# Quick start guide to the Sitecore ItemBinding Framework

Bo Breiting: 19-02-2014

This document describes how to get started with the Sitecore ItemBinding framework. After reading this document you will be able to use the default framework to minimize the code that is required to model and access your Sitecore solution.

## The main purpose of the framework

The main purpose of the framework is to eliminate the repetitive code that is usually required to interact with Sitecore. Also it is intended to standardize the interaction with Sitecore to make it easier to instantly recognize and understand how the various parts of a solution works by using common code patterns. Lastly centralizing the item-to-model binding removes the errors that would normally be scattered throughout the solution and ensures that if any errors are found they can be corrected in a single place in the framework.

## Basic requirements to run the framework

The framework consists of a single assembly ItemBinding. So to get the framework up and running on your project you only need to add this assembly. Of course optionally you can choose to include the source code from the framework in your own solution, if you want to introduce local customizations. However this is in no way required as all the basic functionality can be extended through configuration.

## Getting started

In order to start using the framework all you need to do is create a model class that should represent some kind of item in Sitecore and create a new sublayout that inherits from the ModelBoundUserControl that is located in the framework assembly and bind it to the model that you created. It is that simple.

### The example case

In the following example we assume that we have a Newslist page in Sitecore that inherits from the Newslist template that contains a single line text field named Title. In addition we have a collection of News pages located as child items beneath the Nzewslist page. The News pages inherits from the News template that contains a single line text field named Title, a multi line text field named Teaser and a rich text field named Text. The Newslist page has a NewslistView sublayout attached and the News pages has a NewsView sublayout attached. The next step will be to extract data from the Newslist page and the News pages and display it in the NewslistView and NewsView sublayouts.

### Creating the model classes

First we will need to create two model classes in the solution to bind our Sitecore Newslist and News items to.

The news model class:

using System;

using ItemBinding.Model;

using ItemBinding.Model.BindingContracts;

using Sitecore.Data;

using Sitecore.Data.Items;

using Sitecore.Links;

namespace ItemBindingExamples.Model

{

  [RequiredBaseTemplate("{7F97671B-E1CA-4537-823C-230A7EAC4DED}")]

 public class News : ItemModel

  {

    public News(Item item) : base(item)

    {

    }

    public readonly ID TitleFieldId = new ID("{82B7DEB1-293A-49F4-8434-57B9B62F420F}");

    public readonly ID TeaserFieldId = new ID("{D642174A-591B-42B2-82BA-CFED87CE7B67}");

    public readonly ID TextFieldId = new ID("{15BB6EFA-2CDF-4F43-B14F-B682C09B329E}");

    public DateTime Created

    {

      get

{

return Sitecore.DateUtil.IsoDateToDateTime(InnerItem[Sitecore.FieldIDs.Created]);

}

    }

    public String Url

    {

      get { return LinkManager.GetItemUrl(InnerItem); }

    }

  }

}

The News class inherits from the ItemModel class that will store the Sitecore item argument from the constructor in a public InnerItem property. In addition the News class is decorated with the RequiredBaseTemplate attribute. This attributes specifies that in order for an item to be bound to the model the item has to inherit from the News template. Lastly the model class exposes the Sitecore IDs of the fields from the News template and the two computed properties Created and Url.

The NewsList model class:

using System.Collections.Generic;

using ItemBinding.Model;

using ItemBinding.Model.BindingContracts;

using Sitecore.Data;

using Sitecore.Data.Items;

namespace ItemBindingExamples.Model

{

  [RequiredBaseTemplate("{ECC0A69D-F1B7-41E5-A4FC-83E4B086D995}")]

  public class NewsList : ItemModel

  {

    public NewsList(Item item) : base(item)

    {

    }

    public readonly ID TitleFieldId = new ID("{EE094CA2-E731-4684-8A85-23C0349692A0}");

    public IEnumerable<News> News

    {

      get { return InnerItem.BindChildrenAs<News>(); }

    }

  }

}

The NewsList class also inherits from the ItemModel and is decorated with the RequiredBaseTemplate attribute that specifies that the bound item have to inherit from the NewsList template in order to be bound to the NewsList class. In the same way as the News model it exposes the Sitecore ID of its single Title field. In addition it exposes a collection of News objects that is populated by using the item extension BindChildrenAs<T>() which is provided by the ItemBinding framework. To learn more about the available extensions in the framework please read the detailed reference documentation.

### Creating the sublayouts

Now we need to create the NewsListView and NewsView sublayouts. Start by creating two normal ASP.NET User Controls (.ascx files) called NewsListView and NewsView. Then change the user controls to inherit from the ModelBoundUserControl available in the ItemBinding framework.

The NewsList ModelBoundUserControl codebehind:

using ItemBinding.Presentation;

using ItemBindingExamples.Model;

namespace ItemBindingExamples.Presentation

{

  public partial class NewsListView : ModelBoundUserControl<NewsList>

  {

  }

}

The NewsListView inherits from the ModelBoundUserControl and the model type is set to the Newslist class.

The News ModelBoundUserControl codebehind:

using ItemBinding.Presentation;

using ItemBindingExamples.Model;

namespace ItemBindingExamples.Presentation

{

  public partial class NewsView : ModelBoundUserControl<News>

  {

  }

}

The NewsView inherits from the ModelBoundUserControl and the model type is set to the News class.

### Accessing the Model

That is actually it. You have now successfully created two model classes and bound them to the two sublayouts. Now you can access the Model property exposed by the ModelBoundUserControl either in codebehind or in markup and it will be default bound to the Sitecore.Context.Item.

The NewsListView ModelBoundUserControl markup:

**<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="NewsListView.ascx.cs"**

**Inherits="ItemBindingExamples.Presentation.NewsListView" %>**

**<%@ Register TagPrefix="sc" Namespace="Sitecore.Web.UI.WebControls" Assembly="Sitecore.Kernel" %>**

**<div class="news-list">**

**<h2>**

**<sc:Text Item="<%# Model.InnerItem %>" Field="<%# Model.TitleFieldId %>" runat="server" />**

**</h2>**

**<asp:Repeater DataSource="<%# Model.News.Reverse() %>"**

**ItemType="ItemBindingExamples.Model.News" runat="server">**

**<HeaderTemplate>**

**<div class="news">**

**</HeaderTemplate>**

**<ItemTemplate>**

**<div class="news-item">**

**<div class="date"><%# Item.Created.ToString("dd. MMM yyyy") %></div>**

**<h3><a href="<%# Item.Url %>">**

**<sc:Text Item="<%# Item.InnerItem %>" Field="<%# Item.TitleFieldId %>" runat="server" />**

**</a></h3>**

**<div class="teaser">**

**<sc:Text Item="<%# Item.InnerItem %>" Field="<%# Item.TeaserFieldId %>" runat="server" />**

**</div>**

**</div>**

**</ItemTemplate>**

**<FooterTemplate>**

**</div>**

**</FooterTemplate>**

**</asp:Repeater>**

**</div>**

The Model property from the ModelBoundUserControl can now be accessed and will expose the referenced model class instance bound to the Sitecore.Context.Item.

The NewsView ModelBoundUserControl markup:

**<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="NewsView.ascx.cs"**

**Inherits="ItemBindingExamples.Presentation.NewsView" %>**

**<%@ Register TagPrefix="sc" Namespace="Sitecore.Web.UI.WebControls" Assembly="Sitecore.Kernel" %>**

**<div class="news">**

**<h2>**

**<sc:Text Item="<%# Model.InnerItem %>" Field="<%# Model.TitleFieldId %>" runat="server" />**

**</h2>**

**<p>**

**<sc:Text Item="<%# Model.InnerItem %>" Field="<%# Model.TextFieldId %>" runat="server" />**

**</p>**

**</div>**

The Model property from the ModelBoundUserControl can now be accessed and will expose the referenced model class instance bound to the Sitecore.Context.Item.