

# Not your grandmother's headless

BY MIKKEL KELLER STUBKJÆR

# \$ whoami

Long-time passionate Umbraco developer and seasoned software architect.

- Head of Development at Novicell UK
- 10+ years of experience with Umbraco (2007)



Mikkel Keller Stubkjær  
Head of Development  
Novicell UK

[mks@novicell.co.uk](mailto:mks@novicell.co.uk)  
+44 (0) 7756 634 808

[https://www.linkedin.com/  
in/mikkelkeller/](https://www.linkedin.com/in/mikkelkeller/)



# About 3 years ago

we embarked on an exciting adventure



# Decomposing the monolith

of large scale e-commerce





# Approach

using microservice architecture, headless api's  
and assembling best of breed platforms and services



# We wanted freedom of choice

CMS system, PIM system, commerce platform,  
recommendation engine, personalisation engine etc.

# Choosing best of breed for our client

Contextual to the market, the customers, the competitors as well as the internal capabilities of their organisation

# We are CMS Platform Agnostic



# Best of breed CMS



# Part of that journey

To apply headless at scale



# Strategic requirements

- Performance and scalability as a first principle
  - Minimising request-time workload
  - Caching as a last resort
- Freedom to choose different .NET versions
- Freedom to choose the frontend tech stack
- Reusability, flexibility and extendability
- Headless APIs

# Unfortunately

No mature headless offering was available at the time





# Without headless

- No ability to isolate the workload of the website from the workload of the CMS
  - No control of resource consumption within the website runtime (eg. database requests from within Umbraco etc.)
  - The performance and scalability of the website is affected by Umbraco
- Dependencies on the .NET version of Umbraco
- Dependencies on the frontend framework of Umbraco (ASP.NET mvc)

# Luckily

We are builders



# Actually

Even with a headless api, there will still be a direct performance dependency on the CMS

# Why #1

Direct performance dependency on the workload



# Request-time transformations

When a **controller** **generates** a **view model** by transforming node data. Traversing the content tree, aggregating data from multiple nodes, doing dictionary look-ups etc.

# What if

we could prepare data in the desired form  
and update it whenever changes occur

# Publish-time transformations

When a **publish action** **generates** a **view model** by transforming node data. Traversing the content tree, aggregating data from multiple nodes, doing dictionary look-ups etc.

# Publish-time transformations

- Triggered on publish action in the Umbraco backoffice
  - Listening to Umbraco events
- Generating view models
  - Tailored for a specific page or purpose
- Storing the generated view model in fast cache-level storage
  - Without the need for TTLs and cache invalidation (hardest thing ever)
  - Whenever content changes, a new publish event triggers the existing model to be overridden

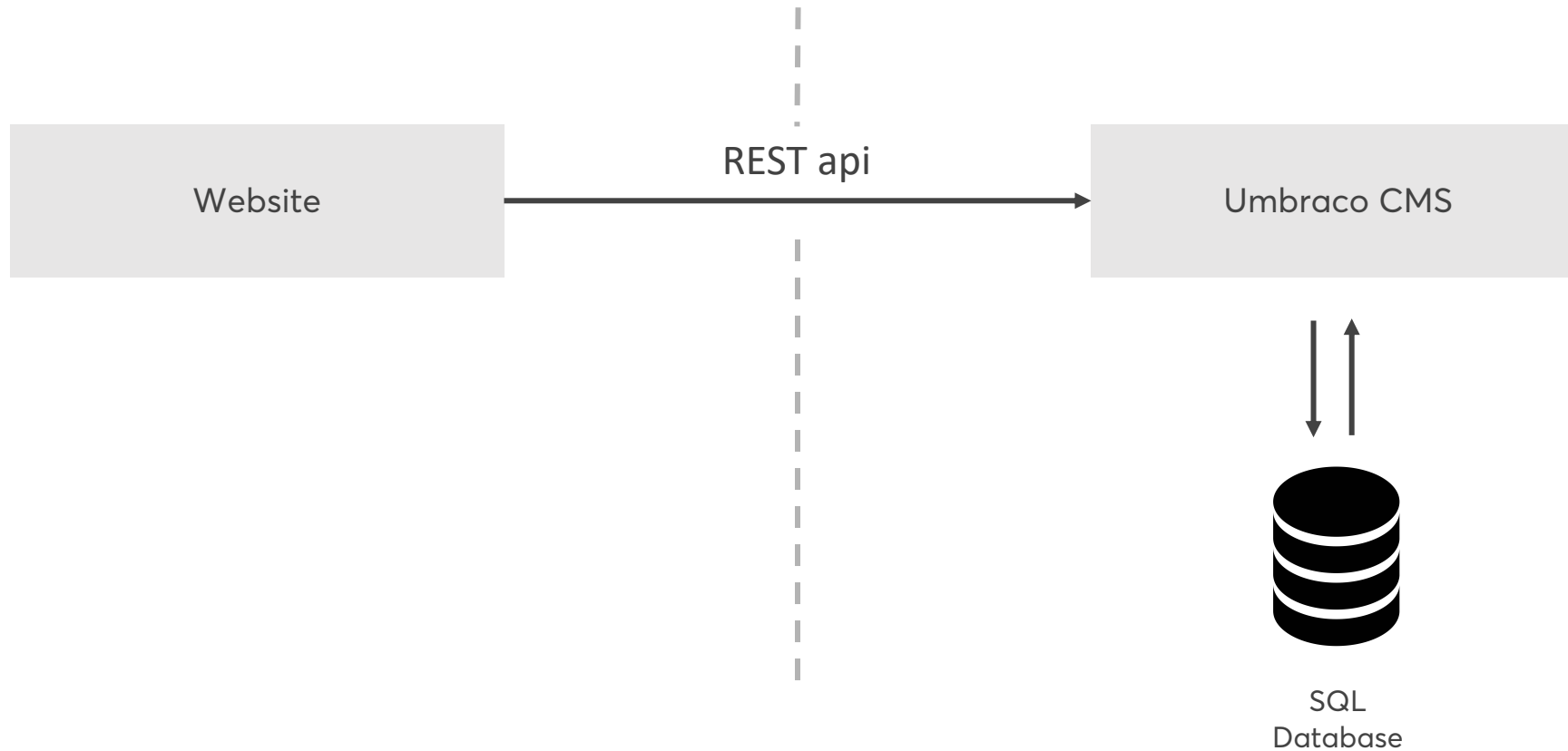


# Why #2

Direct performance dependency on the CMS

# Direct Headless

Separation of website and CMS  
But direct dependency on the CMS

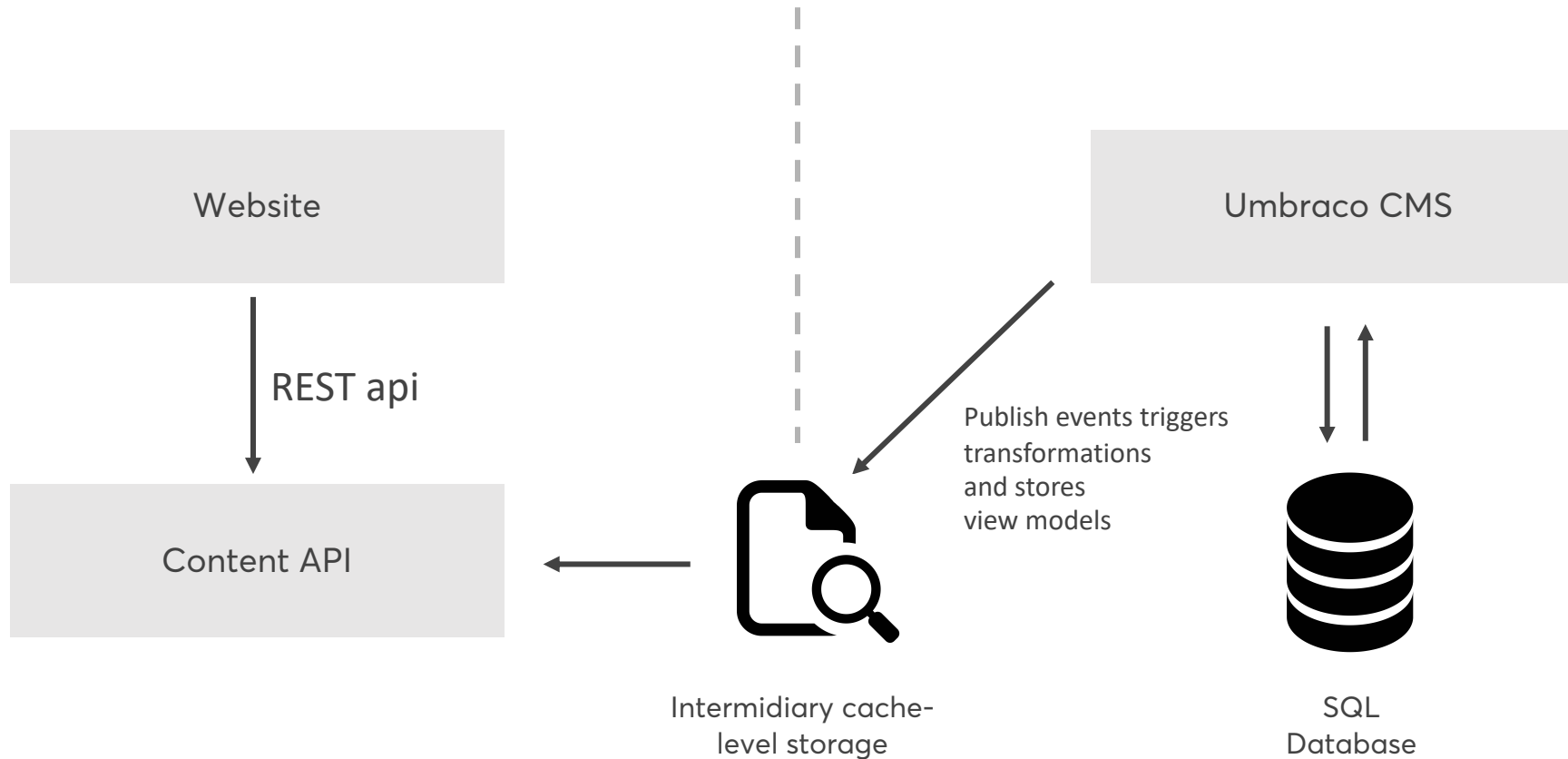


# Indirect Headless

Serving up the view models directly from intermediary storage

# Indirect Headless

Complete separation of website and CMS  
No direct performance dependency on the CMS



# Effectively

Minimising workload at request-time by displaces the workload  
from request-time to publish-time

# Effectively

Moving the workload from our website visitor to the editor



# Effectively

Removing the direct performance dependency on the CMS



# Remember

We wanted freedom of choice





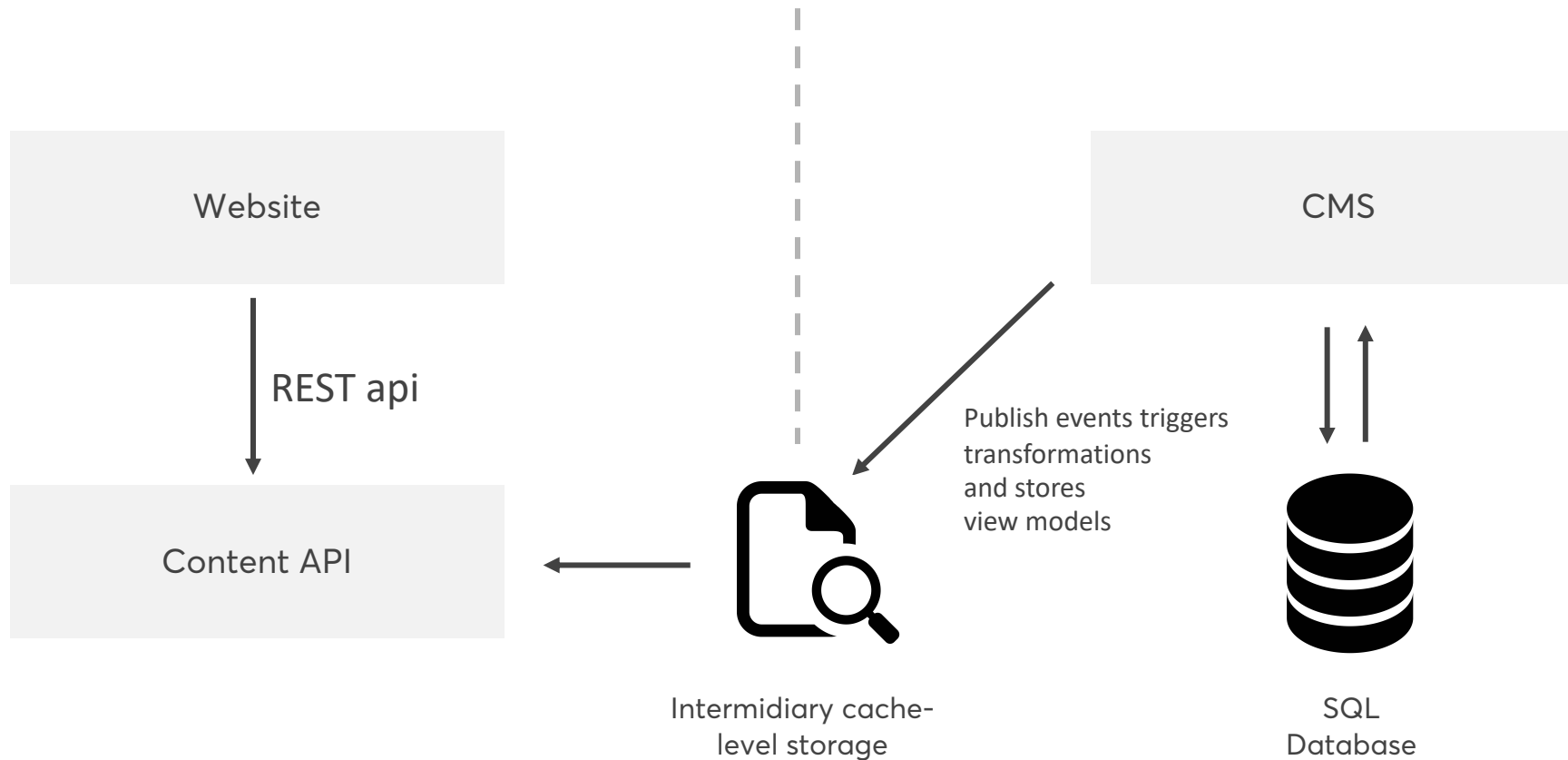
# Building

A CMS agnostic  
headless transformation engine



## API & Website runtime

## CMS runtime



## API & Website runtime

Website

Content API

## CMS runtime

CMS

Intermediary cache-level storage



## API & Website runtime

Website

Content API

## CMS runtime

CMS

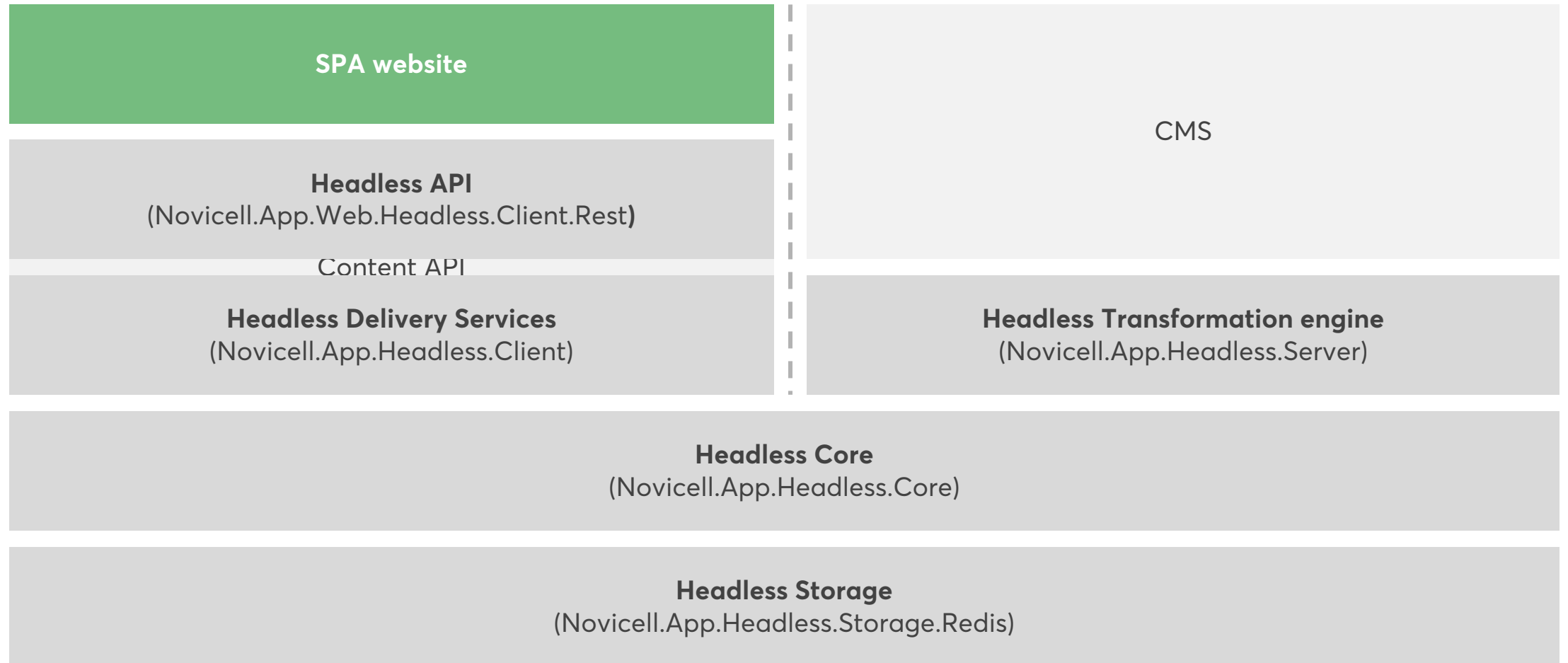
**Headless Transformation engine**  
(Novicell.App.Headless.Server)

**Headless Core**  
(Novicell.App.Headless.Core)

**Headless Storage**  
(Novicell.App.Headless.Storage.Redis)

## API & Website runtime

## CMS runtime



## API & Website runtime

## CMS runtime

**SPA website**

**Umbraco CMS**

**Headless API**

(Novicell.App.Web.Headless.Client.Rest)

**Headless binding to Umbraco**

(Novicell.App.Umbraco.v8.Headless.Server)

**Headless Delivery Services**

(Novicell.App.Headless.Client)

**Headless Transformation engine**

(Novicell.App.Headless.Server)

**Headless Core**

(Novicell.App.Headless.Core)

**Headless Storage**

(Novicell.App.Headless.Storage.Redis)

# Time for some code

# Runtime configuration

## API runtime

```
public class Startup : IStartup
{
    public void Configuration(IApplicationBuilder appBuilder)
    {
        appBuilder.UseNovicellWebApp( configure: webApp =>
        {
            webApp.UseHeadlessClient( configure: cfg :HeadlessClientConfig =>
            {
                // Enable headless REST API
                cfg.EnableRestApi();

                // Apply unified headless configuration
                cfg.ApplyConfiguration( headlessConfigurer: new HeadlessConfiguration());
            });
        });
    }
}
```

## CMS runtime

```
public class Startup : IStartup
{
    public void Configuration(IApplicationBuilder appBuilder)
    {
        appBuilder.UseNovicellDefaultUmbraco( configure: umbraco =>
        {
            umbraco.UseHeadlessServer( configure: cfg :HeadlessServerConfig =>
            {
                // Instantiate headless with Umbraco as the CMS
                cfg.UseUmbraco();

                // Apply unified headless configuration
                cfg.ApplyConfiguration( headlessConfigurer: new HeadlessConfiguration());
            });
        });
    }
}
```

Common headless configuration



# Common headless configuration

```
public class HeadlessConfiguration : IHeadlessConfigurer
{
    public void Configurer(HeadlessConfigurator config)
    {
        StorageConfiguration.Register(config);

        // Global elements
        ConfigureGlobalElements(config);

        // Pages
        ConfigurePages(config);

        // Compositions
        ConfigureCompositions(config);

        // Items
        ConfigureItems(config);
    }
}
```

# Configuring a transformation

```
config.ForContent( params doctype: DocumentTypes.FrontPage)
    .RegisterTransformation<FrontPage>(transform =>
    {
        transform.UseKey().RoutingKey();
    });
```

Doctype alias

Type of view model

Key for storing the view model

# The view model

```
public class FrontPage : ViewModel
{
    public string Headline { get; set; }
    public IGridViewModel GridContent { get; set; }
}
```

# The transformation

View model type

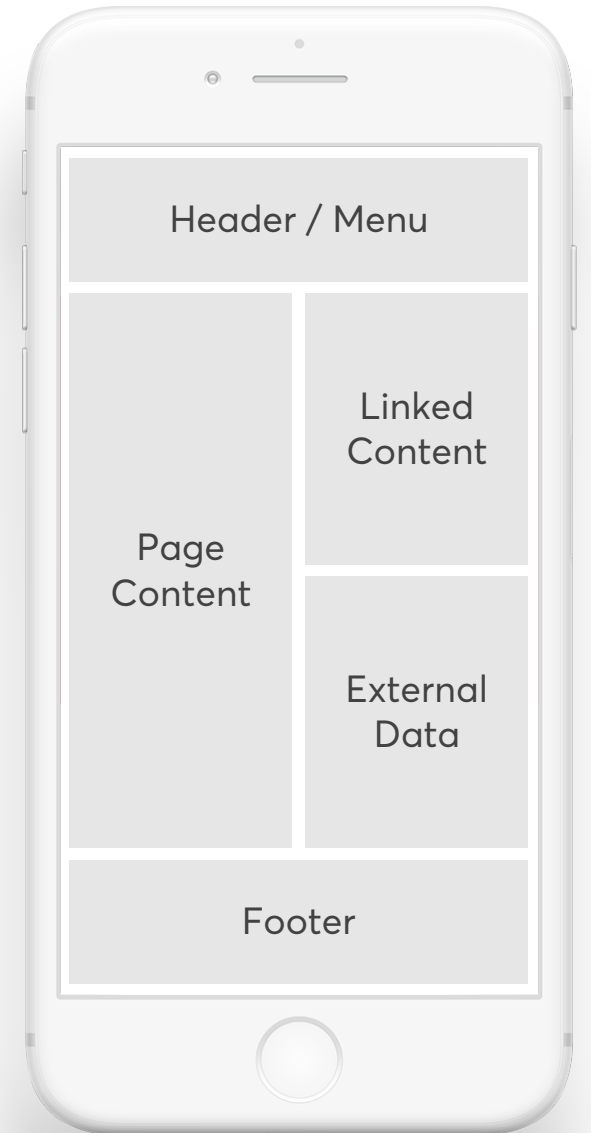
```
public class FrontPageTransformation : IContentTransformation<FrontPage>
{
    public FrontPage Transform(IContentModel content, ITransformationHelper helper, IUrlProvider urlProvider)
    {
        return new FrontPage
        {
            Headline = content.GetProperty<string>(alias: "headline"),
            GridContent = content.TransformPropertyValue<IGridViewModel>(alias: "grid", helper, urlProvider)
        };
    }
}
```

Transfer values to view model

# Re-visiting transformations


# Anatomy of a webpage

- Global elements
  - Header, Menu, Footer etc.
- Routable content
  - Articles, Pages etc.
- Linked Content (aggregated)
  - News, Faqs, employees etc.
- External Data
  - Product information, prices, stock count etc.



# Anatomy of a webpage

- Global elements
  - Header, Menu, Footer etc.
- Routable content
  - Articles, Pages etc.
- Linked Content (aggregated)
  - News, Faqs, employees etc.
- External Data
  - Product information, prices, stock count etc.

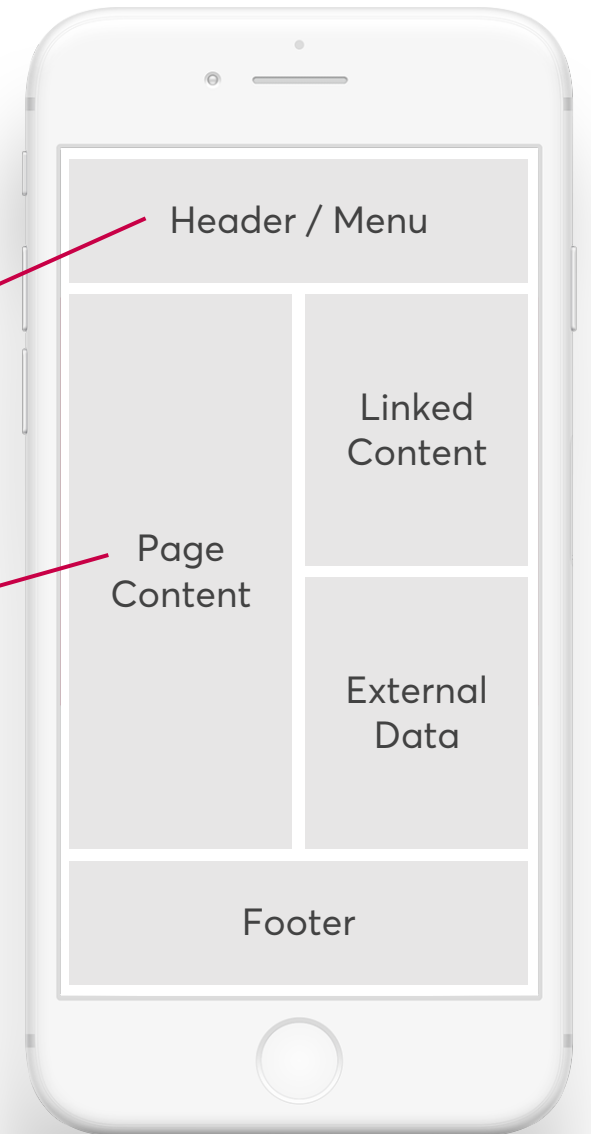
Translates to  


## View Models

# Multiple transformations

2 transformations are triggered when a FrontPage node is published:

- Menu view model
- FrontPage view model





# Configuring global transformation

Register transformation on multiple doctypes

```
config.ForContent(  
    params doctypes: DocumentTypes.FrontPage,  
    DocumentTypes.TextPage)  
    .RegisterTransformation<Navigation>(transform =>  
    {  
        transform.UseKey().DynamicContentKey(DynamicKeyConstants.NavigationKey);  
    });
```

Navigation model key

```
public class DynamicKeyConstants  
{  
    public static Func<IContentModel, string> NavigationKey => $"navigation:{x.GetSiteNodeId().ToString()}";  
}
```

# A single request for aggregated content

Linking global elements to routable elements

# Linking global view models

Going back to the FrontPage doctype

```
config.ForContent( params doctype: DocumentTypes.FrontPage)
    .RegisterTransformation<FrontPage>(transform =>
    {
        transform.UseKey().RoutingKey()
            .WithDynamicPartialKeys(DynamicKeyConstants.NavigationKey);
    });
```

Linking a global element key to the FrontPage view Model

# Intermediary storage

And internal data structures

# Underlying storage



- Key / value storage
  - Efficient for retrieval
- Json in NoSQL storage
  - Efficient for serialization and deserialization
- Utilising fast cache-level storage
  - Originally build using Redis for performance
  - Now also supporting CosmosDB for replicated datasets
  - And Examine, Lucene, SQL server for scaled down instances

# Routing and content keys

*Routing and Content is divided into two distinct key/value storages*

- Routing storage:
  - keys are urls (or paths) eg. "domain.co.uk/", "/" or "/en/"
  - Only routable content has a key/value pair in the routing storage
- Content storage:
  - keys are static eg. "menu" or dynamic eg: "\$navigation:{x.SiteId()}"
  - All view models have a key/value pair in the content storage

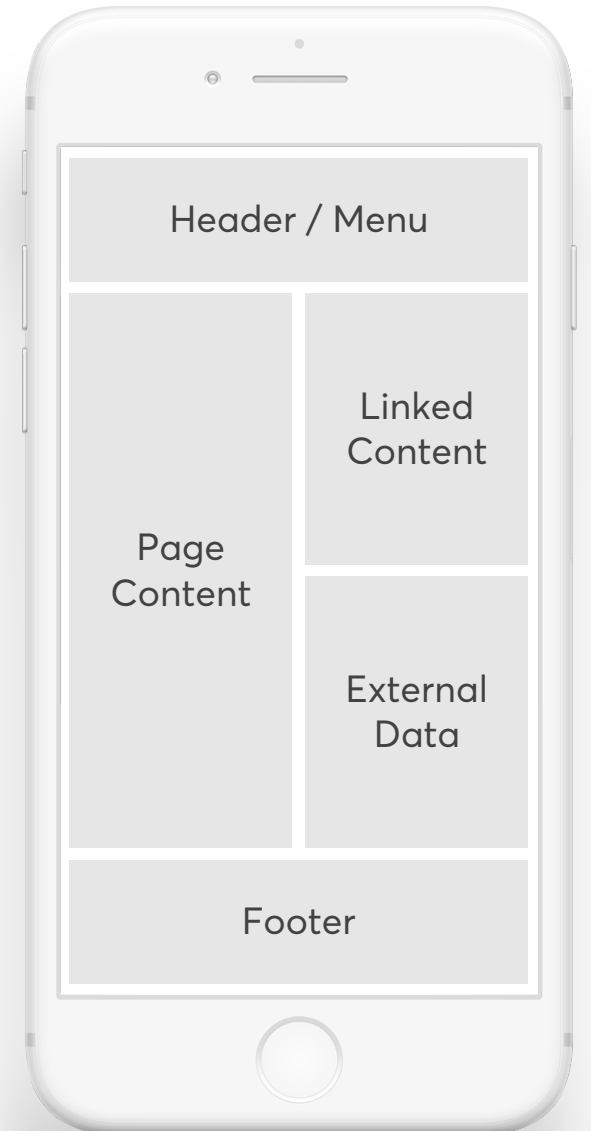
# Routing and content keys

## Routing storage

- Key: "/"

## Content storage

- Key: "1145"
- Key: "menu"
- Key: "footer"



# Routing

- Primary key
  - Key: "/"
- Routable content key
  - ContentKey: "1143" (eg. Frontpage)
- All content keys (including globals)
  - ContentKeys: ["1142", "menu", "footer"]

# Meta model

```
{
  "$type": "Novicell.App.Headless.Core.Models.Route,
Novicell.App.Headless.Core",
  "RoutingKey": "/",
  "Url": "http://novicell.dk/",
  "Controller": "frontpageController",
  "Restricted": false,
  "ContentKeys": {
    "$type": "System.String[], mscorlib",
    "$values": [
      "1143",
      "menu",
      "footer"
    ]
  },
  "LinkedContentKeys": {
    "$type": "System.String[], mscorlib",
    "$values": []
  },
  "PublishAt": null,
  "UnpublishAt": null,
  "Culture": "da",
  "RedirectKey": null,
  "ContentKey": "1143",
  "Path": "-1,1133,1143"
}
```





# Content

- Primary key
  - Key: "1143"
- Meta information
  - Culture,
  - Scheduled publishing
  - Etc.
- View Model

# Meta model

```
{
  "$type":
  "Novicell.App.Headless.Core.Models.ContentMetaModel`1[
  Novicell.App.Headless.Core.Models.IViewModel,
  Novicell.App.Headless.Core.Models]",
  "Key": "1143",
  "ViewModel": { ←
    "$type":
    "Web.v8.Mvc.Website.Core.Models.FrontpageModel,
    Web.v8.Mvc.Website.Core",
    "ActiveFrom": "0001-01-01T00:00:00",
    "ActiveTo": "9999-12-31T23:59:59.9999999",
    "Headline": "A typical headline",
    "body": "<b>body</b>"
  },
  "Culture": "da",
  "PublishAt": null,
  "UnpublishAt": null,
  "LastUpdatedAt": "2019-09-12T21:29:33.0904614+02:00",
  "ContainsTimedVariations": false,
  "LinkedContentKeys": {
    "$type": "System.String[], mscorlib",
    "$values": []
  }
}
```

# Headless backoffice extensions

# Headless Dashboard

Content

Media

Settings

Packages

Users

Members

Forms

Translation

English (United States)

Content

Premium - Law

Recycle Bin

Redirect URL Management

Headless

Headless actions

Rebuild the content, dictionaries or redirects in headless. Be aware that these actions should be used with caution and they can take long time to finish.

Published content

This button lets you rebuild all the published content in headless.

Rebuild published content

Preview content

This button lets you rebuild all the preview content in headless.

Rebuild preview content

Dictionary

This button lets you rebuild all dictionary items in headless

Rebuild dictionary

Redirects

This button lets you rebuild all redirects in headless

Rebuild redirects

Configurations

A list of configurations for each document type in Umbraco. The list shows what transformations are being run and what model they return when you publish or save a piece of content.

Headless Configurations

Site (site)

FrontPageTransformation

FrontPage

SeoSiteMetaCompositionTransformation

SeoSiteMetaDataComposition

SeoPageMetaCompositionTransformation

SeoPageMetaDataComposition

TrackingCompositionTransformation

TrackingComposition

SearchSettingsTransformation

SearchSettings

SearchTileTransformation

SearchTile

HeadlessSettingsTransformation

SettingsViewModel

NavigationTransformation

Navigation

MetaNavigationTransformation

MetaNavigation

BurgerMenuTransformation

BurgerMenu

FooterTransformation

Footer

SeoPageMetaCompositionTransformation

SeoPageMetaDataComposition

FeaturedTransformation

Featured

FooterTransformation

Footer

NavigationTransformation

Navigation

BurgerMenuTransformation

BurgerMenu

MetaNavigationTransformation

MetaNavigation

XmlSitemapTransformation

Sitemap

ChildListTransformation

ChildList

# Headless Content App

ContentMediaSettingsPackagesUsersMembersFormsTranslation

English (United States)

Our peopleEnglish (United States)

ContentHeadlessInfoActions

Content

Premium - Law

About us

Search Results

Our Expertise

Our people

Our locations

Get in touch

404 page

Recycle Bin

RoutesRedirectsReferenced contentConfigurations

Note: These actions do **not** update data in storage - this is only for previewing what the given transformation returns

ContentPageTransformationContentPage

Culture	Content key	Routing key	Domain	Actions
en-US	1080:en-us	premium.novicell.london/en/our-people/	https://premium.novicell.london/en	<div>Generate published content</div> <div>Generate preview content</div>

SeoPageMetaCompositionTransformationSeoPageMetaDataComposition

Culture	Content key	Domain
en-US	1080:en-us:pageosemeta	https://premium.novicell.london/en

SearchTileTransformationSearchTile

Culture	Content key	Domain
en-US	1080:en-us:searchtile	https://premium.novicell.london/en

NavigationTransformationNavigation

ContentMediaSettingsPackagesUsersMembersFormsTranslation

English (United States)

Our people

ContentHeadlessInfoActions

Content

Premium - Law

About us

Search Results

Our Expertise

Our people

Our locations

Get in touch

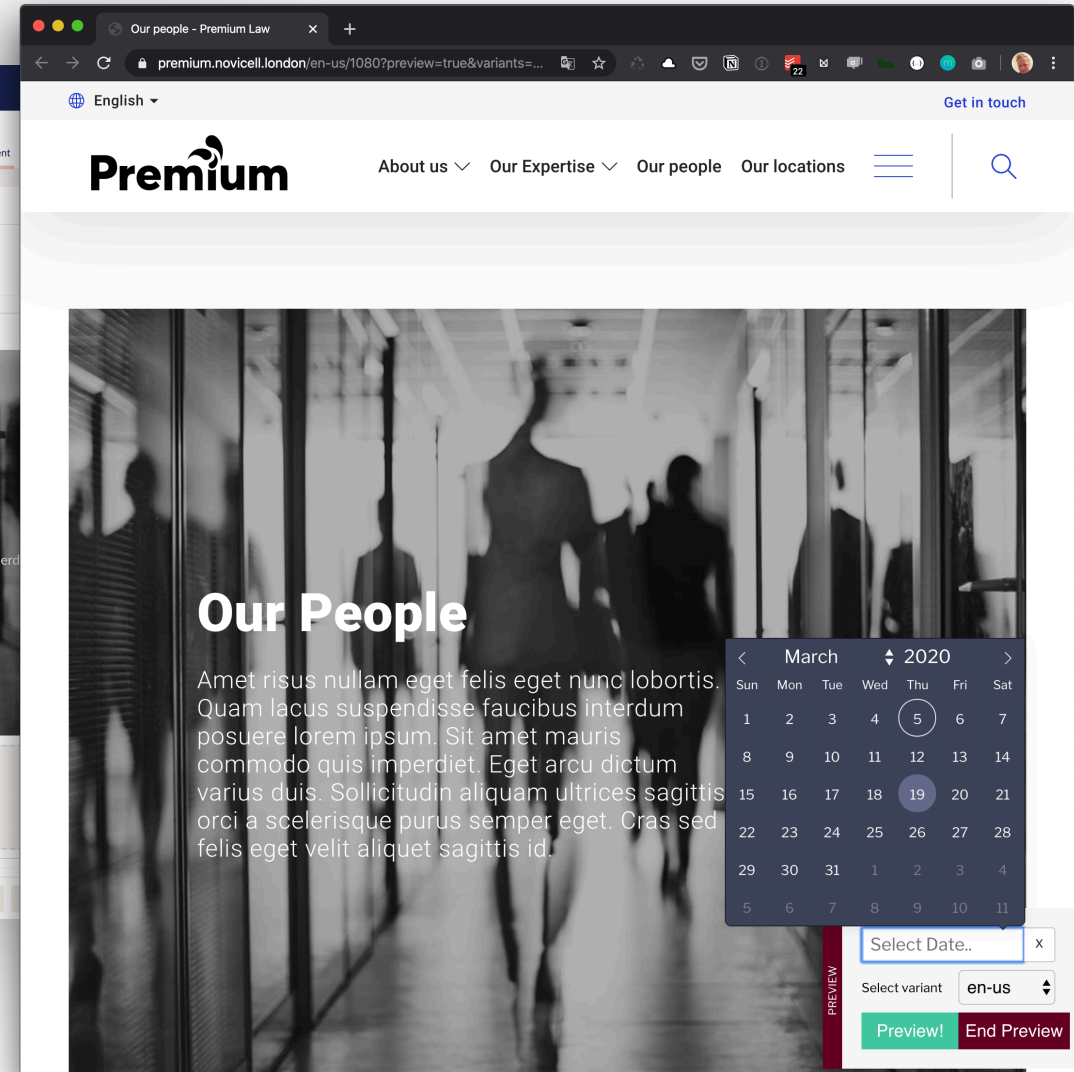
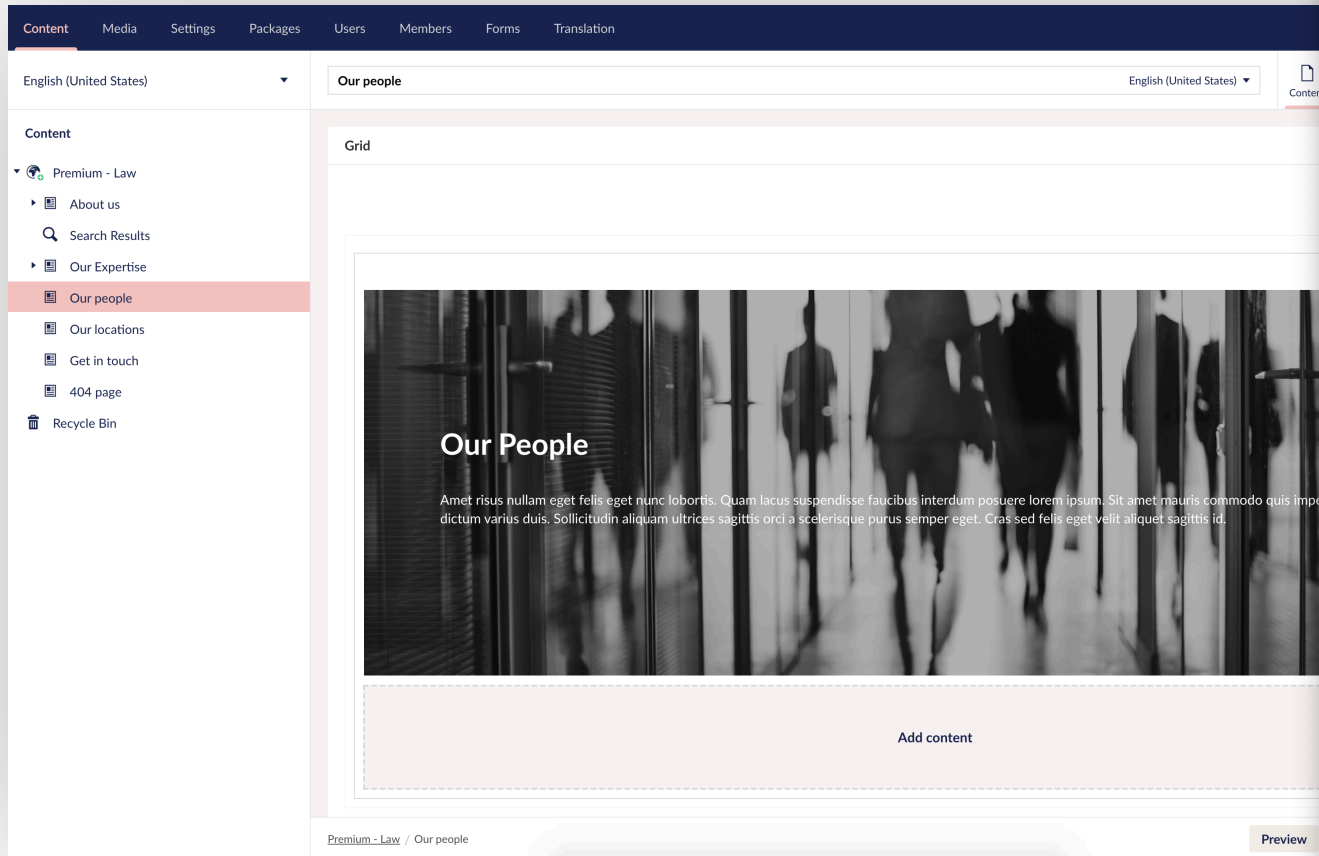
404 page

Recycle Bin

ContentPage (1080:en-us) (en-US) (Took: 5ms)

```
1 {
2   "content": {
3     "sections": [
4       {
5         "config": null,
6         "hasContainer": true,
7         "columns": [
8           {
9             "config": null,
10            "editors": [
11              {
12                "image": {
13                  "path": "/media/ayunopds/our-people.jpg",
14                  "altText": "Our People",
15                  "position": "left",
16                  "focalPoint": {
17                    "left": 0.5,
18                    "top": 0.5,
19                    "center": "0.5,0.5"
20                  },
21                  "url": "/media/ayunopds/our-people.jpg"
22                },
23                "title": "Our People",
24                "description": "Amet risus nullam eget felis eget nunc lobortis. Quam locus suspendisse faucibus interdum posuere lorem ipsum. Sit amet mauris commodo quis imperdiet.",
25                "position": "left",
26                "inverted": true,
27                "callToAction": null,
28                "video": {
29                  "url": null,
30                  "oEmbed": null
31                },
32                "alias": "novicell.gridditor.hero",
33                "columnSize": 14,
34                "render": "/Views/Partials/Grid/Editors/Hero.cshml",
35                "controllerName": null
36              }
37            ],
38            "hasColumns": false,
39            "sizeM4": 14
40          }
41        ]
42      }
43    ]
44  }
45 }
```

# Headless Preview



# Headless Preview and storage

- Extra set of storages
  - Published storage
    - Route storage
    - Content storage
  - Preview storage
    - Route storage
    - Content storage
- Preview storage is routed by id (like native preview in Umbraco) and rewritten in IIS
- Transformations are triggered on save event

# Other supported features

- Dictionary service
- Routing service
- Strong typed grid transformations
- Search based on search models and view model annotations
- Taxonomy filtering, also based on view model annotations
- Linked Content
- External Linked Data
- Content listeners and View Model listeners

# But what about Umbraco Heartcore?

Fantastic, use it, we didn't have the luxury





# However, we still need

- Full control of the Umbraco setup
- The ability to ship custom code with Umbraco
- Performance without the need for caching with publish-time transformations and fast cache-level intermediary storage
- Aggregating Umbraco content with external data
- Since we focus on websites and not Apps, IoT or chatbots, we can actually still allow the editor to preview the website

# THANK YOU!



Mikkel Keller Stubkjær  
Head of Development  
Novicell UK

[mks@novicell.co.uk](mailto:mks@novicell.co.uk)

+44 (0) 7756 634 808



[+44 \(0\)20 8144 8142](tel:+442081448142) \_\_\_\_\_ [hello@novicell.co.uk](mailto:hello@novicell.co.uk)