Core internals for Website Development

Keep it simple-ish

2014/06/11

Stéphane Gay
stephane@umbraco.dk
@zpqrtbnk

Umbraco is great…

And yet Completely Broken ™

Fixing & Refactoring

IAmTheOneToBlameForThoseFancyInterfaceNames

Simple things should be simple
Refactor for simplicity & predictability

Complex things should be possible
Refactor for extensibility & modularity

Constant work-in-progress
New ways to build websites
Community feedback loop

CG14

Tea Spoon World

In Depth

Size matters – what makes a teaspoon

Read more

In Depth

Illegal teaspoon usages

Read more

In Depth

If you can read this you are a

Latest News

June 1, 2014

Queen to discuss teaspoon etiquette with parliament

Read more

May 28, 2014

US to defend right to keep and bear teaspoons

Read more

May 12, 2014

Multiple websites, multiple languages

Sites demo1.com and demo2.com
Both websites should exist both in English and French
Both will publish news items (title, body, image, map location)

Multiple environments

To get things validated by upper management,
Customer wants www.demo1.com and staging.demo1.com

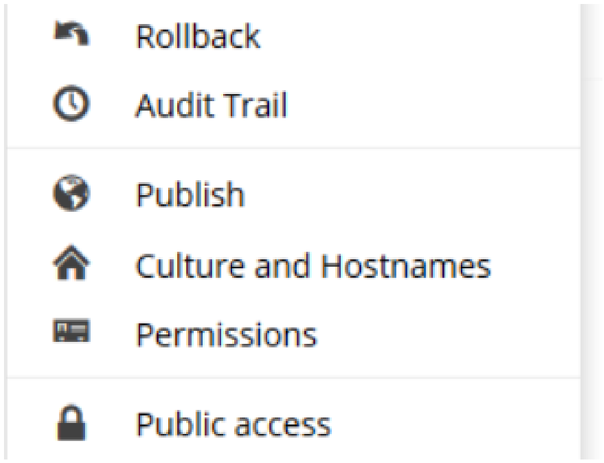
Special URLs

/news/2014-06-11-helicopters-to-deliver-teaspoons

Setup sites, domains and languages

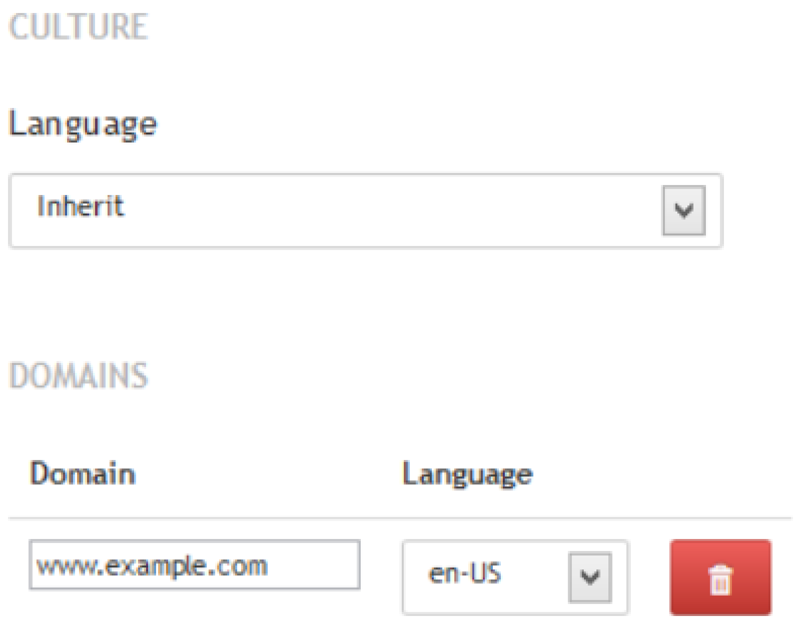
CONTENT

- Demo 1
  - English
    - News
      - Teaspoon something
  - French
- Demo 2
  - English
  - French



CONTENT

- Demo 1
  - English
    - News
      - Teaspoon something
  - French
- Demo 2
  - English
  - French



CULTURE

Language

Inherit

DOMAINS

Domain	Language
www.example.com	en-US

CONTENT

Demo 1	Domains: www.demo1.com, staging.demo1.com
English	Culture: en-US
News	
Teaspoon something	
French	Culture: fr-FR
Demo 2	Domains: www.demo2.com, staging.demo2.com
English	Culture: en-US
French	Culture: fr-FR

http://www.demo1.com/ → ??

Umbraco wants to render node "Demo1"
We want to redirect
  to "Demo1 / English"
  to "Demo1 / French"

PublishedContentRequest.Prepared

```
PublishedContentRequest.Prepared += (sender, args) =>
{
  var request = sender as PublishedContentRequest;

  if (request == null || !request.HasPublishedContent) return;
  if (request.PublishedContent.ContentType.Alias != "Site") return;

  var roots = request.PublishedContent.Children.OfTypes("HomePage");
  if (!roots.Any()) return;

  var httpRequest = request.RoutingContext.UmbracoContext.HttpContext.Request;

  var lang = FigureOutLanguageFromRequest(httpRequest);

  var root = PickTheMostAppropriateRoot(roots, lang);

  request.SetRedirect(root.Url);
}
```

Urls, Relative vs. Absolute

Navigate demo1, request the url of a Demo1 page → relative
Navigate demo1, request the url of a Demo2 page → absolute

Also in the backend eg. admin.demos.com !

Urls, Picking the right domain

www.demo1 ↔ www.demo2
staging.demo1 ↔ staging.demo2

```
SiteDomainHelper.AddSite("www", "www.demo1.com", "www.demo2.com");
SiteDomainHelper.AddSite("staging", "staging.demo1.com", "staging.demo2.com");
```

404 Page Not Found

Just render a page named 404

CONTENT

- Demo 1
  - English
    - News
    - Articles
    - 404
  - French
- Demo 2
  - English
  - French

RIP NotFoundHandler

IContentFinder.TryFindContent(PublishedContentRequest request)

The “last chance” content finder

```csharp
public class Find404 : IContentFinder
{
  public bool TryFindContent(PublishedContentRequest contentRequest)
  {
    if (!contentRequest.HasDomain) return false;

    var contentCache = contentRequest.RoutingContext.UmbracoContext.ContentCache;
    var domainRoot = contentCache.GetById(contentRequest.Domain.RootNodeId);

    var firstSegment = contentRequest.Uri.AbsolutePath.Split(new [] {'/'},
                                  StringSplitOptions.RemoveEmptyEntries).First();
    var root = domainRoot.Children.FirstOrDefault(x => x.UrlName == firstSegment);

    root = root ?? domainRoot.Children.First();

    var page = root.Descendants().FirstOrDefault(x => x.Name == "404");
    if (page == null) return false;

    contentRequest.PublishedContent = page;
    var wcd = Domain.GetDomainsById(root.Id, true).SingleOrDefault(x => x.IsWildcard);
    if (wcd != null)
      contentRequest.Culture = new CultureInfo(wcd.Language.CultureAlias);
    return true;
  }
}
```

```
public class ConfigureMy404Finder : ApplicationEventHandler
{
  public override void ApplicationStarting(
    UmbracoApplicationBase umbracoApplication,
    ApplicationContext applicationContext)
  {
    ContentLastChanceFinderResolver.Current.SetFinder(new Find404());
  }
}
```

Nice & clean urls

“Spécial glöbst & cüsmîdt @ Frånzenøx !”

→ spécial-glöbst-cüsmîdt-@-frånzenøx
→ spécial-glöbst-cüsmîdt-frånzenøx
→ special-globst-cusmidt-franzenox

How many Unicode characters are there?

```xml
<settings>
  <requestHandler>
    <urlReplacing removeDoubleDashes="true">
      <char org="!"></char>
      <char org="#"></char>
      …
    </urlReplacing>
  </requestHandler>
</settings>
```

```xml
<settings>
  <requestHandler>
    <urlReplacing toAscii="true" />
  </requestHandler>
</settings>
```

Simples things
Are becoming simpler
Built-in ASCII filter
Built-in 240 chars limit…

Complex things
Can be complex: IShortStringHelper
Processes every "short" string
Creates aliases, url segments…

String extension methods: ToUrlSegment(), ToCleanString()…

Replace it, extend it, whatever

Custom news item urls

Requirement:	“2014-06-11-helicopters-to-deliver-teaspoons”
Content name:	“Helicopters to deliver teaspoons”
Default url segment:	“helicopters-to-deliver-teaspoons”

IUrlSegmentProvider

Provides the url segment at publish time

Native support (no need for rewriting, content finder, anything)

```
public class UrlSegmentProvider : IUrlSegmentProvider
{
    public string GetUrlSegment(IContentBase content)
    {
      var services = ApplicationContext.Current.Services;
      var contentType = services.GetContentType(content.ContentTypeId);
      if (contentType.Alias != "NewsItem") return null;

      var date = content.GetValue<DateTime>("date").ToString("yyyy-MM-");
      return date + content.Name.ToUrlSegment();
    }
}
```

Beyond the url segment
When you want to create a completely different url

IUrlProvider
Provides the complete content url when required
Needs an inbound strategy (rewriting, content finder, something)

/products/a/aluminum → /products/aluminum

Render content

The NewsItems page

Open the template

```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage
@{
    Layout = null;
}
```

UmbracoTemplatePage
UmbracoTemplatePage<TContent>
UmbracoViewPage
UmbracoViewPage<TModel>
RenderModel

?!

UmbracoViewPage<TModel>
ApplicationContext, UmbracoContext, UmbracoHelper
Maps IRenderModel to/from IPublishedContent

UmbracoViewPage
    : UmbracoViewPage<IPublishedContent>
Model is IPublishedContent

UmbracoTemplatePage
    : UmbracoViewPage<RenderModel>
Model is RenderModel
Model.Content is IPublishedContent
dynamic CurrentPage

So?!

Dynamic
@CurrentPage.MyProperty
→ UmbracoTemplatePage

Otherwise
@Model.GetPropertyValue("myProperty")
@Model.Content.GetPropertyValue<string>("myProperty")
→ UmbracoViewPage

Strongly typed
@Model.MyProperty
→ UmbracoViewPage<MyContentType>
MyContentType : IPublishedContent

Complex model
→ UmbracoViewPage<MyModel>
MyModel : IPublishedContent or IRenderModel for Umbraco.Field()

So?!

Dynamic
Gives easy access to properties eg CurrentPage.MyProperty
Gives easy (?) navigation eg. CurrentPage.Products
But...

Not a bad choice today
Hope to make it redundant at some point

News item page template, first try

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage

<div>@Model.GetPropertyValue("title")</div>

@foreach (var newsItem in Model.Children()
                       .Where(x => x.DocumentTypeAlias == "newsItem")
                       .OrderByDescending(x => x.GetPropertyValue<DateTime>("date"))
                       .Take(8))
{
    <div class="date">@Model.GetPropertyValue<DateTime>("date").ToString("D")</div>
    <div class="title">@Model.GetPropertyValue<string>("title")</div>

    @* HERE I WANT TO SHOW THE IMAGE *@

    @* HERE I WANT TO SHOW A MAP LOCATION *@
}
```

Alternate styling, Index and ToContentSet

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage

@foreach (var newsItem in …Where(x => x.GetPropertyValue<string>("topic") == "food"))
{
    <div class="@(newsItem.IsOdd() ? "news-odd" : "news-even")">

        @* DETAILS *@

    </div>
}
```

Alternate styling, Index and ToContentSet

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage

@foreach (var newsItem in …Where(x => x.GetPropertyValue<string>("topic") == "food")
                             .ToContentSet())
{
    <div class="@(newsItem.IsOdd() ? "news-odd" : "news-even")">

        @* DETAILS *@

    </div>
}
```

Quick win

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage

<div>@Model.GetPropertyValue("title")</div>

@foreach (var newsItem in Model.Children()
    .OfTypes("newsItem")
    .OrderByDescending(x => x.GetPropertyValue<DateTime>("date"))
    .Take(8))
{
    <div class="date">@Model.GetPropertyValue<DateTime>("date").ToString("D")</div>
    <div class="title">@Model.GetPropertyValue<string>("title")</div>

    @* HERE I WANT TO SHOW THE IMAGE *@

    @* HERE I WANT TO SHOW A MAP LOCATION *@
}
```

Show the image…

```
@{
    var imageId = Model.GetPropertyValue<int>("image");
    var image = Umbraco.TypedMedia(imageId);
    if (!image.HasValue("umbracoFile")) image = null;
}
@if (image != null)
{
    var alt = image.GetPropertyValue<string>("legend");
    alt = string.IsNullOrEmpty(alt) ? Image.Name : alt;
    <div class="image"><img src="@image.Url" src="@alt" /></div>
}
```

Show the map location...

```
@{
    var location = Model.GetPropertyValue<string>("location");
    float latitude = 0, longitude = 0;
    if (string.IsNullOrEmpty(location) == false)
    {
        var parts = location.Split(new[] {','});
        latitude = float.Parse(parts[0]);
        longitude = float.Parse(parts[1]);
    }
}
@if (string.IsNullOrEmpty(location) == false)
{
    <div class="location">
        <img src="http://maps.com/?lat=@latitude&lng=@longitude" />
    </div>
}
```

Happy?

IPropertyValueConverter

Could have been IPublishedContentPropertyFromDbValueConversionProvider

Ensures that GetPropertyValue returns a meaningful value

Some are builtin
Write your own

Convert location properties…

```
public class LocationConverter : PropertyValueConverterBase
{
    public override bool IsConverter(PublishedPropertyType propertyType)
    {
        return "Demo.Location".Equals(propertyType.PropertyEditorAlias);
    }

    public override object ConvertDataToSource(PublishedPropertyType propertyType,
                                               object source, bool preview)
    {
        var stringSource = source as string;
        if (string.IsNullOrWhiteSpace(stringSource)) return null;

        var coords = stringSource.Split(new[] {','});
        return new Location(float.Parse(coords[0]), float.Parse(coords[1]));
    }
}
```

Getting better…

```
@{
    var image = Model.GetPropertyValue<IPublishedContent>("image");
    // umbracoFile test done by the converter
}
@if (image != null)
{
    var alt = image.GetPropertyValue<string>("legend");
    alt = string.IsNullOrEmpty(alt) ? Image.Name : alt;
    <div class="image">
        <img src="@image.Url" src="@alt" />
    </div>
}

@{
    var location = Model.GetPropertValue<Location>("location");
}
@if (location != null)
{
    <div class="location">
        <img src="…?lat=@location.Latitude&lng=@location.Longitude" />
    </div>
}
```

Introducing **IPublishedContentModelFactory**
Umbraco 7.1.4

Every IPublishedContent returned by the cache goes through the factory
IPublishedContent CreateModel(IPublishedContent content)

No factory enabled by default
Ships with builtin PublishedContentModelFactory

```
public class NewsItem : PublishedContentModel
{
  public NewsItem(IPublishedContent content)
    : base(content)
  { }

  public DateTime Date
  {
    get { return this.GetPropertyValue<DateTime>("date"); }
  }
}
```

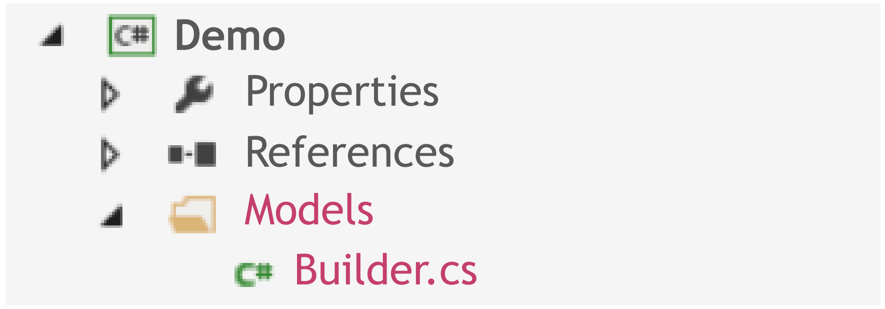
Introducing **Zbu.ModelsBuilder**

Independent tool
github.com/zpqrtbnk

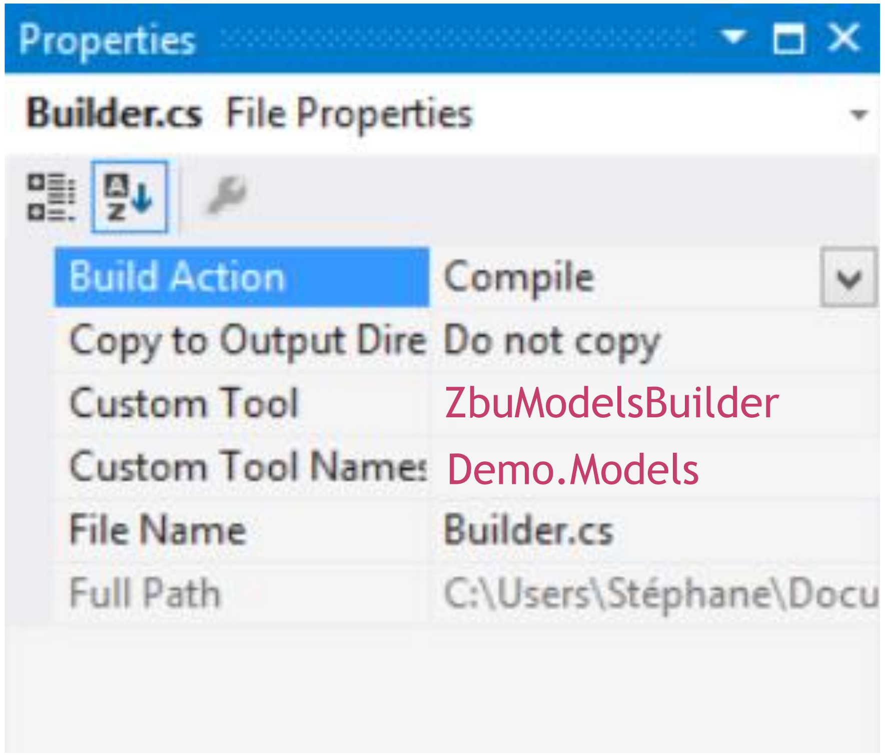
Generates strongly typed models based upon what's in the database
As partial classes
That can be extended
Tweak generation via attributes

Server-side component
Install Umbraco Package or NuGet Package

Visual Studio Extension
Install in Visual Studio
Configure to point to website



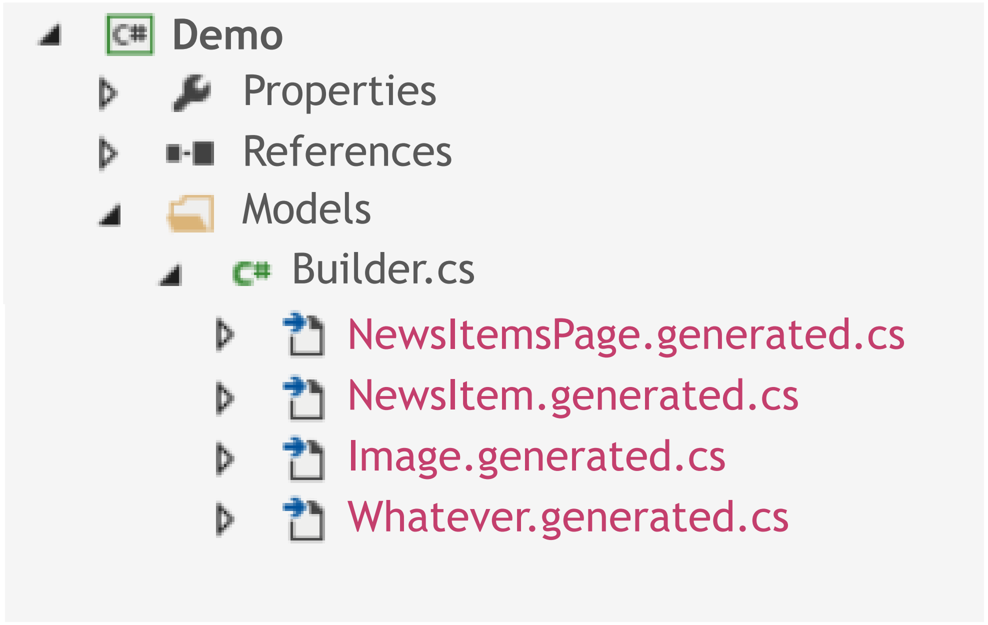
Demo
Properties
References
Models
Builder.cs



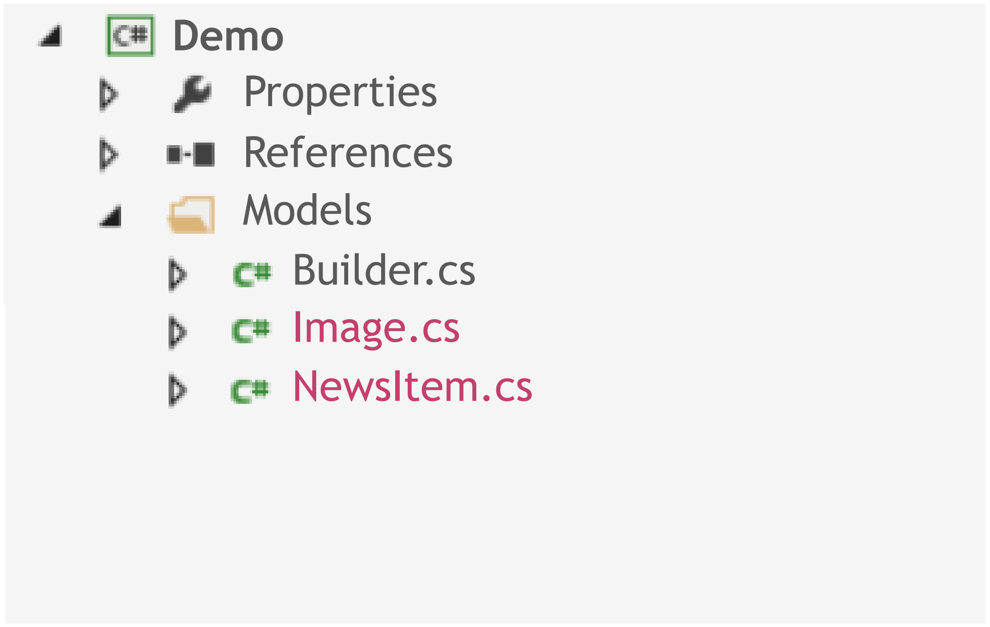
Properties

Builder.cs File Properties

Build Action	Compile
Copy to Output Dire	Do not copy
Custom Tool	ZbuModelsBuilder
Custom Tool Names	Demo.Models
File Name	Builder.cs
Full Path	C:\Users\Stéphane\Docu



```
◢ Demo
    ▷ Properties
    ▷ References
    ◢ Models
        ◢ Builder.cs
            ▷ NewsItemsPage.generated.cs
            ▷ NewsItem.generated.cs
            ▷ Image.generated.cs
            ▷ Whatever.generated.cs
```



Demo
Properties
References
Models
Builder.cs
Image.cs
NewsItem.cs

Image should have an Alt property

```
public partial class Image
{
    public string Alt
    {
        get
        {
            var alt = this.Legend;
            return string.IsNullOrWhiteSpace(alt) ? this.Name : alt;
        }
    }
}
```

NewsItem image should return as an image…

```
public partial class NewsItem
{
    [ImplementPropertyType("image")]
    public Image Image
    {
        get
        {
            var img = this.GetPropertyValue<IPublishedContent>("image") as Image;
            if (img == null || string.IsNullOrEmpty(img.UmbracoFile)) return null;
            return img;
        }
    }
}
```

News item page template

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage<NewsItemsPage>

<div>@Model.Title</div>

@foreach (var newsItem in Model.Children<NewsItem>()
                       .OrderByDescending(x => x.Date).Take(8))
{
    <div class="date">@Model.Date.ToString("D")</div>
    <div class="title">@Model.Title</div>

    @if (Model.Image != null)
    {
        <div class="image">
            <img src="@Model.Image.Url" src="@Model.Image.Alt" />
        </div>
    }

    @if (Model.Location != null)
    {
        <div class="location">
            <img src="…?lat=@Model.Location.Latitude&lng=@Model.Location.Longitude" />
        </div>
    }
}
```

Content models vs. View models

Strongly typed everywhere

Happy?

Simplify content access…

```
@foreach (var newsItem in Model.Children<NewsItem>()
                      .OrderByDescending(x => x.Date)
                      .Take(8))
{
    …

@foreach (var newsItem in Umbraco.Queries.LatestNews(0, 8))
{
    …
```

Queries are still work-in-progress
Experimented during the retreat

What about XPath?

Finally…

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage<NewsItemsPage>

<div>@Model.Title</div>

@foreach (var newsItem in Umbraco.Queries.LatestNews(0, 6))
{
    <div class="date">@Model.Date.ToString("D")</div>
    <div class="title">@Model.Title</div>

    @if (Model.Image != null)
    {
        <div class="image">
            <img src="@Model.Image.Url" src="@Model.Image.Alt" />
        </div>
    }

    @if (Model.Location != null)
    {
        <div class="location">
            <img src="…?lat=@Model.Location.Latitude&lng=@Model.Location.Longitude" />
        </div>
    }
}
```

Almost happy…

```
<div>@Model.Title</div>

<div>@CurrentPage.Title</div>
```

```
@if (CurrentPage.Image != null)
{

@if (CurrentPage.HasImage)
{
```

Happy!

CG14

Feedback & questions!

Many thanks!

2014/06/11

Stéphane Gay
stephane@umbraco.dk
@zpqrtbnk