


Embrace the Buzz: Building Modern Extensions for DotNetNuke

Ian Robinson

Ian Robinson

Top 10%  for [jquery](#) [asp.net](#) [cms](#) [dotnetnuke](#)

Top 20%  for [javascript](#) [css](#)

- **Session Feedback!** <http://www.stldodn.com>
- Twitter [@i-robinson](#)
- GitHub <http://github.com/i-robinson>
- StackOverflow Careers <http://bit.ly/i-robinson>

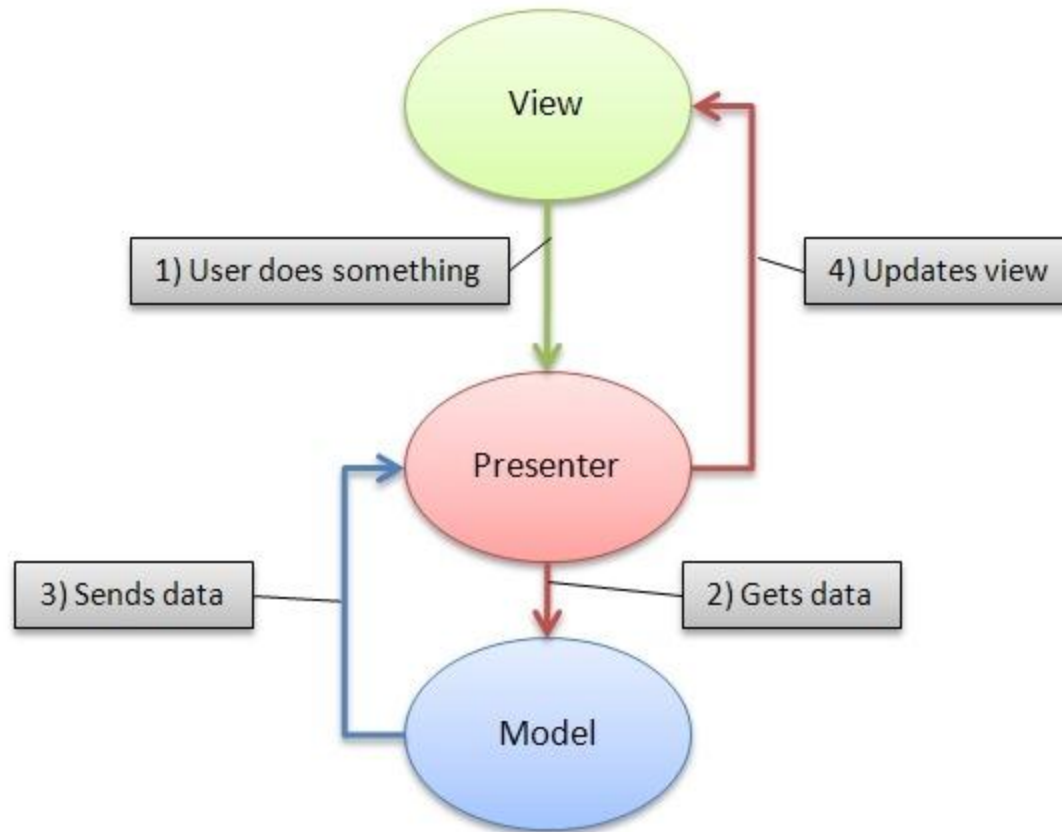
Agenda

- WebFormsMVP / Unit Testing
- jQuery & Ajax
- Razor Scripting Language

WebFormsMVP

- Separation of concerns
- Increased Testability
- “Cleaner Architecture”
- Retains basic characteristics of WebForms

WebFormsMVP



WebFormsMVP Characteristics

- Requires the view to give control to Presenter
- The view and presenter pass around a model
- Interfaces used for testability

WebFormsMVP in DNN

- Introduced in Version 5.3
- WebFormsMVP.dll bundled
- Thin layer on top to provide context

Real World DNN Tests

- DotNetNuke Core Tests
- Vocabulary & Messaging Modules
- DotNetNuke_Community_UnitTests.sln

BeerCollectionMVP

- C#
- WebFormsMVP
- DotNetNuke Module
- Web Application Project
- Visual Studio 2010
- MOQ & MbUnit
- <http://github.com/irobinson>

Key Take-Aways

- Unit testing increases confidence in your code. Code is more changeable.
- Therefore Unit Testing is a desirable practice
- WebFormsMVP allows us to better conduct unit testing in DotNetNuke

Recommended Resources

- Unit Testing
 - The Art of Unit Testing by Roy Osherove
 - Working with Legacy Code by Michael Feathers
- WebFormsMVP
 - Podcast: Hanselminutes Episode 202
 - <http://webformsmvp.com>

“AJAX”

- Does Web 2.0 even mean anything anymore? (Did it ever?)
- Is it over the top animations?
- Or is it a compelling user experience?
- Taste still counts.

AJAX

- Asynchronous server communication
- ASP.NET Sans-Postbacks
- More control / more work?
- “Closer to the metal”

jQuery

- JS Library
- One way of doing things...
- Ease of use
- Adoption / Community
- Documentation

jQuery AJAX

- `jQuery.get()` //http get
- `jQuery.load()` //load page
- `jQuery.post()` //http post
- `jQuery.ajax()` //ajax swiss army knife

jQuery AJAX Events

- Global events for re-use
 - `ajaxStart()`
 - `ajaxError()`
 - `ajaxComplete()`
 - `ajaxStop()`

Working with Data

- Building it all up on the client side.
- `jQuery.data()`

Collecting Simple Data

- `AddUserName(string user)`
- `var username = $('#username').text();`

Collecting Complex Data

```
var budgetItems = new Array();
$('#income-list li').each(function (intIndex) {
    var $item = $(this);
    var incomeItem = {
        categoryId: $item.data('categoryId'),
        Amount: $item.data('amount')
    }
    budgetItems.push(incomeItem);
});
```

Services

- ASMX
- WCF
- ASHX
- ASPX
- Pages Methods
- It's just http...

Service Proxy

- Rick Strahl's ServiceProxy.js
- "JSON ServiceProxy to call WCF or ASMX JSON Services with Date Parsing"
- <http://codepaste.net/i89xhc>

Patterns

- Autocomplete
- Form Submission
- Drag and Drop
- Auto-refresh

More to Think About

- Structuring JavaScript
- JavaScript Combination/Minification
- Creating jQuery plugins
- Automated JavaScript Testing

Introducing...Razor

- View Engine? Scripting language? Templating language?
- Gu Says: “HTML generation ... code-focused templating”
- Clean syntax, fewer characters
- Code is “first class”

syntax

- ASPX

```
<p>Hello Ian, the year is <%= DateTime.Now.Year %></p>
```

- Razor

```
<p>Hello Ian, the year is @DateTime.Now.Year</p>
```

Syntax

- ASPX

```
<ul id="products">
  <% foreach(var p in products) { %>
    <li><%= p.Name %> (%<%= p.Price%>)</li>
  <% } %>
</ul>
```

- Razor

```
<ul id="products">
  @foreach (var p in products) {
    <li>@p.Name ($@p.Price)</li>
  }
</ul>
```

Helpers

```
@helper SayHello(string yadda)
{
    <p>Text: @yadda</p>
}

@SayHello("Hello, world!")
```

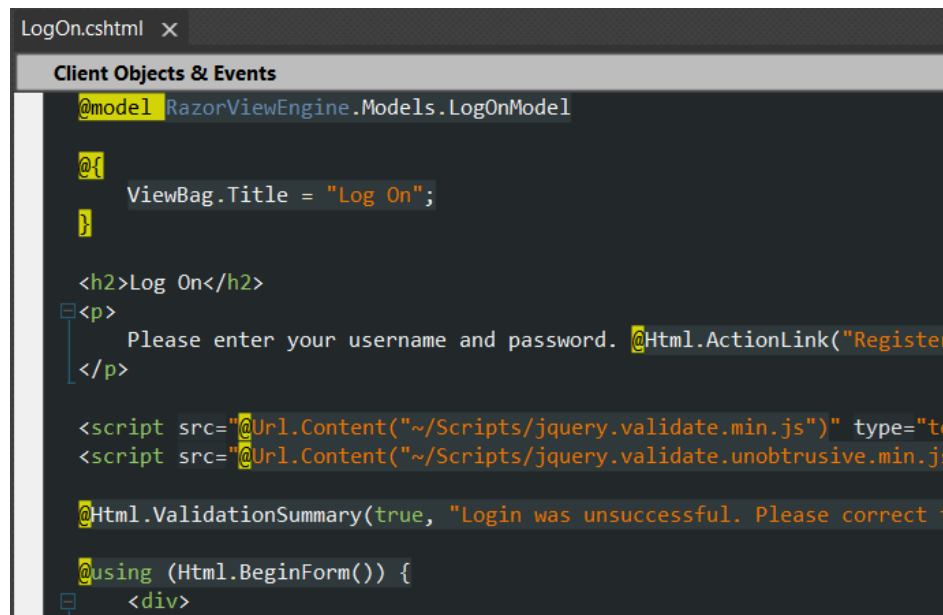
ASP.NET MVC 3

- Ships with choice of ASPX or Razor as view engine
- Displaying and working with data (your model)
- Cool: Automatic HTML Encoding

ASP.NET MVC 3 is an Open Source MVC Framework on .NET

Version 3 Released January 2011

ASP.NET MVC 3 DEMO



The screenshot shows the 'Client Objects & Events' window in an IDE, displaying the code for the 'LogOn.cshtml' view. The code is written in Razor syntax and includes a model, a ViewBag title, HTML tags, a paragraph with a link, script tags for jQuery validation, a validation summary, and a form wrapper.

```
LogOn.cshtml x
Client Objects & Events
@model RazorViewEngine.Models.LogOnModel

@{
    ViewBag.Title = "Log On";
}

<h2>Log On</h2>
<p>
    Please enter your username and password. @Html.ActionLink("Register", "Register", "Account")
</p>

<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")" type="text/javascript"></script>

@Html.ValidationSummary(true, "Login was unsuccessful. Please correct the errors.")

@using (Html.BeginForm()) {
    <div>
```

WEB MATRIX

- Think “PHP Story”
- “Entry level” application creation
- “Web Pages” **.cshtml**
- In this world...
 - **Razor scripting allows dynamic features**

DotNetNuke

- Enable light weight but powerful development
- Not **the** new approach, but **a** new approach
- Razor host module
- Similar to web matrix style development
 - Less the nice IDE
 - Add DNN Context (API)

Razor host module

▼ Edit Script

Select Script:

_TabInfo.cshtml

+ Add New Script File

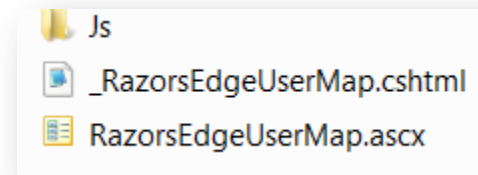
Source File - ~/DesktopModules/RazorModules/RazorHost/Scripts/_TabInfo.cshtml

Script Source:

```
<h2>Razor Page Info Module Example</h2>
<div>The current page is:</div>
<ul>
  <li>Name: @Dnn.Tab.TabName</li>
  <li>Id: @Dnn.Tab.TabID</li>
  <li>Path: @Dnn.Tab.TabPath</li>
</ul>
```


A razor module

- MyModule.ascx
 - Inherits from RazorModuleBase
- _MyModule.cshtml
 - Contains html/js/razor
- Host -> Module Definitions
 - Create new module -> **From Control**



User map demo

```
@using DotNetNuke.Security.Roles
@using DotNetNuke.Entities.Users
@using DotNetNuke.Entities.Profile

@functions {
    public static string GetProfileProperty(UserProfile profile, string propertyName)
    {
        ProfilePropertyDefinition objProperty = profile.GetProperty(propertyName);
        if (objProperty == null || string.IsNullOrEmpty(objProperty.PropertyValue)) return string.Empty;
        return objProperty.PropertyValue;
    }
}
```