Rapid umbraco Development

Paul de Metter, Lennard Fonteijn

11th April 2017

ARLANET DIGITAL ENGINEERS

DUUG Dutch Umbraco User Group

Introduction

Paul de Metter
Founder Arlanet, Chieftain

Lennard Fonteijn
.NET Wizard, Lecturer

Once upon a time...

- We built all our solutions on **designs**
- We built all our solutions on **templates**
- We built all our solutions using **content trees**
- Life was simple and easy...

But times have changed

It is not about design anymore.

It is about **functionality** and **user experience**.

But times have changed

It is not about templates anymore.

It is about **flexibility**.

But times have changed

It is not about content trees anymore.

It is about **content models**.

@ Arlanet we...

- Start with creating a **functional wireframe** using our Bootstrap Generator
- This **focusses the customer** on functionality and user experience before design

@ Arlanet we...

- Continue to **define the content model** and separate content from presentation
- This enables the editors to think about **content** and Umbraco about the **presentation**

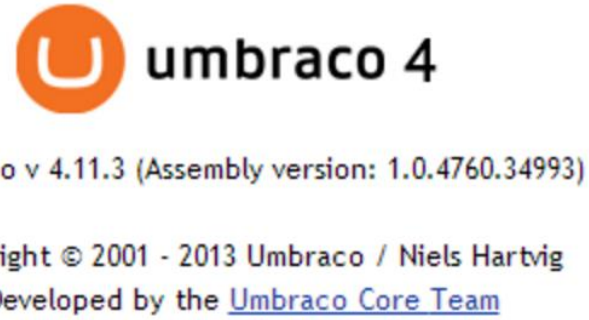
@ Arlanet we...

- Finalize the design process with a **visual design** that builds on the wireframe, UX and content model
- This **focusses the customer** to look at visual design instead of functionality and UX

Now, let's give it to the devs!

Let's get technical (2012)

1. **Download** latest Umbraco.zip and extract to a Visual Studio 2010 project
2. Setup Umbraco, create DocumentTypes, MasterPages, **XSLT** and **Macros**
3. … maybe throw in a custom ASPX control or XSLT library
4. Prepare the content and media tree
5. Test, accept and release



Let's get technical (2017)

1. Install Umbraco with **NuGet** in a Visual Studio 2015 (or 2017) project
2. Think about your **content** for a moment…
3. Setup Umbraco, create DocumentTypes and **Razor views**
4. Add lots of controllers and other custom logic
5. Test, accept and release

You might notice...

Development has shifted from Frontend to Backend

(Which our frontender doesn't like, by the way)

You might notice…

- Content *really* defines the way you work
  - **DocumentTypes** Composition over Inheritance
  - **Content** Maintainability and Reusability over Usability

- Lots of custom logic
  - Thanks to Umbraco gradually opening up more with every release

Umbraco workflow pitfalls

- Mixing old and new habits
- Reinventing the wheel
- Over-engineering solutions
- Ignorance

Mixing old and new habits

- **What?**
  - Using macros for everything
- **Why is that bad?**
  - Macros add a lot of unnecessary overhead
  - Razor introduced Partial Views
- **When to use?**
  - Only when content requires user-defined controls

Reinventing the wheel #1

- **What?**
  - Adding your own cache layer
- **Why is that bad?**
  - Umbraco comes with 3 types of caches out of the box
    - RuntimeCache
    - RequestCache
    - StaticCache
- **When to use?**
  - A SessionCache-variant is not part of the deal
    - Implement your own **ICacheProvider**
    - Be aware of load balancing though!

Reinventing the wheel #2

- **What?**
  - Adding your own ORM into the mix, eg. *EntityFramework*.
- **Why is that bad?**
  - Umbraco comes with *PetaPoco*, which is well suit for most purposes.
  - Umbraco also has support for Migrations
    - Checkout **MigrationBase**
- **When to use?**
  - Only when you really want that one feature…
  - ORM talks to a different database than Umbraco.
  - But… It still sucks, ORM references and **Bin**

Over-engineering solutions

- **What?**
  - Trying to abstract Umbraco too much
- **Why is that bad?**
  - One size fits all is a myth
  - Embrace what Umbraco has to offer
  - KISS
- **When to use?**
  - Only to support development up to a certain level
    - Inversion of Control, eg. Dependency Injection to increase Testability

Ignorance #1

- **What?**
  - Searching through the content-tree every request
- **Why is that bad?**
  - The Umbraco XML cache is *really* slow and eats memory
    - It works fine for a few nodes, but not if you have 1k+
- **What to use?**
  - Use references properties in the root-node(s)
    - Cache these references
    - Load a reference with **Umbraco.TypedContent**
      - Do **not** use **Umbraco.Content**
  - Use the Examine index inside Umbraco…
  - … or write your own (Lucene) indexer

Ignorance #2

- **What?**
  - Keeping track of metadata in content-nodes
- **Why is that bad?**
  - Everytime a save is triggered, a new version is stored
  - Umbraco cannot handle lots of versions well
    - This is due to the diffing algorithm that runs for every publish
- **When to use?**
  - Never, use a **PetaPoco** instead

Ignorance #3

- **What?**
  - Using custom HttpModules to do UrlRewriting to nodes
- **Why is that bad?**
  - Rewriting has to occur before **UmbracoModule** kicks in…
    - But that's problemetic when you need **UmbracoContext**
  - Umbraco offers UrlRewriting out of the box*
  - You can implement your own **IContentFinder**
  - Do both to complement eachother
- **When to use?**
  - When you don't need **UmbracoContext** and logic surpassing that of simple rewrite rules

* This will be removed in Umbraco 7.6, so instead you can use the IIS Rewrite Module. OWIN would still require your own module.

So what can we learn from this?

- Don't be afraid to learn new tricks
  - They can be lifesavers
- Embrace what Umbraco offers out of the box
  - Prevent reinventing the wheel
  - Unlikely to break any time soo
- Refrain from using "hacks" based on assumptions
  - "**Assumption is the mother of all fuckups.**"

What did we do with this knowledge?

- Separate content and data
  - Define context-free
  - DocumentTypes for data
  - Create them under a Content-tree without bindings

What did we do with this knowledge?

- Relational Data
  - Create bidirectional relations between data-objects
    - A doctor does a particular treatment
    - A particular treatment knows which doctors perform it
  - Content-editors don't have to create the same relation both ways
  - Data can be presented any way you like

What did we do with this knowledge?

- Custom Lucene Indexers
  - Small specific indexes over uber-indexes
  - Speed up specific slow portions of your site, eg. Blog-posts
  - ArlaSearch