using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Huiting.DBAccess.Attributes

{

[AttributeUsage(AttributeTargets.Property)]

public class BusinessKeyAttribute : Attribute

{

public string GroupName { get; set; } = "BLL";//业务逻辑层Key

public BusinessKeyAttribute() { }

public BusinessKeyAttribute(string GroupName)

{

this.GroupName = GroupName;

}

}

}

using Huiting.Common;

using System;

namespace Huiting.DBAccess.Attributes

{

/// <summary>

/// 字段特性

/// </summary>

[AttributeUsage(AttributeTargets.Property)]

public class DataFieldAttribute : Attribute

{

/// <summary>

/// 创建表字段类型特性

/// </summary>

/// <param name="typeAndSize">类型</param>

/// <param name="isNull">是否可空 true（默认）：可空 false：不可空</param>

/// <param name="isPrimaryKey">是否主键 true：主键 false（默认）：非主键</param>

/// <param name="isIdentity">是否自增列 true：自增 false（默认）：不自增</param>

/// <param name="isUnique"></param>

/// <param name="isIndex"></param>

/// <param name="defaultValue"></param>

/// <param name="descption"></param>

public DataFieldAttribute(string typeAndSize, bool isNull = true, bool isPrimaryKey = false, bool isIdentity = false, bool isUnique = false, bool isIndex = false, string defaultValue = null, string descption = null)

{

//ColumnName = columnName;

TypeAndSize = typeAndSize;

IsNull = isNull;

IsPrimaryKey = isPrimaryKey;

IsIdentity = isIdentity;

IsUnique = isUnique;

IsIndex = isIndex;

DefaultValue = defaultValue;

this.Descption = descption;

}

/// <summary>

/// 列名称

/// </summary>

//public string ColumnName { get; }

/// <summary>

/// 列类型和大小

/// </summary>

public string TypeAndSize { get; }

/// <summary>

/// 是否为空

/// </summary>

public bool IsNull { get; }

/// <summary>

/// 是否为主键

/// </summary>

public bool IsPrimaryKey { get; }

/// <summary>

/// 是否为自增列

/// </summary>

public bool IsIdentity { get; }

/// <summary>

/// 是否唯一

/// </summary>

public bool IsUnique { get; }

/// <summary>

/// 是否创建索引

/// </summary>

public bool IsIndex { get; }

/// <summary>

/// 默认值

/// </summary>

public string DefaultValue { get; }

/// <summary>

/// 注释

/// </summary>

public string Descption { get; }

/// <summary>

/// 检查字段值是否有效

/// </summary>

/// <param name="strValue"></param>

/// <param name="type"></param>

/// <returns></returns>

public bool CheckDataValueValid(string strValue, Type type)

{

if (IsPrimaryKey || IsNull == false)

{

if (string.IsNullOrWhiteSpace(strValue.ToString()))

return false;

}

if (type == typeof(int))

{

return int.TryParse(strValue, out \_);

}

else if (type == typeof(float))

{

return float.TryParse(strValue, out \_);

}

else if (type == typeof(double))

{

return double.TryParse(strValue, out \_);

}

else if (type == typeof(DateTime))

{

return DateTime.TryParse(strValue, out \_);

}

else if (type == typeof(string))

{

return true;

}

else //if (type == typeof(int))

{

Log.Warn($"未知数据类型：{type.Name}", System.Reflection.MethodBase.GetCurrentMethod());

return true;

}

}

}

}

using System;

namespace Huiting.DBAccess.Attributes

{

/// <summary>

/// 表特性

/// </summary>

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Struct)]

public class DataTableAttribute : Attribute

{

public DataTableAttribute(string tableName, bool isClearData = false)

{

TableName = tableName;

IsClearData = isClearData;

}

/// <summary>

/// 表名称

/// </summary>

public string TableName { get; }

/// <summary>

/// 是否清除数据，清除数据为true；不清除为false默认

/// </summary>

public bool IsClearData { get; }

}

}

using System;

namespace Huiting.DBAccess.Attributes

{

[AttributeUsage(AttributeTargets.Property)]

public class DisplayFieldAttribute:Attribute

{

/// <summary>

/// 显示值

/// </summary>

public string DisplayText { get; set; }

/// <summary>

/// 单位

/// </summary>

public string UnitText { get; set; }

public DisplayFieldAttribute()

{

}

public DisplayFieldAttribute(string DisplayText, string UnitText = null)

{

this.DisplayText = DisplayText;

this.UnitText = UnitText;

}

}

}

using Huiting.DBAccess.Entity;

using System;

namespace Huiting.DBAccess.Entity.Dict

{

/// <summary>

/// Sqlite\_Master表对应Dto

/// </summary>

public class SqliteMasterDto: IDto

{

public string Type { get; set; }

public string Name { get; set; }

public string Tbl\_Name { get; set; }

public string RootPage { get; set; }

public string Sql { get; set; }

public Type TableType { get; set; }

}

}

using Huiting.DBAccess.Entity;

namespace Huiting.DBAccess.Entity.Dict

{

public class TableInfoDto : IDto

{

public long Cid { get; set; }

public string Name { get; set; }

public string Type { get; set; }

public int NotNull { get; set; }

public string Dflt\_Value { get; set; }

public int Pk { get; set; }

}

}

using Huiting.DBAccess.Attributes;

using Huiting.DBAccess.Entity;

using Newtonsoft.Json;

using System;

namespace Huiting.DBAccess.Entity.Dtos

{

[DataTable("projectInfo")]

public class ProjectInfoDto:IDto

{

private long id;

[JsonProperty("id")]

[DataField("integer", false, true, true)]

public long Id

{

get

{

return id;

}

set

{

id = value;

}

}

private String mc;

[JsonProperty("mc")]

[DataField("varchar(100)",false)]

public String MC

{

get

{

return mc;

}

set

{

mc = value;

}

}

private DateTime createtime;

[JsonProperty("createtime")]

[DataField("date",false)]

public DateTime CreateTime

{

get

{

return createtime;

}

set

{

createtime = value;

}

}

private DateTime updatetime;

[JsonProperty("updatetime")]

[DataField("date")]

public DateTime UpdateTime

{

get

{

return updatetime;

}

set

{

updatetime = value;

}

}

private String grouplist;

[JsonProperty("grouplist")]

[DataField("varchar(100)")]

public String GroupList

{

get

{

return grouplist;

}

set

{

grouplist = value;

}

}

private String sortlist;

[JsonProperty("sortlist")]

[DataField("varchar(100)")]

public String SortList

{

get

{

return sortlist;

}

set

{

sortlist = value;

}

}

private String remark;

[JsonProperty("remark")]

[DataField("varchar(250)")]

public String Remark

{

get

{

return remark;

}

set

{

remark = value;

}

}

}

}

using Huiting.DBAccess.Attributes;

using Huiting.DBAccess.Entity;

using Newtonsoft.Json;

using System;

namespace Huiting.DBAccess.Entity.Dtos

{

[DataTable("UnitBasicData")]

public class UnitBasicDataDto : IDto

{

private long id;

[JsonProperty("id")]

[DataField("integer", false, true, true)]

public long Id

{

get

{

return id;

}

set

{

id = value;

}

}

private String dYDM;

[JsonProperty("dydm")]

[BusinessKey]

[DisplayField("单元代码")]

[DataField("varchar(255)", false, false)]

public String DYDM

{

get

{

return dYDM;

}

set

{

dYDM = value;

}

}

private long proID;

[BusinessKey]

[JsonProperty("proid")]

[DataField("integer", false, false)]

public long ProID

{

get

{

return proID;

}

set

{

proID = value;

}

}

private String fgsmc;

[JsonProperty("fgsmc")]

[DisplayField("分公司名称")]

[DataField("varchar(255)", true)]

public String FGSMC

{

get

{

return fgsmc;

}

set

{

fgsmc = value;

}

}

private String cYCMC;

[JsonProperty("cycmc")]

[DisplayField("采油厂名称")]

[DataField("varchar(255)", true)]

public String CYCMC

{

get

{

return cYCMC;

}

set

{

cYCMC = value;

}

}

private String yTMC;

[JsonProperty("ytmc")]

[DisplayField("油田名称")]

[DataField("varchar(255)", false)]

public String YTMC

{

get

{

return yTMC;

}

set

{

yTMC = value;

}

}

private String dYMC;

[JsonProperty("dymc")]

[DisplayField("单元名称")]

[DataField("varchar(255)", false)]

public String DYMC

{

get

{

return dYMC;

}

set

{

dYMC = value;

}

}

private Double hYMJ;

[JsonProperty("hymj")]

[DisplayField("含油面积")]

[DataField("float")]

public Double HYMJ

{

get

{

return hYMJ;

}

set

{

hYMJ = value;

}

}

private Double hQMJ;

[JsonProperty("hqmj")]

[DisplayField("含气面积")]

[DataField("float")]

public Double HQMJ

{

get

{

return hQMJ;

}

set

{

hQMJ = value;

}

}

private Double yYDZCL;

[JsonProperty("yydzcl")]

[DisplayField("原油地质储量", "万吨")]

[DataField("float")]

public Double YYDZCL

{

get

{

return yYDZCL;

}

set

{

yYDZCL = value;

}

}

private String yYKCCL;

[JsonProperty("yykccl")]

[DisplayField("原油可采储量", "万吨")]

[DataField("varchar(255)")]

public String YYKCCL

{

get

{

return yYKCCL;

}

set

{

yYKCCL = value;

}

}

private String yQCLX;

[JsonProperty("yqclx")]

[DisplayField("油气藏类型")]

[DataField("varchar(255)")]

public String YQCLX

{

get

{

return yQCLX;

}

set

{

yQCLX = value;

}

}

private String qDLX;

[JsonProperty("qDLX")]

[DisplayField("驱动类型")]

[DataField("varchar(255)")]

public String QDLX

{

get

{

return qDLX;

}

set

{

qDLX = value;

}

}

private String qBLX;

[JsonProperty("qBLX")]

[DisplayField("圈闭类型")]

[DataField("varchar(255)")]

public String QBLX

{

get

{

return qBLX;

}

set

{

qBLX = value;

}

}

private String kFFS;

[JsonProperty("kffs")]

[DisplayField("开发方式")]

[DataField("varchar(255)")]

public String KFFS

{

get

{

return kFFS;

}

set

{

kFFS = value;

}

}

private String cLLB;

[JsonProperty("cllb")]

[DisplayField("储量类别")]

[DataField("varchar(255)")]

public String CLLB

{

get

{

return cLLB;

}

set

{

cLLB = value;

}

}

private Int64 wZJS;

[JsonProperty("wzjs")]

[DisplayField("完钻井数")]

[DataField("integer")]

public Int64 WZJS

{

get

{

return wZJS;

}

set

{

wZJS = value;

}

}

private String bZ;

[JsonProperty("bZ")]

[DisplayField("备注")]

[DataField("char(255)")]

public String BZ

{

get

{

return bZ;

}

set

{

bZ = value;

}

}

private Double yymd;

[JsonProperty("yymd")]

[DisplayField("原油密度")]

[DataField("float")]

public Double Yymd

{

get

{

return yymd;

}

set

{

yymd = value;

}

}

private String kfzt;

[JsonProperty("kfzt")]

[DisplayField("开发状态")]

[DataField("char(255)")]

public String Kfzt

{

get

{

return kfzt;

}

set

{

kfzt = value;

}

}

private String dyType;

[JsonProperty("dytype")]

[DisplayField("单元类型")]

[DataField("char(255)")]

public String DYType

{

get

{

return dyType;

}

set

{

dyType = value;

}

}

private Int64 involvedEconEval;

[JsonProperty("involvedEconEval")]

[DisplayField("参与经济评价")]

[DataField("integer")]

public Int64 InvolvedEconEval

{

get

{

return involvedEconEval;

}

set

{

involvedEconEval = value;

}

}

private String mainProduct;

[JsonProperty("mainProduct")]

[DisplayField("主产品项")]

[DataField("varchar(100)")]

public String MainProduct

{

get

{

return mainProduct;

}

set

{

mainProduct = value;

}

}

private Double qdzcl;

[JsonProperty("qDZCL")]

[DisplayField("气地质储量", "万方")]

[DataField("float")]

public Double QDZCL

{

get

{

return qdzcl;

}

set

{

qdzcl = value;

}

}

private String qKCCL;

[JsonProperty("yykccl")]

[DisplayField("气可采储量", "万方")]

[DataField("varchar(255)")]

public String QKCCL

{

get

{

return qKCCL;

}

set

{

qKCCL = value;

}

}

private Int64 ishcdy;

[JsonProperty("iSHCDY")]

[DisplayField("是否合成单元")]

[DataField("integer")]

public Int64 IsHCDY

{

get

{

return ishcdy;

}

set

{

ishcdy = value;

}

}

private Int64 hclx;

[JsonProperty("hCLX")]

[DisplayField("合成类型")]

[DataField("integer")]

public Int64 HCLX

{

get

{

return hclx;

}

set

{

hclx = value;

}

}

private String hccondition;

[JsonProperty("hccondition")]

[DisplayField("合成条件")]

[DataField("text")]

public String HCCondition

{

get

{

return hccondition;

}

set

{

hccondition = value;

}

}

}

}

using Huiting.DBAccess.Attributes;

using Huiting.DBAccess.Entity;

using Newtonsoft.Json;

using System;

namespace Huiting.DBAccess.Entity.Dtos

{

[DataTable("UnitDevelopData")]

public class UnitDevelopDataDto : IDto

{

private long id;

[JsonProperty("id")]

[DataField("integer", false, true, true)]

public long Id

{

get

{

return id;

}

set

{

id = value;

}

}

private String ny;

[JsonProperty("ny")]

[DisplayField("年月", "yyyyMM")]

[DataField("char(50)", false, false)]

public String NY

{

get

{

return ny;

}

set

{

ny = value;

}

}

private String dydm;

[BusinessKey]

[JsonProperty("dydm")]

[DisplayField("单元代码")]

[DataField("char(50)",false,false)]

public String DYDM

{

get

{

return dydm;

}

set

{

dydm = value;

}

}

private long proID;

[BusinessKey]

[JsonProperty("proid")]

[DataField("integer", false, false)]

public long ProID

{

get

{

return proID;

}

set

{

proID = value;

}

}

private Double ycy;

[JsonProperty("ycy")]

[DisplayField("月产油","万吨")]

[DataField("float",true,false)]

public Double YCY

{

get

{

return ycy;

}

set

{

ycy = value;

}

}

private Double ycq;

[JsonProperty("ycq")]

[DisplayField("月产气", "万方")]

[DataField("float",true,false)]

public Double YCQ

{

get

{

return ycq;

}

set

{

ycq = value;

}

}

private Double ycs;

[JsonProperty("yCS")]

[DisplayField("月产水", "万吨")]

[DataField("float",true,false)]

public Double YCS

{

get

{

return ycs;

}

set

{

ycs = value;

}

}

private Double yzs;

[JsonProperty("yZS")]

[DisplayField("月注水", "万吨")]

[DataField("float",true,false)]

public Double YZS

{

get

{

return yzs;

}

set

{

yzs = value;

}

}

private Double lcy;

[JsonProperty("lcy")]

[DisplayField("累采油", "万吨")]

[DataField("float",true,false)]

public Double LCY

{

get

{

return lcy;

}

set

{

lcy = value;

}

}

private Double lcq;

[JsonProperty("lCQ")]

[DisplayField("累产气", "万方")]

[DataField("float",true,false)]

public Double LCQ

{

get

{

return lcq;

}

set

{

lcq = value;

}

}

private Double lcs;

[JsonProperty("lcs")]

[DisplayField("累产水", "万吨")]

[DataField("float",true,false)]

public Double LCS

{

get

{

return lcs;

}

set

{

lcs = value;

}

}

private Double lzs;

[JsonProperty("lzs")]

[DisplayField("累注水", "万吨")]

[DataField("float",true,false)]

public Double LZS

{

get

{

return lzs;

}

set

{

lzs = value;

}

}

private Double zyjs;

[JsonProperty("zyjs")]

[DisplayField("钻井数", "口")]

[DataField("float",true,false)]

public Double ZYJS

{

get

{

return zyjs;

}

set

{

zyjs = value;

}

}

private Double kyjs;

[JsonProperty("kyjs")]

[DisplayField("开井数", "口")]

[DataField("float",true,false)]

public Double KYJS

{

get

{

return kyjs;

}

set

{

kyjs = value;

}

}

private Double zsjs;

[JsonProperty("zsjs")]

[DisplayField("钻水井数", "口")]

[DataField("float", true,false)]

public Double ZSJS

{

get

{

return zsjs;

}

set

{

zsjs = value;

}

}

private Double ksjs;

[JsonProperty("kSJS")]

[DisplayField("开水井数", "口")]

[DataField("float", true,false)]

public Double KSJS

{

get

{

return ksjs;

}

set

{

ksjs = value;

}

}

}

}

using Huiting.DBAccess.Attributes;

using Huiting.DBAccess.Entity;

using Newtonsoft.Json;

using System;

namespace Huiting.DBAccess.Entity.Dtos

{

[DataTable("WellDevelopData")]

public class WellDevelopDataDto : IDto

{

private long id;

[JsonProperty("id")]

[DataField("integer", false, true,true)]

public long Id

{

get

{

return id;

}

set

{

id = value;

}

}

private String ny;

[BusinessKey]

[JsonProperty("ny")]

[DisplayField("年月", "yyyyMM")]

[DataField("varchar", true, false)]

public String NY

{

get

{

return ny;

}

set

{

ny = value;

}

}

private String jh;

[BusinessKey]

[JsonProperty("jh")]

[DisplayField("井号")]

[DataField("varchar", true, false)]

public String JH

{

get

{

return jh;

}

set

{

jh = value;

}

}

private String dydm;

[BusinessKey]

[JsonProperty("dydm")]

[DisplayField("单元代码")]

[DataField("char(50)", false, false)]

public String DYDM

{

get

{

return dydm;

}

set

{

dydm = value;

}

}

private long proID;

[BusinessKey]

[JsonProperty("proid")]

[DataField("integer", false, false)]

public long ProID

{

get

{

return proID;

}

set

{

proID = value;

}

}

private double yCYL;

[JsonProperty("yCYL")]

[DisplayField("月产油量", "吨")]

[DataField("int", true, false)]

public double YCYL

{

get

{

return yCYL;

}

set

{

yCYL = value;

}

}

private double yCQL;

[JsonProperty("yCQL")]

[DisplayField("月产气量", "万方")]

[DataField("varchar", true, false)]

public double YCQL

{

get

{

return yCQL;

}

set

{

yCQL = value;

}

}

private double yCSL;

[JsonProperty("yCSL")]

[DisplayField("月产水量", "吨")]

[DataField("int", true, false)]

public double YCSL

{

get

{

return yCSL;

}

set

{

yCSL = value;

}

}

private double lJCYL;

[JsonProperty("lJCYL")]

[DisplayField("累产油量", "万吨")]

[DataField("float", true, false)]

public double LJCYL

{

get

{

return lJCYL;

}

set

{

lJCYL = value;

}

}

private double lJCQL;

[JsonProperty("lJCQL")]

[DisplayField("累产气量", "万方")]

[DataField("float", true, false)]

public double LJCQL

{

get

{

return lJCQL;

}

set

{

lJCQL = value;

}

}

private double lJCSL;

[JsonProperty("lJCSL")]

[DisplayField("累产水量", "万吨")]

[DataField("float", true, false)]

public double LJCSL

{

get

{

return lJCSL;

}

set

{

lJCSL = value;

}

}

private double yZS;

[JsonProperty("yZS")]

[DisplayField("月注水", "万吨")]

[DataField("float", true, false)]

public double YZS

{

get

{

return yZS;

}

set

{

yZS = value;

}

}

private double lZS;

[JsonProperty("lZS")]

[DisplayField("累注水", "万吨")]

[DataField("float", true, false)]

public double LZS

{

get

{

return lZS;

}

set

{

lZS = value;

}

}

private String mQJB;

[JsonProperty("mQJB")]

[DisplayField("目前井别")]

[DataField("varchar", true, false)]

public String MQJB

{

get

{

return mQJB;

}

set

{

mQJB = value;

}

}

private Double scts;

[JsonProperty("scts")]

[DisplayField("生产天数", "天")]

[DataField("float", true, false)]

public Double SCTS

{

get

{

return scts;

}

set

{

scts = value;

}

}

private double dYM;

[JsonProperty("dym")]

[DisplayField("动液面")]

[DataField("varchar", true, false)]

public double DYM

{

get

{

return dYM;

}

set

{

dYM = value;

}

}

private String bZ;

[JsonProperty("bZ")]

[DataField("varchar", true, false)]

public String BZ

{

get

{

return bZ;

}

set

{

bZ = value;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Huiting.DBAccess.Entity

{

public interface IDto

{

}

}

using Huiting.Common;

using Huiting.DBAccess.Entity.Dict;

using Huiting.DBAccess.Helpers;

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Text;

namespace Huiting.DBAccess.Generator

{

/// <summary>

/// 模型生成器

/// </summary>

public static class DtoModelGenerator

{

private static string ModelFolder = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "../../DtoModels");

internal static void GenerateModels()

{

if (Directory.Exists(ModelFolder))

Directory.Delete(ModelFolder, true);

Directory.CreateDirectory(ModelFolder);

var OldCreateSqlList = new List<SqliteMasterDto>();

//查询所有建表语句

OldCreateSqlList = DapperHelper.SqlWithParams<SqliteMasterDto>("select \* from sqlite\_master where type ='table' and sql is not null;")?.ToList();

foreach (var sqliteMasterDto in OldCreateSqlList)

{

string sql = $"PRAGMA table\_info({sqliteMasterDto.Tbl\_Name})";

//查询原表结构

List<TableInfoDto> oldFieldList = DapperHelper.SqlWithParams<TableInfoDto>($"pragma table\_info({sqliteMasterDto.Tbl\_Name});")?.ToList();

//DataTable dtTmp = DBAccessHelper.GetDataTable(sql);

//dtTmp.TableName = tableName;

CreateCSFile(sqliteMasterDto.Name, oldFieldList);

//CreateCSFile(dtTmp);

}

Console.WriteLine("模型文件生成成功");

}

/// <summary>

/// 创建CS File

/// </summary>

/// <param name="dtStruct"></param>

internal static void CreateCSFile(string tableName, List<TableInfoDto> oldFieldList)

{

string tNameUpper = tableName.FirstCharToUpper();

string tNameLower = tableName.FirstCharToLower();

string fileFullName = Path.Combine(ModelFolder, $"{tNameUpper}Dto.cs");

if (File.Exists(fileFullName))

File.Delete(fileFullName);

string sql = $"select \* from {tNameUpper} where 1=2";

DataTable dtEmptyData = DapperHelper.GetDataTable(sql);

StringBuilder sb = new StringBuilder();

sb.AppendLine("using System;");

sb.AppendLine("using Newtonsoft.Json;");

sb.AppendLine("using Huiting.DBAccess.Attributes;");

sb.AppendLine();

sb.AppendLine("namespace Huiting.DBAccess.Entity.Dtos");

sb.AppendLine("{");

sb.AppendLine($"\t[DataTable(\"{tNameLower}\")] ");

sb.AppendLine($"\tpublic class {tNameUpper}Dto");

sb.AppendLine("\t{");

foreach (var item in oldFieldList)

{

string str = new string('\t', 20);

string dcNameUpper = item.Name.FirstCharToUpper(); //dr["name"].ToString().FirstCharToUpper();

string dcNameLower = item.Name.ToLower(); //dr["name"].ToString().FirstCharToLower();

Type dcCodeType = dtEmptyData.Columns[dcNameUpper].DataType;

string dcDBType = item.Type.ToString().ToLower();// dr["type"].ToString().ToLower();

bool isNull = item.NotNull == 0 ? true : false;

bool isPrimaryKey = item.Pk == 0 ? false : true;

sb.AppendLine($"\t\tprivate {dcCodeType.Name} {dcNameLower};");

sb.AppendLine($"\t\t[JsonProperty(\"{dcNameLower}\")]");

sb.AppendLine($"\t\t[DataField(\"{dcDBType}\",{isNull.ToString().ToLower()},{isPrimaryKey.ToString().ToLower()})]");

sb.AppendLine($"\t\tpublic {dcCodeType.Name} {dcNameUpper}");

sb.AppendLine("\t\t{");

sb.AppendLine("\t\t\tget");

sb.AppendLine("\t\t\t{");

sb.AppendLine($"\t\t\t\treturn {dcNameLower};");

sb.AppendLine("\t\t\t}");

sb.AppendLine("\t\t\tset");

sb.AppendLine("\t\t\t{");

sb.AppendLine($"\t\t\t\t{dcNameLower} = value;");

sb.AppendLine("\t\t\t}");

sb.AppendLine("\t\t}");

sb.AppendLine();

}

sb.AppendLine("\t}");

sb.AppendLine("}");

string content = sb.ToString();

System.IO.File.WriteAllText(fileFullName, content);

}

}

}

public static void CreateTable(params Type[] tables)

{

var OldCreateSqlList = new List<SqliteMasterDto>();

//List<SqliteMasterDto> NewCreateSqlList = new List<SqliteMasterDto>();

//查询所有建表语句

OldCreateSqlList = DapperHelper.SqlWithParams<SqliteMasterDto>("select \* from sqlite\_master where sql is not null;")?.ToList();

//所有建表Sql语句

var tableSB = new StringBuilder();

//var isFirstSync = false;

foreach (var table in tables)

{

var dataTableAttribute = ((DataTableAttribute[])table.GetCustomAttributes(typeof(DataTableAttribute), false))[0];

string tableName = dataTableAttribute.TableName;

string sqlStr = SqlGenerator.CreateTableByModel(table, tableName);

CacheService.Instance.AddTableNames(table);

var temp = OldCreateSqlList?.FirstOrDefault(old => old.Tbl\_Name == tableName && old.Type.ToLower() == "table");

//存在新旧表名一致

if (temp != null)

{

var newSqlList = sqlStr.Split(new[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

//新旧表的建表语句不一致，则表需要更新

if (temp.Sql.Replace(" ", "") != newSqlList[0].Replace(" ", "").Replace("IFNOTEXISTS", "")

|| (newSqlList.Count() > 1 && newSqlList[1].Replace(" ", "").Replace("IFNOTEXISTS", "") != OldCreateSqlList?.FirstOrDefault(old => old.Tbl\_Name == tableName && old.Type.ToLower() == "index")?.Sql.Replace(" ", "")))

{

//isFirstSync = true;

var li = new SqliteMasterDto { Name = tableName, Tbl\_Name = tableName, Sql = sqlStr, TableType = table };

UpdateDatabase(temp, li, dataTableAttribute.IsClearData);

}

}

else

{

tableSB.Append(sqlStr);

//isFirstSync = true;

}

}

//CacheInfo.IsFirstSync = isFirstSync;

if (!string.IsNullOrWhiteSpace(tableSB.ToString()))

DapperHelper.Execute(tableSB.ToString());

}