Database2Sharp 代码生成工具

使用帮助

V6.0

伍华聪

2017 年 09 月

目录

1 前言

Database2Sharp是一款代码生成工具和数据库文档生成工具，该工具从2005年开始至今，一直伴随着我们的客户和粉丝们经历着过各种各样的项目开发，在实际开发中能带来效率的提高及编程的快乐。

Database2Sharp 是一款主要用于 C#代码生成以及数据库文档生成的工具，软件支持 Oracle、SqlServer、MySql、PostgreSQL、Sqlite、Access 以及国产达梦等数据库的代码生成，可以生成各种架构代码、生成 Winform 界面代码、Web 界面代码（包括 EasyUI 和 BootstrapWeb 界面）、Entity Framework 实体框架代码、导出数据库文档、浏览数据库架构、查询数据、生成 Sql 脚本等，还整合自定义模板和数据库信息的引擎，方便编写自定义模板调试和开发。生成的框架代码支持多种数据库一起使用，也支持不同业务的数据库切割为多个库进行使用，是一种适应性非常强、弹性很好的应用框架。

Database2Sharp 推荐采用软件功能“**Enterprise Library 代码生成**”来生成项目代码，这个架构体系生成整个项目工程框架，包含实体类、数据访问类、业务类、Web 页面代码、WCF 相关服务层（可选）、Web API 服务层（可选），以及各种服务的调用包装层代码等。该架构利用泛型及缓存机制，良好的架构极大简化代码，强大完善的基类机制使您甚至不用编写一行代码就能顺利运行。一个简单点击几次鼠标就能完成一周代码量的代码生成工具，效率惊人、友好体贴，真正的开发好伴侣。

当然，开发的过程是一个繁复、精细的过程，因此 Database2Sharp 也吸收了来自我们自己的实际需求，以及很多同仁朋友的宝贵意见，一直在改进，一直努力做到更好，以求达到一个更加完美、更加易用的境界。

另外，我们也在代码生成工具中增加“Entity Framework 实体框架代码生成”的架构，该框架利用微软最新的 Entity Framework 实体框架，采用基于泛型的仓储模式实体框架（Generic Repository Pattern)来构建，框架的结构类似“**Enterprise Library 代码生成**”的结构，底层数据访问层采用新的 LINQ 操作。

如果您有好的意见以及建议，如能不吝赐教，则感激不尽，可以通过邮件 wuhuacong@163.com 或者 QQ 6966254 与我们联系。

2 几种开发框架介绍

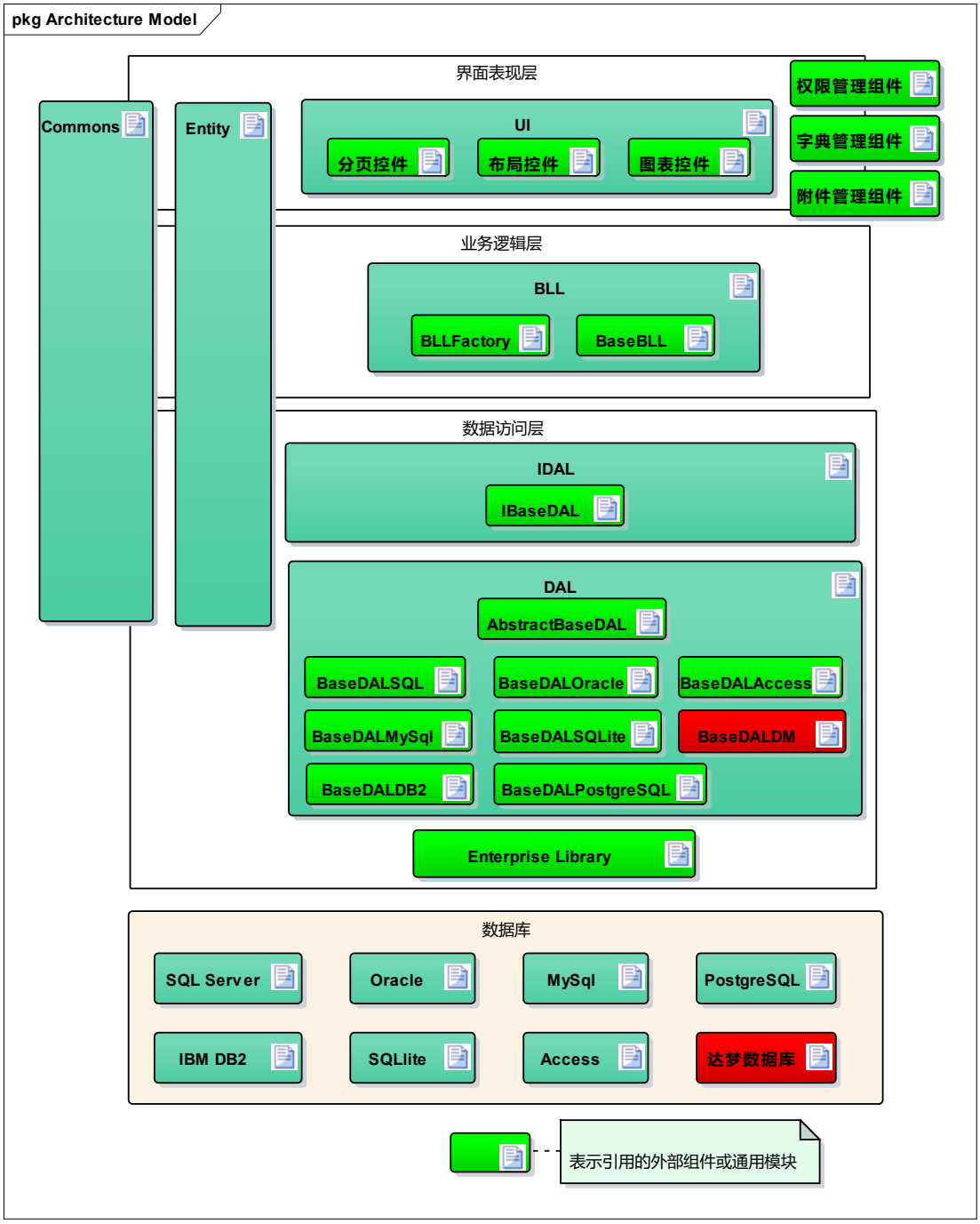
2.1 传统 Winform 开发框架

这里指的传统 Winform 开发框架（简称 Winform 开发框架），就是利用数据库中间件，直接访问数据库的一种应用框架，根据数据库管理系统部署的位置的不同，可能分为单机版（如 Access 数据库、Sqlite 数据库等），局域网网络版（如 SqlServer、Oracle、MySql、PostgreSQL 等数据库）。局域网网络版，一般需要的是把数据库部署在局域网另外一个电脑上，这样应用和数据库分开，也有利于性能的提高和数据的分享。

这种开发模式，在 .NET 里面，就是利用基于 ADO.NET 的操作，实现数据的直接访问，是一种比较常规的开发模式；不过问题就是，不能通过互联网进行数据的访问，只能在单机或者局域网的环境下进行业务管理系统的部署和使用，相对目前很多分布式的应用来讲，有一定的局限性。

不过这种方式也是很常见的模式，常用在一些内部业务管理系统或者一些工作流系统的维护上，由于 Winform 界面的体验性比较好，数据也能有效管理控制，开发部署成本也相对较低，系统开发效率以及应用性能也比较高，因此是一种比较常见的框架表现模式。

这种 Winform 开发框架 是通过数据访问层来访问各种指定的数据库，如 SqlServer 或者 Oracle 等，一般底层封装好一点的框架，基本上都会支持多种数据库，方便在不同的业务中使用。我们的传统 Winform 开发框架的架构设计图，如下所示。



2.2 WCF 开发框架

除了以上的传统的 Winform 开发框架，基于 Winform 的技术和 WCF 的分布式技术，形成了另外一种 Winform 开发框架，即为 WCF 开发框架，我这个 WCF 开发框架的介绍也比较多，绝大多数都是来源于真实的项目应用。其实 WCF 技术，即可用于 Winform 上，也可以用于 Web 上，WCF 技术可以用在很多领域，如 Web 开发、类似 Socket 通讯的即时通讯应用等，这里介绍的 WCF 开发

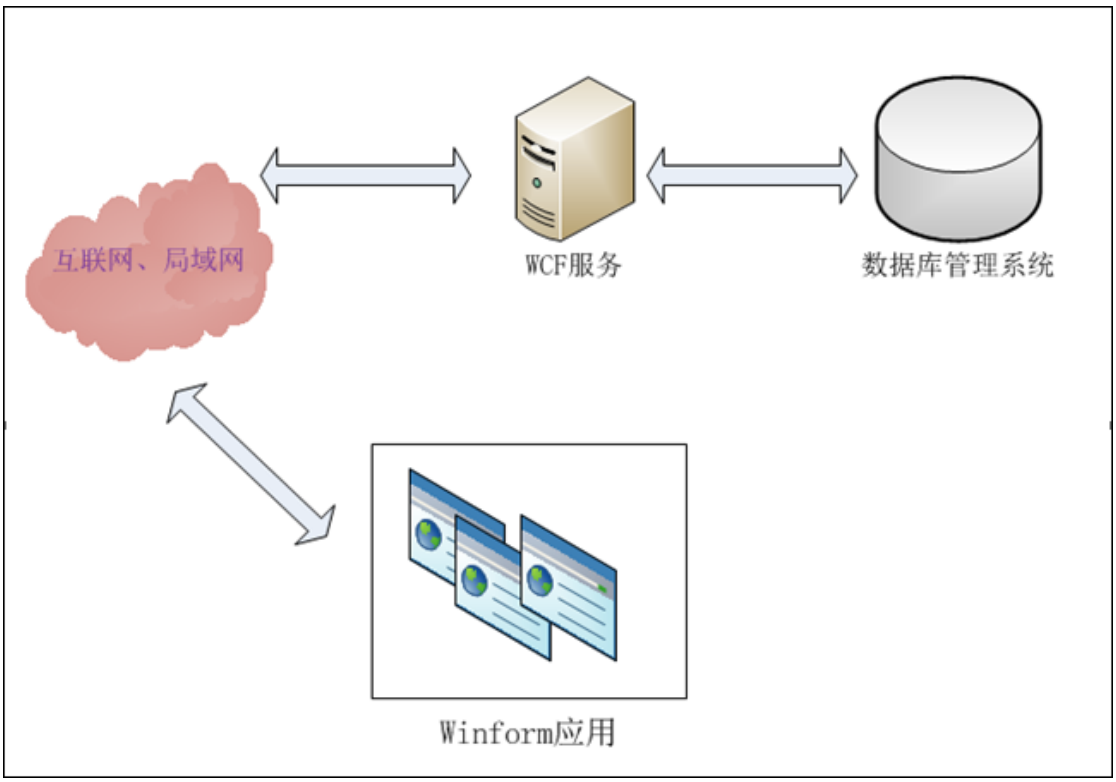
框架，是基于 Winform 的基础上使用的 WCF 技术的应用开发框架。

这里定义的传统 WCF 开发框架，是指利用 WInform 框架的模式，来承载 B/S 的方式获取数据进行展示，本地不存储数据。就是直接获取数据并在列表控件或者其他基础控件上显示数据。

由于 WCF 框架应用了很多新的技术，以及是基于分布式网络环境的应用，因此，需要考虑服务的部署，数据访问的安全性（用户名密码访问、X509 证书加密、其他授权访问），数据响应以及错误处理等方方面面，如果是第一次接触，没有很好的指导及案例学习，可能消化相关的概念并理解，就需要一个不短的学习过程。

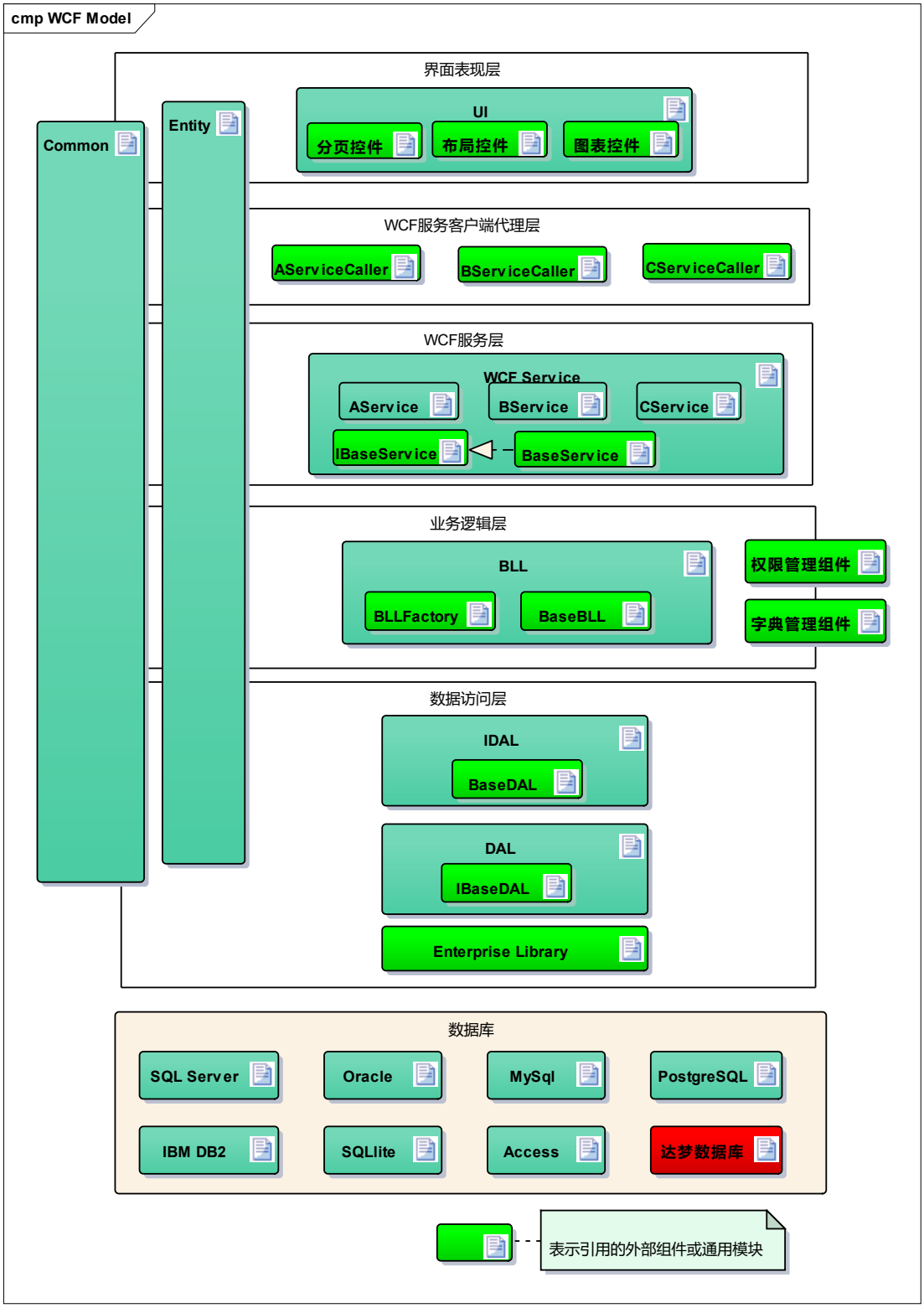
传统的 WCF 开发框架，由于本地不存储数据，实时通过 WCF 服务代理，向 WCF 服务请求数据，获取数据后进行显示的过程，开发思路相对比较简单，因此这种方式在很多 WCF 应用上，是比较常见的一种。

传统的基于 Winform 的 WCF 开发框架，界面可以和上面的 Winform 开发框架界面一样，不过它的获取数据的机制不同，它不是通过数据访问类对本地数据库进行访问，而是通过服务器公布 WCF 服务来获取不同数据对象，然后在 Winform 界面中进行呈现，如下面是它的一个访问机制的示意图。



对于这个传统的 WCF 开发框架的架构，它其实是通过 Winform 的客户端 WCF

代理类，实现和远端服务器的 WCF 服务进行通讯，一般是基于 Soap 协议的 XML 格式，当然可以很好的配置加密机制，如 X509 证书加密，这样传输数据就比较安全；同时对于 WCF 服务，还可以采用基于自定义的用户名、密码的验证方式来有效提高 WCF 服务接口的安全性。以下是 WCF 开发框架的架构设计图，我们从中可以看出，它的应用程序的界面表现层是和 WCF 服务客户端代理层有关联，而代理层和 WCF 服务层有关联，这种隔绝了应用程序直接访问数据库的弊端，提高数据安全性，同时也使得应用程序实现了分布式的开发应用。



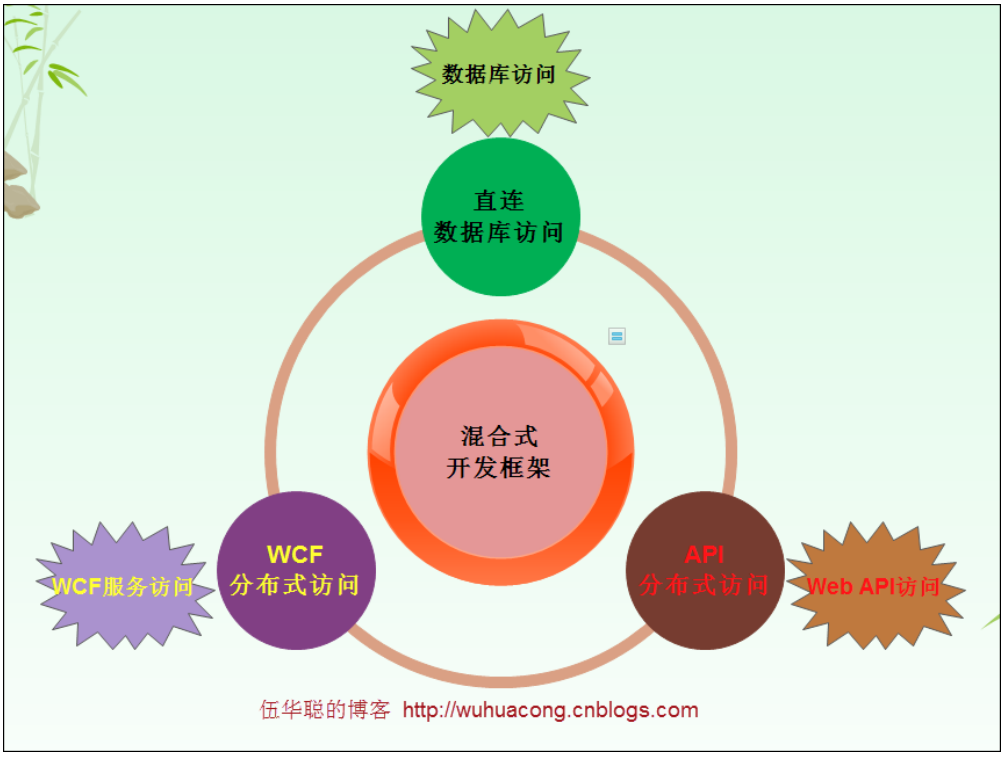
2.3 混合型开发框架

混合型框架可以看成是 Winform 框架高级版本，除了它本身是一个完整的业务系统外，它外围的所有辅助性模块（如通用权限、通用字典、通用附件管

理、通用人员管理、通讯录管理。。。。）均都实现了这种混合型的框架，因此使用非常方便，整个框架如果简化来看，就是在原有的 Winform 界面层，用接口调用方式，避免和业务逻辑类的紧耦合关系。由于他是通过接口方式的调用方式，它本身又可以通过配置指定指向 WCF 的实现（既适应 Winform 集成，也适应 WCF 集成），因此也囊括了 WCF 框架的一切特点。

同时，混合框架也集成了对 Web API 接口的访问封装操作，Web API 是一种应用接口框架，它能够构建 HTTP 服务以支撑更广泛的客户端（包括浏览器，手机和平板电脑等移动设备）的框架， ASP.NET Web API 是一种用于在 .NET Framework 上构建 RESTful 应用程序的理想平台。

《混合式开发框架》混合了传统《Winform 开发框架》、《WCF 开发框架》和 Web API 接口框架的特点，可以在直接访问数据库、利用 WCF 服务获取数据、利用 Web API 服务获取数据三者之间自由切换，统一了系统界面层对业务服务的调用模式，所有组件模块均实现三种方式的调用，是一种弹性化非常好的框架应用，既可用于单机版软件或者基于局域网内的应用软件，也可以用于分布式技术的互联网环境应用，是一种成熟稳定、安全高效的技术框架。



2.3.1 混合型框架的特点

混合型框架具有下面几个特点：

1）**环境适应性强，模块可重用性高**。由于混合型框架，既可以用于传统 Winform 系统开发，也可以用于 WCF 分布式系统开发，还可以用于轻型高效的 Web API 的分布式系统开发，因此环境适应性强；而且由于模块具有这些特点，可重用性更高，特别对于通用性的模块，更是具有无可替代的优越性。

2）**多种集成方式，响应性能更好。**如果是 Winform 程序，那么就使用直接访问数据库方式；如果是 WCF 调用方式，就使用 WCF 的专有通道进行数据处理，更好利用系统资源，高效进行数据处理；如果是 Web API 调用方式，那么就利用 Web API 的统一接入、轻型高效的特点，实现更方便快速的数据处理。

3）**独立配置，更少的代码修改。**所有通用模块，全部通过独立配置文件进行配置 WCF 的连接，减少主配置文件的复杂性；WCF 服务逻辑独立类库，可采用多种服务寄宿方式。Web API 每个服务的连接地址也是可以通过配置文件进行指定，根据需要采用 HTTP 或者 HTTPS 协议进行数据传输。

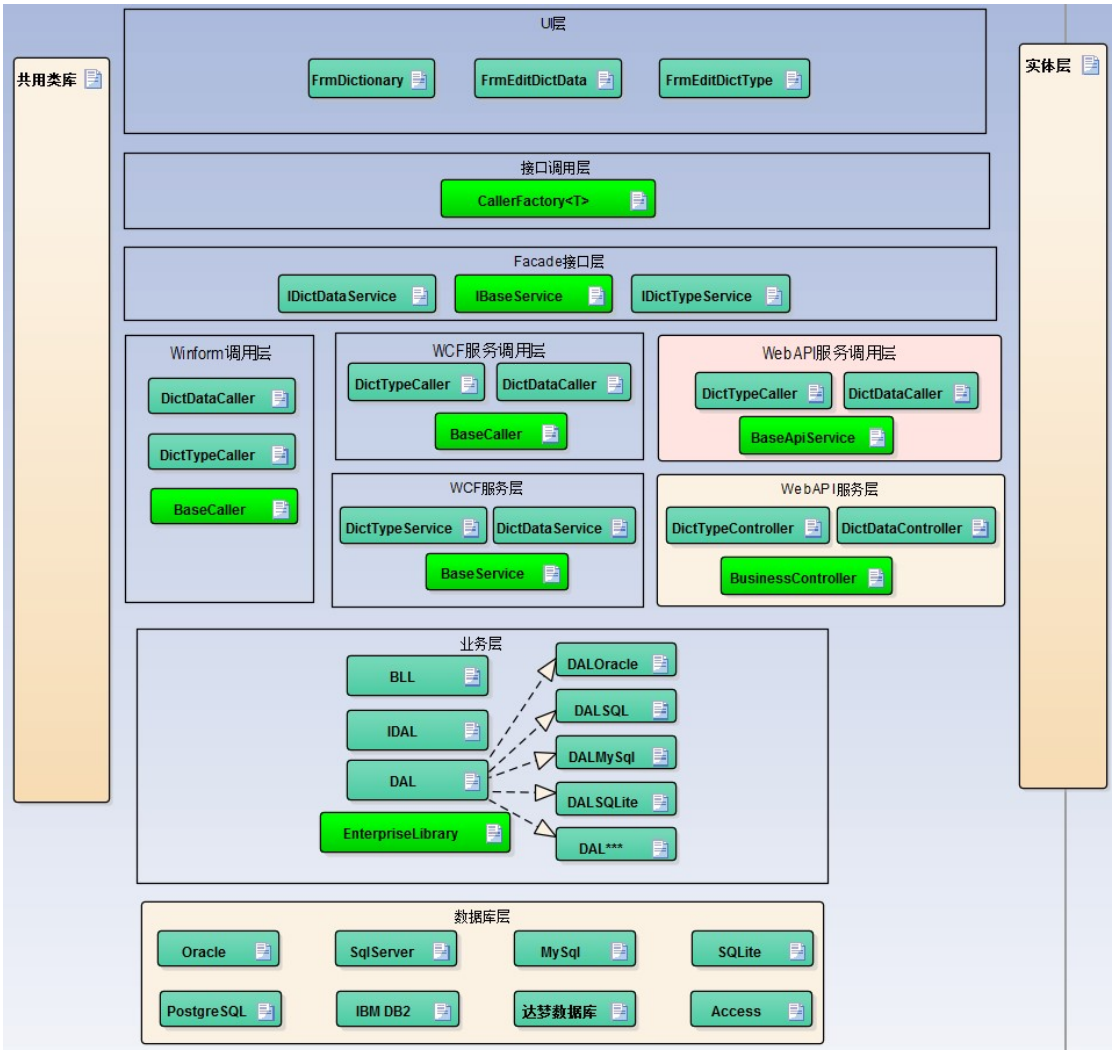
2.3.2 混合型框架总体设计思路

Winform 开发框架之混合型框架，还是秉承模块化的思路，可以把这个框架分为两大块，一块是主要业务系统模块（如备件管理系统），一块是各种辅助性模块（如通用权限、通用字典、通用附件管理、通用人员管理、通讯录管理。。。。），这种两块组合，就是一个完美的系统了。

整个系统的业务系统模块和辅助性模块，都是基于一个思路，通过接口调用开关，决定调用的是 WCF 服务层、Web API 服务层，还是 Winform 业务层（直接访问数据库），当然界面层的调用不管是调用 WCF 服务层、Web API 服务层还是 Winform 业务层，都是基于相同的接口，我们可以把它称为 **Façade 门面层**。辅助性模块则是多种常用模块的组合，它们可能是下面几种的常见模块：通用权限模块、通用字典模块、通用附件管理模块、通用人员管理、通讯录管理模块等等。

2.3.3 混合式框架的代码生成工具支持

当然，虽然混合型框架比传统的 Winform 框架和 WCF 开发框架更为通用，不过由于它引入了多一层，而且为了实现更多模块的分离，增加了一些设计上的复杂性，整个项目工程看起来显得稍微复杂一点，但是对于一个非常具有弹性化的系统架构来说是值得的，如下面就是一个以字典模块为例的混合型框架的内部结构。



从上图我们可以看到，整个混合型框架的架构，分为了 UI 层、接口调用层、Facade 接口层、Winform 调用层、WCF 服务调用层、Web API 服务调用层、业务层、实体层、以及数据库层等；其中的业务层还可以细化为 BLL 业务逻辑层、数据接口层 IDAL、数据访问层 DAL、实体层 Entity 等，整个模块通过实体层进行数据的传输载体。

为了实现更简单化的开发，更快更高效的完成混合型框架的开发工作，我扩展了我们的代码生成工具 Database2Sharp，使其支持这种混合型框架的代码生成工作，这样开发混合型框架就和开发其他两种 Winform 开发框架、WCF 开发框架一样，非常方便了。

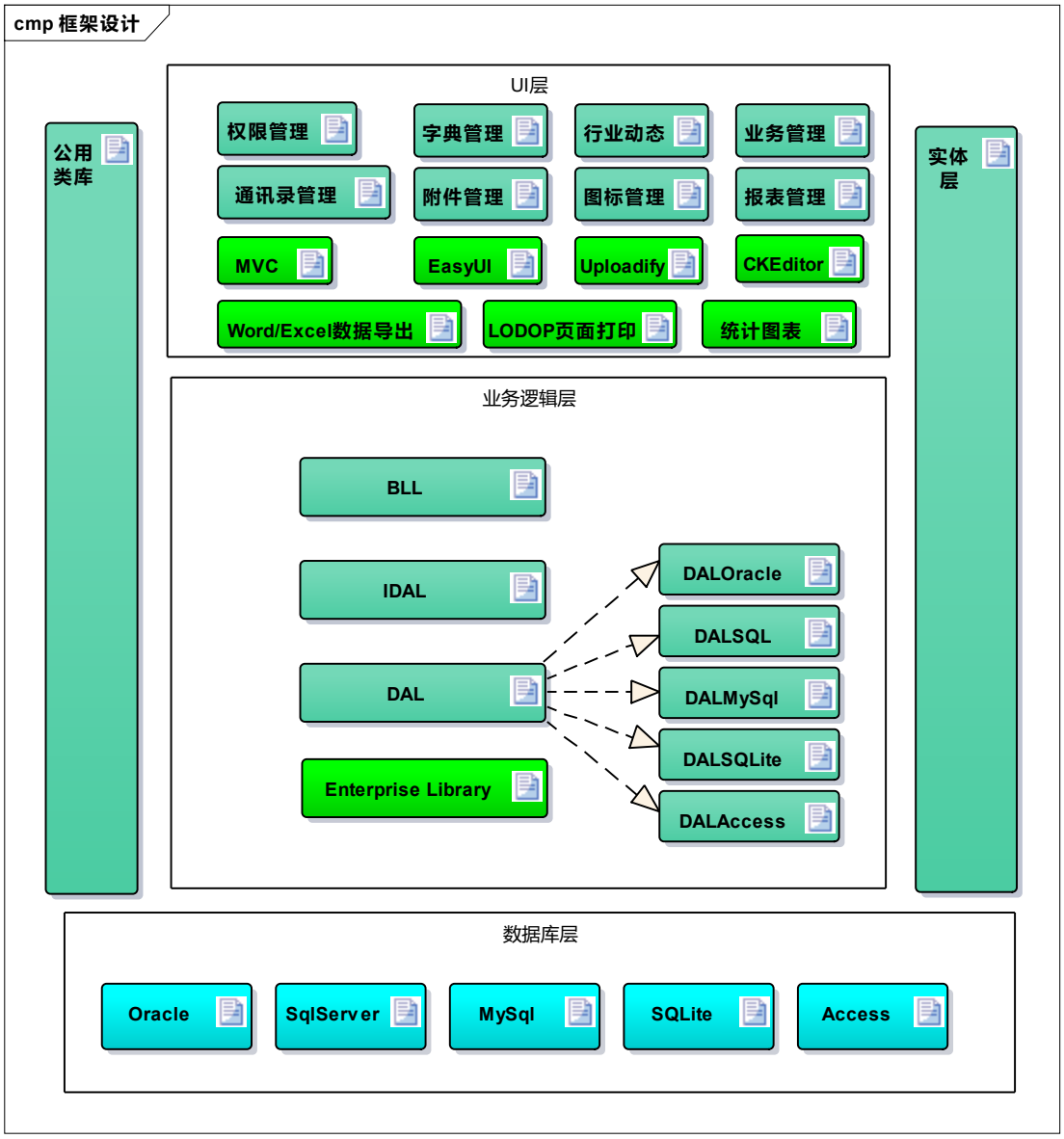
通过代码生成工具的无缝整合，就能够高效开发整个系统框架代码，以及基于各种架构的界面模块代码了。通过代码生成工具的辅助，我们可以高效、统一的实现各个业务模块的快速开发。

2.4 Web 开发框架

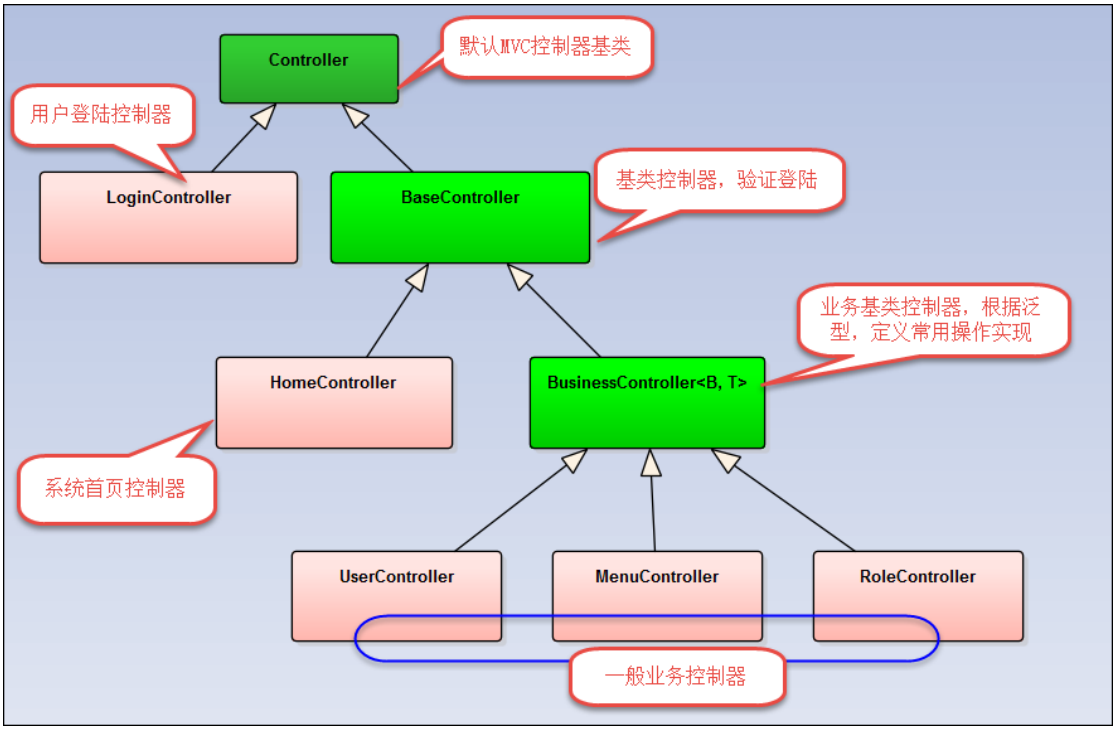
Web 开发框架，目前我们提供了两种 Web 界面风格和不同前端技术的界面框架，两者 Web 界面框架都是基于最新 MVC 技术的；其中一种是基于 EasyUI+JQuery 的界面组件方式，另一种是基于响应式框架 Bootstrap+JQuery 的方式。EasyUI 界面在做后台管理方面非常方便美观，Bootstrap 界面适用于多设备界面的支持，可以在后台界面、网站商城、手机微网站方面具有明显的优点。

2.4.1 基于 EasyUI+MVC 的 Web 开发框架

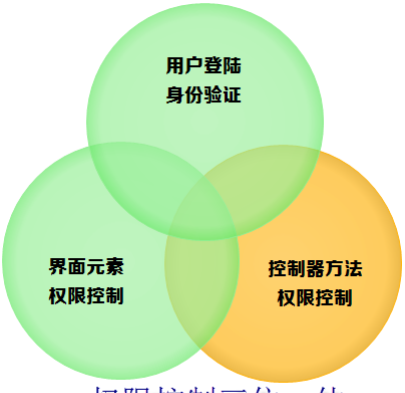
该 Web 开发框架，界面部分采用较新的技术，包括 MVC + JQuery，最新版本的 EasyUI，以及 zTree 树形控件、Uploadify 文件上传组件等模块，另外还结合了打印模块 LODOP 进行页面打印、文件 Word 或者 Excel 导出操作等，数据支持 Oracle、SqlServer、MySql、PostgreSQL、Sqlite、Access 等常规数据库，可通过配置进行自由切换，使用 Enterprise Library 模块进行数据访问的控制，使得数据访问更方便轻松。



Web 开发框架沿用了《Winform 开发框架》的很多架构设计思路和特点，对 Controller 进行了封装。使得控制器能够获得很好的继承关系，并能以更少的代码，更高效的开发效率，实现 Web 项目的开发工作，整个控制器的设计思路如下所示。

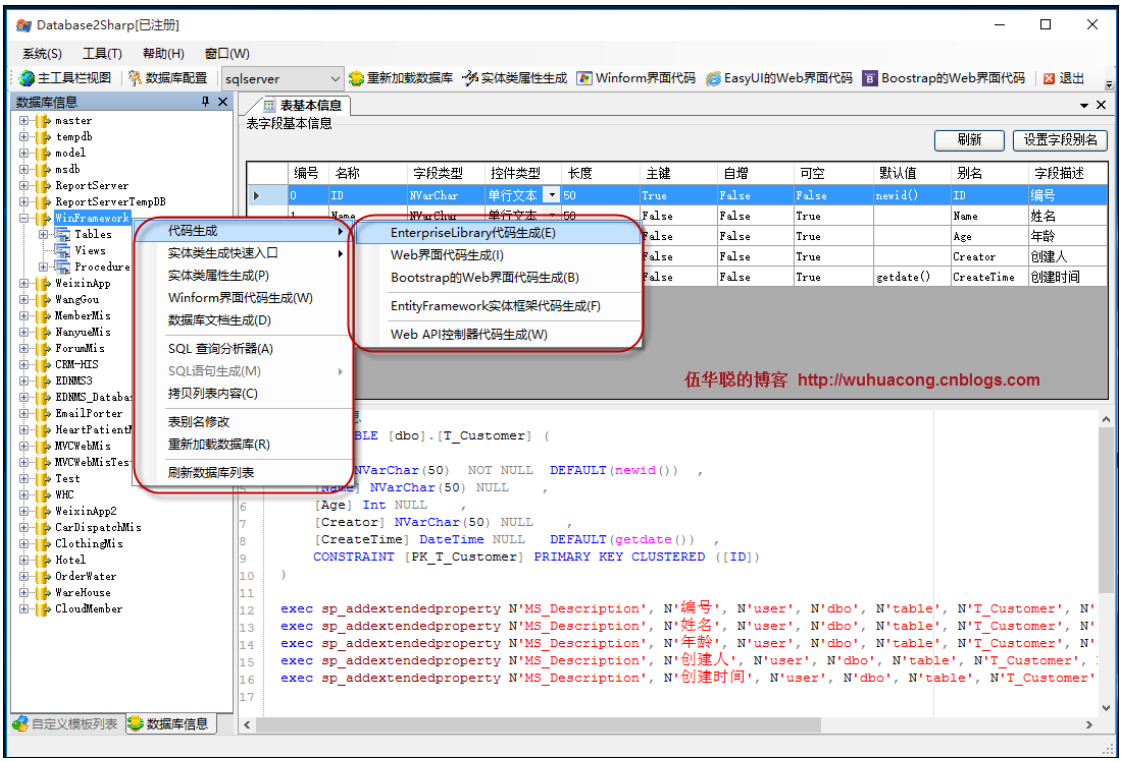


良好的控制器设计规则，可以为 Web 开发框架本身提供了很好用户访问控制和权限控制，使得用户界面呈现菜单、Web 界面的按钮和内容、Action 的提交控制，均能在总体权限功能分配和控制之下。

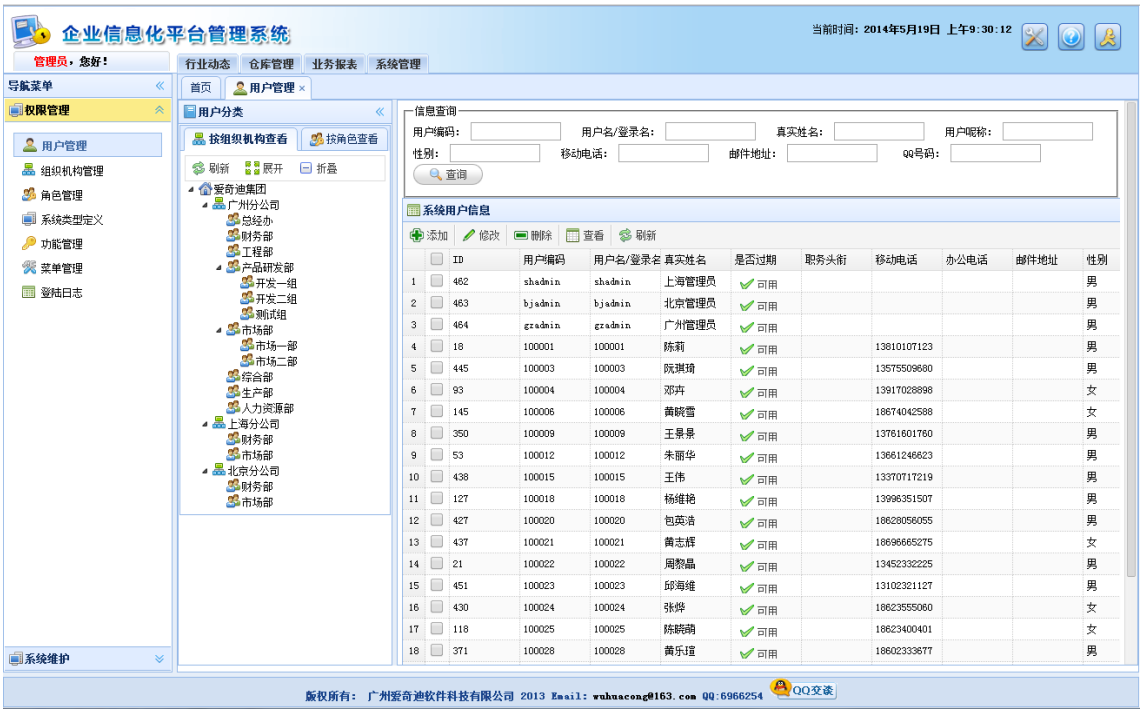


权限控制三位一体

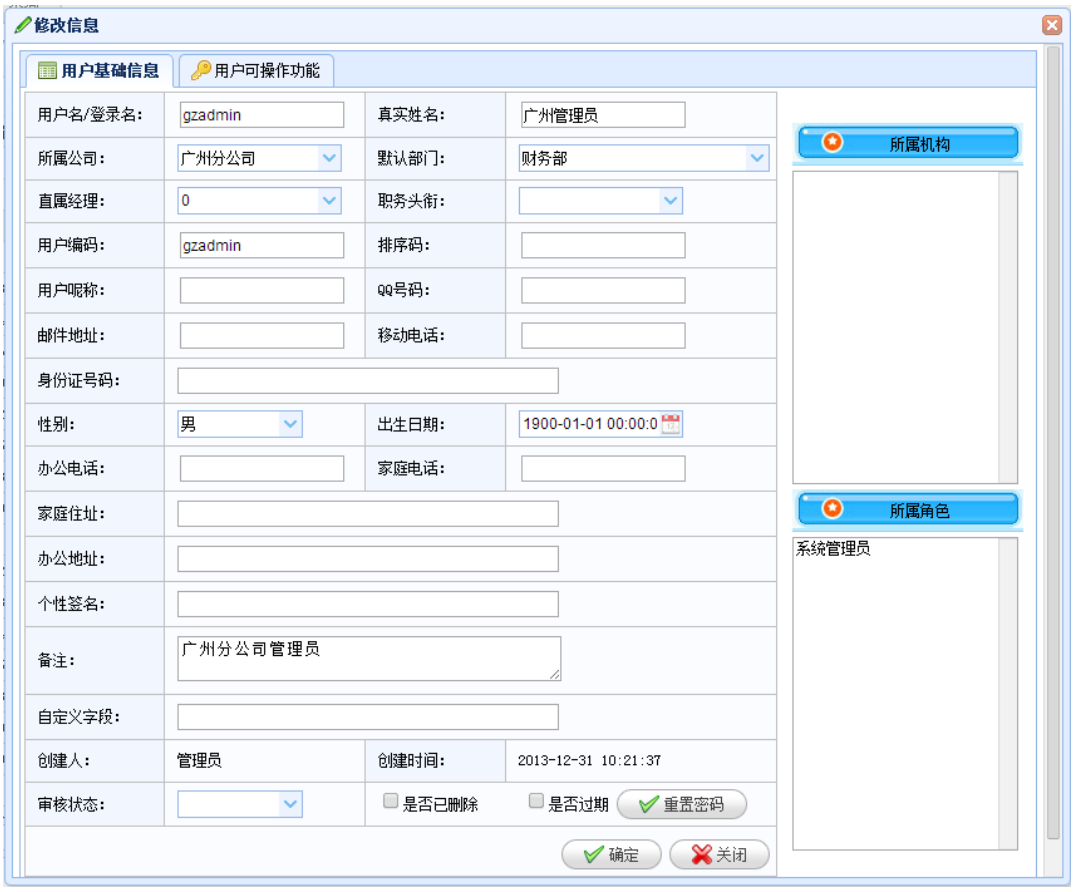
良好的架构使得无论在业务逻辑层、控制器层、Web 界面的 UI 层，均能提供统一的代码逻辑，这些代码均能通过代码生成工具 Database2Sharp 进行生成。Web 界面代码可以充分利用代码生成工具 Database2Sharp 的元数据信息，实现 Web 界面的快速生成。有效减少出错的几率，提高 Web 界面编码的开发效率和乐趣，更可以使得企业内部的编码模式进行高效的统一。



Web 框架的主体界面效果如下所示。



框架提供列表的展示和分页功能，对于单条记录，可以通过双击或者单击按钮，弹出对应的窗体界面进行数据处理。如下面是用户信息编辑界面效果。



2.4.2 基于 Bootstrap+MVC 的 Web 开发框架

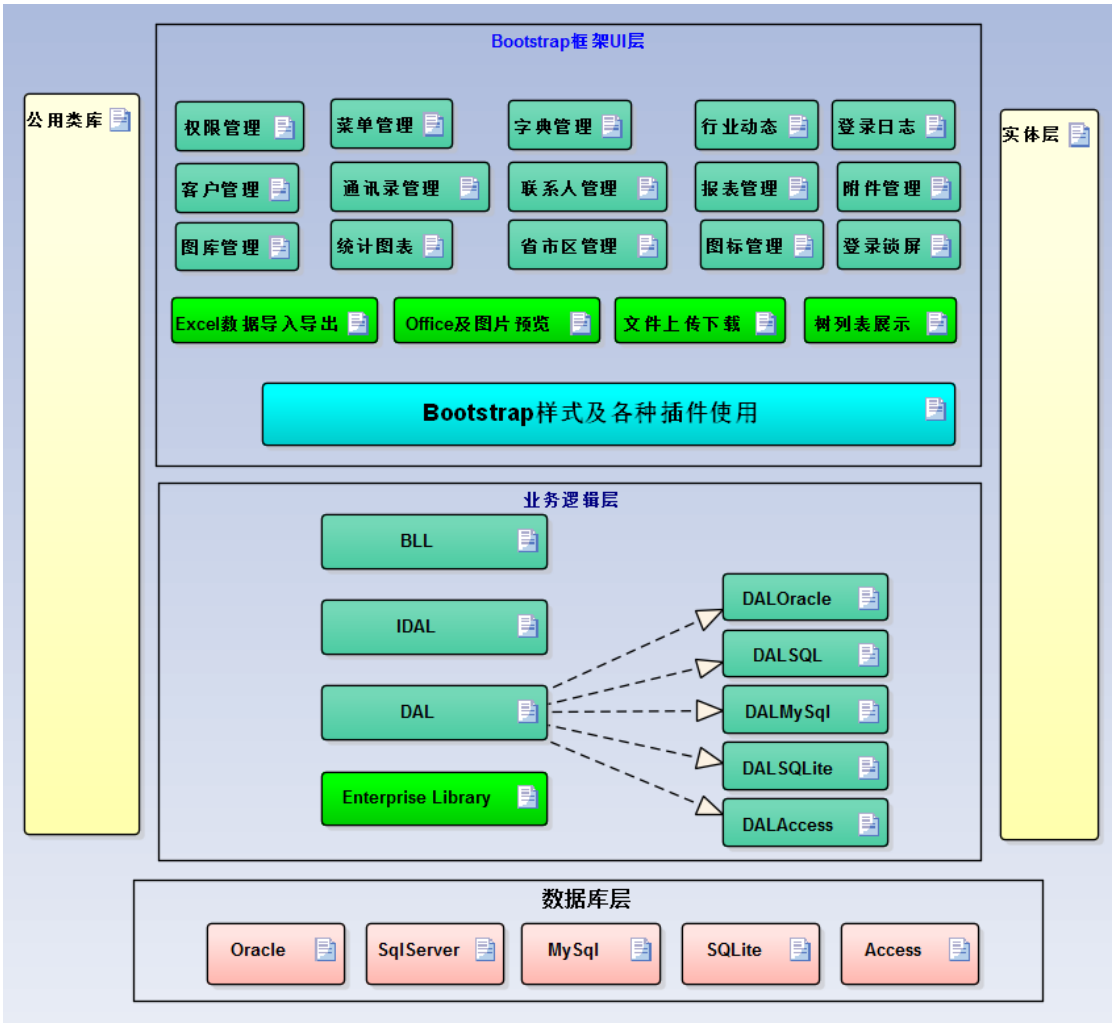
Bootstrap 是一个前端的技术框架，很多平台都可以采用，JAVA/PHP/.NET 都可以用来做前端界面，整合 JQuery 可以实现非常丰富的界面效果，目前也有很多 Bootstrap 的插件能够提供给大家使用，但是在国内很多基于 Bootstrap 的介绍很多还是停留在教学的基础上，介绍 Bootstrap 的各种基础知识和简单的使用；本框架则是以基于 C#的 MVC 实际项目的基础上，对 Bootstrap 开发框架进行全面的案例介绍，直观为大家介绍这方面的经验和心得，同时可以直接使用在实际的开发项目中，实现快速、高效的项目开发工作。

Metronic 是一个国外的基于 HTML、JS 等技术的 Bootstrap 开发框架整合，整合了很多 Bootstrap 的前端技术和插件的使用，是一个非常不错的技术框架。本框架以这个为基础，整合了基于 MVC 的 Bootstrap 开发框架，使之能够符合实际项目的结构需要的实际项目。框架后台采用基于 C#的 MVC 技术，是目前.NET 开发最为成熟流行的技术，框架后台数据库支持 Oracle、SqlServer、MySql、Sqlite、Access 等常规数据库，可通过配置进行自由切换，使用 Enterprise

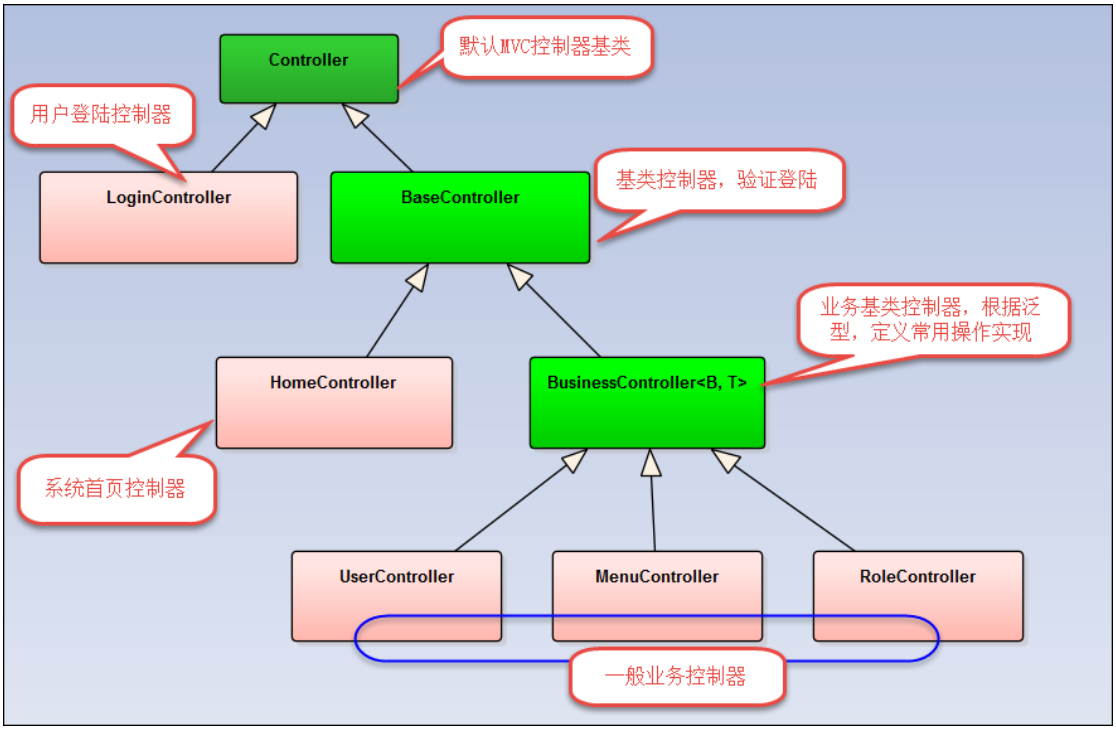
Library 模块进行数据访问的控制，使得数据访问更方便轻松。

整体框架开发采用 Visual Studuio 2013 以及页面编辑工具 Sublime Text 结合开发，页面以及后台代码，通过代码生成工具 Database2Sharp 进行快速开发，实现整体性开发的最大效率提高。

框架的总体结构如下所示：

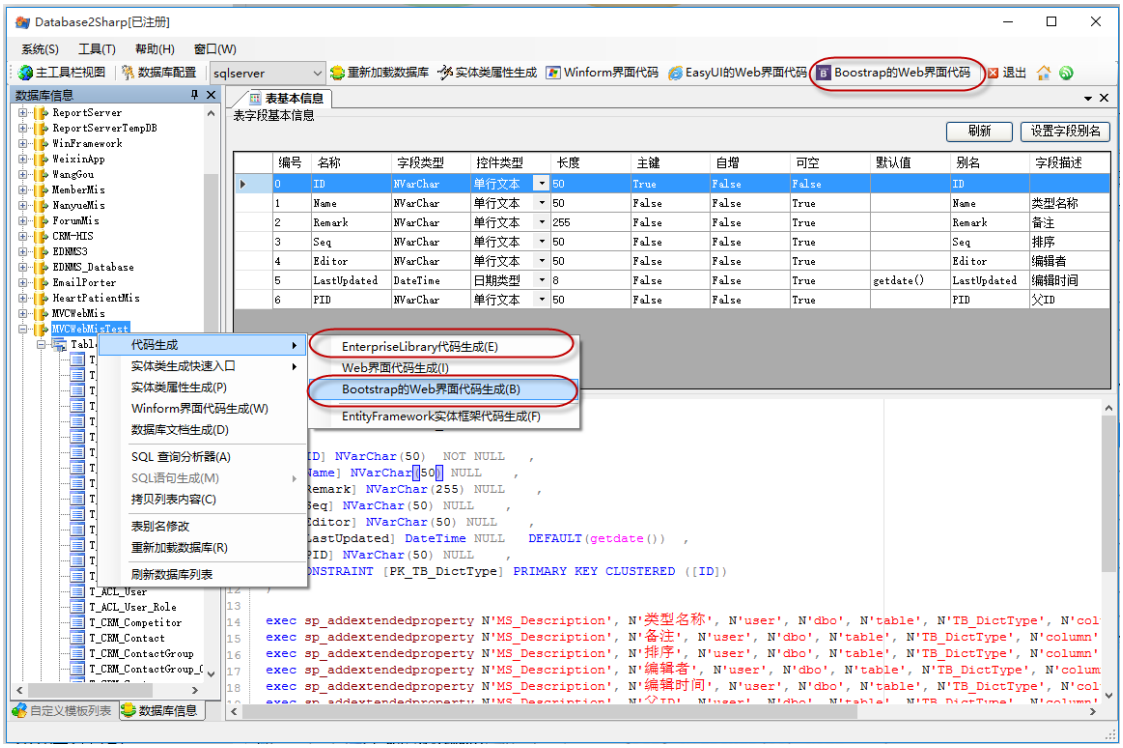


Bootstrap 开发框架沿用了我的《Winform 开发框架》和《基于 EasyUI 的 Web 框架》的很多架构设计思路和特点，对 Controller 进行了封装。使得控制器能够获得很好的继承关系，并能以更少的代码，更高效的开发效率，实现 Web 项目的开发工作。

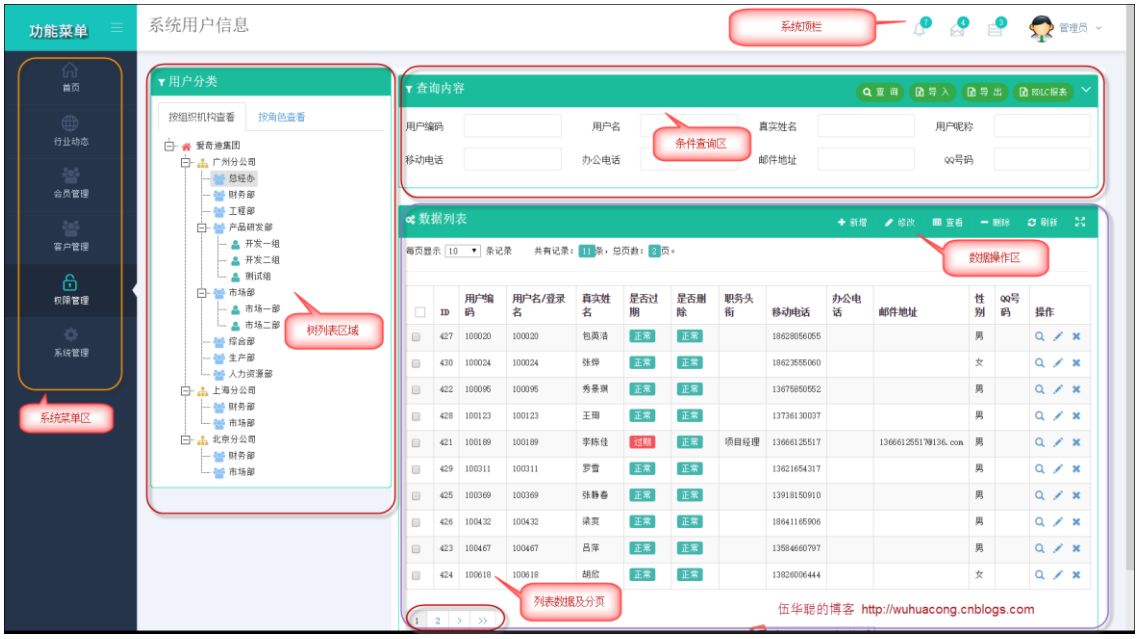


良好的架构使得无论在业务逻辑层、控制器层、Web 界面的 UI 层，均能提供统一的代码逻辑，这些代码均能通过代码生成工具 Database2Sharp 进行生成。Web 界面代码可以充分利用代码生成工具 Database2Sharp 的元数据信息，实现 Web 界面的快速生成。有效减少出错的几率，提高 Web 界面编码的开发效率和乐趣，更可以使得企业内部的编码模式进行高效的统一。

Enterprise Library 代码生成，可以快速生成除界面外的整体性的框架代码，Bootstrap 的 Web 界面代码生成，可以快速生成基于 Metronic 的 Bootstrap 的前端界面代码和后台控制器代码，界面部分包括查询、分页、数据展示、数据导入导出、新增、编辑、查看、删除等基础功能界面，生成后我们可以基于这个基础上进行简单、快速的修改即可符合实际需要，极大提高我们 Web 界面的开发效率。



该框架整体的界面效果如下所示。



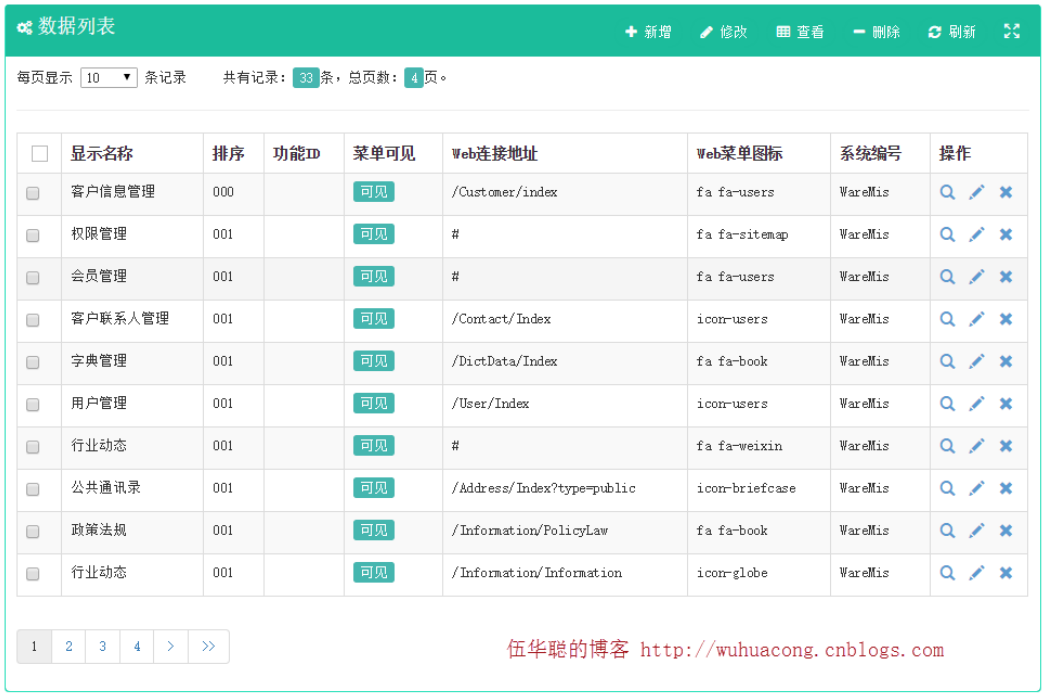
【系统菜单栏】的内容，是动态从数据库里面获取的菜单；【系统顶栏】放置一些信息展示，以及提供用户对个人数据快速处理，如查看个人信息、注销、锁屏等操作内容；内容区一般包括【树列表区】、【条件查询区】和【列表数据及分页】内容，内容区域主要是可视化展示的数据，可以通过树列表控件、表格控件进行展示，一般数据还有增删改查、以及分页的需要，因此需要整合各

种功能的处理。另外，用户的数据，除了查询展示外，还需要有导入、导出等相关操作，这些是常规性的数据处理功能。

菜单的处理和展示：一般为了管理方便，菜单分为三级，选中的菜单和别的菜单样式有所区分，菜单可以折叠最小化，效果如下所示。

整个框架涉及了很多内容，包括常规 Bootstrap 的各种 CSS 特性的使用，以及菜单栏、Bootstrap 图标管理、系统顶栏、树形控件 JSTree、Portlet 容器、Modal 对话框、Tab 控件、Bootstrap-table 控件、下拉列表 Select2、复选框 iCheck、多文本编辑控件 summernote、文件及图片上传展示 fileinput、提示控件 bootstrap-toastr 和 sweetalert、数值调整控件 touchspin、视频播放展示控件 video-player 等等，这些特性在整体性的解决方案里面都有涉及，集合这些优秀的插件，能够为我们的框架提供更强大的功能和丰富的界面体验。

框架的数据列表界面如下所示：



框架的编辑界面效果如下所示。

Bootstrap 框架的底层架构后端代码和前端界面 HTML 代码，都可以根据数据库快速生成，后端代码整合了基础框架的增删改查等常规接口，前端代码通过 Ajax 的 JS 脚本进行访问，在界面实现快速的数据处理。

利用代码生成工具 Database2Sharp 可以快速生成前端后端代码，特别界面的 JS 代码已经一一对应，我们主要的开发工作就是对界面的布局进行调整即可，极大提高开发效率，减少出错机会。

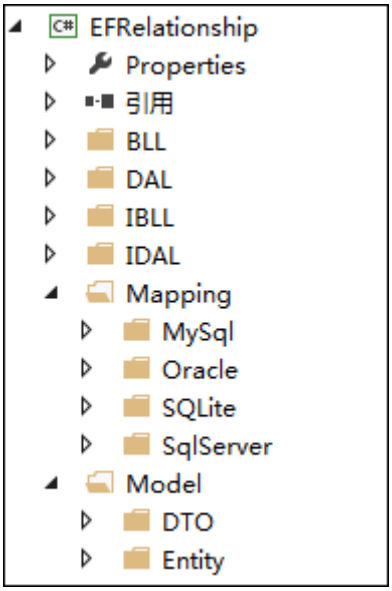
2.5 Entity Framework 实体框架

一个基于泛型的仓储模式的实体框架，全部利用 LINQ 高效的语法进行底层操作，包含数据传输模型 DTO 和实体模型 Entity 层、实体关系映射层、数据访问类、业务逻辑类、WCF 相关服务层等。

该架构利用泛型的仓储设计，传输模型 DTO 和实体模型 Entity 的分离与联合，实体对象的动态映射关系、Unity 依赖注入等特性，融合了微软最新开发技术以及业界的组件应用，能快速、高效、统一实现更好的业务开发。

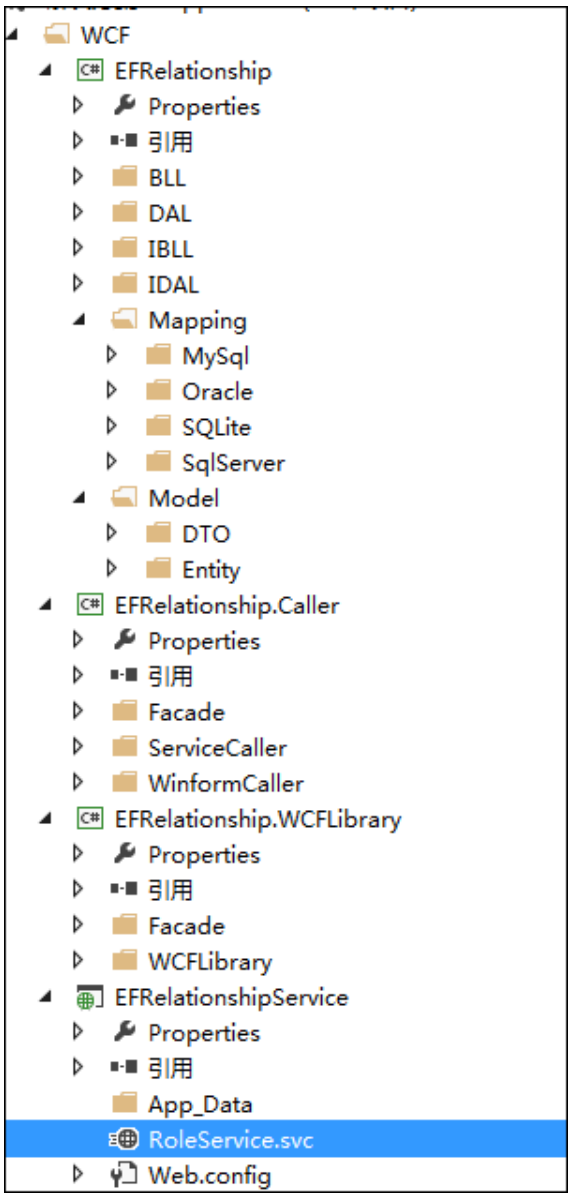
2.5.1 Entity Framework 实体框架结构

基于方便管理的目的，每个模块都可以采用一种固定分层的方式来组织模块的业务内容，每个模块都是以麻雀虽小、五脏俱全的方针实施。实例模块的整个业务逻辑层的项目结构如下所示。



如果考虑使用 WCF，那么整体的结构和我之前的混合框架差不多，各个模

块的职责基本没什么变化，不过由原先在 DAL 层分开的各个实现层，变化为各个数据库的 Mapping 层了，而模型增加了 DTO，具体项目结构如下所示。



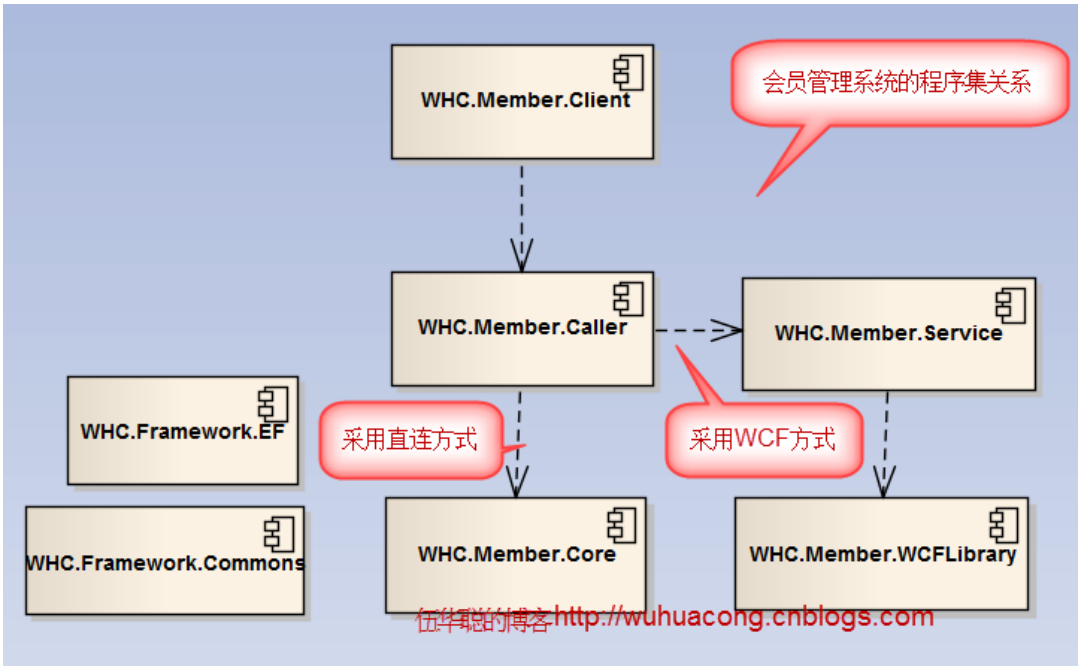
具体的项目说明如下所示：

EFRelationship	系统的业务模块及接口、数据库访问模块及接口、DTO 对象、实体类对象、各种数据库映射 Mapping 类等相关内容。该模块内容紧密结合 Database2Sharp 强大代码生成工具生成的代码、各层高度抽象继承及使用泛型支持多数据库。
EFRelationship.WCFLibrary	系统的 WCF 服务的业务逻辑模块，该模块通过引用文件方式，把业务管理逻辑放在一起，方便 WCF 服务部署及

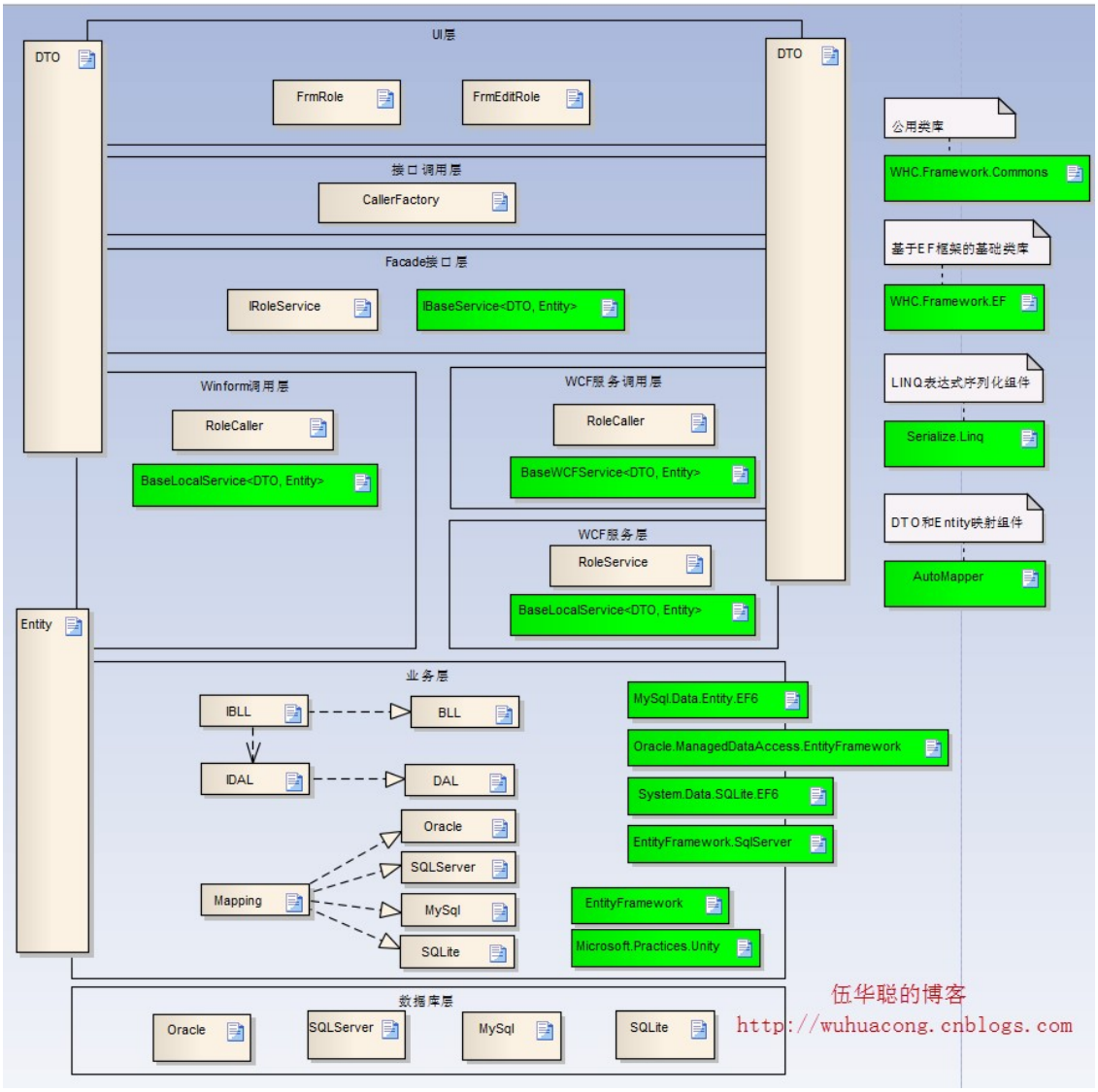
	调用。
EFRelationshipService	框架 WCF 服务模块，包括基础服务模块 BaseWcf 和业务服务模块，他们为了方便，分开管理发布。
EFRelationship.Caller	定义了具体业务模块实现的 Façade 应用接口层，并对 Winform 调用方式和 WCF 调用方式进行包装的项目。

2.5.2 实体框架架构设计图

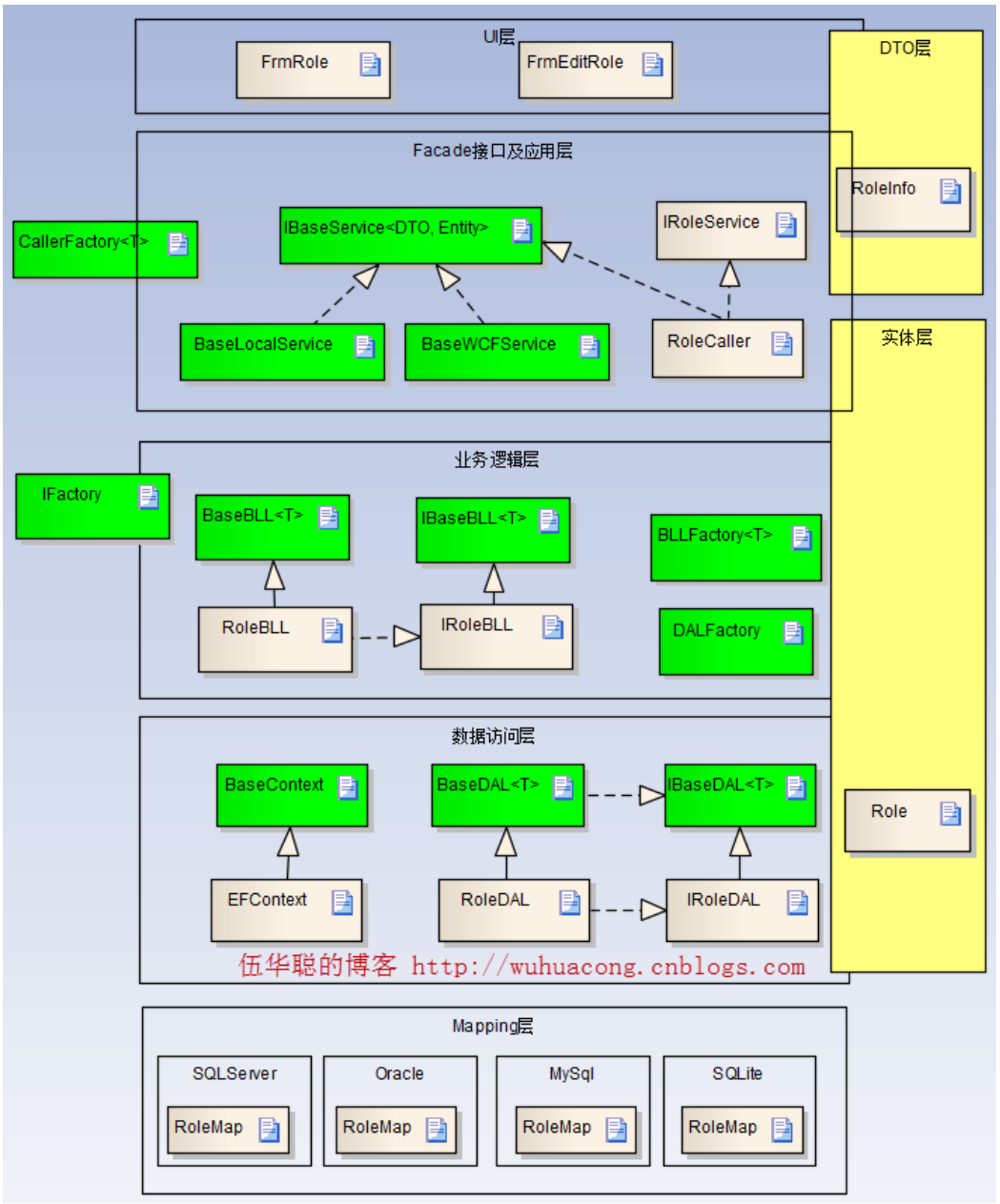
具体我们以一个会员系统设计为例，它的程序集关系如下所示。



我们来看看整个架构的设计效果如下所示。



其中业务逻辑层模块（以及其它应用层）我们提供了很多基于实体框架的公用类库（WHC.Framework.EF)，其中的继承关系我们将它放大，了解其中的继承细节关系，效果如下所示。



上图很好的概述了我们的 EF 实体框架的设计思路，这些层最终还是通过代码生成工具 Database2Sharp 进行一体化的生成，以提高快速生产的目的，并且统一所有的命名规则。

3 适用范围

Database2Sharp 主要定位于 C#应用开发，因此主要针对从实际项目框架提炼而来模板来进行项目开发，力求达到省事省力的目的。下面列出该工具针对的应用场景。

1、**Enterprise Library 代码生成**。生成整个项目工程框架，包含实体类、数据访问类、业务类、Web 页面代码、WCF 相关服务层和客户端调用包装层、Web API 相关服务层和客户端调用包装层等。该架构利用泛型及缓存机制，良好的架构极大简化代码，强大完善的基类使你甚至不用编写一行代码。该模式可以生成基于 Web 框架项目、传统 Winform 框架项目、WCF 框架项目，以及混合型框架项目，它们是非常成熟，应用很广的几种项目框架。

2、**Winform 界面代码自动生成**。可以生成 Winform 界面布局代码，后台逻辑代码，生成即可使用，极大提高开发 Winform 界面的效率，较少枯燥的开发过程。该界面部分包括传统的直接访问数据库的 Winform 界面，以及基于混合式框架的 Winform 界面，生成的界面内置了对业务数据的增删改查、导入导出、分页等基础功能。

3、**基于 EasyUI 的 Web 界面代码生成**。结合我们的《基于 MVC+EasyUI 的 Web 开发框架》，快速生成的界面代码，包括生成列表分页、增加、修改、查看、删除、附件管理、Office 在线查看等界面代码，生成代码可直接在 Web 框架中进行整合运行，界面统一美观。EasyUI 组件提供统一的控件使用以及统一的界面展示，以及很多内置的功能，适合在开发内部业务的管理系统。

4、**基于 Bootstrap 的 Web 界面代码生成**。从 EasyUI 的 Web 框架进行演化，结合我们的《基于 MVC+Bootstrap 的 Web 开发框架》，快速生成的基于 Bootstrap 加 Metronic 界面样式的 Web 界面代码，包括生成列表分页、增加、修改、查看、删除、附件管理、Office 在线查看、客户关系管理等界面代码，以及整合众多开源 Bootstrap 插件模块，生成代码可直接在 Web 框架中进行整合运行，界面美观，Bootstrap 前端应用框架是目前 Web 开发非常广泛使用的框架，结合很多开源插件可以实现丰富的业务界面。

5、**Entity Framework 微软实体框架代码生成**，生成整个基于泛型的仓储模式的实体框架，全部利用 LINQ 高效的语法进行底层操作，包含数据传输模型 DTO 和实体模型 Entity 层、实体关系映射层、数据访问类、业务逻辑类、WCF 相关服务层等。该架构利用泛型的仓储设计，传输模型 DTO 和实体模型 Entity 的分离与联合，实体对象的动态映射关系等特性，良好的架构极大简化代码，强大完善的基类使你甚至不用编写一行代码。完美支持 Winform 开发框

架、混合式 Winfrom 开发框架、基于 MVC4+EasyUI 的 Web 开发框架、基于 MVC+Bootstrap 的 Web 开发框架的整合开发工作。

6、**常规实体类、WCF 实体类、自定义实体类代码快速生成**。提供各种常用的实体类代码生成，直接在窗体中显示，并用语法高亮显示，方便拷贝使用。

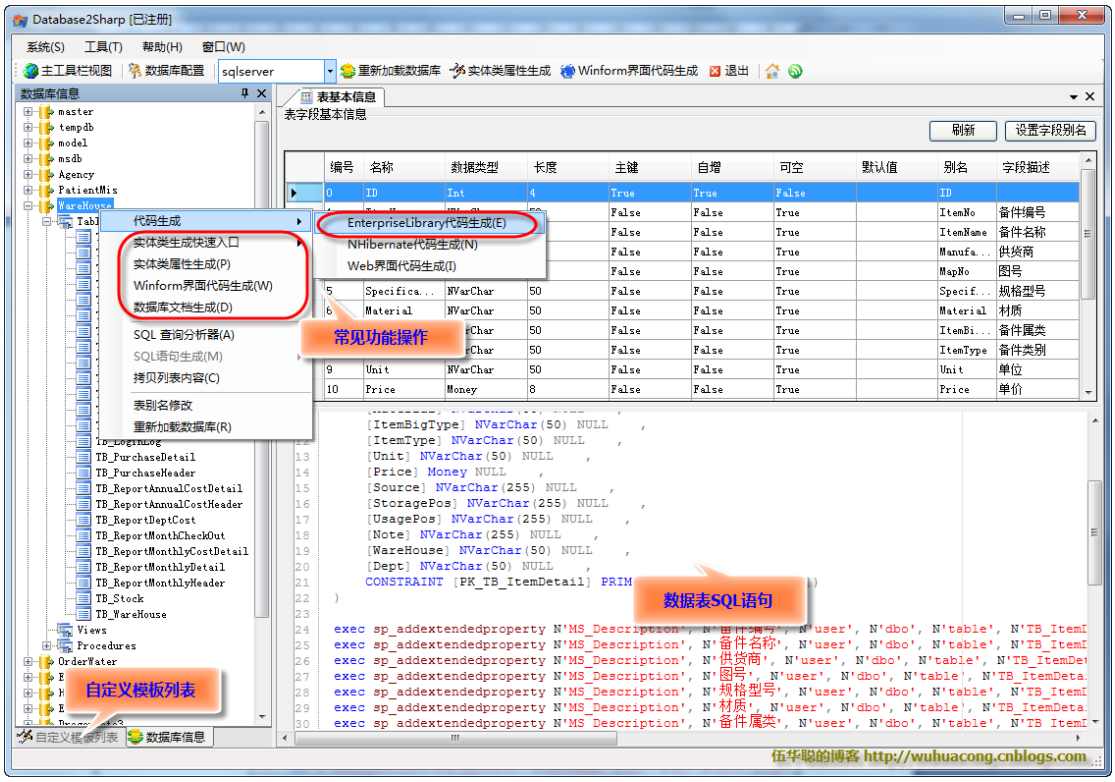
7、**可视化查看数据库信息**。可以查看数据库的信息和实现查询分析器的功能，可以很快地看看数据库的信息。

8、**数据库文档的生成**。支持从数据库直接生成相关的数据库文档，在模块设计中非常有用，谁想去写描述和字段名称，类型的对应关系呢？

9、**数据库语句生成**。增加 Select、Update、Insert 和 Delete 基本语句代码的生成，方便您直接在代码中使用。

10、**自定义模板代码生成**。Database2Sharp 采用了方便灵活、功能强大的 NVelocity 模板引擎作为代码生成的一部分，你可以程序目录中修改相应的代码模板，实现部分自定义的代码生成。工具自带有一些典型语法例子案例供参考学习，也提供了数据库相关的架构信息说明。

4 EnterpriseLibrary 架构代码生成



4.1 代码生成总体概述

EnterpriseLibrary 代码生成时一个整体性项目代码的生成操作，它能根据设计好数据库信息以及模板文件，生成一个完整性非常高的项目。一般结合我们的 **Winform 开发框架、WCF 开发框架、混合型开发框架**或者 Web 开发框架，进行增量式的项目开发，效率更高，而且可以可以利用更多已经开发好的、现成的组件模块的集成，完美的整合，以及模块化的封装，能带给你无穷的开发乐趣同时，使得项目无论从代码风格、用户界面、设计理念，都能保持很好的统一，快速优雅的完成碰到的项目。

使用 Database2Sharp 来生成框架代码，虽然直接生成的代码，就是一个整体方案的代码，基本上可以直接运行。而 Winform 开发框架和代码生成工具生成的项目组织上有所不同。下面提供几个注意的地方。

1）代码生成工具生成的代码是基于 Project 的，而 Winform 开发框架为了项目数量，方便管理，是把业务层、数据访问层、数据接口层、实体层放到一个工程项目中了（**WCF 项目会把实体层独立作为一个项目处理**），因此生成的代码我们复制到对应的目录位置就可以了，默认命名空间不需要改动。

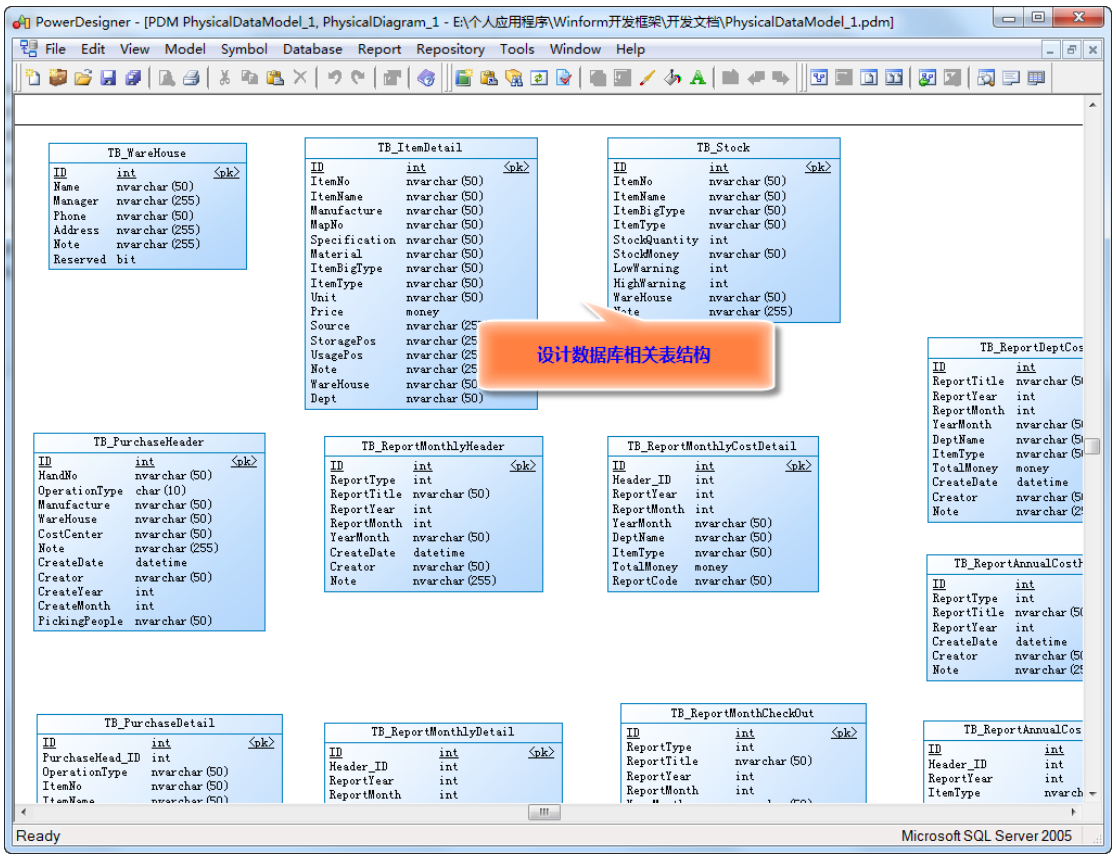
2）为了代码生成方便，代码生成工具需要把数据库字段的中文说明作为代码注释或者说明的一部分，因此，设计数据库（SqlServer、Oracle 等）的时候，我们强烈要求把注释添加到字段说明里面去。

3）数据库表一般需要提供一个主键关键字（建议取名为 ID），主键字段可以为自增长的整形类型，也可以是任意字符型。建议 SqlServer 一般采用自增长整形、Oracle 采用 Number 类型，并为每个表指定一个部分同名的序列名称，如 Seq_ABC，其中 ABC 代表对应的表名。

4.2 数据库表设计

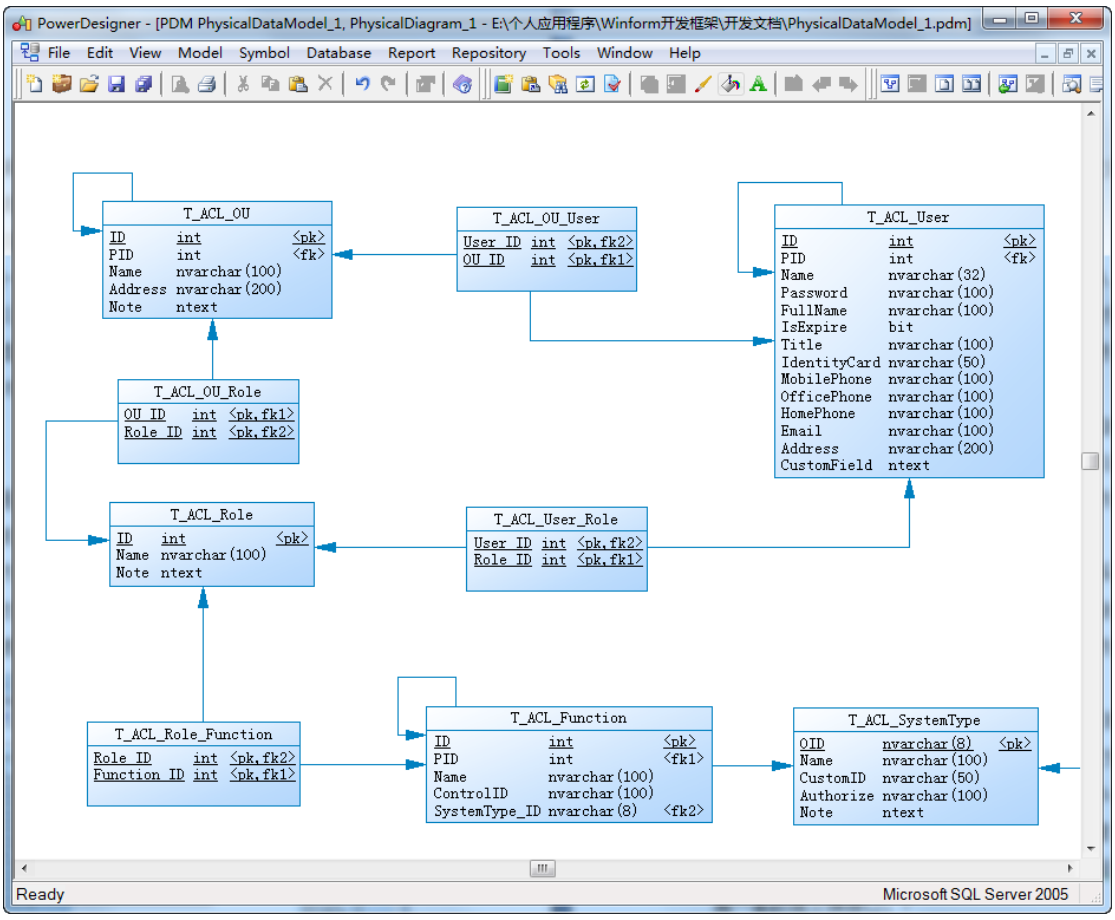
由于我们一般的开发过程是，先设计好数据库，然后生成项目代码框架，再进行调整完善。因此在开发前，我们非常关键的第一步就是要设计好数据库。

数据库设计可以采用多种方式进行，但为了提高设计效率以及方便修改等操作，一般我们最好基于数据库建模软件进行数据库的设计过程，如 PowerDeigner 就是一个很好的数据库设计平台。如下图所示：



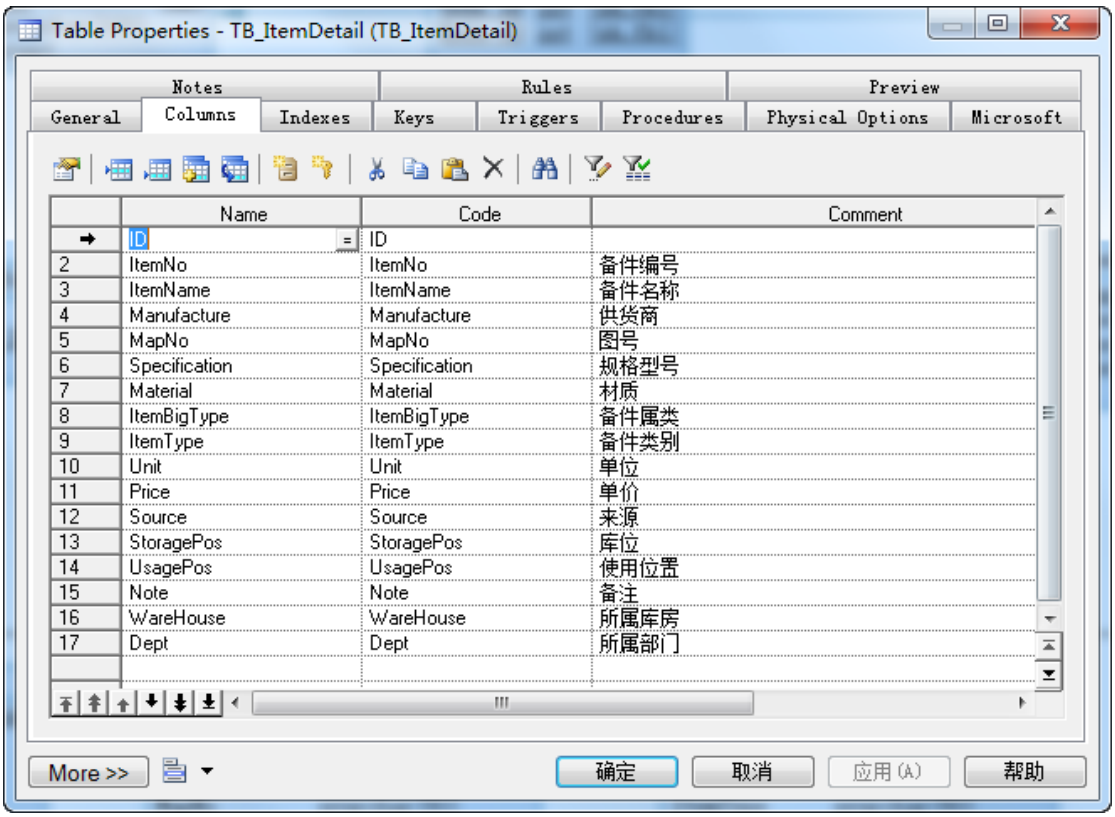
图表 4-1 数据库设计

当然数据库设计的时候，也可以指定它们之间的逻辑引用关系，这样对于数据的完整性校验比较有保证，如下图所示。



图表 4-2 数据库设计 2

无论是上面那种设计关系，我们都需要在设计过程中，注意到表**字段备注信息**，由于在代码生成工具生成代码的时候，很多时候需要使用中文的字段名称来描述，如实体类字段的备注信息、界面的查询字段说明、列表的表头提示等等，这些都是从你设计的数据库表字段备注里面来，因此要特别注意。在 PowerDesigner 设计软件里面，只需要指定 Comment，然后生成 SQL 的时候，就会有相关的备注信息了。

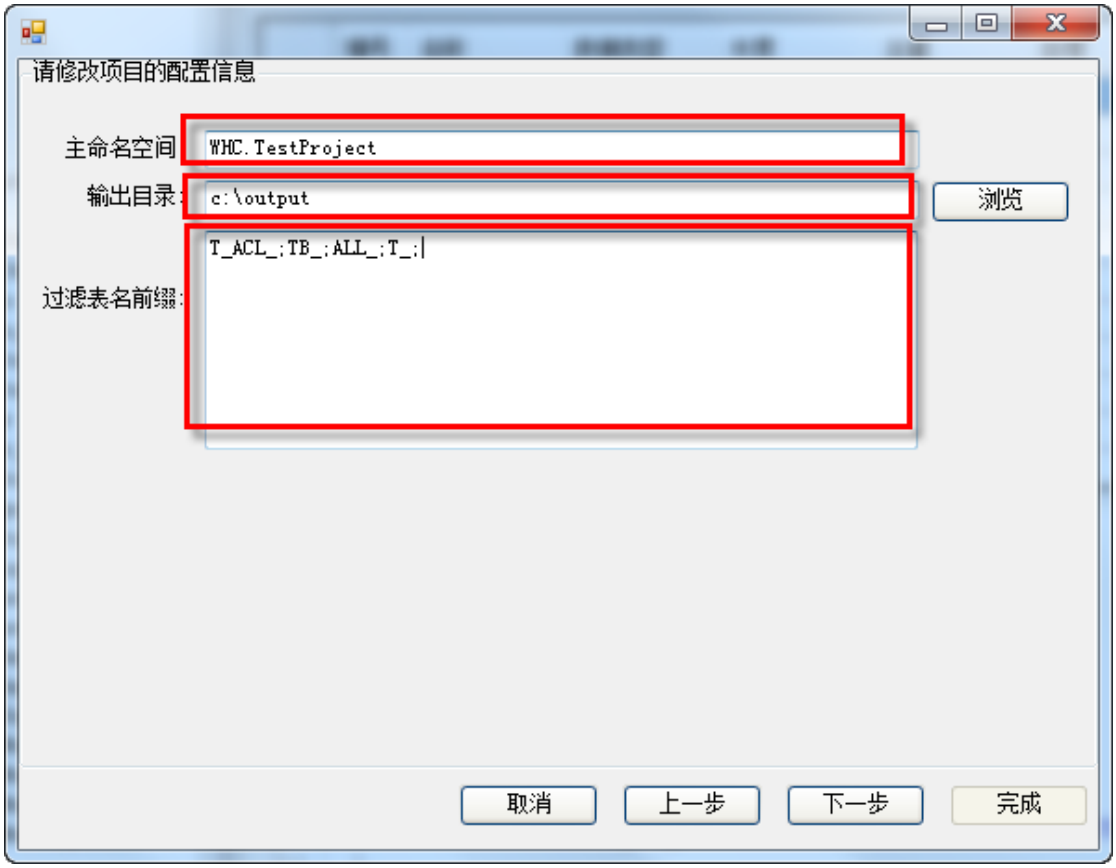


图表 4-3 数据库设计 3

4.3 代码生成参数配置

基于前面设计好了数据库，生成相关的 SQL，然后在数据库管理系统（Oracle/SqlServer/MySql 等）上执行脚本，创建数据库成功后。就可以利用代码生成工具进行代码生成了，但生成代码前，需要配置几项参数，以求能够更完美生成项目工程，实现快速使用。

（一）基础参数



图表 4-4 基础参数设置

在上图我们可以看到，代码生成工具只需要很少的几个属性，就能较好的生成所需的代码。

主命名空间，就是我们的项目代码的命名空间的前面基本不变的部分。如实体类的命名空间一般为 WHC.TestProject.Entity，那么主命名一般为 WHC.TestProject 即可，相应界面层生成后的命名空间为 WHC.TestProject.UI，业务逻辑层的命名空间为 WHC.TestProject.BLL，数据访问层根据不同的数据库生成不同的命名空间，如 Oracle 数据访问层命名空间为 WHC.TestProject.DALOracle，SqlServer 数据访问层命名空间为 WHC.TestProject.DALSQL，而 Access 数据访问层命名空间为 WHC.TestProject.DALAccess 等如此类推。

输出目录，顾名思义就是我们代码最终的生成目录了。

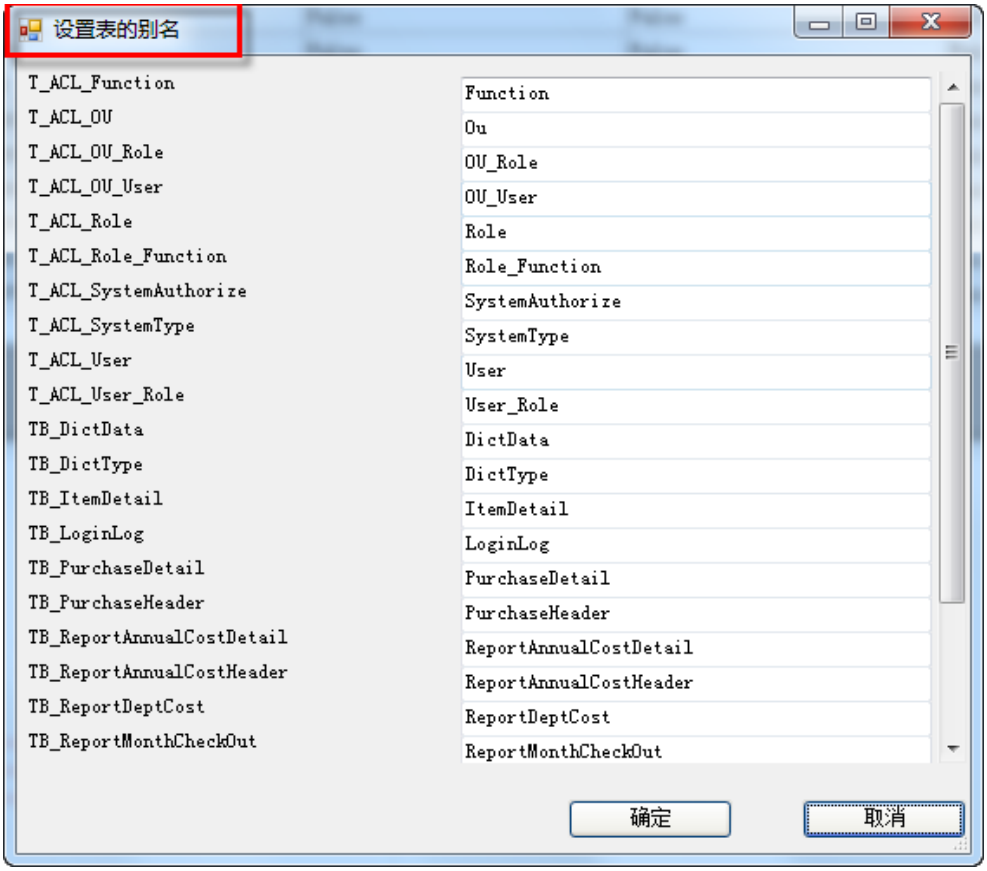
过滤表名前缀，是减少代码出现多余的表名前缀。一般在大一些项目上，或者仅仅保持良好的命名习惯上，我们都应该给不同应用范畴的表，通过前缀来进行区分，如我对于基础类可能用“TB_”前缀来区分，对于权限范畴的可

能用“T_ACL_”来区分，其他的可能用“T_”来区分等等。这样我们在生成代码的时候，就应该去掉这些多余的前缀，使得我们的业务类名称更容易阅读。如字典大类表名称为“TB_DictType”，那么对应的业务类生成应该就是“DictType”了。

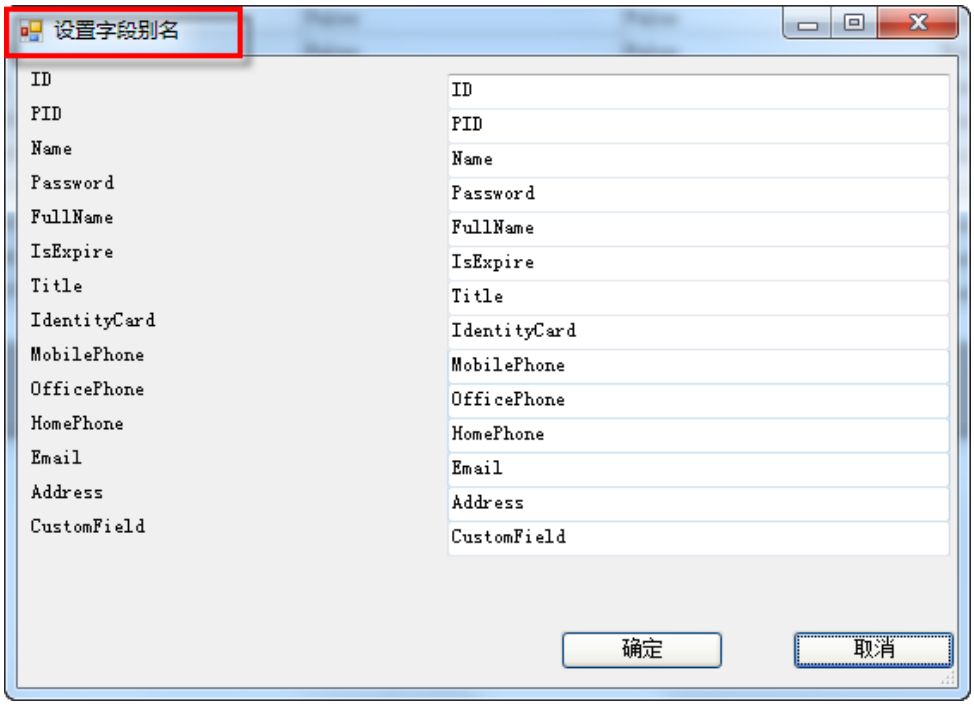
（二）别名修改

除了以上的几项设置外，有时候，为了较好调整表名或者字段属性的名称，也增加了一项修改别名的功能，如“ABC”的表名我们不知道是什么意思，如果把它的别名修改为“Company”这样的单词，一般人可能就知道这个类代表的意义了，别名修改就是为了这样需求而出现的。一般情况下，我们尽可能在表设计的时候取有意义的名称，这样就可以通过表名前缀的方式自动调整了。

别名修改，有表名的别名，和表字段的别名修改两种，如下所示



图表 4-5 表别名修改



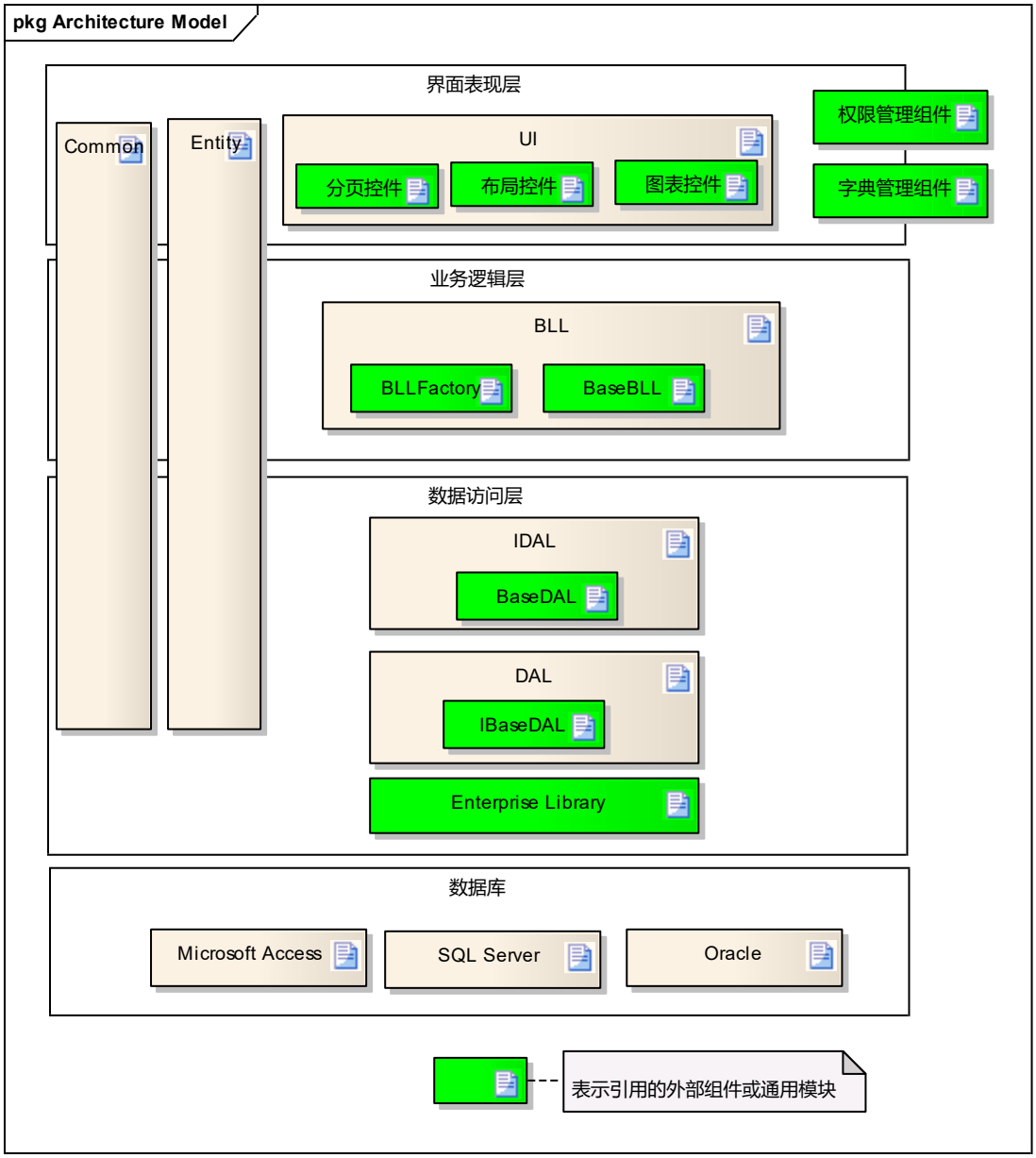
图表 4-6 表字段别名修改

4.4 代码生成

4.4.1 框架介绍

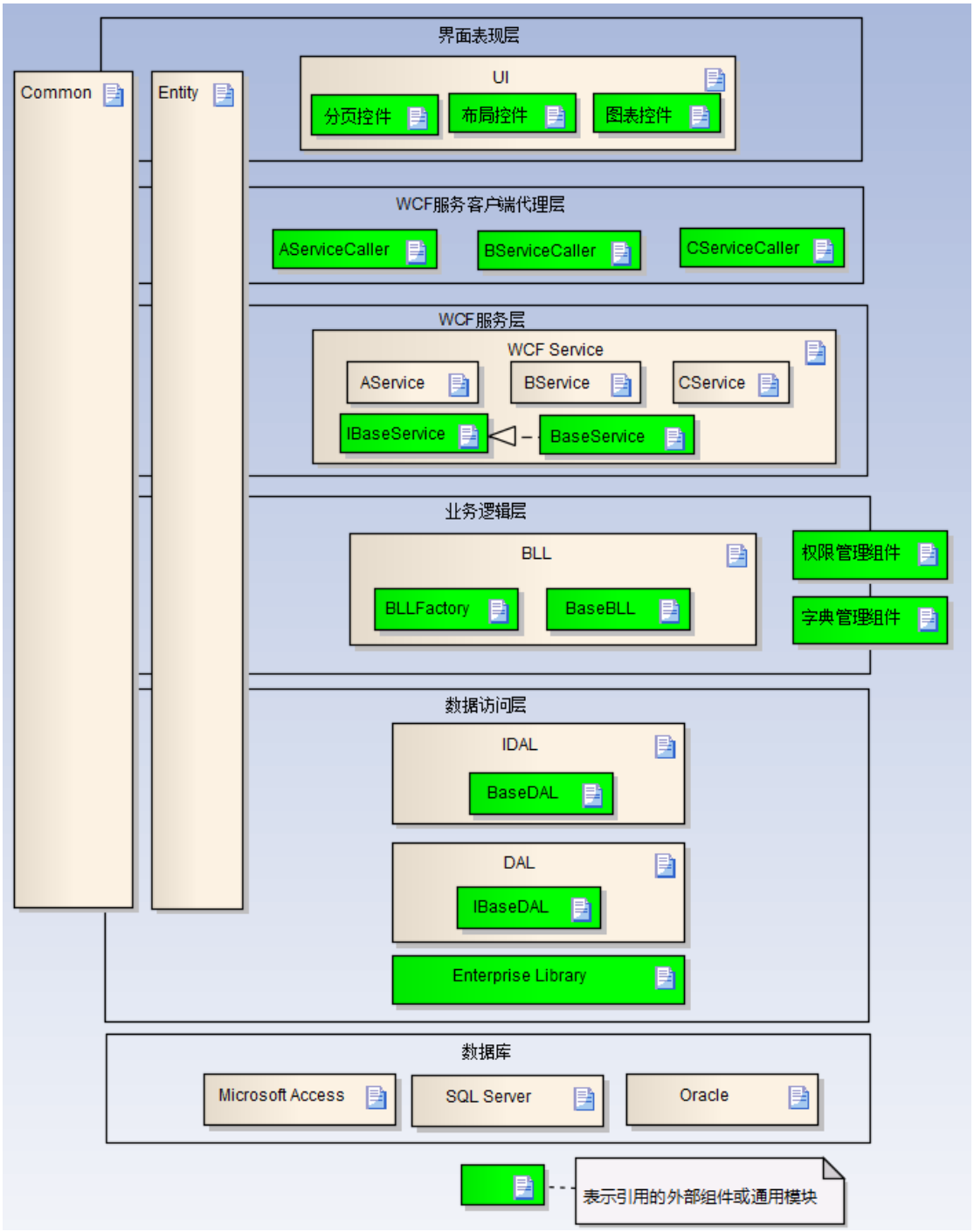
无论是在项目开始阶段的全新代码生成或者在框架搭建起来后（如基于 Winform 开发框架、WCF 开发框架、混合型开发框架），虽然起点有所不同，但是都是一个框架模式搭建完善相关的业务模块的。

无论是基于 Winform、MVC Web 的开发应用，甚至是基于分布式程序开发的 WCF 开发模式，利用 Database2Sharp 都能使你感到事半功倍的成就感，由于这几种开发框架都是在基于 EnterpriseLibrary 的框架代码生成基础上完成的，这个框架模式是通用于上述几种不同的应用开发框架，它们大致的框架布局如下所示。



图表 4-7 Winform 开发框架/Web 开发框架设计图

Winform 开发框架以及 Web 开发框架，它们就是在 BLL 业务逻辑层之上搭建一层界面展示层而已，而 WCF 开发框架则还需要在 BLL 业务逻辑层之上搭建一层 WCF 服务层，然后在界面层和 WCF 服务层之间，通过服务应用的方式，增加一层 WCF 服务层的代理层，如下所示。



图表 4-8 WCF 开发框架设计图

Winform 开发框架适用于开发用户体验好、功能强大的业务管理系统，可以基于单机版数据库（如 Access、Sqlite 等数据库）或者基于局域网的数据库，如 Oracle、SqlServer、MySql、Mongodb、DB2 等网络型的数据库应用，也就是我们传统所说的 C/S 架构模式。

Web 开发，一般也可以开发功能强大的业务管理系统，不过较 Winform 来讲，它的用户体验会差一些，而且需要部署在 IIS 上，部署会比较麻烦一些，

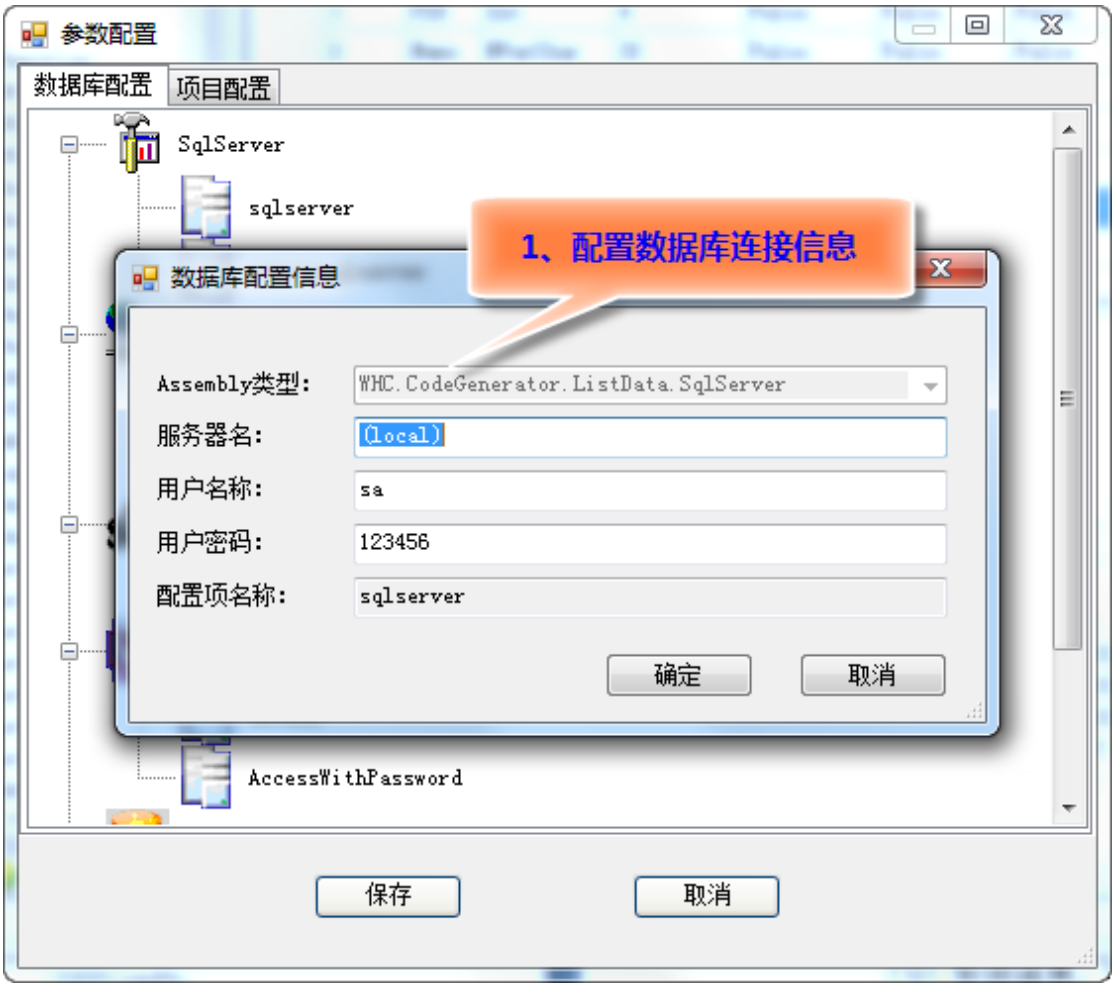
但特点是能够部署分布式的应用，而且不用安装软件，直接在 IE、Chrome 浏览器上打开 URL 连接即可使用，是传统所说的 B/S 应用模式。

而 WCF 开发框架，是利用了 C/S 应用模式里面的程序良好的界面体验性、响应快速性等特点，也充分利用了 B/S 应用模式里面的分布式架构特点，是一种融合前两种框架特点应用模式，对于要求分布式，有要较好的用户体验性，这种框架是很好的选择。

4.4.2 数据库连接配置

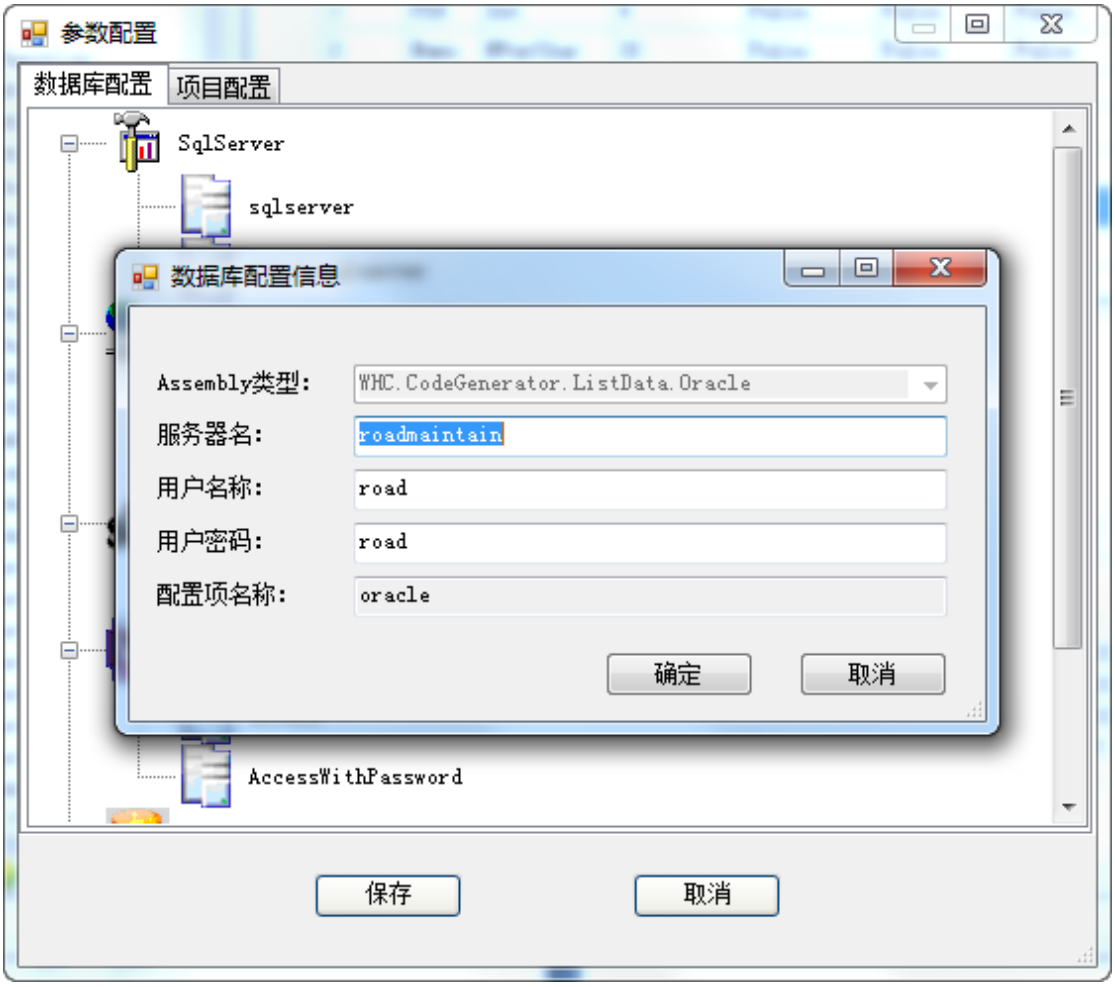
了解了不同的框架用途，我们就可以利用代码生成工具来搭建我们所需的业务管理系统了。

第一步，配置对应数据库连接信息。数据库 SqlServer 的配置信息如下所示。



图表 4-9 SqlServer 数据库连接配置

Oracle 数据库配置信息如下所示：

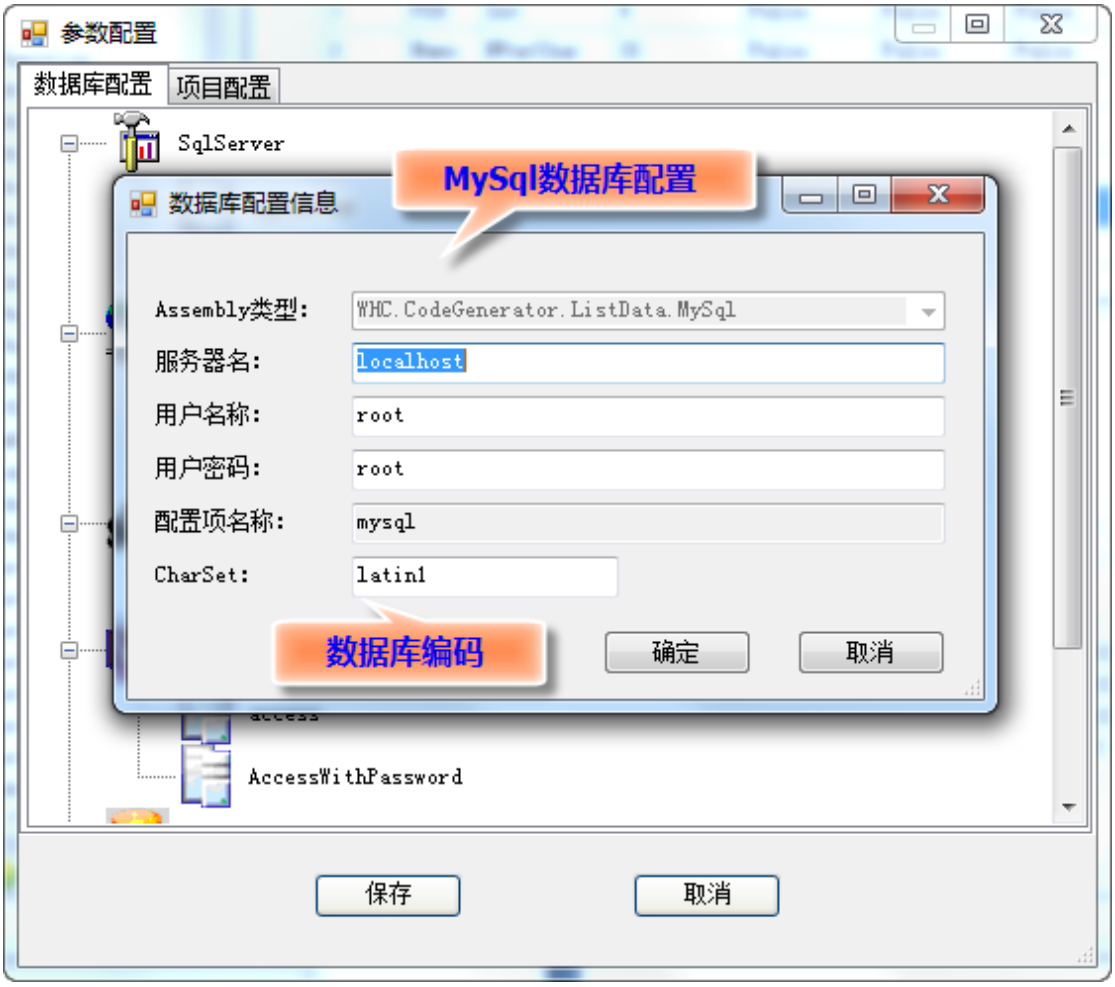


图表 4-10 Oracle 数据库连接配置

Oracle 数据库除了配置以上信息外，还要注意的是，需要利用 NetManager 来配置好对应 Oracle 数据库名称的侦听配置，如下所示。也就是 Oracle 需要安装好对应版本的 Oracle 客户端，配置好数据库侦听信息才能使用代码生成工具 Database2Sharp 访问 Oracle 数据库表信息，这样是一般常规 Oracle 工具需要做的操作，包括 PLSQL Developer。

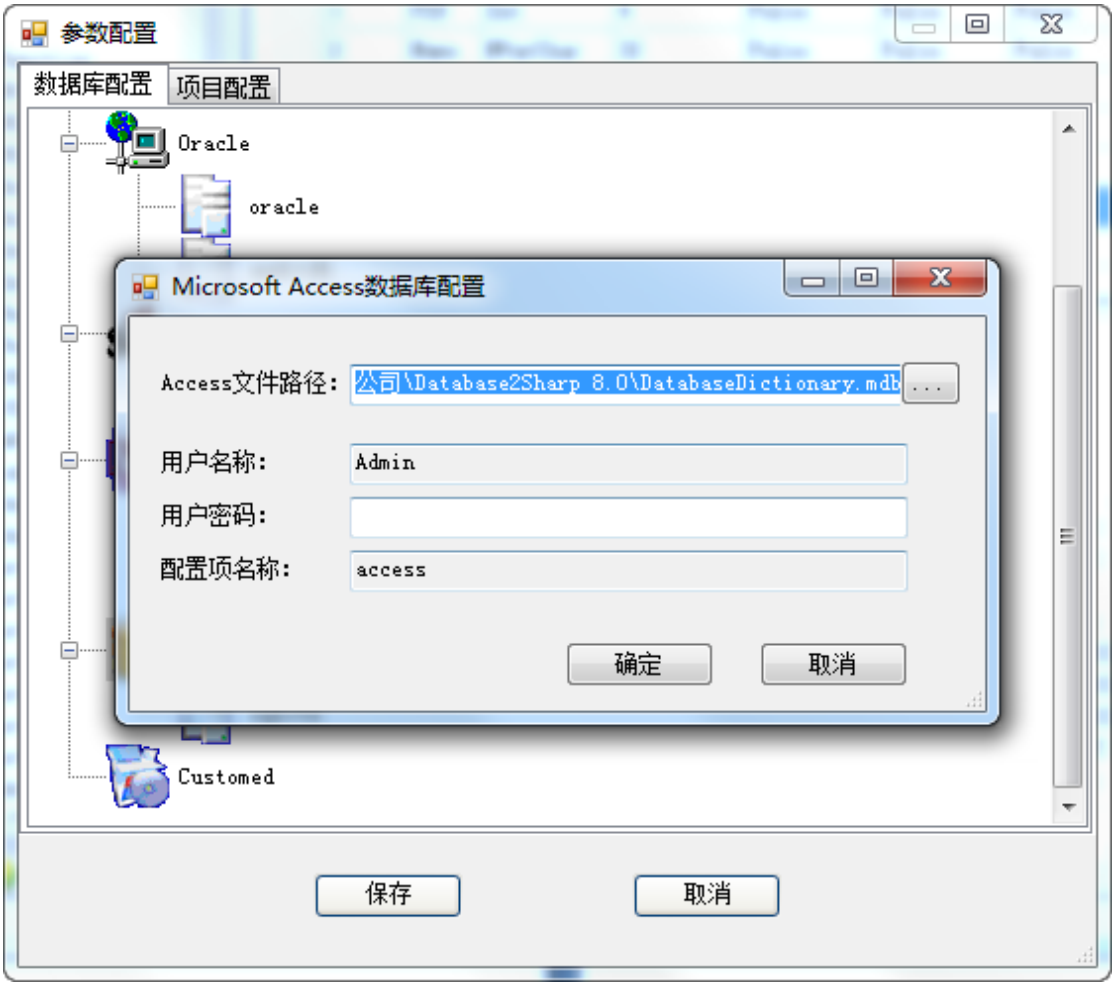
图表 4-11 Oracle 数据库连接配置 2

MySql 数据库配置信息如下所示，注意的是，MySql 数据库一般有一些编码的设置，为了有效获取对应数据库表、字段的备注等中文信息，需要设置正确的字符编码才能获取到：



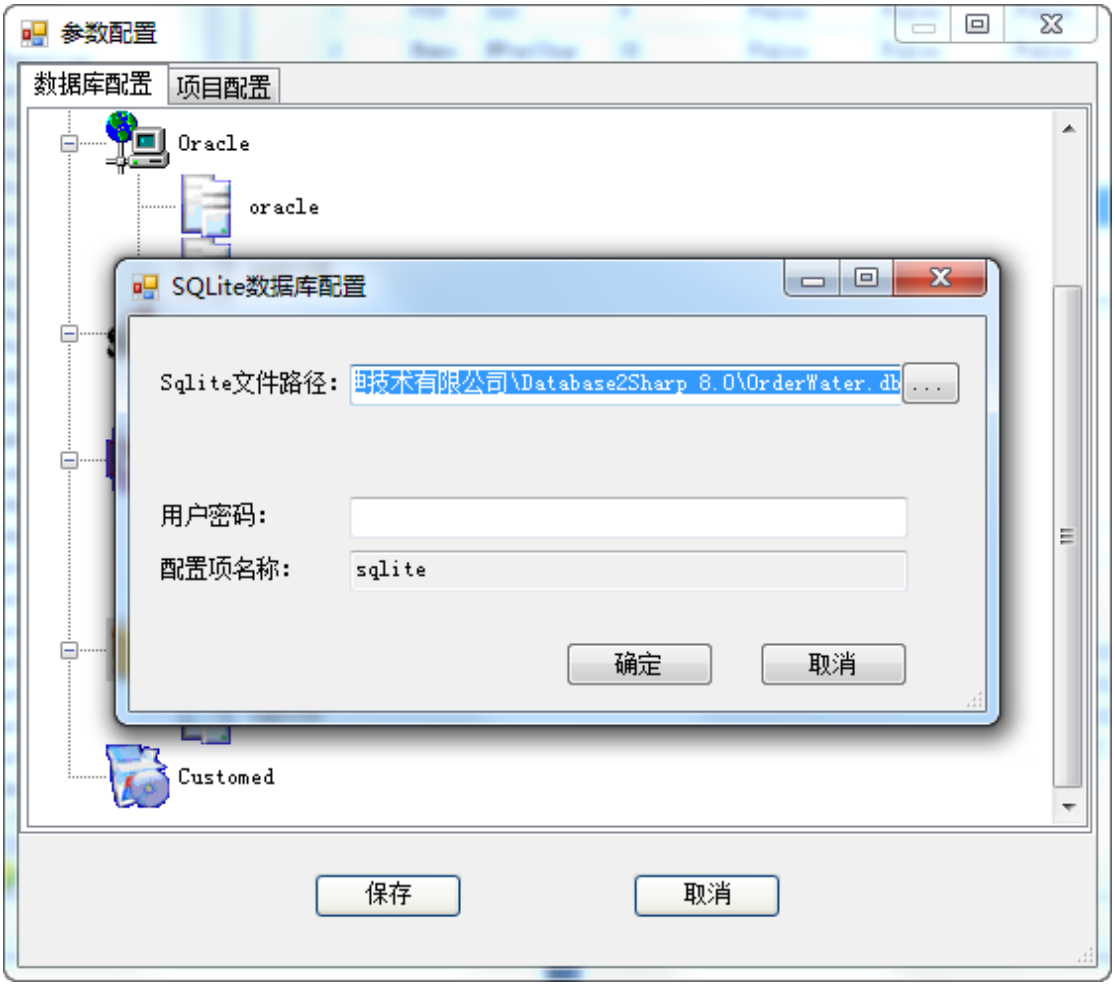
图表 4-12 MySql 数据库配置

Access 数据库配置信息如下所示，一般情况下，只需要指定 Access 数据库路径即可，如果有密码，填上数据库密码信息。



图表 4-13 Access 数据库配置

Sqlite 数据库配置信息如下所示，一般情况下，只需要指定 Sqlite 数据库路径即可。



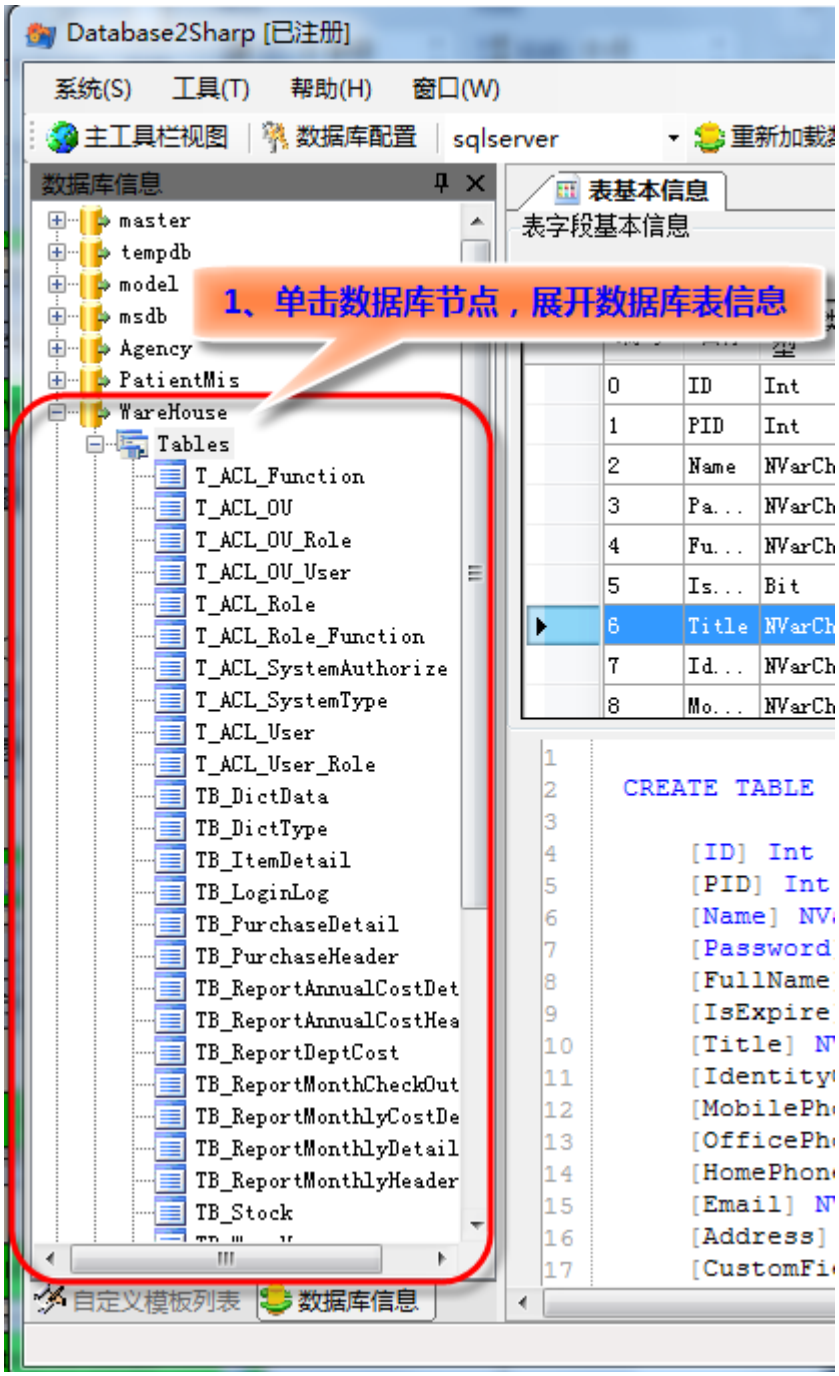
图表 4-14 Sqlite 数据库配置

4.4.3 代码生成

代码生成工具，是结合数据库信息进行代码生成的过程，因此需要先获取对应数据库信息。

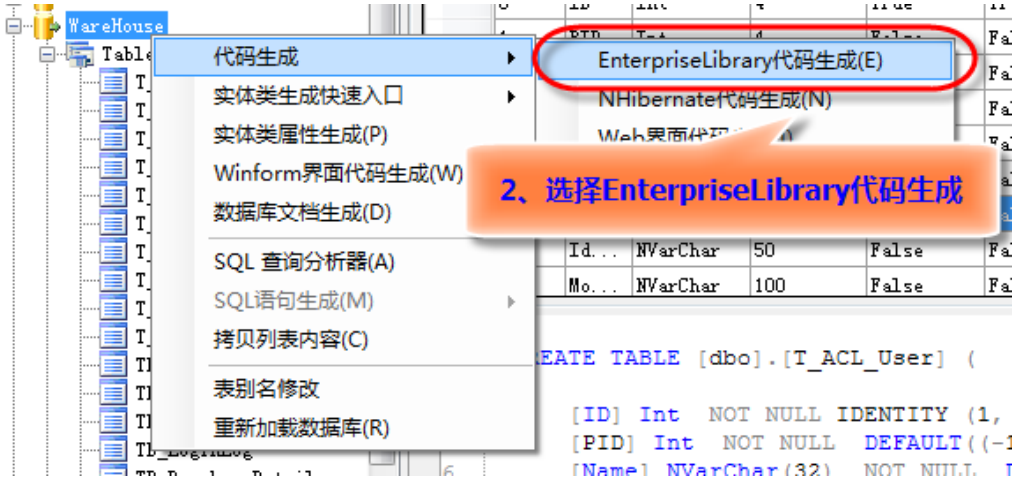
第一步，在 Database2Sharp 左边的树形数据库列表中，单击展开对应数据库的详细表，由于 Database2Sharp 是一次性加载方式，加载数据库表的信息同时，也会加载相关的视图、存储过程（Oracle、SqlServer）、以及表字段和表之间的关系等等信息。

单击其中一个表，我们可以看到对应该表的字段信息，以及创建表的 SQL 语句，也就是我们常说的 DDL 脚本。



图表 4-15 展开对应数据库表

第二步，在数据库节点上，单击鼠标右键，选择【代码生成】-【EnterpriseLibrary 代码生成】菜单，开始生成代码如下所示。



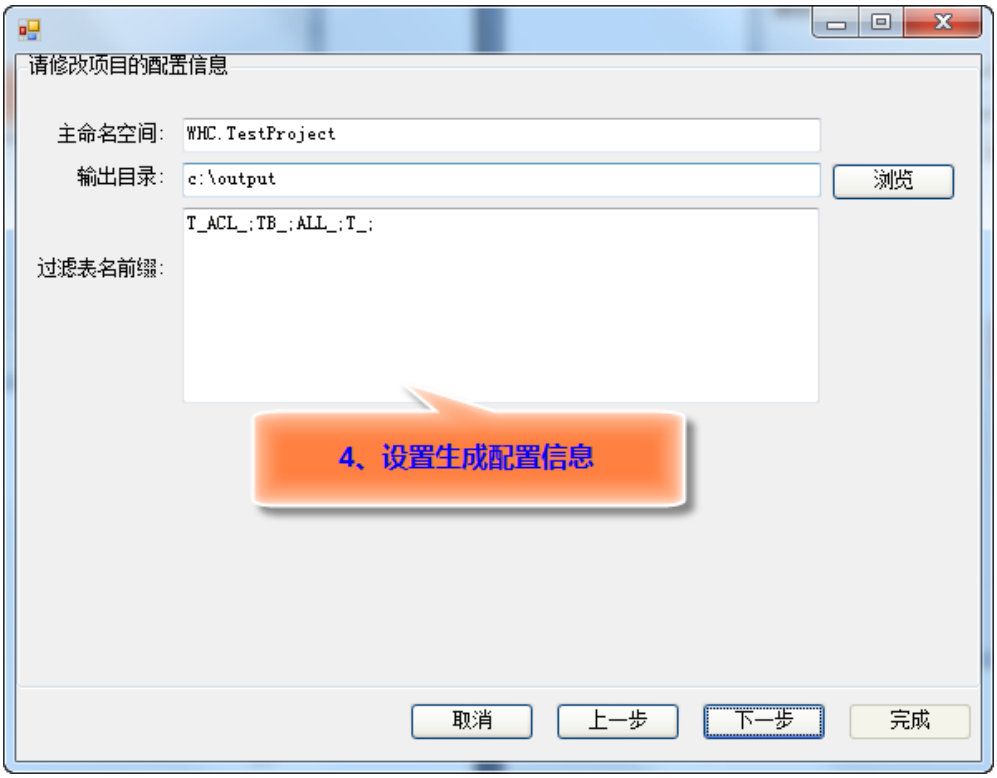
图表 4-16 开始代码生成

第三步，选择相应的数据库以及数据库表，用于代码的生成。

图表 4-17 选定数据库

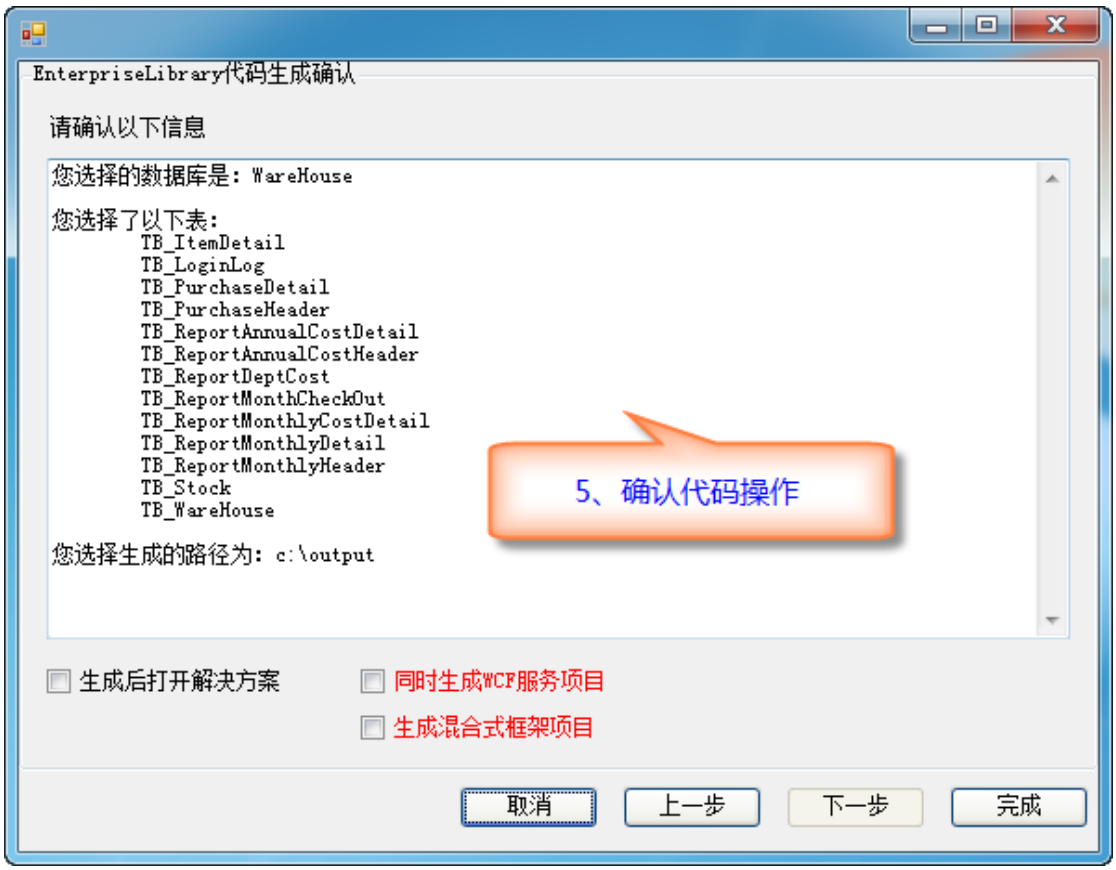
图表 4-18 选定数据库表

第四步，设置相关的代码生成配置信息，包括项目主命名空间，输出目录，以及过滤表名前缀等。



图表 4-19 设置生成配置信息

第五步，确认代码生成操作，一般确认，就会对选定的表进行项目代码的生成操作，生成一体化的整体性解决方案。

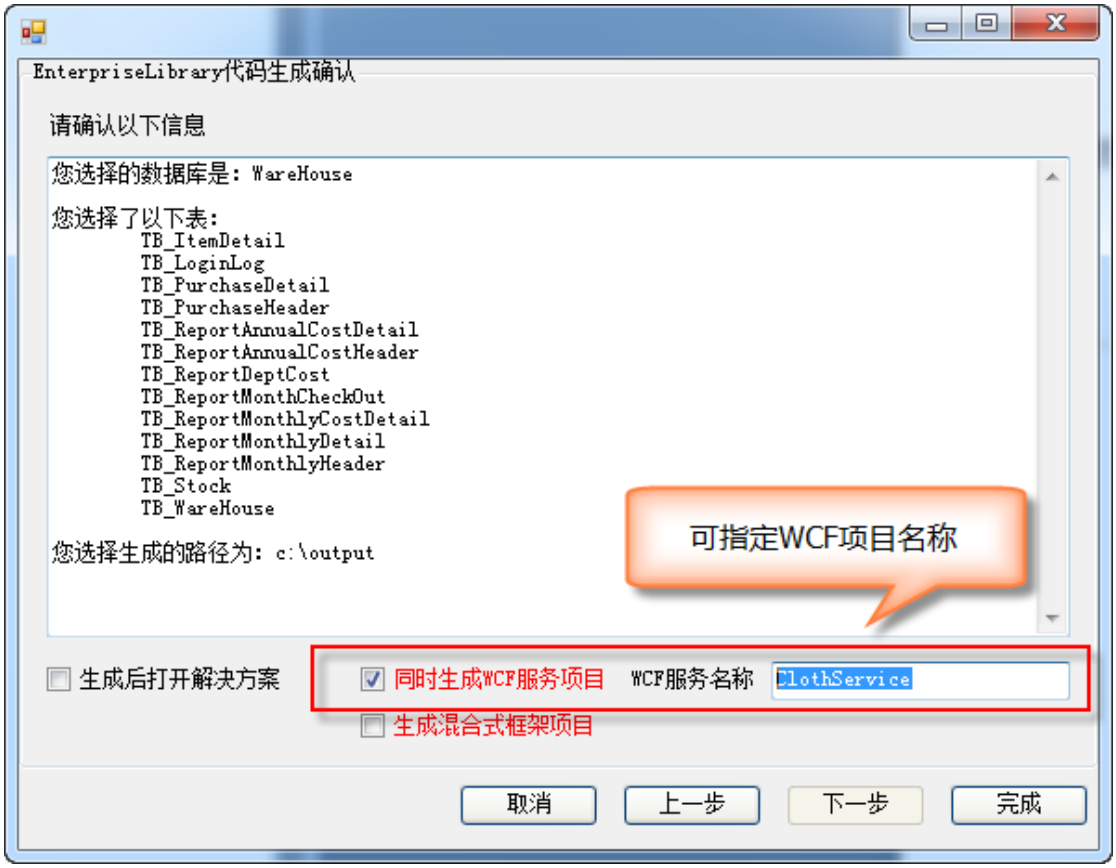


图表 4-20 确认代码生成操作

如果项目是要生成基于 WCF 开发框架的，那么勾选“同时生成 WCF 服务项目”选项即可（工具注册用户可用），勾选后，可以修改 WCF 服务的项目名称，方便创建不同的 WCF 业务项目。WCF 项目是基于 VS2010 开发环境、.NET4.0 框架的项目。

如果项目是生成基于混合型开发框架的，那么勾选“生成混合式框架项目”选项即可（工具注册用户可用），勾选后，“同时生成 WCF 服务项目”也会默认勾选，因为混合型框架式基于 Winform 和 WCF 两种的混合体项目。WCF 项目是基于 VS2010 开发环境、.NET4.0 框架的项目，这是目前最优的配置信息了。

如果勾选“生成后打开解决方案”，那么代码生成完成后，就会自动打开项目工程，你可以尝试编译，一般情况下，可以完全编译通过的。



图表 4-21 WCF 生成信息配置

4.5 数据库设计约定

- 框架的统一约定，方便使用
  - 主键字段统一为ID，类型可以为字符型，整形等，建议采用字符型ID主键。
  - 外键字段一般用“***_ID”方式命名，如Member_ID
  - 字段统一采用Pascal命名方式，Oracle多个名称可用_分开，如CREATE_TIME
  - 除ID外，每个字段设计时都需添加备注，方便生成代码备注和界面标签信息
  - 建议在SQLServer里面进行框架代码生成，然后迁移到其他数据库，如SQLite、MySQL等。
- 推荐字段约定
  - 创建人用Creator，字符型
  - 创建时间用CreateTime，日期型
  - 编辑人用Editor，字符型
  - 编辑时间用EditTime，日期型
  - 部门ID用Dept_ID，字符型
  - 公司ID用Company_ID，字符型

Creator	Creator	创建人	nvarchar(50)
CreateTime	CreateTime	创建时间	datetime
Editor	Editor	编辑人	nvarchar(50)
EditTime	EditTime	编辑时间	datetime
Dept_ID	Dept_ID	所属部门	nvarchar(50)
Company_ID	Company_ID	所属公司	nvarchar(50)

- 表的前缀区分
  - 一般基础表使用 “TB_” 定义前缀，权限系统表使用"T_ACL_"定义前缀，工作流表使用“TBAPP_”，业务表使用"T_"等，
- 命名规则
  - 字段的命名，建议一简单为主，如客户名称，直接使用Name来命名即可，不需要使用CustomerName这样啰嗦的名称。
- 其他约定
  - 如果采用字符型的ID主键，那么我们如果需要正确排序的时候，可能需要增加一个CreateTime的日期类型，方便我们根据日期进行排序。
  - 对于字符型的ID主键字段，代码生成的时候，已经自动添加默认属性值（GUID：System.Guid.NewGuid().ToString() ）。
  - 对于自增长整形的ID主键字段，代码生成的时候，会在映射代码部分进行处理，不在要求插入ID字段的内容。
  - 对于Oracle数据库，则会在构造函数里面，指定使用序列名称，如果使用序列，也会不要求插入ID字段，而改用序列的值进行写入。

- 工具使用
  - 数据库模型设计，使用PowerDesigner进行设计，修改数据库类型为SQLServer2005；
  - 数据库默认使用SQLServer2005或以上版本
  - Oracle数据库使用Oracle9i及以上
  - SQLite数据库管理，可以采用SQLite Developer，如果SQLite数据库损坏，可以采用SQLite Expert Professional进行修复。

5 界面层代码的生成

5.1 Web 界面代码生成

界面开发，无论对于 Web 开发，还是 Winform 开发，都需要耗费一定的时间，特别对于一个数据库字段比较多的界面，一般就需要在编辑界面上摆的更多的控件来做数据显示，每次碰到这个，都有点头痛，反复的机械操作让人挺累，也很烦，但是又必须这样做。

由于数据库字段和界面的排版都有一定的关联关系，因此可以通过代码生成工具 Database2Sharp 的数据库元数据，包含表名称、备注信息、字段列表，以及每个字段的名称、备注、类型等信息，构造一个基础的界面，把重复机械的部分给快速完成，这就是我所说的界面快速生成。当然，对于精致的界面，机械的生成肯定不能满足我们的需要，因此真正的界面需要在这个基础上修改完善一下，但是由于重复劳动部分，已经给工具处理掉了，因此，界面开发效率会大大提高。

Web 界面代码包含目录两种类型的 Web 框架页面生成，包括基于 EasyUI 的 Web 界面代码生成，以及基于 Bootstrap 技术的 Web 界面代码生成。

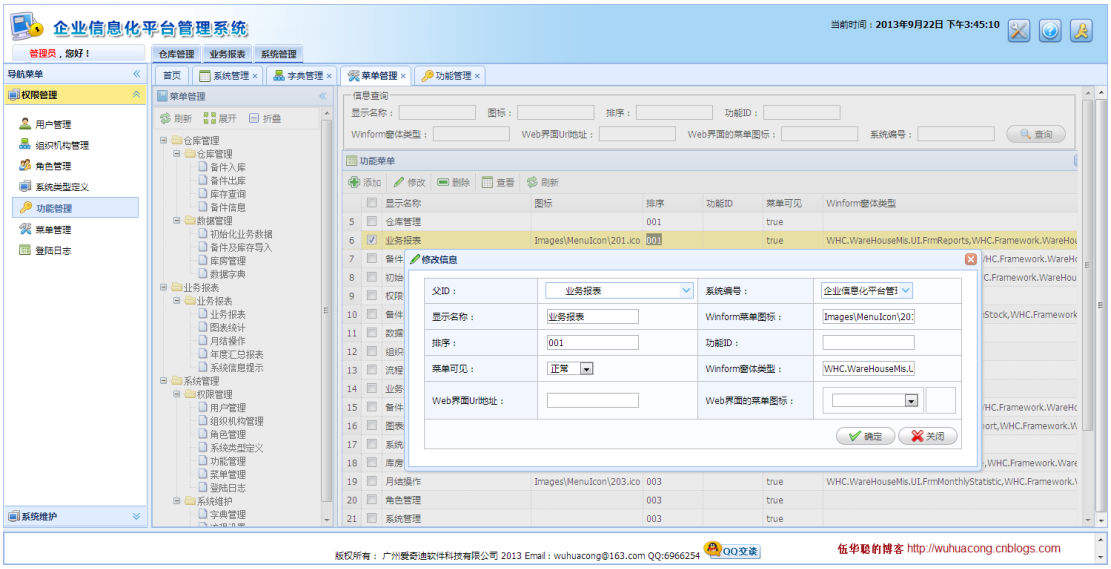
Web 界面代码的生成，一般是基于 Web 框架的基础上进行界面代码的增量开发，基于 Web 框架的基础上，能够快速利用已有的插件模块、类库代码、相关引用等关系，从而节省更多的时间用于 Web 的开发。

5.1.1 基于 EasyUI+MVC 的 Web 界面生成

EasyUI 样式的 Web 界面，一般也是按传统方式的布局进行展示，重要的是需要展示列表内容，新增、修改、查看界面内容，以及相关的导入导出等常规功能。

一般来说，列表带有对数据记录展示、以及分页处理功能，单击添加/修改界面后，会在窗体中间弹出一个层来展示需要处理的内容界面，具体界面效果如下所示。

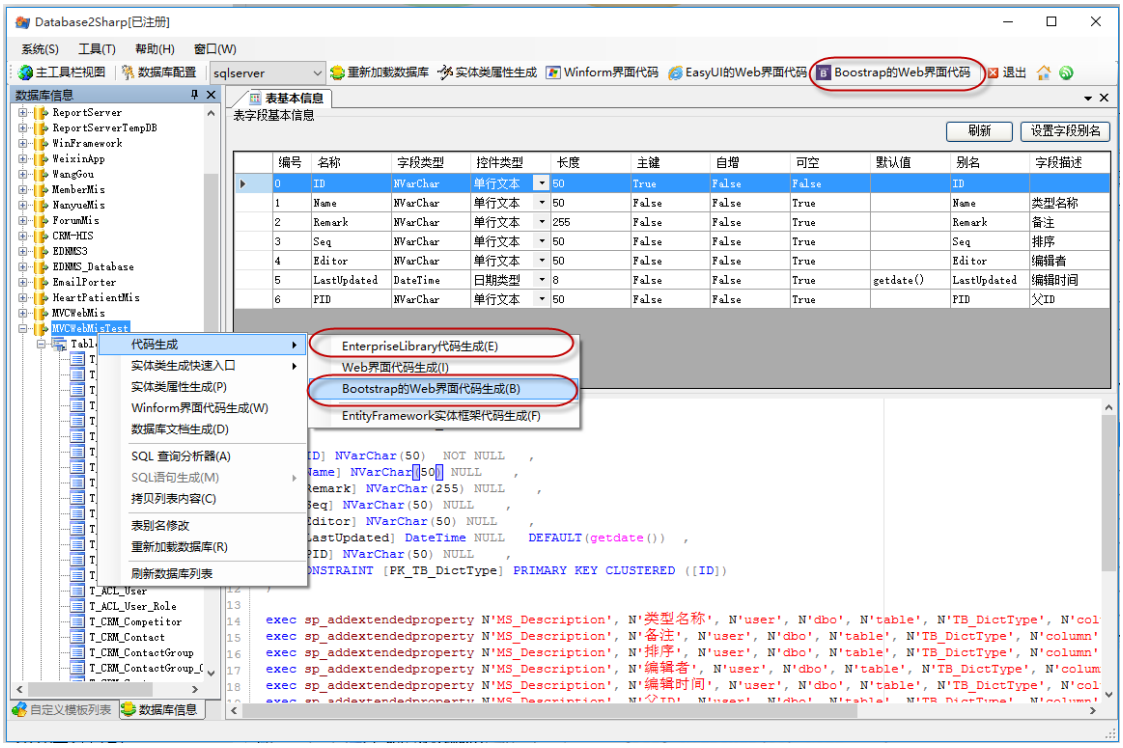
如果是基于 EasyUI 的 Web 开发框架基础上，那么整个界面就可以嵌入在框架里面进行展示，并且只需要增加一个菜单就可以实现界面的展示了，一般情况下，我们可以变调整界面，变查看效果，实现快速可见效果的修改。



有了代码生成工具的 Web 界面生成，界面开发其实是一件不再枯燥的事情，节省更多的时间，做更有意义的事情。

5.1.2 基于 Bootstrap+MVC 的 Web 界面生成

Enterprise Library 代码生成，可以快速生成除界面外的整体性的框架代码，Bootstrap 的 Web 界面代码生成，可以快速生成基于 Metronic 的 Bootstrap 的前端界面代码和后台控制器代码，界面部分包括查询、分页、数据展示、数据导入导出、新增、编辑、查看、删除等基础功能界面，生成后我们可以基于这个基础上进行简单、快速的修改即可符合实际需要，极大提高我们 Web 界面的开发效率。

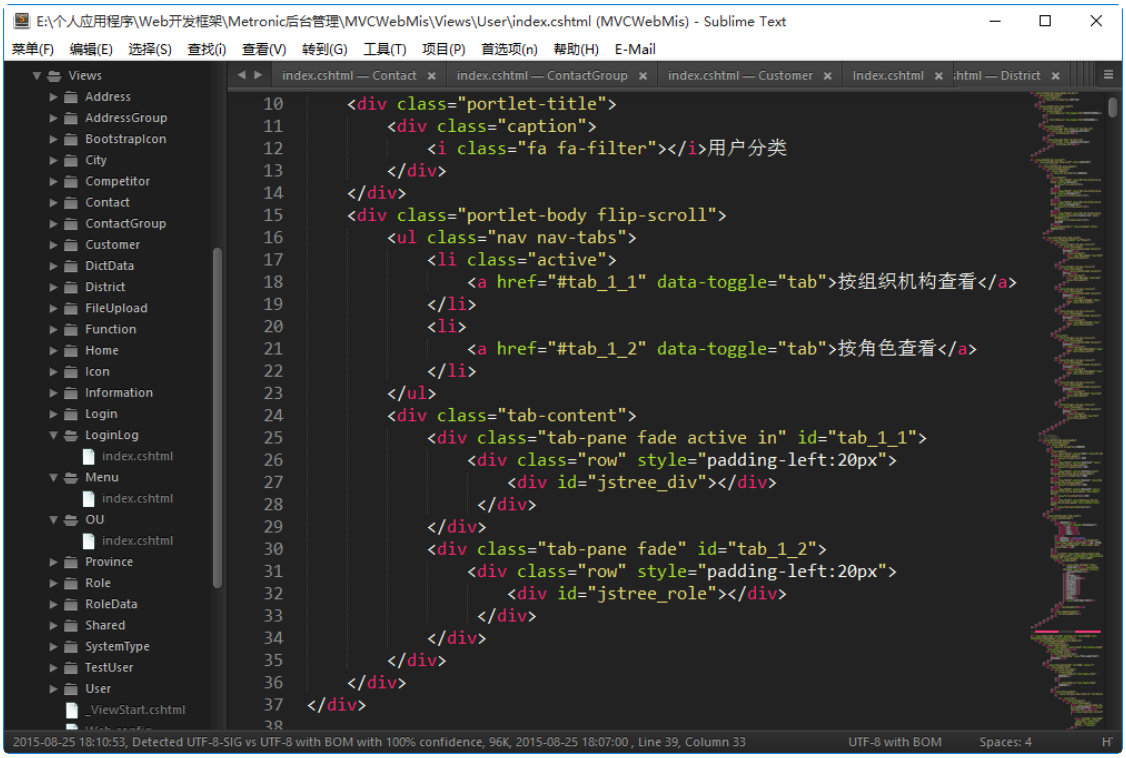


使用 Boostrap 的 Web 界面代码生成，会逐步让您选定具体的表进行操作，生成的代码是标准的列表分页、排序、增删改查等操作的层代码，我们可以根据具体的需要进行布局的调整和控件的调整。生成后的标准 HTML 视图代码如下所示，其中包含查询分页部分，添加修改界面部分，查看信息部分，以及导入的处理界面等内容。

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
    ViewBag.Title = "用户登录日志信息";
}

@*脚本引用放在此处可以实现自定义函数自动提示*@
<script src="~/Scripts/CommonUtil.js"></script>

<div class="portlet box green-meadow col-md-2">...</div>

<!--------------------------数据查询及分页处理---------------------------->
<div class="portlet box col-md-10">...</div>

<!--------------------------添加/修改信息的弹出层---------------------------->
<div id="add" class="modal fade bs-modal-lg">...</div>

<!--------------------------查看信息的弹出层---------------------------->
<div id="view" class="modal fade bs-modal-lg">...</div>


<!--导入数据操作层-->
<div id="import" class="modal fade bs-modal-lg">...</div>


@section footerScript {

<script>...</script>

<!--添加对uploadify控件的支持-->
<script>...</script>

}
```

一般情况下，我们生成的 Web 界面内容，以及后台代码，都可以在 VS 开发环境上进行编辑使用，不过修改 Web 的视图 HTML 代码，我们一般推荐使用 Sublime Text 这个强大的编辑工具的，丰富的插件、智能语法提示等，会让您用了之后爱不释手，是编辑视图页面非常快速的利器，强烈推荐使用。

VS 一般我们用来做文件管理，以及编译等处理就可以了。

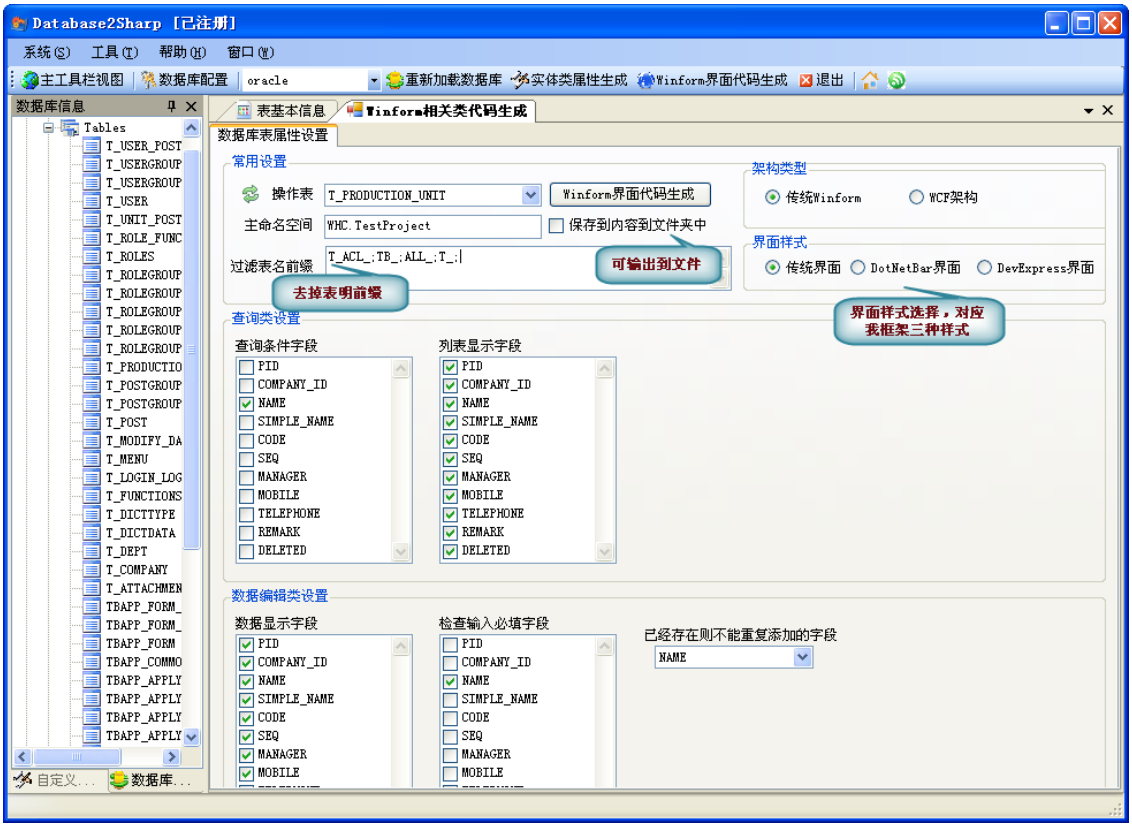


5.2 Winform 界面代码生成

除了 Web 的界面开发，Winform 的界面开发，当然也很重要，枯燥的事情一样可以交给代码生成工具 Database2Sharp 进行生成，不需要让人工反复的做这些无用功，或者是技术含量不太高的东西。

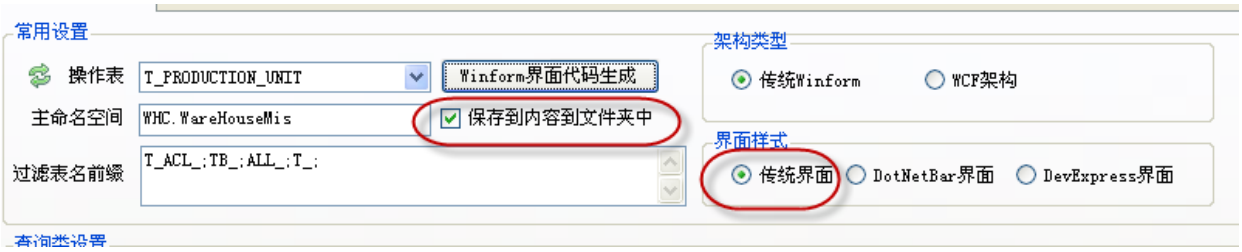
上面 Web 界面工程，我是采用 SQLServer 的数据库进行生成代码的，这个例子，我们介绍一下 Oracle 数据库的生成方式，其实这个代码生成工具，对应用什么数据库不重要，都会很好的给你生成相关的界面信息，不管你是用 SQLServer 还是 Oracle，或者是 Sqlite、Access、MySql 等等，都一样可以很好的生成所需的开发代码的。

Winform 界面代码生成的主界面如下所示：



图表 5-1 Winform 界面代码生成

Winform 界面生成界面提供了很多参数进行控制，以期生成精细化的界面内容。Winform 界面生成对应我们的 Winform 开发框架，提供了三种不同的界面样式，包括传统界面样式、DotNetbar 界面样式和 DevExpress 界面样式，这几种样式有不同的应用场景，是目前应用最广的几种方式。

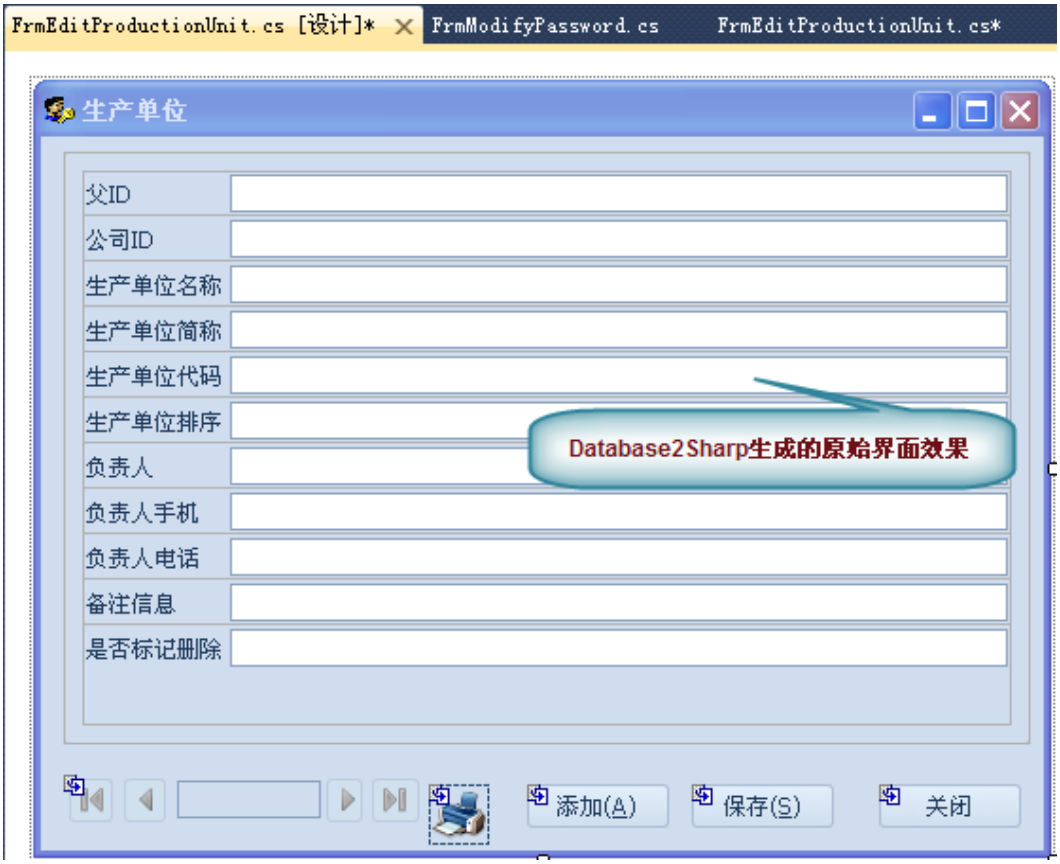


图表 5-2 Winform 界面代码生成 2

5.2.1 Winform 数据编辑界面生成

（一）基于 DevExpress 的 Winform 界面快速生成

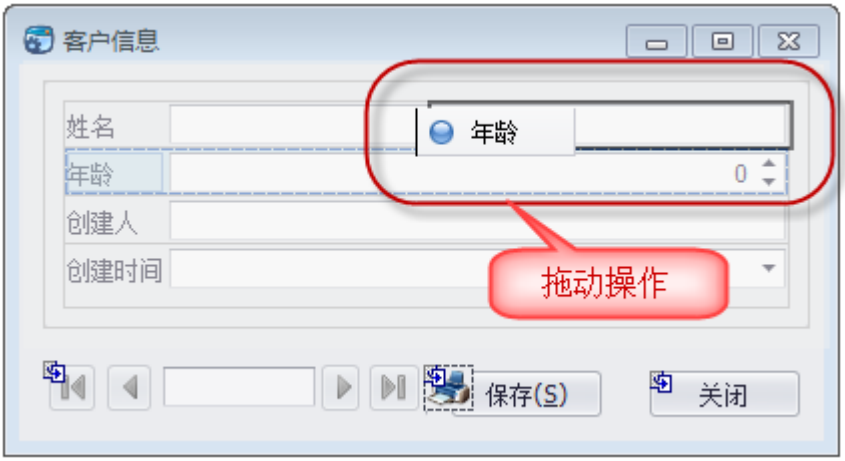
工具生成了界面布局代码，以及界面后台逻辑代码，这样的代码正是我们开发所需要的，我们看到生成后的界面代码（没有修改代码的），在 VS 里面的真实效果如下。

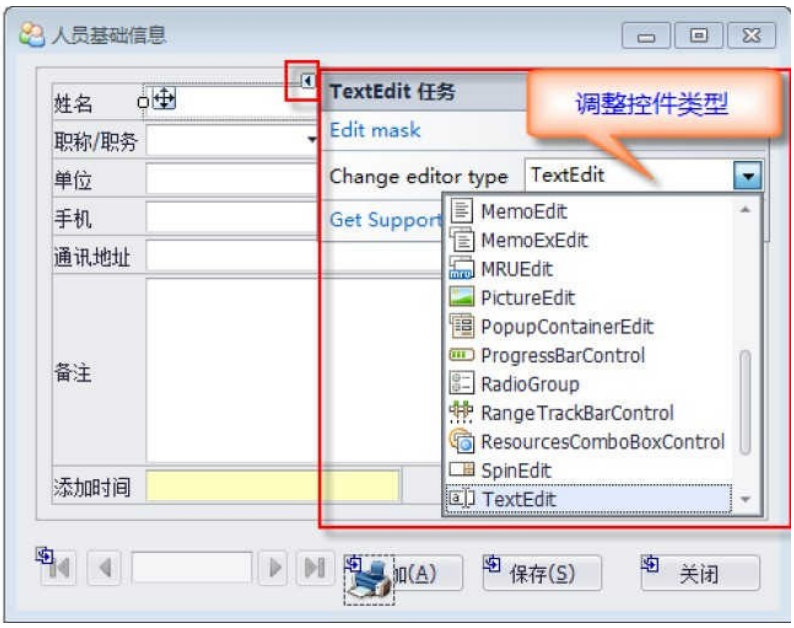


图表 5-3 Winform 编辑界面 1

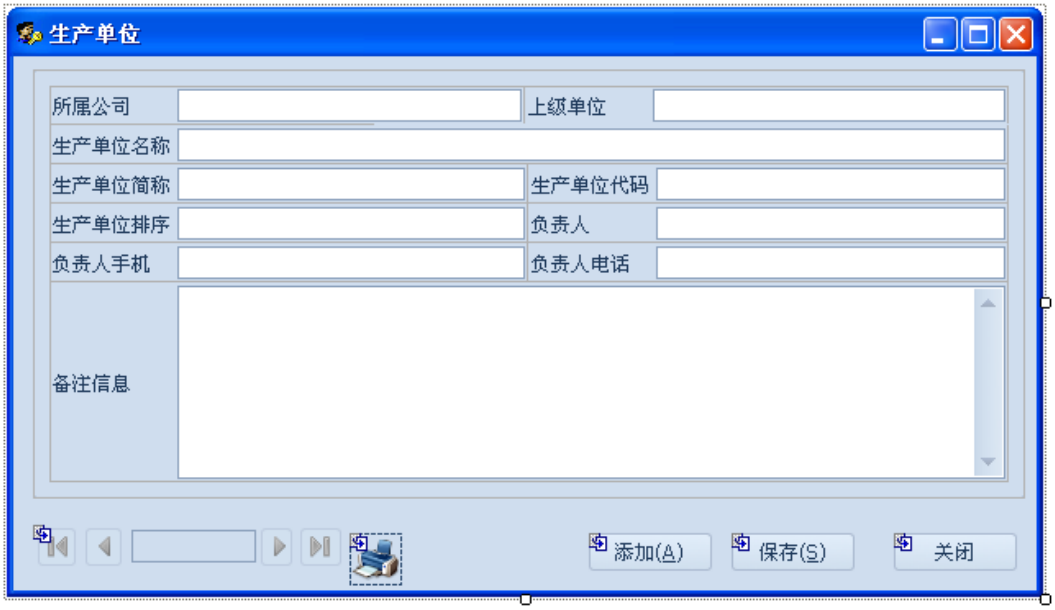
上面的布局采用了 LayoutControl 来进行布局控制，这是一种很好的布局控制方法，它除了使得界面更加美观外，还能非常自由调整每个控件的位置及大小。

当然，我们一般为了美观需要，会对界面进行一定的调整，包括控件布局位置，控件类型的调整。





由于 DevExpress 控件类型变化切换很方便，所以这种调整很自由高效，调整后的界面如下所示。



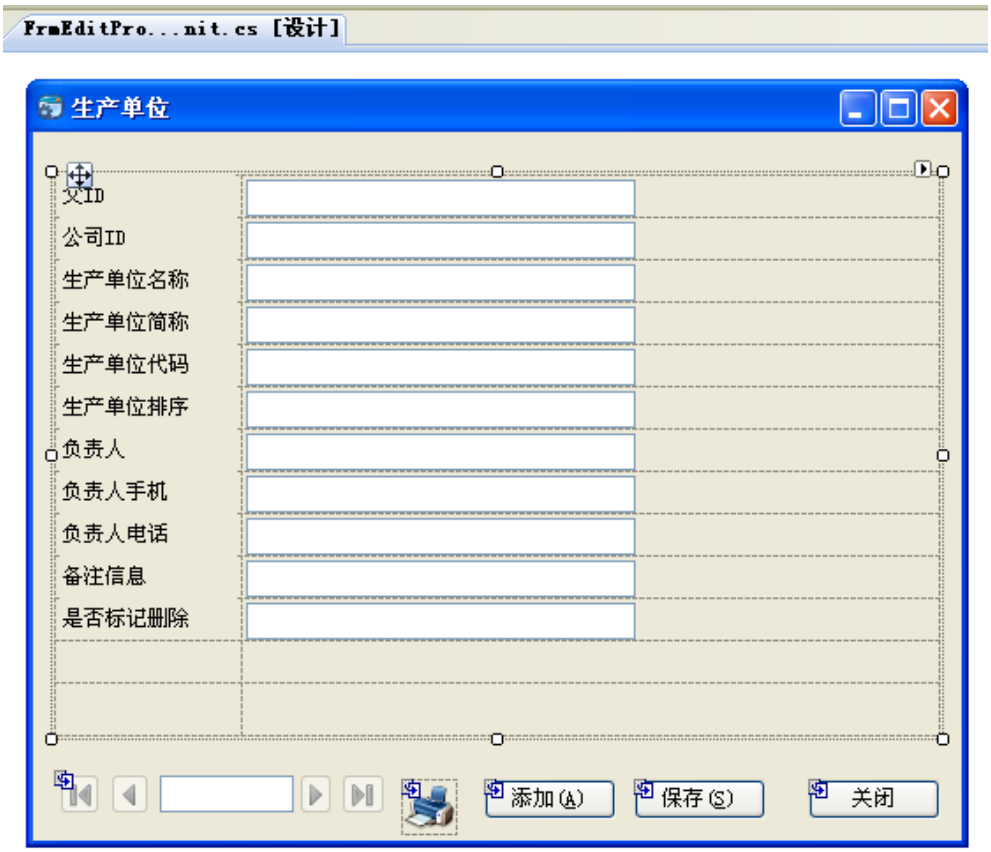
图表 5-4 Winform 编辑界面 2

比起利用 Database2Sharp 工具，十几秒钟弄好这样一个界面，纯手工做这样的界面，没有十几分钟，估计弄不好的，而且容易出错，最重要的问题，就是团队开发的时候，这个统一性就很有保证，开发效率高，带来 Bug 也会很少。

（二）基于传统的 Winform 界面快速生成

传统的 Winform 界面和 DotNetBar 样式的界面生成，和上面的 DevExpress 样式代码很类似，不过控件不同而已，由于不同的需要，用传统界面元素开发，

也是一种常见的开发模式，因此我们的代码生成工具也支持这种常规的界面生成，来辅助大家做好界面的工作。得到的界面效果如下所示。



图表 5-5 Winform 编辑界面 3

和 DevExpress 控件界面代码类似，这里也使用了 TableLayout 的布局控件，用来较好控制布局的高度宽度，以及每行位置等，这样整体效果就很好，也容易进行界面元素的控制。

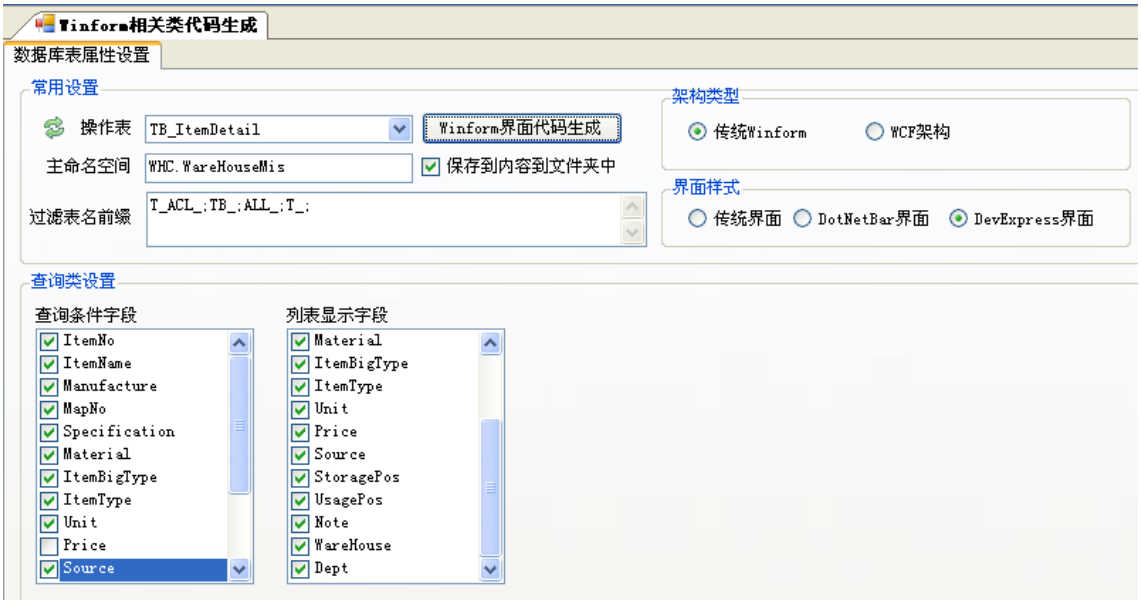
5.2.2 Winform 查询列表界面生成

很多情况下，查询列表界面很常见，如果能快速生成标准的界面，除了可以节省时间，提高开发效率外，也给我们统一界面风格及代码风格等方面，提供更好的支持。

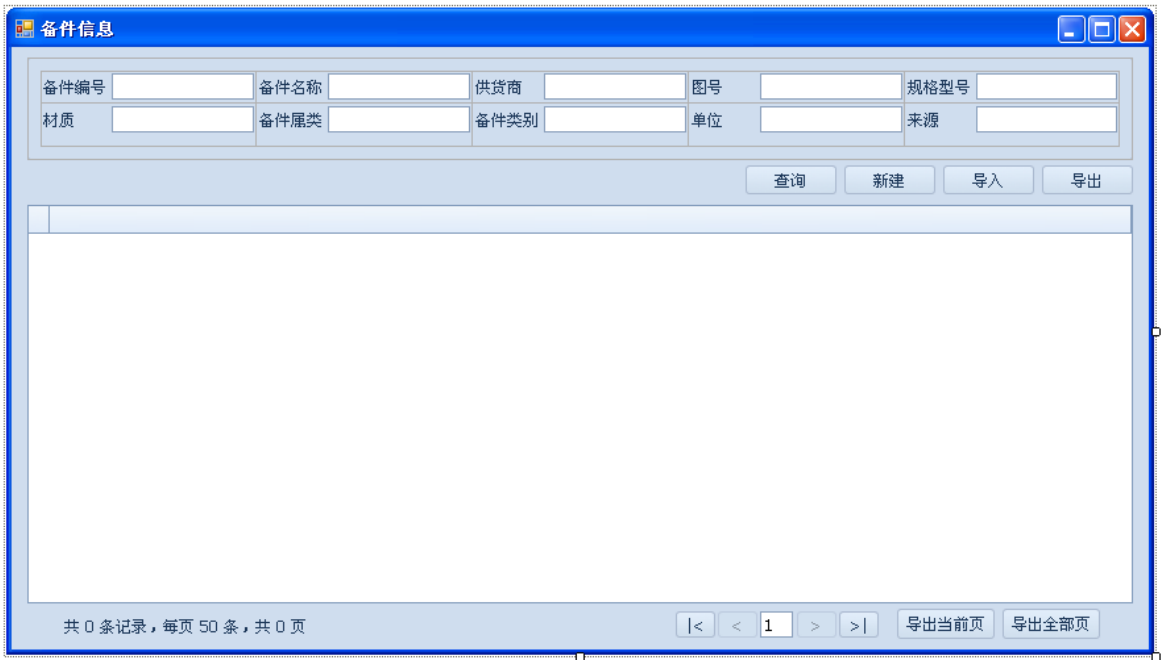
复杂累赘的界面能够自动生成，绝对是开发过程的一大提升，让我们更加享受开发的乐趣。

1）设置好相关的界面参数，如指定列表的查询字段、列表显示字段，选择界面样式，以及设定代码的主命名空间等参数。

2）生成界面代码到文件后，把文件直接复制到项目中，不用修改直接就可以看到列表界面效果，Yeah，正是我们需要的样式。



图表 5-6 Winform 查询列表界面生成设置

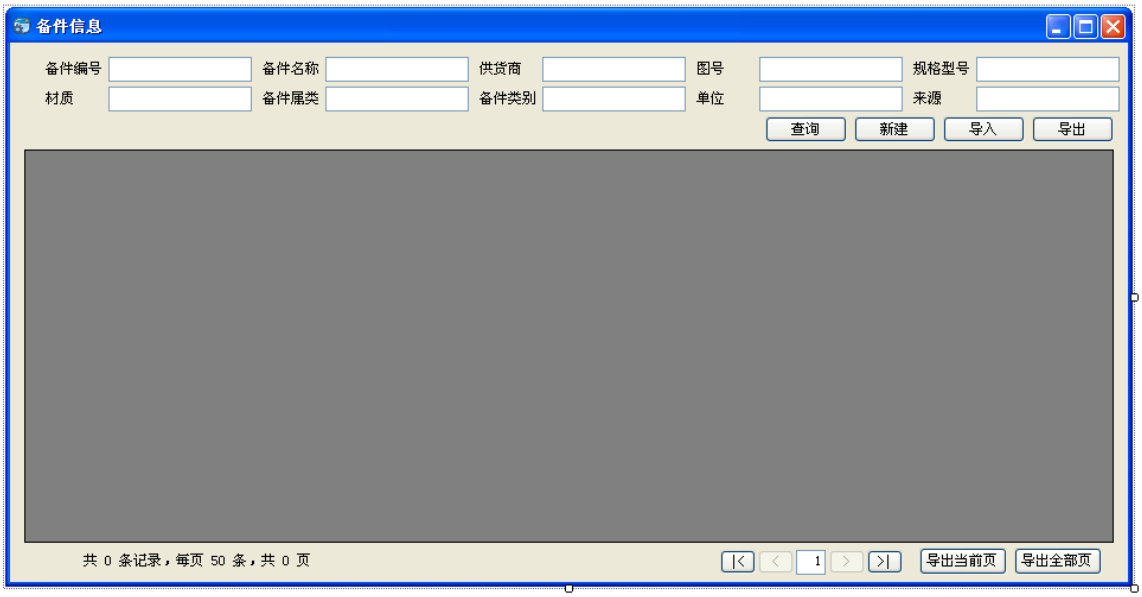


图表 5-7 Winform 查询列表界面效果

当然，上面的查询条件可以一行，也可以两行或者多行，界面生成的时候，会自动合理计算好布局，保证完美展现我们想要的列表界面效果。以上界面几乎不用任何修改就直接可以编译运行，里面的后台代码也同时生成了。

后台代码里面生成包括，分页控件展示及列表查询显示、Excel 数据导入、数据导出等功能的后台界面代码。

传统界面效果生成操作，只需要选择传统界面样式，生成即可，得到的初始化界面如下所示。



如果是只是指定了几个查询条件（一行的情况），那么工具会自动计算好布局位置，得到界面效果如下所示。

代码生成工具 Database2Sharp 还提供了生成基于 DotNetbar 的样式的查询列表界面代码，和上图类似，在此不再赘述。

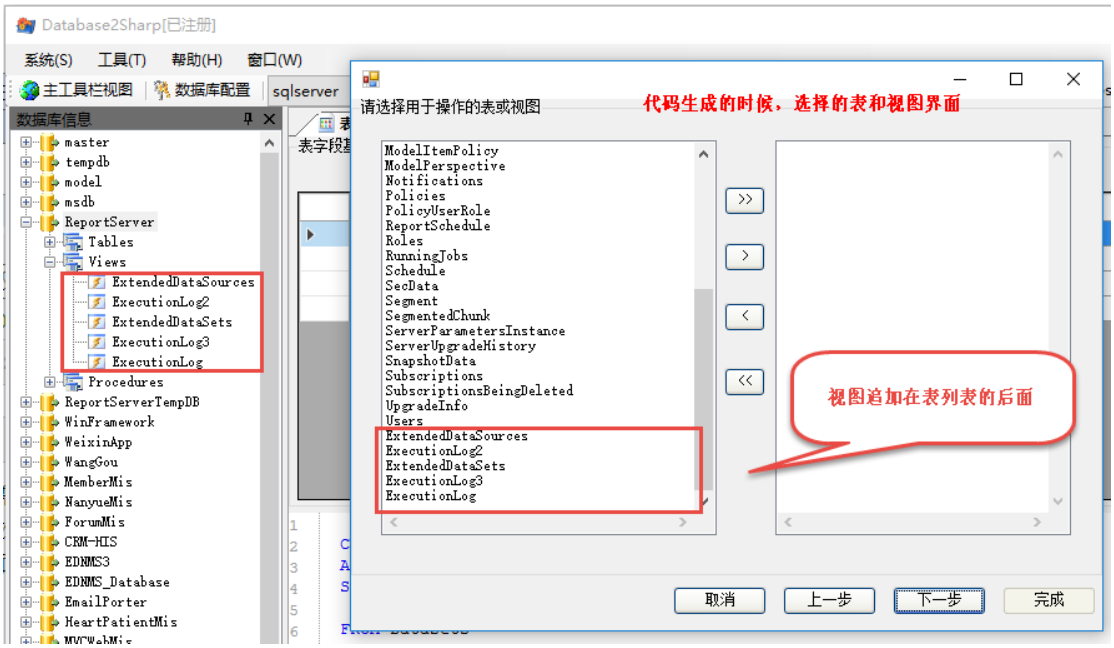
通过代码生成工具，不仅可以生成整体性的 EnterpriseLibary 框架结构代码，连我们繁琐的 Web 界面、Winform 界面都能快速、高标准生成，开发界面其实是一件很轻松快速的事情，不要整天从数据库字段和界面属性设置中来回切换了。有了 Database2Sharp 代码生成工具，一切变得宁静很多，但更加快速。

5.2.3 基于视图的代码生成

在代码生成工具的各种功能规划中，我们一向以客户的需求作为驱动，因此也会根据需要增加一些特殊的功能或者处理。在实际的开发中，虽然我们一般以具体的表进行具体

业务开发，但是有些客户提出有时候视图开发也是很常见的，为了提高代码生成和界面生成的效率，基于视图开发的过程也应该支持。还有主从表的界面生成操作，在很多实际的业务领域也是很常见的。基于上面的需求，我们在 Database2Sharp 中增加视图的代码生成以及主从表界面生成功能，为客户的高效率开发快马加鞭、保驾护航。

为了支持视图的相关代码生成，我们把代码生成工具底层的元数据进行了优化整合，是指在代码生成方面，具体的表和视图不再有具体的差异，基本上都是可以统一对待，实现快速的框架代码生成、Winform 界面生成、Web 界面生成操作的，所有的表的相关属性，视图也具有，因此在代码模板方便，不需要进行调整，兼容了代码模板的属性处理，提高了已有代码模板的安全性。

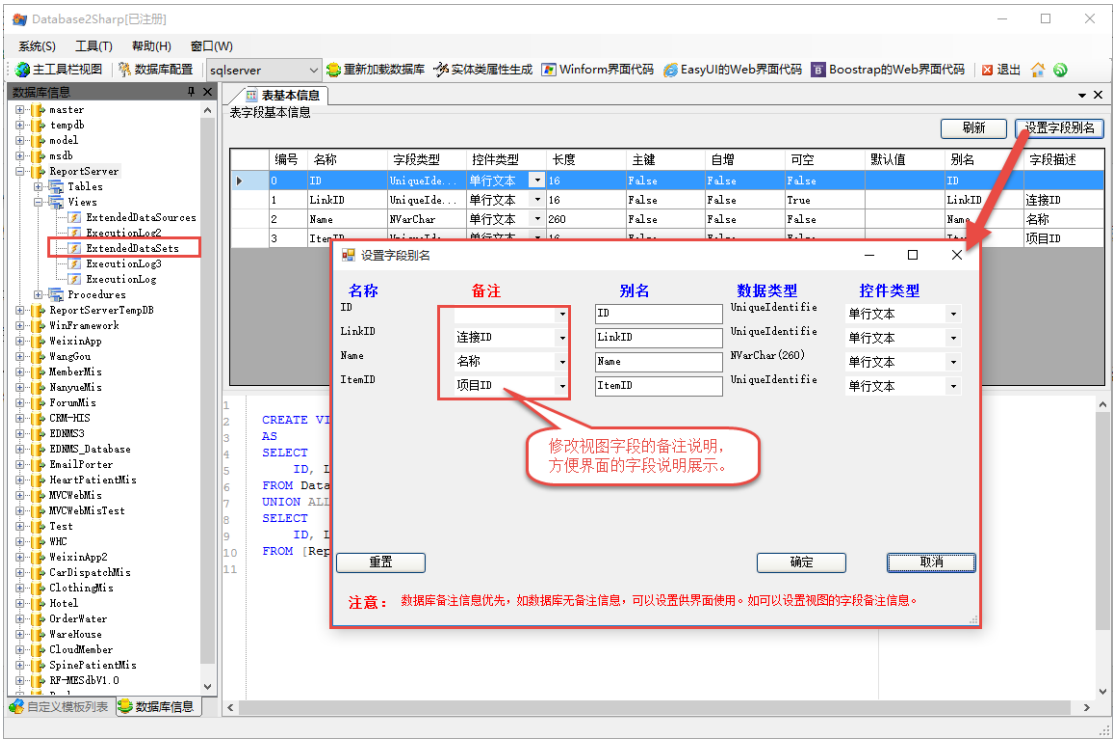


如上所示，为了区分表和视图的位置，我们把视图统一放在了表的后面，方便选择处理，在选择的时候，表和视图都是一视同仁，所以代码生成的处理适用于表的，也是适用于视图的，可以统一一并生成，极大的提高了代码生成的效率。

当然，视图的增删改操作，和表毕竟不一样，一般视图只是为了查询数据进行展示的，所以在实际开发的时候，可以适当屏蔽视图的增删改操作，或者自定义相关的接口进行处理。在 Winform 界面的生成的时候，我们也一样整合了视图的列表，可以基于视图进行界面代码的生成。

不过我们注意到，一般表我们使用备注信息作为 Winform 界面的字段说明信息的，如果是视图，那么是无法获取到它的视图字段备注信息的，因为视图的字段备注是不存在的，为了实现和表一样具有中文备注的界面，我们增加了一个对视图字段进行备注信息维

护的界面，有着字段的备注，我们生成 Winform 界面的时候，对应字段的标签就有中文信息了。



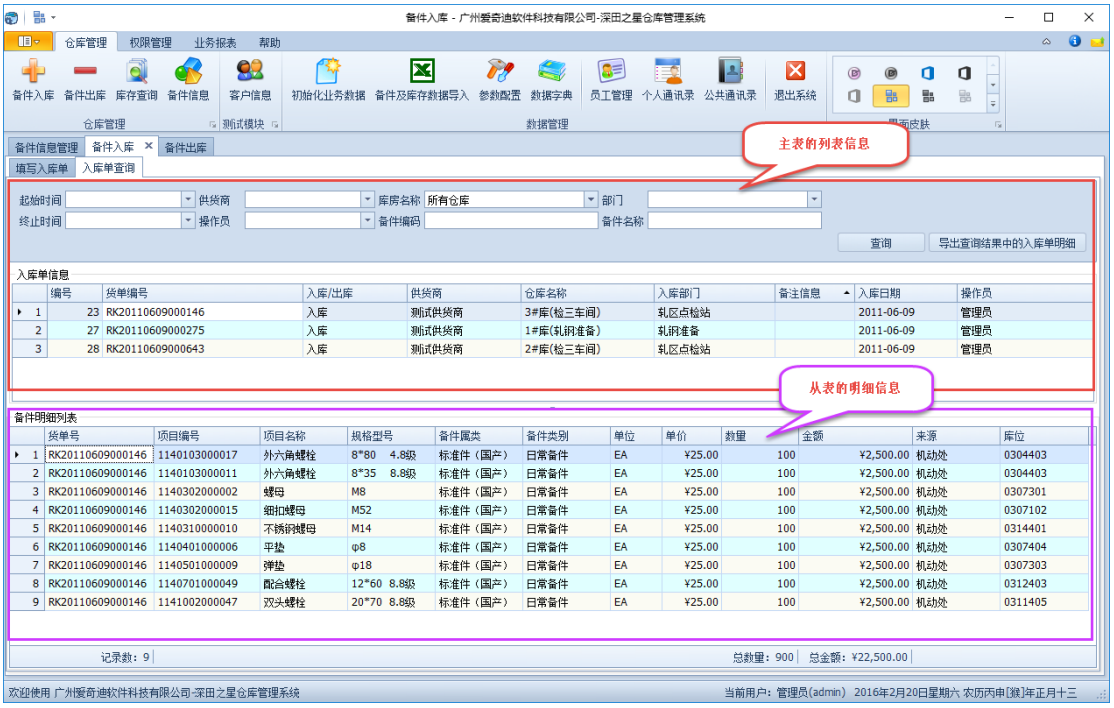
通过 Enterprise Library 架构生成的相关代码，也同时具有相关的备注信息，如下代码所示。

通过代码生成工具里面的 Winform 界面代码生成，当然也会具有相关的备注信息，可以在界面上显示对应的中文标签信息了。视图的 Winform 界面代码生成和普通的表生成的 Winform 界面操作过程一样，不在赘述。这样生成对应视图的 Winform 界面操作和普通表的处理方式一致，而且对应的视图字段也有了备注信息，因此在界面上的标签说明也就和表一样，可以显示备注信息了。

5.2.4 主从表的界面生成操作

在有些情况下，有些业务表是具有主从关系的，如一个是汇总信息，一个是明细信息，如仓库的入库、出库操作，会员的消费等操作，都是典型的主从表应用场景，可以把它们作为一个界面生成的案例进行处理。

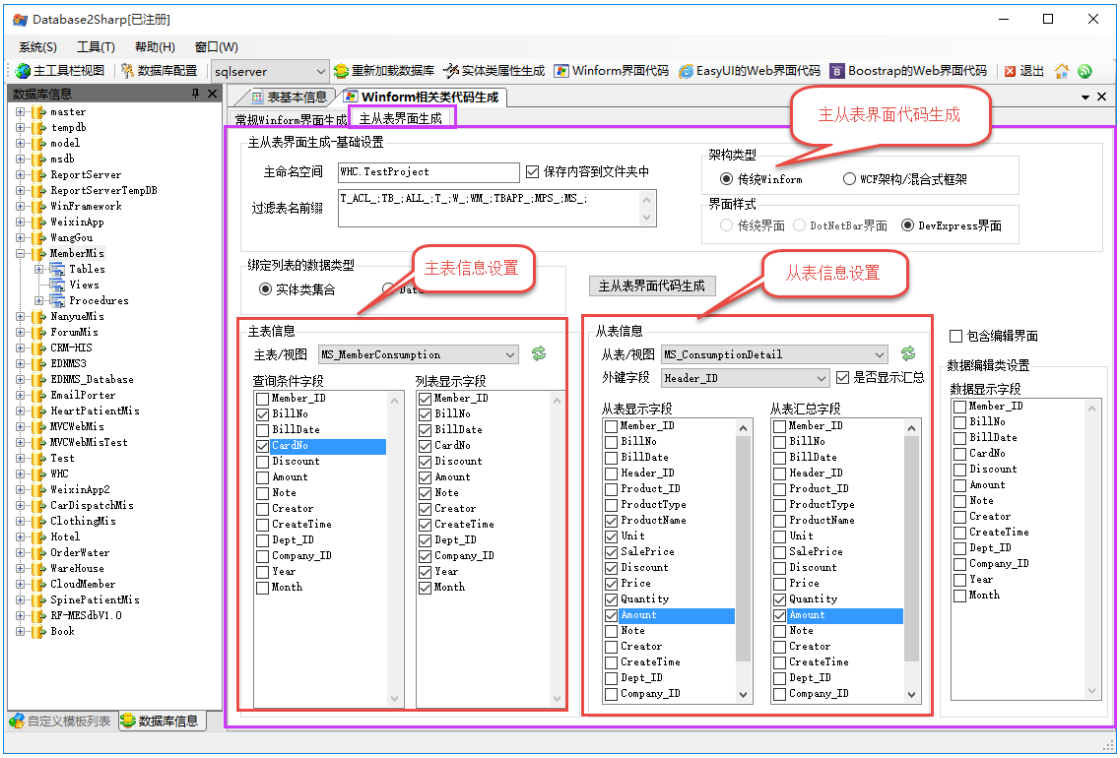
标准的主从表界面如下界面所示。



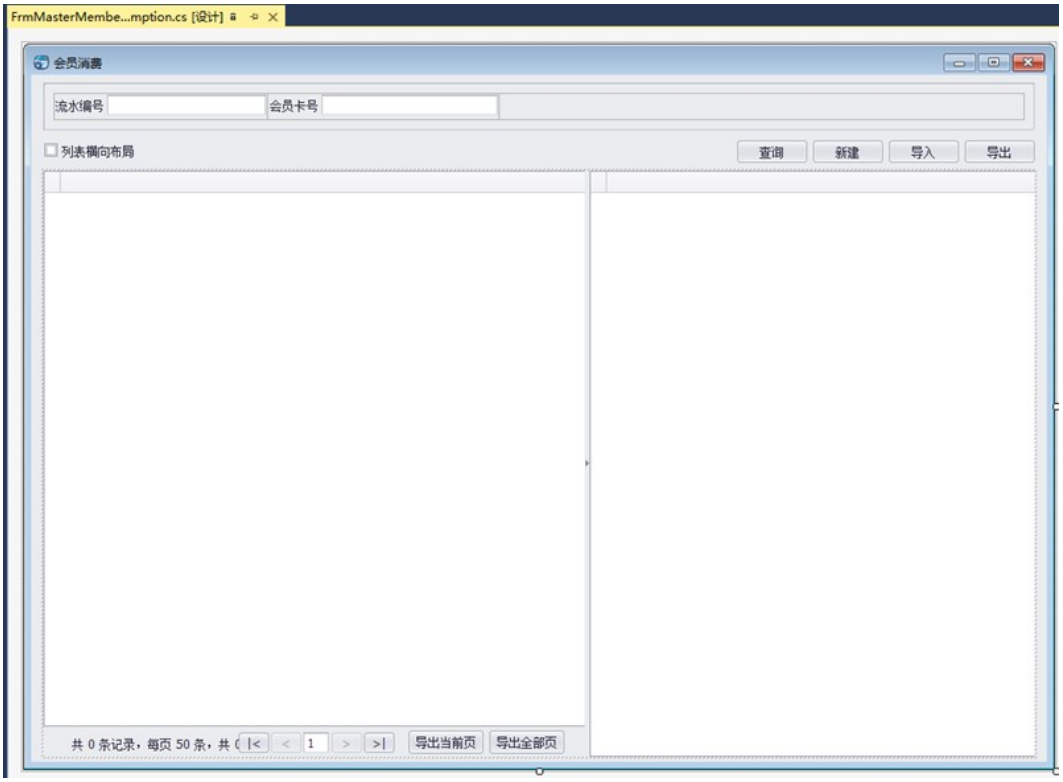
或者会员消费信息的横向界面展示如下所示。

在我之前的代码生成界面里面，主要是生成标准的分页列表展示，以及编辑明细两个标准界面，为了更加丰富开发工具的界面生成，根据上面两种界面的综合情况，增加一个主从表的界面生成操作，这样可以更加适应实际的业务开发工作，高效进行界面的快速生成。

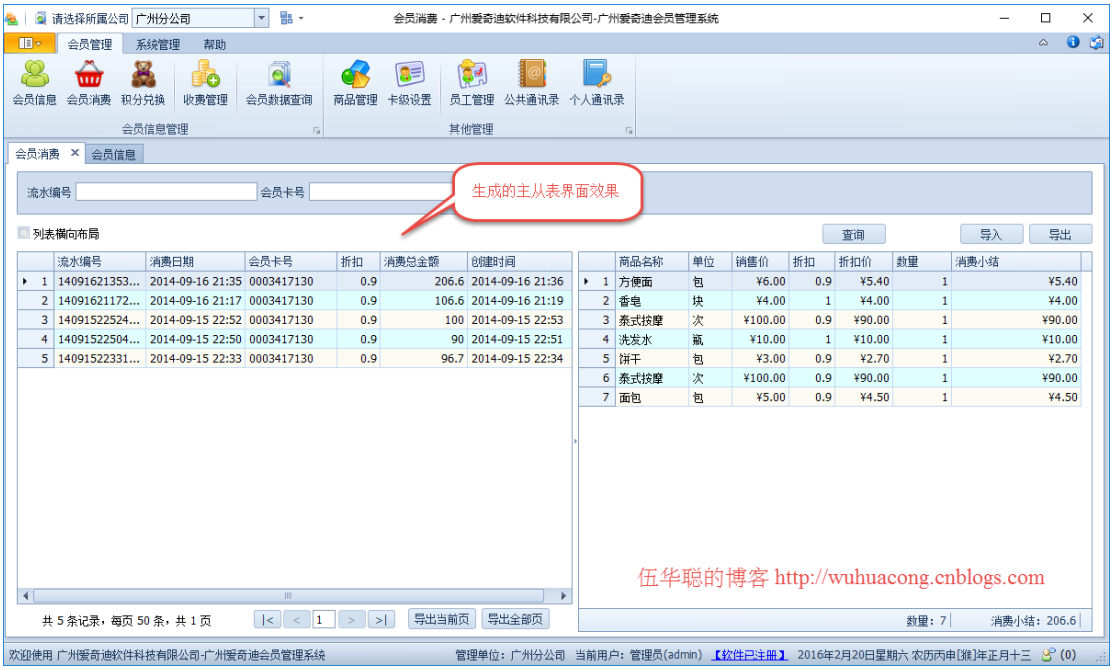
如在代码生成工具里面，设置主从表的界面生成如下所示。



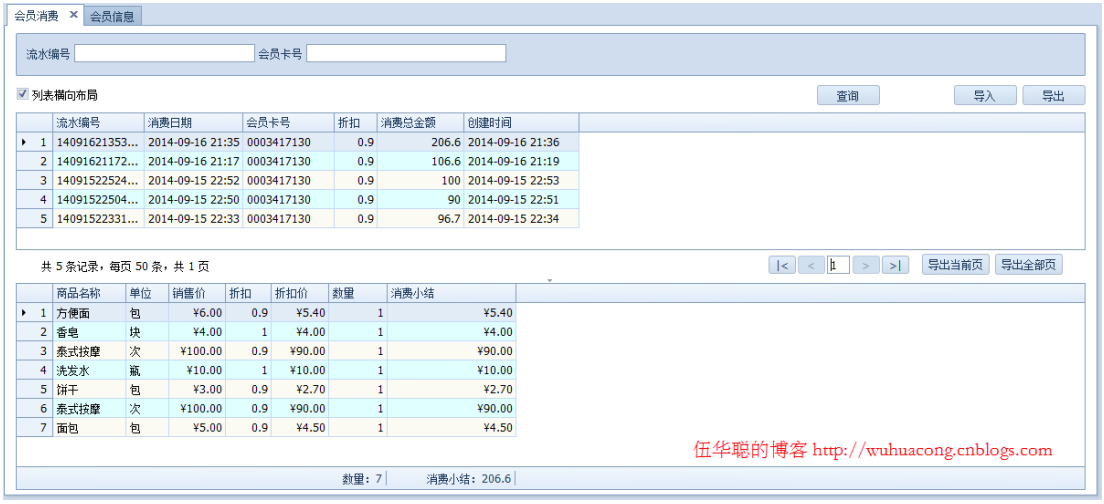
生成 Winform 界面代码后，在 VS 编辑器里面，可以看到如下所示界面。



最后在实际程序上运行生成的界面，就可以看到预览的界面效果了，界面效果如下所示。



为了方便，上面界面增加了一个复选框，用来切换横向或者纵向展示明细列表的，勾选后进行横向显示，如下所示。



上面主从表的展示，还包括了明细表信息的汇总功能，这样基本上满足了主从表的信息展示了，如果需要，还可以在这基础上进行更方便的改进了。

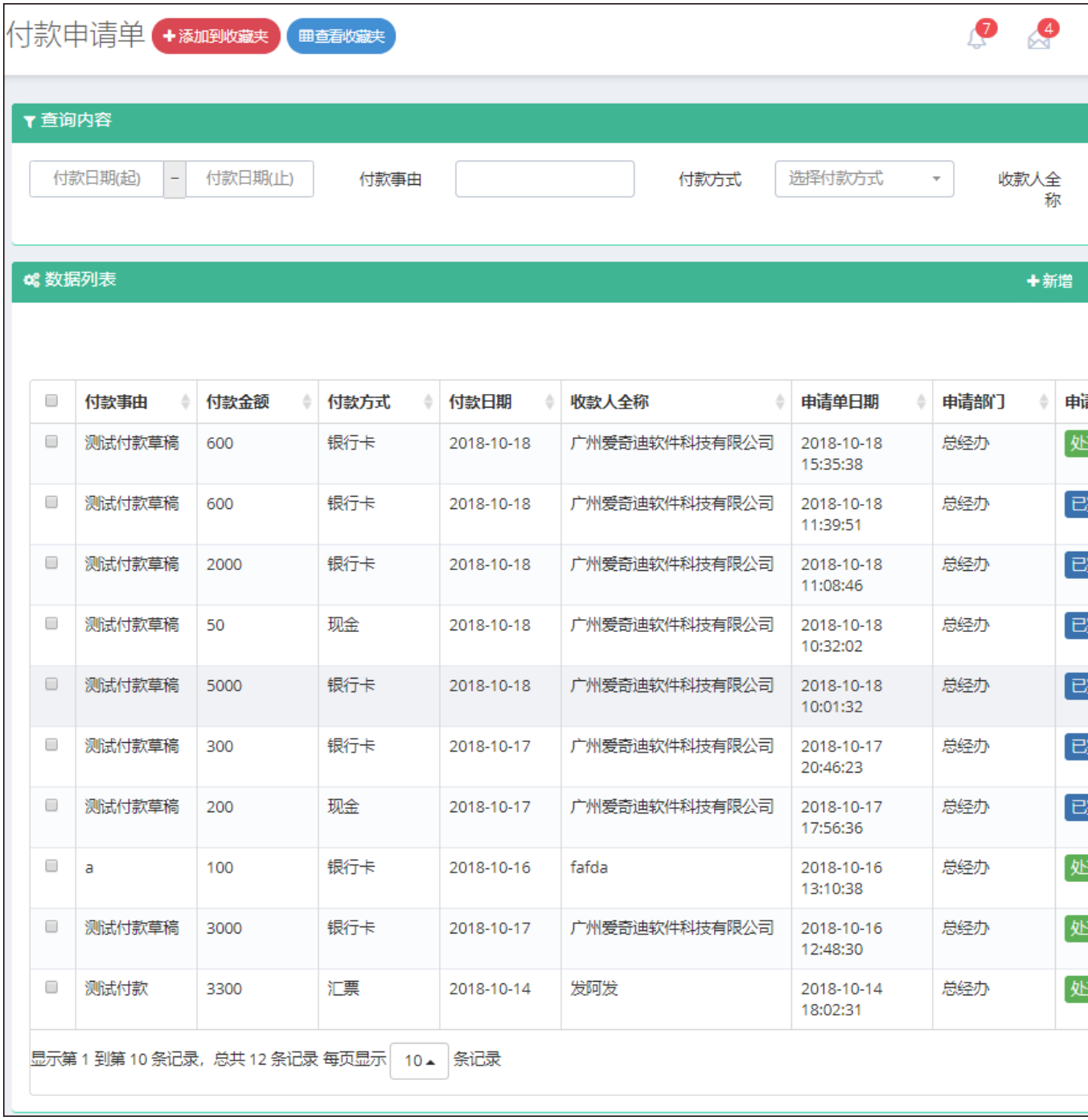
6 工作流界面代码生成

6.1 Bootstrap+MVC 的工作流界面生成

6.1.1 工作流界面功能

工作流模块如果要增加一个业务表单的处理，那么界面包括了列表界面，创建和编辑申请单界面，查看申请单明细这几个界面，以及对应后台控制器的代码。其他共用的界面和代码，则是在整个工作流模块中通用的，不需要变化。

我们来关注下如果增加一个业务表单的情况下，需要的列表界面，创建和编辑申请单界面，查看申请单明细这几个界面。

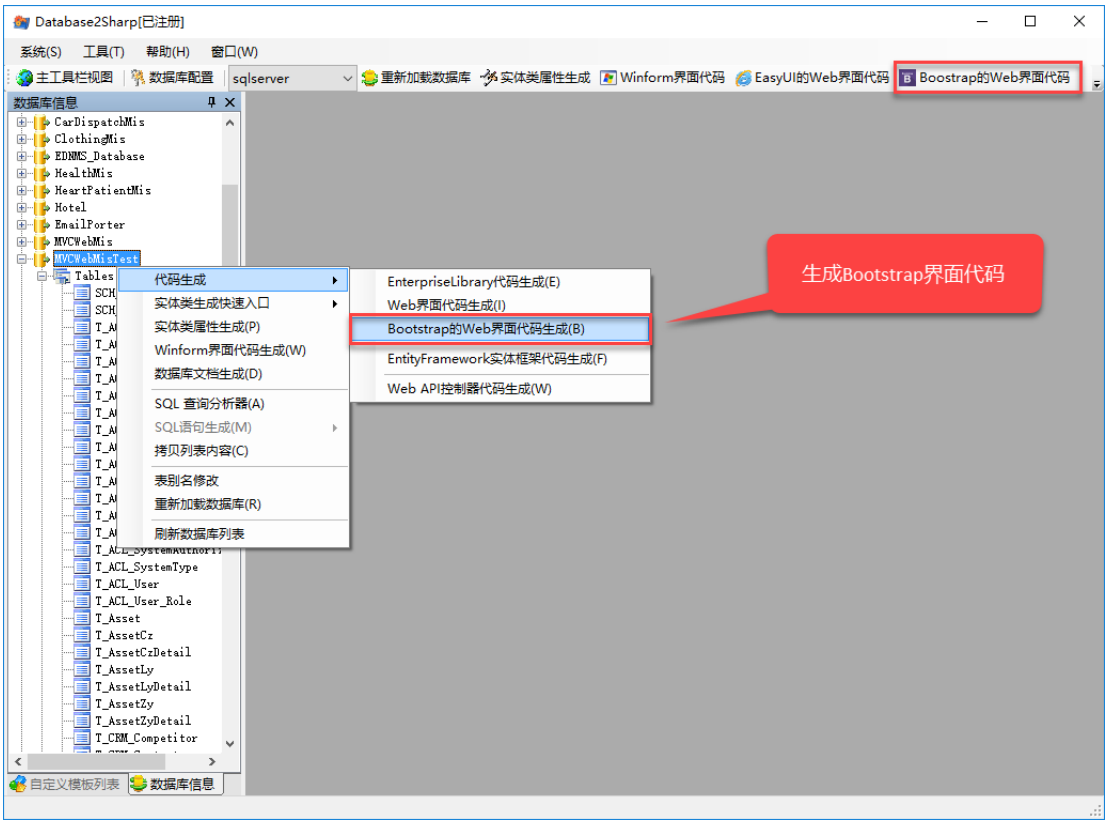


这些使用代码生成工具 Database2Sharp 快速生成工作流模块界面，是集成了我们整个工作流处理方式，包括列表界面可以分页查询数据、编辑表单中选择用户、处理附件，

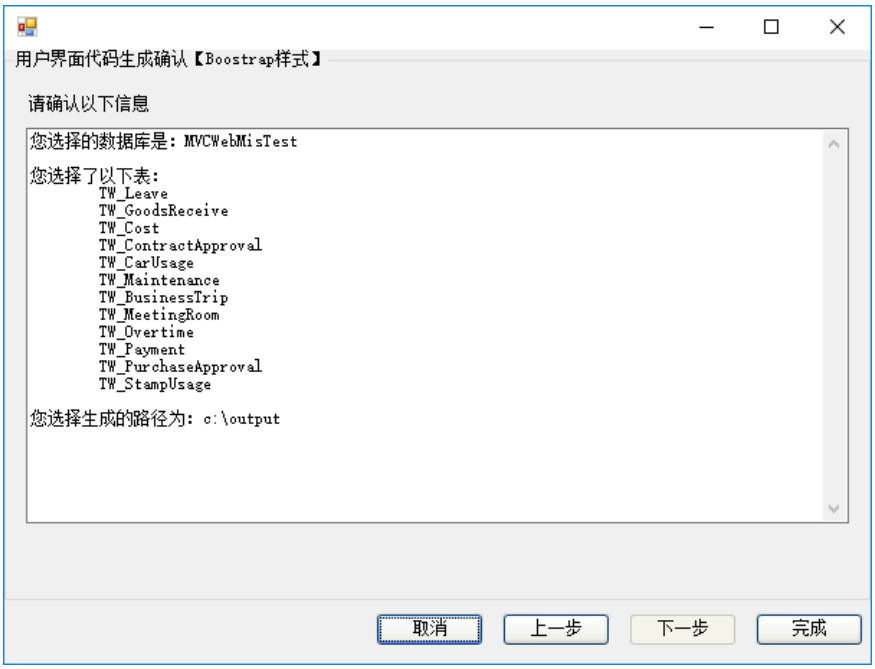
以及查看明细界面中集成的各种流程处理步骤，包括审批、会签、退回、拒绝、查看流程日志、打印表单等等常规处理步骤。

6.1.2 使用代码生成工具快速生成工作流界面

和常规的代码生成工具生成代码一样，我们打开代码生成工具，然后展开数据库表后，通过菜单的【Boostrap 的 Web 界面代码生成】生成对应的代码即可。



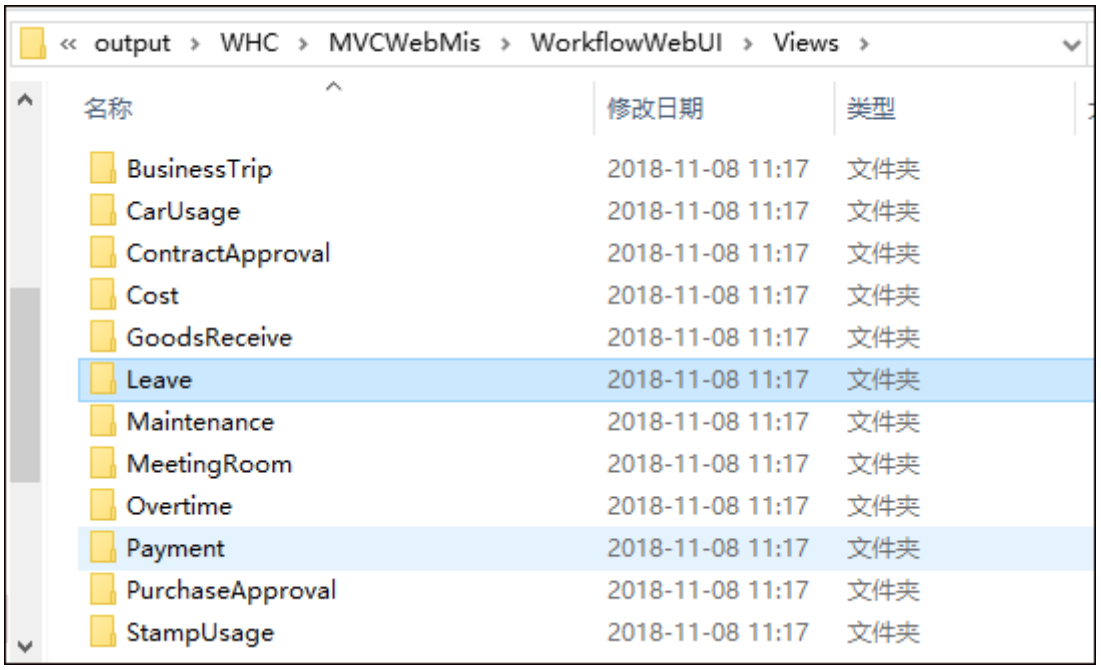
通过选中对应的数据库表，就可以继续一步步处理了，最后确认代码生成即可。



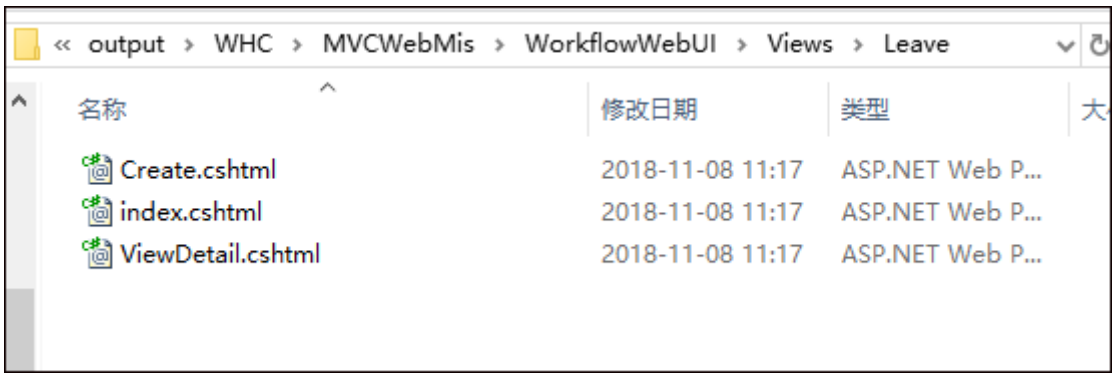
生成代码后，我们可以看到在对应的目录有两个目录，MVCWebUI 和 WorkflowWebUI 目录，如下所示。

两个目录 MVCWebUI 和 WorkflowWebUI，其中 MVCWebUI 包含了常规 Bootstrap 框架的页面视图和控制器代码文件，如下所示。

而 WorkflowWebUI 目录则是我们这里需要重点关注的工作流视图页面代码文件，如下所示。

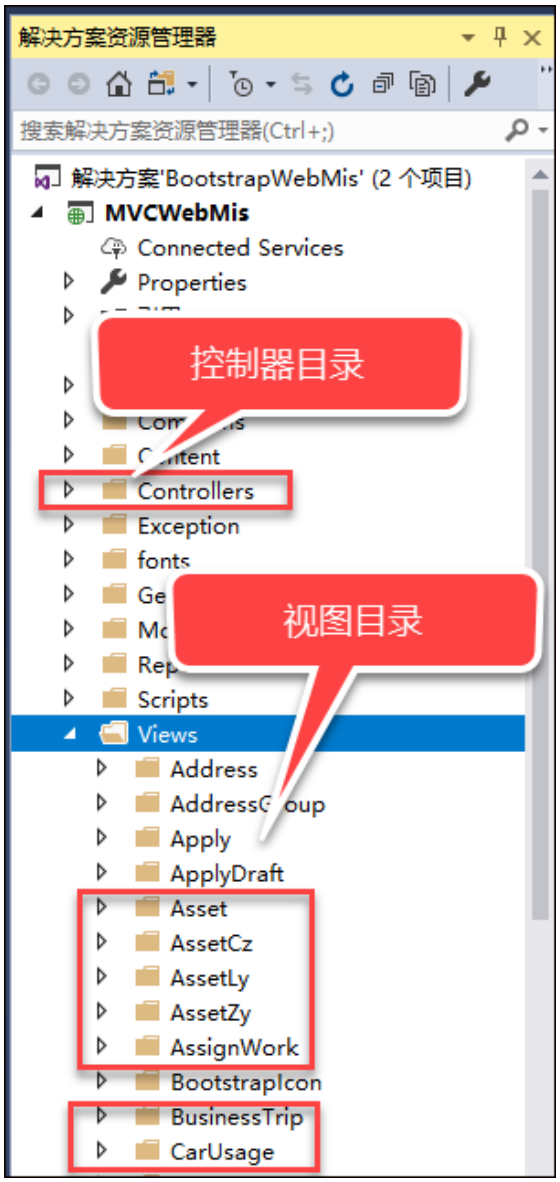


上面各个目录是对应我们业务表的内容，目录下面是有几个工作流模块中包括了列表界面，创建和编辑申请单界面，查看申请单明细这几个界面。

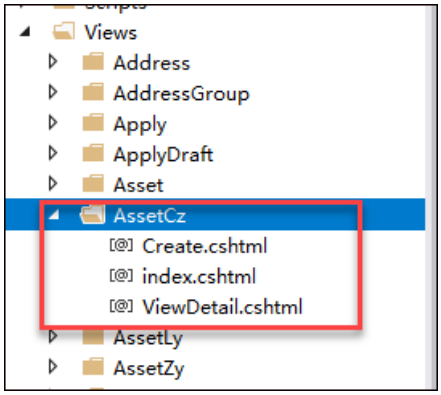


6.1.3 在项目中集成工作流界面代码

这几个工作流界面我们连同他们的目录一同复制到项目的视图目录里面即可，同时把常规 Bootstrap 界面中控制器复制到项目的控制器目录即可。



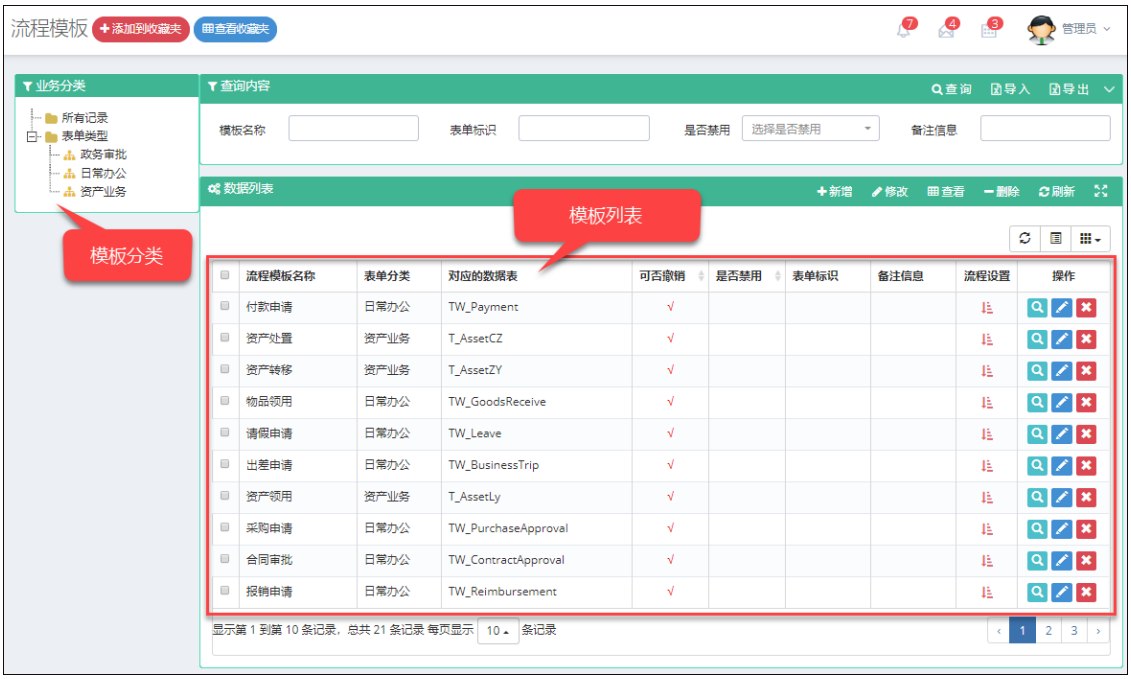
上面红框中就是我们一些工作流业务表单的视图目录，因此我们需要看看目录下面的几个文件。



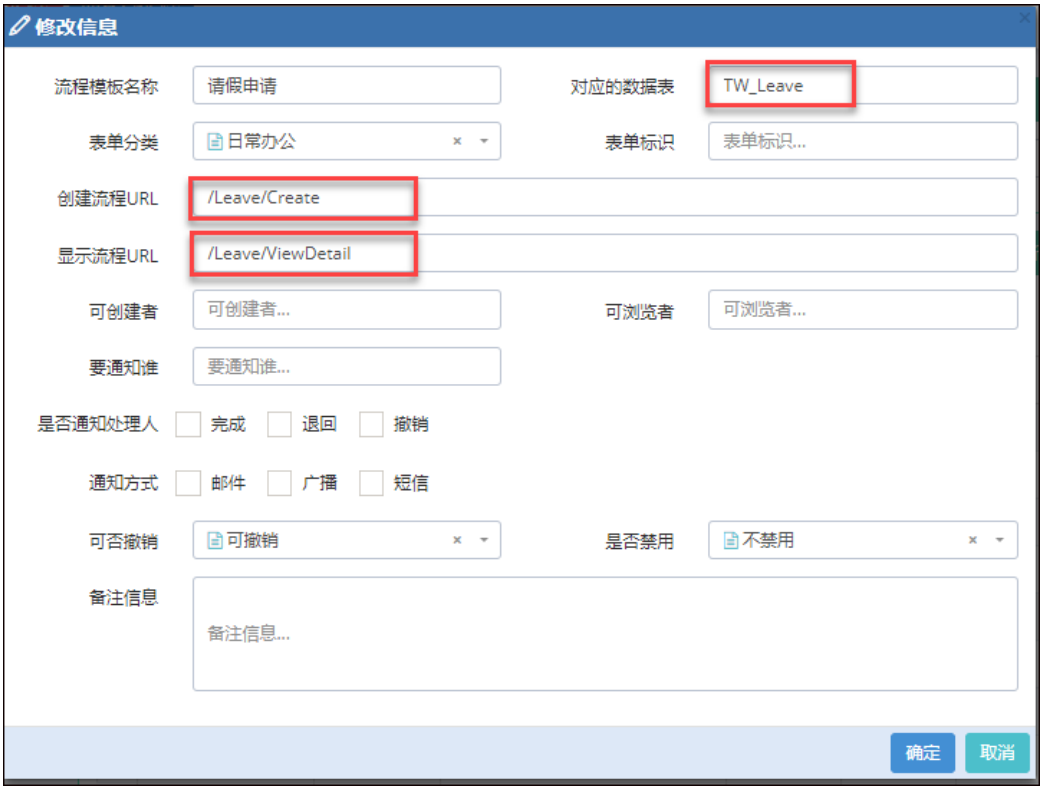
集成这些页面代码后，我们还需要做一些基础的处理才能使用起来，就是需要定义一个业务表单信息

6.1.3.1 流程模板定义

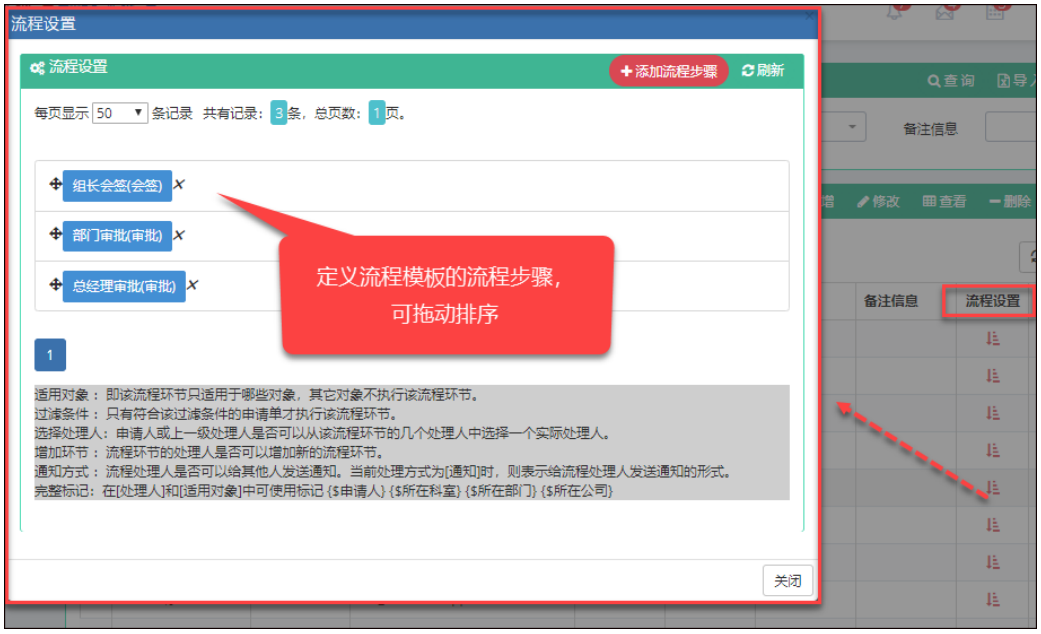
流程模板是我们开展一些工作流的基础，也就是说，我们先有特定流程的模板，然后才有具体的流程示例。



流程模板需要指定它的具体名称，另外有几个字段是必须注意的，就是它的对应业务表名和创建流程 URL、查看流程 URL 这几个信息。



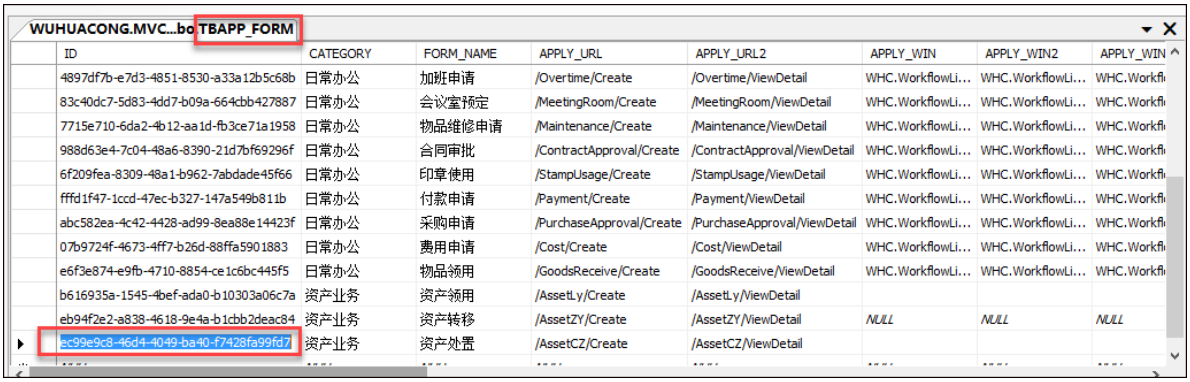
定义流程模板基本信息后，我们需要为这个流程模板设置对应的步骤，如下所示是增加一些流程步骤。



6.1.3.2 修改列表界面的表单 ID

定义一个新的流程模板后，由于我们在流程管理界面中需要创建对应的申请单，那么我们需要知道这个流程模板的表单 ID，因此需要在上面生成的工作流 index.cshtml 页面里面修改一个表单 ID

创建定义完毕流程模板后，我们打开对应的表单记录，找到对应的表单 ID



WUHUACONG.MVC...bo TBAPP_FORM

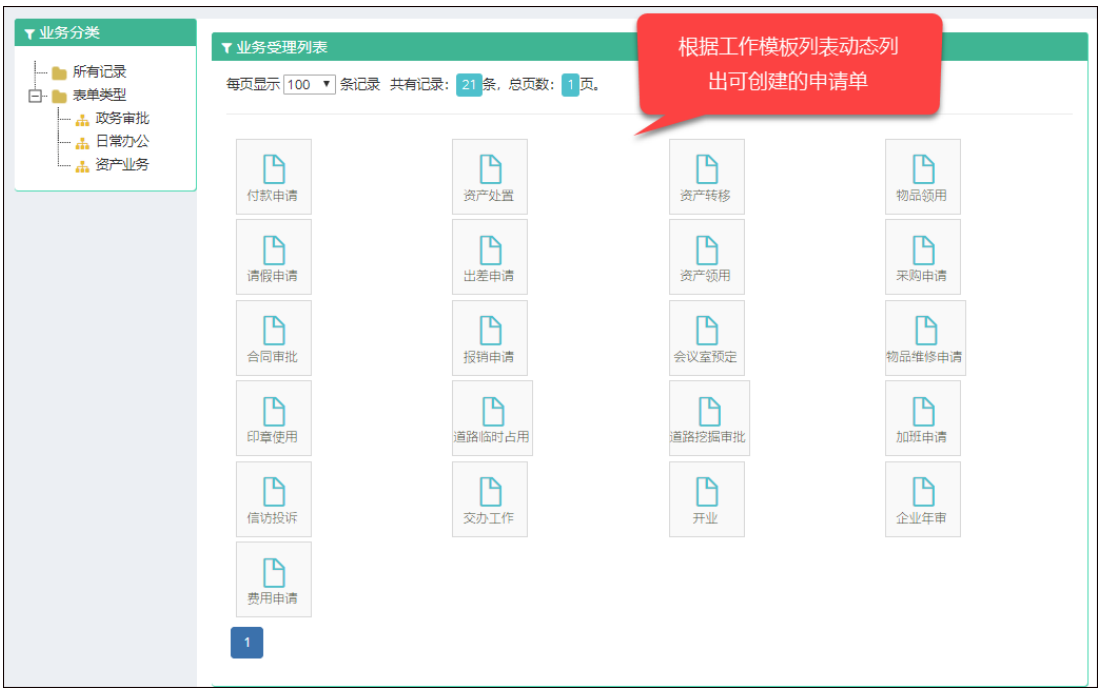
ID	CATEGORY	FORM_NAME	APPLY_URL	APPLY_URL2	APPLY_WIN	APPLY_WIN2	APPLY_WIN
4897df7b-e7d3-4851-8530-a33a12b5c68b	日常办公	加班申请	/Overtime/Create	/Overtime/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
83c40dc7-5d83-4dd7-b09a-664cbb427887	日常办公	会议室预定	/MeetingRoom/Create	/MeetingRoom/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
7715e710-6da2-4b12-aa1d-fb3ce71a1958	日常办公	物品维修申请	/Maintenance/Create	/Maintenance/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
988d63e4-7c04-48a6-8390-21d7bf69296f	日常办公	合同审批	/ContractApproval/Create	/ContractApproval/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
6f209fea-8309-48a1-b962-7abdade45f66	日常办公	印章使用	/StampUsage/Create	/StampUsage/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
fffd1f47-1ccd-47ec-b327-147a549b811b	日常办公	付款申请	/Payment/Create	/Payment/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
abc582ea-4c42-4428-ad99-8ea88e14423f	日常办公	采购申请	/PurchaseApproval/Create	/PurchaseApproval/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
07b9724f-4673-4ff7-b26d-88ffa5901883	日常办公	费用申请	/Cost/Create	/Cost/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
e6f3e874-e9fb-4710-8854-ce1c6bc445f5	日常办公	物品领用	/GoodsReceive/Create	/GoodsReceive/ViewDetail	WHC.WorkflowLi...	WHC.WorkflowLi...	WHC.Workfl
b616935a-1545-4bef-ada0-b10303a06c7a	资产业务	资产领用	/AssetLy/Create	/AssetLy/ViewDetail			
eb94f2e2-a838-4618-9e4a-b1cbb2deac84	资产业务	资产转移	/AssetZY/Create	/AssetZY/ViewDetail	NULL	NULL	NULL
ec99e9c8-46d4-4049-ba40-f7428fa99fd7	资产业务	资产处置	/AssetCZ/Create	/AssetCZ/ViewDetail			

然后修改对应列表界面的 formId 为这个流程模板 ID 即可。

```
index.cshtml
359
360        var formId = 'ec99e9c8-46d4-4049-ba40-f7428fa99fd7';//硬编码指定你的表单ID
361
362        if(formId == ''){
363            showTips("请设置表单ID:formId");
364            return;
365        }
366        //定位到创建页面
367        url = '/AssetCz/Create?formid=' + formId;
368        window.location.href = url;
369    }
```

至此，这样整个界面就可以跑起来，而且也可以在列表页面里面直接创建对应表单的流程，类似下面的创建申请单界面。

创建业务申请单，那么也可以在业务受理列表里面创建。

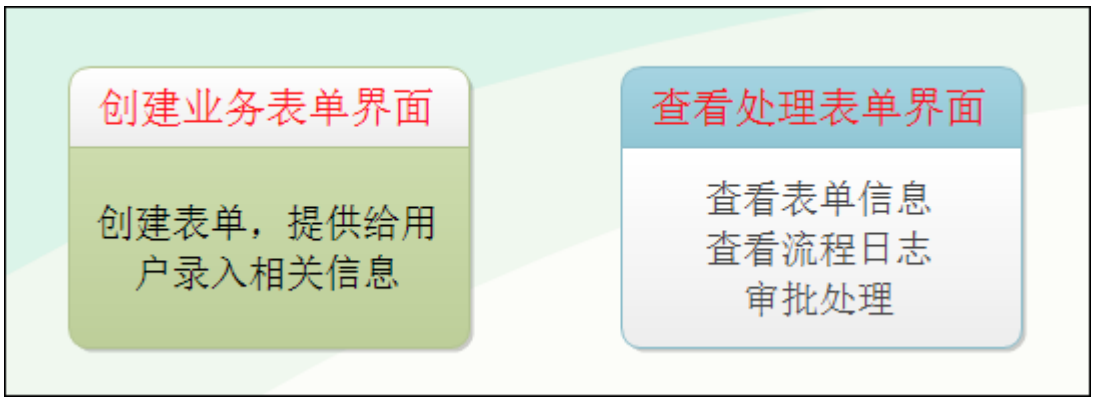


6.2 Winform 工作流界面代码生成

在我们开发工作流的时候，往往需要设计到具体业务表单信息的编辑，有些是采用动态编辑的，有些则是在开发过程中处理的，各有各的优点，动态编辑的则方便维护各种各样的表单，但是数据的绑定及处理则比较麻烦，而自定义开发的，则数据弹性很大，方便修改调整。本篇随笔基于表单的开发设计过程，介绍在工作流中如何新增一个业务表单，以便快速的实现审批业务的上线处理。

6.2.1 业务表单的基类继承

首先我们来了解一下业务表单的对应关系，一般创建一个业务流程处理，都需要有一个具体的创建业务表单的界面，以及一个查看处理表单的界面。



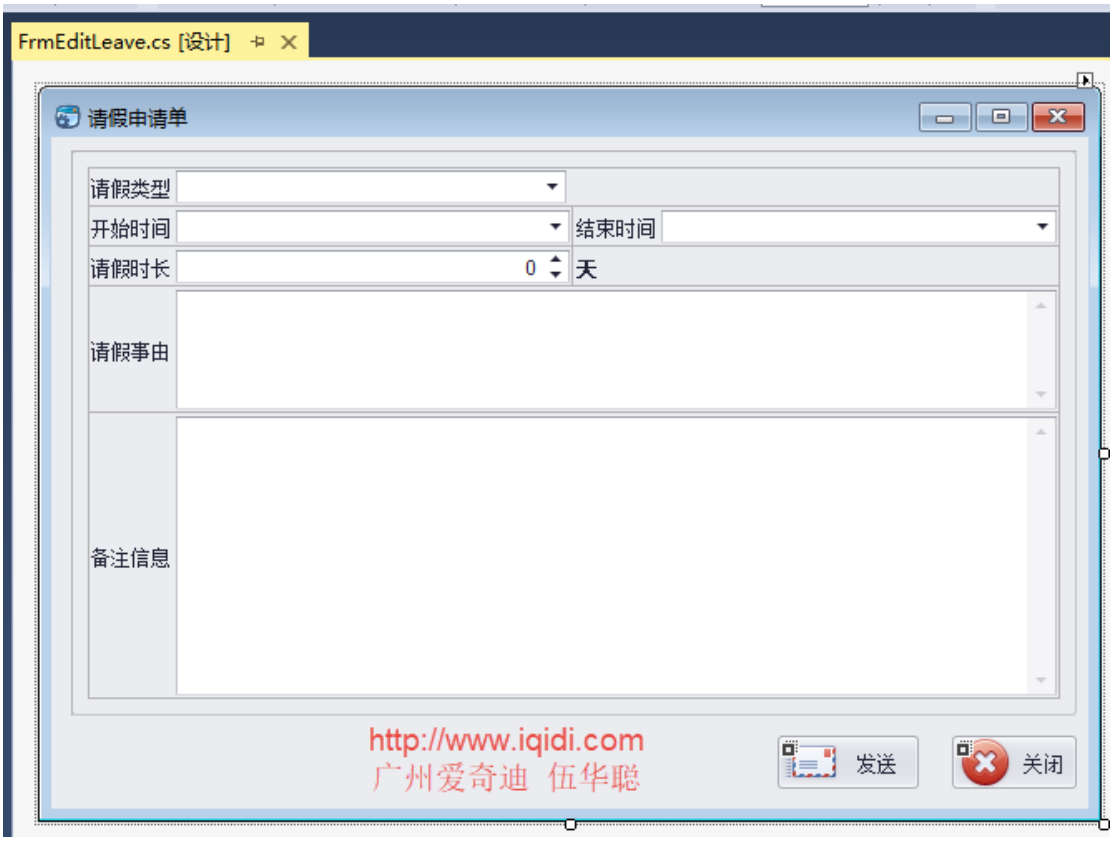
为了方便，我们尽可能减少代码编写，我们需要把大多数的逻辑处理放在基类实现，这样我们在新增一个业务表单的时候就可以减少很多代码编写及维护了。

如对于 FrmAddApply 类定义如下，我们定义一些抽象接口用于下面的业务表单实现

```
/// <summary>
/// 创建申请单的窗体基类
/// </summary>
public partial class FrmAddApply : BaseForm
{
    /// <summary>
    /// 表单ID
    /// </summary>
    public string FormID { get; set; }

    /// <summary>
    /// 申请单ID
    /// </summary>
    public string ApplyId { get; set; }

    public FrmAddApply()
    {
        InitializeComponent();
    }

    /// <summary>
    /// 显示数据的函数(子类必须实现)
    /// </summary>
    public virtual void DisplayData() { }

    /// <summary>
    /// 实现控件输入检查的函数(子类必须实现)
    /// </summary>
    /// <returns></returns>
    public virtual bool CheckInput() { return true; }

    /// <summary>
    /// 编辑状态下的数据保存(子类必须实现)
    /// </summary>
    /// <returns></returns>
    public virtual bool SaveUpdated() { return true; }

    /// <summary>
    /// 新增状态下的数据保存(子类必须实现)
    /// </summary>
    /// <returns></returns>
    public virtual bool SaveAddNew() { return true; }

    ..................
```

这样我们创建一个新的业务表单，只需要利用代码生成工具，生成所需要的各层框架代码，然后再生成 Winform 窗体代码，复制部分界面处理代码过来这个业务表单的子类即可。

下面是一个请假申请的业务表单设计，如下所示。



我们看到这个表单可以使用代码生成工具 Database2Sharp 快速生成后进行一定调整的，而这个编辑表单的界面，我们只需要使用自动生成的部分代码即可。

相关代码只需要复制上面的新增、更新、显示数据的代码即可。

```
/// <summary>
/// 请假申请单的编辑界面
/// </summary>
7 个引用
public partial class FrmEditLeave : FrmAddApply
{
    /// <summary>
    /// 创建一个临时对象，方便在附件管理中获取存在的GUID
    /// </summary>
    private LeaveInfo tempInfo = new LeaveInfo();

    2 个引用
    public FrmEditLeave()
    {
        InitializeComponent();
    }

    /// <summary>
    /// 实现控件输入检查的函数
    /// </summary>
    /// <returns></returns>
    4 个引用
    public override bool CheckInput()...

    /// <summary>
    /// 初始化数据字典
    /// </summary>
    1 个引用
    private void InitDictItem()...

    /// <summary>
    /// 更新界面控件数据显示
    /// </summary>
    4 个引用
    public override void DisplayData()...
```

对于查看申请单的基类 FrmViewApply 类，我们更加简单，我们需要把它的自定义界面控件加载出来即可。

下面是查看申请单的基类，封装了相关的处理逻辑。

```
/// <summary>
/// 本窗体是通用的查看申请单界面基类。
/// 为减少开发相关页面的工作量，只需要创建一个新窗体，并继承本窗体，然后在子窗体Form_Load函数里面，初始化对应的申请单显示控件即可。
/// </summary>
public partial class FrmViewApply : BaseDock
{
    /// <summary>
    /// 申请单ID
    /// </summary>
    public string ApplyId { get; set; }

    /// <summary>
    /// 申请单自定义控件
    /// </summary>
    public BaseUserControl ApplyControl { get; set; }

    /// <summary>
    /// 默认构造函数
    /// </summary>
    public FrmViewApply()
    {
        InitializeComponent();
    }

    private void FrmViewApply_Load(object sender, EventArgs e)
    {
        if (!this.DesignMode)
        {
            InitToolBar();
        }
    }

    /// <summary>
    /// 初始化申请单控件
    /// </summary>
    protected virtual void InitApplyControl(BaseUserControl control)
    {
        if (control != null)
        {
            this.ApplyControl = control;
            this.ApplyControl.Dock = DockStyle.Fill;
            this.Controls.Add(control);
        }
```

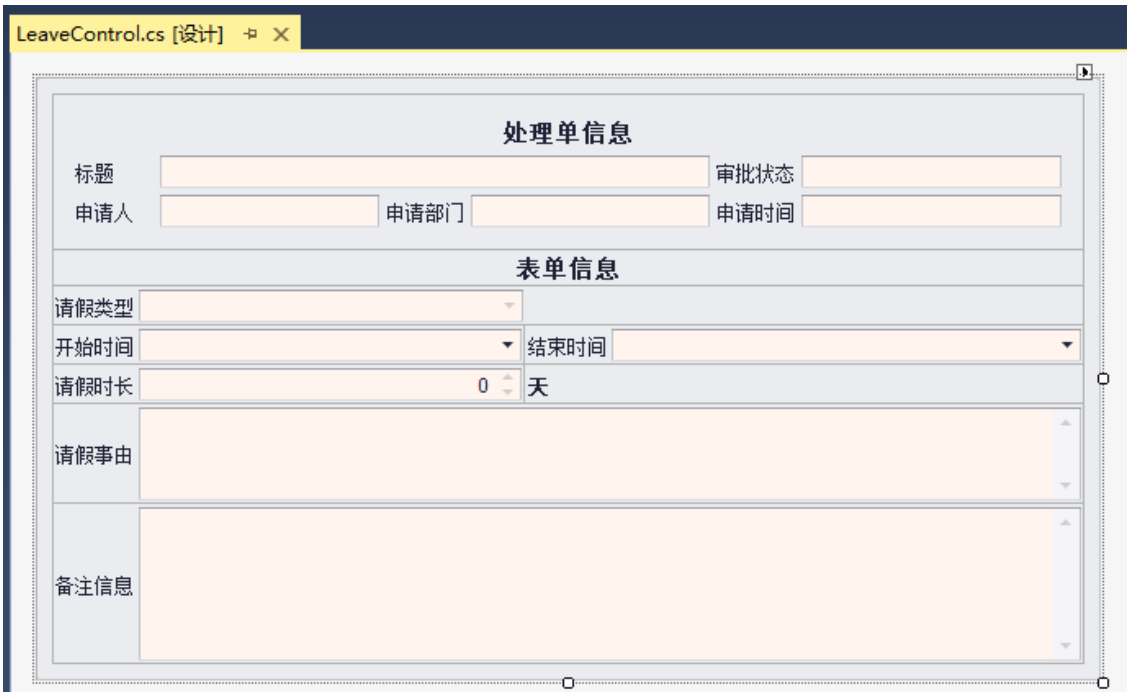
```
        /// <summary>
        /// 打印申请单控件内容(默认调用窗体打印)
        /// </summary>
        protected virtual void PrintApplyControl()
        {
            if(this.ApplyControl != null)
            {
                PrintFormHelper.Print(this.ApplyControl, false);
            }
        }

        /// <summary>
        /// 表单另存为
        /// </summary>
        protected virtual void ApplySaveAs()
        {
        }

        /// <summary>
        /// 初始化工具栏的按钮和状态
        /// </summary>
        protected virtual void InitToolBar()
        {
               ................//基类实现，控制什么时候该做什么审批处理，以及一些常见按钮
        }

    ................
```

查看请假申请单的窗口就是继承这个 FrmViewApply 即可，如下所示。

```
        /// <summary>
        /// 查看请假申请单的窗体
        /// </summary>
        public partial class FrmViewLeave : FrmViewApply
        {
            private LeaveControl control = null;

            public FrmViewLeave()
            {
                InitializeComponent();
            }

            private void FrmViewLeave_Load(object sender, EventArgs e)
            {
                //初始化控件并展示在基类窗体里面
                control = new LeaveControl();
                control.ApplyId = this.ApplyId;
                control.DisplayData();

                base.InitApplyControl(control);
            }
        }
```

这个就是全部的窗体源码了，主要的内容我们看到是在 LeaveControl 这个用户控件类里面的了，

而这个控件主要就是上面编辑请假申请单的界面设计，并复制相关的显示数据代码即可。

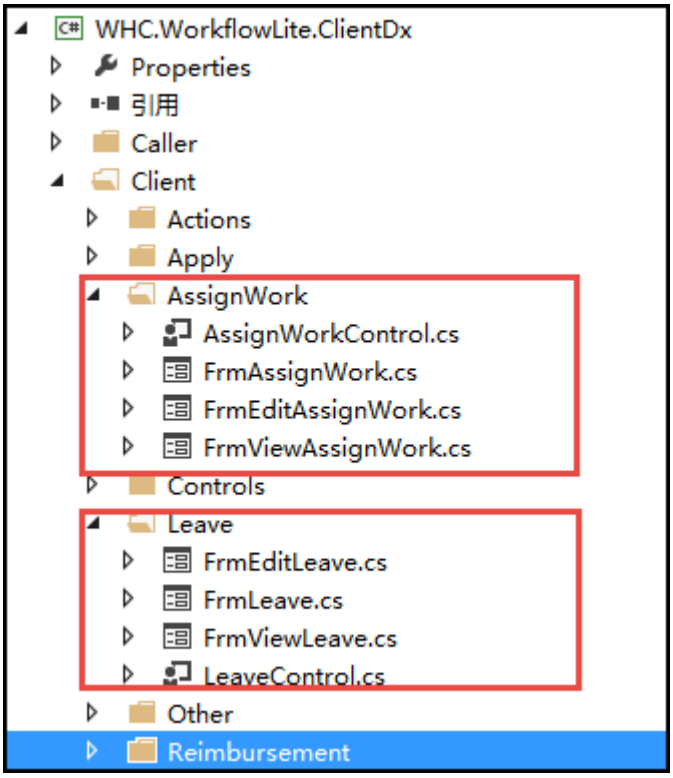


相关界面代码如下所示。

```
/// <summary>
/// 查看请假申请单的内容显示控件
/// </summary>
public partial class LeaveControl : BaseUserControl
{
    /// <summary>
    /// 申请单ID
    /// </summary>
    public string ApplyId { get; set; }

    public LeaveControl()
    {
        InitializeComponent();

        SetReadOnly();
    }

    /// <summary>
    /// 设置整个窗体布局为只读并设置只读的背景颜色
    /// </summary>
    private void SetReadOnly()
    {
        this.layoutControl1.OptionsView.IsReadOnly = DevExpress.Utils.DefaultBoolean.True;
        this.layoutControl1.Appearance.ControlReadOnly.BackColor = Color.SeaShell;
    }

    private void LeaveControl_Load(object sender, EventArgs e)
    {
        this.applyInfoControl1.ApplyId = this.ApplyId;
        this.applyInfoControl1.BindData();
    }

    /// <summary>
    /// 初始化数据字典
    /// </summary>
    private void InitDictItem()
    {
        //初始化代码
    }

    /// <summary>
    /// 数据显示的函数
    /// </summary>
    public void DisplayData()
    {
        InitDictItem();//数据字典加载（公用）
```

通过上面定义的对应表单的窗体基类，可以减少我们重复编码的需要，我们只需要利用最有效率的生成界面，然后复制代码后调整即可快速生成我们所需要的不同表单界面。

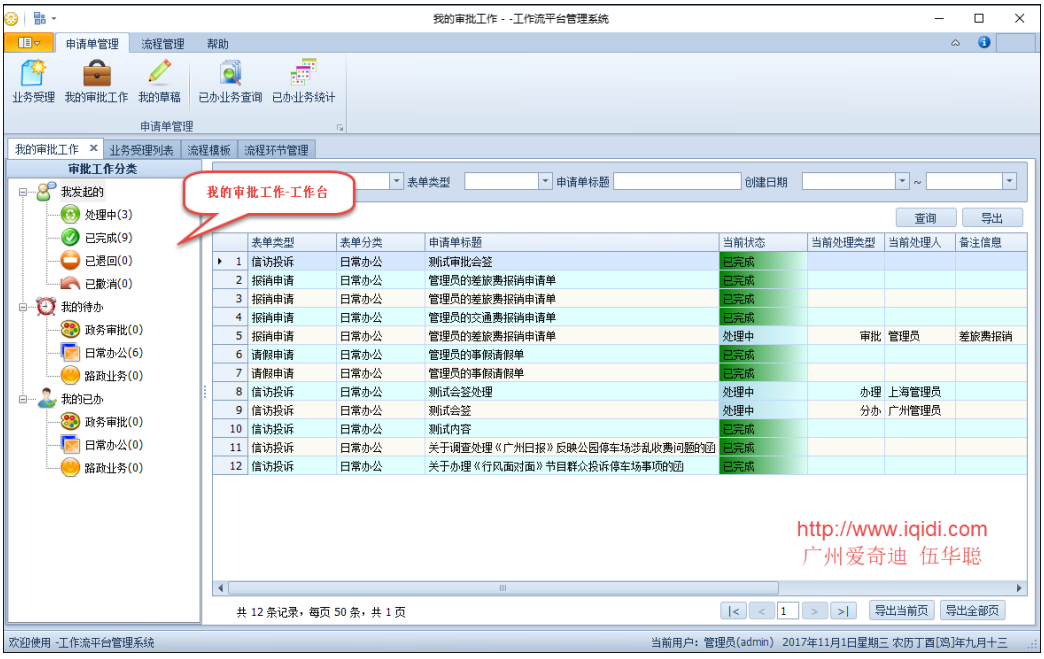
每个表单我们放在一个目录上，这样我们就可以很好管理它们了。



6.2.2 业务表单的动态展示处理

上面介绍了业务表单的填写、查看两个不同的窗口，我们在申请单的审批界面里面，统一显示不同的表单，以及创建不同的业务表单界面，这种动态的处理可以实现不同业务表单的创建及显示界面。

如我的审批工作中，表单的显示界面如下所示，查看具体表单后，可以动态展示不同的业务窗口界面。



另外我们在创建业务表单的时候，根据数据库的配置信息，动态展示所有可以展示的创建入口，单击相关的按钮，可以动态调用创建对应的表单界面。创建流程业务表单的入口如下所示。

在我的审批工作界面，动态创建对应的查看表单窗体代码如下所示。

```
/// <summary>
/// 分页控件编辑项操作
/// </summary>
private void winGridViewPager1_OnEditSelected(object sender, EventArgs e)
{
    //获取记录ID和表单ID
    string ID = this.winGridViewPager1.gridView1.GetFocusedRowCellDisplayText("ID");
    string FormId = string.Concat(this.winGridViewPager1.gridView1.GetFocusedRowCellValue("FormId"));

    if (!string.IsNullOrEmpty(ID) && !string.IsNullOrEmpty(FormId))
    {
        var formInfo = BLLFactory<BLL.Form>.Instance.FindByID(FormId);
        if (formInfo != null && !string.IsNullOrEmpty(formInfo.ApplyWin2))
        {
            try
            {
                //根据配置的查看窗体，动态构建查看申请单对象
                FrmViewApply dlg = Assembly.GetExecutingAssembly().CreateInstance(formInfo.ApplyWin2) as FrmViewApply;
                if (dlg != null)
                {
                    dlg.ApplyId = ID;
                    dlg.OnDataSaved += new EventHandler(dlg_OnDataSaved);

                    if (DialogResult.OK == dlg.ShowDialog())
                    {
                        BindData();
                    }
                }
            }
            catch (Exception ex)
            {
                LogHelper.Error(ex);
                MessageDxUtil.ShowError(ex.Message);
            }
        }
    }
}
```

这个代码替代了需要手动创建不同对象的处理。

```
var dlg = new FrmViewAssignWork();
dlg.ApplyId = ID;
dlg.OnDataSaved += new EventHandler(dlg_OnDataSaved);

if (DialogResult.OK == dlg.ShowDialog())
{
    BindData();
}
```

同理，对于创建编辑界面，我们也可以同样的方法动态创建相关的编辑表单界面，如下代码所示。

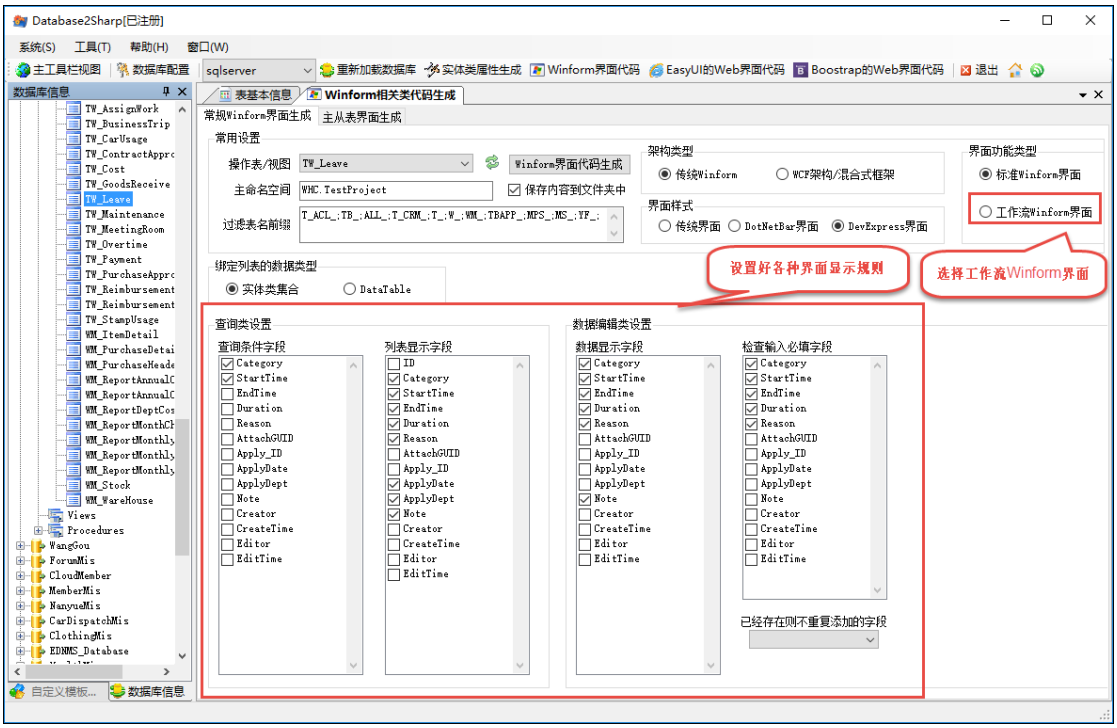
```
/// <summary>
/// 单击某个动态生成的按钮，触发的申请表单创建界面
/// </summary>
1 个引用
void button_Click(object sender, EventArgs e)
{
    SimpleButton button = sender as SimpleButton;
    if (button != null)
    {
        var formId = button.Tag.ToString();
        var formInfo = BLLFactory<BLL.Form>.Instance.FindByID(formId);
        if (formInfo != null && !string.IsNullOrEmpty(formInfo.ApplyWin))
        {
            try
            {
                //动态构建创建申请单的界面窗体并赋值
                var dlg = Assembly.GetExecutingAssembly().CreateInstance(formInfo.ApplyWin) as FrmAddApply;
                dlg.FormID = button.Tag.ToString();
                dlg.ShowDialog();
            }
            catch(Exception ex)
            {
                LogHelper.Error(ex);
                MessageDxUtil.ShowError(ex.Message);
            }
        }
        else
        {
            MessageDxUtil.ShowTips(button.Text + "暂未开通");
        }
```


6.2.3 工作流业务界面的代码生成

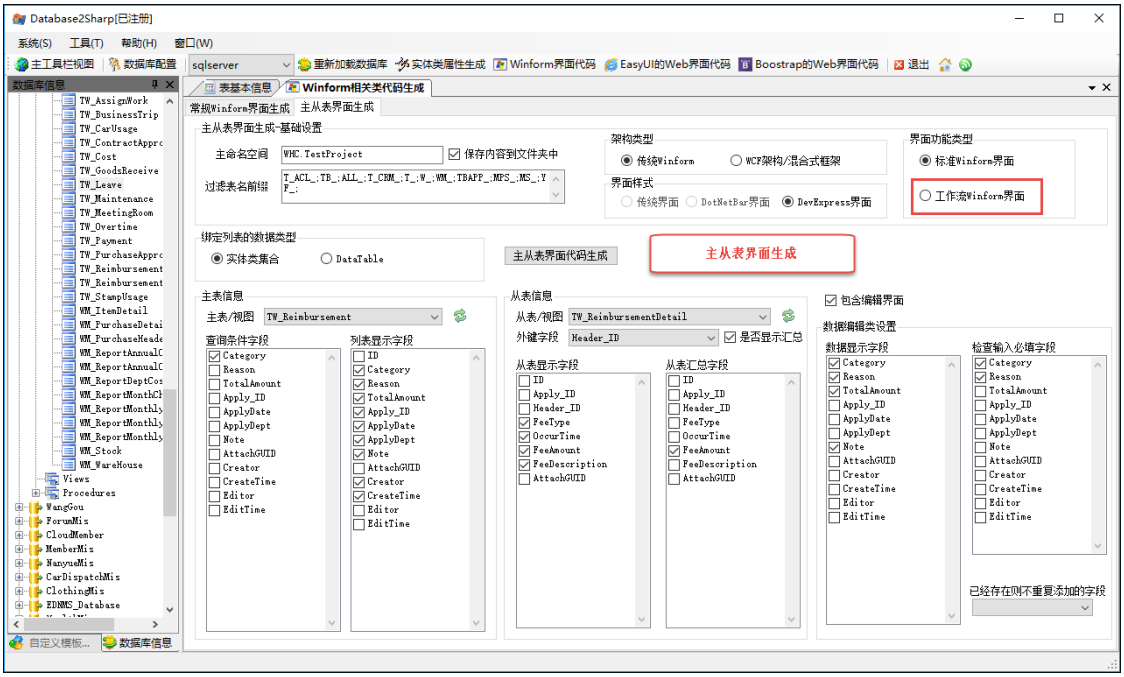
从上面我们可以看到，其中对于工作流业务表单的窗体界面都可以实现标准的处理了，继承自某个基类，然后整合相关的数据处理规则即可。

那么我们提炼业务信息后，可以使用代码生成工具快速生成，这样可以极大提高我们的开发效率。

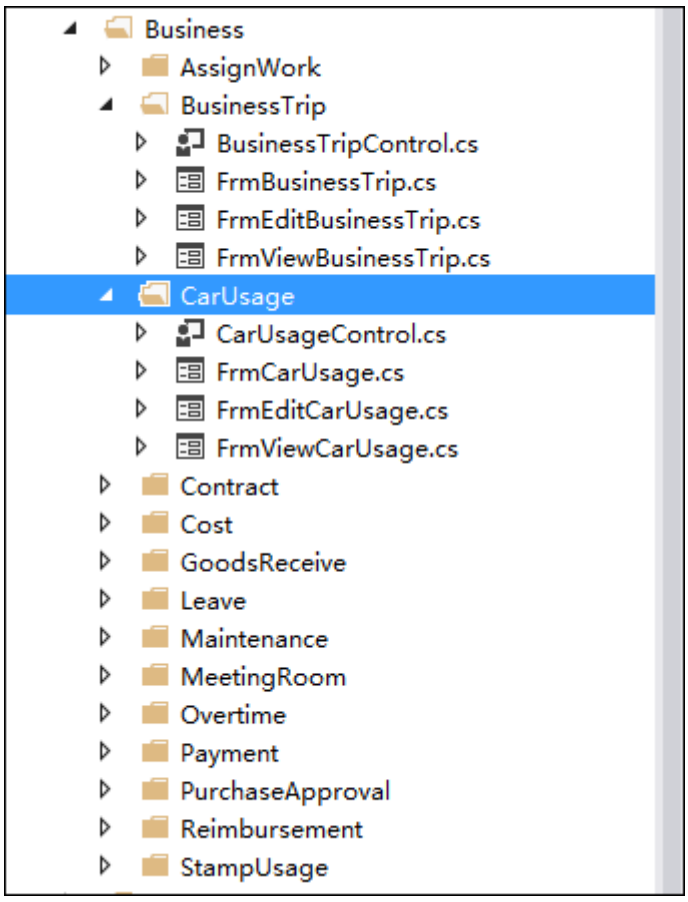
针对上面的构想，我们花费了好几天的时间，创建了工作流界面的自动生成规则和反复校验，最终整合到代码生成工具中方便开发。



对于主从表表的界面，我们依旧也可以使用代码生成工具进行快速的工作流界面生成。



根据代码生成工具生成相应的业务表单，在项目中展示如下所示，其中每个业务表单包括查看控件、申请单查询界面、新增申请单界面、查看申请单界面。

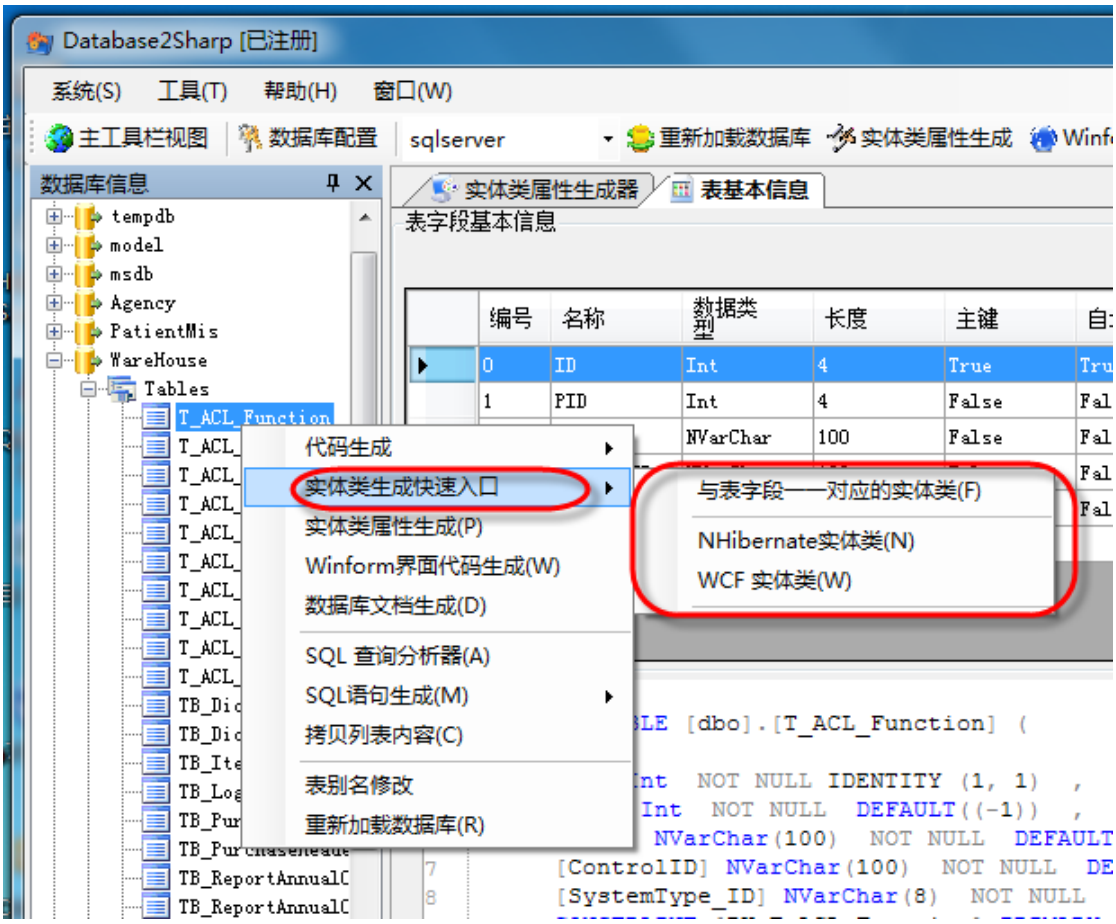


7 其他辅助功能

7.1 实体类快速生成

7.1.1 基于数据库的实体类快速生成

有时候，仅仅是一些特殊的需要，需要生成相应的实体类信息，供使用或者参考，那么我们在具体的表名上，右键菜单【实体类生成快速入口】会列出三种不同的实体类生成：与表字段一一对应的实体类、Nhibernate 实体类、WCF 实体类，如下所示。

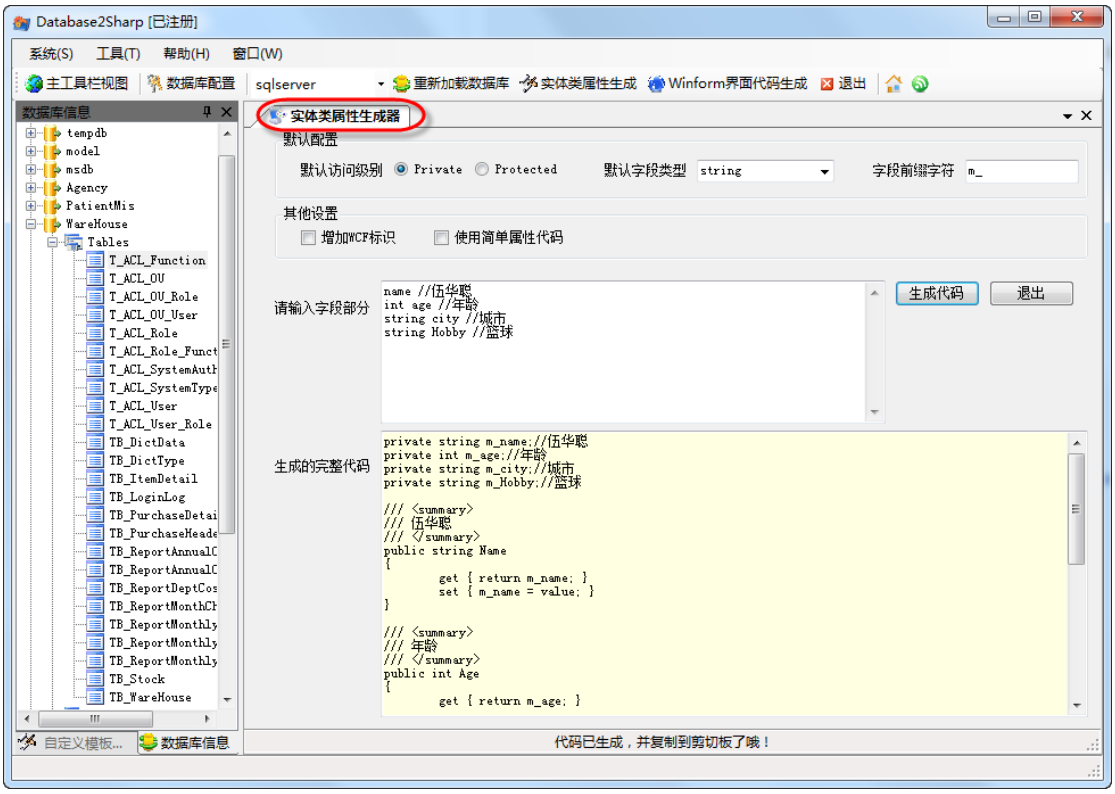


图表 7-1 实体类生成快速入口

7.1.2 实体类属性快速生成

做了很久的代码生成工具，基本上都是基于表生成实体类属性的，把数据库表的信息拿出来，然后之乎者也后生成一个标准的实体类，包含字段、属性、描述等东西。

不过后来在做一些非数据库的项目的实体类，还有一些不是基于表一一对应关系的实体类，写这些字段属性的代码就显得比较乏味，那么是否有可能通过输入一些简单的信息，就能构造出一个符合代码标准的实体类信息呢，技术就是为了方便而诞生的，因此虽然 Database2Sharp 是基于整个数据库生成整个框架代码的工具，我也整合了自定义实体类的生成功能，以求方便我们实际的开发工作。



图表 7-2 实体类属性快速生成

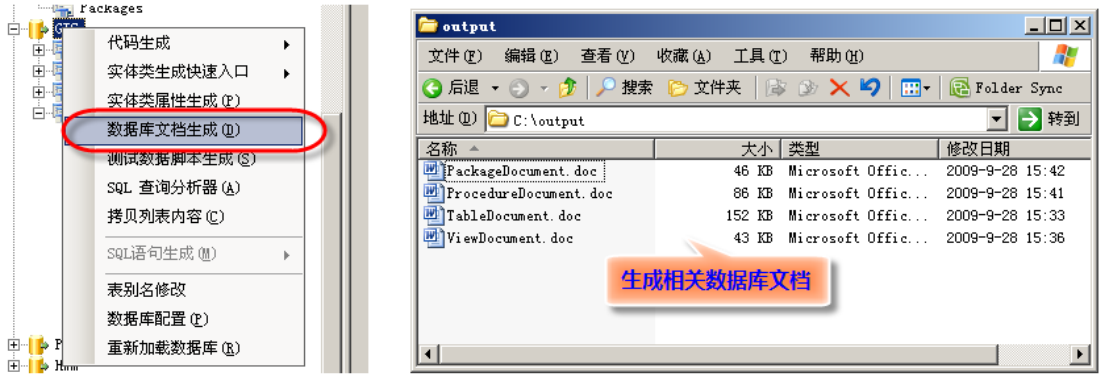
7.2 数据库文档生成

7.2.1 数据库文档生成

很多时候，项目需要提供一份数据库设计文档，如果表很多，每次需要更新这个文档，就是一项非常复杂、非常麻烦的事情，虽然技术含量不高，但是很折腾人，而且也很耗费尽力。

如果能有工具自动生成符合《数据库设计说明书》格式的文档，那么肯定为你节省很多宝贵的时间，你可以在进度上安排自己编写《数据库设计说明书》的时间，而利用这些时间做一些有意义的事情，或者甚至是休闲，是不是很好呢。

本代码生成工具生成的数据库设计文档，包括数据库表设计文档、存储过程及视图的设计文档，很方便，效果如下所示。

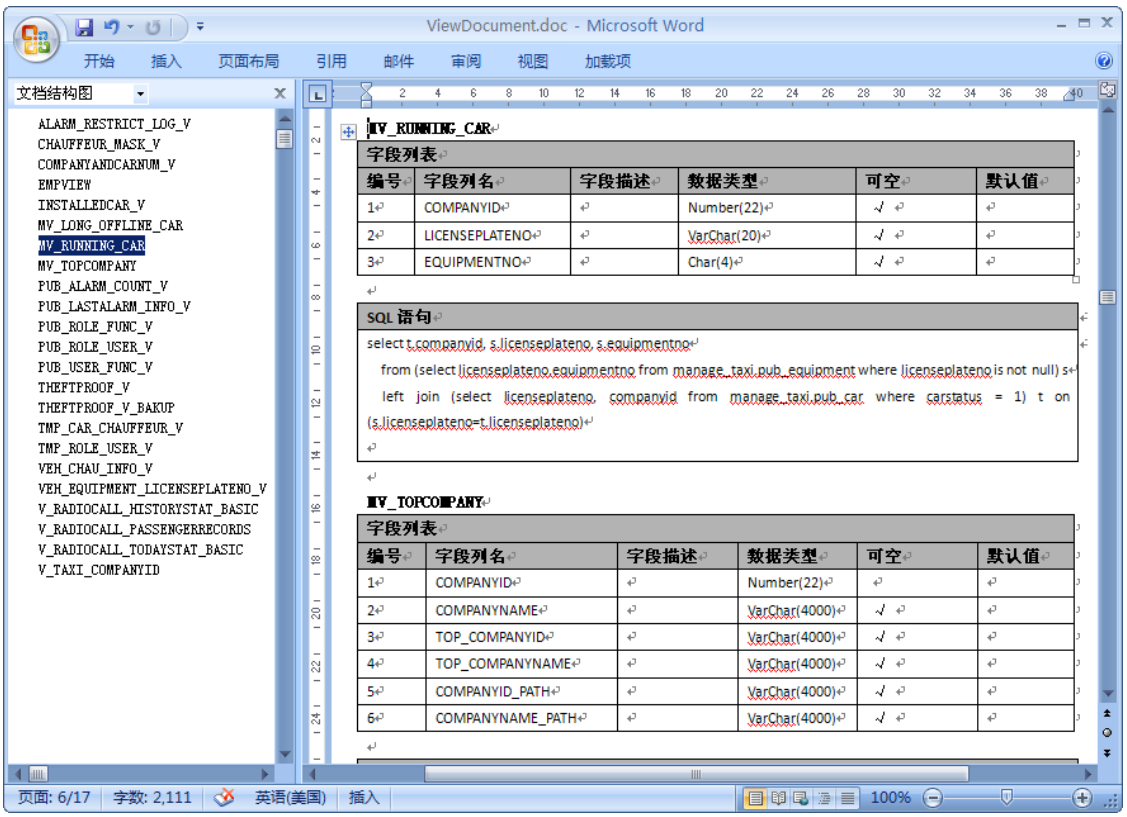


图表 7-3 数据库文档生成操作

表数据的文档格式如下所示：

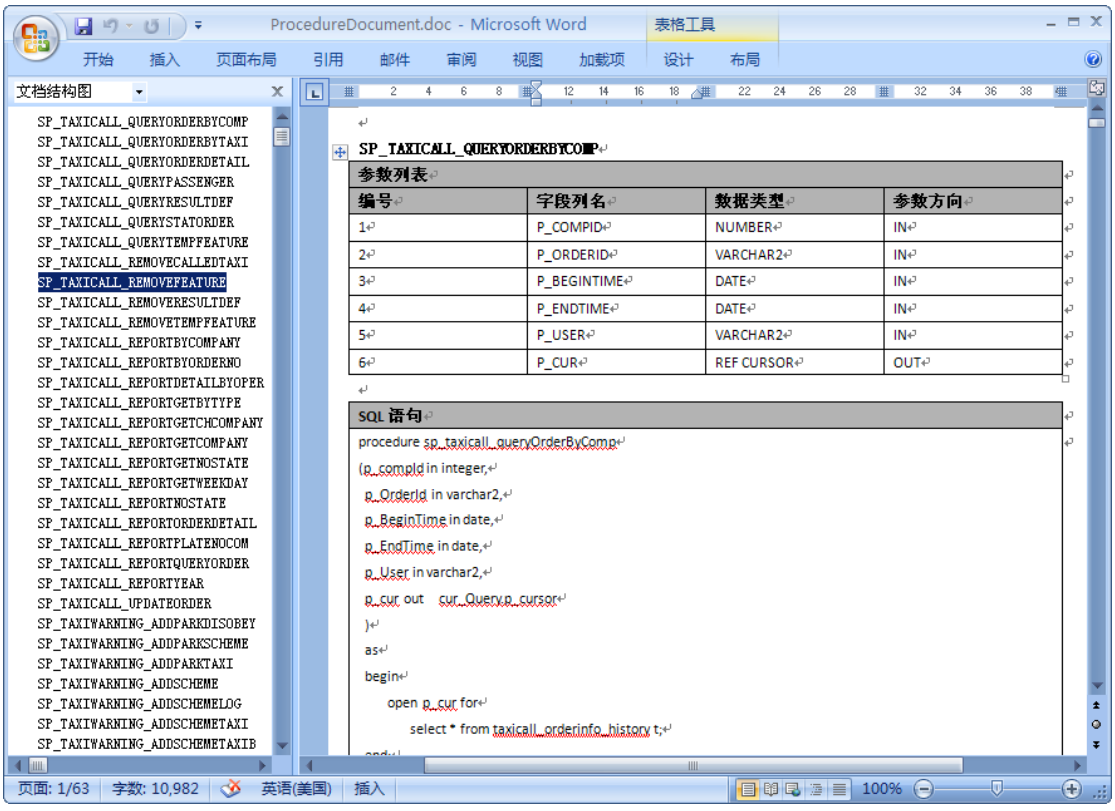
图表 7-4 数据库表文档效果

视图数据的文档格式如下所示：



图表 7-5 数据库视图文档效果

存储过程的文档格式如下所示：



图表 7-6 数据库存储过程文档效果

包数据的文档格式如下所示：

图表 7-7 Oracle 数据库包文档效果

7.3 自定义模板代码生成

7.3.1 自定义模板语法介绍

代码生成工具一直是很多从事开发人员的必备，一般开发人员都会选择一款高效、适合自己开发模式的代码生成工具，我也不例外，我一直让我们的Database2Sharp(http://www.iqidi.com/database2sharp.htm)代码生成工具围绕我们的开发框架来生成代码，至今已经经过快 7 个年头的洗礼，主要是提供高效、快速的一键生成整个项目框架源码的操作。

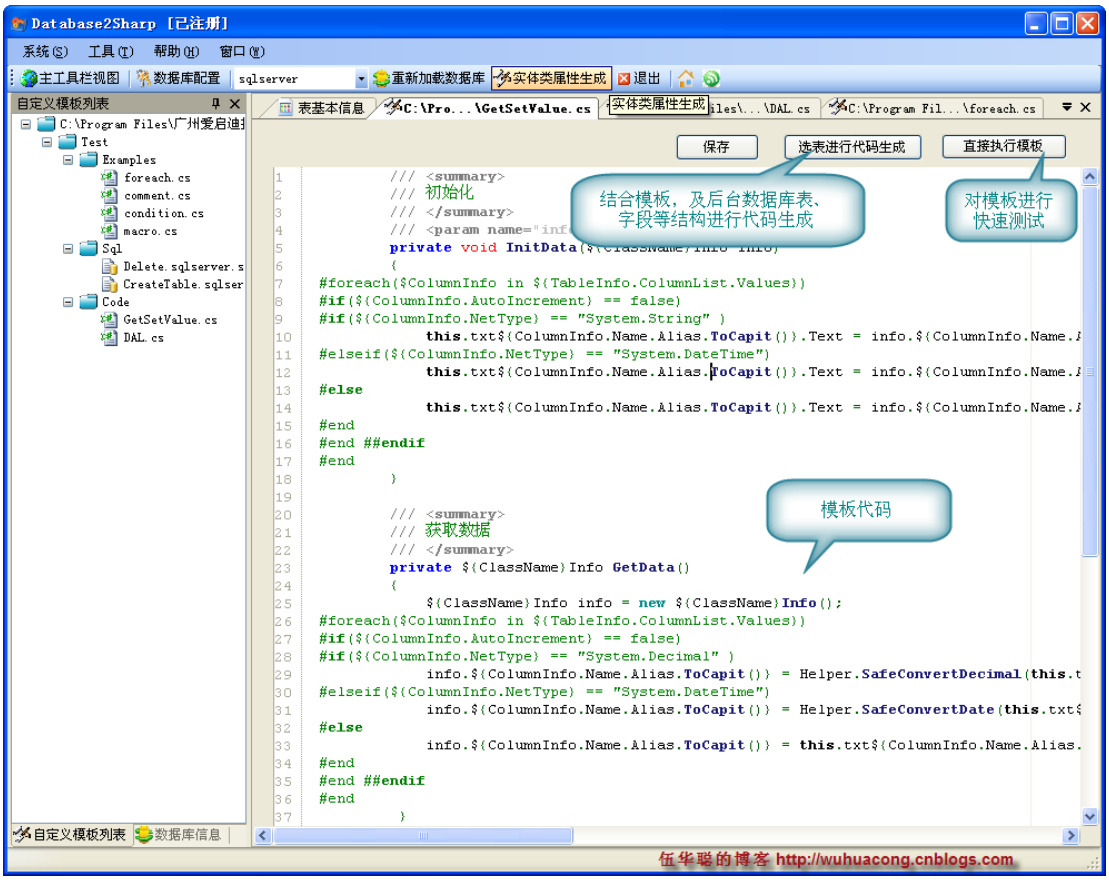
代码生成工具 Database2Sharp 自一开始，就采用基于模板方式的代码生成方式，这样提高生成效率，并且减少维护成本（相对某些硬编码代码生成的方式更优），方便用户对模板进行适当的修改等特点，不过虽然是提供了模板修改的功能，但是由于在工具的主界面上，并未提供对自定义模板文件的管理，因此以前的版本一直没有完成用户自定义模板管理维护的功能，本次版本更新就是弥补这一不足，发挥模板应有的魅力和功能。

在实际开发当中，自定义模板生成是很多代码生成工具生成代码的重要补

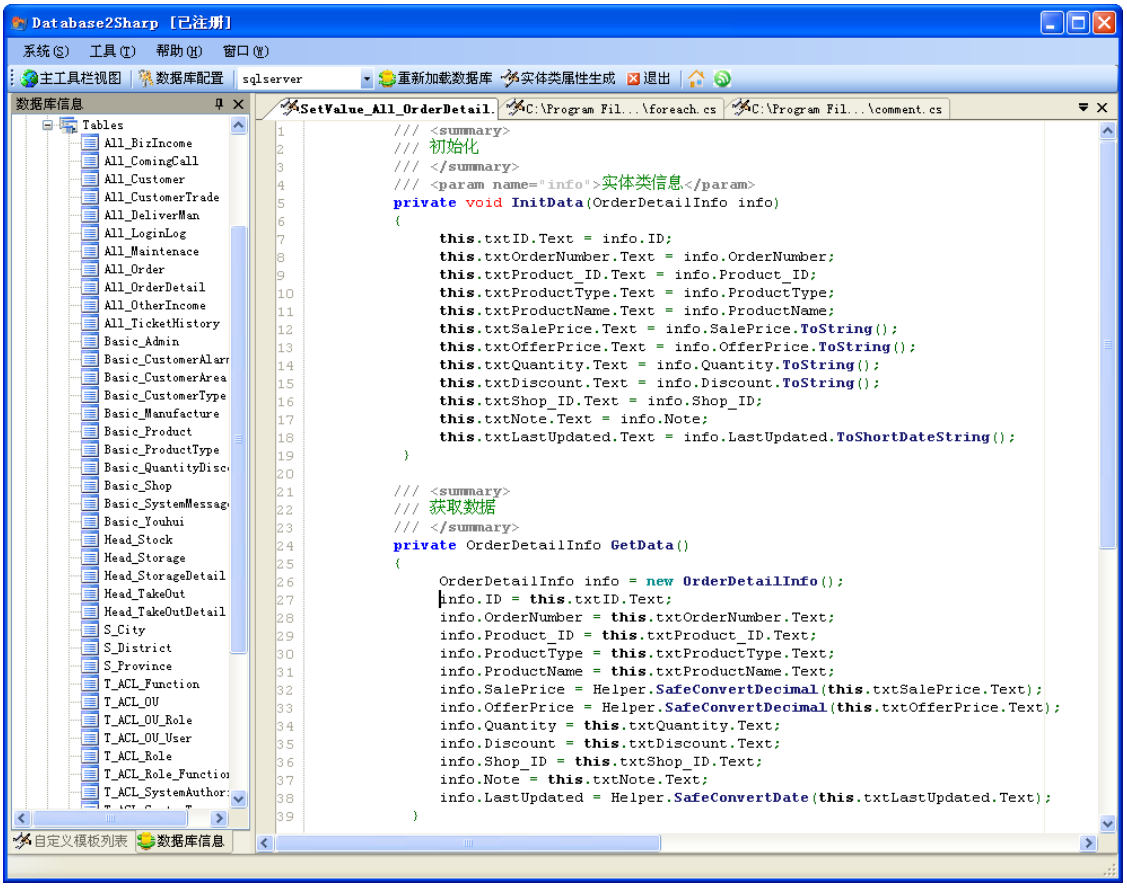
充，例如有些项目代码片段需要（例如控件赋值语句，或者控件数据显示语句等），如果能通过自定义模板方式，利用模板引擎的灵活特点，以及已有数据库的结构信息，就很方便生成重复性强、有一定规律的代码。

因此，本次 Database2Sharp 版本更新，主要就是提供一个对自定义模板管理，方便利用自定义模板灵活、高效的特点，并且利用后台数据库表、字段等信息，为实际的项目代码片段生成服务，本次还调整了整体的软件界面布局，利用成熟的 Weifenluo 布局控件，更好展示多文档的信息。

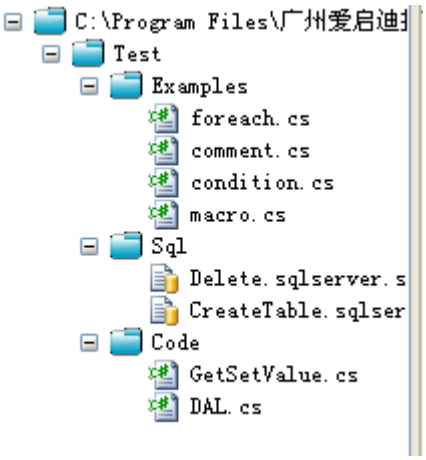
我们来看看自定义代码生成模块的功能是如何实现的，首先提供一个自定义模板列表进行维护，可以通过右键菜单进行添加、修改、重命名、删除等操作，模板代码可以进行【直接执行模板】和【选表进行代码生成】两种方式。



利用【选表进行代码生成】方式，可以很好利用后台的数据库表、字段等信息，结合模板生成高效的代码，如下所示。



另外，为了方便大家对模板引擎 NVelocity 的了解，在软件工具安装的时候，附带了几个常见的例子，如下所示。



例子的代码大致如下所示，主要是让大家快速了解 Nvelocity 的模板语言 VTL 的使用。如果对模板引擎更加深入的了解，可以查看我之前的随笔《强大的模板引擎开源软件 NVelocity》进一步了解。

遍历及注释，赋值语句例子。

```
##Foreach语法操作函数
#set( $criteria = ["name", "address"] )
#foreach( $criterion in $criteria )
    $criterion
#end

##注释操作

注释
单行注释
## This is a single line comment

多行注释
#*
 Thus begins a multi-line comment. Online visitors won't
 see this text because the Velocity Templating Engine will
 ignore it.
*#
```

条件结构的例子

```
##此处为注释说明
##简单例子(主要规则：引用以$开头用于取得什么东西，而指令以# 开始用于做什么事情)
##在VTL中，所有变量标识符的开头要加上$字符，如$Name，也可以用一种更加明确的方法表示，例如${name}。

#set($foo = false)
#if ($foo)
    this is true
#elseif ($bar)
    this is false
#elseif (true)
    this should be followed by two blank lines
#end

## this is a single line comment

#*
this is a multi line comment
#if (
*#


#set($user = "jason")
```

```
#set($login = false)
#set($count = 5)

#if ($user == "jason")
   the user $user is logged in!
#end

#if ($count == 5)
   the count is 5!
#end

#if ($login == false)
   the user isn't logged in.
#end

#if ($count != 3)
   \$count is not equal to 3
#end
```

宏脚本例子

```
## #macro 脚本元素允许模板设计者在VTL 模板中定义重复的段。
## Velocimacros 不管是在复杂还是简单的场合都非常有用。
## 下面这个Velocimacro，仅用来节省击键和减少排版错误，介绍了一些NVelocity宏的概念。
## 可以带参数，参数放在宏名称的后面，空格隔开

#macro( d )
<tr><td></td></tr>
#end

#d()
```

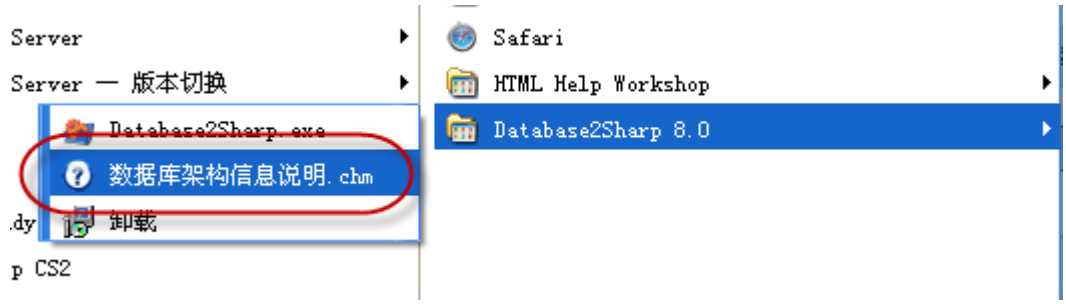
以及一些常用的赋值及数据显示代码片段

```
      /// <summary>
       /// 初始化
       /// </summary>
       /// <param name="info">实体类信息</param>
       private void InitData(${ClassName}Info info)
       {
#foreach($ColumnInfo in ${TableInfo.ColumnList.Values})
#if(${ColumnInfo.AutoIncrement} == false)
```

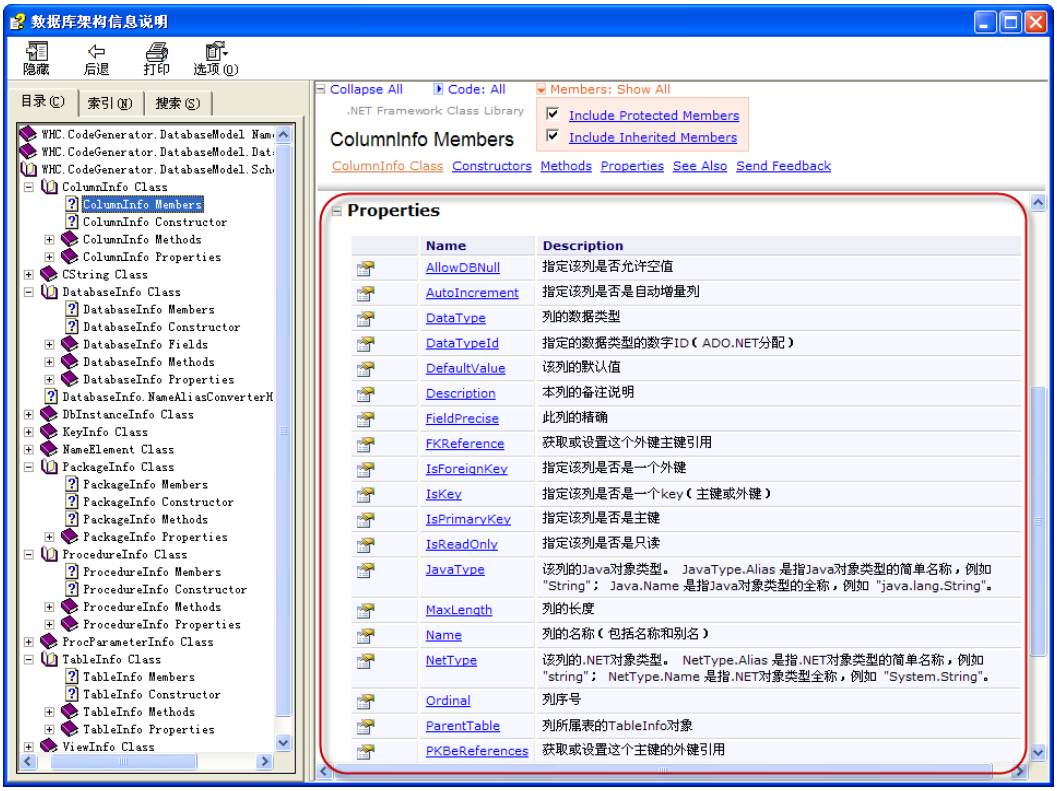
```
#if(${ColumnInfo.NetType} == "System.String" )
          this.txt${ColumnInfo.Name.Alias.ToCapit()}.Text =
info.${ColumnInfo.Name.Alias.ToCapit()};
#elseif(${ColumnInfo.NetType} == "System.DateTime")
          this.txt${ColumnInfo.Name.Alias.ToCapit()}.Text =
info.${ColumnInfo.Name.Alias.ToCapit()}.ToShortDateString();
#else
          this.txt${ColumnInfo.Name.Alias.ToCapit()}.Text =
info.${ColumnInfo.Name.Alias.ToCapit()}.ToString();
#end
#end ##endif
#end
       }

       /// <summary>
       /// 获取数据
       /// </summary>
       private ${ClassName}Info GetData()
       {
          ${ClassName}Info info = new ${ClassName}Info();
#foreach($ColumnInfo in ${TableInfo.ColumnList.Values})
#if(${ColumnInfo.AutoIncrement} == false)
#if(${ColumnInfo.NetType} == "System.Decimal" )
          info.${ColumnInfo.Name.Alias.ToCapit()} =
Helper.SafeConvertDecimal(this.txt${ColumnInfo.Name.Alias.ToCapit
()}.Text);
#elseif(${ColumnInfo.NetType} == "System.DateTime")
          info.${ColumnInfo.Name.Alias.ToCapit()} =
Helper.SafeConvertDate(this.txt${ColumnInfo.Name.Alias.ToCapit()}
.Text);
#else
          info.${ColumnInfo.Name.Alias.ToCapit()} =
this.txt${ColumnInfo.Name.Alias.ToCapit()}.Text;
#end
#end ##endif
#end
        }
```

当然，了解这些可能还不够，还需要了解模板后台，能够利用的数据属性等信息，包括数据库、表、字段等相关的属性，方便在模板引擎中进行调用，生成更加强大的代码片段。

软件安装后，有一个帮助文件，是提供给开发者进行了解模板后台数据的相关属性和方法的，如下所示。



运行后如下所示：



除了以上新增的功能外，一键生成基于我们的 Winform 框架结构的项目代码，是最为重要的功能，可以体验一下。另外，代码生成工具生成的数据库文档，快速生成实体类信息等模块，也是开发非常常用的功能，如果想了解更加关于我们的代码生成工具的信息，可以参考我博客里面的标签【代码生成工具】。