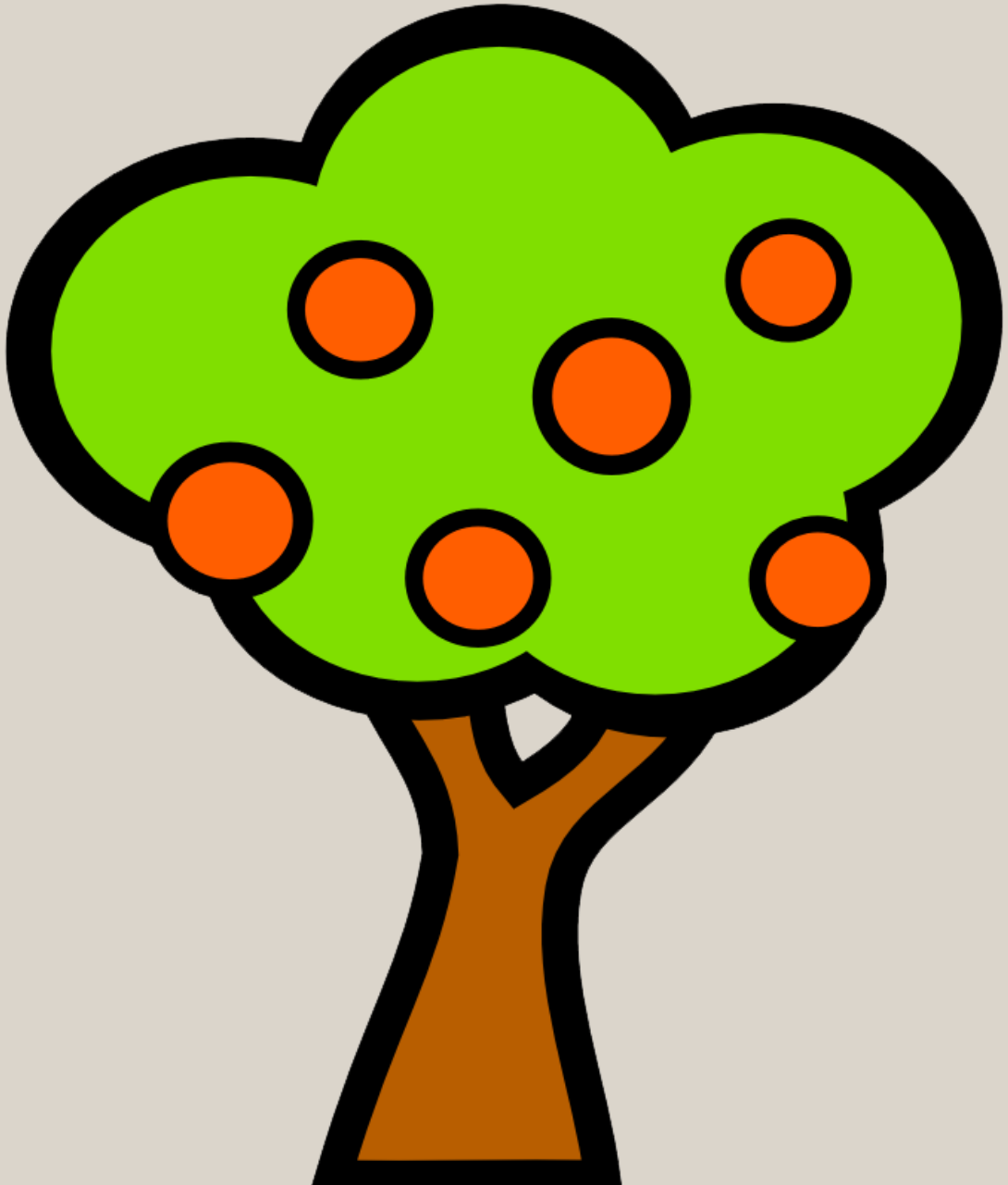
Building Blocks – Part 2

Before we begin

And now a word from our sponsor...

Our Vision



UI Toolkit

Spend less time writing code and more time solving problems

Top 25 (out of 106)

- NotificationBox 154
- DataTextBox 137
- Grid 122
- RockDropDownList 118
- RockTextBox 118
- ModalAlert 76
- RockCheckBox 74
- DeleteField 60
- TermDescription 44
- HighlightLabel 41
- ModalDialog 39
- GridFilter 37
- BoolField 34
- PersonPicker 26
- RockRadioButtonList 25
- DateRangePicker 22
- ReorderField 20
- DateTimeField 20
- PanelWidget 19
- DateField 18
- Toggle 18
- NumberBox 17
- CategoryPicker 17
- RockControlWrapper 16
- BootstrapButton 14

Rock Control Gallery

- This example block has a variety of example Rock controls
- Add this block to a page to explore

Rock:Notificationbox

Rock:Badge

Rock:HighlightLabel

Pickers

EXAMPLE

Rock:DatePicker

```
<Rock:DatePicker ID="dpExample" runat="server" Label="Rock:DatePicker" />
```

EXAMPLE

Rock:DateTimePicker

```
<Rock:DateTimePicker ID="dtpExample" runat="server" Label="Rock:DateTimePicker" />
```

```
<Rock:HighlightLabel ID="hlblExample" runat="server" LabelType="Danger" IconCssClass="fa fa-flag" Text="errors" />
```

Rock.Web.UI.Controls

- Several are documented (old) a bit on our wiki
- github.com/SparkDevNetwork/Rock/wiki/UI-Toolkit
- They will be included in an upcoming Rock Developer guide over at rockrms.com/Rock/Developer/Code

Methods

A few handy methods you're going to use

Getting passed values

- **PageParameter(string name)** - Checks the page route parameters and query string parameters for a parameter with the given name and returns the value if found.

- **LinkedPageUrl(string attributeKey, Dictionary params)** - Returns the a url to use to navigate to a page specified in a LinkedPage attribute of the block.

NavigateTo...

- **NavigateToParentPage()** - will redirect the user to the "parent" page of the current block.

- **NavigateToParentPage(Dictionary params)** – Like previous but with query string parameters.

- **NavigateToLinkedPage(string attributeKey, Dictionary params)** - Redirects user to a page specified in a LinkedPage attribute of the block.

And Path Methods

It’s all relative

ResolveRockUrl(string)

- When you're taking charge of building URLs
- Use ~ to resolve web application root (i.e., / or /Rock or /Foo
- Use ~~ to resolve theme root (i.e., /Foo/Themes/Stark, etc.)
  - because you can't assume which theme is being used.

Examples

```
content = String.Format( "<div class='alert alert-
warning'><h4>Warning</h4>Could not find the template _{1}.liquid in
{0}.</div>", ResolveRockUrl( "~~/Assets/Liquid" ), match.Groups[1].Value );
```

```
protected string FormatPersonLink(string personId)
{
    return ResolveRockUrl( string.Format( "~/Person/{0}", personId ) );
}
```

Breadcrumbs

I’m on my way...home sweet home.

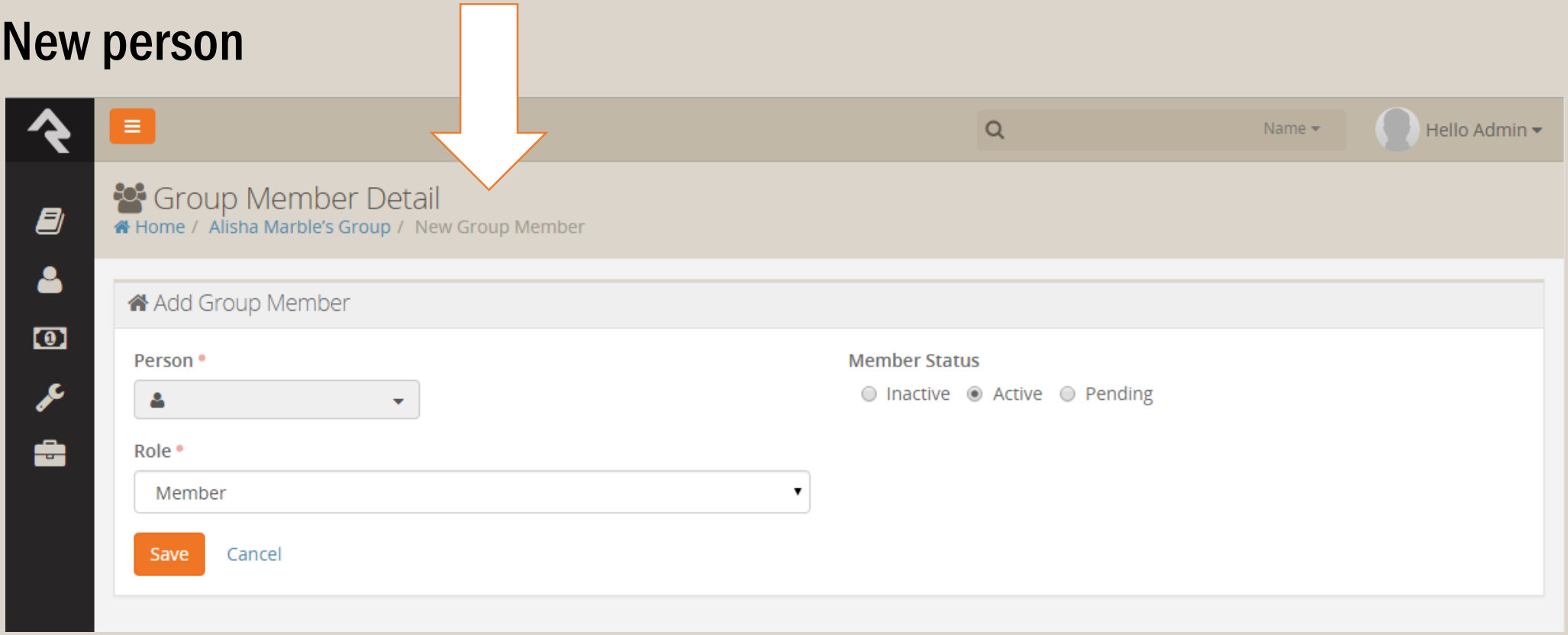
Override GetBreadCrumbs(…)

- You're taking over building the breadcrumb(s) for that page.

- Nutshell
  - Use the given page reference to get the id of the item in question,
  - Use that id to get that item's title, and then add a new BreadCrumb() onto a list of breadcrumbs that is returned to the caller.

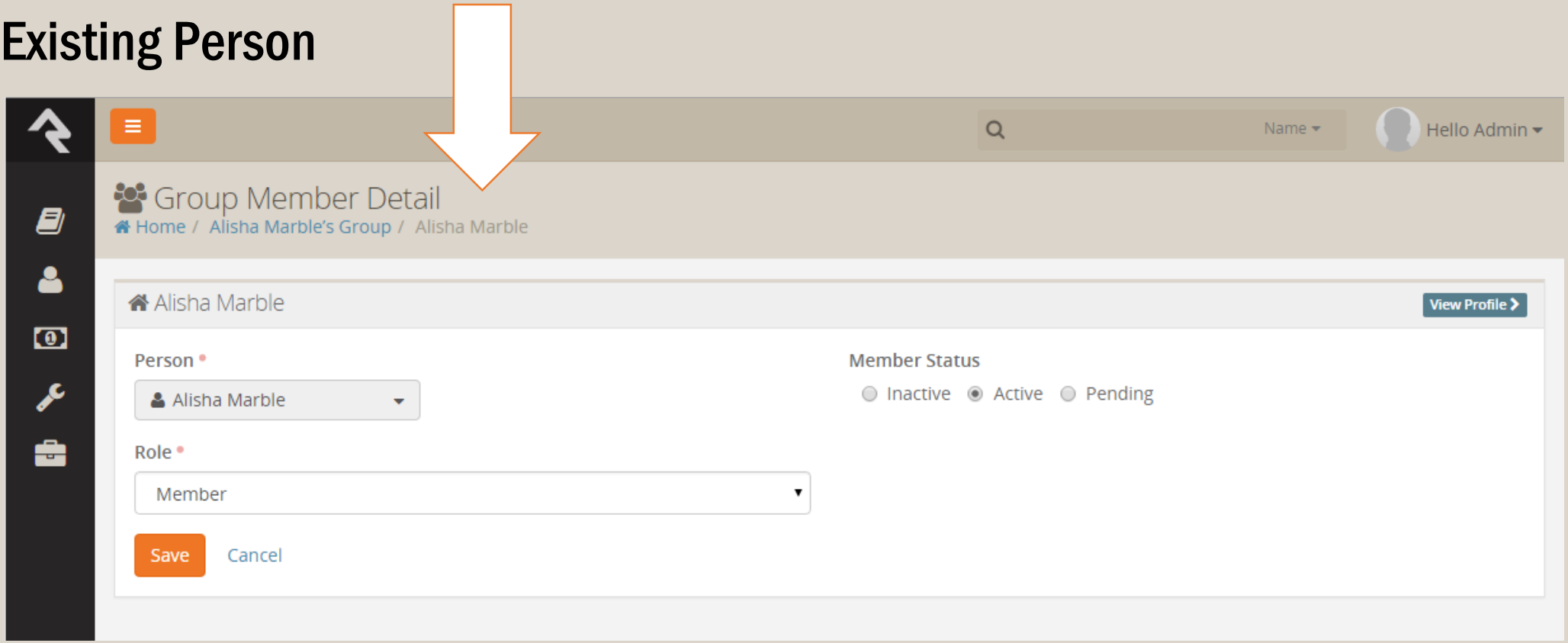
Consider the Group Member Details

- New person



Consider the Group Member Details

- Existing Person

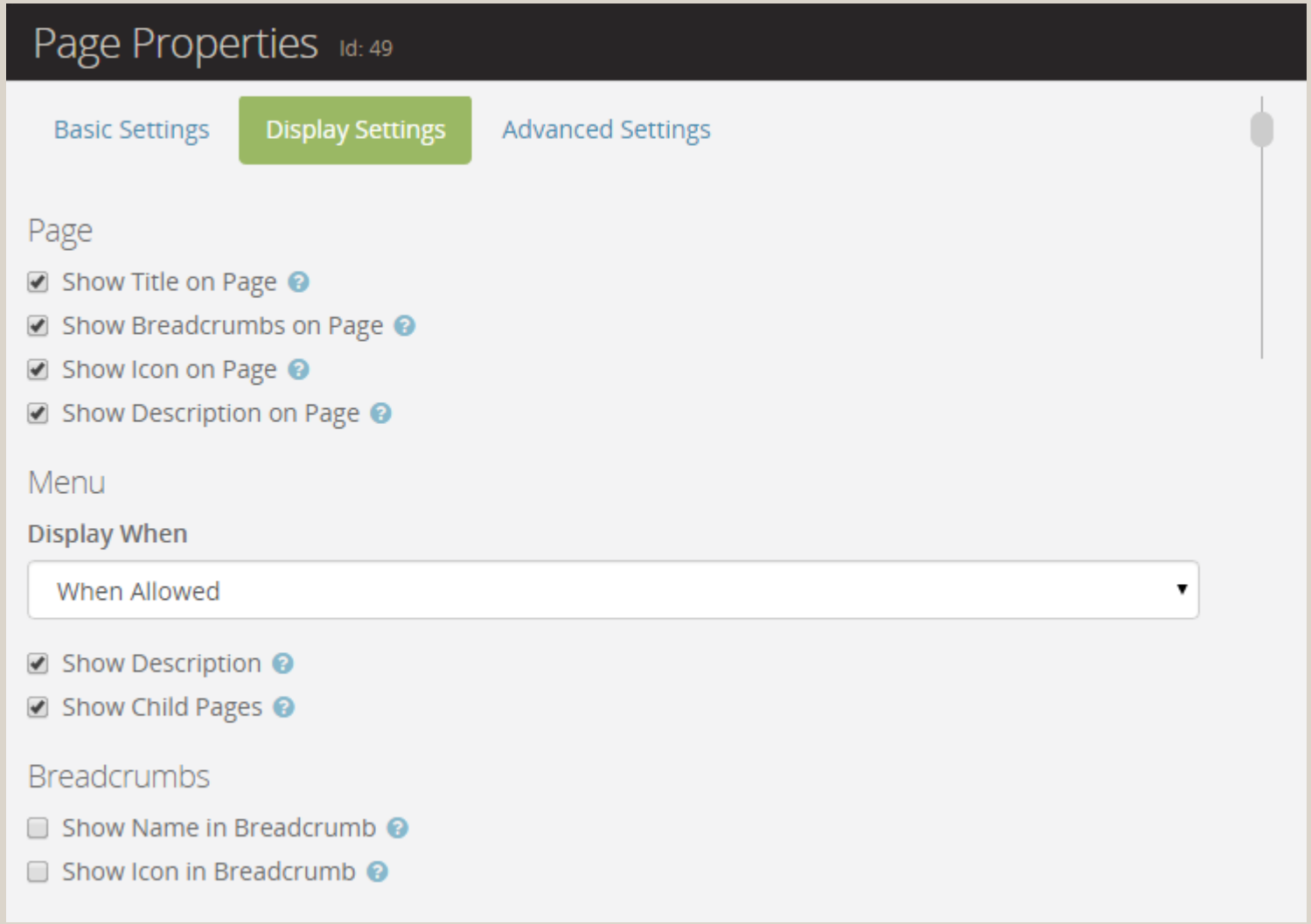


Code

```
public override List<BreadCrumb> GetBreadCrumbs( PageReference pageReference )
{
    var breadCrumbs = new List<BreadCrumb>();

    int? id = PageParameter( pageReference, "GroupMemberId" ).AsIntegerOrNull();
    if ( id != null )
    {
        GroupMember member = new GroupMemberService( new RockContext() ).Get( id.Value );

        if ( member != null )
            breadCrumbs.Add( new BreadCrumb( member.Person.FullName, pageReference ) );
        else
            breadCrumbs.Add( new BreadCrumb( "New Group Member", pageReference ) );
    }

    return breadCrumbs;
}
```

Breadcrumb Page, Child Page

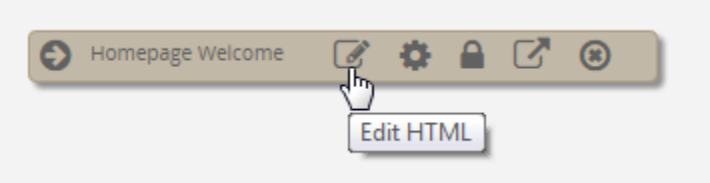
- Controlled by Page Property
  Display Settings



Block Configuration Slide-Out Tool Bar

Injecting your own controls into that thingy

This thing...



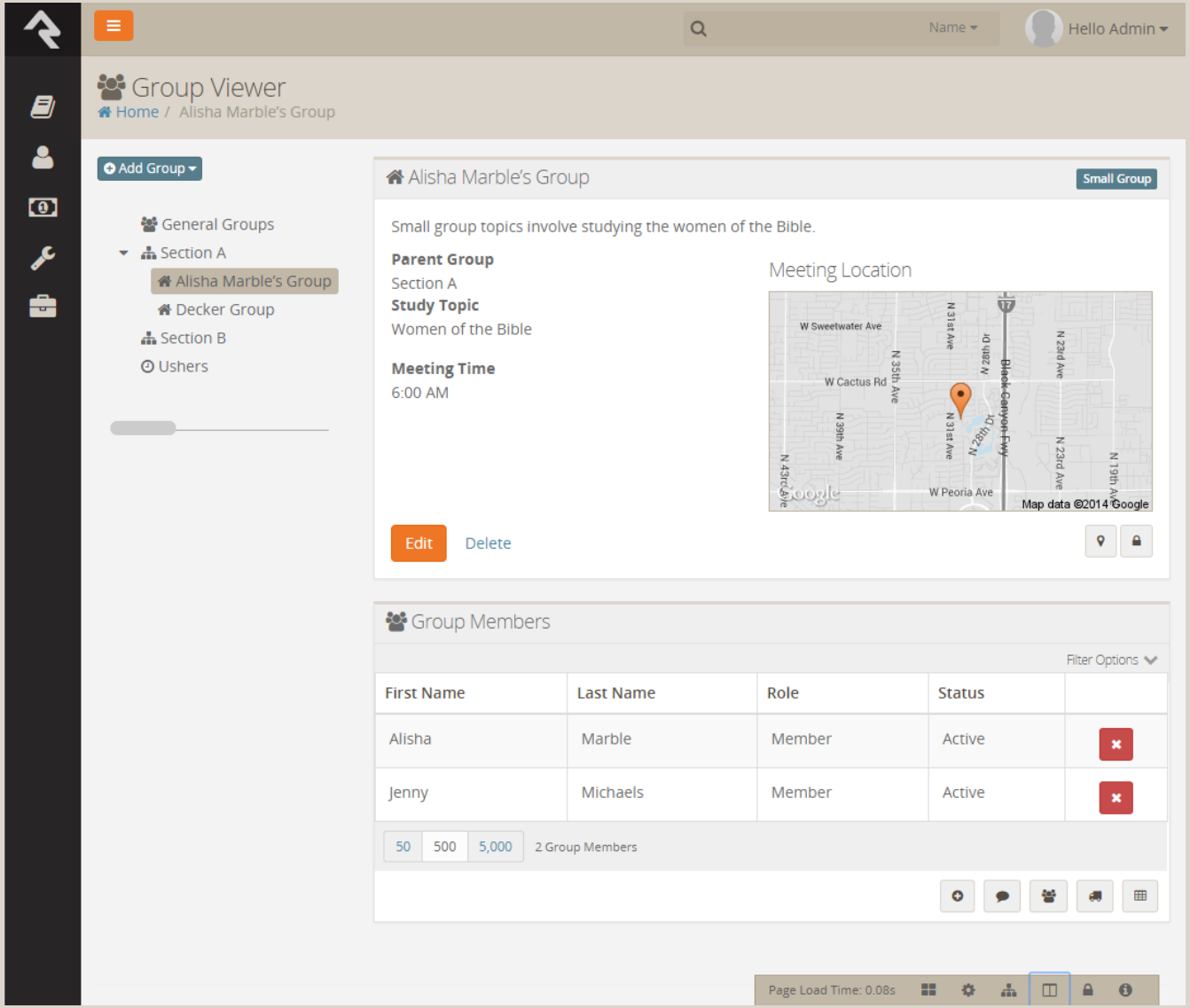
- Consider the implementation in the HtmlContentDetails block...

```
public override List<Control> GetAdministrateControls( bool canConfig, bool canEdit )
{
    List<Control> configControls = new List<Control>();

    // add edit icon to config controls if user has edit permission
    if ( canEdit )
    {
        LinkButton lbEdit = new LinkButton();
        lbEdit.CssClass = "edit";
        lbEdit.ToolTip = "Edit HTML";
        lbEdit.Click += lbEdit_Click;
        configControls.Add( lbEdit );
        HtmlGenericControl iEdit = new HtmlGenericControl( "i" );
        lbEdit.Controls.Add( iEdit );
        lbEdit.CausesValidation = false;
        iEdit.Attributes.Add( "class", "fa fa-pencil-square-o" );

        // will toggle the block config so they are no longer showing
        lbEdit.Attributes["onclick"] = "Rock.admin.pageAdmin.showBlockConfig()";

        ScriptManager.GetCurrent( this.Page ).RegisterAsyncPostBackControl( lbEdit );
    }

    configControls.AddRange( base.GetAdministrateControls( canConfig, canEdit ) );

    return configControls;
}
```

Block Cooperation: Coordinated Visibility

Making loosely coupled blocks play together

IDetailBlock & ISecondaryBlock



GroupMemberDetail

GroupMemberList

Interfaces

```
public interface IDetailBlock
{
    void ShowDetail( int itemId );
}
```

```
public interface ISecondaryBlock
{
    void SetVisible( bool visible );
}
```

DetailBlock

- Should call HideSecondaryBlocks(bool) when appropriate.
- And Rock calls SetVisible(bool) for all secondary blocks on the page.

Block Cooperation: Context

It’s a bird, it’s a plane, no it’s _____

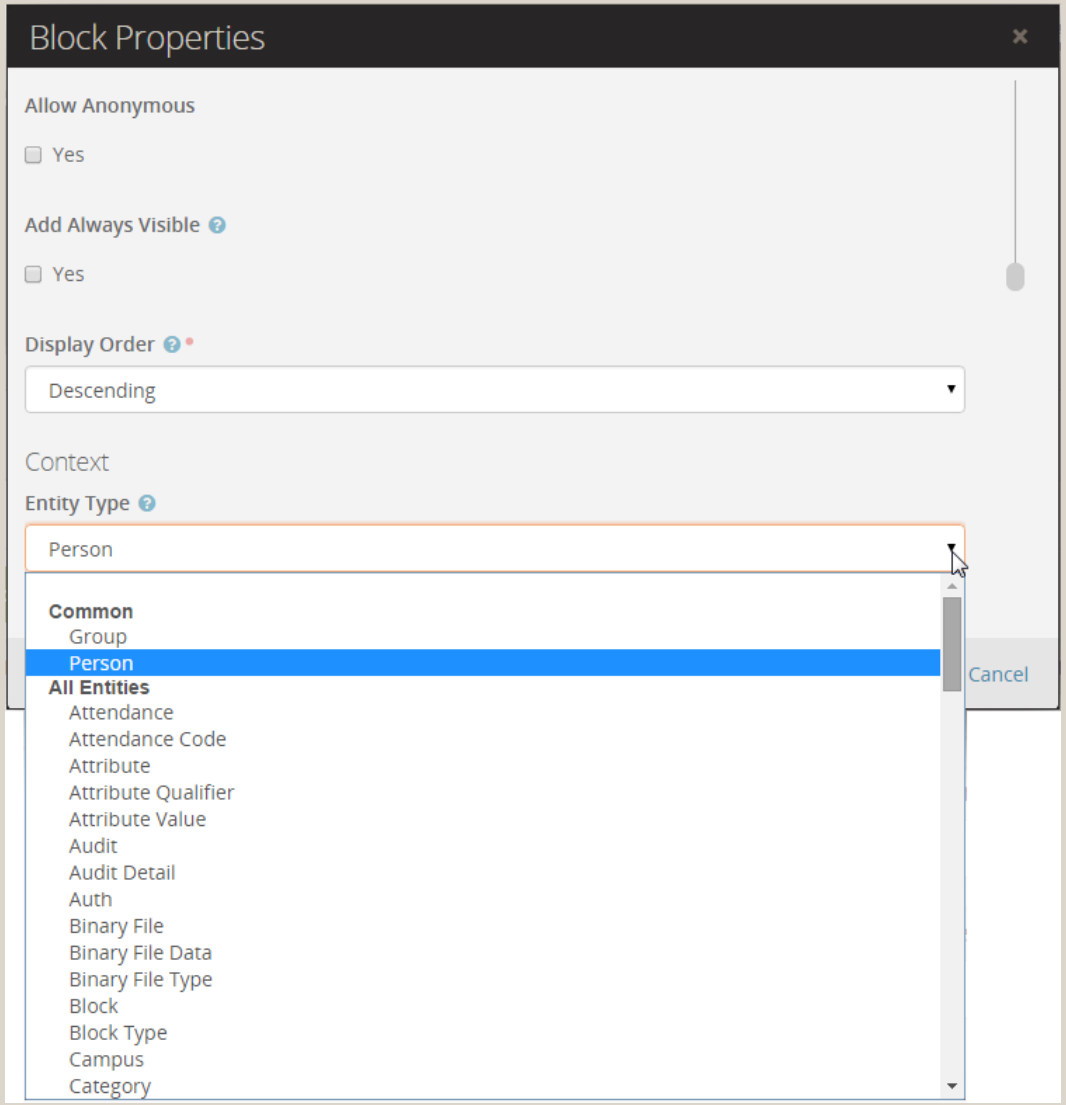
Notes

- Add example

[ContextAware] Blocks

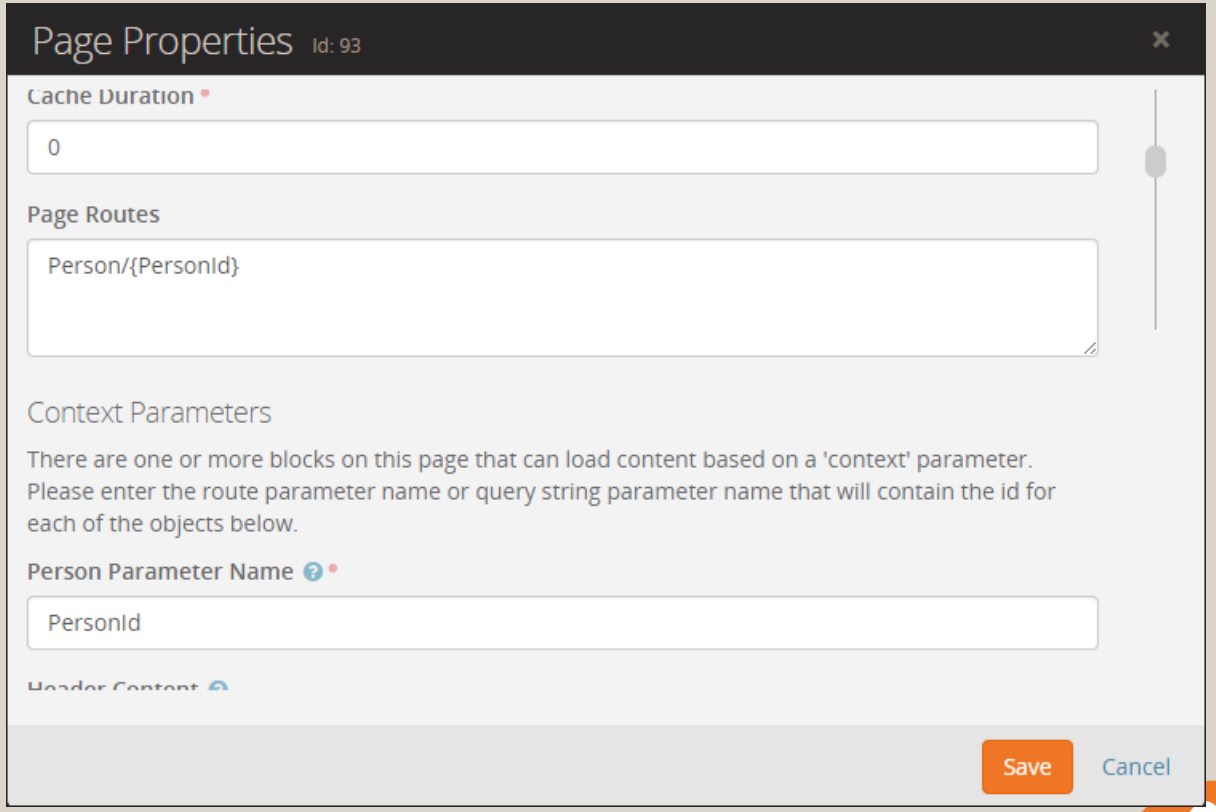
- Bound to an Entity Type
  - Stored in ContextTypesRequired
- Uses entity set in the page context

```
var contextEntity = this.ContextEntity();
if ( contextEntity != null )
{
    if ( contextEntity is Person )
    {
        _person = contextEntity as Person;
    }
    // ...
}
```



Page Coordinator

- Under Advanced Settings
  - Page puts an entity into Context via parameter



Block Cooperation: Sharing

How to share stuff

Get an Item, Save an Item

- Useful if you want to use an item that's probably already been loaded by another block.

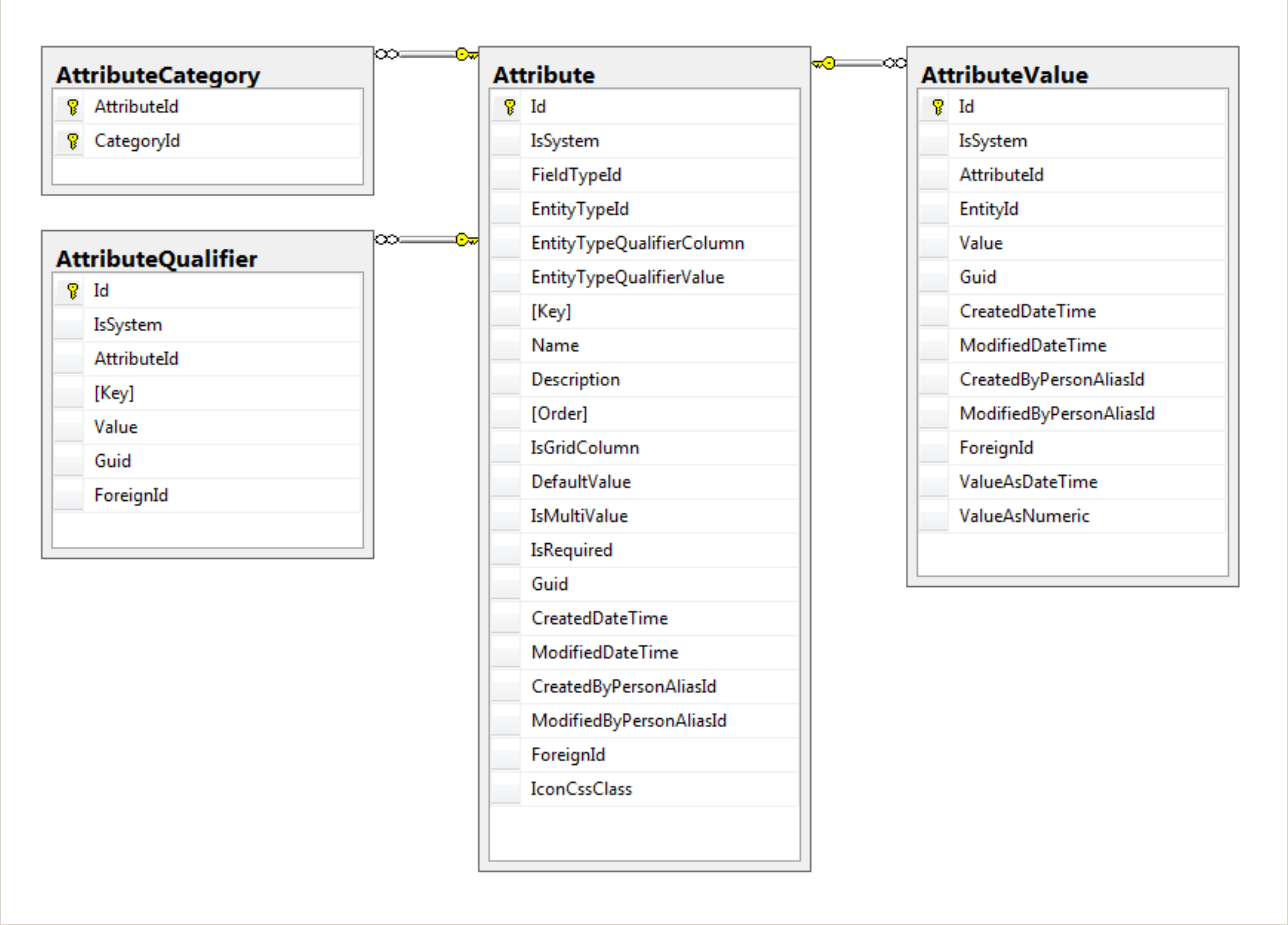
```
var key = string.Format( "Group:{0}", groupId );

var group = RockPage.GetSharedItem( key ) as Group;
if ( group == null )
{
    group = new GroupService( new RockContext() ).Queryable( "GroupType" )
                .Where( g => g.Id == groupId || g.Guid == groupGuid )
                .FirstOrDefault();

    RockPage.SaveSharedItem( key, group );
}
```

Attributes, Attributes

Where for art thou attributes?

What are they?

- They are for any entity
- And have a value
- They can be:
  - categorized
  - qualified (value is another entity; via EntityTypeQualifier*)
- The AttributeQualifier explains how the value is *handled*



Getting Entity Attribute Values

- *<entity>*.LoadAttributes() first, then
- *<entity>*.GetAttributeValue(key)

```
definedValue.LoadAttributes();
string fileUrl = definedValue.GetAttributeValue("DownloadUrl");
```

- Another example:

```
person.LoadAttributes();
string personAbilityLevelGuid = person.GetAttributeValue( "AbilityLevel" );
```

Saving Entity Attribute Values

```
groupMember.LoadAttributes( rockContext );
```

- After you've loaded the attributes... you can set values

```
groupMember.SetAttributeValue( key, value );
groupMember.SaveAttributeValues( rockContext );
```

- Another example:

```
p.LoadAttributes( rockContext );
p.SetAttributeValue( "AbilityLevel", selectedAbilityLevelGuid.ToUpper() );
p.SaveAttributeValues( rockContext );
```

Beyond Blocks

Custom Data

When groups and attributes just don’t fit

Entity Framework*ish*

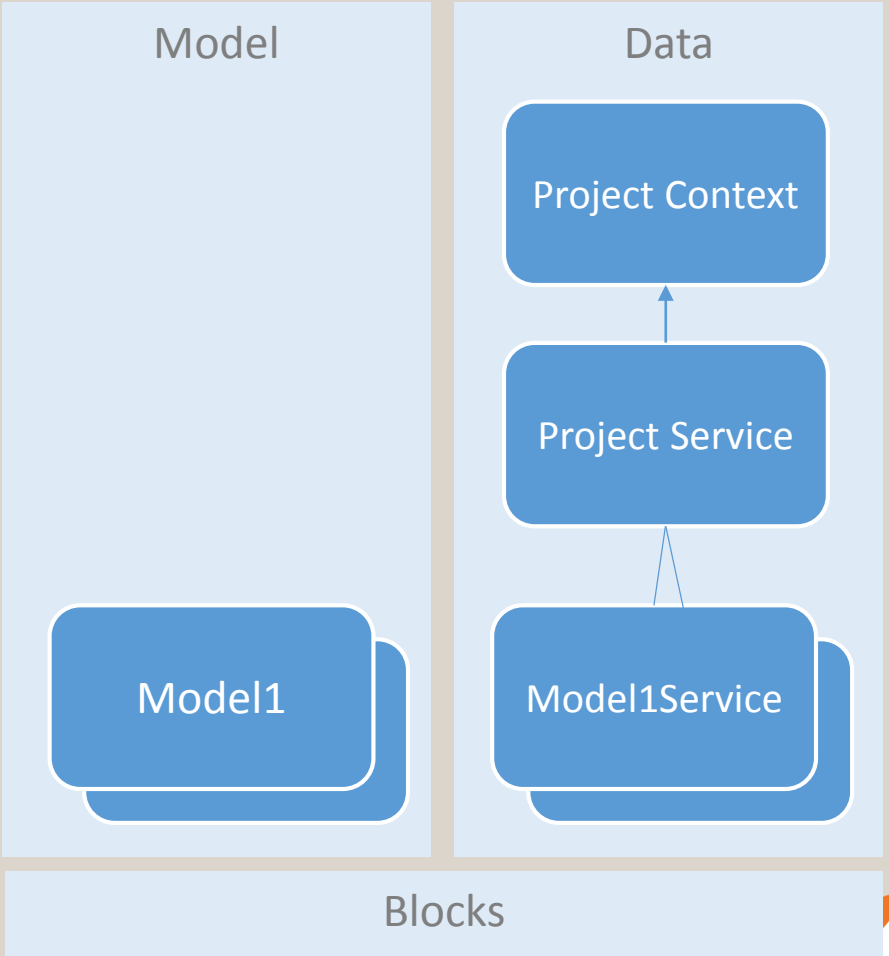
- Rock uses EF6 with a code-first approach
- Plugins use EF6 with code-first-then-database-second approach ;)
- You can’t really use Add-Migration to generate your schema SQL
  - You create your table create scripts and include them in your project
- Rock will integrate your entities and entity service classes
- You don’t worry about database connections
- Rock will run your migrations in your plugins

Add a Class Library Project

- Target 4.5.1
- From the Framework Assemblies, select:
  - System.ComponentModel.DataAnnotations
  - System.Runtime.Serialization
- From the Browse, navigate to your RockWeb/bin folder and select:
  - EntityFramework.SqlServer.dll
  - EntityFramework.dll
  - DotLiquid.dll
  - Rock.dll

Class Overview

- A model class w/ configuration class (for each model)
- A context class (for the project)
- A service base class (uses the context class)
- A service class (for each model)



Create a Model

- Replace *org.rocksolidchurch.SampleProject* with your stuff

```
namespace org.rocksolidchurch.SampleProject.Model
{
    [Table( "_org_rocksolidchurch_SampleProject_Foo" )]
    [DataContract]
    public class Foo : Rock.Data.Model<Foo>
    {
        // ...
    }
}
```

- Extend Rock.Data.Model base class

Add Your Properties

- Example

```
[MaxLength( 100 )]
[Required( ErrorMessage = "Name is required" )]
[DataMember( IsRequired = true )]
public string Name { get; set; }

[DataMember]
public int PersonId { get; set; }

[DataMember]
public int? MyTypeThingValueId { get; set; }
```

- Id, Guid, CreatedByPersonAliasId, CreatedDateTime, ModifiedByPersonAliasId, and ModifiedDateTime all come from the base class.**

** but you still have to include these columns in your table create script

Add Virtual & Navigation Properties

- Virtual

```
public virtual PersonAlias Person { get; set; }

[DataMember]
public virtual DefinedValue MyTypeThingValue { get; set; }
```

- Navigational Collections

```
[DataMember]
public virtual ICollection<Response> Response { get; set; }
```

A Configuration class

- Extends EntityTypeConfiguration

```
public partial class FooConfiguration : EntityTypeConfiguration<Foo>
{
    public FooConfiguration()
    {
        this.HasRequired( r => r.Person ).WithMany().HasForeignKey(r =>
            r.PersonId).WillCascadeOnDelete(false);

        this.HasOptional( r => r.MyTypeThingValue ).WithMany().HasForeignKey(
            p => p.MyTypeThingValueId ).WillCascadeOnDelete( false );

        this.HasMany( r => r.Responses ).WithRequired(
            r => r.Foo ).HasForeignKey( r => r.FooId );
    }
}
```

The Rest is Easy Breezy

- The remaining classes are basically boilerplate code.

Context Class

```
namespace org.rocksolidchurch.SampleProject.Data
{
    public partial class SampleProjectContext : Rock.Data.DbContext
    {
        #region Models
        public DbSet<Foo> Foos { get; set; }
        // ... Add all your other models here
        #endregion

        public SampleProjectContext() : base( "RockContext" ) {}

        protected override void OnModelCreating( DbModelBuilder modelBuilder )
        {
           Database.SetInitializer<SampleProjectContext>( new NullDatabaseInitializer<SampleProjectContext>() );
           Rock.Data.ContextHelper.AddConfigurations( modelBuilder );
           modelBuilder.Configurations.AddFromAssembly( System.Reflection.Assembly.GetExecutingAssembly() );
        }
    }
}
```

Service Class

```
namespace org.rocksolidchurch.SampleProject.Data
{
    public class SampleProjectService<T> : Rock.Data.Service<T> where T :
        Rock.Data.Entity<T>, new()
    {
        public SampleProjectService( SampleProjectContext context )
            : base( context )
        {
        }

        public virtual bool CanDelete( T item, out string errorMessage )
        {
            errorMessage = string.Empty;
            return true;
        }
    }
}
```

Lastly Your Model's *Service* Class

- Create one for each model
- Add your custom get-fetch methods (i.e, GetFoosByBlah()) in your service classes.

```
namespace org.rocksolidchurch.SampleProject.Model
{
    public class FooService : SampleProjectService<Foo>
    {
        public FooService( SampleProjectContext context ) : base( context ) { }
    }

    public IQueryable<Foo> GetByBLAH ( int blah )
    {
            //...
    }
}
```

Plugin Migrations

- Put in your project Migrations folder and filename order them:

```
Migrations/
   001_CreateDb.cs
   002_AddSystemData.cs
```

- Extend the Rock.Plugin.Migration class

Table Create Script – Up()

```
using Rock.Plugin;

namespace org.rocksolidchurch.SampleProject.Migrations
{
    [MigrationNumber( 1, "1.1.0" )]
    public class CreateDb : Migration
    {
        public override void Up()
        {
            Sql( @"
    CREATE TABLE [dbo].[_org_rocksolidchurch_SampleProject_Foo](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](100) NOT NULL,
    [PersonAliasId] [int] NULL,
    [MyThingValueId] [int] NULL,
    [Guid] [uniqueidentifier] NOT NULL,
    [CreatedDateTime] [datetime] NULL,
    [ModifiedDateTime] [datetime] NULL,
    [CreatedByPersonAliasId] [int] NULL,
    [ModifiedByPersonAliasId] [int] NULL,
    [ForeignId] [nvarchar](50) NULL,
```

Ordinal, Rock version dependency

Down()

```
public override void Down()
{
    Sql(@"
    ALTER TABLE [dbo].[_org_rocksolidchurch_SampleProject_ReferralAgency] DROP CONSTRAINT
[FK_dbo._org_rocksolidchurch_SampleProject_ReferralAgency_dbo.PersonAlias_ModifiedByPersonAl
iasId]
    ALTER TABLE [dbo].[_org_rocksolidchurch_SampleProject_ReferralAgency] DROP CONSTRAINT
[FK_dbo._org_rocksolidchurch_SampleProject_ReferralAgency_dbo.PersonAlias_CreatedByPersonAli
asId]
    ALTER TABLE [dbo].[_org_rocksolidchurch_SampleProject_ReferralAgency] DROP CONSTRAINT
[FK_dbo._org_rocksolidchurch_SampleProject_ReferralAgency_dbo.Campus_CampusId]
    ALTER TABLE [dbo].[_org_rocksolidchurch_SampleProject_ReferralAgency] DROP CONSTRAINT
[FK_dbo._org_rocksolidchurch_SampleProject_ReferralAgency_dbo.DefinedValue_ReferralAgencyTyp
eValueId]
    DROP TABLE [dbo].[_org_rocksolidchurch_SampleProject_ReferralAgency]
" );
}
```

Add Data (Pages, Blocks, etc.)

```
namespace org.rsc.SampleProject.Migrations
{
    [MigrationNumber( 2, "1.1.0" )]
    public class AddSystemData : Rock.Plugin.Migration
    {
        public override void Up()
        {
            RockMigrationHelper.AddPage( "7F2581A1-941E-4D51-8A9D-5BE9B881B003", "D65F783D-
87A9-4CC9-8110-E83466A0EADB", "Referral Agencies", "", "223AC4F2-CBED-4733-807A-
188CFBBFA0C8", "fa fa-check-square-o" ); // Site:Rock RMS
            RockMigrationHelper.AddPage( "223AC4F2-CBED-4733-807A-188CFBBFA0C8", "D65F783D-
87A9-4CC9-8110-E83466A0EADB", "Referral Agency Details", "", "4BF8FA57-AE86-4103-B07E-
80ECE0000AEE", "fa fa-check-square-o" ); // Site:Rock RMS
            Sql( @"
    UPDATE [Page] SET [BreadCrumbDisplayName] = 0 WHERE [Guid] = '4BF8FA57-AE86-4103-B07E-
80ECE0000AEE'" );

            RockMigrationHelper.UpdateBlockType( "Referral Agency Detail", "Displays the
details of a Referral Agency.", "~/Plugins/org_rsc/SampleProject/ReferralAgencyDetail.ascx",
"org_rsc > Sample Project", "2F130DF6-1EE4-45CE-9410-CBB0517EB33E" );
```

RockMigrationHelper

- AddBlockType
- UpdateBlockTypeAttribute
- AddBlock
- AddPage
- AddPageRoute
- AddDefinedType
- AddDefinedValue
- AddEntityAttribute
- AddSecurityAuth
- AddSecurityAuthForPage
- AddSecurityRoleGroup
- AddPage
- AddPageRoute
- AddDefinedType
- AddDefinedValue
- AddEntityAttribute
- ...and many more

github.com/SparkDevNetwork/Rock/wiki/Creating-Rock-Migrations