Umbraco Packages in v8

Shannon Deminick | twitter: @shazwazza | web: shazwazza.com

Migrating
Building
Packaging

Articulate

- Uses a **lot** of Umbraco surface area
- A good way to test v8!

- Custom routes & controllers
- Virtual nodes
- Content finders
- Event handlers
- Dashboards
- Property Editors
- Package manifest
- Lots of views
- Custom searching
- Custom DB queries

Migrating

MVC

Model

Published content and getting property values

View

APIs available in Razor views

Controller

Changes to controllers and how to create your own

Model changes

- RenderModel doesn't exist, just IPublishedContent
- One way to get property values
- No more dynamcis

Page model(v7)

```
@inherits UmbracoTemplatePage
<!--
    This is equivalent to
    @inherits UmbracoViewPage<RenderModel>
-->
<html>
    <body>
        <h1>
        @Model.Content.GetPropertyValue("title")
        </h1>
    </body>
</html>
```

Page model(v8)

```
@inherits UmbracoViewPage
<!--
    This is equivalent to
    @inherits UmbracoViewPage<IPublishedContent>
-->
<html>
    <body>
        <h1>
        @Model.Value("title")
        </h1>
    </body>
</html>
```

@Model.Value ?!

Property values(v7)

```
<header>
    <img src='@Umbraco.Field("logo", recurse: true)' />
</header>
<h1>@Model.Content.Properties["title"].Value</h1>
<main>
    <div>@Umbraco.Field("description")</div>
    <div>Price: @(Model.Content.GetPropertyValue<double>("price"))</div>
</main>
```

Property values(v8)

```
<header>
    <img src='@Model.Value("logo", fallback:Fallback.ToAncestors)' />
</header>
<h1>@Model.Value("title")</h1>
<main>
    <div>@Model.Value("description")</div>
    <div>Price: @(Model.Value<double>("price"))</div>
</main>
```

View APIs - Traversal

(v7)

```
var root =
  @Model.Content.Site();

var news =
  root.Children("news");

var news2 =
  root.Children<News>();

var art1 =
  news.Descendants("article");

var art2 =
  news.Descendants<Article>();
```

(v8)

```
var root =
  @Model.Content.Root();

var news1 =
  root.ChildrenOfType("news");

var news2 =
  root.Children<News>();

var art1 =
  news.DescendantsOfType("article");

var art2 =
  news.Descendants<Article>();
```

View APIs - URLs

(v7)

```
var nUrl1 =
  news.Url;

var nUrl2 =
  @Umbraco.NiceUrl(news.Id);

var nUrlAbs1 =
  @news.UrlAbsolute();

var nUrlAbs2 =
  @Umbraco.NiceUrlWithDomain(news.Id)
```

(v8)

```
var nUrl1 =
  news.Url;

var nUrl2 =
  @UmbracoContext.Url(news.Id);

var nUrlAbs1 =
  @news.UrlAbsolute();

var nUrlAbs2 =
  @UmbracoContext.UrlAbsolute(news.Id
```

View APIs - Lookup

(v7)

```
var c1 =
  @Umbraco.TypedContent(123);
var c2 =
  @Umbraco.Content(123); //dynamic

var img1 =
  @Umbraco.TypedMedia(987);
var img2 =
  @Umbraco.Media(987); //dynamic

var m1 =
  @Umbraco.TypedMember(555);
var m2 =
  @Umbraco.Member(555); //dynamic
```

(v8)

```
var c1 =
  @Umbraco.Content(123);

//no dynamics!

var img1 =
  @Umbraco.Media(987);

//no dynamics!

var m1 =
  @Umbraco.Member(555);

//no dynamics!
```

~~@CurrentPage~~

XPath vs Linq

*v8's cache is **objects**, not XML*

View APIs - Services

(v7)

```
@Umbraco

@UmbracoContext

@ApplicationContext

@Services

@Members

@PublishedContentRequest
```

(v8)

```
@Umbraco

@UmbracoContext

//No ApplicationContext!

@Services

@Members

@PublishedRequest
```

No singletons please

Example:

`@UmbracoContext.Current`

instead of just

`@UmbracoContext`

- Typical controllers are still auto routed

```
SurfaceController
UmbracoApiController
```

- Controllers are constructed by DI
- Controllers must be registered in the DI Container
- Typical controllers are auto-registered

```
PluginController
IRenderMvcController
UmbracoApiController
```

Custom Controllers

DI = Dependency Injection

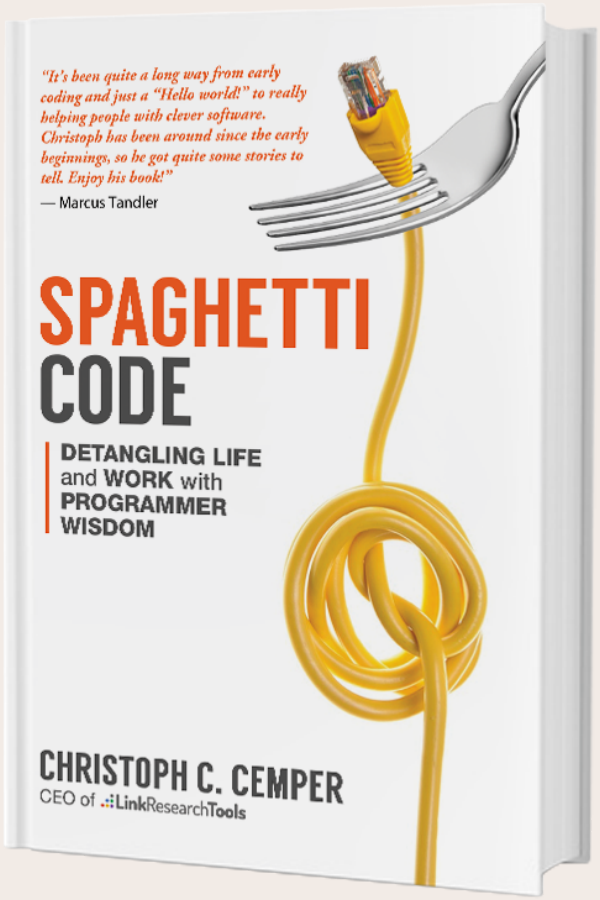
Guaranteed to make you code better

DI = Dependency Injection

It's much easier than you think

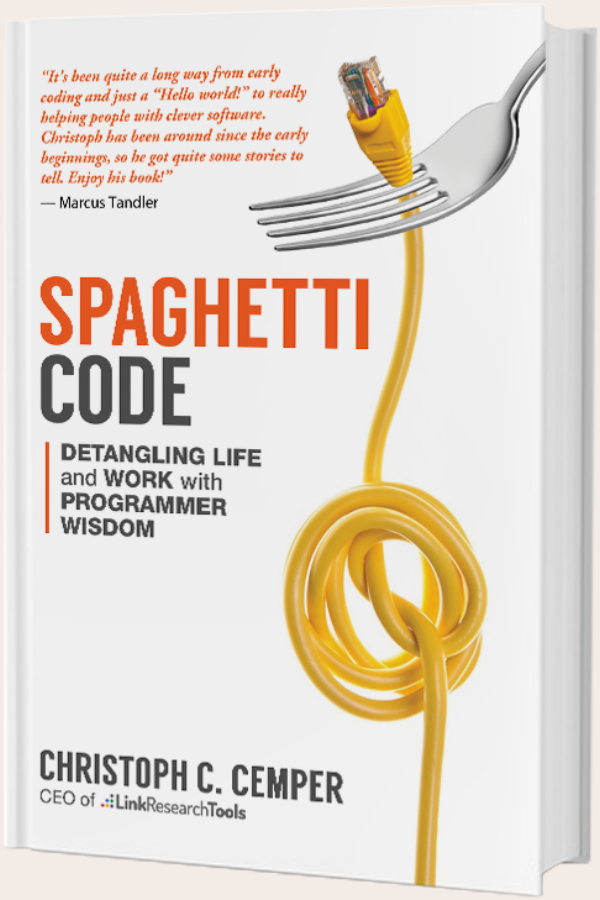
Singletons are no good

Bad Singletons



- Easily cause interdependencies
- Can't view dependency graph
- Difficult to debug
- Difficult to test
- Creates spaghetti code

Good Singletons?



- c# Attributes
- **rarely** in extension methods

(v7) Singleton APIs - If you must (v8)

```
LogHelper
```

```
Current.Logger
```

```
ApplicationContext
  .Current
```

```
// No ApplicationContext
```

```
ApplicationContext
  .Current.Services
```

```
Current.Services
```

```
ApplicationContext
  .Current.ApplicationCache
```

```
Current.AppCaches
```

```
UmbracoContext.Current
```

```
Current.UmbracoContext
```

(v7)

Composers

(v8)

```
public class MyApp : ApplicationEventHandler
{
  protected override void ApplicationStarting(
      UmbracoApplicationBase umbracoApplication,
      ApplicationContext applicationContext)
  {
    ContentFinderResolver.Current
        .Insert<MyFinder>();
  }
}
```

```
public class MyComposer : IComposer
{
  public void Compose(Composition composition)
  {
    composition.ContentFinders()
        .Insert<MyFinder>();
  }
}
```

https://www.zpqrtbnk.net/posts/composing-umbraco-v8/

Register custom controller(v8)

```
public class MyComposer : IComposer
{
  public void Compose(Composition composition)
  {
    //register MyController - lifetime is IMPORTANT!
    composition.Register<MyController>(Lifetime.Request)

    //Example of a custom Singleton service
    composition.RegisterUnique<MyService>();
  }
}
```

(v7) Controller changes (v8)

```
public class MyController : RenderMvcController
{
  public override ActionResult Index(
    RenderModel model)
  {
      //... do stuff...

      return base.Index(model);
  }
}
```

```
public class MyController : RenderMvcController
{
  public override ActionResult Index(
    ContentModel model)
  {
      //... do stuff...

      return base.Index(model);
  }
}
```

Controller injection(v8)

```
public class MyController : RenderMvcController
{
  public MyController(MyService myService,
      ...) : base(...)
  {
      _myService = myService;
  }

  private readonly MyService _myService;

  public override ActionResult Index(
    ContentModel model)
  {
      _myService.DoStuff(model);
```

Building

Package Manifest

/App_Plugins/MyPlugin/package.manifest

```
{
  "propertyEditors": [],
  "parameterEditors": [],
  "gridEditors": [],
  "javascript": [],
  "css": [],
  "dashboards": [],
  "sections": [],
  "contentApps": []
}
```

Dashboards

https://our.umbraco.com/Documentation/Extending/Dashboards/index-v8

```
{
  "dashboards":  [
    {
      "alias": "myCustomDashboard",
      "view":  "~/App_Plugins/MyPackage/dashboard.html",
      "sections":  [ "content", "settings" ],
      "weight": -10,
      "access": [
        { "deny": "translator" },
        { "grant": "admin" }
      ]
    }
  ]
```

Dashboards in c#

https://our.umbraco.com/Documentation/Extending/Dashboards/index-v8

```
[Weight(-10)]
public class MyDashboard : IDashboard
{
    public string Alias => "myCustomDashboard";
    public string[] Sections => new[] { "content", "settings" };
    public string View => "~/App_Plugins/MyPackage/dashboard.html";
    public IAccessRule[] AccessRules => Array.Empty<IAccessRule>();
}
```

***Currently** assembly scanned*

Sections

~~/config/applications.config~~

```
{
  "sections":  [
    {
      "alias": "myPackage",
      "name":  "My Package"
    }
  ]
}
```

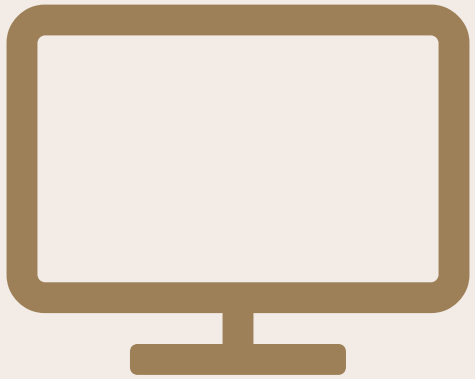
Sections in c#

~~/config/applications.config~~

```
public class MyPackageSection : ISection
{
    public string Alias => "myPackage";
    public string Name => "My Package";
}
```

Manually registered

Full Screen Sections!?



- Kind of happened by fluke
- Just a section + dashboard
- No tree required
- = No c# required

Let's see!

Content Media Settings Packages Users

Articulate

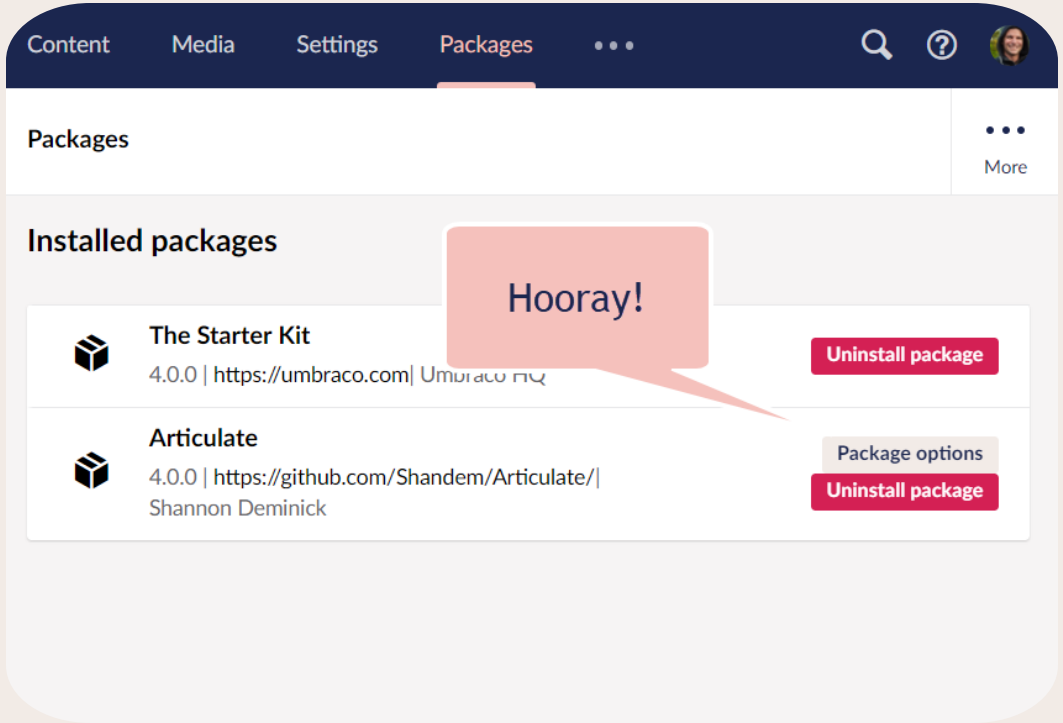
Articulate installed

The installer has installed all of the required Umbraco nodes including some demo data. You can either modify this demo data or simply remove it once you are comfortable with how Articulate works. The demo data includes: a blog post, an author, a category and a tag.

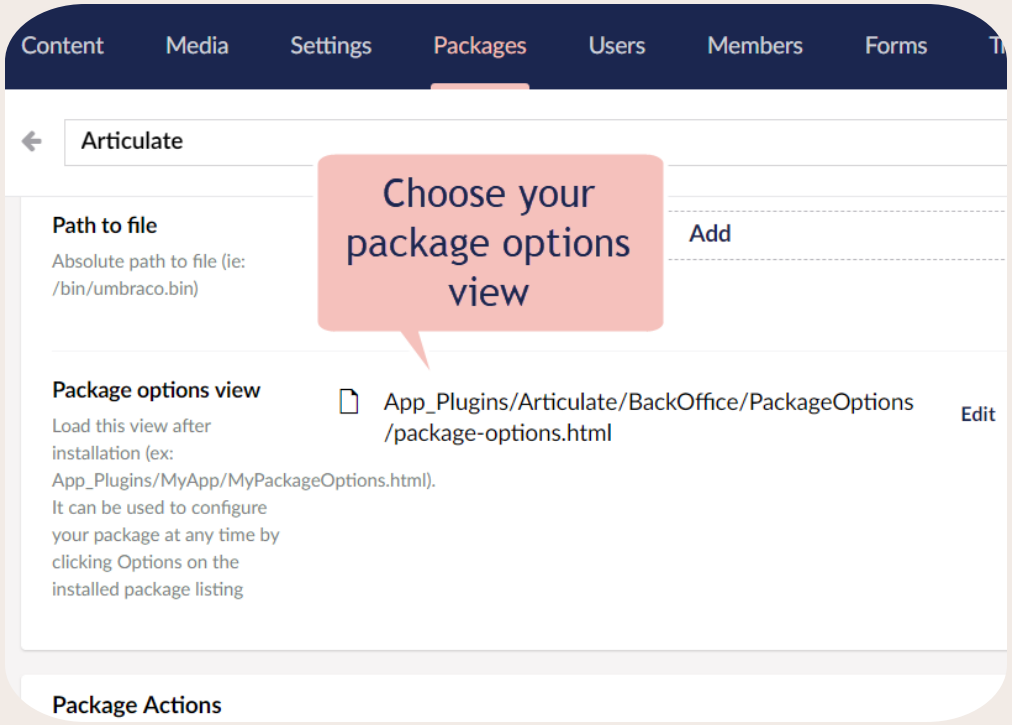
To customize your blog navigate to the Content section and click on the Articulate 'Blog' node. Here you can customize the look and feel of your blog, including changing the theme, adding Google analytics tracking, etc... If you want comments enabled you should sign up for a Disqus account and ensure you enter your Disqus details on the 'Blog' node.

Click Here to view Articulate documentation.

Package Options



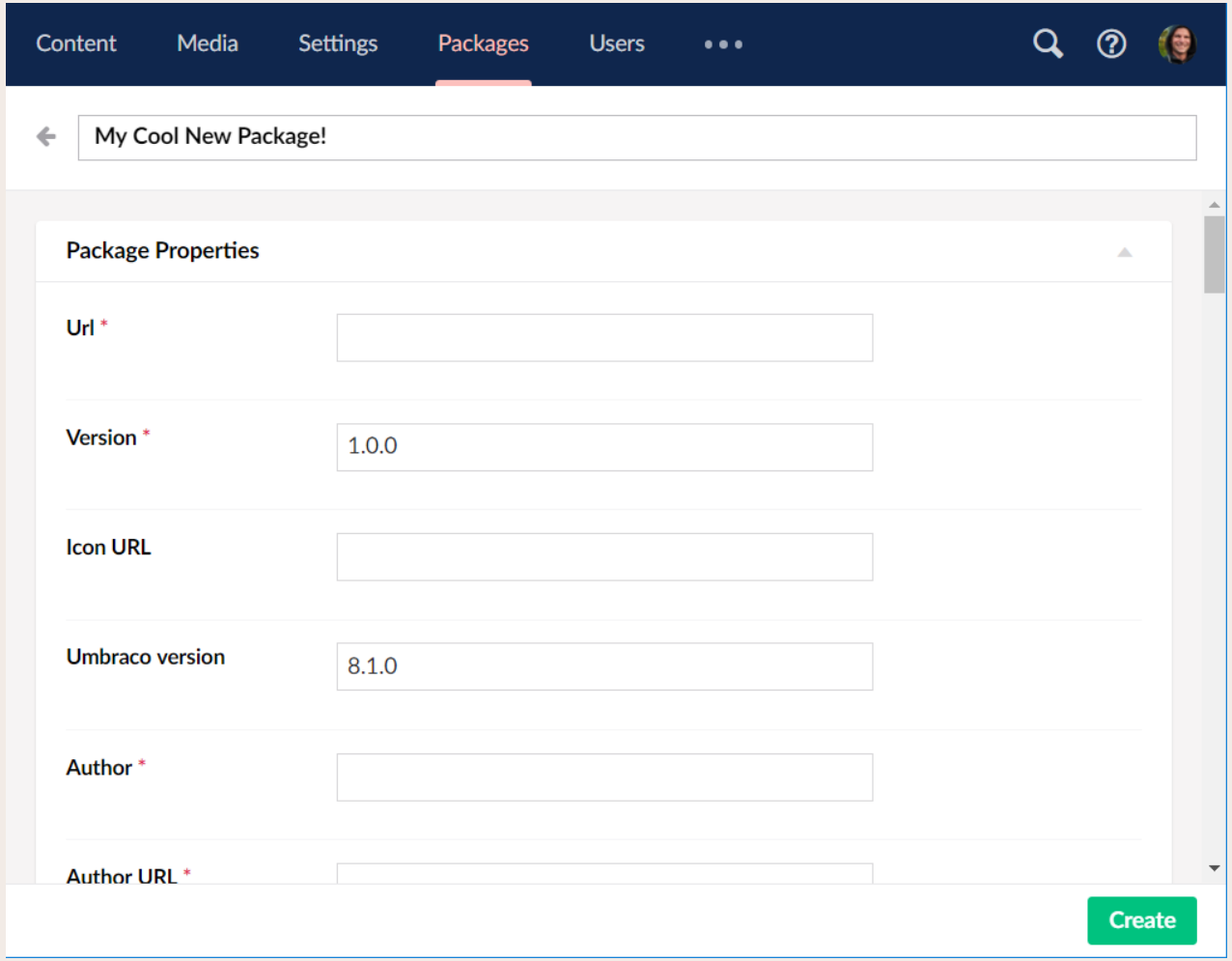
Package Options



Let's see!

Packaging

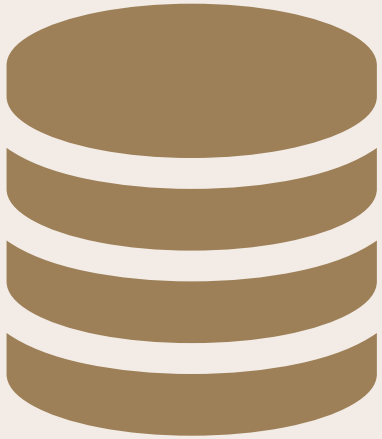
Packaging UI



What about CI/CD?

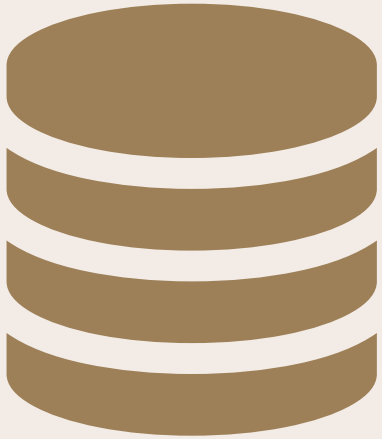
...and Nuget?

Umbraco package data installation



- Packager installs files
- Packager installs schema
- c# code to install content + media
- Package Action to invoke c#

Nuget package data installation



- Nuget installs files
- Package Options installs schema
- c# code to install content + media

Can this be better?

Package Migrations FTW

Questions?

Shannon Deminick | twitter: @shazwazza | web: shazwazza.com