

VISUAL STUDIO HACKS™

Tips & Tools for Turbocharging the IDE



O'REILLY®

James Avery

HACK
#49

Generate Strongly Typed DataSets

Make DataSets a little smarter by using Visual Studio and the XSD tool to generate typed DataSets.

ADO .NET's DataSet objects provide powerful functionality when working with data. They allow you to easily scroll, filter, search, and sort data, as well as work with hierarchical data and its relationships. However, your typical DataSet object is untyped, meaning it doesn't really know and understand what it is storing. It's just a smart container for your data. You can make a DataSet object even smarter by turning it into a strongly typed DataSet object.

When working with an untyped DataSet, accessing the value of a row looks something like this:

```
String artistName =  
    ds.Tables["Artists"].Rows[0]["Artist"].ToString();
```

When working with a typed DataSet, you will have IntelliSense for the list of tables as well as the column names, and best of all, each property is stored in its correct type. This means you don't need to call ToString() or cast the type when pulling it out of the DataSet. Performing the same operation with a typed DataSet would look like this:

```
string artistName = ds.Artists[0].Artist;
```

Creating a strongly typed DataSet object is done one of three ways: by writing the code yourself, by letting Visual Studio help you, or by using the *xsd.exe* tool. Hand-writing the code to make strongly typed DataSet objects can be long and tedious, so it's best to use one of the other two methods to do the heavy lifting for you.

Let Visual Studio Help

Visual Studio provides a nice shortcut to creating your own strongly typed DataSet objects. Follow these simple steps:

1. Open your Visual Studio Project (or create a new one).
2. Select the project in the Solution Explorer and click the Show All Files button.
3. Right-click on the project in the Solution Explorer, click Add, and select Add New Item.
4. In the Add New Item dialog box, select the DataSet template. In the Name field, type the name of your DataSet (e.g., *Customers.xsd*) and click Open.

Generate Strongly Typed DataSets

You will see that the XSD file was added to your project. There are two things to note here. First, right-click on the XSD file and select Properties. This is where you can specify a custom tool to generate your strongly typed DataSet objects. By default, the custom tool is named `MSDataSetGenerator`, which is provided out of the box by Visual Studio (see Figure 6-8).

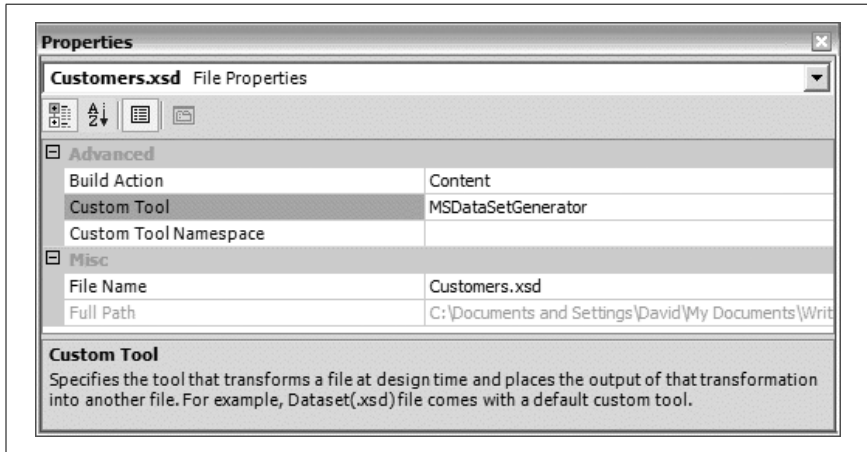


Figure 6-8. Properties of DataSet XSD file

The other thing to know is that the class file for the strongly typed DataSet object was autogenerated for you. You can see it by turning on Show All Files and expanding the XSD file. Figure 6-9 shows that the `Customers.cs` class file was autogenerated to create a DataSet object of type Customers.

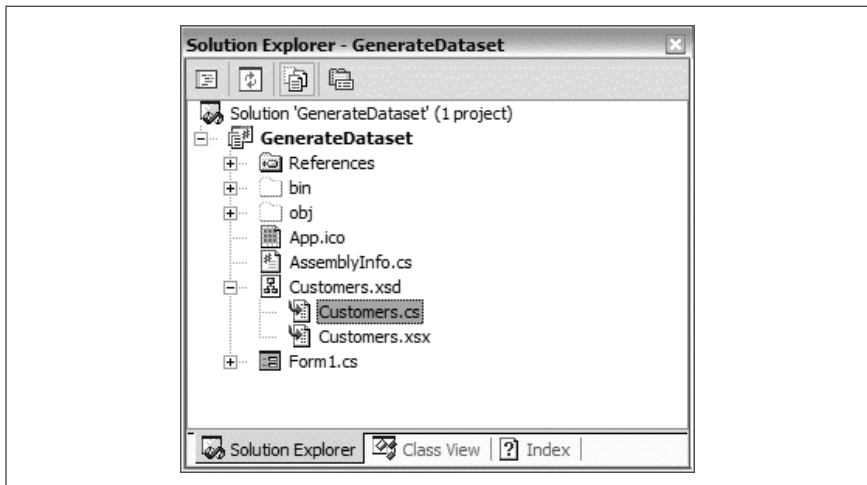


Figure 6-9. Autogenerated DataSet class file

Using the `xsd.exe` Tool

The .NET Framework SDK includes a tool named `xsd.exe` that can be used for a few different reasons; one of the handiest is for creating strongly typed DataSet objects. The best and easiest way to accomplish this is by first creating your own XML schema definition file (`*.xsd`). Because a DataSet object natively supports XML (as with everything else in .NET), using XSD files as the basis for your data and its structure makes the creation of strongly typed DataSet objects rather simple.

Assuming your XSD file is already created, using the `xsd.exe` tool will save you a significant amount of time because it generates your classes automatically based on the information contained in your XML schema definition.

To get started, let's first take a quick look at a sample XSD, which is based on the Customers table in the Northwind database:

```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="Customers"
targetNamespace="http://www.tempuri.org/Customers.xsd"
xmlns:mstns="http://www.tempuri.org/Customers.xsd"
xmlns="http://www.tempuri.org/Customers.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
attributeFormDefault="qualified" elementFormDefault="qualified">
<xs:element name="Customers" msdata:IsDataSet="true">
<xs:complexType>
<xs:choice maxOccurs="unbounded">
<xs:element name="Customer">
<xs:complexType>
<xs:sequence>
  <xs:element name="CustomerID" type="xs:string" />
  <xs:element name="CompanyName" type="xs:string" />
  <xs:element name="ContactName" type="xs:string" minOccurs="0" />
  <xs:element name="ContactTitle" type="xs:string" minOccurs="0" />
  <xs:element name="Address" type="xs:string" minOccurs="0" />
  <xs:element name="City" type="xs:string" minOccurs="0" />
  <xs:element name="Region" type="xs:string" minOccurs="0" />
  <xs:element name="PostalCode" type="xs:string" minOccurs="0" />
  <xs:element name="Country" type="xs:string" minOccurs="0" />
  <xs:element name="Phone" type="xs:string" minOccurs="0" />
  <xs:element name="Fax" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
<xs:unique name="Constraint1" msdata:PrimaryKey="true">
  <xs:selector xpath="//mstns:Customers" />
  <xs:field xpath="mstns:CustomerID" />
</xs:unique>
</xs:element>
```

```
</xs:schema>
```

Notice that the XSD is rather simple, but does contain some database schema information. For instance, by studying the *Customers.xsd*, you can see that the *CustomerID* and *CompanyName* fields are required but all other fields can be null and that the *CustomerID* field is the primary key.

To create a strongly typed DataSet object of type Customers, feed the *Customers.xsd* file into the *xsd.exe* tool and have it generate the Customers class for you. To do this, perform the following:

1. Open the Visual Studio .NET command prompt and browse to the directory where your XSD file exists.
2. Type the following at the prompt: **xsd.exe SchemaName.xsd /DataSet /language:CS**, where *SchemaName.xsd* is the name of your XSD file. In this example, the command line looks like this: `xsd.exe Customers.xsd /DataSet /language:CS`. For Visual Basic .NET, use **/language:VB**.
3. Look for your generated class file. In this sample, a file named *Customers.cs* will be created.

There are several options and flags for the *xsd.exe* tool, but the important flag for creating your strongly typed DataSet object is the `/DataSet` flag. This tells the tool to parse through the XSD file and autogenerate a strongly typed DataSet object for you. A view of the command line looks like this:

```
C:\>xsd.exe Customers.xsd /DataSet /language:CS
Microsoft (R) Xml Schemas/DataTypes support utility
[Microsoft (R) .NET Framework, Version 1.1.4322.573]
Copyright (C) Microsoft Corporation 1998-2002.
All rights reserved.
```

```
Writing file 'C:\Customers.cs'.
```

```
C:\>
```

Again, you can see the class file that was generated in this example was named *Customers.cs*. Taking a look at this class, you can see that the *xsd.exe* tool created a class named Customers that inherits from the DataSet class, thus creating a strongly typed DataSet object:

```
[Serializable()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Diagnostics.DebuggerStepThrough()]
[System.ComponentModel.ToolboxItem(true)]
public class Customers : DataSet {
```

Including all of the code here would be excessive—the generated class is more than 600 lines long, but it is important to note that the Customers

class is not the only class contained in this file. The following classes were also generated:

- CustomerDataTable
- CustomerRow
- CustomerRowChangeEvent
- CustomerRowChangeEventHandler

I would encourage you to examine the generated classes further to better understand how strongly typed DataSets work.

—*Dave Donaldson*