Design Time Data

In this lab, we will code in bursts. Get the starting source code and follow along. I will commit changes after every burst, so you can get caught up if you get lost.

## Gitting the Source Code

If you have a git client installed, then clone the repository. Create a working branch. For example:

cd c:\projects

git clone git://github.com/dallasxaml/DesignTimeData.git

cd DesignTimeData

git checkout –b take1

After each successful burst, commit your changes. If you ever get lost, commit that branch, go back to master, and create a new one:

git add –A

git commit –m "I missed that."

git checkout master

git pull

git checkout –b take2

## Downloading the Source Code

If you don’t have git installed, then go to the following URL and click on the “ZIP” button:

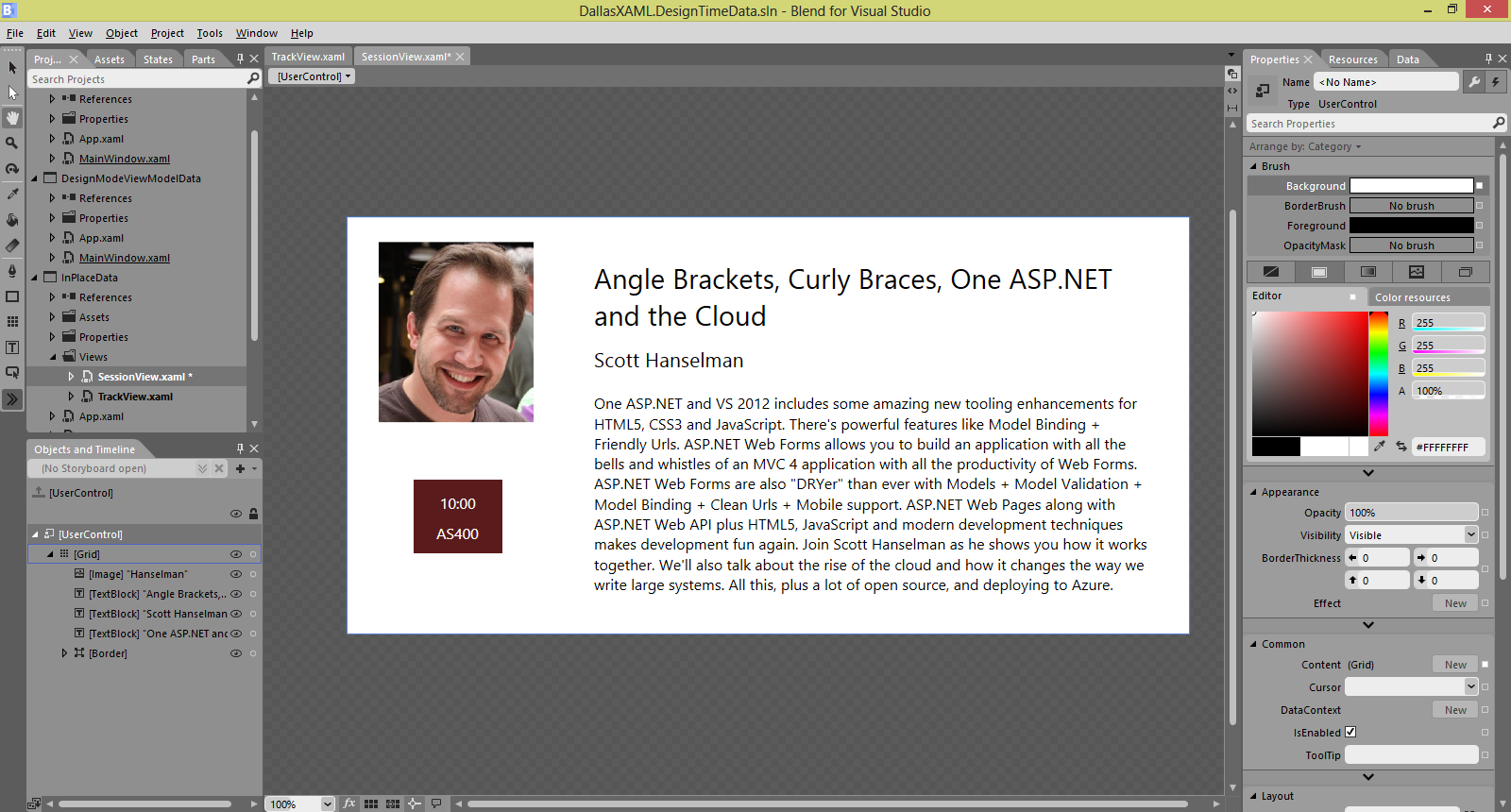
https://github.com/dallasxaml/DesignTimeData

Unzip to your project folder. If you ever get lost, go back to the web page and download the zip again.

# In Place Data

When starting an application, let yourself be free to brainstorm. Don’t try to organize right away. Just put data directly your design.

Let’s build a conference app. Open the InPlaceData application and start building this screen:



Important things to notice:

* No Lorem Ipsum! This is real content.
* Eyeball everything. No grid layouts.
* Copy the image from the web and just paste into Blend.

Open the TrackView control. Populate the list with strings:

xmlns:s="clr-namespace:System;assembly=mscorlib"

<ListBox>

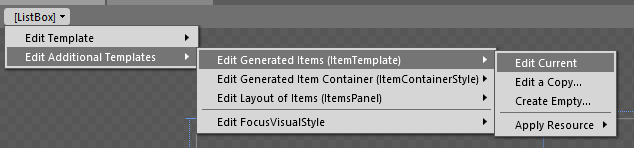
<s:String>Angle Brackets in the Cloud</s:String>

<s:String>Angular Applications</s:String>

<s:String>Ouch! These darn Angle Brackets!</s:String>

</ListBox>

Then edit the ItemTemplate:

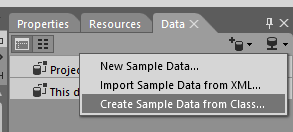


Data bind to the data context to bring in the string.

# Sample Data

Since we are working in the WPF stack, we can use sample data. Windows Store apps can’t do this, yet.

In the SampleData project, there’s a TrackViewModel, which contains a list of SessionHeaderViewModels. Open the TrackView and create sample data based on the TrackViewModel class.



Drag the Name property of the TrackViewModel to the title to data bind it. Drag Sessions to the list box. Hey, where’d my template go!

Undo that, and data bind ItemsSource through the properties window instead. Edit the template, and notice how the gold rectangle changes. Drag to data bind the properties.

Oops! We forgot to add Sponsor to the track view model! Add the Sponsor property, and try to data bind it. It’s not there!

Check out the SampleData folder. This is populated with XML files for the sample data. You *could* add the new property to the XML. But don’t do that. It’s not that big of a deal. Just delete the sample data and re-generate it.

# Design Mode View Model Data

We are starting to put together the view model layer to support the view. Let’s populate it with realistic data instead of Lorem Ipsum.

Open the DesignModeViewModelData project. Create a class in the DataSources folder called DesignModeDataSource. Give it a single property called Track that returns a TrackViewModel, populated with design-mode data.

public class DesignModeDataSource

{

    public TrackViewModel Track

    {

        get

        {

            return new TrackViewModel

            {

                Name = "Architecture",

                Sessions = new List<SessionHeaderViewModel>

                {

                    new SessionHeaderViewModel

                    {

                        Title = "Real-Time Web Programming with SignalR",

                        Speaker = "Brian Sullivan",

                        Room = "The Papasan",

                        Time = new DateTime(2013, 3, 16, 16, 0, 0)

                    }

                }

            };

        }

    }

}

Add the data source to the application resources:

xmlns:ds="clr-namespace:DesignModeViewModelData.DataSources"

<Application.Resources>

    <ds:DesignModeDataSource x:Key="DesignModeDataSource" />

</Application.Resources>

Bring the design-mode data into the TrackView:

d:DataContext="{Binding Track, Source={StaticResource DesignModeDataSource}}"

Now bind the view to the correct properties and see how it looks with realistic data.