Sparrow.CommonLibrary使用说明

## Cache

封装后的通用缓存接口，本身并没有缓存功能，它提供类似Helper的功能。可以对具体缓存的组件解偶。

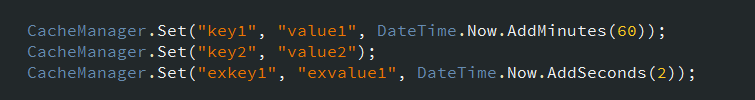
### 主要的类

CacheManager

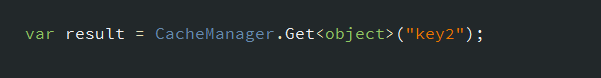
ICache

默认**CacheManager**使用本地缓存，如果系统需要使用第三方，如：Memcahe、Redis这类缓存系统，则需要实现**ICache**接口。

### 添加缓存

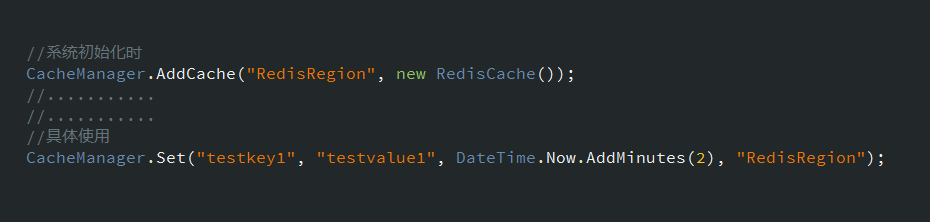


### 获取缓存



### 使用第三方缓存

使用自定义第三方的缓存系统，RedisCache是一个自己实现的ICache接口。系统初始化可以通过代码方式，也可以通过配置文件配置。



## Common

一些不便归类至各个功能模块的通用类，当然其它的几个功能模块也使用到了时边的一些类。

### 主要的类

AppSettings

Buffered

DbValueCast

PathBuilder

Timestamp

### AppSettins

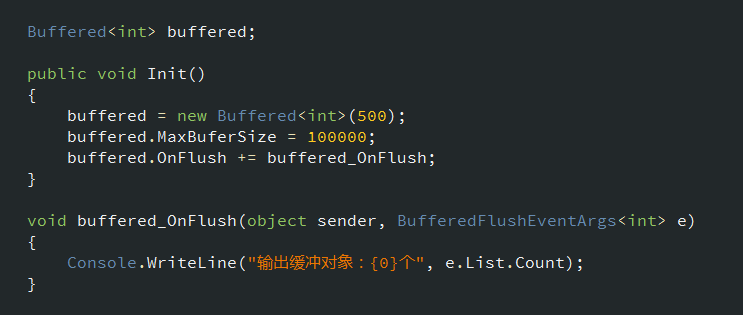
app.config/web.config配置节点：appSettings，当配置文件中不包含需要的配置时，使用默认设置的配置。





### Buffered

多线程安全的缓冲区，通过定时或定量触发操作，以节省CPU和网络资源，降低主业务线程时间。主要适用于可延迟的操作（如：日志）。

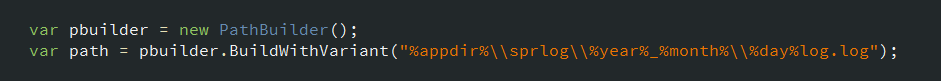


向缓存区中写入一个对象：



### PathBuilder

文件路径生成辅助工具，生成一个动态路径（如：应用于日志文件）



字符串中的变量以“%”开始和结束，默认支持的变量包括：

**%appdir%** 应用程序启动的目录

**%windir%** windir|systemroot|temp返回的是同一个值（主要是与windows操作系统的习惯保持一致）。

**%temp%** 返回当前系统的临时文件夹的路径。

**%rndname%** 随机文件夹或文件名

**%date%** 返回当前日期，格式：yyMMdd

**%year%** 返回当前年份，格式：yyyy

**%month%** 返回当前月份，格式：MM

**%day%** 返回当前日期的天，格式：dd

**%hour%** 返回当前时间的小时，格式：hh

**%minute%** 返回当前时间的分钟，格式：mm

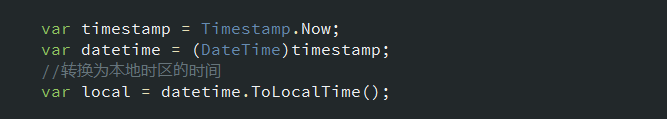
### Timestamp

基于格林威志的Unix时间截（1970-1-1 0:0:0），可以将一个数字或一个日期对象转换为Timestamp。

将日期转换为Unix时间截：



将时间截转换为日期（格林威志时间，需要手动转换成为本地时间）：



## Cryptography

数据加密（对称和非对称）以及数据签名封装类。算法都是.net框架所支持的加密签名算法（注：所有数据加密或签名的默认编码为utf8）。

### 主要的类

Crypto

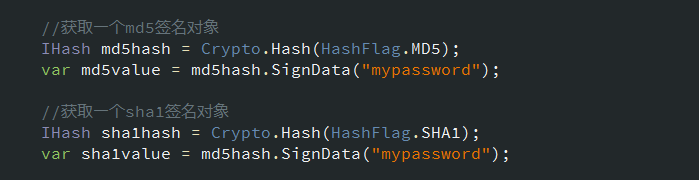
AsymmetricFlag

HashFlag

SymmetricFlag

### 数据签名

主要的名称算法有：MD5、RIPEMD160、SHA1、SHA256、SHA384、SHA512



### 数据加密

主要的加密算法有：

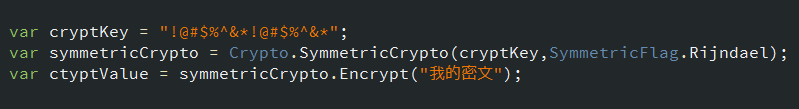
Rijndael（推荐使用）：AES（高级加密标准）的一个实现，速度快，安全级别高。

TripleDES：基于DES，对一块数据用三个不同的密钥进行三次加密，强度更高。

RC2：RC2用变长密钥对大量数据进行加密，比DES快。

DES（不推荐使用）：数据加密标准，速度较快，安全级别低，易被破解。

对称加密，不同加密算法对key的长度要求不一样，具体请参考MSDN文件说明



### 数据加密（非对称加密）

（暂无）

## Database

对ADO.NET的二次封装，隐藏连接对象的创建、打开、关闭这些繁琐步骤。针对config文件中的connectionStrings/add[**providerName**]自动选择数据库驱动。

### 主要的类

DatabaseHelper

ParameterCollection

SqlBatch

### 执行一个查询

以下方法只适合返回小数据量的集合操作，如果返回的数据结果集非常大，建议使用DataSet处理。



### 执行一组查询，同时返回多个结果集合



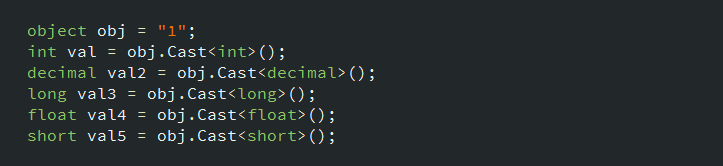
## Entity

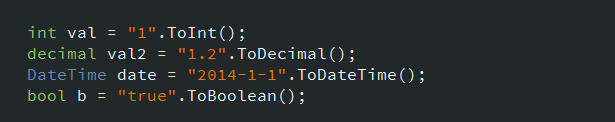
（暂无，高级编的时候再作讲解，以内部使用为主）

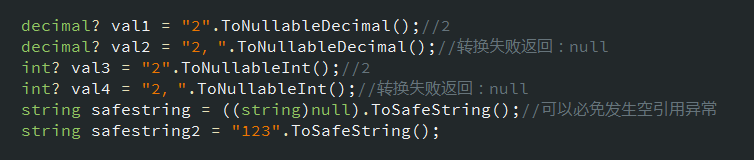
## Extenssions

一系列扩展方法，主要包括数据类型转换、DataTable操作、数据格式验证等。

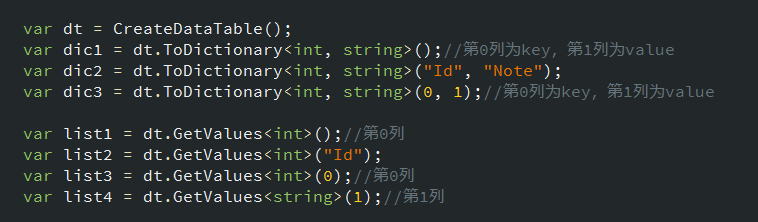
### 数据类型转换







### DataTable操作



### 数据格式验证



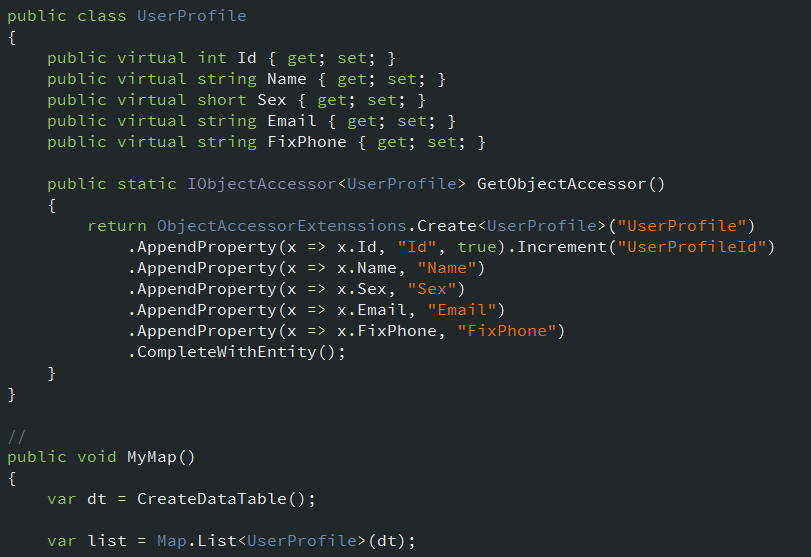
## Mapper

将DataTable/IDataReader结合Entity模块，将结果集转换为实体对象。

### 主要的类

Map

### 数据实体映射



### 创建一个适用数据修改的实体

该操作通常需要Repository模块配合着一起使用



## Net

Http操作相关功能模块。

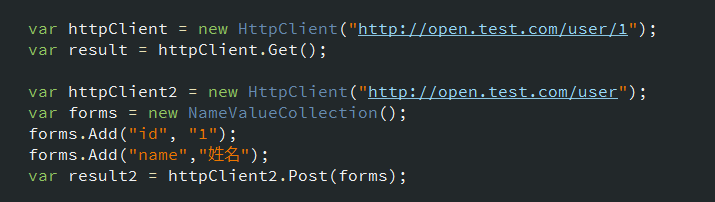
### 主要的类

HttpClient

RecognizeSpider

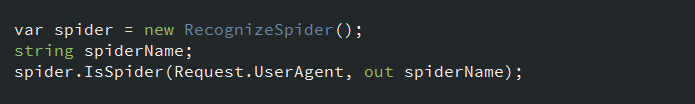
### HttpClient

应用于网络请求，.net4.5框架中已经有可替代的HttpClient，如果使用.net4.5框架时，可以忽略它。



### RecognizeSpider

网络爬虫识别工具类，识别客户端发出的请求（UserAgent）是否有爬虫标识。



## Query

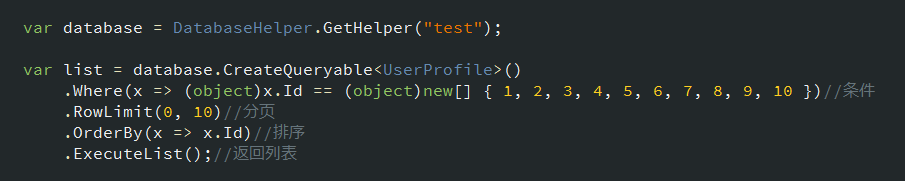
主要为Respository提供单表查询的能力

### 主要的类

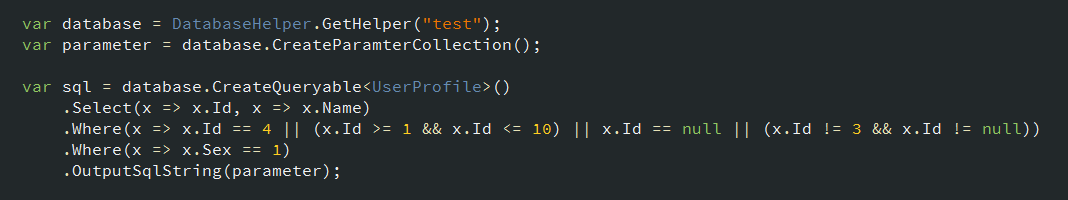
Queryable<T>

Queryable采用的是Fluent风格的代码结构，只有调用ExecuteList、ExecuteReader、OutputSqlString才会执行最终的表达式。

### 查询返回一个结果集列表



### 生成sql语句



## Repository

提供数据插入、修改、查询操作，目标为达到单表操作完全基于对象，不需要写任何Sql语句就可以完成，降低Sql书写错误减少不必要的调式工作以提高开发效率。

### 主要的类

DatabaseRepository<T>

### 插入一个实体对象



### 修改一个实体对象

修改一行数据时，Map.Create<UserProfile>()与new UserProfile()对象完全会达到不一样的效果。

以下场景是所有表操作经常出现的一种，Map.Create<UserProfile>()创建的对象,Repository会知道对象哪些属性被赋值过，只生成有过赋值的属性。

例：UPDATE UserProfile SET Email=@email WHERE Id=@id

如果使用new UserProfile()

例：UPDATE UserProfile SET Name=@name,Email=@email,FixPhone= @fixphone ....... WHERE Id=@id

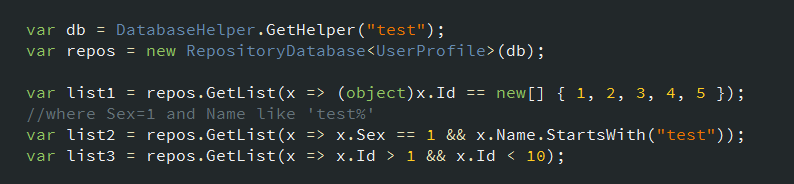


### 插入或修改

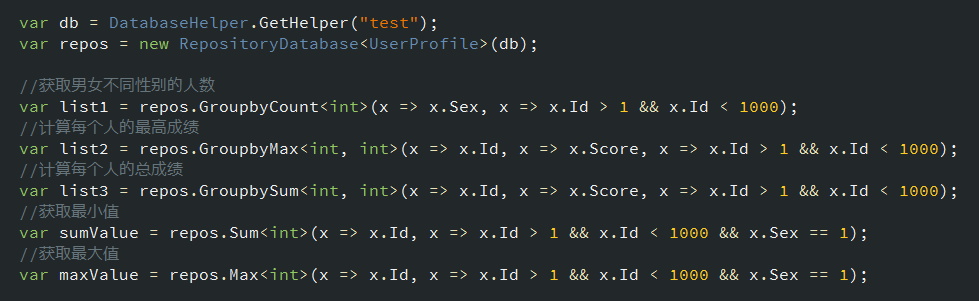
如果数据已经存在则执行update操作，否则执行insert操作。



### 查询数据



### 聚合查询



## Retrying

（暂无）