SubSonic 指南中文版

翻译:王鹏程 张原 王伟

策划:毛凌志

2009 年 1 月

北京工业大学软件学院

PS:有问题反馈至http://lexus.cnblogs.com

Getting Started with SubSonic

By Scott Kuhl (http://www.geekswithblogs.net/scottkuhl)

SubSonic is an open-source toolset, created by Rob Conery, as an attempt to put the fun back into programming and just get the job done. Inspired by Ruby on Rails, SubSonic takes a minimalist approach to coding and emphasizes convention over configuration. While it takes its inspiration from Ruby on Rails, it is not a port of it. (Check out MonoRail it that's what you're looking for.) Instead, SubSonic takes the best ideas of Ruby on Rails and adapts them into the already existing ASP.NET framework. Currently SubSonic, version 1.05, implements two core features:

SubSonic 是一种开源工具，由 Rob Conery 创造，作为一种尝试把乐趣带入计划并且完成这项工作。灵感来自 Ruby on Rails，SubSonic 采取了最低限度的办法，编码，并且强调结构上的惯例。虽然它的灵感来自 Ruby on Rails，但是这并不是它的一种端口。（查看单轨这是您要找的内容。）相反，SubSonic 继承了 Ruby on Rails 最好的思想，并且使它们融入已经现有的 ASP.NET 框架。目前，1.05 版本 SubSonic 有两个核心特点：

ActiveRecord

This design pattern in its simplest form is one class per database table, one object per database row. SubSonic includes a build-time code generator to implement this pattern that acts as an object-relational mapper eliminating the need to write SQL. It also includes a dynamic query tool and simple store procedure support to extend the model when needed.

这种设计模式的最简单的形式是每个数据库表一类，每个数据库行一个对象。SubSonic 包括建立时间码发生器实施这一模式，作为一个对象关系映射无需编写 SQL。它还包括一个动态查询工具和简单的存储过程支持在需要时扩展该模型。

Scaffolding

Admin pages are a chore that scaffolding helps remove. Simply point a scaffold control at a table and you get the standard grid view and detail view to find and update data in the underlying table. While not meant to ever be shown to users, it makes a nice, quick and easy developer tool.

管理的网页是件 scaffolding 帮助删除的苦差事。只要在表格中指点棚架控制，你得到标准的网格视图和细节视图，用来在潜在的表格中找到并更新数据。虽然并不意味着永远向用户显示，但它提出了好的、快速简便的开发工具。

Requirements　需求

SubSonic will work fine with Visual Web Developer 2005 Express Edition and SQL Server 2005 Express Edition, so you can get started without dropping a dime. You can also use MySQL or any database that can be accessed through Enterprise Library for .NET Framework 2.0, but SQL Server is probably the most likely setup. Note: The sample web site included with the SubSonic source code includes an SQL script to create the Northwind database. This article will use that database when examples are needed.
SubSonic可以与Visual Web Developer 2005 Express Edition和SQL Server 2005 Express Edition很好的工作，这样你就可以未留任何遗漏的使用。您也可以将MySQL或任何Enterprise Library可读取的数据库用于.NET Framework 2.0，但SQL Server的也许是最有可能的安装。注：样本的网站包括SubSonic的源代码包含一个SQL脚本创建Northwind数据库。本文在例子需要时将使用该数据库。

Setup　安装

Setup is easy, just download SubSonic from CodePlex and reference SubSonic.dll found in the bin directory. Alternatively, you can open the solution and compile a release build yourself.
(You will need Visual Studio 2005 Standard Edition to open the solution because it also includes a sample web site or you can use Visual C# 2005 Express Edition to open just the project.)
安装非常简单，只需从CodePlex下载SubSonic并参考SubSonic.dll在bin路径中建立。另外，您也可以打开解决方案，并自己新建一个。
（您需要的Visual Studio 2005标准版开来打开解决方案，因为它也包括一个范例网站或您可以使用Visual C# 2005 Express Edition仅仅来打来项目。　）

Configuration　配置

SubSonic requires a minimal amount of configuration to get going.
SubSonic的使用要求一个最低的配置数量。

SubSonic Configuration Section　SubSonic配置节

Start by adding a SubSonic configuration section inside the configuration tag in the web.config file. This default configuration should work for most projects.
首先，在Web.config文件中，在<configSections>内增加一个SubSonic配置节。这种默认配置应该对大多数项目有用。

```
    <configSections>
        <section name="SubSonicService" type="SubSonic.SubSonicSection, SubSonic"
        allowDefinition="MachineToApplication" restartOnExternalChanges="true"
        requirePermission="false"/>
    </configSections>
```

Data Provider　数据供给者

Second, you will need to setup a data provider. Three are currently supported by SubSonic: SQL Server, MySQL and Enterprise Library. The following are sample configurations for each of these. This information is also added inside the configuration tag.
第二，您将需要设置一个数据提供者。目前被SubSonic支持的有三个： SQL Server，MySQL和Enterprise Library。下面是这三个的示例配置。这些信息也添加在<configSections>内。

```
    <SubSonicService defaultProvider="SqlDataProvider" spClassName="SPs"
    fixPluralClassNames="true">
        <providers>
             <add name="SqlDataProvider" type="SubSonic.SqlDataProvider, SubSonic"
            connectionStringName="NorthwindConnection"/>
              <add name="ELib2DataProvider" type="ActionPack.ELib2DataProvider,
            ActionPack" connectionStringName="NorthwindSQL"/>
            <add name="MySqlDataProvider" type="ActionPack.MySqlDataProvider,
            ActionPack" connectionStringName="NorthwindMySQLConnection"/>
        </providers>
     </SubSonicService>
```

There are five values that can be set in the SubSonicService tag.
在<SubSonicService>内有五个值可以设置。

Ÿ **defaultProvider** - Multiple providers can be setup in the configuration. This value indicates which provider to use.
  n 多个供应者可以在配置内设置。这个值表示使用的是哪个供应者。

Ÿ **fixPluralClassNames** - SubSonic can remove the plural characters from the end of table names to make class names more consistent. For example, the Products table would produce a Product class.
  n SubSonic可以从表名末尾删除复数字符，以使类别名一致。例如，产品表会显示产品类别。

Ÿ **generatedNamespace** - By default all classes generated will be part of the project's global namespace. This value overrides that behavior and includes all classes in the given namespace. For example, by setting this to Northwind you would get Northwind.Product.
  n 默认情况下产生的所有类别将是该项目的全局命名空间的一部分。这个值忽略那种行为，并且在特定的命名空间包含所有类别。例如，通过设置这值为Northwind你会获得Northwind.Product 。

Ÿ **spClassName** - Each stored procedure will generate a method of the same name. The value will be the class these methods are included under. For example, by setting this to SPs the CustOrderHist stored procedure would be SPs.CustOrderHist. Using the above namespace example in conjunction with this value would produce Northwind.SPs.CustOrderHist.

- n 每个存储过程会产生相同名称的一种方法。这个值是将这些方法包含在其中的类。例如，通过设置这值为SPs，CustOrderHist存储过程将会成为SPs.CustOrderHist 。使用上述命名空间例子与这个值相结合将会产生Northwind.SPs.CustOrderHist 。

Ÿ **templateDirectory** - It is possible to override the code generated by SubSonic. This directory would contain the code templates to override the default templates supplied. This will be covered in greater detail later when discussing Code Generation Templates.

- n 有可能覆盖SubSonic产生的代码。此目录将包含的代码模板，以取代默认模板提供的。这将会在稍后讨论代码生成模板时更详细地讲解。

Ÿ **useSPs** - If you do not want a class generated for stored procedures, set this value to false.

- n 如果你不想要一个为存储程序而产生的类，设置此值为false。

Database Connection String 数据库连接字符串

Third, you need to define a database connection string.
第三，你需要定义一个数据库连接字符串。

```
<connectionStrings>
    <add name="NorthwindConnection" connectionString="Data
    Source=localhost\SQLExpress; Database=Northwind; Integrated Security=true;"/>
</connectionStrings>
```

Build Provider Configuration 建立供应商配置

Fourth, you need to setup a build provider to create the auto generated classes. This needs to be added to the compilation tag.
第四，你需要设置一个建立供应商来产生自动生成的类。这需要在<compilation>内添加。

```
<buildProviders>
    <add extension=".abp" type="SubSonic.BuildProvider, SubSonic"/>
</buildProviders>
```

Build Provider Definition 建立供应商的定义

Last, you need to create an .abp file for this build provider to use. You do this by adding a text file named Builder.abp to the App_Code folder. Inside this file you indicate which database tables should have auto generate classes. If you want all tables, just enter *, otherwise, list the tables one per line.
最后，您需要创建一个.abp 文件供此建立供应商使用。通过在 App_Code 文件夹中增加一个命名为 Builder.abp 的文本文件来这样做。在这个文件中你说明哪些数据库表应该有自动生成类。如果你想所有表格，只需输入*，否则，每行一个的列出各个表。

Summary 总结

In summary, these are the items you need to configure.
总之，这些是你需要配置的项。

Ÿ SubSonic Configuration Section 配置节
Ÿ Data Provider 数据供应者
Ÿ Database Connection String 数据库连接字符串
Ÿ Build Provider Configuration 建立供应商配置
Ÿ Build Provider Definition 建立供应商定义

Here is a sample web.config with all the SubSonic required values defined. You can also use the web.config included in the sample web site downloaded along with the SubSonic source code as a starting point, which is where these sample values are derived from.
这里是一个范例Web.config中的所有SubSonic要求的值的定义。您也可以将Web.config包含于下载的范例网站中，和SubSonic的源代码一起作为一个起始点，这是这些示例值是来源。

```
<?xml version="1.0"?>
<configuration>
    <configSections>
        <section name="SubSonicService" type="SubSonic.SubSonicSection,
        SubSonic" allowDefinition="MachineToApplication"
        restartOnExternalChanges="true" requirePermission="false"/>
    </configSections>
    <appSettings/>
    <connectionStrings>
        <add name="NorthwindConnection" connectionString="Data
        Source=localhost\SQLExpress; Database=Northwind; Integrated
        Security=true;"/>
    </connectionStrings>
    <SubSonicService defaultProvider="SqlDataProvider" spClassName="SPs"
    fixPluralClassNames="true">
        <providers>
            <add name="SqlDataProvider" type="SubSonic.SqlDataProvider,
            SubSonic" connectionStringName="NorthwindConnection"/>
        </providers>
    </SubSonicService>
    <system.web>
        <compilation debug="true" defaultLanguage="C#">
            <buildProviders>
                <add extension=".abp" type="SubSonic.BuildProvider, SubSonic"/>
            </buildProviders>
            <assemblies>
                <add assembly="System.Management, Version=2.0.0.0,
                Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>
                 <add assembly="System.Data.OracleClient, Version=2.0.0.0,
                Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                <add assembly="System.Configuration.Install, Version=2.0.0.0,
                Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>
                <add assembly="MySql.Data, Version=1.0.7.30072, Culture=neutral,
                PublicKeyToken=C5687FC88969C44D"/>
            </assemblies>
        </compilation>
    <authentication mode="Windows"/>
     </system.web>
</configuration>
```

To make sure everything is working, build the web site, then open up the Class View window and you should see class names that match your tables. For example, using the Northwind database, you should now see classes named Category, Customer and Employee.
为了确保一切工作，建立网站，然后打开类视图窗口，您应该可以看到与表格相匹配的类名。例如，使用Northwind数据库，您现在应该可以看到命名Category，Customer和Employee的类。

Trust Level 信任水平

Build providers will not work in a medium trust environment, they need full trust. This could be a problem if you plan on having someone else host your web application. Most hosting providers are not set up to run in a full trust environment. As a work around, there are two class generators provided in the SubSonic Starter Kit, which will be covered in detail later, which output code in text form that can be included and compiled directly into your application.
建立供应商将不会在中等信任的环境中工作，他们需要充分的信任。如果你计划让别人托管你的应用程序这可能是一个问题。大多数托管服务提供商在完全信任的环境中不会装配运行。作为一个工作循环，在SubSonic启动套件中提供两个类产生器，这将稍后详细讲解，其中以文本方式输出可以在你的应用里包含并直接编辑的代码。

Two current exceptions to this hosting rule are Ultima Hosts and Discount ASP.net. It's a good idea to check with your hosting provider before beginning a SubSonic project.
目前两个对于主办规则的例外是Ultima Hosts和Discount ASP.net 。在SubSonic项目开始前协调托管服务提供商是一个好方法。

Classes 类

At its heart, the most useful feature of SubSonic is the auto generated classes. As stated earlier, there will be one class for each table in the database. The next several sections will take an in-depth look at the functionality provided. The examples in the section will use the Products table from the Northwind database as their foundation.

SubSonic 的核心，最实用的特征便是自动生成出来的类。正如之前的叙述，会有一个关于数据库中每一张数据表的类。之后的一些部分将深度研究所提供的功能。在这部分中的实例将会用到 Northwind 数据库中的 Products 数据表作为基础。

Extending the Model 扩展模型

The creator of SubSonic anticipated that not any one model would accommodate everyone's needs. Beyond providing the source code for you to change, there is another way you can adapt and extend the generated model without altering the core code base. SubSonic creates partial classes, a feature introduced in .NET 2.0 to make code generation tools more accessible. Partial classes allows you to add additional properties and methods without altering generated code and worrying about the generation process overwriting your changes.

SubSonic 的创造者意识到没有任何一个模型能够容纳每个人的需求。除了为你提供源代码去修改，还有另一种方法你能够接受并且扩展生成的模型而不必去修改基础的核心代码。SubSonic 创建了分部类，它是在.NET 2.0 中被引进使代码生成工具更容易实现的机制。分部类允许你添加附加的属性和方法而不必去修改已生成的代码也不必担心生成过程会重写了你的改动。

Constructors 构造函数

Creating a new object is straight forward.

```
Product product = new Product();
```

There are also two overloaded constructors that are used to load an existing object from the database. These are covered below when discussing the many ways SubSonic has of retrieving data.

直接创建一个新的对象。

```
Product product = new Product();
```

还有两个被重载的构造函数用于载入在数据库中已经存在的对象。在探讨 SubSonic 的多种检索数据的方法时，它们也会被涵盖。

Properties 属性名

Each object will have one property for each column in the table it is derived from. So a Product object would contain the following properties: CategoryID, Discontinued, ProductID, ProductName, QuantityPerUnit, ReorderLevel, SupplierID, UnitPrice, UnitsInStock and UnitsOnOrder.

每个对象会对从相应的数据表中获得的每一列用一个属性名来标识。所以 Product 对象会包含以下属性名：CategoryID、Discontinued、ProductID、ProductName、QuantityPerUnit、ReorderLevel、SupplierID、UnitPrice、UnitsInStock 和 UnitsOnOder。

Rules Enforcement 强制性规则

None of the properties enforce any authorization, validation or business rules. That is left up to you. How to implement these rules effectively is discussed later under Business Rules.

没有任何一个属性强制执行任何授权、验证或商业规则。那是由您来决定的。如何在商业规则下有效地实施这些规则会在之后探讨。

Object Key 对象主键

Each object will have a key property, in the case of Product that key is ProductID. SubSonic uses this property as the object ID because ProductID is defined as the primary key in the database. (SubSonic will not work with tables that do not have a primary key.) Generally tables will have one of three different types of primary keys: GUID, an auto incrementing integer or a natural key. Any key that can be represented as a GUID, integer or string is supported, so all of these are covered. The one thing that is missing is support for multiple column primary keys.

每个对象会有一个主键属性，比如在 Product 数据表中那个主键是 ProductID。SubSonic 利用这个属性名作为对象 ID 因为 ProductID 在数据库中被定义为主键。（SubSonic 对没有主键的数据表不起作用。）一般的数据表拥有三种不同主键中的一种：GUID，一种自动增长的整数或自然数的键。能够代表 GUID 的任何键，包括整型或字符串型，都涵盖其中。还未交代的一件事是对多主键的支持。

State 状态

There are three properties that help you manage the current state of an object:

- IsDirty - Does the object have changes that have not yet been saved to the database?
- IsLoaded - Was the object loaded from the database?
- IsNew - Was the object created in memory? In other words, this object does not exist in the database.

有三个属性帮助你管理对象的当前状态：

- IsDirty - 是否对象已经改变却没有存入到数据库中？
- IsLoaded - 对象从数据库中读取出来了么？
- IsNew - 对象在存储体中创建了么？换句话说，这个对象在数据库中不存在。

Columns 列

Sometimes you will need to pass in the name of a column in string format as a parameter to a method, as you will see later in the FetchByParameter methods. Instead of using a string literal, you use the static Columns collection found on each class.

```
Product.Columns.ProductName
```

Its also possible to get the column value by calling the GetColumnValue method found on each object, passing in the column name.

```
product.GetColumnValue(Product.Columns.ProductName);
```

有时你需要传递字符串形式的列名作为方法参数，如同您在之后看到的 FetchByParameter 方法。不必逐字输入字符串，您可以使用每个类建造的静态 Columns collection。

```
Product.Columns.ProductName
```

也可以通过调用每个对象建造的 GetColumnValue 方法，传入列名来获取某一列的值。

```
product.GetColumnValue(Product.Columns.ProductName);
```

Retrieving a Single Object 检索单个对象

It is possible to load data at the object level using one of three overloaded Load methods by passing in either a DataRow, DataTable or DataReader. But you are more likely to use the static FetchByID method or an overloaded constructor.

可以通过使用三种重载的读取方法中的一种，传入 DataRow、DataTable 或 DataReader 之一在对象级读取数据。然而您更有可能使用静态的 FetchByID 方法或者重载的构造函数。

FetchByID FetchByID 函数

FetchByID takes has one parameter, the primary key value of the object in GUID, integer or string format, and returns a populated object.

```
Product product = Product.FetchByID(id);
```

If the object is not found, SubSonic will not throw an error. Instead you should check to see if the key property matches the passed in ID to determine if the object was found.

```
int id = 1;
Product product = Product.FetchByID(id);
if (product.ProductID == id)
```

```
{
    // Success
}
else
{
    // Not Found
}
```

FetchByID 有一个参数，在 GUID 中对象主键的值，整型或字符串型，并且返回一个限定的对象。

```
Product product = Product.FetchByID(id);
```

如果对象没有找到，SubSonic 也不会抛出错误。取而代之的是您应该查看下主键属性是否符合传入的 ID 来决定对象是否能被找到。

```
int id = 1;
Product product = Product.FetchByID(id);
if (product.ProductID == id)
{
    // Success
}
else
{
    // Not Found
}
```

Constructor 构造函数

To do the same thing using the overloaded constructor method that takes the primary key looks like this.

```
Product product = new Product(id);
```

By using another overloaded constructor that takes a column name and value as the parameters, it is possible to load the object based on another uniquely identifying column instead of the primary key.

```
Product product = new Product(Product.Columns.ProductName, "Chai");
```

Be careful when using the last form. If you use a column that does not have a unique constraint and more than one record is returned by the database, your object will be populated with the first one.

使用重载的构造函数的方法同样可以像下面这样去获取主键。

```
Product product = new Product(id);
```

列名和值作为参数，通过使用另一个重载的构造函数，可以读取基于其它的唯一标识列的对象来代替主键检索。

```
Product product = new Product(Product.Columns.ProductName, "Chai");
```

当使用这种形式检索时，请务必注意。如果您所使用的列没有唯一限制并且有多于一条的数据库返回记录，您的对象将被第一条记录所限定。

Loading and Saving State 读取和存储状态

There is one other way to load an object. Each object has a method called NewFromXML which creates an object from an XML string. This method is meant to be used with the ToXML method to write the object's state to a temporary location such as Session state.

```
Session["Product"] = product.ToXML();
product = (Product)product.NewFromXML(Session["Product"].ToString());
```

还有另一种读取对象的方法。每个对象都有个名为 NewFromXML 的方法用于将 XML 字符串创建为对象。这个方法显然要与将对象的状态写入暂存空间例如会话状态的 ToXML 方法搭配使用。

```
Session["Product"] = product.ToXML();
product = (Product)product.NewFromXML(Session["Product"].ToString());
```

Retrieving Multiple Objects 检索多个对象

FetchAll FetchAll 函数

The easiest way to return a list of objects is the FetchAll static method. It does just what the name says, returning a list of every object of that type in the database in DataReader format, making it easily bindable to data controls like the GridView.

```
GridView1.DataSource = Product.FetchAll();
GridView1.DataBind();
```

You can also pass the FetchAll method a SubSonic OrderBy.

```
GridView1.DataSource =
Product.FetchAll(SubSonic.OrderBy.Desc(Product.Columns.ProductName));
GridView1.DataSource =
Product.FetchAll(SubSonic.OrderBy.Asc(Product.Columns.ProductName));
```

返回多个对象最简单的方法便是静态方法 FetchAll。正如它方法名的意思一样，以 DataReader 形式返回在数据库中相应类型的每一个对象，使得它很容易实现对数据操控的屏蔽就像 GridView。

```
GridView1.DataSource = Product.FetchAll();
GridView1.DataBind();
```

你也可以把 SubSonic 的排序传入 FetchAll 方法。

```
GridView1.DataSource=
Product.FetchAll(SubSonic.OrderBy.Desc(Product.Columns.ProductName));
```

GridView1.DataSource=
Product.FetchAll(SubSonic.OrderBy.Asc(Product.Columns.ProductName));

FetchByParameter FetchByParameter 函数

If you want to list a subset of data rather than the entire list you can use FetchByParameter instead. At a minimum you only need to supply a column name and value to match.

```
GridView1.DataSource =
Product.FetchByParameter(Product.Columns.SupplierID, 1);
```

Just like the FetchAll, you can apply an OrderBy.

```
GridView1.DataSource =
Product.FetchByParameter(Product.Columns.SupplierID, 1,
SubSonic.OrderBy.Desc(Product.Columns.ProductName));
```

If you would rather find a range of objects, you can use the overloaded FetchByParameter that takes a SubSonic Comparison. Again you could also choose to append an OrderBy.

```
GridView1.DataSource =
Product.FetchByParameter(Product.Columns.SupplierID,
SubSonic.Comparison.GreaterThan, 1);
```

比起整个对象集如果您更需要其中一部分数据的话，您可以使用 FetchByParameter 方法。您至少要提供列名和值去匹配。

GridView1.DataSource = Product.FetchByParameter(Product.Columns.SupplierID, 1);

与 FetchAll 方法类似，您可以使用排序。

GridView1.DataSource = Product.FetchByParameter(Product.Columns.SupplierID, 1, SubSonic.OrderBy.Desc(Product.Columns.ProductName));

如果您就想找出一定范围内的对象，您可以使用重载的具有 SubSonic 比较的 FetchByParameter 方法。您还是可以选择是否附加排序。

GridView1.DataSource = Product.FetchByParameter(Product.Columns.SupplierID, SubSonic.Comparison.GreaterThan, 1);

FetchByQuery FetchByQuery 函数

There is also a FetchByQuery method that takes a SubSonic Query.

```
         GridView1.DataSource = Product.FetchByQuery(query);
```

Queries will be discussed later along with more information on comparisons.

还有个具有 SubSonic 查询的 FetchByQuery 方法。

GridView1.DataSource = Product.FetchByQuery(query);

查询会在之后与更多比较的情况一起探讨。

Find Find 函数

The Find static method allows you to retrieve matching records based on an existing object. For example, if you want to find all Products with SupplierID of 1 and CategoryID of 1 you can create a new Product object with those values and pass it into the Find method.

```
Product product = new Product();
product.SupplierID = 1;
product.CategoryID = 1;
GridView1.DataSource = Product.Find(product);
```

The Find method also has the option of passing an OrderBy.

静态方法 Find 允许您检索基于已经存在对象的匹配记录。例如，如果您需要找出所有 SupplierID 为 1 并且 CategoryID 为 1 的 Products，您可以创建一个写入那些值的新对象，并把它传入 Find 方法。

```
Product product = new Product();
product.SupplierID = 1;
product.CategoryID = 1;
GridView1.DataSource = Product.Find(product);
```

Find 方法也可以考虑传入排序。

Querying 查询

If none of the Fetch or Find methods will work for a particular situation, you can still fall back on SubSonic querying or stored procedures which are discusses later.

如果没有一种 Fetch 或 Find 方法能够在实践中起作用，您仍然可以回归到 SubSonic 查询或之后要探讨的存储进程。

Updating the Database 数据库更新

Saving your changes to the database is as easy as calling the Save method on the object.

```
product.Save();
```

But it's important to know what is happening when you call this method. The above example does not pass the current user information. As you'll see later when discussing conventions, SubSonic has the ability to keep some basic history information. If your tables are setup to record this information, you need to pass either the User ID in integer or GUID format, or the User Name in string format.

```
product.Save(User.Identity.Name);
```

If the primary key is a GUID or auto incrementing integer and the object is new, the save process will update the key property in the object. The save process will also set the IsNew and IsDirty properties to false.

将您的更改保存到数据库只需调用当前对象的 Save 方法一般简单。

```
product.Save();
```

但是了解您在调用这个方法时发生了什么是非常重要的。上面的那个例子无法传递当前用户的信息。正如您会在之后探讨协定时看到，SubSonic 拥有保持一些基本历史信息的能力。如果您的数据表被设定去记录这条信息，您需要传递整型或者 GUID 型的 User ID，或者字符串格式的 User Name。

```
product.Save(User.Identity.Name);
```

如果主键是 GUID 或者自增加整数而且对象是新的，保存过程将会更新对象的主键属性。保存过程也会把 IsNew 和 IsDirty 属性设为否。

Insert and Update 插入和更新

It is also possible to insert or update data through static class methods.

```
Product.Insert("Product Name", 1, 1, "10", 10.00, 10, 10, 1, false);
Product.Update("Product Name", 1, 1, "10", 10.00, 10, 10, 1, false);
```

These methods do all the work of instantiating the object, setting the properties and calling the Save method. The downside is that neither method returns an auto generated primary key value.

通过静态类的方法插入或更新数据同样是可行的。

```
Product.Insert("Product Name", 1, 1, "10", 10.00, 10, 10, 1, false);
Product.Update("Product Name", 1, 1, "10", 10.00, 10, 10, 1, false);
```

这些方法做了实例化对象，设定属性和调用 Save 方法的所有工作。缺点就是没有任何一个方法返回自动生成的主键的值。

Deleting 删除

There are two types of deletes: logical and permanent. A logical delete does not remove the record from the database, rather a column is used to mark the record as deleted. SubSonic will treat a column named "Deleted" or "IsDeleted" as this flag. To perform a logical delete, your table must have one of those columns defined as a BIT, then the IsDeleted property will show up on the object that can be set to true. When the Save method is called, the object will be marked as deleted. There is also a static method called Delete that takes an object key value and does the same thing.

```
Product.Delete(product.ProductID);
```

Note that this example will not work with the default Northwind database since the

Products table does not have either type of delete column. You need to add a "Deleted" column to make it work.
Unfortunately, SubSonic treats this purely as convention when retrieving data. You will need to filter out deleted records, for example, by using a FetchByParameter method.
Permanent deletes are easier. Just call the static Destroy method.

```
Product.Destroy(product.ProductID);
```

有两种删除方式：逻辑的和永久的。逻辑删除并不从数据库中删除记录，而是把一列用于标识记录被删除。SubSonic 会把这个称作“Deleted”或者“IsDeleted”的列作为标志位。为了实现逻辑删除，您的数据表必须其中有一列定义为 1 比特，之后 IsDeleted 属性才能够在对象中起作用被设定。当 Save 方法被调用时，对象会被标识为删除。还有一个叫做 Delete 的静态方法通过获取对象的键值能够达到同样的目的。

```
Product.Delete(product.ProductID);
```

注意这个例子将不会对默认的 Northwind 数据库起作用由于 Products 数据表没有任何格式的删除列。您需要去添加一个“Deleted”列使它运行。
不幸的是，在检索数据时 SubSonic 仅仅把它当作协定。您需要过滤掉被删除的记录，例如，当使用 FetchByParameter 方法时。
永久性删除更加容易。只需要调用静态的 Destroy 方法。

```
Product.Destroy(product.ProductID);
```

Business Rules 商业规则

There are two places that business rules can be injected into the Save process. A PreUpdate method is called at the beginning of the Save process and a PostUpdate method is called at the end. Both of these methods are virtual methods defined in each class that take no parameters and have no return value. To define these methods, create a partial class file with the same name as the class.

```
public partial class Product
{
    protected override void PreUpdate()
    {
        // Do something
    }

    protected override void PostUpdate()
    {
        // Do something
    }
}
```

This is an easy way to add simple authorization, validation and business rules. If you need more flexibility, check out the Controller pattern implementation discussed later in Conventions.

保存过程中有两处商业规则能够插入进来。保存过程开始时，PreUpdate 方法被调用并且结束时 PostUpdate 方法被调用。这些方法都是定义在每个类中不需要任何参数也没有返回值的虚函数。要定义这些方法，用和指定类相同的名称创建一个分部类文件。

```
public partial class Product
{
    protected override void PreUpdate()
    {
        // Do something
    }

    protected override void PostUpdate()
    {
        // Do something
    }
}
```

这是一种添加授权、验证和商业规则的简单方法。如果您需要更加灵活的方式，查询之后协定里对 Controller 类型实现的探讨。

Underlying Data 下层数据

Auto generated classes may expose a simple object based interface, but below the surface the data is being held in ADO.NET DataTables. SubSonic exposes a few methods that allow you to get at the underlying structure. The TableName property returns the name of the actual table associated with the object.

```
product.TableName;
```

The static class method GetTableSchema returns the underlying table.

```
SubSonic.TableSchema.Table tableSchema = Product.GetTableSchema();
```

Note: There is also a static property Schema that returns the same information. For the property to work, the class must be instantiated at least once. The GetTableSchema takes care of this for you. Both of these methods will be used later when examining Queries. The Inspect method returns an HTML representation of the object using <table> markup tags. There is an overloaded version of the method that takes a single boolean parameter, useHtml, that can be set to false to return plain text.

```
string html = product.Inspect();
```

自动生成类也许给出基于接口的简单对象，但是表面之下数据存在于 ADO.NET 数据表里面。SubSonic 给出一些方法允许您获取下层的结构。TableName 属性返回关系到这个对象的实表名称。

```
product.TableName;
```

静态类的方法 GetTableSchema 返回下层数据表。

```
SubSonic.TableSchema.Table tableSchema = Product.GetTableSchema();
```

注意：还有个静态属性叫 Schema 能够返回相同的信息。使用这个属性时，类必须实例化至少一次。GetTableSchema 为您关注到了这点。在之后检查查询时这些方法都会被用到。

Inspect 方法返回对象的 HTML 呈现并使用<table>作为标签。此方法有一个重载版本需要传入一个布尔参数 useHtml 设置为否去翻译回原来的文本。

```
string html = product.Inspect();
```

Collections

For a more traditional object-oriented approach to multiple objects you can use the auto-generated collections. Just like classes, SubSonic will also generate a collection for every table. (The naming convention for collections is ClassCollection.) These collections are implemented as Generic Lists that include some additional functionality.
你可以使用自动生成的集合类 collections，以一种更加传统的面向对象的方式来处理多个对象。就像根据表名生成的类一样，SubSonic 也会为每一张表生成一个集合类。（生成的集合类的命名习惯是 ClassCollection）这些生成的集合类会成为包含一些传统的集合处理方法的集合序列。

Loading a Collection

The simplest way to load a collection is by calling the Load method with no parameters to retrieve every record.

```
ProductCollection products = new ProductCollection();
products.Load();
```

You can also pass the Load method an IDataReader, DataTable or SubSonic Query.

```
IDataReader reader = Product.FetchAll();
products.Load(reader);
reader.Close();
```

(Make sure you close the reader. The Load method will not do this for you.)
载入一个集合
最简单的载入一个集合到内存的方法就是调用 Load 方法，在不给出任何参数的情况下，该方法将返回集合的每一条记录。

```
ProductCollection products = new ProductCollection();
products.Load();
```

也可以给 Load 方法一个参数，这个参数可以是 IDataReader, DataTable，或者 SubSonic Query 类型的。

```
IDataReader reader = Product.FetchAll();
products.Load(reader);
reader.Close();
```

（要确保在使用完之后写代码关闭 reader，记住，Load 方法不会帮你关闭这个 reader.）

Ordering a Collection

You can order a collection based on a single column by calling OrderByAsc or OrderByDesc before calling the Load method.

```
ProductCollection products = new ProductCollection();
products.OrderByAsc(Product.Columns.ProductName);
products.Load();
```

按序取得一个集合

你可以根据某个单一行来排序集合。这样需要在调用 Load 方法之前使用 OrderByAsc 或 OrderByDesc 方法来说明要进行升序或者降序的排序。

```
ProductCollection products = new ProductCollection();
products.OrderByAsc(Product.Columns.ProductName);
products.Load();
```

Filtering a Collection

You can filter a collection by calling BetweenAnd, Where or WhereDatesBetween before calling the Load method.

```
OrderCollection orders = new OrderCollection();
orders.BetweenAnd(Order.Columns.OrderDate, new DateTime(1980, 1, 12),
DateTime.Now);
orders.Load();
```

The Where method works just like the SubSonic Query Where which is covered in the next section.

过滤一个集合的结果

可以在调用 Load 方法之前通过使用 BetweenAnd, Where 或者 WhereDatesBetween 方法，过滤一个集合的取值结果。

```
OrderCollection orders = new OrderCollection();
orders.BetweenAnd(Order.Columns.OrderDate, new DateTime(1980, 1, 12),
DateTime.Now);
orders.Load();
```

在这里的 Where 方法作用的结果类似于 SubSonic Query Where 方法（这个方法将在下面的部分进行介绍）

Queries

Beyond all the basic filtering available in Fetch methods and collection filtering, SubSonic also provides a way to dynamically build SQL queries. To get started, create a new Query object, using the CreateQuery method.

```
SubSonic.Query query = Product.CreateQuery();
```

or pass it the table name

```
SubSonic.Query query = new SubSonic.Query(product.TableName);
```

or pass it the table schema

```
SubSonic.Query query = new SubSonic.Query(Product.GetTableSchema());
```

The last option will prevent SubSonic from loading the table information from the database, eliminating an extra call.

Note: You can also get the table schema by calling the static method BuildTableSchema or the property Schema.

除了上面介绍的在 Fetch 方法中用到的用于过滤查询结果的方法和在集合 collection 中使用的过滤方法外，SubSonic 同时提供了一种方式用于动态的生成 SQL 查询。这种情况下需要通过使用 CreateQuery 方法实例化一个 Query 实例。

```
SubSonic.Query query = Product.CreateQuery();
```

或者可以将表名作为参数传递给方法。

```
SubSonic.Query query = new SubSonic.Query(product.TableName);
```

也可以将数据表的 schema 作为参数传递给方法

```
SubSonic.Query query = new SubSonic.Query(Product.GetTableSchema());
```

最后一种方法可以放置 SubSonic 从数据库中读入表的具体信息，免除了再调用一个另外的方法。

注释：也可以通过调用静态方法 BuildTableSchema 或者使用 Schema 属性来获取数据表对象的 schema.

Running the Query

Executing the query follows the same rules as executing ADO.NET queries. The Execute method runs the query and returns nothing.

```
query.Execute();
```

The ExecuteDataSet method returns a DataSet.

```
GridView1.DataSource = query.ExecuteDataSet();
```

The ExecuteReader method returns a DataReader.

```
GridView1.DataSource = query.ExecuteReader();
```

And finally, the ExecuteScalar returns a single value.

```
Label1.Text = query.ExecuteScalar().ToString();
```

执行查询语句

在 SubSonic 中执行 query 查询与在 ADO.NET 中执行 query 查询所遵循的要求是相同的。当使用 Execute 方法执行 query 查询时运行查询但不返回任何值。

```
query.Execute();
```

ExecuteDataSet 方法返回一个 DataSet

```
GridView1.DataSource = query.ExecuteDataSet();
```

ExecuteReader 方法返回一个 DataReader

```
GridView1.DataSource = query.ExecuteReader();
```

ExecuteScalar 返回一个单一值

```
Label1.Text = query.ExecuteScalar().ToString();
```

Ordering

You can order a query by setting the OrderBy property to either ascending or descending and providing a column name.

```
query.OrderBy = SubSonic.OrderBy.Asc(Product.Columns.ProductName);
```

or

```
query.OrderBy = SubSonic.OrderBy.Desc(Product.Columns.ProductName);
```

Unfortunately at this time there is no way to order by multiple columns.

排序

可以通过设置 query 的 OrderBy 属性为升序或者降序来对 query 的查询结果进行排序。当然，这种情况下需要提供一个列名作为参数。

```
query.OrderBy = SubSonic.OrderBy.Asc(Product.Columns.ProductName);
```

或者

```
query.OrderBy = SubSonic.OrderBy.Desc(Product.Columns.ProductName);
```

很不幸的是，我们现在还没有一种根据多行的结果排序的方法。

Filtering

Columns

By default, the query will return all columns in the table. This can be changed by supplying a comma delimited list of columns to the SelectList property.

```
 query.SelectList   =   Product.Columns.ProductName   +   ",   "   +
Product.Columns.SupplierID;
```

Rows

The query will also return all rows in the table. This can be changed several ways, the easiest of which is the Top property, which can be set to return only the given number of rows.

```
query.Top = "10";
```

(I'm not sure the reasoning, but this value needs to be a string instead of an integer.) Filtering by a date range is possible with the AddBetweenAnd method which takes a column name, start date, and end date. This method can be called multiple times to limit by more than one column.

```
 query.AddBetweenAnd(Order.Columns.OrderDate, new DateTime(1980, 1, 1),
DateTime.Now);
```

Or you can do the same thing with non-date values using AddBetweenValues.

```
query.AddBetweenValues(Product.Columns.ProductName, "A", "F");
```

The final and most powerful method is AddWhere. Like the AddBetween methods it can be called multiple times to create a complete WHERE clause. AddWhere has several different constructors, the simplest of which takes a column name and matching value.

```
query.AddWhere(Product.Columns.SupplierID, 2);
```

You can also supply a Comparison instead of doing an exact match. (The complete comparison possibilities are Blank, Equals, GreaterOrEquals, GreaterThan, LessOrEquals, LessThan, Like, NotEquals, and NotLike.)

```
query.AddWhere(Product.Columns.SupplierID,
SubSonic.Comparison.GreaterThan, 2);
```

SubSonic also supports the concept of paging data by setting the PageSize and PageIndex properties.

```
 query.PageIndex = 2;
 query.PageSize = 5;
```

过滤查询结果

列

在默认情况下，query 将会返回查询结果的所有行。如果只想要得到过滤查询结果某几列的结果，可以将 SelectList 属性设置为用逗号间隔的列名的形式。

```
 query.SelectList   =   Product.Columns.ProductName   +   ",   "   +
Product.Columns.SupplierID;
```

行

默认是，query 同样也返回查询的所有行。有几种方式可以让 query 只返回其中的部分行。最简单定的方式是通过设置 Top 属性，来限制只返回顶部的几行。

```
 query.Top = "10";
```

（注意，此处的参数是 string 形式，而非 int 形式）

可以通过使用 AddBetweenAnd 方法来限制查询结果中某列的值在某个日期的范围之内。需

要的参数是一个列名，起始和截止日期。该方法可以被多次调用，以限制多列的日期范围。

```
 query.AddBetweenAnd(Order.Columns.OrderDate, new DateTime(1980, 1, 1),
DateTime.Now);
```

当参数并非日期类型是，可以使用下面的方法 AddBetweenValues

```
 query.AddBetweenValues(Product.Columns.ProductName, "A", "F");
```

最后的一个，也是功能最强大的一个方法是 AddWhere.就像是 AddBetween 方法一样，它可以被多次调用，以形成一个完整的 Where 句型约束。AddWhere 类有几个重载的构造函数，其中最简单的一个需要的参数是一个列名和一个此列中要进行匹配的值。

```
 query.AddWhere(Product.Columns.SupplierID, 2);
```

此处的比较可以是一个进行比较的符号，而不一定是一个具体的匹配值。(具体的比较符号可以是空值，相等，不等，大于，大于等于，小于，小于等于，不匹配某个字符串。)

```
 query.AddWhere(Product.Columns.SupplierID,SubSonic.Comparison.Greate
 rThan, 2);
```

SubSonic 也支持对查询结果的分页处理，如果要进行分页，需要设置相应的 PageIndex 和 PageSize 属性。

```
 query.PageIndex = 2;
 query.PageSize = 5;
```

Updating and Deleting

It is possible to issue insert, update and delete commands through the query object by setting the QueryType property. (Your options are Delete, Insert, Select, Update).

```
query.QueryType = SubSonic.QueryType.Insert;
```

If you want to add a specific update setting, such as setting all records to a specific value, you can use the AddUpdateSetting method, which can be called multiple times.

```
query.AddUpdateSetting(Product.Columns.UnitsInStock, 100);
```

更新和删除

通过设置 query 对象的 QueryType 属性，可以实现插入，更新，删除数据库的命令。(你的可选项包括 Delete, Insert, Select, Update）.

```
 query.QueryType = SubSonic.QueryType.Insert;
```

同时也可以多次是使用 AddUpdateSetting 方法

```
 query.AddUpdateSetting(Product.Columns.UnitsInStock, 100);
```

Aggregate Functions

Three aggregate functions are included as static methods. To get a column average call GetAverage with the table and column name, and an optional Where object.

```
SubSonic.Query.GetAverage(Product.Schema.Name, Product.Columns.UnitPrice.ToString());
```

You can also do the same for GetCount and GetSum.

使用集合方法

Query 类包含了 3 个静态方法，用于集合方法的调用。如果要求得一列的平均值，要调用 GetAverage 方法，将表和列名作为方法的参数出入。同时，可以加入一个可选的 Where 对象作为参数。

```
SubSonic.Query.GetAverage(Product.Schema.Name, Product.Columns.UnitPrice.ToString());
```

对于 GetCount 和 GetSum 方法，有相似的使用方法。

Commands

If the querying power of SubSonic falls short for your needs, you can still use the existing functionality and extend it by accessing the commands that are being built before execution. (These can also be very helpful when debugging.) The query object has four commands: BuildCommand, BuildDeleteCommand, BuildSelectCommand and BuildUpdate that all return QueryCommand objects, as well as the GetSql method that returns the raw SQL.

```
string sql = query.GetSql();
```

Each auto-generated object also has the ability to return a QueryCommand by calling one of four methods: GetInsertCommand, GetSelectCommand, GetUpdateCommand and GetDeleteCommand.

```
 SubSonic.QueryCommand                    cmd                    =
product.GetInsertCommand(User.Identity.Name);
```

命令

如果你觉得 SubSonic 的 query 类的查询能力还不能满足需求，你可以在使用现有的方法之前，通过写入一些命令来扩展这些方法的功能。（这个功能在 debug 的时候也很有用）query 对象有四个命令可以被利用， 他们是： BuildCommand, BuildDeleteCommand, BuildSelectCommand, BuildUpdate.这些命令都会返回一个 QueryCommand 对象。GetSql 方法将返回原 sql 语句。

```
 string sql = query.GetSql();
```

每一个自动生成的对象同样能过通过调用下面四种方法之一：GetInsertCommand, GetSelectCommand, GetUpdateCommand, GetDeleteCommand 来返回一个 QueryCommand 对象。

```
 SubSonic.QueryCommand                    cmd                    =
product.GetInsertCommand(User.Identity.Name);
```

Stored Procedures

Some tasks are just too complicated for dynamic query building and/or require a greater level of control. To handle this, SubSonic supports stored procedures. Each stored procedure will produce an equivalent static method in the class defined in the configuration file. By default this is SPs. Each method will have one parameter for each stored procedure parameter and return a StoredProcedure object.

```
 SubSonic.StoredProcedure sp = SPs.CustOrderHist(customerID);
```

The stored procedure can then either call Execute, ExecuteScalar, GetDataSet or GetReader to execute and return the data.

```
 GridView1.DataSource = sp.GetReader();
```

Or combining the two statements into one:

```
 GridView1.DataSource = SPs.CustOrderHist(customerID).GetReader();
```

You can also work with the QueryCommand by referencing the Command property.

存储过程

一些数据库的任务对于动态生成的 query 查询过于复杂。为了解决这个问题，SubSonic 提供了对存储过程的支持。每一个存储过程都会在类中生成一个对应的静态方法。而这个对应于存储过程的类就是 SPs。每一个存储过程的方法都会为存储过程的参数提供在方法中使用

的一个对应的参数，同时会返回一个 StoredProcedure 对象。

```
 SubSonic.StoredProcedure sp = SPs.CustOrderHist(customerID);
```

之后就可以调用生成的存储过程对象中的 Excute，ExcuteScalar，GetDataSet，GetReader 方法，来执行相应的操作和获得结果。

```
 GridView1.DataSource = sp.GetReader();
```

可以把上述两个步骤写到一起

```
 GridView1.DataSource = SPs.CustOrderHist(customerID).GetReader();
```

可以通过设置 Command 属性来实现各种 QueryCommand 的功能。

Scaffolding

The scaffold control is used to quickly create developer level admin pages. By dropping a single control on the page, you get a GridView and update controls. This control should appear in your Toolbox under SubSonic Components. Just set the TableName property and you're ready to go.

<cc1:Scaffold ID="Scaffold1" runat="server" TableName="Products"></cc1:Scaffold>

 It is also possible to apply some visual formatting through the EditTableCSSClass, EditTableItemCSSClass, EditTableLabelCSSClass and GridViewSkinID properties. And you can set the delete confirm message with the DeleteConfirm property.

Sacffolding

这种控制方法是为了快速的创建开发人员一级的管理页面。当托一个 control 控件到你的页面上时，你会得到一个 GridView 和 Update 控件。这个控件会出现在你的 vs 中的工具箱中，在 SubSonic 控件的列表中。使用这个控件，你只需要设置一下 TableName 属性即可。

<cc1:Scaffold ID="Scaffold1" runat="server" TableName="Products"></cc1:Scaffold>

一些可视化的格式的调整可以通过 EditTableCSSClass，EditTableItemCSSClass，EditTableLabelCSSClass 和 GridViewSkinID 属性来进行设置。通过对 DeleteConfirm 属性的设置，可以改变删除时的确认信息。

Code Generation Templates

There is one more way you can tweak SubSonic without changing the code. SubSonic creates the auto-generated classes by using standard text. By adding alternate text files to the templateDirectory defined in the configuration file, you can change this behavior. For a full sample list of these files, check out the CodeTemplates directory in the sample web site.

代码生成模板

可以在不用改变代码的情况下，改变 SubSonic 生成代码的格式。SubSonic 是根据一个标准的文本生成类的结构的。通过增加标准文件到配置文件的模板库中，你可以改变 SubSonic 生成代码的结构。你可以到实例网站的 CodeTemplates directory 中查询这些标准文件。

Conventions

Controllers

Instead of accessing the data functions directly, the Controller design pattern is recommended. To implement this, create a Controller class in the App_Code directory.

```
public class ProductController
```

Then add methods to retrieve the data using collections.

```
public ProductCollection GetAllProducts()
```

And to update the data.

```
public void UpdateProduct(Product product)
```

While there is nothing enforcing these rules, it will make it easier to insert business rules and re-use methods from a single location.

控制类 Controllers

如果要对数据进行修改等操作，除了可以直接修改数据类中的方法外，我们还推荐您使用控制器类 Controller 类的方法。如果要实现这种方法，可以在 App_Code 文件夹中创建的 Controller 类。（现在的版本中还用自己创建 Controller 类吗？不用了吧）

```
public class ProductController
```

之后在其中增加方法并返回 collections 类型的查询结果。

```
public ProductCollection GetAllProducts()
```

在修改数据时，可以创建下面的方法

```
public void UpdateProduct(Product product)
```

这种通过使用 Controller 类的方式来进行数据库的增上改查的方法不是必须的。但这种方式有一种好处，这将是在同一个地方加入业务逻辑层的规则和代码重用变得更简单。

Database

These conventions are pulled directly from the SubSonic documentation.

Every database table needs to have a primary key. You can't use SubSonic if this isn't the case.

Integer-based keys are preferable, for performance. I personally use GUIDs only as identifiers and not keys. This isn't required.

Every table should have some auditing ability built in, but this is not required. These fields are:

o CreatedOn (datetime)
o CreatedBy (nvarchar(50))
o ModifiedOn (datetime)
o ModifiedBy (nvarchar(50))

If you want to use logical deletes, you can by adding a field called "Deleted" or "IsDeleted"

• Table names should be singular

Column names should never contain reserved words (system, string, int, etc)

Column names should not be the same as table names

数据库

下面的文字直接来自 SubSonic 文档

在使用 SubSonic 之前，首先要确保数据库中每一张表都有一个主键。如果不符合这个规范，SubSonic 将不能正常使用。

从程序的运行效率角度来讲，最好能使用整数型的主键。我个人只会将 GUID 型用于

表中的一个唯一标示，而不用与主键。但这只是建议，不是必须的。
每张表最好有一些可以用来记录和审查数据的列，这同样不是必须的。这些字段是：
o CreatedOn (datetime)
o CreatedBy (nvarchar(50))
o ModifiedOn (datetime)
o ModifiedBy (nvarchar(50))
如果你想使用逻辑删除功能（并非直接从数据库中删除记录，而是将记录的某一字段进行标记，表示该数据已删除），你可以想表中加入"Deleted"或者"IsDeleted"字段。
•表名应为单数，列名不应该与表名相同，也不能将保留字作为列名，如system, string, int等。

Sample Web

The sample web that is downloaded along with the SubSonic source has been discussed periodically throughout this guide. The following is a list of what is provided.

App_Code\Utility.cs - Miscellaneous functions commonly needed when building web applications.

App_Themes - A default theme to get you started.

Dev\CodeTemplates - Text file representations of the templates used to generate classes. Use these as a starting point if you need to create custom templates.

Dev\DB - Sample database scripts for schema and data loading.

Dev\BatchClassGenerator.aspx - If your environment does not allow running in Full Trust mode, run this page to generate classes that can be compiled into the build. Add these classes to your project's App_Code folder and remove the build provider settings from the configuration file.

Dev\BatchScaffoldGenerator.aspx - Create scaffold pages automatically for each table.

Dev\ClassGenerator.aspx - Similar to BatchClassGenerator, but only creates code for a single class that is meant to be copy and pasted into a class file.

Scripts - SQL file for Northwind.

示例网站项目
这个示例已经通过这个指南被多次的讨论过了。他将和 SubSonic 被捆绑下载。下面的列表介绍了这个示例提供了什么。

App_Code\Utility.cs - 建立 Web Application 项目通常要使用到的各种方法。

App_Themes - 帮助你开始建立项目的默认的主题

Dev\CodeTemplates - 对自动生成类的模板的描述文件。如果你需要定制生成类的模板，可以参照这个文件。

Dev\DB - 用于操作 schema 和数据下载的示例数据库脚本。

Dev\BatchClassGenerator.aspx - 如果你的系统环境不能设置为在完全信任的模式下运行，请运行此页，然后将生成的类加入到 App_Code 文件夹，并且修改配置文件，删除 build provider settings 节点。

Dev\BatchScaffoldGenerator.aspx - 可以为每张表自动的建立 scaffold 管理页面

Dev\ClassGenerator.aspx - 作用和 BatchClassGenerator 类似，但是这个页面是用于给某一个类单独生成代码，以用于复制和粘贴。

Scripts - Northwind 数据库的 SQL 建立语句。

Starter Kits　启动套件

SubSonic Starter Kit　启动套件

From the CodePlex site, you can also download the SubSonic Starter Kit. To use this, from Visual Studio, select New -> Web Site and pick SubSonic Starter Site. This will give you a good starting template for your own site that includes the following.
你也可以从CodePlex网站下载SubSonic启动套件。若要使用此，从Visual Studio ，选择New -> Web Site，并选择SubSonic入门网站。这将给你的网站一个包括以下内容良好的开端模板。

- _Dev\ASPNET_Membership - Web pages for user and role membership editing.
  - 用户和任务成员编辑的网页。
- _Dev\Generators - The same generators found in the sample web site plus a MigrationGenerator page. Run this page and it will create SQL scripts containing your database schema, data or both.
  - 建立在示例网站加上 MigrationGenerator 页的相同的产生器。运行此网页，它会产生包含你的数据库架构，数据或两者都有的 SQL 脚本。
- _Dev\SampleQueries.aspx - Some SubSonic code samples.
  - 一些 SubSonic 代码样本。
- App_Code\TestCondition.cs - Some simple data validation.
  - 一些简单的数据验证。
- App_Code\Utility.cs - Same as the sample web.
  - 相同的样本网页。
- App_Themes\Default - Same as the sample web.
  - 相同的样本网页。
- images - Some useful, generic sample images such as a loading item and processing spinner.
  - 一些有用的，普通的样品图片，例如加载项目和加工微调。
- js - JavaScript helper files from Scriptaculous.
  - Scriptaculous 中的 JavaScript 帮助文件。
- Modules\ResultMessage.ascx - Format result message control.
  - 格式信息控制的结果。
- Default.aspx - Three column CSS formatted starting page.
  - 三栏的 CSS 格式化的开始页面。
- Login.aspx - Sample login page.
  - 示例的登录页面。
- PasswordRecover.aspx - Sample password recovery page.
  - 示例密码恢复页。
- Web.config - Pre-configured for SubSonic and Atlas (AJAX.NET).
  - SubSonic 和 Atlas（AJAX.NET）的预先设定。

Commerce Starter Kit　商务入门套件

The Commerce Starter Kit 2.0 is now available which also uses SubSonic.
采用SubSonic的商业入门套件2.0现在已经可用。