Helix in practice
&
Helix.Skeleton demo

User name:

tamas.tarnok

Password:

*

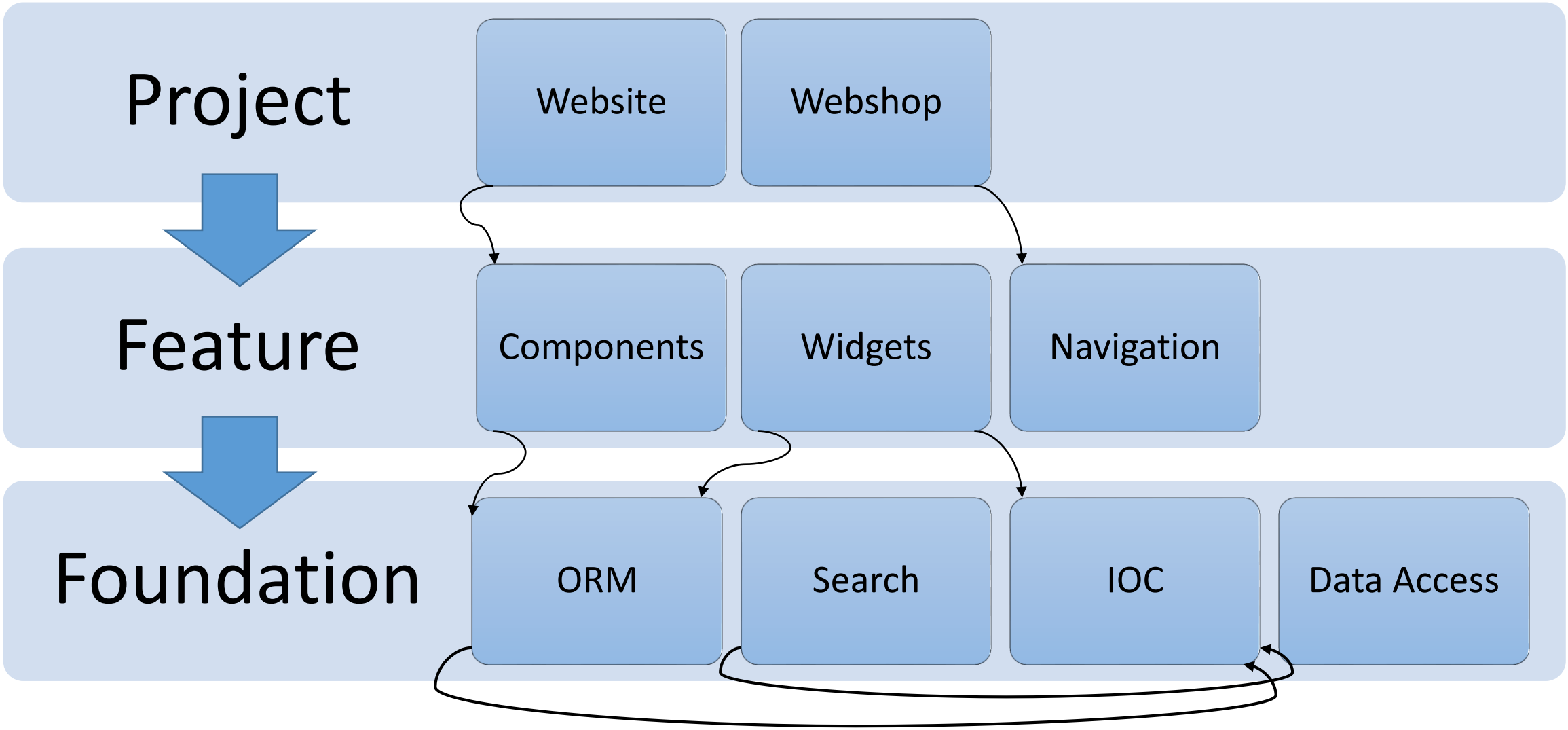
Log in

Remember me | Forgot your password?

What is Helix?

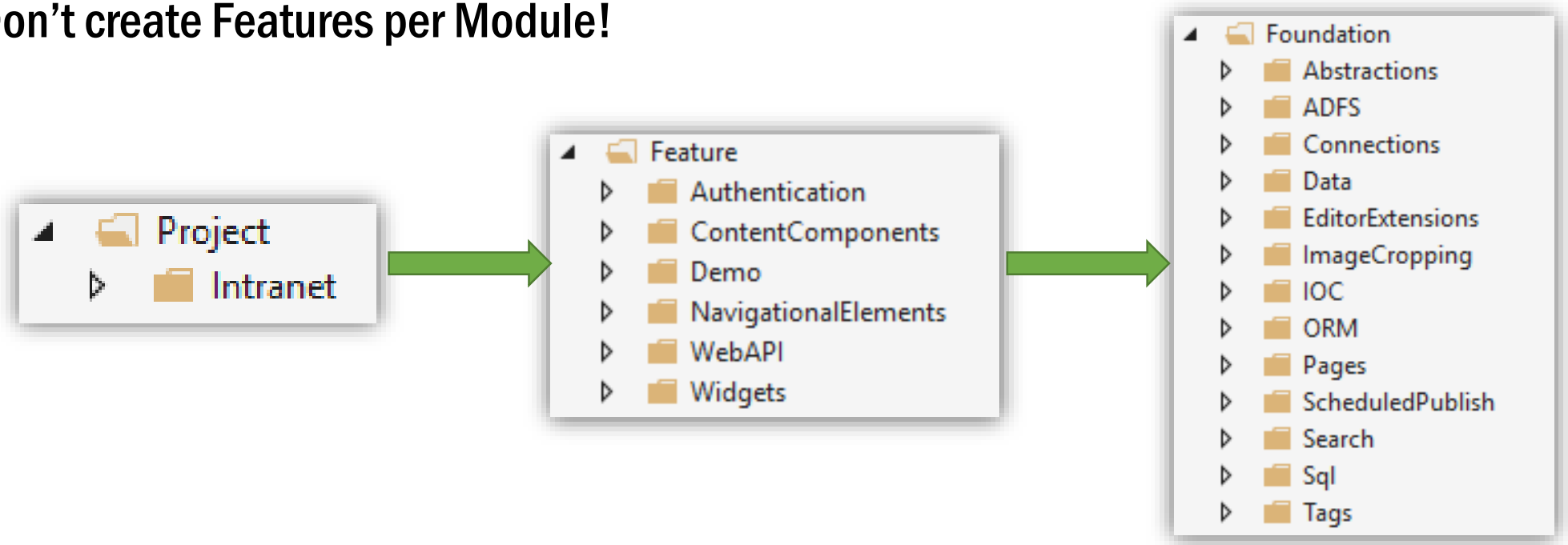
- „Overall design principles applied to a Sitecore project”
- „Helix is a set of recommended principles and conventions from Sitecore itself, it is not a set of rules.”
- Inspired by Modular Architecture and Component-based architecture
- The reason why should use it:
  - Decoupled code
  - Strict and easily understandable dependency flow
- Habitat is an example provided by Sitecore but don’t use this as a starter kit! Get your ideas from it instead ☺

Dependency flow



How to categorize your projects?

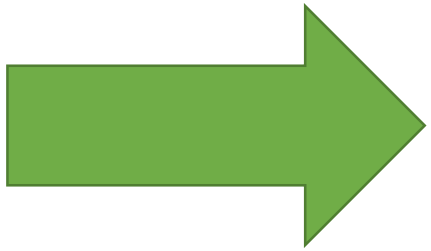
- Categorize them by its functionality
- Try to avoid categorizing by used technology like ~~Foundation.GlassMapper~~
- Inside a Feature you can have several Modules
  - Don't create Features per Module!



How GlassMapper supports Helix

- Use interfaces instead of classes to support „multiple” inheritance like in Sitecore templates
- Split your page templates to smaller base templates, avoid huge templates

- Teaser Data
  - Headline
  - Description
- Page Title
  - Title
  - Subheadline
- Tags



- Page list module
- New interface which is inherited from Teaser Data, Page Title and Tags

How GlassMapper supports Helix

```
public interface IPageList : IGlassBase
{
    IEnumerable<IPageListItem> Items { get; set; }
}

public interface IPageListItem : ITeaserData, IPageTitle, ITags
{
}

public interface ITeaserData : IGlassBase
{
    string Headline { get; set; }
    string Description { get;set; }
}

public interface IPageTitle : IGlassBase
{
    string Title { get; set; }
    string Subheadline { get; set; }
}

public interface ITags : IGlassBase
{
    ITagList TagList { get; set; }
}
```

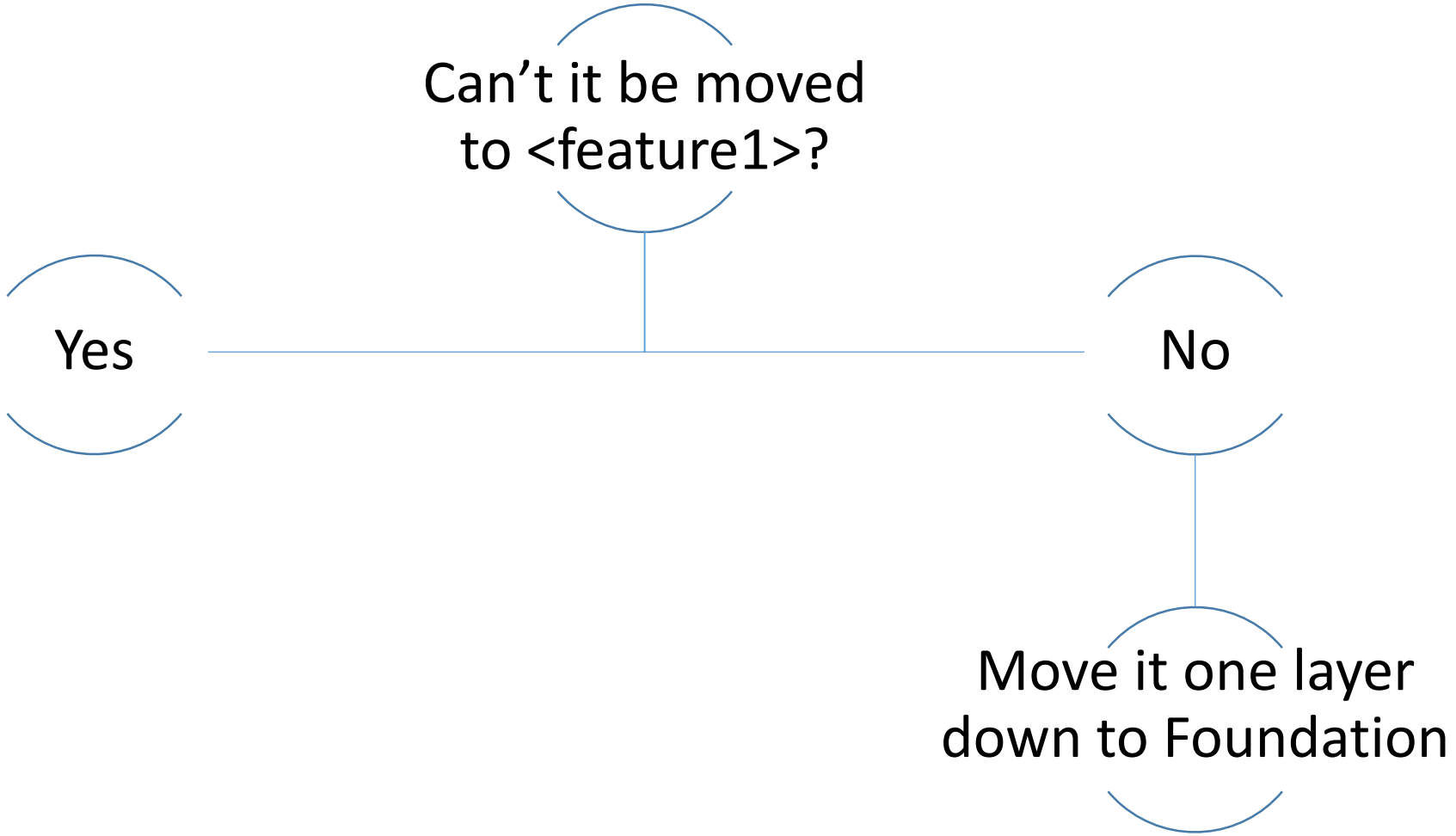
```
public ActionResult PageListModule()
{
    return this.PartialView(
        sitecoreContext.GetItem<IPageList>(
            RenderingContext.Current.Rendering.DataSource));
}
```

Some common use cases

Some common use cases

Description	Layer
Custom field implementation	Foundation
Field/item validations • Use parameters to make it flexible	Foundation
Search • Basic search • Specific search for modules	 Foundation Feature
Placeholders in your layout	Project
Feature related placeholders	Feature
Container grids	Project
Workflow and its custom implementation	Foundation
External service integration • One-feature specific, like Google Maps • More foundational, like a CRM integration	 Feature Foundation

I need <feature1.template> in my new <feature2>, what should I do?

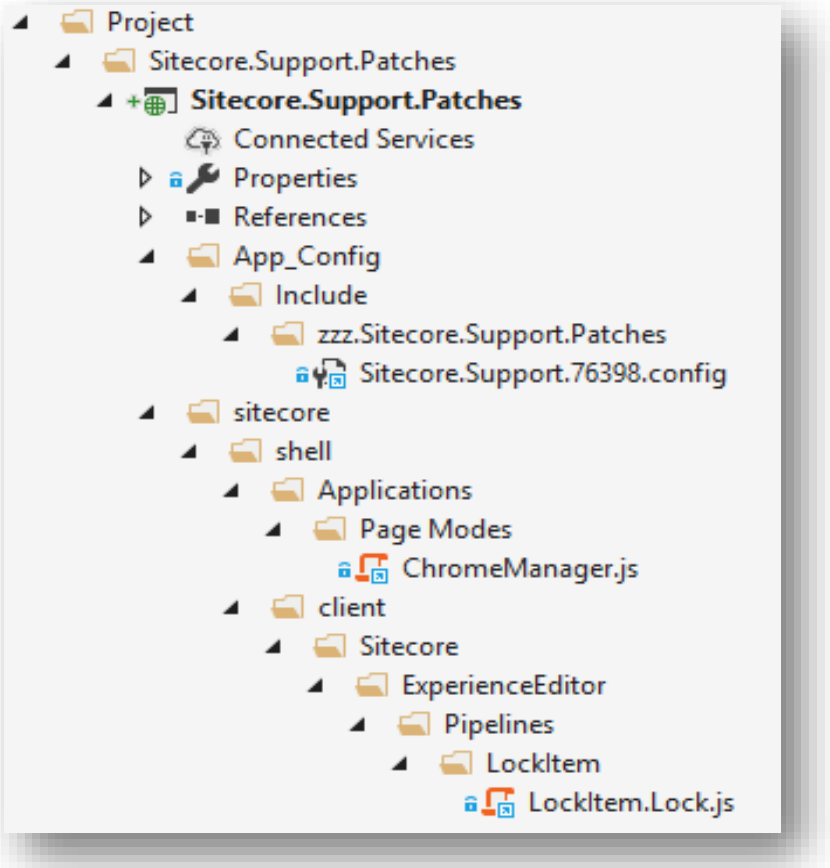


Development with frontend dev team

- Usually frontend development is splitted from backend development
- They should also follow Helix principles
  - Split HTML, JS and CSS to components
- At the end we get the minified CSS, JS and HTML
- Where to put their CSS and JS? → Project layer

How to include Sitecore Support patches

- Create a new project for them
- Use Visual Studio „Insert file as a link” function → Easy to maintain and split them even if they have a strict structure in the patch



Unicorn vs. TDS for Helix

- Create serialization configuration (Unicorn) or project (TDS) per project
- Separate to content and development related items
  - Development related items: templates, renderings, placeholders etc. (deploy always)
  - Content: dictionary, page items (deploy once)
- Use dependencies for your configuration
  - TDS: VS project configuration
  - Unicorn: use the „dependencies” attribute in the config file
- Be aware that TDS create duplications because it seralializes the whole tree

Helix.Skeleton

Accelerator and developer tool for Helix based solutions

Helix.Skekeleton – Evolution of the project

1. Created a base solution with some common Foundation and a Demo Feature
2. Project name replacer implementation (init.ps1)
3. Nuget package version replacer
    1. Use JSON file configuration
    2. Big refactor to possibly use custom configuration
4. Project addition implementation (add.ps1)
    1. Automatic unique GUID generation
    2. Automatic static GUID generation

Demo

Thank you!

Join and find me on twitter, sitecore.stackexchange, github and Sitecore community slack channel: **@trnktms**

Tamás Tárnok – ALLWIN Informatika

Sources:

- PowerPoint template by @jammykam
- Official Helix documentation: http://helix.sitecore.net/
- Habitat: https://github.com/Sitecore/Habitat
- Helix Skeleton: https://github.com/trnktms/Helix.Skeleton