大家好，本期视频介绍最重要的功能，就是业务对象的编写了。要开始写代码啦！

现在，我们假设要开发一个“客户管理”模块，功能就是对“客户信息”做增删改查操作。

先实现第一个版本，客户信息有：姓名、年龄、出生日期、是否审核、用户评分、颜色标识、简历、备注这些内容。

其中：用户评分，可以认为是企业对客户的一个评估的分值，针对不同分词可能以后的销售策略是不同的。评分可以填写如：9.68这样的小数形式。

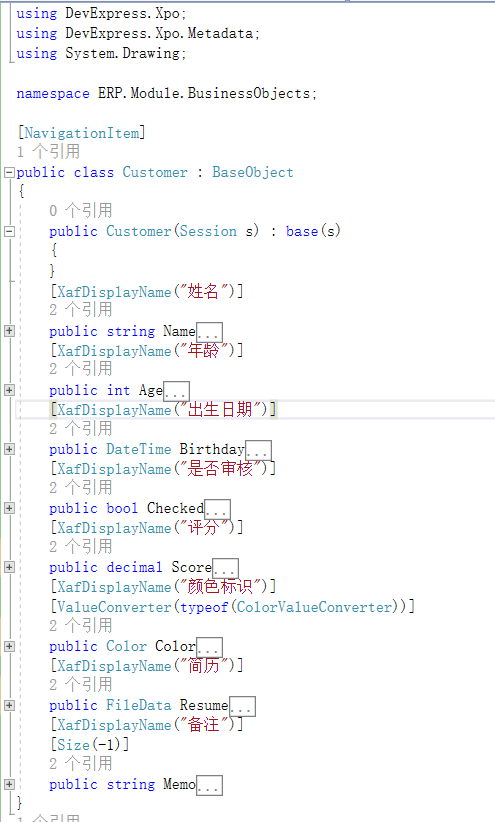
简历：可以是上传的一个文件。

颜色标识：为了方便区分不同类型的客户，使用颜色进行区分，这样在查看数据时更醒目。

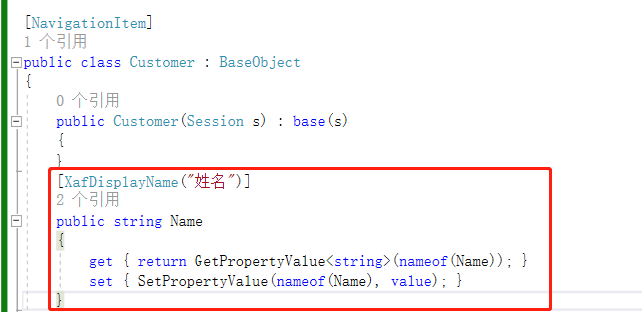
我们花5分钟来写一个业务对象类：写好之后是这个样子的。

注：作者写好的代码都放在github上了,地址是<http://www.github.com/tylike/bilibiliVideoText>

你也可以直接去下载看效果。



这个customer类中包含了8个属性。展开一个属性的详细情况来看看，以Name为例：



1.类型Customer必须是继承自BaseObject的，当然，还有很多可以用的基类，这个后面再细说。

2.构造函数必须接受一个Session类型的参数，并且调用基类的构造函数。先这样写死吧，以后再细说原因。

3.Name属性必须是public的，string为属性的类型。get和set必须要按图上写。

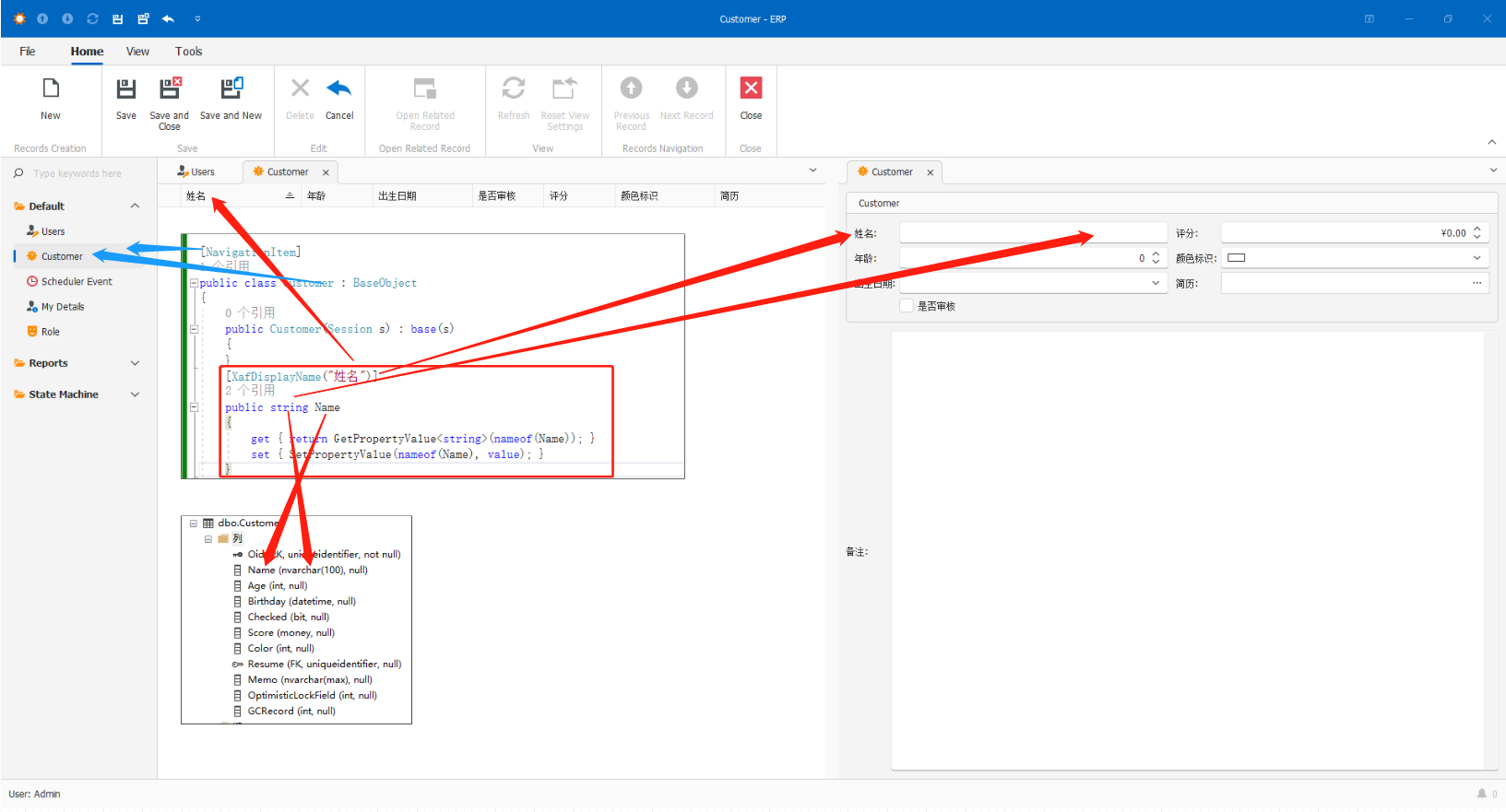
4.XafDisplayName(“姓名“)，这一句的功能是让界面上显示的内容。

我们先运行起来看看吧！

运行起来后，点击左侧的导航栏中的”Customer”条目，会出现“客户信息”的列表界面。

点击左上角的“New”按钮，会弹出新建客户信息的详细界面。

其实在系统启动时，框架自动创建了数据库，为了方便讲解。我把代码、数据库中自动生成的表结构、框架自动生成的两个界面，都放到了一张图中。你可以暂停视频，看看他们的对应关系。



简单来说，框架通过反射技术读取到了开发人员写下的代码，得到了写下的内容的结构。这部分被称作“元数据”，“元数据”也并不神秘，就是结构化的代码的一部分信息。比如类名称、基类、属性名称、属性类型等等这些东东。

然后，框架根据类名称，生成一张表，表的字段就是类中声明的那些属性。

每个属性的类型最终一定会与字段的类型都有对应关系。

比如上面的示例中，字符串类型(string)的姓名(Name)推导时被确定为nvarchar长度为100,

DateTime日期、整数、布尔型等，也都是同样的道理。总之都可以有个对应关系就好了。

使用过O R M的同学们一定知道这不是什么新玩意儿。但对于以前使用vb或更老的开发语言如delphi,power builder的老家伙们这可能就是新知识啦！

当然O R M的功能还有很多，其实主要功能就是把程序中的数据结构与数据库中的表进行映射和转换，使开发人员使用时不需要了解太细节的SQL语句组装过程。就是程序帮你拼SQL了。

X A F框架在O R M的基础上，又往前走了一步，即O R M只是将程序数据结构生成SQL和数据库结构，而框架则是用这些“元数据”生成用户界面。

用户界面分类：

在传统的开发项目中，用户界面可能是“页面”，如在web中。可能是“窗体”，如在windows forms程序中。但总之就是一种灵活划分方法。

在框架中，对应的概念是“视图”，视图就是显示数据用的，你可以理解成你有数据的内容有很多，视图是显示出一部分内容的功能。当然也可以是数据的全部分内容。

这和S Q L数据库中的视图差不多。

框架中的视图是统称，又分为4种视图：

1. 列表视图：ListView，即显示多条数据时用的。通常是在这个界面在中选择一条或多条，进行继续操作。
2. 详细视图：DetailView，用于显示或编辑一条数据时用的。
3. 搜索视图：本质上也是列表视图，但是是在一个详细视图中要选择另一个业务对象时，列出可用的数据，选择一条或多条后，使用选中的数据。这里列出可用的数据的，就是搜索视图。既然是和列表视图一样的为什么又多弄出一个概念呢？这是由于，通常，搜索视图显示的列（属性、字段）较少，或与列表视图不一样，不需要那么多内容，又或者，需要只过滤出一些数据，或者排序顺序不一样，总之，就是不一样，有哪些不一样，要看业务中是怎么规定的了。
4. 组合视图：DashboardView,这个名字和Dashboard模块只差了一个View，但却是完全不同的两个东东。这个组合视图就是将多个4种视图进行组合。当然DashboardView可以嵌套DashboardView，即套娃。但这个功能我不太推荐使用，有点麻烦，看应用场合吧。

视图类型就这么多，也挺容易理解的。有一个问题，这里的一条数据和多条数据说的挺简单，但实际上可能会很复杂。

数据结构通常像一棵树一样。

比如，我们现在看的客户信息，都是简单的属性，那么假如，我们扩展一下，客户有N个地址，假如客户会下订单买我们的产品，他会寄往不同的地址，有家庭地址、公司地址、丈母娘家、自己老家等等。

每个地址又有收货地址、收货电话、邮编等等属性。

那么，客户信息下面应该有一个集合属性，咱们的示例中还没有说这个情况，但先不管如何写代码，先想想前面提到的数据条数问题。

即一个客户下面又有4个地址数据，这算是一条数据吗？

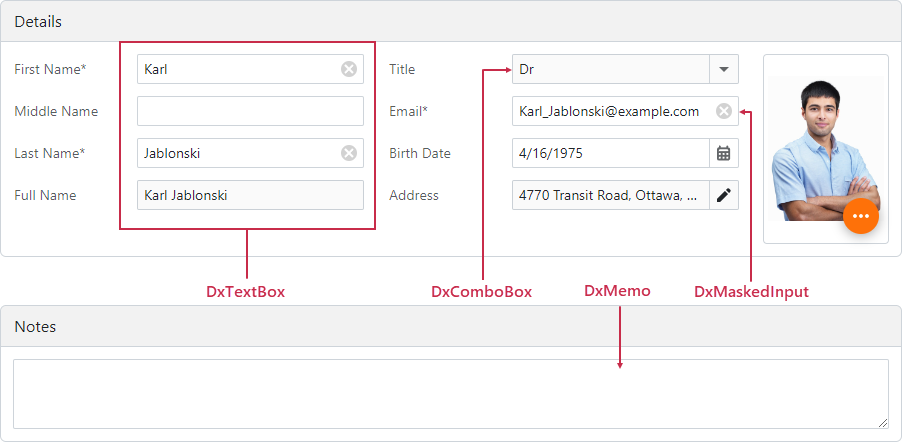
是的，这算一条数据，对于视图来讲，数据的条数只计算根级别的数据数量。比如两个客户要显示出来，那就用列表。对于一个客户要显示或编辑时就用详细视图，不管有几个地址的问题。地址是属性，也会在详细视图中显示出一个列表，详细视图可以嵌套列表视图。

接下来让我们看一下详细视图中的内容吧。在传统开发中，“姓名”后面的这个控件是输入内容的，就是那个文本框，在框架中，引入了一个比“控件”更大的概念，叫做“属性编辑器”。你可能会说这群天杀的为什么没事就造概念呢？

其实这和非得把web叫成云，数据库叫做大数据，弄几个3D就叫元宇宙是不一样的，还是有必要的，哈哈！

“属性编辑器”是把是显示的标签，即“姓名”这两个字的控件加上后面的编辑控件合并到了一起。同时，在数据验证时，比如报错了，显示出一个错误的小图标的功能。在界面布局时如姓名和后面的控件是如何摆放的，也进行了控制。比如，左右排列、上下排列、是不是要显示标题等等。再比如根据规则如果是必填的，就在标题上显示一个“\*”号。

对于整个系统来讲，可以整体进行控制了，如，系统中所有的界面的属性编辑器，都上下排列。可以通过程序的方式进行统一设置。可以更规范的控制了。



接下来我们看看在代码中写的属性类型与属性编辑器的默认对应关系：

这里有一张表格，

|  |  |
| --- | --- |
| C#代码中定义的属性的类型 | 对应的属性编辑器 |
| String | 单行文本：文本输入框  多行文本：多行文本输入框  有预定义值：下拉框  富文本编辑器 |
| Numeric,int、long、decimal、float等所有的数字类型都在此类。显示样式如下图所示： | 带有上下调节的增减数值的文本输入框  但只能输入合法的数字 |
| DevExpress XAF：Blazor 的数字属性编辑器 | |
| 日期时间类型（DateTime） | 日期下拉选择框 |
| 布尔型（Boolean） | 三种显示样式：   1. 显示一个复选框 2. 显示一个下拉框 3. 和2一样，但是可以带个图标   XAF 布尔属性 WinForms |
| BLOB Image | 图像编辑框  下拉框中显示图像编辑 |
| Collection | 显示一个嵌套的列表视图 |
| File Attachment | 文件附件编辑器 文件附件属性 Blazor |
| Type | 下拉选择框 |
| 条件表达式（Criteria）字符串类型 | 1.直接编辑的如图所示。 2.在弹出窗口中显示编辑。 XAF ASP.NET Core Blazor 弹出过滤器属性编辑器 |
| Color | 颜色选择下拉框：  XAF 颜色属性 WinForms |
| Enumeration，所有枚举类型 | 下拉框，可以带图标 |
| Reference (Foreign Key, Complex Type) | 1. 搜索编辑器 2. 显示一个引用编辑器，并可以直接弹出编辑框 |
|  |  |

所有的类型都可以对应到属性编辑器上，上表中已经包含了多数常用的情况。

如果表中找不到你需要的类型，也可以进行自定义属性编辑器，本节中不做展开了。

上表中的集合类型、引用类型、枚举类型都没有详细解释。后续的章节中我们进行讲解。

本节内容就分享这么多吧，创作不易，请帮忙一键三连，谢谢！