Hydrogen Sessions

#3 – Organização da Solução

Source Code

A solução do Hydrogen está no source code na pasta $/Lithium/Core/Hydrogen.

Esta área “Core” do Lithium refere-se a componentes fundamentais (o Hydrogen e o Identity Server).

Os micro serviços ficam todos na pasta “Microservices” (organizados por categoria) e o SDK na pasta com o mesmo nome.

Existem 3 linhas de código:

- Mainline-v1: contém a V1 e é utilizada apenas para efeitos de manutenção corretiva (existem ainda vários serviços que referenciam esta versão).
- Development-v2: contém a “versão em desenvolvimento” da V2.
- Mainline-v2: contém a “versão produtiva” da V2, a única que outros componentes (SDK, micro serviços, etc.) devem referenciar.

A existência de 2 linhas para a V2 é um “preciosismo” que serve apenas para assegurar que é possível corrigir um bug em produtivo quando estiverem ainda em curso desenvolvimento da linha Dev que não possam ser sincronizados. Na prática isso ainda não aconteceu nunca, mas será mais provável agora que já há micro serviços com essa versão em produtivo.

Note-se que o Orinoco opera apenas sobre a linha Dev, pelo que os desenvolvimentos devem ser feitos sempre nessa linha e rapidamente sincronizados para a Mainline.

Naming (Solução, Projetos)

Um problema clássico dos projetos em TFS prende-se com o tamanho máximo do path dos ficheiros em disco. Por isso, optou-se por abreviar até ao razoável todos os nomes.

A solução chama-se Primavera.Hydrogen.sln (podia ser Hydrogen.sln) e cada projeto tem apenas o nome significativo da assembly (o sufixo para lá de Primavera.Hydrogen): Core, EventBus.Abstractions, EventBus.Azure.

Obviamente, o nome do projeto mapeia diretamente com nome da assembly e com o namespace por defeito. Por exemplo:

Storage.Azure.csproj:

<AssemblyName>Primavera.Hydrogen.Storage.Azure</AssemblyName>

<RootNamespace>Primavera.Hydrogen.Storage.Azure</RootNamespace>

* Estes valores são definidos no csproj.

* A única exceção é o projeto Core que não usa esse prefixo no namespace (Primavera.Hydrogen), apenas no assembly name (Primavera.Hydrogen.Core), porque esse sufixo não acrescenta nada.

Obviamente, o nome do projeto e da assembly é uma decisão muito importante e que deve ser tomada com atenção para refletir corretamente as funcionalidades que serão incluídas.

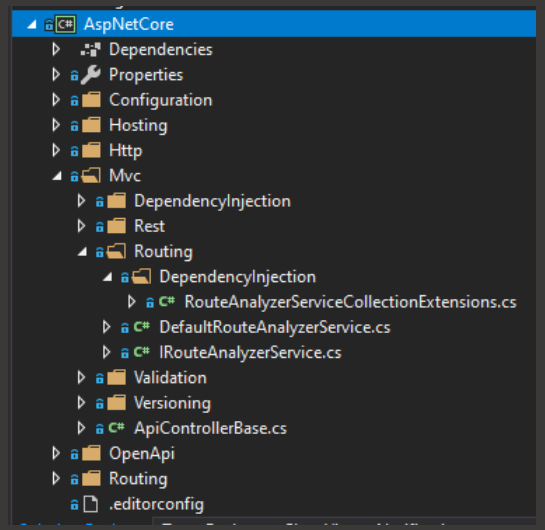
Organização dos projetos

Existem 5 tipos de projetos na solução:

- Os componentes runtime (AspNetCore, AspNetCore.Authentication, etc.). São colocados todos na raiz da solução.
- Os componentes design-time (usados pela própria SLN ou pelo SDK). São colocados numa pasta com o nome “DesignTime” (no disco tb).
- Os testes unitários. Na pasta _UnitTests.
- Os testes de integração. Na pasta _IntegrationTests.
- Aplicações de demonstração. Na pasta _Apps.

Namespaces e Organização das Pastas

Em cada projeto, os tipos (classes, interfaces, etc.) devem ser colocados numa pasta com o nome correspondente ao namespace.



A única exceção é a pasta “DependencyInjection” em que se devem colocar as extensões usadas pelo mecanismo de injeção de dependências, sem adicionar esse sufixo ao namespace.

Ou seja, Primavera.Hydrogen.AspNetCore.Mvc.Routing.RouteAnalyzerServiceCollectionExtensions.cs.

Target Framework

Todos os projetos devem, por defeito, compilar para .NET Standard 2.0 (para maximizar a compatibilidade com os produtos .NET):

```
<TargetFramework>netstandard2.0</TargetFramework>
```

A exceção são os projetos que devem fazer uma afirmação sobre o runtime em que devem ser executados. Nomeadamente os projetos relacionados com o ASP.NET Core.

```
<TargetFramework>netcoreapp3.1</TargetFramework>
```

AssemblyInfo

Todos os projetos definem as propriedades no ficheiro AssemblyInfo.cs:

```
using System.Reflection;
using System.Resources;
using System.Runtime.CompilerServices;
// NOTE: When changing the assembly description, remember to change it also in the CSPROJ NuGet properties
[assembly: AssemblyTitle("PRIMAVERA Hydrogen ASP.NET Core")]
[assembly: AssemblyDescription("Class library that contains types that support the development of Web applications with ASP.NET Core.")]
[assembly: AssemblyProduct("PRIMAVERA Hydrogen")]
[assembly: AssemblyCompany("PRIMAVERA Business Software Solutions, S.A.")]
[assembly: AssemblyCopyright("Copyright © 2017-2020 PRIMAVERA")]
[assembly: AssemblyTrademark("PRIMAVERA")]
[assembly: NeutralResourcesLanguage("en-US")]
[assembly: AssemblyVersion("2.0.0.0")]
[assembly: AssemblyFileVersion("2.0.0.0")]
[assembly: AssemblyInformationalVersion("*development*")]
```

NOTA: Parte destas propriedades são sobrepostas nas builds.

NuGet Packages

Praticamente todos os projetos produzem um package .NuGet.

Uma vez que o publisher não é capaz de ler as propriedades de AssemblyInfo, é necessário definir algumas dessas propriedades no csproj:

```
<PropertyGroup>
  <IsPackable>true</IsPackable>
  <Description>Class library that contains types that support the development of Web applications with ASP.NET Core.</Description>
  <Authors>PRIMAVERA Business Software Solutions, S.A.</Authors>
  <PackageTags>PRIMAVERA;Lithium;Hydrogen</PackageTags>
</PropertyGroup>
```

Primavera.Hydrogen.DesignTime.Configuration

Este package é adicionado a todos os projetos e configura:

- As regras de code analysis (analyzers)
- A signature key
- O ficheiro .editorconfig (mais regras de code analysis)
- O ficheiro .tfignore

Testes e Code Coverage

As builds Dev e Mainline executam os testes e analisam a code coverage e podem falhar dependendo do nível de code coverage:

- Development: executa testes unitários apenas e valida que a code coverage é superior a 50%.
- Mainline: executa testes unitários e de integração e valida que a code coverage é superior a 75%.