business logic

DapperRepository.Web: client operation

1. Adapt to the design idea of multi-instance database multi-instance



Generic interface Irepository<T>: the most basic CURD

Instance interface ICustomerRepository: inherits ICustomerRepository and extensions, generic interface (for different database types)

The underlying abstract class RepositoryDataTypeBase: This defines the abstract property key of the database connection string (because my database connection string is configured as a json file, post-serialized into a dictionary), the abstract property DataType (which specifies which type of database), and the abstract property TableName ( The name of the data table under the current database), the attribute DbSession (database session attribute, the instance implements the transaction operation)

Generic abstract class RepositoryBase<T>: method for implementing generic interface Irepository<T>

The instance generic class MySqlRepositoryBase<T>: defines an instance generic class of a specific database type, rewrites the abstract property defined by the underlying abstract class, thereby obtaining the specific connection string key, database type and table name.

Concrete instance class CustomerRepository: implement generic interface Irepository<T> and instance interface IcustomerRepository

1. Cache design redis

The cache design is all referenced to NopCommerce, so this part is not my personal design, here is probably talk about, redis is a distributed cache, so nop uses RedLock.net as a distributed lock, and is subsequently registered as a redis singleton. The serialization format is json, and the rest is to cache daily operations.

1. Dependency injection Autofac

Dependency injection uses autofac, which is mainly used for decoupling between modules and project maintainability.



Here I am not using a single injection one by one, but through the agreement to find the unified injection after reflection, I also wrote an extension registration to support the unified registration of other examples that are still needed.

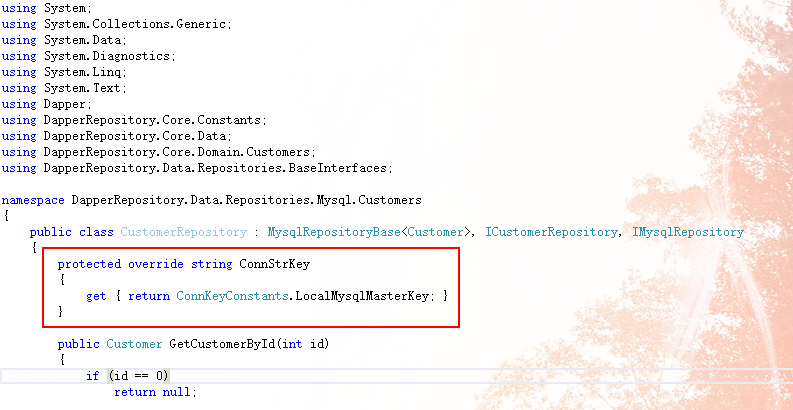
1. Demonstrate basic configuration and operation
2. Switch database instance and modify table name



This static class defines the database type key and the database connection string key. The default implementation is in the instance generic class MySqlRepositoryBase<T>, such as:



Currently defined database type is mysql, database connection string and data table name, note that the DataType property modifier is sealed, so the subclass can not be override, because the convention is to use mysql, if you need to switch the current data instance, you need to re The class overrides ConnStrKey, and the TableName is the same, such as:



This makes it possible to switch database instances (specifically based on business needs). Of course, this change is only valid for the current instance (so the framework can implement the current single database type and multiple database instances)

1. Switch database type



Just modify it to switch the database type of the current project.

This should be consistent with the defined key (convention)



1. Modify the database connection string

The database connection string is defined in the ~App\_Data/DbConnSettings.json file and can be modified by yourself.

Any questions are welcome to contact, contact information:

Author:HuangZhongQiu

QQ Email: 877558989

Google Email:huangzhongqiu25@gmail.com