# https://www.cnblogs.com/igeekfan/p/freesql-generator.html

# [FreeSql.Generator命令行代码生成器是如何实现的](https://www.cnblogs.com/igeekfan/p/freesql-generator.html)

## **目录**

* FreeSql介绍
* FreeSql.Generator
* RazorEngine.NetCore
* 源码解析
* FreeSql.Tools

## **FreeSql**

FreeSql 是功能强大的对象关系映射技术(O/RM)，支持 .NETCore 2.1+ 或 .NETFramework 4.0+ 或 Xamarin。

有一个强大的ORM，也方便我们开发一个代码生成器。

一般情况下，我们开发数据库相关的应用，主要分为三种code first、db first、model first

常用前二种，

* code first，代码优先，数据库都是根据实体类生成，所有的关系，可以是逻辑关联，也可以是物理关联。
* DB First: 数据库优先，直接设计表结构，用设计工具生成表，设计主键，外键、索引，关联关系等。

当我们使用DB First时，设计好的数据库，我们怎么生成一些实体类、通用的代码、控制器、服务层、Dto呢。今天我来给大家介绍一下FreeSql项目中的一些工具。当然，不使用此ORM的小伙伴也能使用此工具，因为他是通用。

## **FreeSql.Generator 命令行方式**

通过几行命令，就可实现生成项目中通用的代码结构，不需要复制一段代码后修改，加快开发速度，减少重复劳动，少用一根头发。

由于每个人的项目结构，代码位置各不相同，对于ORM来说，不同的业务逻辑各不相同，所以该项目没有相应的模板，相信使用过Razor的同学一定能实现自己的模板。

**FreeSql.Generator 命令行**

* 对于此工具的使用可参考 [https://github.com/dotnetcore/FreeSql/wiki/DbFirst](https://github.com/dotnetcore/FreeSql/wiki/DbFirst" \t "https://www.cnblogs.com/igeekfan/p/_blank)
* 源码位置 [https://github.com/dotnetcore/FreeSql/tree/master/Extensions/FreeSql.Generator](https://github.com/dotnetcore/FreeSql/tree/master/Extensions/FreeSql.Generator" \t "https://www.cnblogs.com/igeekfan/p/_blank)
* 前提是本地安装了.net core 3.1 的sdk.

怎么使用呢。

1. 安装 dotnet-tool 生成实体类

dotnet tool install -g FreeSql.Generator

1. 新建目录，在地址栏输入 cmd 快速打开命令窗口，输入命令：

FreeSql.Generator --help

我们可以看到

C:\Users\igeekfan\Desktop\code>FreeSql.Generator --help

\_\_\_\_ \_\_\_\_ \_\_

/ \_\_/ \_\_\_\_ \_\_\_ \_\_\_ / \_\_/ \_\_\_ \_ / /

/ \_/ / \_\_// -\_)/ -\_) \_\ \ / \_ `/ / /

/\_/ /\_/ \\_\_/ \\_\_/ /\_\_\_/ \\_, / /\_/

/\_/

# Github # https:*//github.com/2881099/FreeSql v1.5.0*

使用 FreeSql 快速生成数据库的实体类

更新工具：dotnet tool update -g FreeSql.Generator

# 快速开始 #

> FreeSql.Generator -Razor 1 -NameOptions 0,0,0,0 -NameSpace MyProject -DB "MySql,Data Source=127.0.0.1;..."

-Razor 1 \* 选择模板：实体类+特性

-Razor 2 \* 选择模板：实体类+特性+导航属性

-Razor "d:\diy.cshtml" \* 自定义模板文件

-NameOptions \* 总共4个布尔值，分别对应：

# 首字母大写

# 首字母大写,其他小写

# 全部小写

# 下划线转驼峰

-NameSpace \* 命名空间

-DB "MySql,Data Source=127.0.0.1;Port=3306;User ID=root;Password=root;Initial Catalog=数据库;Charset=utf8;SslMode=none;Max pool size=2"

-DB "SqlServer,Data Source=.;Integrated Security=True;Initial Catalog=数据库;Pooling=true;Max Pool Size=2"

-DB "PostgreSQL,Host=192.168.164.10;Port=5432;Username=postgres;Password=123456;Database=数据库;Pooling=true;Maximum Pool Size=2"

-DB "Oracle,user id=user1;password=123456;data source=//127.0.0.1:1521/XE;Pooling=true;Max Pool Size=2"

-DB "Sqlite,Data Source=document.db"

-DB "Dameng,server=127.0.0.1;port=5236;user id=2user;password=123456789;database=2user;poolsize=2"

Dameng 是国产达梦数据库

-Filter Table+View+StoreProcedure

默认生成：表+视图+存储过程

如果不想生成视图和存储过程 -Filter View+StoreProcedure

-Match 正则表达式，只生成匹配的表，如：dbo\.TB\_.+

-FileName 文件名，默认：{name}.cs

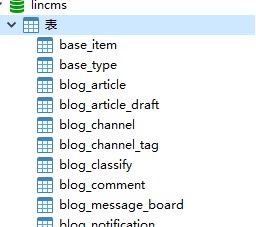
-Output 保存路径，默认为当前 shell 所在目录

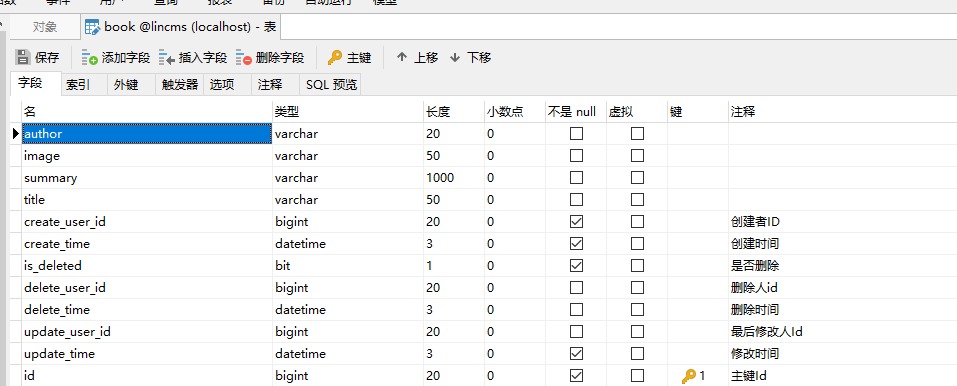
推荐在实体类目录创建 gen.bat，双击它重新所有实体类

* 更新命令行

dotnet tool update -g FreeSql.Generator

1. 这里lin-cms-dotnetcore这个项目来测试。

[](https://pic.downk.cc/item/5ef1c55614195aa5940207cf.jpg)

[](https://pic.downk.cc/item/5ef1c57914195aa594022b44.jpg)

* 数据库表名是下划线，字段也是下划线方式。
* -Razor 指定 第一个模板
* -NameOptions 0,0,0,1 最后一个1，代表 下划线转驼峰，满足C#命名规则
* -NameSpace 指定了命名空间 LinCms.Core.Entities
* -DB 就是数据库的相关配置
* mysql 本地地址 127.0.0.1 3306端口 用户名 root 密码123456 数据库 lin-cms
* -Match book 这样就能只生成book，支持正则表达式，如 -Math lin\_user 就会生成以lin\_user开头的表。如dbo.TB\_.+，会生成以TB开头的表。即只生成匹配的表

1. 执行此命令。

FreeSql.Generator -Razor 1 -NameOptions 0,0,0,1 -NameSpace LinCms.Core.Entities -DB "MySql,Data Source=127.0.0.1;Port=3306;User ID=root;Password=123456;Initial Catalog=lincms;Charset=utf8;SslMode=none;Max pool size=2"

这时候代码已经生成了

[](https://pic.downk.cc/item/5ef1c60f14195aa59402b271.jpg)

其中一个代码 生成如下。这些类是partial ，熟悉C#的同学，应该知道，类的定义使用此关键字，我们能在不同的地方为该类扩展。以防止重新同步数据库的结构时，丢失改动的字段。

namespace LinCms.Core.Entities {

[JsonObject(MemberSerialization.OptIn), Table(Name = "book")]

public partial class Book {

*/// <summary>*

*/// 主键Id*

*/// </summary>*

[JsonProperty, Column(Name = "id", IsPrimary = true, IsIdentity = true)]

public long Id { get; set; }

[JsonProperty, Column(Name = "author", DbType = "varchar(20)")]

public string Author { get; set; } = string.Empty;

[JsonProperty, Column(Name = "image", DbType = "varchar(50)")]

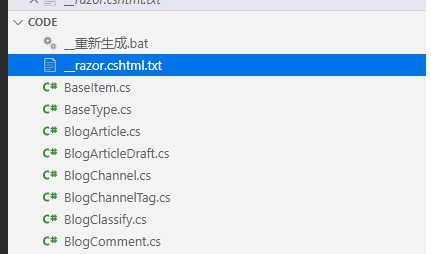
public string Image { get; set; } = string.Empty;

*//更多xxx*

}

}

* 最终效果图如下

[](https://pic.downk.cc/item/5ef1c74614195aa59403b019.jpg)

此时会生成二个文件  
\_\_重新生成.bat,下次重新点击他就能重新生成实体类了。

FreeSql.Generator -Razor "\_\_razor.cshtml.txt" -NameOptions 1,1,0,1 -NameSpace MyProject -DB "MySql,Data Source=127.0.0.1;Port=3306;User ID=root;Password=123456;Initial Catalog=lincms;Charset=utf8;SslMode=none;Max pool size=2" -FileName "{name}.cs"

上面的命令-Razor 指定了这个txt文件 \_\_razor.cshtml.txt

我们可以定义自己的模板，以生成符合自已业务的的代码，从而实现快速开发。

我们可以看下模板中的文件内容,他就是asp.net下的mvc 结构下的razor后端模板渲染,把这个.txt后缀去掉，就很明了了。对于asp.net mvc的razor，我们可以将控制器下方法的值替换掉cshtml中的值。这个过程是有一个类库在帮我们实现的,叫RazorEngine，不过那个是.net framework下的实践。[.NET Framework 下的RazorEngine代码生成介绍](https://blog.csdn.net/q710777720/article/details/91358617" \t "https://www.cnblogs.com/igeekfan/p/_blank)

@using FreeSql.DatabaseModel;@{var gen = Model as RazorModel;

Func<string, string> GetAttributeString = attr => {

if (string.IsNullOrEmpty(attr)) return null;

return string.Concat(", ", attr.Trim('[', ']'));

};

Func<DbColumnInfo, string> GetDefaultValue = col => {

if (col.CsType == typeof(string)) return " = string.Empty;";

return "";

};

}*//xxx*namespace @gen.NameSpace {

@if (string.IsNullOrEmpty(gen.table.Comment) == false) {

@:*/// <summary>*

@:*/// @gen.table.Comment.Replace("\r\n", "\n").Replace("\n", "\r\n /// ")*

@:*/// </summary>*

}

[JsonObject(MemberSerialization.OptIn)@GetAttributeString(gen.GetTableAttribute())]

public partial class @gen.GetCsName(gen.FullTableName) {

@foreach (var col in gen.columns) {

if (string.IsNullOrEmpty(col.Coment) == false) {

@:*/// <summary>*

@:*/// @col.Coment.Replace("\r\n", "\n").Replace("\n", "\r\n /// ")*

@:*/// </summary>*

}

@:@("[JsonProperty" + GetAttributeString(gen.GetColumnAttribute(col)) + "]")

@:public @gen.GetCsType(col) @gen.GetCsName(col.Name) { get; set; }@GetDefaultValue(col)

@:

}

}

@gen.GetMySqlEnumSetDefine()

}

## **RazorEngine.NetCore**

到了.NET Core时代，我看了下FreeSql.Generator用的这个类库[RazorEngine.NetCore](https://github.com/fouadmess/RazorEngine" \t "https://www.cnblogs.com/igeekfan/p/_blank)，实现动态操作cshtml，生成需要的文本。

Razor Engine是基于微软Razor解析的模板引擎，它允许你使用Razor语法构建动态模板，你只需要使用Engine的静态方法，Engine.Razor.RunCompile等。

创建一个控制台应用，然后安装包。

Install-Package RazorEngine.NetCore

using RazorEngine;using RazorEngine.Templating; *// For extension methods.*

string template = "Hello @Model.Name, welcome to RazorEngine!";

var result = Engine.Razor.RunCompile(template, "templateKey", null, new { Name = "World" });

Console.WriteLine(result);

* 输出如下内容

Hello World, welcome to RazorEngine!

此处使用的**RunCompile**方法是扩展方法，您需要引用RazorEngine.Templating命名空间。

The "templateKey" 保持唯一值，比如使用guid值。字符串，并且你可以根据此字符串key重新运行缓存的模板。

如果再次根据此key，可使用原本的模板。

var result = Engine.Razor.Run("templateKey", null, new { Name = "Max" });

* 会输出如下内容

Hello Max, welcome to RazorEngine!

上面中的RunCompile第三个参数，传null,因为我们第四个参数使用的是匿名类，

根目录创建一个HelloWord.cshtml，要选择属性，->如果较新则复制 内容，

Hello @Model.Name, welcome to RazorEngine!

控制台如下代码。

string templateFilePath = "HelloWorld.cshtml";var templateFile = File.ReadAllText(templateFilePath);string templateFileResult = Engine.Razor.RunCompile(templateFile, Guid.NewGuid().ToString(), null, new

{

Name = "World"

});

Console.WriteLine(templateFileResult);

* 控制台输出

Hello World, welcome to RazorEngine!

* 使用强类型 CopyRightUserInfo.cs生成一个版权所有

using System;namespace OvOv.Razor

{

public class CopyRightUserInfo

{

public string UserName { get; set; }

public string EmailAddress { get; set; }

public DateTime CreateTime { get; set; }

public string FileRemark { get; set; }

}

}

根目录创建一个CopyRightTemplate.cshtml，要选择属性，->如果较新则复制 内容，

@{

var gen = Model as OvOv.Razor.CopyRightUserInfo;

}*//=============================================================// 创建人: @gen.UserName// 创建时间: @gen.CreateTime// 邮箱： @gen.EmailAddress//==============================================================*

控制台如下代码。

string copyRightTemplatePath = "CopyRightTemplate.cshtml";var copyRightTemplate = File.ReadAllText(copyRightTemplatePath);string copyRightResult = Engine.Razor.RunCompile(copyRightTemplate, Guid.NewGuid().ToString(), typeof(CopyRightUserInfo), new CopyRightUserInfo

{

CreateTime = DateTime.Now,

EmailAddress = "710277267@qq.com",

UserName = "IGeekFan"

});

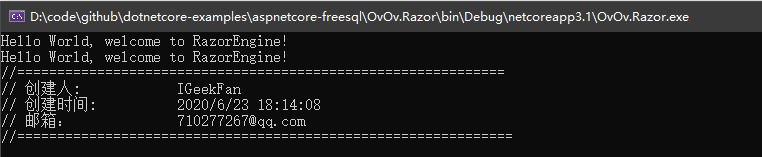
Console.WriteLine(copyRightResult);

Console.ReadKey();

* 控制台输出

*//=============================================================// 创建人: IGeekFan// 创建时间: 2020/6/23 18:14:08// 邮箱： 710277267@qq.com//==============================================================*

全放到控制台下，输出如下结果。代码生成器最重要的一点解决了，我们就能实现自己的代码生成器，先构建自己的模板，实现输入（命令行，WPF，WEB端及更多），输出（生成文件）。

[](https://pic.downk.cc/item/5ef1d60914195aa5940ff447.jpg)

* 以上源码已放到示例代码中 [https://github.com/luoyunchong/dotnetcore-examples/blob/master/aspnetcore-freesql/OvOv.Razor/Program.cs](https://github.com/luoyunchong/dotnetcore-examples/blob/master/aspnetcore-freesql/OvOv.Razor/Program.cs" \t "https://www.cnblogs.com/igeekfan/p/_blank)

## **源码解析**

首先这是一个控制台应用，Main(string[] args)可接收多个参数。

1. 处理无参数，--help
2. 处理args数组，解析出所有的参数，如果没有设置，则为默认值。（处理一些参数异常问题）最重要的是根据-Razor，选定对应的模板。

ArgsRazor=""//根据-Razor 1 不是2 还是模板的路径，取出的模板文本值。var razorId = Guid.NewGuid().ToString("N");RazorEngine.Engine.Razor.Compile(ArgsRazor, razorId);

1. 根据数据库连接串，取出参数过滤后的表，视图，存储过程
2. 循环数据库的表等，

* model为模板中需要的数据
* razorId与上文的razorId相同，
* sw为生成后的文本保存的值。

var sw = new StringWriter();var model = new RazorModel(fsql, ArgsNameSpace, ArgsNameOptions, tables, table);

RazorEngine.Engine.Razor.Run(razorId, sw, null, model);

1. 将sw字符串保存生成类.cs文件（根据参数配置生成文件名）
2. 另外生成一个\_\_重新生成.bat,\_\_razor.cshtml.txt，方便后续用户重新生成实体类。

## **FreeSql.Tools**

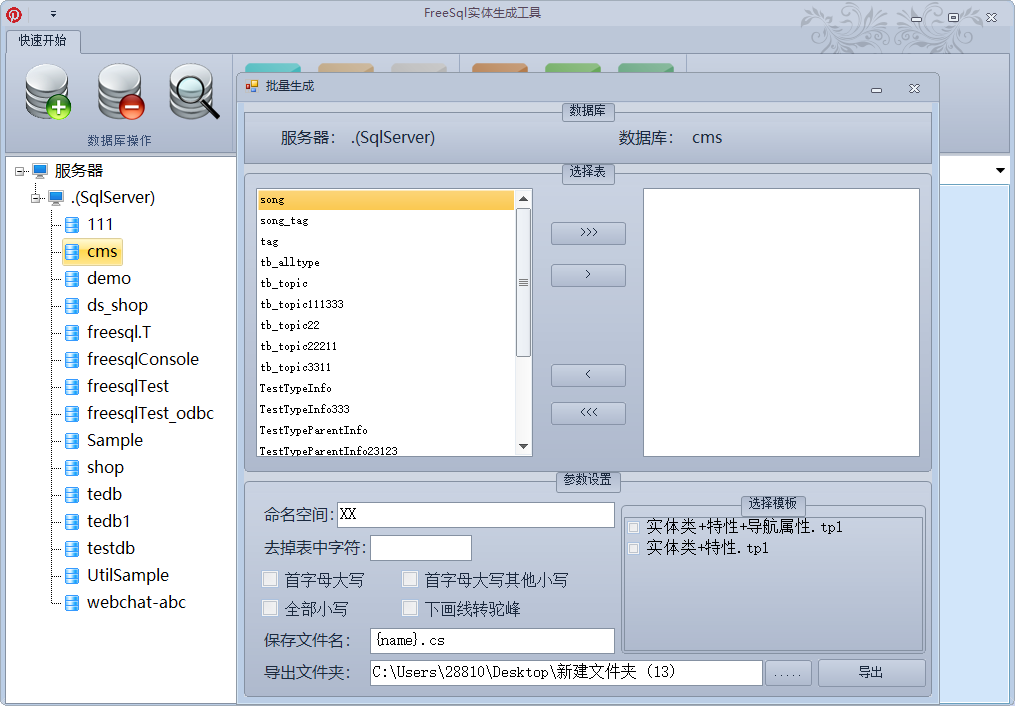
这是 FreeSql 衍生出来的辅助工具包，内含生成器等功能；作者：mypeng1985  
因为这个不兼容mac,linux,所以作者建议使用dotnet-tool 命令行工具生成实体类，从而支持MAC/Linux系统。对于不是使用FreeSql的开发者，也能使用此工具，你只需要修改对应的模板即可。

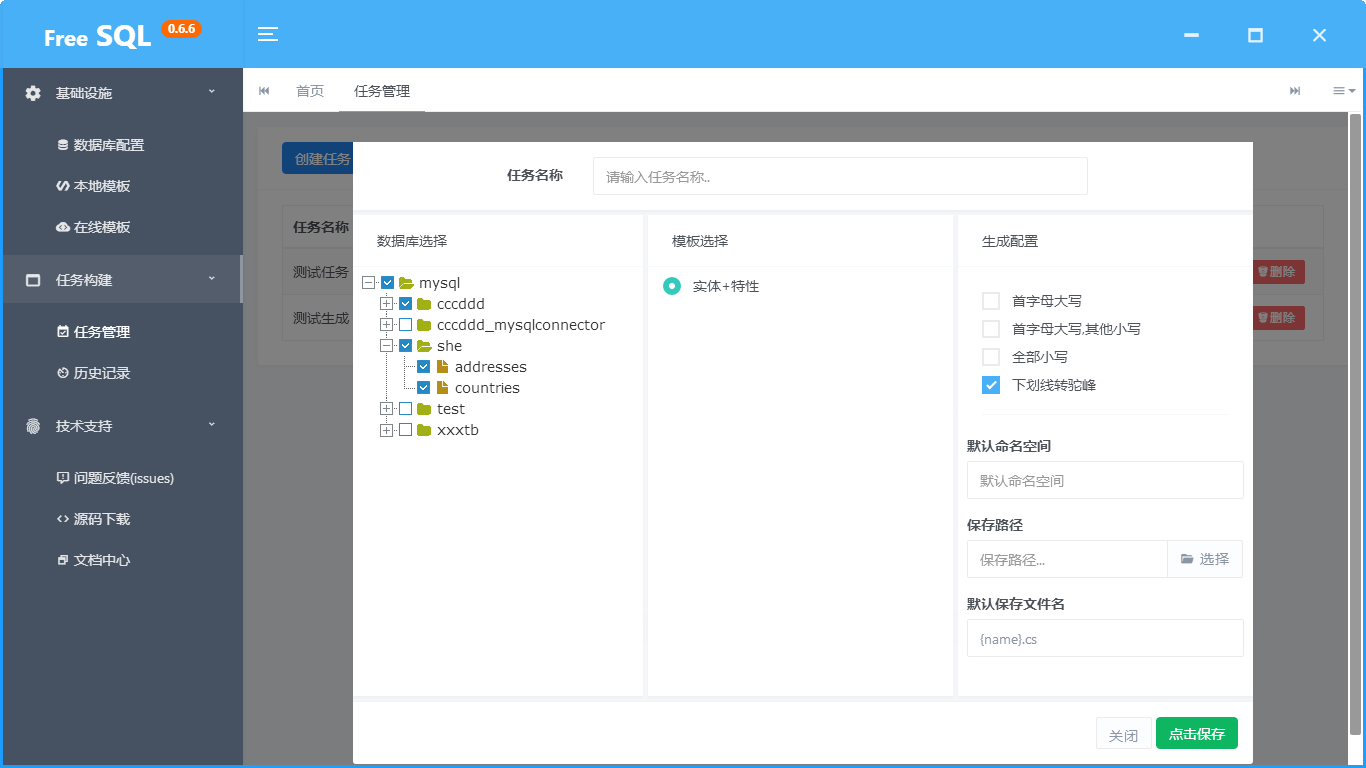
使用方式：不多介绍。

* [https://github.com/2881099/Freesql.tools](https://github.com/2881099/Freesql.tools" \t "https://www.cnblogs.com/igeekfan/p/_blank)
* 分为WPF ，WinForm + DSkin 版本（套网页）
* 看了下代码，底层生成代码逻辑也是用的RazorEngine [.NET Framework 下的RazorEngine代码生成介绍](https://blog.csdn.net/q710777720/article/details/91358617" \t "https://www.cnblogs.com/igeekfan/p/_blank)

## **FreeSql官方群 4336577**

##### **预览图**

[](https://pic.downk.cc/item/5ef1e12314195aa59418d82e.png)

[](https://pic.downk.cc/item/5ef1e11a14195aa59418cfb9.png)