



Flip the Switch

An Introduction to Visual Studio Lightswitch for Visual FoxPro Developers

Eric Selje
Geeks and Gurus, Inc.
Madison, WI
Voice: 608/213-9567
Email: ejselje@geeksandgurus.com
Twitter: EricSelje

LightSwitch is Microsoft's latest development tool and it is aimed squarely at the same market who may have considered Visual FoxPro in the past. LightSwitch creates attractive Silverlight applications which can run on Windows, Mac, and theoretically Linux too.

If you're creating line-of-business applications, you should definitely have LightSwitch on your radar.

Yet Another Development Environment?

Microsoft's recent roll-out of this new tool called Visual Studio LightSwitch has created quite a bit of confusion. Marketing is having a hard time explaining where LightSwitch fits into the development ecosystem: Is it a successor to Visual FoxPro? Access? Is it Visual Studio for Dummies? Is it aimed at developers? Power users? Tech-savvy secretaries? How does it compare with Microsoft's other recent tools, like ASP.Net MVC 3 (or 4), Silverlight, and C# 5.0 with WinRT. What does it really even *do*?

In this paper I'll introduce application development from the Visual FoxPro developer's perspective. I'll point out the many similarities LightSwitch has with Visual FoxPro, what LightSwitch adds that Visual FoxPro does not, and what's remarkably different and downright odd.

Hopefully this whitepaper will clear up some of these issues for you, the Visual FoxPro developer. You can ignore Microsoft's marketing team and decide for yourself if LightSwitch is the right tool for your next project.

What is LightSwitch?

Let's start with what Jason Zander, corporate v.p. for the Visual Studio team, said on the day LightSwitch was officially released (July 26, 2011)ⁱ:

Visual Studio LightSwitch 2011 is a simplified, self-service development tool that enables you to create business applications quickly and easily for the desktop and cloud. It starts with the premise that most business applications consist of data and the screens that users interact with. LightSwitch simplifies attaching to data with data source wizards or creating data tables with table designers. It also includes screen templates for common tasks so you can create clean interfaces for your applications without being a designer. Basic applications can be written without a line of code. However, you can add custom code that is specific to your business problem without having to worry about setting up classes and methods.

Parts of that sure sound a lot like Visual FoxPro right? Data and screens are FoxPro's thang. But it also sounds a bit like Access when he says you don't have to write a line of code. What he doesn't mention is that LightSwitch generates Silverlight applications, which makes LightSwitch sound a bit like a fancy app wizard. It also can use SQL databases or "local" data, so we're back to Visual FoxPro's domain again. It creates 3-tier applications, which sounds like it steps on ASP.Net MVCs turf. Lastly, you can publish to Azure which is just plain awesome.

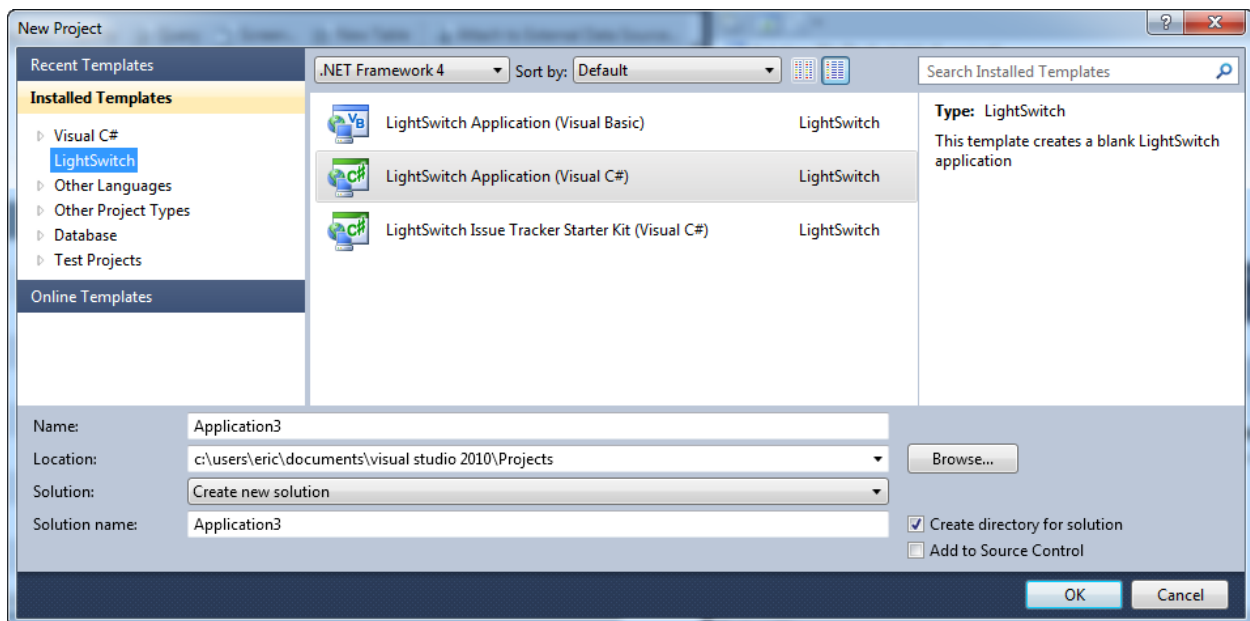
There's actually so much to say about LightSwitch that it would take a long book to discuss all of its features, but let's start working on an actual application. In this whitepaper I'll point out things of interest to Visual FoxPro devs and then we can decide for ourselves what LightSwitch is...

Getting Started

If you haven't yet, download Visual Studio LightSwitch. It's available on MSDN for those of you who subscribe to that, or you can get a free 30 day trial version from Microsoft's download center. **Visual Studio LightSwitch is not free!** Even if you already own Visual Studio 2010, LightSwitch does cost a little bit of money (around \$300, which is less than Visual FoxPro used to cost). If you don't already have Visual Studio, the download does install a copy that allows you to create LightSwitch projects. If you already have Visual Studio, LightSwitch becomes a new template you can choose when starting new projects.

LightSwitch also installs a copy of SQL Express on your machine to manage local databases. It does this regardless of whether you already have a copy of SQL Server installed.

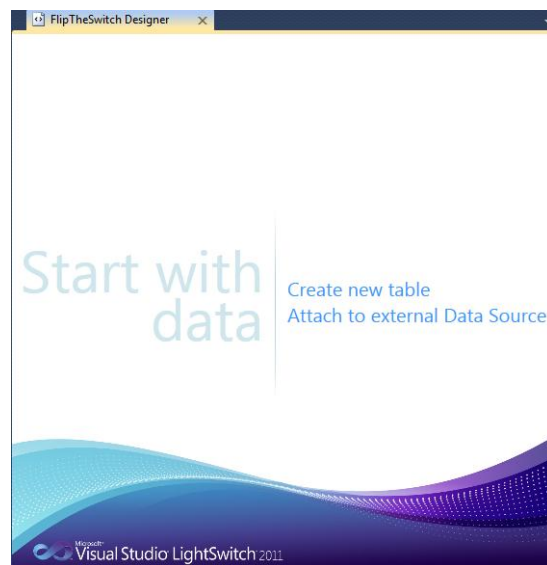
After LightSwitch is installed, you create a new LightSwitch application by selecting File, New Project. You can choose either Basic or C# as your language of choice, and for my examples I'm going to choose C# because I find it very FoxPro-like (in fact, some of the FoxPro team worked on C#)



After you've given your application a good name and click Ok, you'll come across the first screen that lets you know we're dealing with something special - LightSwitch is encouraging us to "Start with data." Yes! This is something we Visual FoxPro developers can get behind!

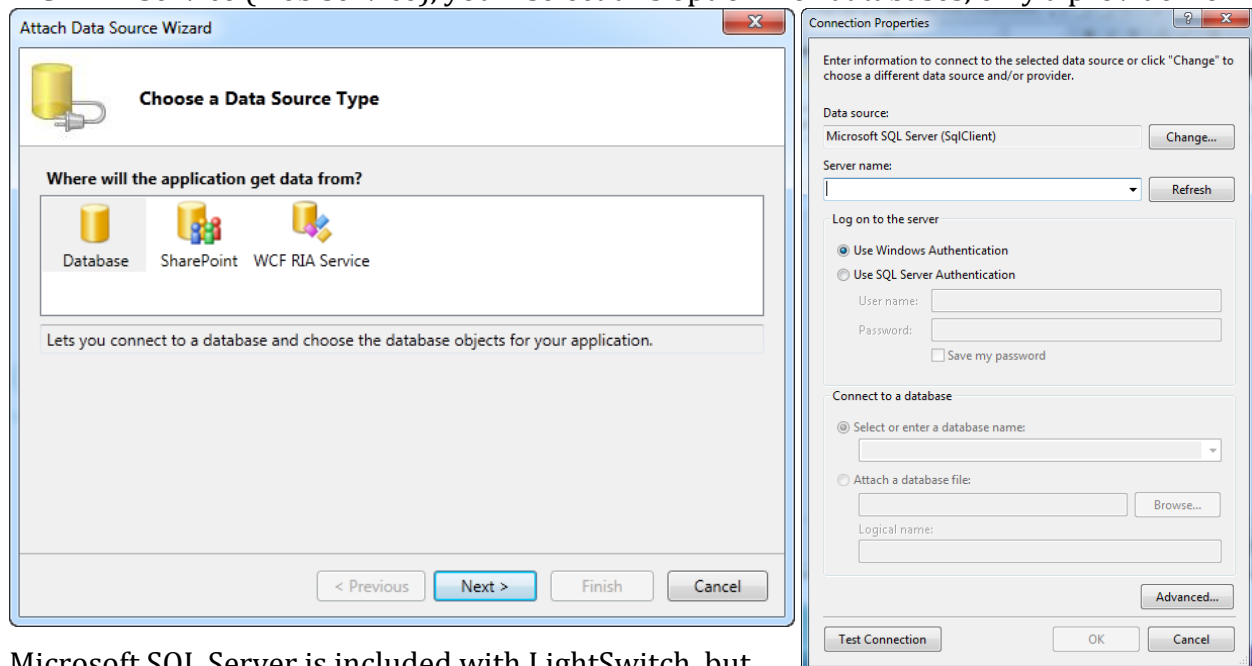
Create new table

This option will bring up a database designer that doesn't look all that much different than FoxPro's, although it has some nice additions. The tables actually get created in a local copy of SQL Express while you're developing the application. When you deploy the application you can decide if you want to deploy the database as well or connect to something else. (We'll discuss that much further down, in the deployment section).



Attach to external Data Source

If you've already got data in a SQL Server database, a SharePoint "list," or accessible via WCF RIA Service (web service), you'll select this option. For databases, only a provider for



Microsoft SQL Server is included with LightSwitch, but 3rd parties are creating some very intriguing ADO.Net data providers. For example, RSSBusⁱⁱ has created data providers for Facebook, Twitter, QuickBooks, Google Data (Gmail, etc.), Salesforce, PowerShell and more already using the Extension Manager in Visual Studio.

What about FoxPro Databases?!

The obvious question we FoxPro developers will want to know is "Can I use Visual FoxPro databases with LightSwitch?" The short answer is "Yes" but it's a bit of a hassle and it doesn't make a lot of sense. If you're starting a new application that needs to talk to local data, just use the SQL Express that comes with LightSwitch. If you're migrating an existing application to LightSwitch and want to keep your data in DBCs, create WCF RIA services that wrap around your existing business objects. This will provide a layer that LightSwitch

can use to talk to your data. Your data will then remain on your server, and your LightSwitch app will call out to the services that will fetch and return the data back to your application. Creating WCF RIA services is beyond the scope of this session, but it's actually not terribly difficult. Also be sure to check out the "Linq to VFP" project on CodePlex.ⁱⁱⁱ

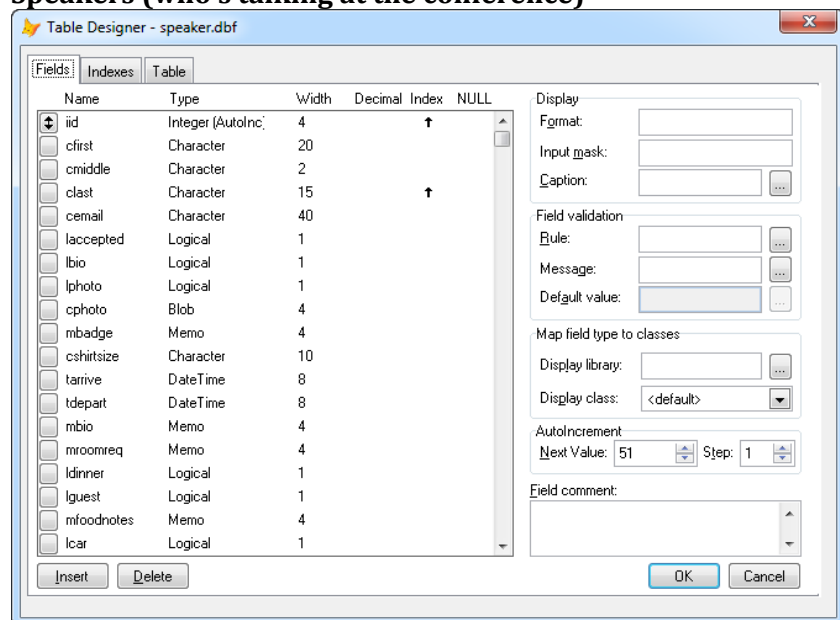
Having a desktop LightSwitch application that talks to FoxPro databases that reside on the same machine doesn't make a lot of sense to me, but if you can provide an argument for why that would be useful I'd love to hear it.

Entities

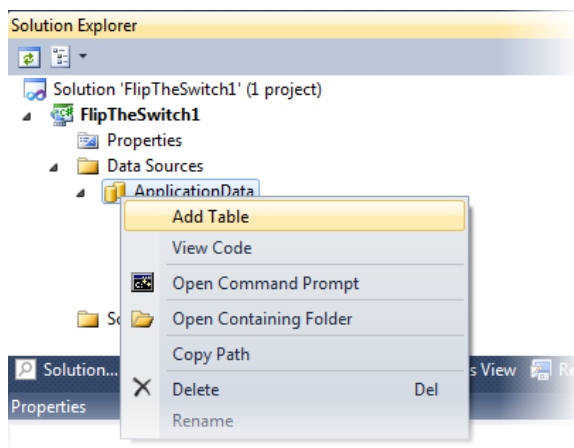
When you add a data source to LightSwitch, you're not actually communicating directly with the databases like you are when you use SQL Pass-through in Visual FoxPro. LightSwitch instead inserts a layer called the Entity Data Model between your data and your user interface. It's a bit like remote views in FoxPro, but the entities are actually classes so you get a lot more functionality.

Let's create some data from scratch and you'll get a better sense of what I mean. For the example I'm going to create the very tables that are used to maintain the Southwest Fox data. Let's start with who's speaking:

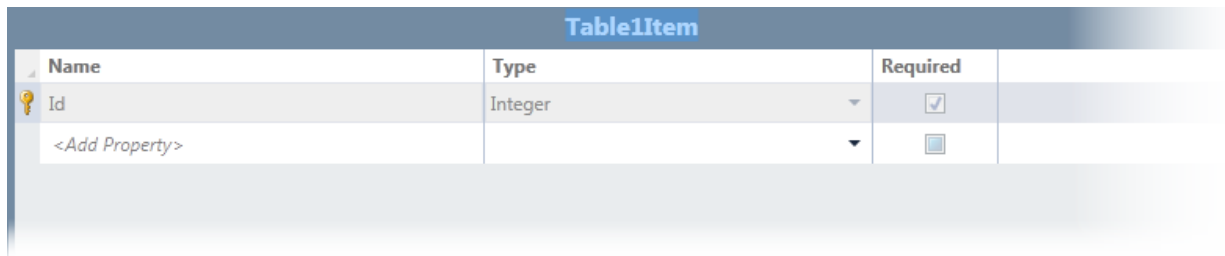
Speakers (who's talking at the conference)



Add a table to your solution (either by clicking "Create New Table" from the welcome screen or right clicking "Add Table" on the Application Data in the Solution Explorer).



It may not be obvious that the cursor is sitting on the name of the table (Table1Item) and you need to change it. Keep the name singular (you'll see why in a moment).



Name	Type	Required
Id	Integer	<input checked="" type="checkbox"/>
<Add Property>		

Notice how *Id* gets automatically inserted and cannot be deleted. It's an autoincrementing primary key field, which every table needs if it's designed well. It's checked as *Required* which cannot be changed.

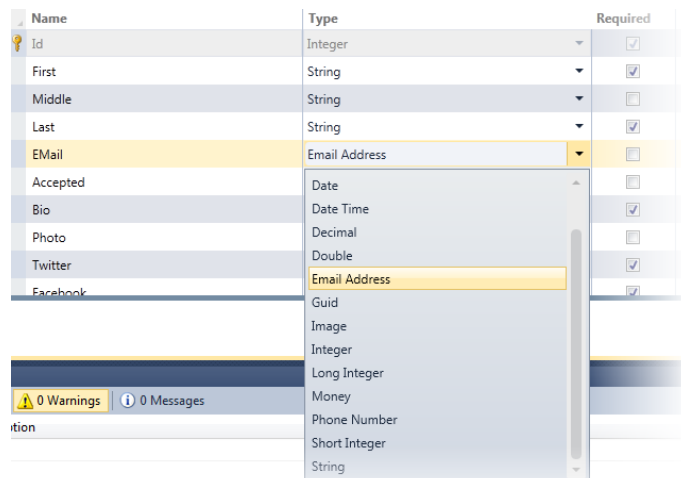
The first field we can actually add is the first name field, "First". In FoxPro we'd have to try to figure out the maximum length of this field, but in LightSwitch there's no "length" property. All Strings are variable length, with an effective maximum length of 4,000 characters. So even though the "Middle" field may only need to be one character, we also give that a type of String; The same with "Last." Let's check First and Last as *Required* and you'll see in a minute how that comes into play.

Email Addresses & Phone Numbers

Next we come to the email address, and this is where it becomes clear that while it looks like we're defining the schema of a table, we're actually setting the properties of an entity. In the dropdown choices for Type, there is an option for "Email Address." Now we know that there is no SQL Server field type called Email Address, so what's going on here? LightSwitch is going to define the field as a String at the data level, but will automatically detect whether the value in this field looks like a valid email address! Y

That is very handy, especially for those of us who struggle with regular expressions. See Appendix A for a complete list of data types supported in LightSwitch and Appendix B for how SQL Server data types are represented in LightSwitch entities.

The "Phone Number" data type gives similar functionality, except formatted for phone numbers. It's able to recognize multiple different countries' phone number formats, and you can tailor exactly which formats are permitted in your data by clicking "*Phone Number Formats...*" in the properties window.



Name	Type	Required
Id	Integer	<input checked="" type="checkbox"/>
First	String	<input checked="" type="checkbox"/>
Middle	String	<input type="checkbox"/>
Last	String	<input checked="" type="checkbox"/>
Email	Email Address	<input type="checkbox"/>
Accepted	Date	<input type="checkbox"/>
Bio	Date Time	<input checked="" type="checkbox"/>
Photo	Decimal	<input type="checkbox"/>
Twitter	Double	<input checked="" type="checkbox"/>
Facebook	String	<input checked="" type="checkbox"/>

Images

The “Photo” field in the Speaker table will be of type image, which as we’ll see in a moment gives you some nice functionality in the User Interface. Unlike FoxPro’s “General” or “Blob” field, the Image data type does a good job of storing photos with very little friction.

Computed Fields

LightSwitch entities also give us the ability to create “computed” fields. In the *Speakers* field, for example, let’s create a field called Name which is the concatenation of the First, Middle, and Last properties. Check the “Is Computed” box in the field’s properties, and you’ll get the partial class definition for the Speaker class, which is mostly empty at this point. Fill in the new method with this code:



```
public partial class Speaker
{
    partial void Name_Compute(ref string result)
    {
        // Set result to the desired field value
        StringBuilder cReturn = new StringBuilder();
        cReturn.AppendFormat("{0} {1} {2}", First, Middle, Last);
        result = cReturn.ToString();
    }
}
```

(Notice the computed field functions pass in the result by reference, so that’s the name of variable where you want to store your return value.)

Another value that you might like to have calculated automatically is “Age.” If you add a Date field for “Birthday”, you can create a computed field for Age with this code:

```
partial void age_Compute(ref int result)
{
    // Set result to the desired field value
    DateTime today = DateTime.Now;
    result = today.Year - birthday.Year;

    if (today.Month < birthday.Month || (today.Month == birthday.Month && to
day.Day < birthday.Day))
        result--;
}
```

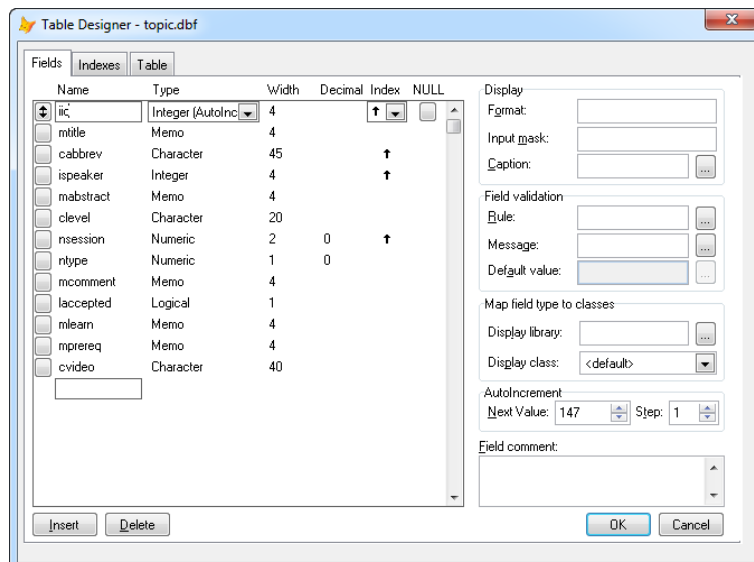
Partial Class?

If you haven't coded much in C#, you may not have seen the Partial Classes yet. There is no real equivalent in Visual FoxPro, but it's very cool. It allows you to split the class definition among several (though usually just two) different files on disk. . The bulk of the code, which might be changed via an upgrade or code generation, for example, goes into one file. Your customized parts go in a separate file (such as our computed field methods above). Class definitions in C# are straight text files, not binary files like in Visual FoxPro.

By doing this you don't lose your modifications when you regenerate the class because it won't overwrite your file. This is somewhat similar to subclassing, except we're not actually creating that extra abstraction layer.

Choice Lists

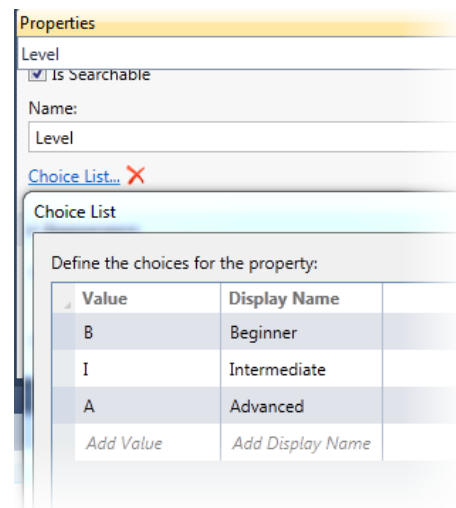
Let's add the *Topic* table to our application. That looks like this in Visual FoxPro:



In this table, we want to restrict the “Level” field to either be “Beginner,” “Intermediate,” or “Advanced.” The figure to the right shows what that looks like. In the User Interface, you'll soon see this gives us a drop down list control for the *Level* field with these options available to us.

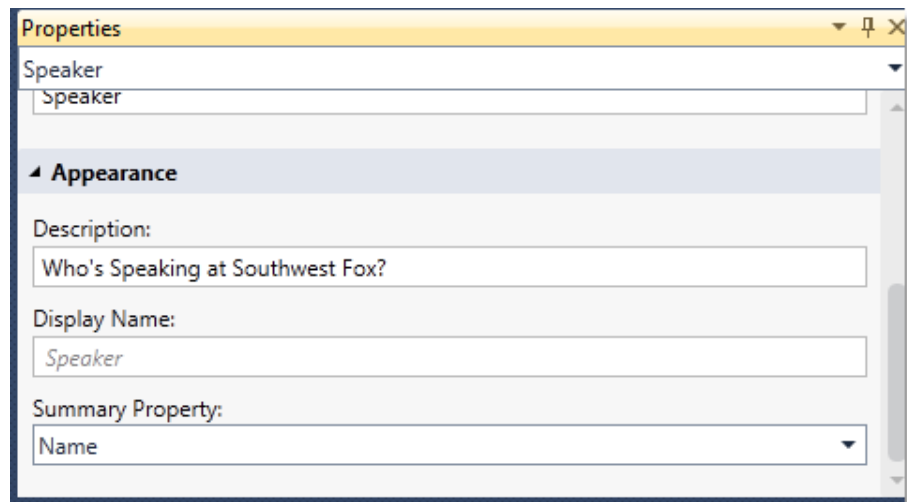
You should now be able to add these fields to your new Topic entity without any trouble, as these field types are all pretty straightforward.

One powerful feature that is available in LightSwitch is the “Choice List”. You can restrict the value of a field to an enumerated set of values by selecting “Choice List...” on the field's property sheet.



Summary Property

Every entity has one property (field) that can be designated the “Summary Property.” That field is then used on screens that show a list of all the entities. For example if you’re showing the list of Speakers to choose from, you would want to show the full name of the speaker rather than just their first or last name, so we’ll designate our Computed property “Name” as the Summary Property for the speakers table, using the property sheet.

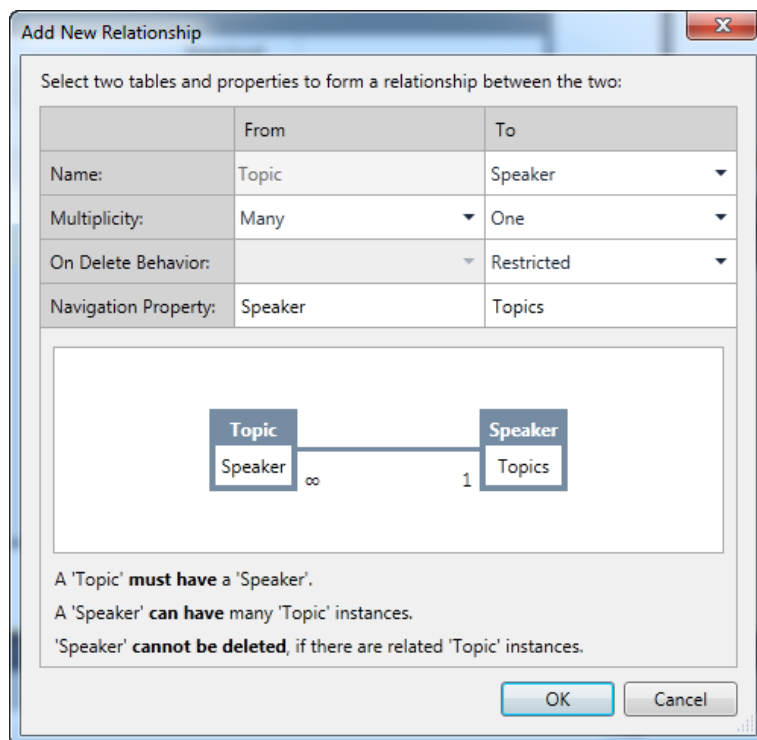


The screenshot shows the 'Properties' window for the 'Speaker' entity. The 'Appearance' section is expanded, showing fields for 'Description' (containing 'Who's Speaking at Southwest Fox?'), 'Display Name' (containing 'Speaker'), and 'Summary Property' (set to 'Name').

Relationships

While you can define relationships in a Visual FoxPro database container and set up referential integrity, LightSwitch takes that even further. Adding a relationship to an entity actually makes the related entity appear as a *collection* when you select an entity.

I’ll show you what I mean: In the Topic table, right click on the left margin and choose “Add a Relationship.” A dialog comes up that looks similar to what’s in Visual FoxPro:



The 'Add New Relationship' dialog box shows the configuration for a relationship between the 'Topic' and 'Speaker' tables. The 'Name' is 'Topic', 'Multiplicity' is 'Many' to 'One', 'On Delete Behavior' is 'Restricted', and 'Navigation Property' is 'Speaker' to 'Topics'. A visual diagram shows 'Topic' (Speaker) connected to 'Speaker' (Topics) with a cardinality of ∞ to 1. Below the diagram, text states: 'A 'Topic' must have a 'Speaker'.', 'A 'Speaker' can have many 'Topic' instances.', and 'Speaker cannot be deleted, if there are related 'Topic' instances.'

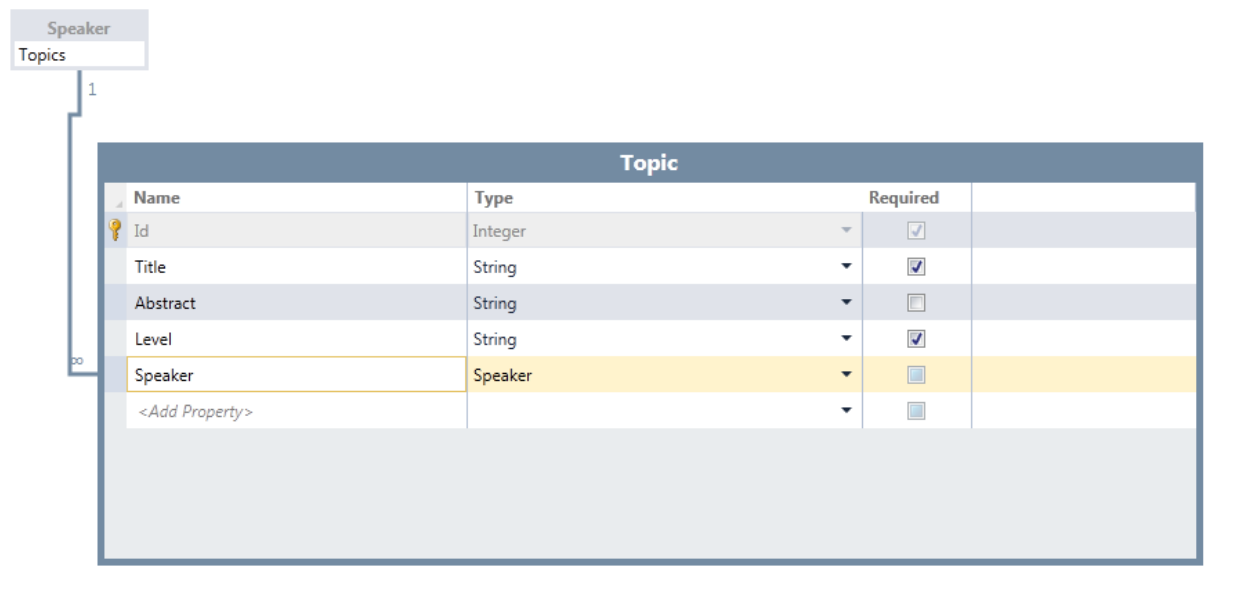
	From	To
Name:	Topic	Speaker
Multiplicity:	Many	One
On Delete Behavior:		Restricted
Navigation Property:	Speaker	Topics

A 'Topic' must have a 'Speaker'.
A 'Speaker' can have many 'Topic' instances.
'Speaker' cannot be deleted, if there are related 'Topic' instances.

Select *Speaker* in the dropdown list for the related table. You can define the On Delete behavior as either Cascading (delete all topics if the speaker is deleted) or Restricted (you cannot delete a speaker if they have any topics). I like how LightSwitch spells what is going to happen right out for you.

The Visual FoxPro dev might be wondering why there are no referential integrity options for “On Update” or “On Insert” like we have in VFP. Because LightSwitch forces an immutable primary key field on every table, that Id field will never be updated so there is no need for an “On Update.” Similarly, you simply cannot insert a foreign key into a LightSwitch table that doesn’t link back to a primary key in the parent table. This is good database design and I’m happy with this restriction.

Your entity-relationship design now looks like this. Notice the new “Field” in Topic named Speaker that appears to have a data type called Speaker. This is in-fact exactly the case. An entity is actually a data type that can be used just like any other data type.



If you DoubleClick the *Speaker* entity above, you’ll see the other side of the relationship. The Speaker table now has a field that contains a collection of Topic entities:

Speaker		
Name	Type	Required
Photo	Image	<input type="checkbox"/>
Twitter	String	<input checked="" type="checkbox"/>
Facebook	String	<input checked="" type="checkbox"/>
Name	String	<input type="checkbox"/>
birthday	Date	<input checked="" type="checkbox"/>
age	Integer	<input type="checkbox"/>
Topics	Topic Collection	<input type="checkbox"/>
<Add Property>		

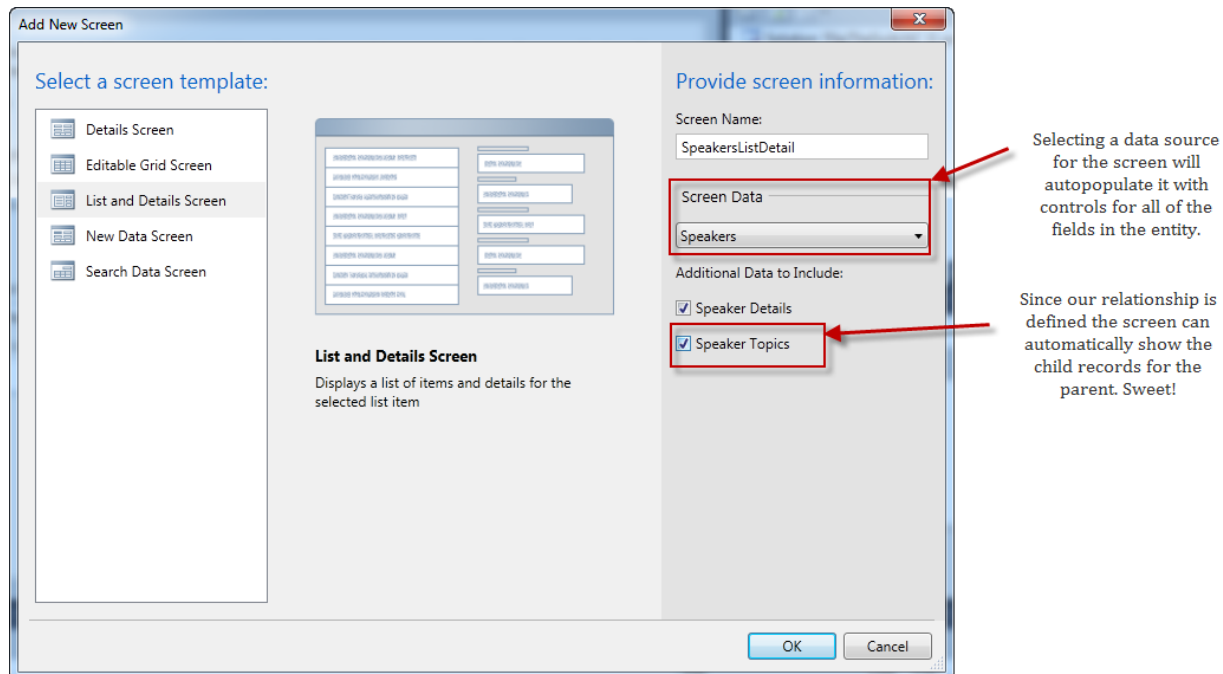
That's pretty cool in and of itself, but the big payoff comes when we're creating screens, which we'll do in the next step.

Screens

One thing LightSwitch and Visual FoxPro both have in common is the concept of data-entry forms, which they both call Screens. After dealing with *Forms* for the last few projects, it's good to be dealing with *Screens* again.

Right Click on Screens in the Solution Explorer and select “Add a Screen.”

LightSwitch comes with five screen templates to choose from, and you can create your own or download templates that others have created. Of the ones available in the templates, the “List and Details Screen” provides a great starting point for looking at our data.

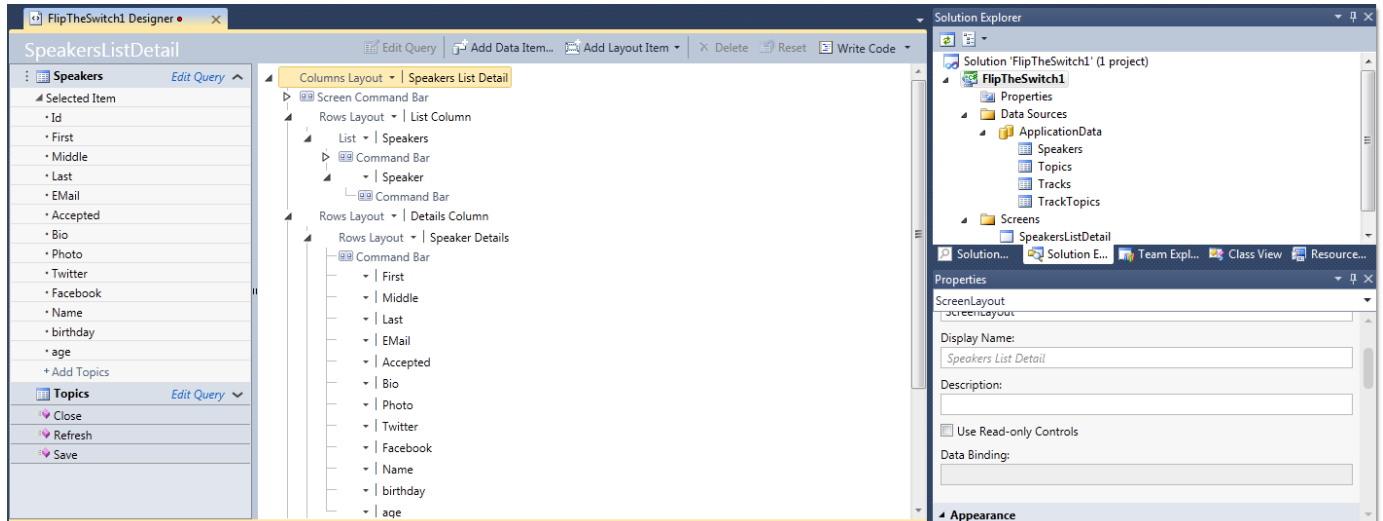


The “Details Screen” template is good for looking at one entity at a time, such as a speaker and all of their topics. The “Editable Grid Screen” is much like a browse window. The “New Data Screen” is much like the “Details Screen,” customized for adding records. The “Search Data” screen all the entities in a read-only grid more suitable for finding records.

If you’ve set up relationships, the details screens will automatically show the child records in a grid if you tell it to. Notice in the figure above that the “Speaker Topics” checkbox is selected.

WYS IS NOT WYG

After using the WYSIWYG screen editor introduced in Visual FoxPro 2.0, the VFP developer is likely to be appalled, at least initially, by what passes for a screen designer in LightSwitch.



In the left hand column we get the entities used by the screen.

In the center column there's some sort of hierarchical outline of the controls.

In the right hand column there's a property sheet for the selected control.

But where is the layout? How do you precisely place the objects where you want them?

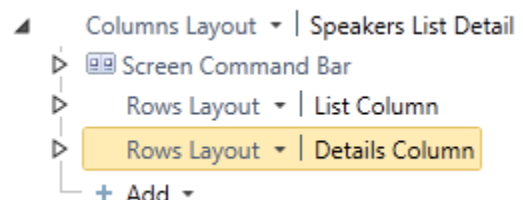
How do you get to “design mode”?

This is definitely not the screen designer that you're currently used to.

Instead of dictating where things go with Top and Left and Width and Height, LightSwitch screens use “flows” to layout the screen. This is similar to the way HTML lays out pages, and directly analogous to Silverlight's WPF layout. And while this may seem incredibly limiting at first, it actually frees the developer from a lot of restrictions. You no longer have to worry about a “resize” event or what the dimensions or resolution of the user interface might be, because this is all taken care of for you automatically regardless whether you're running on a large monitor or a portable device.

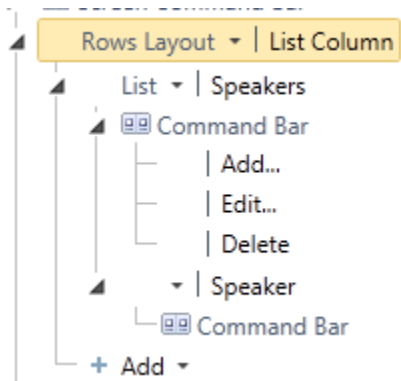
Tip: I find it helpful to collapse the hierarchy in order to figure out the layout. By doing that in the SpeakerListDetail screen, it becomes clearer that this screen is set up in a Columns Layout, so each subcontrol of the page will be placed side by side.

There are three controls on the page so you might think you'll get three columns but it is actually only two columns (the Command Bar is part of the layout itself): the List Column and the Details Column.

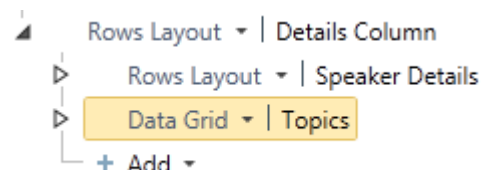


The List Column is configured in a Rows Layout, so the flow within that column moves downwards instead of across the page. There is in fact only one control in that column, the list control called *Speakers*, so it could have actually flowed either way.

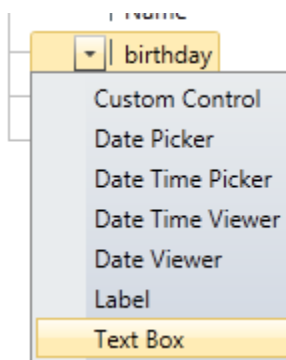
Notice how each entry in the outline of the screen follows a pattern of *[Layout/Control] / Property*. The top level control here is *Rows Layout / List Column*, and the next control is *List / Speakers*. After the command bar though, there's a control that doesn't specify which control to use, it just says */ Speaker*. When no specific control is designated, LightSwitch will automatically use the *default control* for the item. In this case, *Speaker* is the entity containing the speaker data, and inside of a list entities are represented by the Summary Property we selected when we initially setup the Entity, which for *Speakers* was the Computed property we called *Name* for the full name of the speaker.



Details column is also set up in a Rows Layout and contains only two controls, the *Speaker Details* and the *Topics* that the speaker is giving, represented by a Data Grid control.

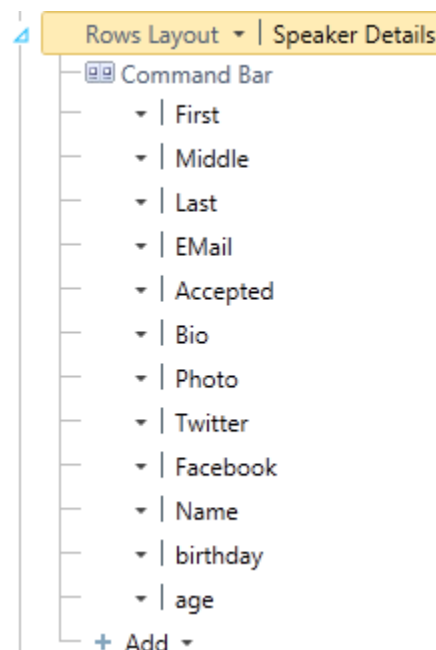


The Speaker Details also flow in a Row (top to bottom) Layout, with a control for every property in the Speaker entity. Notice how all the entities use the default control. We can override that if we would like to. If you've purchased 3rd party custom controls for LightSwitch (there are many available) or written your own controls (beyond the scope of this paper, but entirely possible to do), you can select that control either on the Property Sheet or by clicking on the arrow next to the field name. For example, date fields such as *Birthday* will use the date picker control, but if we wanted to switch that to a textbox we could do that.



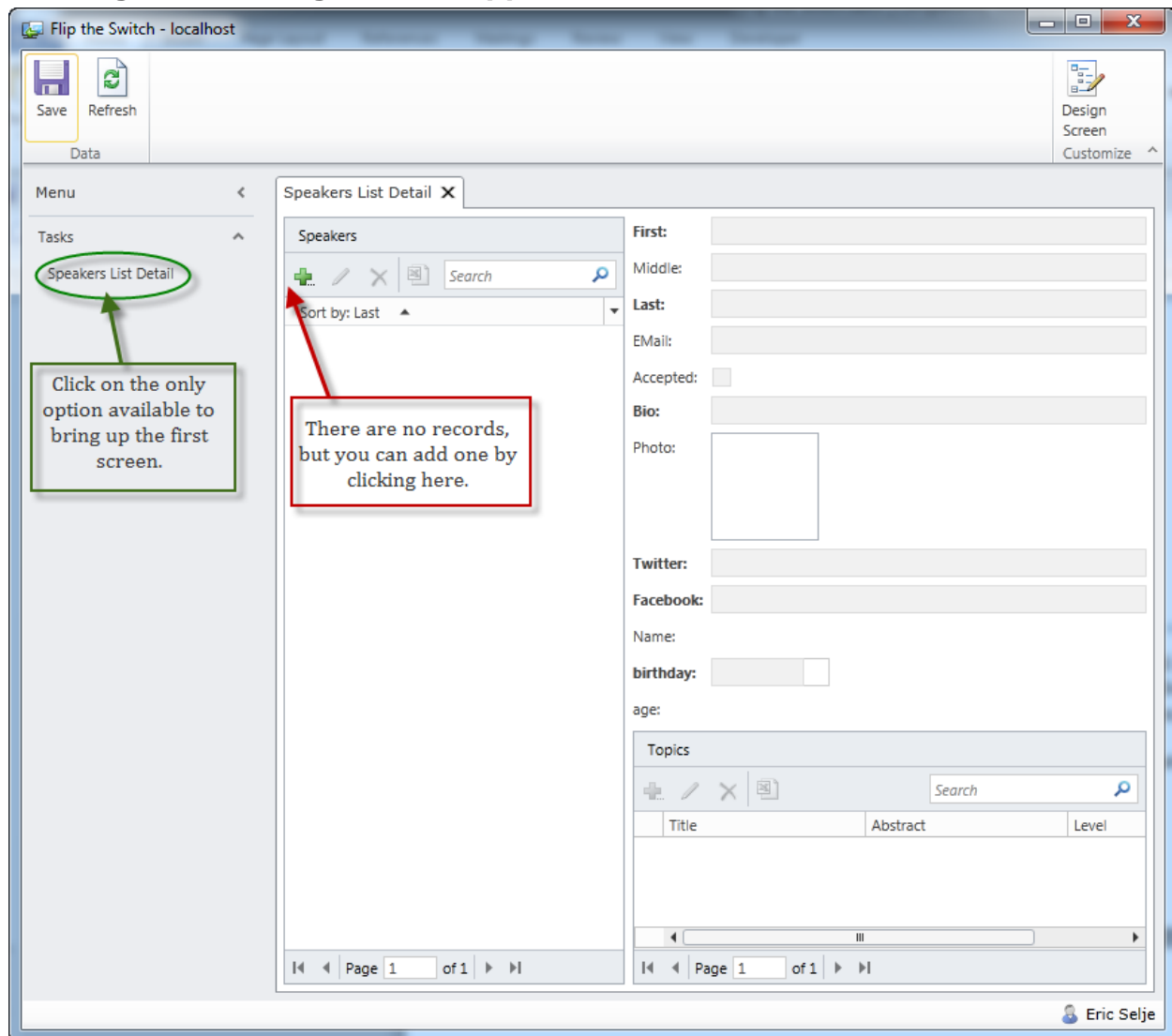
Wait, LightSwitch has a built-in Date Picker control? Sweet!

The other control in the Speaker Details layout is the Data Grid that shows the Topics that the selected speaker is presenting. We don't have to do any coding to make this work; LightSwitch uses the intelligence built into the Speaker Entity to show the correct children.



Without doing any modifications to the screen, hit **Ctrl+F5** to see what our application looks like for the very first time:

Viewing our First LightSwitch Application



The application is attractive, with a customizable Command Bar along the top and navigation along the left-hand side. Clicking the “Speaker List Detail” menu option brings up our new screen. There are no records yet, but click the green “Plus” icon brings up this:

Wait, we didn’t design this screen, did we?
No, we did not.

LightSwitch can auto generate attractive screens with intelligent defaults with our intervention or coding. That’s a very nice feature! (And of course we can override this default screen with a custom screen we write.)

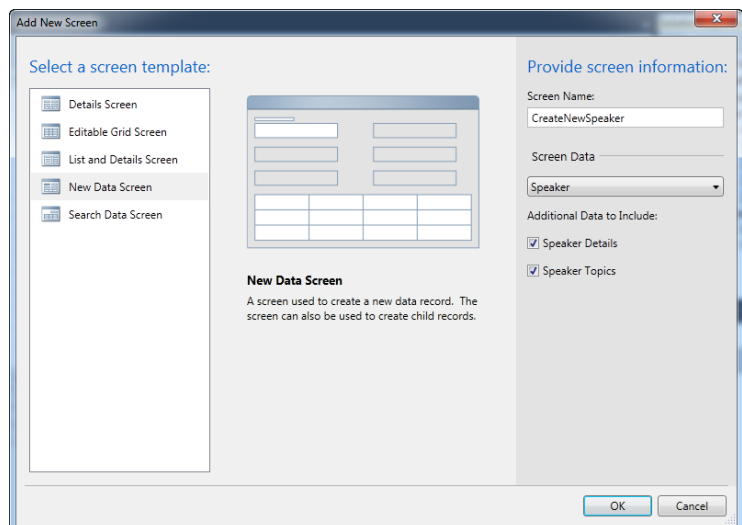
Here are a few other nice things that LightSwitch does for us:

- The fields with bold captions are the ones we marked as “Required” in the entity definition, and LightSwitch automatically enforces that in the User Interface.
- The *Email* field doesn’t require a value, but if one is entered LightSwitch will automatically verify that what’s entered looks like a valid email address.
- The *Accepted* field has a three-way checkbox for Yes, No, and Unanswered (or null).
- The *Photo* field has built-in functionality to load images from a file on your hard drive. Once a picture is loaded it shows the image automatically.
- The *Birthday* field has a date picker control, as mentioned.
- The *Name* field is updated when the First, Middle, or Last field is updated, and the *Age* field is computed automatically and updates when the Birthday is changed. No `Thisform.Refresh()` required!
- If we’d had a field that we specified a Choice List for, we’d automatically get a drop-down list with those options.
- All of the object-relational mapping all happens automatically. We don’t have to write any code to insert the data from the Entity into SQL or retrieve it back out. We can add the parent record and children on the form before clicking Save and LightSwitch keeps track of the new primary key and inserts it into the child table automatically.



This default screen isn’t perfect. Even though we told the Speaker entity that the *Bio* field is 4,000 characters long, LightSwitch didn’t consider making the textbox “multi-line,” or the equivalent of FoxPro’s EditBox control. This is easily achieved on custom screens by tweaking the settings for the field on the property sheet, but cannot be modified on the default screen.

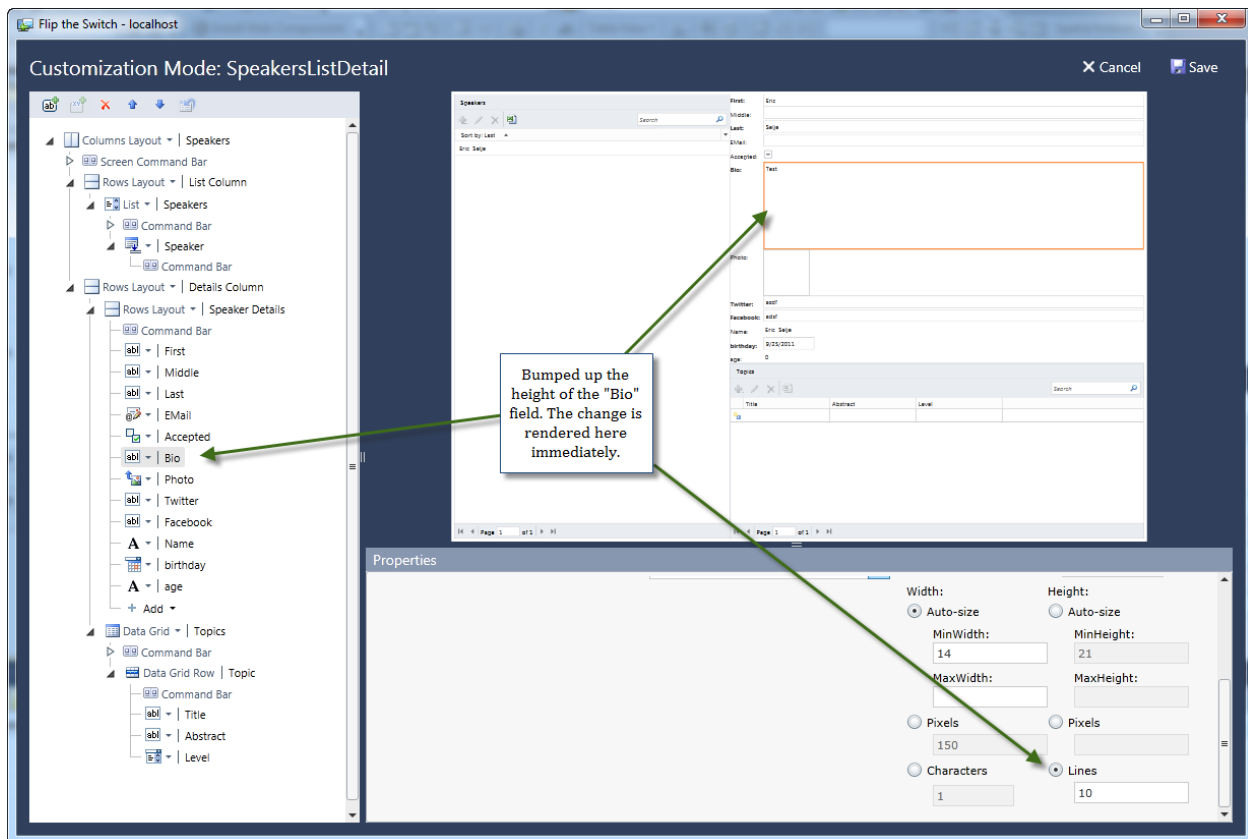
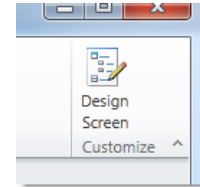
To create a custom screen that gets invoked when a new Speaker Entity is wanted, choose *Project, Add Screen* from the menu and select a “New Data Screen”. Specify *Speaker* in the Screen Data pulldown.



But wait, WYSI kinda WYG

LightSwitch actually *does* have a WYSIWIG editor, but oddly you don't get it while you're developing the screens in Visual Studio.

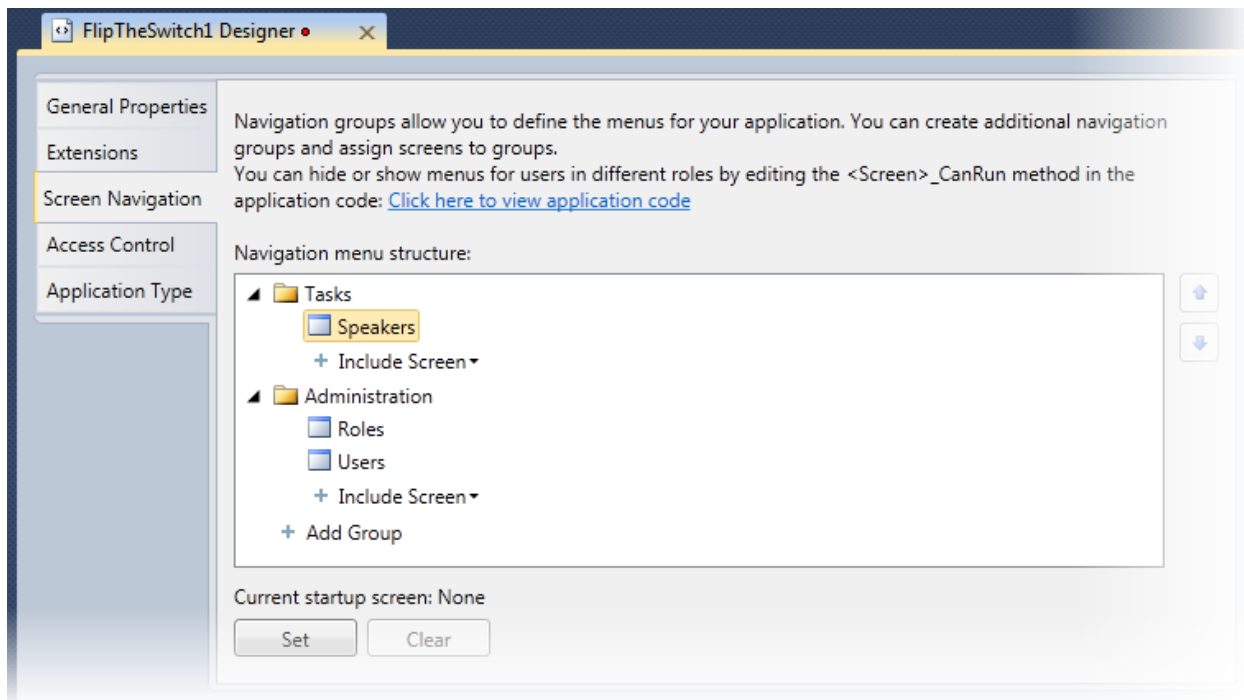
In the upper-right hand corner of screens that are running in development mode, there is a link to "Design Screen." Rather than taking you back to Visual Studio's normal screen designer, it pulls up hybrid WYSIWIG screen designer that shows an actual rendering of the screen with data:



In the example above, I changed the height of the *Bio* field to 10 lines (effectively changing it from a "textbox" to an "editbox" and the change is immediately rendered in the screen preview.

Navigation

When a new screen is added to your application, LightSwitch automatically adds a link to that screen in the Navigation pane on the left-hand side of the app. You may not want a link to every screen right from the main navigation, or you may want to re-order the choices. You do that by selecting *Project, <ProjectName> Properties* from the menu and selecting the *Screen Navigation* tab.



There is currently no way to create hierarchical navigation using the native controls.

If you'd like to have a particular screen appear automatically when the application starts, you can select that here as well by highlighting the screen name and clicking Set.

Application Security

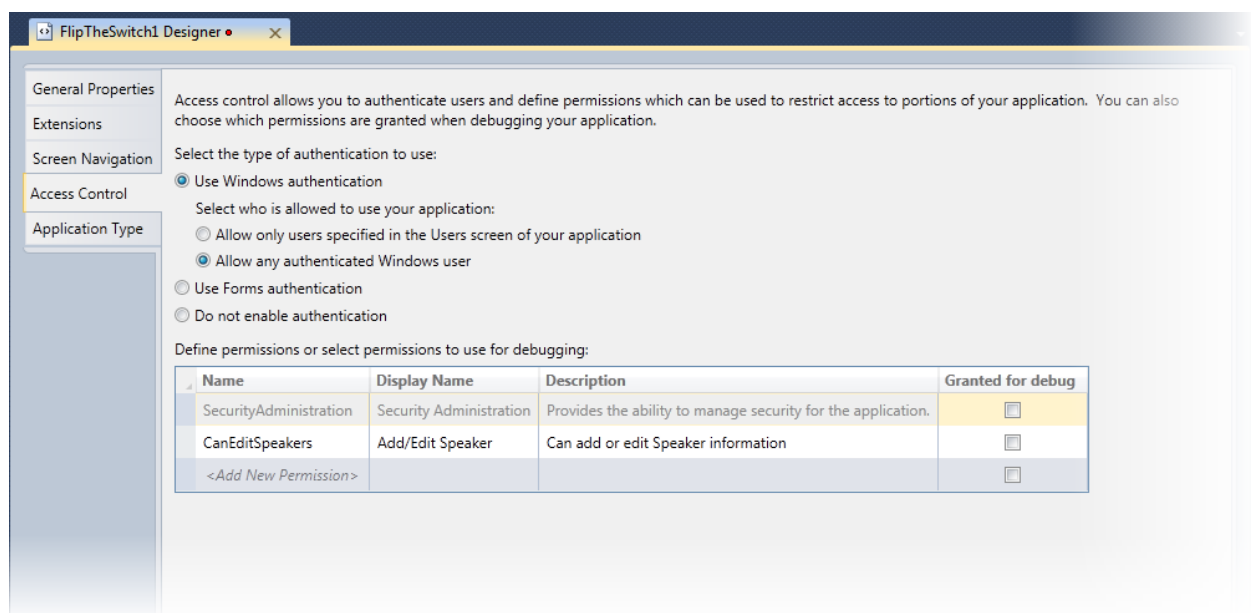
LightSwitch incorporates security right into the framework, so if you want to limit parts of your application to only certain users, you can do that. LightSwitch respects one of four different options for authentication:

- Windows Authentication, where any valid Windows User can get in.
- Windows Authentication, where you can specify which Users get in.
- Forms Authentication, where you use .Net's Security Provider database to control who has access to the application.
- No authentication at all.

There is no option for “mixed mode” authentication in LightSwitch. If you want authentication and your application is going to be hosted within your environment you'll probably want to use Windows Authentication, otherwise you'll use Forms authentication.

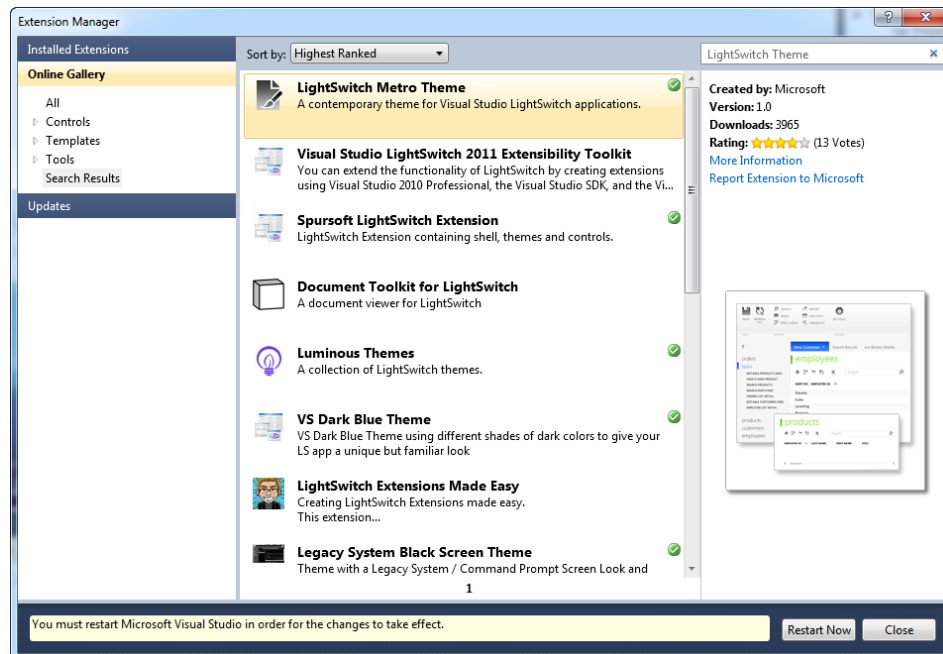
If you choose Forms Authentication or Windows Authentication where you choose who gets access to what features, LightSwitch includes the forms to specify the Roles and the Users in the framework. There isn't currently a facility included in LightSwitch for adding users using a federated account, or a “Forgot Password” function on the login screen. You'll probably want to use 3rd party tools for that.

There are many resources^{iv} which discuss Authentication in depth. It's a feature that Visual FoxPro doesn't include in the box, and it is a welcome addition for developers who've had to develop that ourselves in the past.

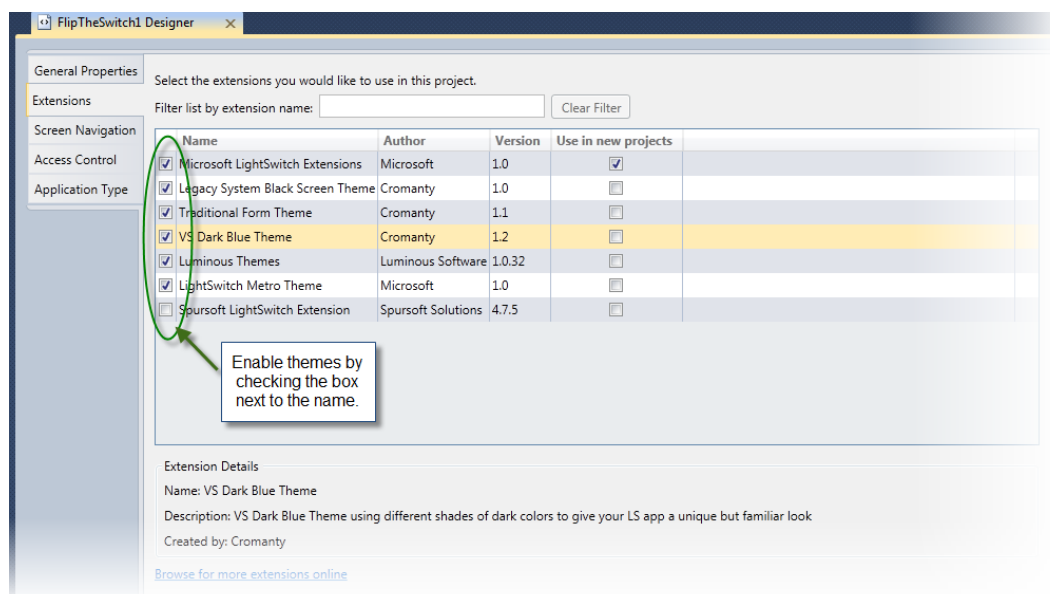


Themes

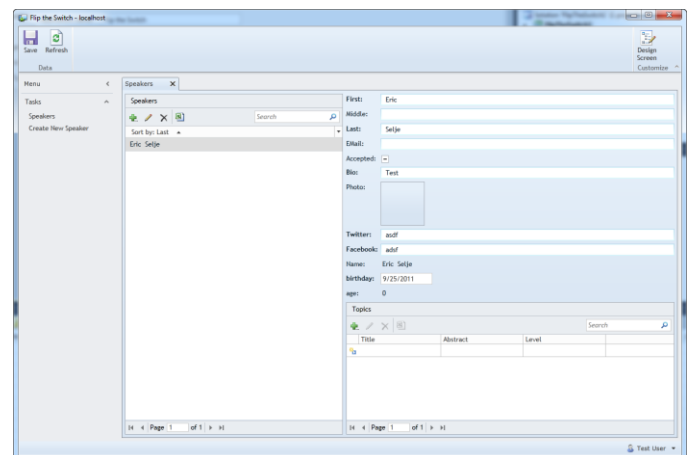
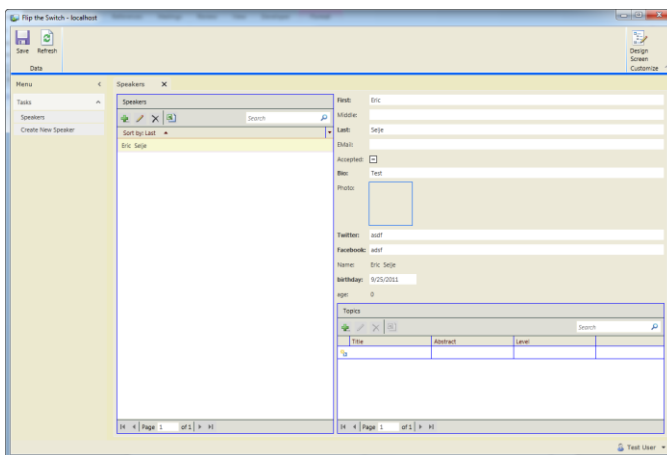
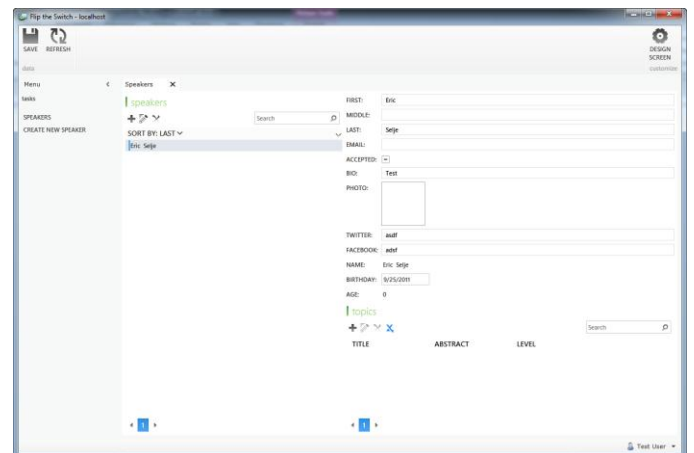
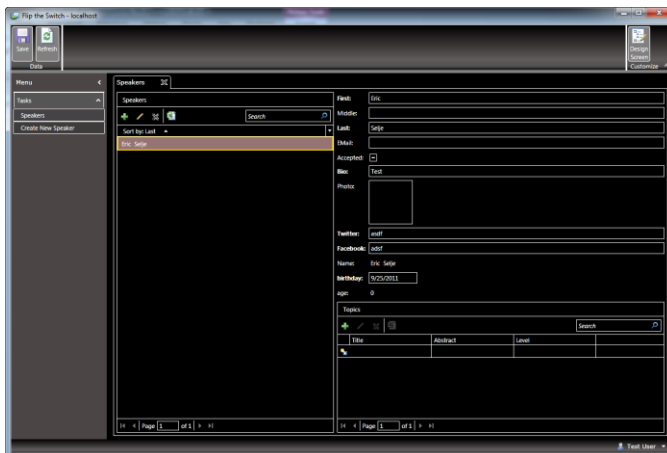
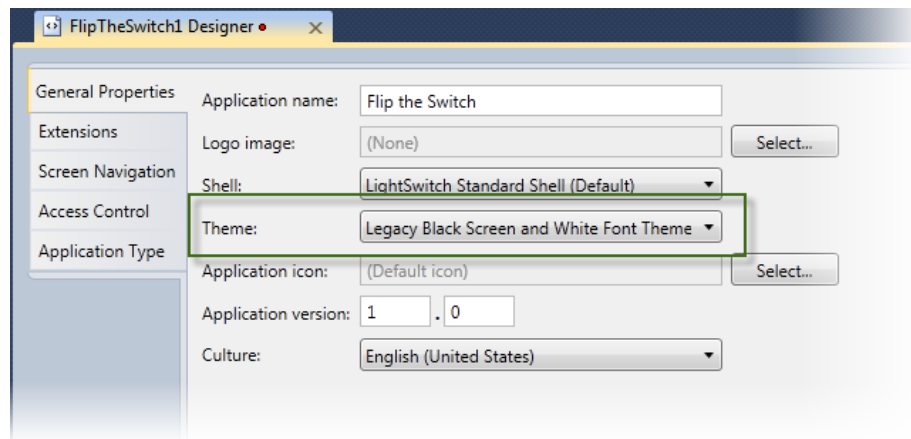
If you aren't particularly fond of the default look of the applications LightSwitch generates, do not fret. LightSwitch applications are "themable," and switching themes is easy. Simply search Visual Studio's Extension Manager Online Gallery for "LightSwitch Theme", download the themes, and restart Visual Studio (ugh).



To use the theme, you must first enable it in the Properties of your application:



And to switch to one of the enabled themes, select it in the General Properties. Here are screenshots of this simple application using four different themes.



Deployment

Once the application is ready to go out the door, LightSwitch gives you multiple options for deployment:

Client Configuration

1. Desktop Client
The application runs on the users desktop and can access Office and other program's on the user's computer.
2. Web Client
The application runs inside the user's browser and has no access to other programs on their computer.

Server Configuration

For delivering the application to the client, you also have a few options:

1. Local
The application is delivered to the user's machine and run from a shortcut.
2. IIS Server
The application is hosted on a web server accessible to the client, and is accessed via a URL.
3. Windows Azure
The application is hosted on rented space in Windows Azure. The database may also be in Windows Azure, giving you a complete "cloud-based" solution!

The deployment wizard can automatically generate scripts for creating the database on the target server as well. If you've already deployed the application it can generate just the scripts necessary to update the schema of your database.

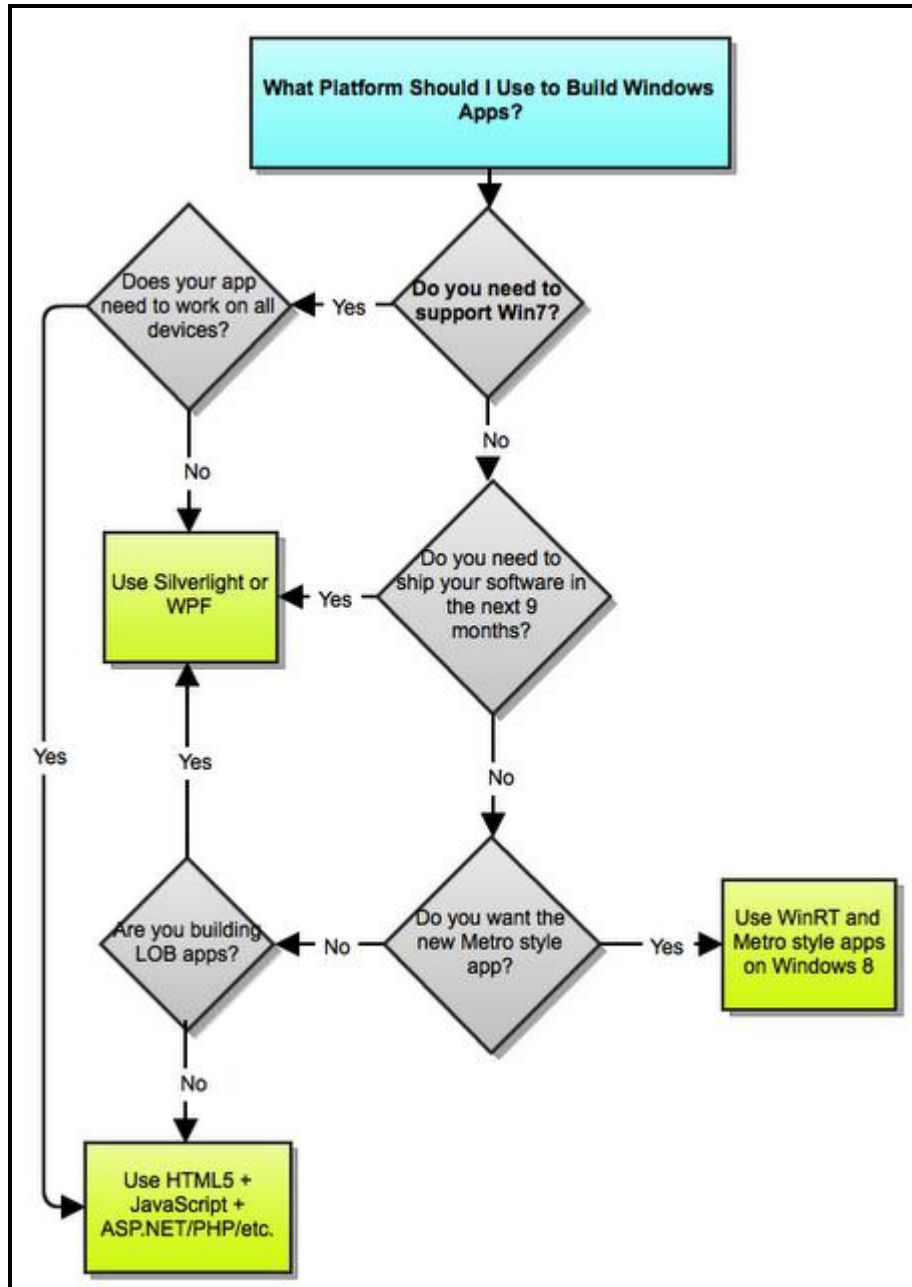
AutoUpdating Applications

If you choose options 2 or 3 above for delivering your applications, you can automatically update your clients simply by incrementing the version number in your application's properties and republishing your applications. The clients will download the newer version automatically. Nice!

Summary

Hopefully this whitepaper has given you as a Visual FoxPro developer a good sense of what LightSwitch can do and how it compares and contrasts to our favorite development tool.

Microsoft has created a lot of uncertainty about its commitment to Silverlight and .Net, and with the impending release of Windows 8 that has only increased. Here's a handy chart by Telerik that may clarify when you'd choose LightSwitch for your next project.



The Bottom Line

I hope my session and this whitepaper have elucidated what Visual Studio LightSwitch can do and where it might fit into your development toolbox. There really is a lot more that I just couldn't possibly cover so if you're interested please check out the resources listed below.

For Visual FoxPro developers, I won't say LightSwitch is the heir apparent to Visual FoxPro, but it definitely is the favored nephew of our beloved development tool. It'll be interesting to see where Microsoft takes this tool. Right now it only generates Silverlight code and Microsoft has not made it clear that it support Silverlight in the long term.

If LightSwitch could learn to generate HTML5 websites that bind to the tables as nicely as we are used to (possibly using a JavaScript library such as Amplify.js or Lawnchair) and event driven (with jQuery) and using something like the Knockout.js library for MVVM goodness, it would be an unbeatable tool for developing web apps that can run on any platform or device.

Where do I go from here?

Visual Studio LightSwitch does not lack support or documentation. I'm actually astounded by the resources Microsoft has created for supporting LightSwitch: videos, training kits, blog posts, help files, sample applications. Many independent bloggers have also posted a ton of information about LightSwitch in a very short time. Here's a short list of some of the sources I used while putting this session together:

Official LightSwitch Documentation: <http://msdn.microsoft.com/en-us/library/ff851953.aspx>

Sample Applications: <http://www.silverlight.net/showcase/>

Northwind Samples:

<http://www.microsoft.com/downloads/en/details.aspx?familyid=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en>

Starter Kits: <http://blogs.msdn.com/b/bethmassi/archive/2011/08/01/getting-started-with-the-lightswitch-starter-kits.aspx>

Training Kit: <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23746>

LightSwitch Developer Center on MSDN: <http://msdn.microsoft.com/en-us/lightswitch/ff938857.aspx>

How Do I video series: <http://msdn.microsoft.com/en-us/lightswitch/gg604823>

Tutorials: <http://www.dotnetfunda.com/misc/page36.aspx>

Copyright, 2011, Eric Selje.

Appendix A

Data types supported by the entities in LightSwitch

(Via LightSwitch Team):

LightSwitch Type	VB Type	C# Type	Range	Remarks
Binary	Byte()	byte[]	Each byte is 0 to 255	Variable length array of bytes; MaxLength specifies the maximum number of bytes
Boolean	Boolean	bool	True or False	
Byte	Byte	byte	0 to 255	
Date	Date	DateTime	Jan 1, 0001 AD (CE) to Dec 31, 9999 AD (CE)	A DateTime treated as date only; LightSwitch truncates any time portion
DateTime	Date	DateTime	00:00:00 Jan 1, 0001 AD (CE) to 23:59:59 Dec 31, 9999 AD (CE)	
Decimal	Decimal	decimal	$\pm 1.0\text{e}-28$ to $\pm 7.9\text{e}28$	Fixed decimal point value with 28-29 significant digits; suitable for financial and monetary values; stored with specific precision and scale
Double	Double	double	$\pm 5.0\text{e}-324$ to $\pm 1.7\text{e}308$	Floating decimal point with 15-16 digits precision; suitable for scientific numbers
Email Address	String	string		A String treated as an email address
Guid	Guid	Guid	{00000000-0000-0000-0000-000000000000} to {FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF}	A 128-bit integer used as a unique ID
Image	Byte()	byte[]		A Binary treated as an image
Int16	Short	short	-32,768 to 32,767	A signed 16-bit integer
Int32	Integer	int	-2,147,483,648 to 2,147,483,647	A signed 32-bit integer
Int64	Long	long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	A signed 64-bit integer
Money	Decimal	decimal		A Decimal treated as a monetary value

PhoneNumber	<u>String</u>	<u>string</u>		A <u>String</u> treated as a phone number
SByte	<u>SByte</u>	<u>sbyte</u>	-128 to 127	A signed 8-bit integer
Single	<u>Single</u>	<u>float</u>	$\pm 1.5\text{e-}45$ to $\pm 3.4\text{e}38$	Floating decimal point with 7 digits precision
String	<u>String</u>	<u>string</u>	A sequence of zero or more Unicode characters	A variable length character string; MaxLength specifies the maximum number of characters
TimeSpan	<u>TimeSpan</u>	<u>TimeSpan</u>	$\pm 10675199.02:48:05.4775807$	A time interval in days, hours, minutes, seconds, and fractions of a second When stored in SQL the range is 00:00:00.0000000 to 23:59:59.9999999

Appendix B

SQL Server data types and their equivalents in LightSwitch

(again, via Lightswitch Team):

Imported Column Type	LightSwitch Type	Supported Attributes
binary(n)	Binary	Required, MaxLength=n
image	Binary	Required, MaxLength=Max
timestamp	Binary	Required, MaxLength=8
varbinary(n)	Binary	Required, MaxLength=n
bit	Boolean	Required
tinyint	Byte	Required
date	Date	Required
datetime	DateTime	Required
datetime2(n)	DateTime	Required
smalldatetime	DateTime	Required
decimal(p,s)	Decimal	Required, Precision=p, Scale=s
money	Decimal	Required, Precision=19, Scale=4
numeric(p,s)	Decimal	Required, Precision=p, Scale=s
smallmoney	Decimal	Required, Precision=10, Scale=4
float	Double	Required
uniqueidentifier	Guid	Required
smallint	Int16	Required
int	Int32	Required
bigint	Int64	Required

real	Single	Required
char(n)	String	Required, MaxLength=n
nchar(n)	String	Required, MaxLength=n
ntext	String	Required
nvarchar(n)	String	Required, MaxLength=n
text	String	Required
varchar(n)	String	Required, MaxLength=n
xml	String	Required, MaxLength=Max
time(n)	TimeSpan	Required
datetimeoffset	<i>not supported</i>	
geography	<i>not supported</i>	
geometry	<i>not supported</i>	
hierarchyid	<i>not supported</i>	
sql_variant	<i>not supported</i>	

ⁱ <http://blogs.msdn.com/b/jasonz/archive/2011/07/26/visual-studio-lightswitch-2011-is-available-today.aspx>

ⁱⁱ <http://www.rssbus.com>

ⁱⁱⁱ <http://linqtovfp.codeplex.com/>

^{iv} <http://blogs.msdn.com/b/bethmassi/archive/2010/10/06/implementing-security-in-a-lightswitch-application.aspx>