Working with local databases

26 Aug 2005 4:24 PM

Overview

In Visual Studio 2005 we added a number of features to help developers build and deploy applications that need a local data store. Here's a quick peek at how it works.

In order to work with a local database file, you can simply add the file to your project (e.g. using the Project/Add Existing Item... menu). We currently support adding SQL Server data files (.mdf), Jet (Access) data files (.mdb) and SQL Mobile data files (.sdf). Note that in order to be able to use .mdf files, you need to have installed SQL Server Express. With SQL Server Express installed, you will also be able to create new databases through 'Project/Add New Item…/Database'.

Once the database file is in the project, VS will do a few things:
1. It will automatically add a connection in the Database Explorer so you can edit the database schema or the data.
2. It will make sure that the connection strings are serialized using a relative path (more on this below).
3. The first time the file is added, VS will also launch the Data Source wizard to create a new typed dataset.

Full path vs relative path

One of the reasons why it was hard to work with database files before is that the full path to the database was serialized in different places. This made it harder to share a project and also to deploy the application. In this version, the .NET runtime added support for what we call the DataDirectory macro. This allows Visual Studio to put a special variable in the connection string that will be expanded at run-time. So instead of having a connection string like this: "Data Source=.\SQLExpress;AttachDbFileName=c:\program files\app\data.mdf"

You can have a connection string like this:"Data Source=.\SQLExpress;AttachDbFileName=|DataDirectory|\data.mdf"

This connection string syntax is supported by the SqlClient and OleDb managed providers.

By default, the |DataDirectory| variable will be expanded as follow:

- For applications placed in a directory on the user machine, this will be the app's (.exe) folder.
- For apps running under ClickOnce, this will be a special data folder created by ClickOnce
- For Web apps, this will be the App_Data folder

Under the hood, the value for |DataDirectory| simply comes from a property on the app domain. It is possible to change that value and override the default behavior by doing this:

```
AppDomain.CurrentDomain.SetData("DataDirectory", newpath)
```

For customizing the connection string at runtime, please see our team blog at:http://blogs.msdn.com/smartclientdata/archive/2005/07/25/443034.aspx

Where is my data? -- Understanding the file copy for desktop projects

One of the things to know when working with local database files is that they are treated as any other content files. For desktop projects, it means that by default, the database file will be copied to the output folder (aka bin) each time the project is built. After F5, here's what it would look like on disk

MyProject\Data.mdf

MyProject\MyApp.vb

MyProject\Bin\Debug\Data.mdf

MyProject\Bin\Debug\MyApp.exe

At design-time, MyProject\Data.mdf is used by the data tools. At run-time, the app will be using the database under the output folder. As a result of the copy, many people have the impression that the app did not save the data to the database file. In fact, this is simply because there are two copies of the data file involved. Same applies when looking at the schema/data through the database explorer. The tools are using the copy in the project, not the one in the bin folder.

If this copy behavior is not what you want, there are few ways to work around it:

1. If you select local database file in the Solution Explorer window, you will see a property called ‘Copy to Output’ in the Properties window. By default it is set to ‘Copy Always’ which means that on each build, the data files in the project folder will be copied to the output folder overwriting the existing data files if any. You can set this property to Copy Never and then manually put a copy of the data file in the output folder. This way, on subsequent builds, the project system will leave the datafile in the output folder and not try to overwrite it with the one from the project. The downside is that you still have two copies so after you modify the database file using the app, if you want to work on those changes in the project, you need to copy it to the project manually and vise-versa.

2. You can leave the data file outside the project and create a connection to it in Database Explorer. When the IDE asks you to bring the file into the project, just say no. This way, both the design-time and the run-time will be using the same data file but the downside is that the path in the connection string will be hard coded and therefore it’ll be harder to share the project and deploy the app. Before deploying the app, just make sure to replace the full path in the settings with a relative path.