UMBRACO SPARK #SPARKQA

Umbraco on .NET Core Status

BY BJARKE BERG

umbraco

Agenda

- Motivation & Project goals
- ASP.NET Core
- What's Changes So Far
- Coding Challenges
- Breaking Changes
- Community Involvement
- Project Overview

umbraco

Motivation & Project Goals

Why are this project so important

Motivation for moving to .NET Core

- .NET Framework end-of-life
- Cross platform support
- Performance improvements
- New tech

The Project Goals

- Easy transition of customer projects
- Improved code quality
- Community involvement

umbraco

The Unicore Team

- Helps other community members to contribute
- Helps HQ identify tasks that can be up-for-grabs
- Pick up coding tasks with tight deadline
- Pick up more advanced tasks

umbraco

ASP.NET Core

Is not that different after all

Biggest changes compared to ASP.NET

- Setup
  - Basically a self hosted program.
  - Pipeline is very flexible
- HttpContext
- Unified MVC + Web API
  - Everything is MVC
- Configuration
- Native Dependency Injection Container

What is almost the same

- Controllers
  - Need to inherit from another base controller
- Views
  - New options
    - @inject dependencies
    - Tag Helpers
    - View components

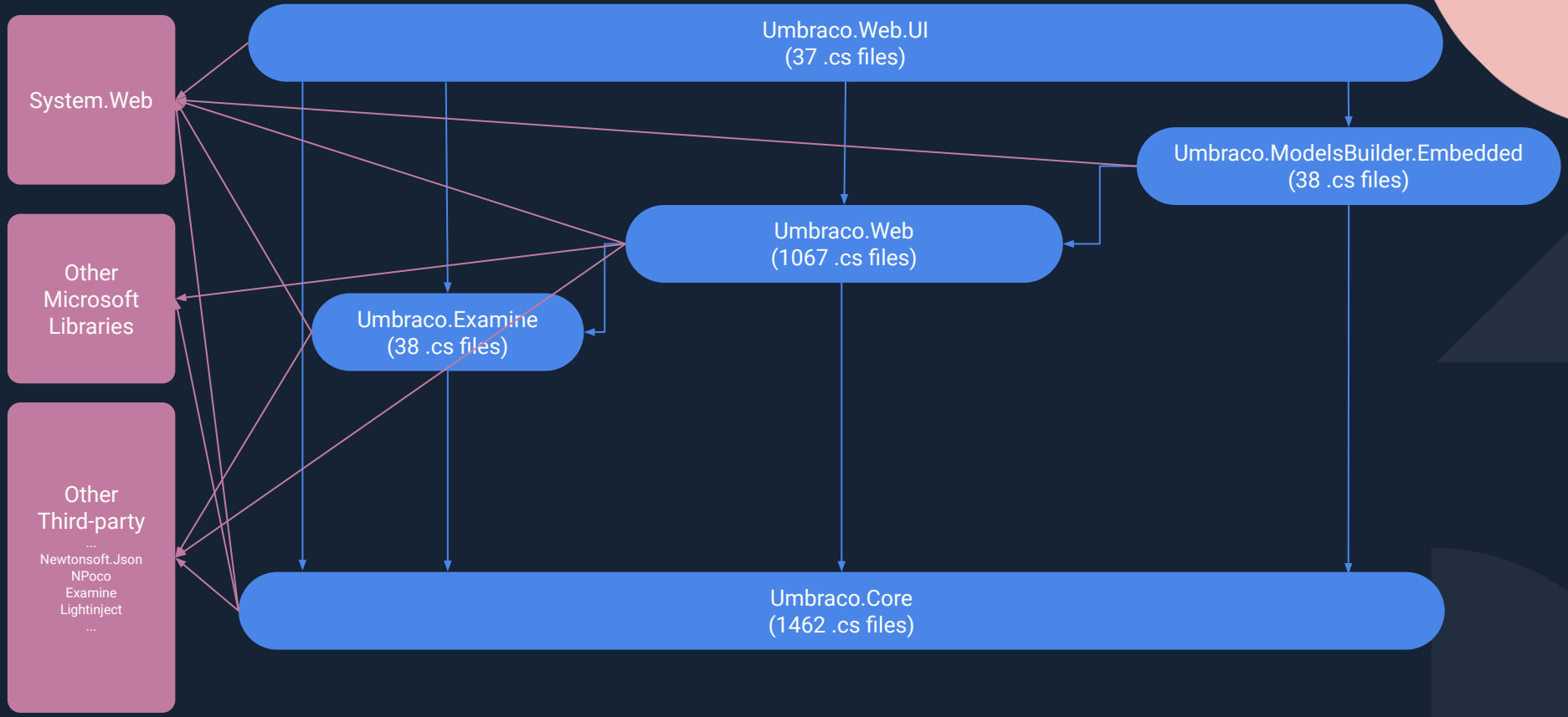
umbraco

What's Changed So Far

Evolvement of the architecture

umbraco

Umbraco 8



Introduced new Core Assembly

- Pure Umbraco business logic and abstractions
  - No third-party frameworks
  - Models
  - Services
- Half of the existing Umbraco.Core exists here
- General rule
  - If the there is no third party dependencies - The class belongs here

Introduced Infrastructure Assembly

- Third-party frameworks used
  - Repositories (NPoco)
  - Container (LightInject)
  - Logging (Serilog)
  - Search (Examine)
- The second half of Umbraco.Core exists here

Introduced SqlCe Assembly

- Cannot be compatible with .NET Standard
- Implements interfaces that exists in Umbraco.Core
    - Classes needs to be injected

Introduced Configuration Assembly

- Implements our configuration interfaces that exists in Umbraco.Core
- Uses System.Configuration for now
    - Plan to replace with ConfigurationBuilder pattern fra .NET Core

umbraco

Moved web dependencies from Core to Web

- Classes that use System.Web and logically belongs to Web is moved

Moved non-web classes from Web to Infrastructure or Core

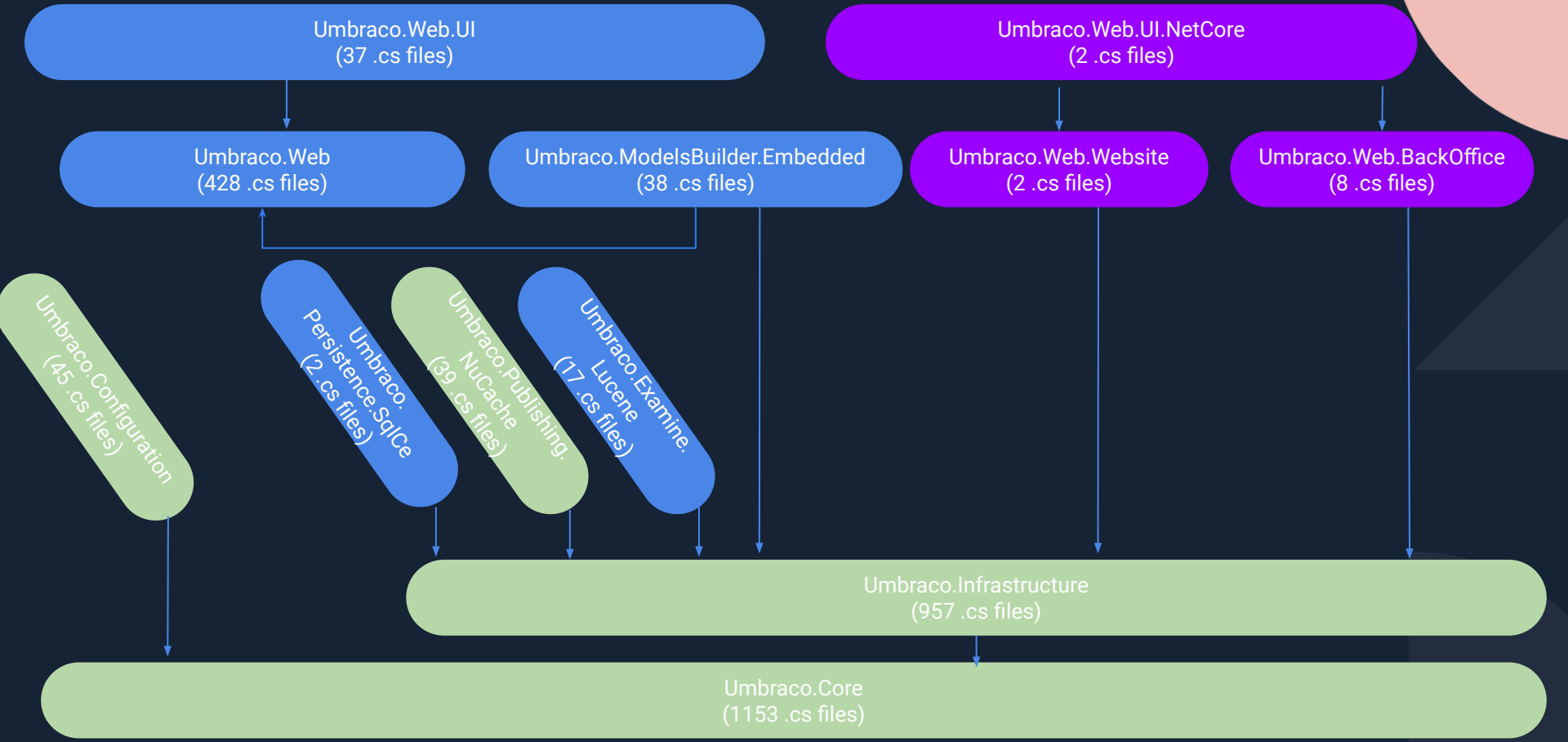
- Classes that did not logically belongs to Web is moved to
  - Core
    - If no third-party frameworks is used
  - Infrastructure
    - If third-party frameworks is used

Abstracted usage of HttpContext

- IRequestCache
  - Instead of HttpContext.Items
- ICookieManager
  - Instead of HttpContext.{Request/Response}.Cookies
- IUserAgentResolver
  - Instead of HttpContext.Request.UserHostAddress
- ...

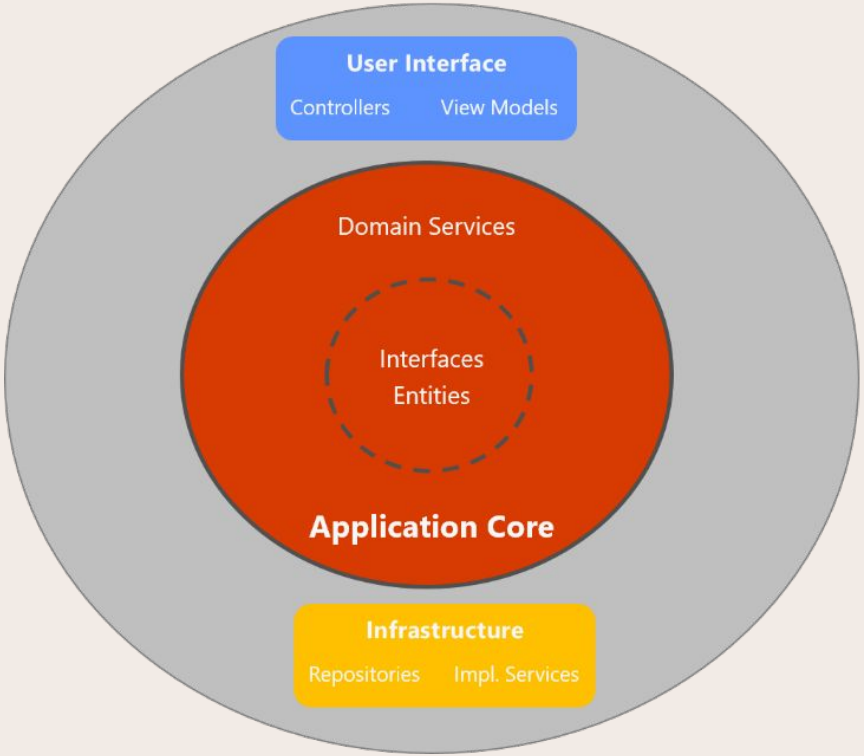
umbraco

Unicore Current Status



umbraco

Clean Architecture Layers (Onion View)



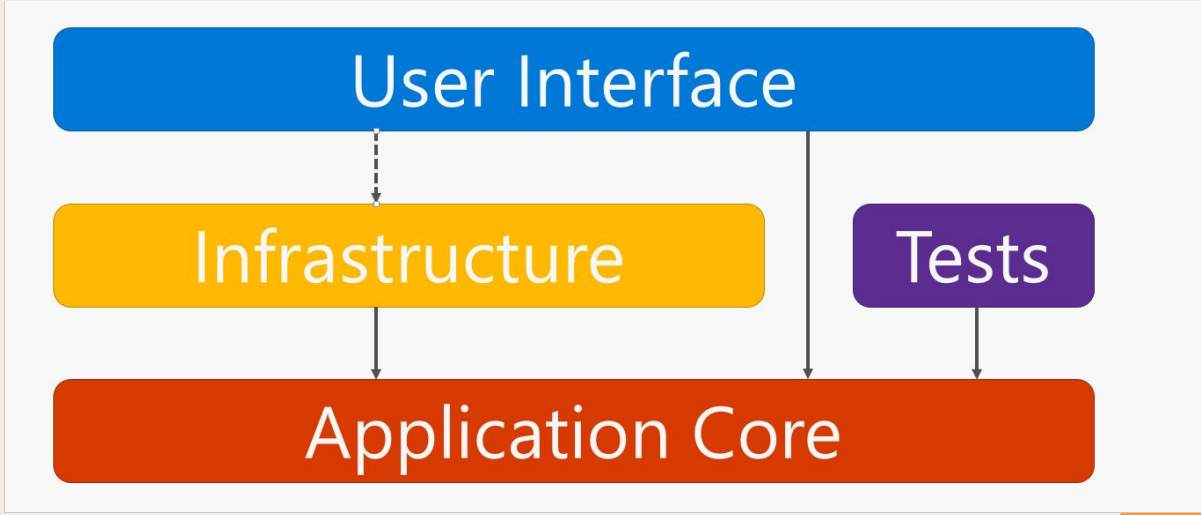
External Dependencies

umbraco

Clean Architecture Layers

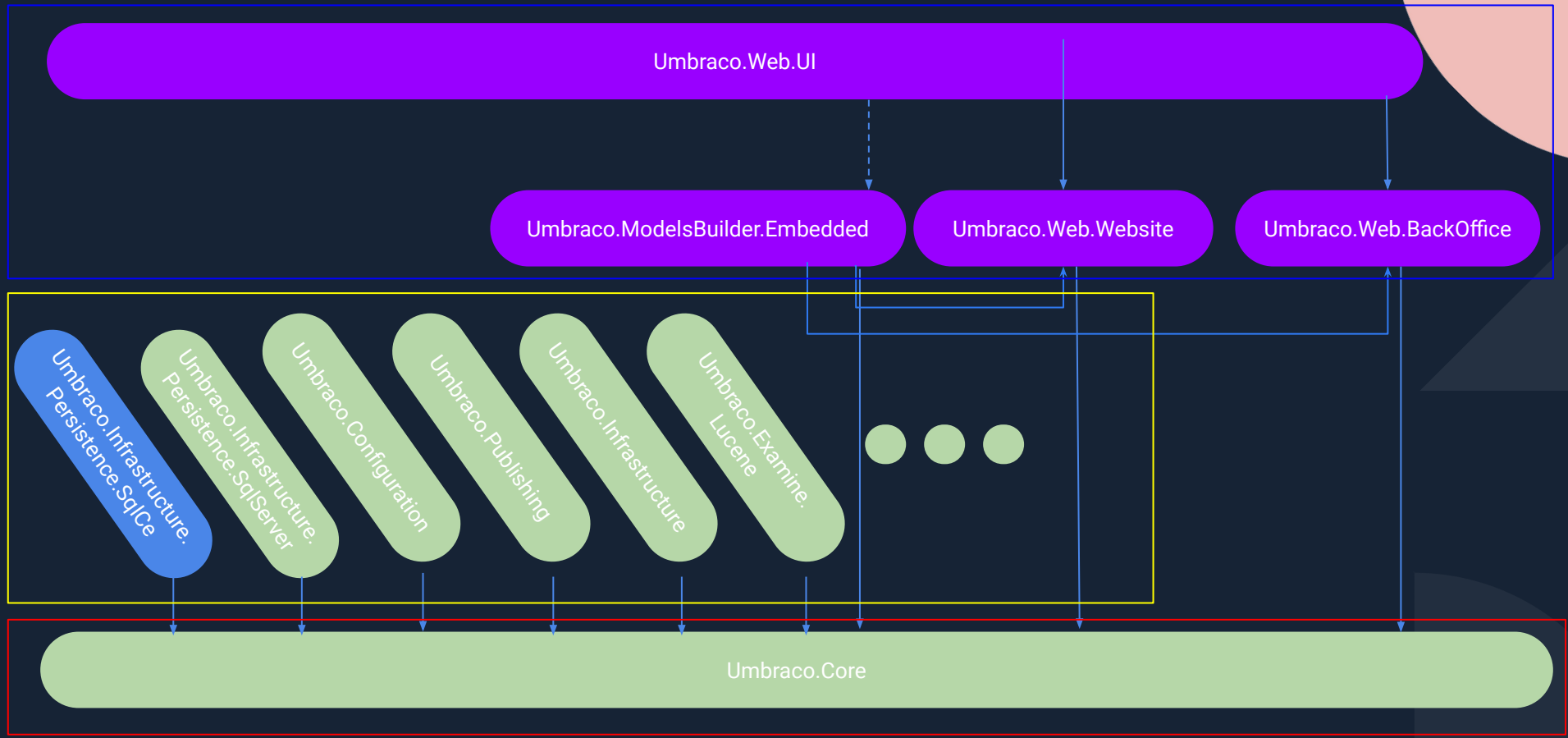
- Business logic independent of
  - Frameworks
  - Database
  - Externals
  - UI

- Testable business logic



umbraco

Architecture End Goal



umbraco

Coding Challenges

And how we solve them

Circular Dependencies

- ModelsBuilder bundled as package
  - Runtime crash, on breaking changes
  - ModelsBuilder is now embedded correct into the codebase
- Between classes
  - Break vs inject lazy

Static classes with third-party dependencies

- HostingEnvironment ⇒ IHostingEnvironment
- IOHelper ⇒ IIOHelper
- UmbracoVersion ⇒ IUmbracoVersion
- TypeFinder ⇒ ITypeFinder
- ...

Logical CallContext not available in .NET Standard

- CallContext was used to keep state in nested scopes as fallback to HttpContext
- Replaced using `ConcurrentDictionary<string, AsyncLocal>`

System.Web known everywhere

- The ASP.NET part of System.Web is completely rewritten in ASP.NET Core
- Static HttpContext should be accessed using IHttpContextAccessor
- Static HostingEnvironment not available
- HttpModule
    - Need to be rewritten as Middleware in ASP.NET Core

IRegisteredObject

- Do not exist in .NET Standard
- For now we have our own interface and and implementation for Framework
- In .NET Core IHostApplicationLifetime exists that can be used for the same, but does not require an interface.

Marshal

- Used to provide extra exception information to some of the logged errors
- Do not exist in .NET Standard
  - Exists in .NET Core 3.1 and .NET Framework 4.8
- Implement specific Framework implementation for now

Imaging

- We have introduced an interface to generate Urls
  - Multiple implementations
    - ImageProcessor - .NET Framework
    - *ImageSharp - .NET Standard*
    - Others?

NuCache dependency not .NET Standard

- BPlusTree
  - Replaced with fork of the project, that is updated

Tightly coupled to dependencies

- Often in `if-else` or `switch-case` statements
  - `BulkInsertRecords` now use a `IBulkSqlInsertProvider`

WebForms

- We still had some dependencies on WebForms
  - NoNodes.aspx
    - Rewritten to MVC
  - Macros
    - Need to clean up and abstract

UmbracoContext + UrlProvider

- Known everyware
  - Has dependency on HttpContext
  - Abstracted into `IUmbracoContext` and `IPublishedUrlProvider`

Models / DTOs using services or type collections

- Solved in different ways
  - Refactored to not use the service
  - Introduced factory
  - Deserialisation handled by custom JsonConverters
  - Moved properties to extension methods
    - Inject dependencies

umbraco

Breaking Changes

And what we do not change

Constructors

- Constructors are updated to inject their dependencies
  - If you use the DI container you shouldn’t care

New Abstractions and factories

- ~~new MenuItemCollection()~~ ⇒ IMenuItemCollectionFactory.Create()
- ~~UmbracoContext~~ ⇒ IUmbracoContext
- ~~IOHelper~~ ⇒ IIOHelper
- ~~HostingEnvironment~~ ⇒ IHostingEnvironment
- …
- Get these using the DI Container

Configuration

- Don’t use ConfigurationManager directly
- Inject the config interface
- Change to ConfigurationBuilder leads to transformation changes.
- JSON as default

What do we not expect to break

- The database schema
  - The Umbraco 8 database can be used
- The AngularJS code
  - Users/Members will have small changes
- The backoffice API
- Views
  - The old structure of views can be reused.
  - You will most likely need to inject dependencies

umbraco

Community Involvement

Outcome of team visit in January

- Imaging
  - 4 up-for-grabs tasks
- Install process / NuGet
  - RFC
- Publishing / NuCache
  - PR solving most of issue
- Members & Users
  - Rock Solid Knowledge take charge of this sub-project

How to contribute

- Currently
  - Code: Look for label `project/net-core` on GitHub
  - Look for RFCs - you comments are valuable
- Future
  - Look for documentation that is changed, and send PR with updates
  - Try out the alphas/betas/rcs

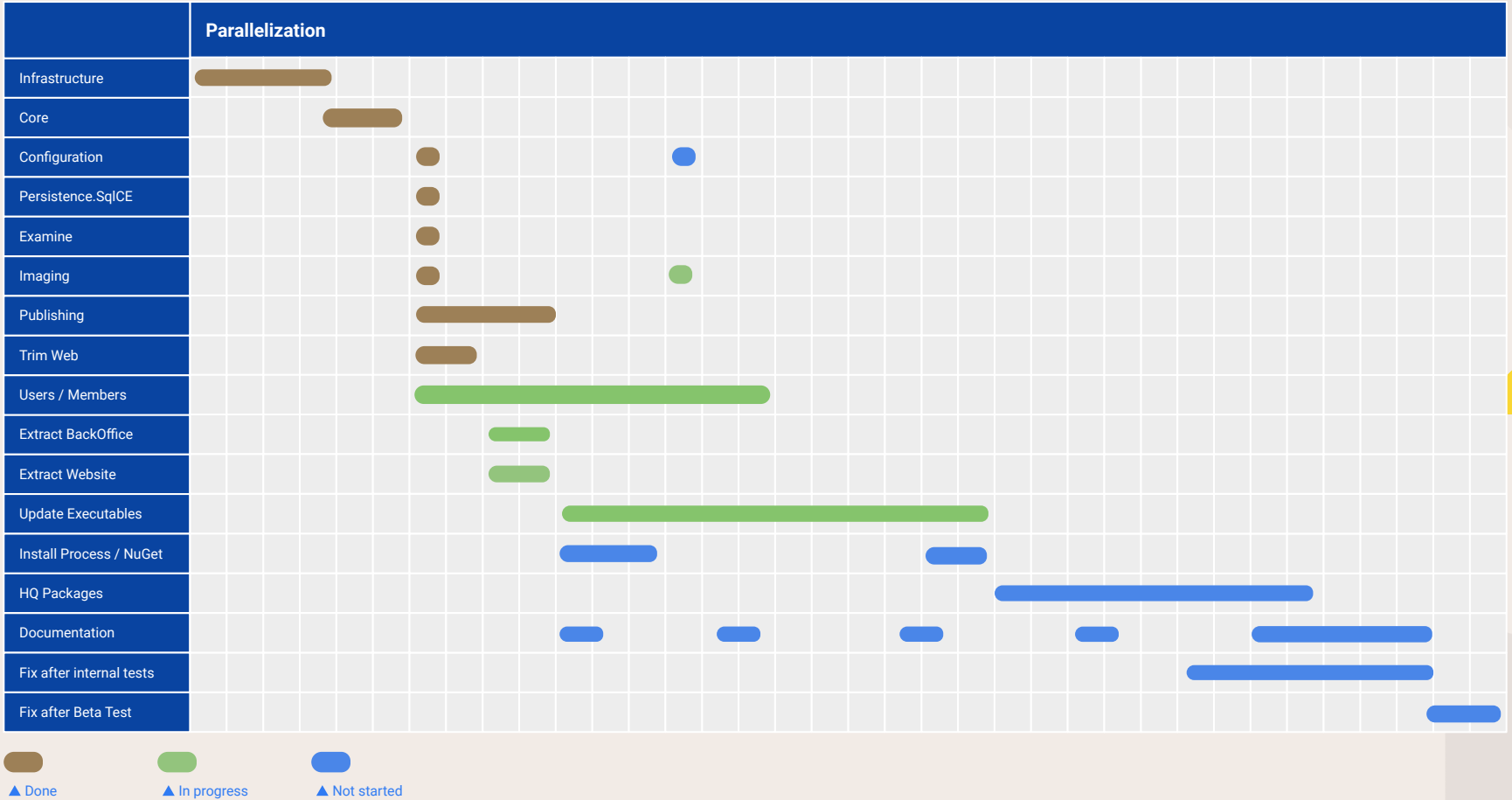
Tasks already closed by community

- Inject tasks
  - A lot of different tasks that replaced usages of Current.* with injection
- Configuration
  - New configurations types
- Imaging
  - Abstracting url generation

umbraco

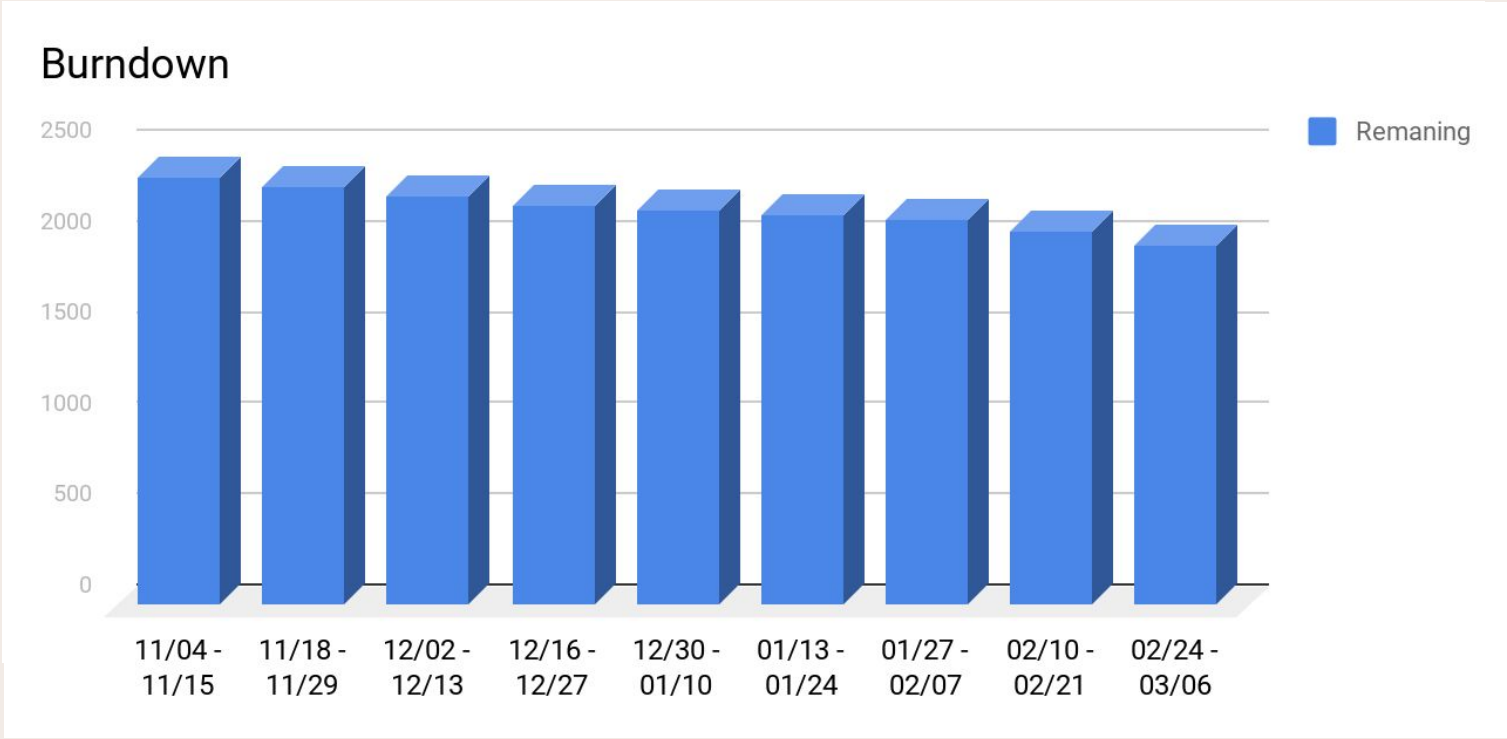
Project Overview

umbraco


Parallelization
Infrastructure
Core
Configuration
Persistence.SqlCE
Examine
Imaging
Publishing
Trim Web
Users / Members
Extract BackOffice
Extract Website
Update Executables
Install Process / NuGet
HQ Packages
Documentation
Fix after internal tests
Fix after Beta Test
▲ Done
▲ In progress
▲ Not started

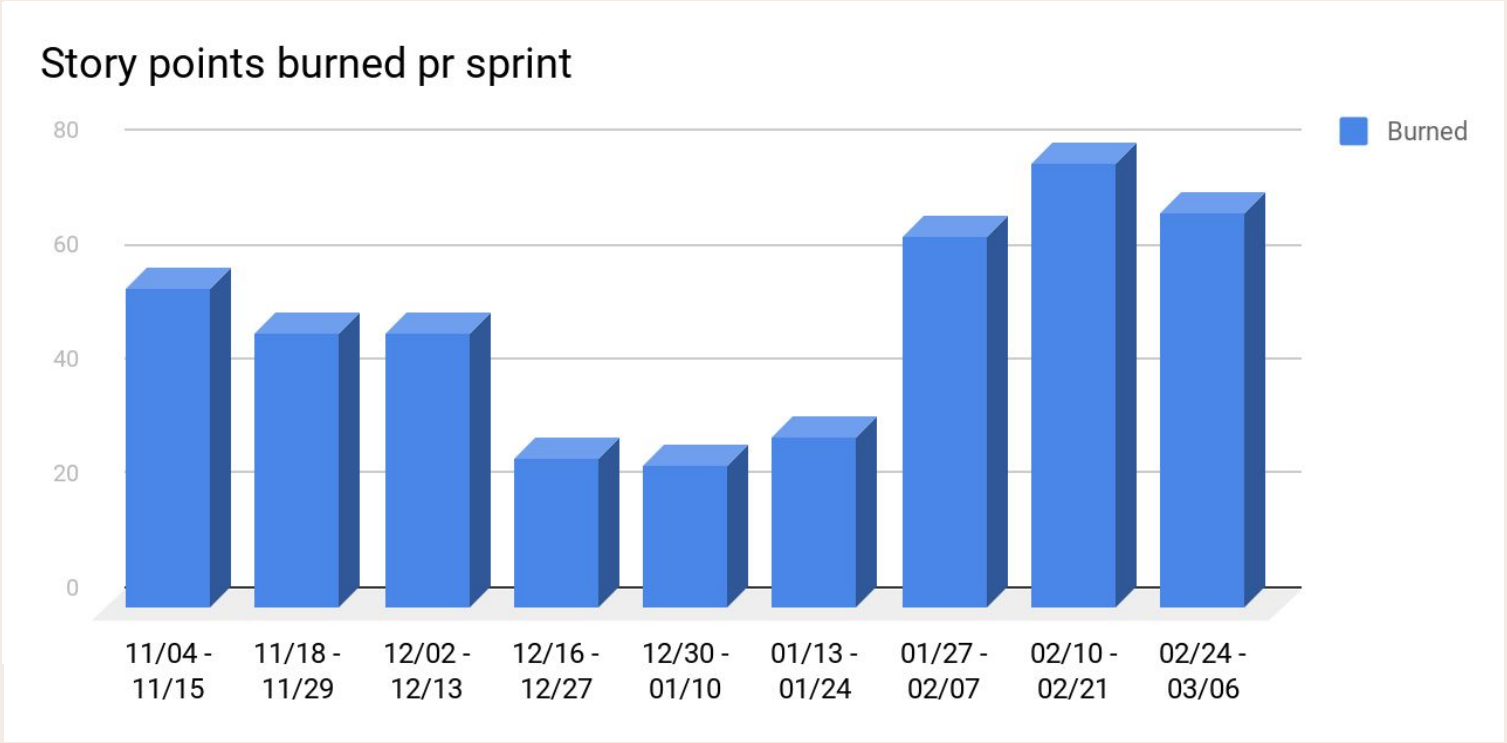
umbraco

Burndown - Start Estimate 2354



umbraco

Velocity



Triangular Estimation

- All estimates are made using Triangular Estimation
  - *B* - Best case
  - *M* - Most likely
  - *W* - Worst case
  - Estimate = *(B+4M+W) / 6*

- By the end of the day, all estimates are just guesses

umbraco

Largest estimates

- Documentation
  - 240
- Users / Members
  - 220
- Executable - Website
  - 220
- Buffer for fix of issues found in Beta and RC
  - 220
- HQ Packages
  - 216
- Buffer merge/reimplement v8 features
  - 160
- NuGet / Install process
  - 120
- Publishing
  - 120
- Executable - BackOffice
  - 100

Current activities

- Macros cleaned up and abstracted
- Starting the Backoffice executable
- NoNodes.aspx reimplemented in MVC

Next activities

- Executable Backoffice in .NET Core
- Imaging handled by ImageSharp
- Abstracting runtime minification and bundling

umbraco

Definitions

- Milestone 1 (T-1300)
  - We have something running on .NET Core (3.1 LTS)
  - Not feature complete
  - Missing either website or backoffice functionality
  - Expect lots of undocumented breaking changes
  - Missing members
  - Missing NuGet/dotnet new template
- Milestone 2 (T-550)
  - We expect the CMS to be complete
  - Expect few documented breaking changes
  - Package development is expected to begin
- Milestone 3 (T-250)
  - HQ packages available
    - Starter kits
    - Forms
    - Deploy

umbraco

Availability - Best Guess (Disclaimer)

- Milestone 1
  - H2 2020
- Milestone 2
  - H2 2020
- Milestone 3
  - H1 2021

Thank You