### 针对数据库通用底层的SQL配置

### 增加可扩展函数的类库说明

2020年5月22日

##### 前置、后置 函数的说明 文档中函数名均为示例函数，不在生产环境使用

前置函数为：处理还未执行的SQL语句中的参数或者字段、比如在查询条件时，验证用户名是否为空，则可以使用前置条件，对字段及其参数进行验证 比如

select \* from userinfo where VALNULL[userid={0}] AND password=ENPWD[{1}]

此SQL包含两个前置函数 VALNULL：验证用户名是否为空 ENPWD 加密密码

后置函数为：处理已经执行过SQL语句后的结果集。

GETLIST[select\* from userinfo where dept ={0}]

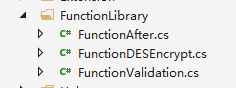
此SQL语句外围有一个后置函数 GETLIST，可以表述为把查询到的结果转为List列表，或者进行其他的一些你想要的操作，比如把查询的结果集中的某个字段的值0/1 转为是或否

## 自定义函数-特性 CustomFunAttribute.cs

1. 这个特性是针对方法的描述，包括方法名、方法类型、方法说明、调用方式（示例）、如果有其他需求，可在此进行扩展
2. 针对此特性，提供根据特性获取所有函数的函数列表 即：CSIHelper.Helper.GetFunctionList() 即可获取函数列表

## 自定义函数方法 FunctionLibrary函数库

1.在FunctionLibrary文件夹中，添加所需要的函数，这个文件夹中只存放扩展的函数



2.此文件夹所有的类为分布类，区别仅仅是类名不同，内部共用一个统一类名FunctionExe

3.添加一个函数



其中 参数 condition 为用花括号包围的数字{0}、或者是经过处理过一次的参数 ’123456’，代码中有很详细的说明可以参考

## SqlFilter.cs 主程序入口文件

在通用底层的控制器中使用，或者其他需要处理SQL语句的类或者方法中使用

1.首先获取函数列表-包括 前置、后置

2.把前台传入的参数通过$符号进行分割转为List<string>列表

3.把sql语句、参数列表、函数列表传入 SqlFilter.cs类中的GetFilterSql即可过滤掉包含前置条件的Sql语句；示例如下

Task<string> ret = SqlFilter.GetFilterSql(sql3, para3, funAllList.Where(x=>x.FunType=="前置").ToList());

4.如果需要处理后置函数的话，需要先把前置条件进行过滤完毕之后，再对剩余的后置函数进行处理

string afterSql = SqlFilter.GetAfterSql(ret.Result, funAllList).Result;

①afterSql为过滤掉后置函数的sql语句，执行此SQL语句，获取原始数据集dstmp

List<Function> functions = SqlFilter.GetOrderFunction(ret.Result, funAllList);

② functions为sql语句包含的后置函数列表

var obj = SqlFilter.GetAfterFun(dstmp, functions);

③把数据集、后置函数列表传入GetAfterFun函数进行函数处理，即可得到最终结果

## 使用说明-测试

在通用底层系统的前台页面中，对需要配置的SQL增加所需要的函数，前提是必须是后台已经扩展完成的函数，

使用示例均为 函数名[] 即 函数名+用中括号[]进行包围的函数体

VALNULL[userid={0}] password=ENPWD[{1}]