Re: Deployment + Vista

Source:
http://www.tech−archive.net/Archive/DotNet/microsoft.public.dotnet.framework.adonet/2007−03/msg00239.html

- *From*: "Steven Spencer \(Spinalogic\)" <Spence−Spinalogic@xxxxxxxxxxxxxxx>
- *Date*: Sun, 18 Mar 2007 15:41:46 +1000

You use business objects AND a dataset? Have I understood your pattern correctly?

I don't have an issue changing the data adapters around, other than its an extra redirection that has to be done for every data adapter. I do not use business objects behind the dataset, I simply wish to create the sql data adapter and use it. The problem is it's connection string is always hard coded to an application setting.

I'm trying to play nice with UAC, and given that we need logo certification for our system, we have to. Standard user can't write to the application directory, it will fail. I however know of the commonapplicationdata environment directory so that will fit the bill nicely for the location of the file.

My problem is still how to tell the designer when it makes the sqldataadapter to get its connection string from somewhere OTHER THAN an application setting. Once I have that one worked out, everything will be fine.

I've looked and I think I can set it to 'None'. At least that way at testing would get a failure if the setting was set. That would do.

Cheers for your input.

As for astonishment, you wouldn't be the only one. Vista
"Earl" <brikshoe@xxxxxxxxxxxxxxxxx> wrote in message
news:%23TNEznQaHHA.1300@xxxxxxxxxxxxxxxxxxxxxxx

> No, the user does not edit the xml directly. In my apps, they have a Server settings form. The changes they make there are persisted to the xml (I've seen no evidence of Vista discarding the changes made to those files). Fundamentally, an xml file is simply a text file with structure. Vista doesn't discard your Notepad files, so I don't see how this could possibly occur with a file created by your app. Talk about violating "the rule of least astonishment"! If it is indeed doing this with app.config, I guess it's a good thing I do not use that for settings.

In any event, when the app starts up for the first time, it checks for the server and database. If not found, it automatically loads the Server settings form. The user enters those settings, clicks save, and the login dialog loads. When the app starts again, it checks the serversettings.xml, makes a connection and intead of loading the server settings form, it loads the login dialog.

You have several options from there. You could read the file anytime you needed a connection string, or you could set up static properties to hold the server and database names, then construct a string on the fly.

I cannot take credit for re−inventing the wheel. I've learned bits and pieces of how to make this work dynamically as I've went along and then constructed the best situation that worked for me. You mileage may vary, but I think this is the general approach you are looking for.

Here is the simple .xml file (it has a corresponding .xsd):

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
<ServerSettings>
<SQLServer>.\SQLEXPRESS</SQLServer>
<Database>LastChanceTen</Database>
</ServerSettings>
</NewDataSet>
```

Here is the code to pull the settings out (I've trimmed out the Try−Catch):

```
public string GetConnectionString()
{
// this method returns the concatenated string plus sets the
individual name and server public properties
string AppPath;
bool bolSchemaFileExists = false;
DataSet dsIn = new DataSet();
bool bolXMLFileExists = false;

Assembly MyAssembly = Assembly.GetExecutingAssembly();
string F_ConfigLocation;
F_ConfigLocation = MyAssembly.Location;

AppPath = Path.GetDirectoryName(F_ConfigLocation);

//reading in the schema first
string schemaFile = AppPath + "\\serversettings.xsd";

if (File.Exists(schemaFile) == true)
{
bolSchemaFileExists = true;
```

```
System.IO.FileStream fsSchema = new
System.IO.FileStream(schemaFile, FileMode.Open, FileAccess.Read);
XmlTextReader trSchema = new XmlTextReader(fsSchema);

dsIn.ReadXmlSchema(trSchema);
//failure to close the streamreader will result in an
exception if deleting or overwriting
fsSchema.Close();
}
else { bolSchemaFileExists = false; }

//reading in the actual file data
string dataFile = AppPath + "\\serversettings.xml";

if (File.Exists(dataFile) == true)
{
bolXMLFileExists = true;

FileStream fsData = new FileStream(dataFile,
FileMode.Open, FileAccess.Read);
XmlTextReader trData = new XmlTextReader(fsData);
dsIn.ReadXml(trData);
fsData.Close();
}
else { bolXMLFileExists = false; }

if (bolSchemaFileExists == true && bolXMLFileExists ==
true)
{
m_strSQLServerName =
dsIn.Tables["ServerSettings"].Rows[0]["SQLServer"].ToString();
m_strDatabaseName =
dsIn.Tables["ServerSettings"].Rows[0]["Database"].ToString();
}

dsIn.Reset();

m_strConnection = "data source=" + m_strSQLServerName + ";" +
"initial catalog="+m_strDatabaseName+";integrated
security=SSPI;";

return m_strConnection;
}
```

I haven't hard−coded a connection string in a couple of years. As I noted in the earlier post, the strongly typed datasets stand alone. Set up the dataset structure and Merge the data from your objects. Then pass the changes from the datasets back to your object methods as a datatable, again, the data adapters use the dynamically created connection string instead of being hard−coded to ANYTHING.

"Steven Spencer (Spinalogic)" <Spence−Spinalogic@xxxxxxxxxxxxxxxx> wrote in message news:etBPaDQaHHA.1388@xxxxxxxxxxxxxxxxxxxxxxx

> I'm talking about the user being able to edit the server connection setting without manually editing XML.
>
> Secondly my issue is with the SqlDataAdapters being hard bound to the connection string which is an application setting. I'm aware that you can change them to use a separate scheme and then have to call some method to set the con string for you. This is an ommission on the part of the framework and my question is does anyone know of a supported way at the designer to redirect where this app string goes.
>
> As for the manual way, if we do that where do you plan to put the server config files? As I said you can't put them in the application directory, as Vista will simply cache the change and throw it away when your program exits, not persisting it (This is what I was trying to say in my last post). I'm trying to allow the end user to change the settings in a dialog in an easy way, and make our deployment much simpler eg First run of the application spawns a dialog saying please enter dbase settings. You COULD use the users directory, but then if you change which user you wish to access the application, then you need to re enter obscure settings for an end user, which should really already be there.
>
> You certainly wont have a problem at deployment of a solution such as yours especially with xcopy, but do you change the server config per MSI you generate? or have a dialog in your msi (which is how we have done it as the MSI will allow you to edit the app directory, even in Vista)
>
> Any ideas?

"Earl" <brikshoe@xxxxxxxxxxxxxxxx> wrote in message news:%235DyoPMaHHA.5080@xxxxxxxxxxxxxxxxxxxxxxx

> > I'm guessing something didn't get cut and pasted into that reply, cuz
> > you lost me with the early part of your reply.
> >
> > But interestingly enough, this past week I had to deploy an app onto a
> > small network. A few days later one of the offices had a new Vista
> > installation. Within a matter of minutes, I had XCopied the app and the
> > server.xml and server.xds files into a directory, fixed an unrelated
> > network connectivity issue (the server is on a different machine), and
> > we were off and running. Have had absolutely no problems with the Vista

machine at all in the scenario I described to you.

I'm not a big fan of using the app.config file to handle my server
settings. It's been a few years since I quit trying to use the app.config file for that purpose, mainly because I wanted a more control
over the file and file structure that was being handled in order to save
and retrieve those settings.

I'm not sure how "Vista will temp your changes" because these are being
hard−written into the "server.xml" file (NOT the app.config file).
Indeed it seems to me that this is exactly the scenario where a file
that Microsoft does not control would be a better solution.

"Steven Spencer (Spinalogic)"
<Spence−Spinalogic@xxxxxxxxxxxxxxx> wrote
in message
news:OlXlbZFaHHA.5080@xxxxxxxxxxxxxxxxxxxxxx

> Your description vista wise just fell over at
> "(into the app
> directory), then give the user an input
>
> > panel so that they can
> > specify any changes" Vista
> > will temp your
> > changes to the app directory
> > and then discard them upon
> > exiting the
> > form.
>
> Secondly:
>
> > Then I can call a
> > GetConnectionString()
> > function that calls the
> > retrieval function in order to
> > put together the connection
> > string
> > something like:
> > m_strConnection = "data
> > source=" +
> > m_strSQLServerName +
> > ";initial catalog=" +
> > m_strDatabaseName +

";integrated security=SSPI;"

I'm well aware of this pattern, and it is in fact the one I was using previous to discovering the app.config setting.

My only problem with it is it is error prone for a multi developer environment, if someone updates the data layer they may forget that call and the app will look like it works fine in testing until we deploy...

I guess my question was exactly what I said: Is there a way at design time to tell the sqldataadapters where to get its connection string from? OR a way of using an application setting that is actually writeable at runtime.

Either way, I do not wish to have to have to change the setting during
"Earl" <brikshoe@xxxxxxxxxxxxxxxxx>
wrote in message
news:u0AtES6YHHA.1580@xxxxxxxxxxxxxxxxxxxxxxx

> I know nothing about Vista,
> but you do not have to bind
> your connection
> string to your strongly typed
> datasets.
>
> In the display layer of the
> app, I keep the
> strongly−typed datasets
> empty, then when I need
> data, I call .Merge() on the
> strongly typed
> datasets in order to return
> datatables from the
> appropriate data layer
> classes. Within those classes
> is where I open dynamically
> created
> connections.

I deploy a small .xml and .xsd file with the default server name and database settings (into the app directory), then give the user an input panel so that they can specify any changes. Separately, I create a class with separate functions to save and retrieve the settings, and with a couple of properties for the server and database name. Then I can call a GetConnectionString() function that calls the retrieval function in order to put together the connection string something like: m_strConnection = "data source=" + m_strSQLServerName + ";initial catalog=" + m_strDatabaseName + ";integrated security=SSPI;"

As you can see, the only hard−coded aspect of the string is simply the security. If necessary, you could even modify that, although if you are deploying with SqlExpress, you already know what the security setting will be.

"Steven Spencer (Spinalogic)" <Spence−Spinalogic@xxxxxxxxxxxxxxxx> wrote in message news:eqTquLrYHHA.984@xxxxxxxxxxxxxxxxxxxxxxx

> We use strongly typed datasets, and thus our

connection string is bound to an application setting.

Currently we can deploy and alter the app.exe.config file at installation time to successfully install, and use some basic XML editing to allow the target database to be changed.

Writing to the application directory in vista will throw compatibility errors, and not persist the changes at all.

What do we do? Is there a better way of getting the connection string in your application, or a way of telling the designer where to get your connection

setting
from??

.