Dapper and Repository Pattern in MVC框架介绍

1. 前言

该框架是基于dapper.net的repository底层框架，并遵循SOLID设计原则，同时支持多个不同类型数据库（mssql、mysql、oracle...）的数据库多实例并存，同类型数据库不同数据库多实例并存。只需修改配置文件即可任意迁移。使用当下一些流行的工具（依赖注入：autofac，缓存：redis等），底层封装了基本的CURD操作。

先建立数据库DapperRepositoryDb，然后执行项目下的sql文件

1. 项目分层：

分层参考Nopcommerce：[https://www.nopcommerce.com](https://www.nopcommerce.com/)

各层相关说明及目前状态：

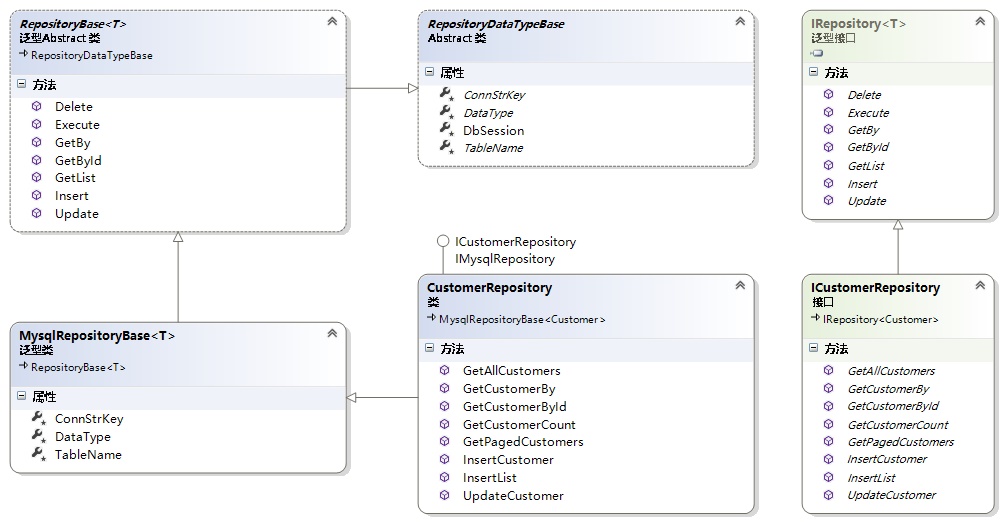
DapperRepository.Core: 放置相关数据接口和实体对象、缓存、公用类

DapperRepository.Data:数据操作层，实现具体Repository和数据库连接及访问

DapperRepository.Services:业务逻辑层，处理相关业务逻辑

DapperRepository.Web:web端，客户端操作

1. 适应多类型数据库多实例的设计思想



泛型接口Irepository<T>：最基本的CURD

实例接口ICustomerRepository：继承ICustomerRepository及扩展，通用接口（对于不同数据库类型）

基础抽象类RepositoryDataTypeBase：这里定义数据库连接串的抽象属性key（因为我数据库连接串是配置成json文件，后序列化为字典），抽象属性DataType（指定具体哪一种类型数据库），抽象属性TableName（当前数据库下的数据表名称）,属性DbSession（数据库会话属性，该实例实现事务操作）

泛型抽象类RepositoryBase<T>：实现泛型接口Irepository<T>定义的方法

实例泛型类MySqlRepositoryBase<T>：定义具体那种数据库类型的实例泛型类，重写基础抽象类定义的抽象属性，从而获取具体连接串key，数据库类型及表名

具体实例类CustomerRepository：实现泛型接口Irepository<T>及实例接口IcustomerRepository



缓存设计全部参考于NopCommerce，故这部分不属于我个人设计，这里大概谈一下，redis是分布式缓存，故nop用RedLock.net作为分布式锁，并在后续注册为redis单例。序列化格式为json，其它的就是缓存日常操作了。

1. 依赖注入Autofac

依赖注入采用autofac，依赖注入主要用于模块之间的解耦及项目可维护性。



这里我不是用单一逐条的注入，而是通过约定查找反射后的统一注入，我还写了一个扩展注册，用于支持统一注册其它还需要的实例。

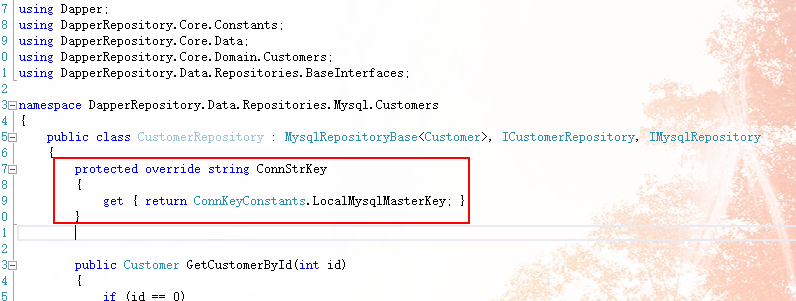
1. 演示基本配置及操作
2. 切换数据库实例及修改表名



这个静态类定义数据库类型key及数据库连接串key，默认实现在实例泛型类MySqlRepositoryBase<T>，如：



当前定义数据库类型为mysql，数据库连接串及数据表名，注意DataType属性修饰符为sealed，所以子类无法再重写，因为这里约定就是用的mysql，如果需要切换当前数据实例，则需要再子类重写ConnStrKey，TableName也一样，如：



这样就实现了切换数据库实例（具体使用根据业务需要）。当然这样改只针对当前这个实例有效（故该框架可实现当前单一数据库类型而多个数据库实例）

1. 切换数据库类型



只需要修改这个即可切换当前项目的数据库类型，这个要跟



定义的key一致（约定）

1. 修改数据库连接串

数据库连接串定义在~App\_Data/DbConnSettings.json文件，自行修改即可。

有什么疑问欢迎咨询，联系方式：

Author：HuangZhongQiu

QQ：875755898