Very first version of the CK-Env documentation.

Not finished yet but some important and useful information should be KNOWN.

Contents

[CK-Env & World Concept 1](#_Toc535340069)

[Anatomy of a World xml file 2](#_Toc535340070)

[How XTypedObjects work 3](#_Toc535340071)

[Artifact repositories 4](#_Toc535340072)

[Solution Settings 5](#_Toc535340073)

[Solution Settings Composability 5](#_Toc535340074)

[NuGetSources & RemoveNuGetSourceNames 5](#_Toc535340075)

[ArtfifactTargets 6](#_Toc535340076)

[Plugins 6](#_Toc535340077)

[Other properties 6](#_Toc535340078)

[Folders, Repositories & Solutions 6](#_Toc535340079)

[Command Line Interface 8](#_Toc535340080)

[The Workflows 9](#_Toc535340081)

[Checking out a World 16](#_Toc535340082)

[Repositories status and Global state overview 16](#_Toc535340083)

[Homogenization of configurations: ApplySettings 18](#_Toc535340084)

[Global operations on the Git repositories 19](#_Toc535340085)

[Building: Introducing the CodeCakeBuilder interface and the ZeroVersion 9](#_Toc535340086)

[CI Building globally: Introducing package propagation 20](#_Toc535340087)

[Working across the World: The -local branch. 22](#_Toc535340088)

[Releasing: From roadmap to publishing 23](#_Toc535340089)

[A World is Multiple: Long-Term-Support branches 24](#_Toc535340090)

# CK-Env & World Concept

The goal of CK-Env is to provide a way to handle multiple repositories and solutions in a coherent manner: package versions can be unified and propagated across the different solutions, all solutions can be compiled, tested and released.

To achieve this, CK-Env uses “Worlds”. A World describes a set of Git repositories that contain a .Net Solution in a “develop” branch. Worlds are xml files. Currently they are contained in CK-Env repository at the root.

Some rules must be followed, most of them are conventions that we already use:

* A Solution should not belong to more than one World.
* The solution name is the same as the repository name and uses ‘-‘ instead of ‘.’ (this allows to easily differentiate between a Project/Package (CK.SqlServer) and the Repository/Solution (CK-SqlServer).
* The Primary solution is a .sln file with a CodeCakeBuilder project that is the “Build Project”: the build chain of the Solution is entirely inside this project: CK-Env is an external tool that can help edit/transform a Solution but a Solution can always be used independently (Checkout, Modify, Commit, Push, Release, etc.)
* Tests projects: names must end with “.Tests” and reside in a “Tests” root folder.

CK-Env is NOT public and is currently “source based”: the repository <https://gitlab.com/Signature-Code/CK-Env.git> must be checked out.

The only executable project is **CKli**.

On start, CKli displays the available Worlds in a menu:

- A1Test

> 1 - <Current> => C:\Dev\CKEnvTest\A1Test

- A2Test

> 2 - <Current> => C:\Dev\CKEnvTest\A2Test

- CK

> 3 - <Current> => C:\Dev\CK

- ENGIE

> 4 - <Current> => (No local mapping)

- Signature

> 5 - <Current> => (No local mapping)

> x - Exit

By design, CK-Env handles the CK-World in its own folder (if CK-Env has been checked out in “D:\Test\Dev”, CK would be mapped to this folder). Other worlds must be mapped thanks to the “LocalWorldRootPathMapping.txt” file (also at the root of the CK-Env folder). This is a simple file that define local mappings for worlds (sample content):

A1Test > C:\Dev\CKEnvTest\A1Test

A2Test > C:\Dev\CKEnvTest\A2Test

The full CK-World can be found here: <https://gitlab.com/Signature-Code/CK-Env/blob/develop/CK-World.xml>

# Anatomy of a World xml file

Once a world has been selected (by number), its xml description is read and analyzed. Any error in the xml will be logged and the process stops (xml world files have to be valid).

A World xml files describes a set of solution repositories, configurations for them and tools or definitions. Order matters: xml elements are concretized as actual objects that depends on each other. Some elements define “services” that will be used by other services and objects. They appear at the top of the file:

<SharedHttpClient />

<ArtifactCenter />

<LocalFeedProvider />

<CKEnvKeyVault />

<NuGetClient />

<CKSetupClient />

<SolutionCentral />

Some elements are configuration objects, others represent repository, branches, solutions, etc. Before describing them, one must understand the way the XML is used to build the actual “program” of a World.

## How XTypedObjects work

Xml elements of a world are object definitions. For instance, <SharedHttpClient /> creates an instance of the class below:

public class XSharedHttpClient : XTypedObject, IDisposable

{

HttpClient \_shared;

public XSharedHttpClient( Initializer initializer )

: base( initializer )

{

initializer.Services.Add( this );

}

public HttpClient Shared => \_shared ?? (\_shared = new HttpClient());

void IDisposable.Dispose() => \_shared?.Dispose();

}

The constructor publishes itself as a service: children AND siblings of this element can then use it. The <NuGetClient /> uses it:

public class XNuGetClient : XTypedObject, IDisposable

{

readonly XSharedHttpClient \_sharedHttpClient;

readonly XSecretKeyStore \_secretKeyStore;

readonly NuGetClient \_nuGetClient;

public XNuGetClient(

XSharedHttpClient sharedHttpClient,

XSecretKeyStore secretKeyStore,

XArtifactCenter artifact,

FileSystem fs,

Initializer initializer )

: base( initializer )

{

\_sharedHttpClient = sharedHttpClient;

\_secretKeyStore = secretKeyStore;

\_nuGetClient = new NuGetClient( \_sharedHttpClient.Shared, \_secretKeyStore );

fs.ServiceContainer.Add<INuGetClient>( \_nuGetClient );

artifact.ArtifactCenter.Add( \_nuGetClient );

initializer.Services.Add( this );

}

public INuGetClient NuGetClient => \_nuGetClient;

void IDisposable.Dispose() => \_nuGetClient.Dispose();

}

Service availability follows the XML structure: the injected objects are available to all next siblings but not to parent (or other children of parent objects).

Note that some objects publish services or objects only to their children, for instance a Solution is not exposed to the remainder of the XML elements, only its children may need it:

public abstract class XSolutionBase : XPathItem

{

readonly XBranch \_branch;

protected XSolutionBase(

Initializer initializer,

XBranch branch,

XSolutionCentral central,

XSolutionSettings solutionSettings,

NormalizedPath branchBasedSolutionFilePath )

: base( initializer,

branch.FileSystem,

FileSystemItemKind.File,

branch.FullPath.Combine( branchBasedSolutionFilePath ) )

{

\_branch = branch;

SolutionSettings = solutionSettings;

initializer.ChildServices.Add( this );

branch.Register( this );

}

// …(truncated)

}

## Artifact repositories

<ArtifactRepositories>

<Repository Type="NuGetAzure"

Organization="Signature-OpenSource"

FeedName="Default"

CheckName="NuGetAzure:Signature-OpenSource-Default"

CheckSecretKeyName="AZURE\_FEED\_SIGNATURE\_OPENSOURCE\_PAT" />

<Repository Type="CKSetup"

CheckName="CKSetup:Public"

CheckSecretKeyName="CKSETUPREMOTESTORE\_PUSH\_API\_KEY" />

</ArtifactRepositories>

This one defines the artifact repositories that will receive the produced artifacts. All repositories must be identified by a Name (to be referenced later) and use a “secret” to be able to push artifacts in it.

To homogenize the naming, these 2 names can be computed by the actual objects and the convention used depend on their actual type. CheckName and CheckSecretKeyName (that are optional) are used to expose these names in the xml.

There are currently 3 types of repositories:

* CKSetup
  + Handles CKSetup component store. The public one name is by design CKSetup:Public, but other store can be defined with a required name and url, for instance:

<Repository Type="CKSetup"

Name="CKEnvTest"

Url="https://ckenvtest.cksetup.invenietis.net"

CheckName="CKSetup:CKEnvTest"

CheckSecretKeyName="CKSETUPREMOTESTORE\_CKENVTEST\_PUSH\_API\_KEY" />

(We can see here the automatic derivation of the final name and required secret key name.)

* NuGetStandard
  + Defines NuGet feed. Its configuration requires an explicit name for the secret key (hence, the CheckSecretKeyName is obviously useless):

<Repository Type="NuGetStandard"

Name="nuget.org"

Url="https://api.nuget.org/v3/index.json"

SecretKeyName="NUGET\_API\_KEY"

CheckName="NuGetStandard:nuget.org" />

(This has not been currently tested since we only use Azure feeds.)

* NuGetAzure:
  + Define Azure feed: the Organization determines the Personal Access Token that will be used:

<Repository Type="NuGetAzure"

Organization="Signature-OpenSource"

FeedName="CKEnvTest3"

CheckName="NuGetAzure:Signature-OpenSource-CKEnvTest3"

CheckSecretKeyName="AZURE\_FEED\_SIGNATURE\_OPENSOURCE\_PAT" />

Above is a test feed definition (not to be used).

## Solution Settings

This element defines the solution configuration (its content will drive a lot of stuff).

<SolutionSettings>

<NuGetSources>

<add Name="Signature-OpenSource"

Url="https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/Default/nuget/v3/index.json" >

<Credentials UserName="SignatureOpenSource"

Password="c56geu5acyhfgeiqhdl3t2xo5xumlztnjf736zdbogyxtaulydlq" />

</add>

<add Name="CKEnvTest3"

Url="https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/CKEnvTest3/nuget/v3/index.json" >

<Credentials UserName="SignatureOpenSource"

Password="c56geu5acyhfgeiqhdl3t2xo5xumlztnjf736zdbogyxtaulydlq" />

</add>

</NuGetSources>

<RemoveNuGetSourceNames>

<add Name="CKEnvTest" />

<add Name="Invenietis - Release" />

<add Name="Invenietis - Preview" />

<add Name="Invenietis - CI" />

</RemoveNuGetSourceNames>

<ArtifactTargets>

<add Name="NuGetAzure:Signature-OpenSource-CKEnvTest3" />

<add Name="CKSetup:CKEnvTest" />

</ArtifactTargets>

<Plugins>

<add Type="CK.Env.Plugins.SolutionDriver, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.CodeCakeBuilderFolder, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.CodeCakeBuilderCSProjFile, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.CodeCakeBuilderKeyVaultFile, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.CommonFolder, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.SharedPropsFile, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.CKSetupStoreTestHelperConfigFile, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.RepositoryXmlFile, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.NugetConfigFile, CK.Env.Plugins" />

<add Type="CK.Env.Plugins.SolutionFiles.GitIgnoreFile, CK.Env.Plugins" />

<add Type="CK.BuildSystem.Appveyor.AppveyorFile, CK.BuildSystem.Appveyor" />

</Plugins>

</SolutionSettings>

### Solution Settings Composability

This element is “composable”. Any instance of <SolutionSettings> that appears in the xml is merged with the current one (see *How XTypedObjects work*) and applies to the remainder of the xml. This is the reason why the <add … /> element: <clear /> and <remove … /> are also supported to handle collections.

### NuGetSources & RemoveNuGetSourceNames

This element drives the NuGet sources that must be used by the solution: the nuget.config file is updated (see ApplySeetings below). With the above configuration, the nuget.config file is:

<configuration>

<packageSources>

<clear />

<add key="Signature-OpenSource"

value="https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/Default/nuget/v3/index.json" />

<add key="CKEnvTest3"

value="https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/CKEnvTest3/nuget/v3/index.json" />

</packageSources>

<packageSourceCredentials>

<Signature-OpenSource>

<add key="Username" value="SignatureOpenSource" />

<add key="ClearTextPassword" value="c56geu5acyhfgeiqhdl3t2xo5xumlztnjf736zdbogyxtaulydlq" />

</Signature-OpenSource>

<CKEnvTest3>

<add key="Username" value="SignatureOpenSource" />

<add key="ClearTextPassword" value="c56geu5acyhfgeiqhdl3t2xo5xumlztnjf736zdbogyxtaulydlq" />

</CKEnvTest3>

</packageSourceCredentials>

</configuration>

RemoveNuGetSourceNames enables to define source names that must be removed from the nuget.config file.

### ArtfifactTargets

This is the set of artifacts repositories where the solution must push its artifacts. There must be at most one instance of artifact type in this set… A choice has been made here: we don’t handle multiple push targets per artifact type.

If pushing into more than one target repository is required, this must be done elsewhere, typically at by the primary repository that receives the artifacts.

### Plugins

This defines the CK-Env plugins that will handle solutions. The ones presented here are required (some may not be used depending on the configuration or the content of the solution but they must be kept as-is).

### Other properties

Solution settings holds other properties that are not visible in the above configuration: they use their default values.

|  |  |
| --- | --- |
| bool NoUnitTests | Gets whether the solution has no unit tests.  Defaults to false. |
| bool NoStrongNameSigning | Gets whether no strong name singing should be used.  Defaults to false. |
| bool UseCKSetup | Gets whether the solution uses CKSetup components.  When true (and when NoUnitTests is false), a RemoteStore.TestHelper.config file is created during build so that stores in CK-Env local folders are used instead of the default local (%UserProfile%AppData\Local\CKSetupStore) and default remote (https://cksetup.invenietis.net). |
| bool DisableSourceLink | Gets whether source link is disabled.  Impacts Common/Shared.props file. |
| string SqlServer | Gets the name of the SqlServer that is used.  Defaults to null.  Names are the ones of Appveyor (<https://www.appveyor.com/docs/services-databases/>).  "2008R2SP2", "2012SP1", "2014", "2016", "2017", etc. |

## Folders, Repositories & Solutions

Solutions reside inside Branches of Git repositories that reside in Folder and are bound to the url of the ‘origin’ remote repository:

<Folder Name="Yodii-Projects">

<GitFolder Name="Yodii-Script" Url="https://github.com/Invenietis/Yodii-Script.git">

<Branch Name="develop">

<PrimarySolution />

</Branch>

</GitFolder>

</Folder>

This solution above is a very standard one. Some are a bit more complicated because they are altering the default solution settings (by composing the ambient one), or defines more specific information on the solution itself:

<Folder Name="SimpleGitVersion">

<GitFolder Name="CodeCake" Url="https://github.com/SimpleGitVersion/CodeCake.git">

<SolutionSettings NoUnitTests="true" NoStrongNameSigning="true" />

<Branch Name="develop">

<PrimarySolution />

</Branch>

</GitFolder>

<GitFolder Name="SGV-Net" Url="https://github.com/SimpleGitVersion/SGV-Net.git">

<SolutionSettings DisableSourceLink="true" />

<Branch Name="develop">

<PrimarySolution />

</Branch>

</GitFolder>

</Folder>

The PrimarySolution can specify some useful properties:

<Folder Name="CK-Database-Projects">

<GitFolder Name="CK-Setup" Url="https://gitlab.com/Signature-Code/CK-Setup.git">

<SolutionSettings UseCKSetup="true" />

<Branch Name="develop">

<PrimarySolution CKSetupComponentProjects="CKSetup.Runner;CKSetup" />

</Branch>

</GitFolder>

<GitFolder Name="CK-Database" Url="https://gitlab.com/Signature-Code/CK-Database.git">

<SolutionSettings SqlServer="2016" UseCKSetup="True" />

<Branch Name="develop">

<PrimarySolution CKSetupComponentProjects="CK.StObj.Model;CK.StObj.Runtime;CK.StObj.Engine;CK.Setupable.Model;CK.Setupable.Runtime;CK.Setupable.Engine;CK.SqlServer.Setup.Model;CK.SqlServer.Setup.Runtime;CK.SqlServer.Setup.Engine" />

</Branch>

</GitFolder>

<GitFolder Name="CK-DB" Url="https://github.com/Invenietis/CK-DB.git">

<SolutionSettings SqlServer="2016" UseCKSetup="True" />

<Branch Name="develop">

<PrimarySolution TestProjectsArePublished="True" />

</Branch>

</GitFolder>

<GitFolder Name="CK-DB-GitHub" Url="https://github.com/Invenietis/CK-DB-GitHub.git">

<SolutionSettings SqlServer="2016" UseCKSetup="True" />

<Branch Name="develop">

<PrimarySolution TestProjectsArePublished="True" />

</Branch>

</GitFolder>

</Folder>

This describes the structure of a World and is enough to enable global handling of multiple repositories.

# Command Line Interface

The command line interface is quite simple but powerful enough to correctly support basic user interactions and, more importantly, very easy to extend (any objects of the context can register some of its methods as Commands but detailing this is not in the scope of this document).

As described before, the very first thing that CKli requires is to select a World:

- A1Test

> 1 - <Current> => C:\Dev\CKEnvTest\A1Test

- A2Test

> 2 - <Current> => C:\Dev\CKEnvTest\A2Test

- CK

> 3 - <Current> => C:\Dev\CK

- ENGIE

> 4 - <Current> => (No local mapping)

- Signature

> 5 - <Current> => (No local mapping)

> x - Exit

Once done, CKli exposes commands to support user interactions. Commands are:

* Identified by a unique name that is a path.
* Are enabled or disabled dynamically depending on the system/object state.

This Command Line Interface (names, parameters, confirmations, etc.) is generally **case sensitive**!

The CKli prompt offers 5 intrinsic commands:

> World: CK - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

Three basic ones are:

* **cls**: Clears screen like in the good (?) old DOS.
* **restart**: Closes the current World and enables to pick another one.
* **exit**: Quits CKli application.

The **list** command lists all or a subset of the available commands. To select one or more commands, “globbing” is used on the command path. The CKEnvVault XTypedObject for instance exposes 3 commands, however at start, only 2 of them are displayed:

> World: CK - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> list \*\*/\*Vault

Available Commands matching '\*\*/\*Vault':

KeyVault\ClearKeyVault

KeyVault\OpenKeyVault

> World: CK - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> list Key\*\*

Available Commands matching 'Key\*\*':

KeyVault\ClearKeyVault

KeyVault\OpenKeyVault

Note that this is a “path globbing”: \*\* matches any number of “sub directories” and \* matches any number of characters in a part of the path (and ? matches a unique character).

Let’s run the OpenKeyVault command:

> World: A2Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Open\*Vault\*\*

[required] - passPhrase: a

> Trace: Executing KeyVault\OpenKeyVault.

| - Info: Opened existing Key Vault with keys: AZURE\_FEED\_SIGNATURE\_OPENSOURCE\_PAT, CODECAKEBUILDER\_SECRET\_KEY, APPVEYOR\_ENCRYPTED\_CODECAKEBUILDER\_SECRET\_KEY, CKSETUPREMOTESTORE\_PUSH\_API\_KEY, CKSETUPREMOTESTORE\_CKENVTEST\_PUSH\_API\_KEY.

(For this test world A2Test, the pass phrase is just ‘a’.)

Now that the key vault has been opened, the available commands have changed:

> list Key\*\*

Available Commands matching 'Key\*\*':

KeyVault\ClearKeyVault

KeyVault\SaveKeyVault

The SaveKeyVault command can change the pass phrase of the vault thanks to an optional parameter:

> World: A2Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Save\*\*

[default value: <null>] - newPassPhrase: A better passphrase.

> Trace: Executing KeyVault\SaveKeyVault.

| - Info: Saved Key Vault with keys: AZURE\_FEED\_SIGNATURE\_OPENSOURCE\_PAT, CODECAKEBUILDER\_SECRET\_KEY, APPVEYOR\_ENCRYPTED\_CODECAKEBUILDER\_SECRET\_KEY, CKSETUPREMOTESTORE\_PUSH\_API\_KEY, CKSETUPREMOTESTORE\_CKENVTEST\_PUSH\_API\_KEY.

When more than one command names collide, this may be fine… if and only if they have the same argument names (internally named payload):

> World: A2Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*K\*\*

> Warn: Pattern '\*\*K\*\*' matches require 2 different payloads.

| - Warn: (newPassPhrase): KeyVault\SaveKeyVault

| <No payload>: KeyVault\ClearKeyVault

When it is the same payload, multiple commands can be executed at once. Most commands require a confirmation before the actual run (this is not the case of the KeyVault commands we’ve seen before):

> run \*\*CK-Core/\*\*ApplySettings

Confirm execution of:

CK-Core-Projects\CK-Core\branches\develop-local\CodecakeBuilder\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\CodecakeBuilder\CodeCakeBuilder.csproj\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\CodecakeBuilder\CodeCakeBuilderKeyVault.txt\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\Common\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\RemoteStore.TestHelper.config\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\RepositoryInfo.xml\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\NuGet.config\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\.gitignore\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\Appveyor.yml\ApplySettings

Y/N?

This ends the overview of the CKli command line interface since this is enough to use the tool.

# Building Basics: Local Feed, CodeCakeBuilder and ZeroBuild

Building is one of the primary goal of CK-Env. Build, Test, and Publication of Solutions are actually handled by the CodeCakeBuilder project of each Solution.

Everything has been made to enable purely local operations without any side effect on the external world (ie. push to feeds, stores, etc). The LocalFeed folder plays an important role in the “locality”, but before explaining the LocalFeed, the ZeroBuild and whole world builds, let’s have a look at a simple command (that is not really useful in itself): the simple Build command.

## The simple Build command

This command is exposed by the SolutionDriver plugin and simply runs the CodeCakeBuilder of the Solution. Below is the result of a Build in a World that contains only one simple Solution: analyzing it enables to understand key aspects of the system.

This Build command is not really useful, most of the times it is the “AllBuild” or “Release” commands that are used. It is exposed as a way to test the CodeCakeBuilder project of each Solution, and here it offers us a way to present some basics of the build process itself.

> World: A2Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Build

[default value: True] - upgradeLocalDependencies:

[default value: True] - withUnitTest:

Confirm execution of:

Projects\CK-UnitsOfMeasure\branches\develop\SolutionDriver\Build

With payload:

[default value: True] - upgradeLocalDependencies - Value = 'True'

[default value: True] - withUnitTest - Value = 'True'

Y/N?

The Build command, by default, upgrades the project dependencies based on the locally available packages in the Local Feed (LocalFeed is described below).

Unit tests, by default, are executed but can be skipped.

Y> Trace: Executing Projects\CK-UnitsOfMeasure\branches\develop\SolutionDriver\Build.

| > Trace: Loading solution Projects\CK-UnitsOfMeasure\branches\develop\CK-UnitsOfMeasure.sln.

| | > Trace: Loading project file Projects\…(truncated)…\develop\CodeCakeBuilder\CodeCakeBuilder.csproj.

| | > Trace: Loading project file Projects\…(truncated)…\CK.UnitsOfMeasure\CK.UnitsOfMeasure.csproj.

| | | > Trace: Loading project file Projects\CK-UnitsOfMeasure\branches\develop\Common\Shared.props.

| | > Trace: Loading project file …(truncated)…\Tests\CK.UnitsOfMeasure.Tests\CK.UnitsOfMeasure.Tests.csproj.

This displays the load of a Solution and shows that the .sln file is analyzed, the .csproj it contains are also analyzed since the <Import Project="../Common/Shared.props" /> has been handled.

Once a Solution is loaded, we know:

1. Its Projects
2. Its target frameworks.
3. The Project references between projects inside the Solution.
4. The Packages used by each project with their version and conditional frameworks if any.

This is where the upgrade of the dependencies takes place (but here this project has no dependencies, so nothing is done).

| > Info: Committing changes in 'Projects\CK-UnitsOfMeasure' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

A commit is automatically created here.

CAUTION: any currently modified files will be committed as well!

| - Info: Version to build: '0.1.2--0039-develop'.

| Using CodeCakeBuilder with source compilation (dotnet run).

| > Info: Replacing Projects\CK-UnitsOfMeasure\branches\develop\RepositoryInfo.xml.

| > Info: Replacing Projects\CK-UnitsOfMeasure\branches\develop\NuGet.config.

1. SimpleGitVersion is used by CK-Env: the computed version number is displayed (CodeCakeBuilder will do the same at its level).
2. Since there is no ZeroBuilder available for this Solution (this is explained later), the CodeCakeBuilder is ran “by compilation”, just like on a CI server.

If the ZeroBuilder was available you would have had:

| Using available CodeCakeBuilder published Zero version.

1. Some files are modified and this is an important aspect to understand: Builds by CK-Env are done on a “dirty working folder”. This is achieved thanks to the Repository.xml option: <https://github.com/SimpleGitVersion/SGV-Net/wiki/Optional-RepositoryInfo.xml-file#debugging>

| > Info: Building develop/CK-UnitsOfMeasure, Target Version = 0.1.2--0039-develop

| | > Trace: dotnet run --project CodeCakeBuilder -autointeraction -PublishDirtyRepo=Y -PushToRemote=N

| | | - Info: <StdOut> Working in Solution directory: 'C:\Dev\CKEnvTest\A2Test\Projects\CK-UnitsOfMeasure'.

| | | <StdOut> Path(s) added:

| | | <StdOut> Dynamic pattern path(s) added: %UserProfile%/.nuget/packages/\*\*/tools\*

| | | <StdOut> Reading environment variables from CodeCakeBuilderKeyVault.txt file.

| | | <StdOut> Environment variable 'AZURE\_FEED\_SIGNATURE\_OPENSOURCE\_PAT' set from key vault.

| | | <StdOut> Using RepositoryInfo.xml: <RepositoryInfo xmlns="http://csemver.org/schemas/2015">

| | | <StdOut> <Debug IgnoreDirtyWorkingFolder="true" />

| | | <StdOut> <StartingVersionForCSemVer>v0.1.0-alpha</StartingVersionForCSemVer>

| | | <StdOut> <Branches>

| | | <StdOut> <Branch Name="develop" VersionName="develop" CIVersionMode="LastReleaseBased" />

| | | <StdOut> </Branches>

| | | <StdOut> </RepositoryInfo>

| | | <StdOut> Working folder is Dirty!

| | | Checking this has been disabled since IgnoreDirtyWorkingFolder is true.

| | | <StdOut> At least one Modified file found: NuGet.config.

| | | <StdOut> No tag found on commit itself.

| | | <StdOut> Base tag below: 0.1.1

| | | <StdOut> CI release: '0.1.2--0039-develop'.

| | | <StdOut> Possible version(s) : 0.1.2-alpha, 0.1.2-beta, ………(truncated)……… 1.0.0-rc, 1.0.0

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Check-Repository

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Check-Repository

| | | <StdOut> Adding NuGet local feed: C:\Dev\CKEnvTest\A2Test\LocalFeed\CI

| | | <StdOut> Push to Remote feeds? (Y/N): N

| | | <StdOut> Answered by command line argument -PushToRemote=N.

| | | <StdOut> Initializing with sources:

| | | <StdOut> Signature-OpenSource

| | | => https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/Default/nuget/v3/index.json

| | | <StdOut> CKEnvTest3

| | | => https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/CKEnvTest3/nuget/v3/index.json

| | | <StdOut> LocalFeed-Develop => C:\Dev\CKEnvTest\A2Test\LocalFeed\CI

| | | <StdOut> Created 0 feed end point(s) in VSS\_NUGET\_EXTERNAL\_FEED\_ENDPOINTS.

| | | <StdOut> Package CK.UnitsOfMeasure must be published to remote feed 'LocalFeed-Develop'.

| | | <StdOut> ==> 1 package(s) must be published to remote feed 'LocalFeed-Develop'.

| | | <StdOut> Feed 'LocalFeed-Develop': All 1 packages must be pushed.

| | | <StdOut> Should actually publish 1 out of 1 projects

| | | with version=0.1.2--0039-develop and configuration=Debug: CK.UnitsOfMeasure

| | | <StdOut> Finished executing task: Check-Repository

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Clean

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Clean

| | | <StdOut> Cleaning directory C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/CK.UnitsOfMeasure/bin

| | | <StdOut> Cleaning directory C:/Dev/CKEnvTest/………(truncated)………/Tests/CK.UnitsOfMeasure.Tests/bin

| | | <StdOut> Cleaning directory C:/Dev/CKEnvTest/………(truncated)………/CodeCakeBuilder/Releases

| | | <StdOut> Deleting file C:/Dev/CKEnvTest/………(truncated)………/TestResult.Net461.xml

| | | <StdOut> Finished executing task: Clean

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Build

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Build

| | | <StdOut> Copying file CK-UnitsOfMeasure.sln to

| | | C:/Dev/CKEnvTest/……(truncated)……/CK-UnitsOfMeasure.slna76d2a8581a84cdd90652003608015c4.sln

| | | <StdOut> Executing: "C:/Program Files/dotnet/dotnet.exe" build "C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/CK-UnitsOfMeasure.slna76d2a8581a84cdd90652003608015c4.sln" --configuration Debug /p:CakeBuild="true" /p:Version="0.1.2--0039-develop" /p:AssemblyVersion="0.1" /p:FileVersion="0.0.24419.16421" /p:InformationalVersion="0.1.2--ci.39.develop (0.1.2--0039-develop) - SHA1: e2fb1bcb1a4ce90cef54cee84f03c1171862b6d1 - CommitDate: 2019-01-16 10:04:19Z"

| | | <StdOut> Microsoft (R) Build Engine version 15.8.169+g1ccb72aefa pour .NET Core

| | | <StdOut> Copyright (C) Microsoft Corporation. Tous droits réservés.

| | | <StdOut>

| | | <StdOut> Restauration des packages pour C:\Dev\……(truncated)……\CK.UnitsOfMeasure.csproj...

| | | <StdOut> Restauration des packages pour C:\Dev\……(truncated)……\CK.UnitsOfMeasure.Tests.csproj...

| | | <StdOut> Restauration effectuée dans 242,57 ms pour ……(truncated)……CK.UnitsOfMeasure.Tests.csproj.

| | | <StdOut> Restauration effectuée dans 237,88 ms pour ……(truncated)……CK.UnitsOfMeasure.csproj.

| | | <StdOut> CK.UnitsOfMeasure ->

| | | C:\Dev\CKEnvTest\A2Test\……(truncated)……\bin\Debug\netstandard2.0\CK.UnitsOfMeasure.dll

| | | <StdOut> CK.UnitsOfMeasure.Tests ->

| | | C:\Dev\CKEnvTest\A2Test\……(truncated)……\net461\CK.UnitsOfMeasure.Tests.dll

| | | <StdOut>

| | | <StdOut> La génération a réussi.

| | | <StdOut> 0 Avertissement(s)

| | | <StdOut> 0 Erreur(s)

| | | <StdOut>

| | | <StdOut> Temps écoulé 00:00:02.67

| | | <StdOut> Deleting file ……(truncated)……/CK-UnitsOfMeasure.slna76d2a8581a84cdd90652003608015c4.sln

| | | <StdOut> Finished executing task: Build

| | | <StdOut> Run Unit Tests? (Y/N): Y

| | | <StdOut> Mode -autointeraction (and no command line -RunUnitTests="value" argument found):

| | | automatically answer with the first choice: Y.

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Unit-Testing

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Unit-Testing

| | | <StdOut> Testing via NUnit (net461): ……(truncated)……/bin/Debug/net461/CK.UnitsOfMeasure.Tests.dll

| | | <StdOut> Executing: "C:/Users/olivi/.nuget/packages/nunit.runners.net4/2.6.4/tools/nunit-console.exe" "C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/Tests/CK.UnitsOfMeasure.Tests/bin/Debug/net461/CK.UnitsOfMeasure.Tests.dll" "-framework:v4.5" "-result:C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/Tests/CK.UnitsOfMeasure.Tests/TestResult.Net461.xml"

| | | <StdOut> NUnit-Console version 2.6.4.14350

| | | <StdOut> Copyright (C) 2002-2012 Charlie Poole.

| | | <StdOut> Copyright (C) 2002-2004 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov.

| | | <StdOut> Copyright (C) 2000-2002 Philip Craig.

| | | <StdOut> All Rights Reserved.

| | | <StdOut>

| | | <StdOut> Runtime Environment -

| | | <StdOut> OS Version: Microsoft Windows NT 6.2.9200.0

| | | <StdOut> CLR Version: 4.0.30319.42000 ( Net 4.5 )

| | | <StdOut>

| | | <StdOut> ProcessModel: Default DomainUsage: Single

| | | <StdOut> Execution Runtime: v4.5

| | | <StdOut> ...........................................................

| | | <StdOut> Tests run: 59, Errors: 0, Failures: 0, Inconclusive: 0, Time: 0,60041188716166 seconds

| | | <StdOut> Not run: 0, Invalid: 0, Ignored: 0, Skipped: 0

| | | <StdOut>

| | | <StdOut> Finished executing task: Unit-Testing

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Create-NuGet-Packages

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Create-NuGet-Packages

| | | <StdOut> C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/CK.UnitsOfMeasure

| | | <StdOut> Executing: "C:/Program Files/dotnet/dotnet.exe" pack "C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/CK.UnitsOfMeasure" --output "C:/Dev/CKEnvTest/A2Test/Projects/CK-UnitsOfMeasure/CodeCakeBuilder/Releases" --no-build --include-symbols --configuration Debug /p:CakeBuild="true" /p:Version="0.1.2--0039-develop" /p:AssemblyVersion="0.1" /p:FileVersion="0.0.24419.16421" /p:InformationalVersion="0.1.2--ci.39.develop (0.1.2--0039-develop) - SHA1: e2fb1bcb1a4ce90cef54cee84f03c1171862b6d1 - CommitDate: 2019-01-16 10:04:19Z" /p:IsPackable=true

| | | <StdOut> Microsoft (R) Build Engine version 15.8.169+g1ccb72aefa pour .NET Core

| | | <StdOut> Copyright (C) Microsoft Corporation. Tous droits réservés.

| | | <StdOut>

| | | <StdOut> Successfully created package

| | | '……(truncated)……/Releases\CK.UnitsOfMeasure.0.1.2--0039-develop.nupkg'.

| | | <StdOut> Successfully created package

| | | '……(truncated)……/Releases\CK.UnitsOfMeasure.0.1.2--0039-develop.symbols.nupkg'.

| | | <StdOut> Finished executing task: Create-NuGet-Packages

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Push-NuGet-Packages

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Push-NuGet-Packages

| | | <StdOut> Pushing packages to 'LocalFeed-Develop' => 'C:\Dev\CKEnvTest\A2Test\LocalFeed\CI'.

| | | <StdOut> NuGet: Pushing CK.UnitsOfMeasure.0.1.2--0039-develop.nupkg

| | | to 'C:\Dev\CKEnvTest\A2Test\LocalFeed\CI'...

| | | <StdOut> NuGet: Your package was pushed.

| | | <StdOut> Finished executing task: Push-NuGet-Packages

| | | <StdOut>

| | | <StdOut> ========================================

| | | <StdOut> Default

| | | <StdOut> ========================================

| | | <StdOut> Executing task: Default

| | | <StdOut> Finished executing task: Default

| | | <StdOut>

| | | <StdOut> Task Duration

| | | <StdOut> --------------------------------------------------

| | | <StdOut> Check-Repository 00:00:00.2499200

| | | <StdOut> Clean 00:00:00.1044872

| | | <StdOut> Build 00:00:26.5103665

| | | <StdOut> Unit-Testing 00:00:01.8448591

| | | <StdOut> Create-NuGet-Packages 00:00:01.9022725

| | | <StdOut> Push-NuGet-Packages 00:00:00.2356985

| | | <StdOut> Default 00:00:00.0004319

| | | <StdOut> --------------------------------------------------

| | | <StdOut> Total: 00:00:30.8480357

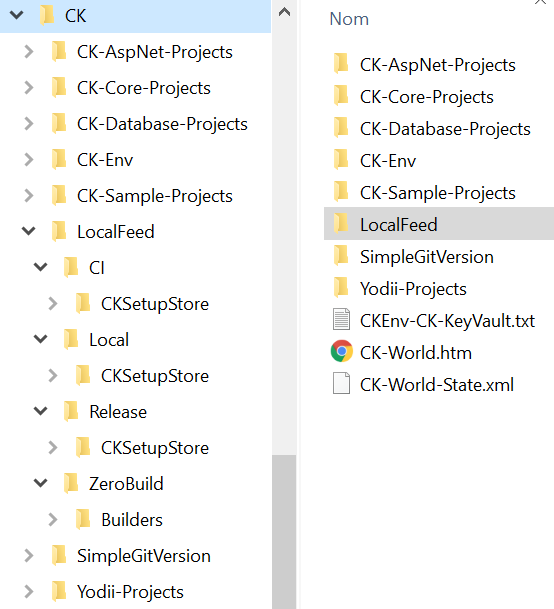
CodeCakeBuilder is launched with parameters that drives its behavior. The Build script MUST be aware of this and follow these parameters conventions.

| > Info: Reset --hard changes in 'Projects\CK-UnitsOfMeasure' (branch 'develop').

The last step is to reset the working folder.

## The LocalFeed folder

When a world is checked out, a LocalFeed folder is created at the root of the World. Below is a view of the CK-World folder:

The sub folders “CI”, “Local” and “Release” are local NuGet feeds. This is where builds produce their packages. Each of them also contains a local CKSetup store.

The “ZeroBuild” folder contains everything that is required to support build across the whole stack:

“ZeroBuild/Builders” folder contains all the CodeCakeBuilder projects, published in netcoreapp2.1 with a “0.0.0-0” version. This version is the ZeroVersion (this is the absolute minimal SemVer 2.0 valid version).

These CodeCakeBuilder projects CAN depend on packages that are produced by the world and this creates a rather complex situation: we have an egg-and-chicken problem.

To solve this egg-and-chicken problem, the “ZeroBuild” folder is also a NuGet local feed that contains all the NuGet packages required to build… the CodeCakeBuilers.

## The ZeroBuild command

This command compiles the CodeCakeBuilder projects and all their dependencies based on the current source code in the different Solutions. Once all these projects are compiled (and published in netcoreapp2.1) they are ran directly instead of using the “dotnet run -project CodeCakeBuilder” that compiles and runs the project. This has (at least) 2 advantages:

1. Compilation of the CodeCakeBuilder projects is done once for all.
2. Regardless of the different Solution states and dependencies among Solutions, the “build project” is based on the current source code and can always be compiled.

The latter is the most important since it solves the chicken-and-egg issue of cyclic dependencies by totally ignoring the Solutions structure.

To achieve this, CK-Env resolves the dependency graph of CodeCakeBuilder projects and their dependencies (this graph is by design acyclic with the CodeCakeBuilder projects having no predecessors: Rank = 0).

CK-Env builds, packages and publishes the dependent project as NuGet packages with a ZeroVersion into “LocalFeed/ZeroBuild” and build and publish in netcoreapp2.1 the CokeCakeBuilder executables into “LocalFeed/ZeroBuild/Builders”.

During this process, all the .csproj files involved are globally modified to reference the package “0.0.0-0” version package for all their dependencies (that belong to the World), including their Project references, when the dependency is produced by their own Solution, are transformed into ZeroVersion Package references.

Once done, CK-Env tracks any further builds that do not modify the build projects and updates a cache of branch SHA1 to avoid recompiling these build projects for nothing.

The good news is that you should barely directly use the ZeroBuild: this is done and maintained automatically by global build commands. The command is nevertheless exposed.

To understand it, it is easier to first look at a run where nothing is done because the build projects are up to date. The dump below is on a subset of CK-World with only 8 repositories:

> World: A1Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Zero\*\*

Confirm execution of:

World\ZeroBuildProjects

Y/N?

Y> Trace: Executing World\ZeroBuildProjects.

| > Info: Fixing Build projects.

| | > Info: Computing SolutionDependencyContext for branch develop.

| | > Info: Building ZeroVersion projects.

| | | - Info: File 'C:\Dev\CKEnvTest\A1Test\LocalFeed\ZeroBuild\CacheZeroVersion.txt' contains 13 entries.

| | | > Trace: Resolving drivers and reading Sha signatures.

| | | > Trace: Analysing dependencies.

| | | | > Info: CK.Text <= (no dependency).

| | | | < Project 'CK.Text' is up to date. Build skipped.

| | | | > Info: Code.Cake <= (no dependency).

| | | | < Project 'Code.Cake' is up to date. Build skipped.

| | | | > Info: CSemVer <= (no dependency).

| | | | < Project 'CSemVer' is up to date. Build skipped.

| | | | > Info: SimpleGitVersion.Core <= CSemVer.

| | | | < Project 'SimpleGitVersion.Core' is up to date. Build skipped.

| | | | > Info: SimpleGitVersion.Cake <= SimpleGitVersion.Core, CSemVer, Code.Cake.

| | | | < Project 'SimpleGitVersion.Cake' is up to date. Build skipped.

| | | | > Info: CK-ActivityMonitor/CodeCakeBuilder <= CK.Text, Code.Cake, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CK-ActivityMonitor/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CK-Core/CodeCakeBuilder <= CK.Text, Code.Cake, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CK-Core/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CK-Monitoring/CodeCakeBuilder <= CK.Text, Code.Cake, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CK-Monitoring/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CK-Text/CodeCakeBuilder <= CK.Text, Code.Cake, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CK-Text/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CK-UnitsOfMeasure/CodeCakeBuilder <= CK.Text, Code.Cake,

| | | | SimpleGitVersion.Cake, SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CK-UnitsOfMeasure/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CodeCake/CodeCakeBuilder <= Code.Cake, CK.Text, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CodeCake/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CSemVer-Net/CodeCakeBuilder <= Code.Cake, CK.Text, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | < Project 'CSemVer-Net/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: SGV-Net/CodeCakeBuilder <= SimpleGitVersion.Cake, SimpleGitVersion.Core, CSemVer,

| | | | Code.Cake, CK.Text.

| | | | < Project 'SGV-Net/CodeCakeBuilder' is up to date. Build skipped.

| | | - Info: Nothing to build. Build projects are up-to-date.

| < Success => WorkStatus = Idle, GlobalGitStatus = Develop

This is fast since nothing is done. Artificially touching the SimpleGitVersion.Core folder (I just created a Dirty.txt file and committed it), triggers more work:

> World: A1Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Zero\*\*

Confirm execution of:

World/ZeroBuildProjects

Y/N?

Y> Trace: Executing World/ZeroBuildProjects.

| > Info: Fixing Build projects.

| | > Info: Computing SolutionDependencyContext for branch develop.

| | > Info: Building ZeroVersion projects.

| | | - Info: File 'C:/Dev/CKEnvTest/A1Test/LocalFeed/ZeroBuild/CacheZeroVersion.txt' contains 13 entries.

| | | > Trace: Resolving drivers and reading Sha signatures.

| | | > Trace: Analysing dependencies.

| | | | > Info: CK.Text <= (no dependency).

| | | | < Project 'CK.Text' is up to date. Build skipped.

| | | | > Info: Code.Cake <= (no dependency).

| | | | < Project 'Code.Cake' is up to date. Build skipped.

| | | | > Info: CSemVer <= (no dependency).

| | | | < Project 'CSemVer' is up to date. Build skipped.

| | | | > Info: SimpleGitVersion.Core <= CSemVer.

| | | | | - Info: ReasonToBuild#2: Current Sha signature differs from the cached ones.

| | | | > Info: SimpleGitVersion.Cake <= SimpleGitVersion.Core, CSemVer, Code.Cake.

| | | | | - Info: ReasonToBuild#3: Rebuild dependencies are SimpleGitVersion.Core.

| | | | > Info: CK-ActivityMonitor/CodeCakeBuilder <= CK.Text, Code.Cake, SimpleGitVersion.Cake,

| | | | SimpleGitVersion.Core, CSemVer.

| | | | | - Info: ReasonToBuild#3: Rebuild dependencies are SimpleGitVersion.Core, SimpleGitVersion.Cake.

| | | | > Info: CK-Core/CodeCakeBuilder <= CK.Text, Code.Cake, SimpleGitVersion.Cake, SimpleGitVersion.Core,

| | | | CSemVer.

| | | | | - Info: ReasonToBuild#3: Rebuild dependencies are SimpleGitVersion.Core, SimpleGitVersion.Cake.

| | | | > Info: ……(same for CK-Monitoring/CodeCakeBuilder, CK-Text/CodeCakeBuilder,

| | | | CK-UnitsOfMeasure/CodeCakeBuilder, CodeCake/CodeCakeBuilder, CSemVer-Net/CodeCakeBuilder,

| | | | and SGV-Net/CodeCakeBuilder)……

| | | > Trace: Creating protected scopes.

| | | > Trace: Build/Publish 10 build projects: SimpleGitVersion.Core, SimpleGitVersion.Cake, CK-ActivityMonitor/CodeCakeBuilder, CK-Core/CodeCakeBuilder, CK-Monitoring/CodeCakeBuilder, CK-Text/CodeCakeBuilder, CK-UnitsOfMeasure/CodeCakeBuilder, CodeCake/CodeCakeBuilder, CSemVer-Net/CodeCakeBuilder, SGV-Net/CodeCakeBuilder

| | | | > Info: Publishing SimpleGitVersion.Core.

| | | | | - Trace: Primary solution 'Projects/SGV-Net/branches/develop/develop.sln' refreshed

| | | | | due to successful reload.

| | | | | - Info: Deleting C:/Users/olivi/.nuget/packages/SimpleGitVersion.Core/0.0.0-0.

| | | | | Removed SimpleGitVersion.Core package in version 0.0.0-0 from local NuGet cache.

| | | | | - Trace: Update in develop/SGV-Net/SimpleGitVersion.Core: CSemVer from 5.0.0 to 0.0.0-0.

| | | | | > Info: Replacing Projects/………/SimpleGitVersion.Core/SimpleGitVersion.Core.csproj.

| | | | | - Info: Deleting C:\Dev\CKEnvTest\A1Test\Projects\SGV-Net\SimpleGitVersion.Core\bin.

| | | | | Deleting C:\Dev\CKEnvTest\A1Test\Projects\SGV-Net\SimpleGitVersion.Core\obj.

| | | | | > Info: Replacing Projects/SGV-Net/branches/develop/NuGet.config.

| | | | | > Trace: dotnet pack --output "C:/Dev/CKEnvTest/A1Test/LocalFeed/ZeroBuild" --no-dependencies --configuration Debug /p:Version="0.0.0-0" /p:AssemblyVersion="0.0.0" /p:FileVersion="0.0.0.0" /p:InformationalVersion="0.0.0-0 (0.0.0-0) - SHA1: 0000000000000000000000000000000000000000 - CommitDate: 0001-01-01 00:00:00Z"

| | | | | | - Info: <StdOut> Microsoft (R) Build Engine version 15.9.20+g88f5fadfbe pour .NET Core

| | | | | | <StdOut> Copyright (C) Microsoft Corporation. Tous droits réservés.

| | | | | | <StdOut>

| | | | | | <StdOut> Restauration des packages pour …\SimpleGitVersion.Core.csproj...

| | | | | | <StdOut> Génération du fichier MSBuild …\SimpleGitVersion.Core.csproj.nuget.g.props.

| | | | | | <StdOut> Génération du fichier MSBuild …\SimpleGitVersion.Core.csproj.nuget.g.targets.

| | | | | | <StdOut> Restauration effectuée dans 226,08 ms pour …\SimpleGitVersion.Core.csproj.

| | | | | | <StdOut> SimpleGitVersion.Core -> …\SimpleGitVersion.Core.dll

| | | | | | <StdOut> SimpleGitVersion.Core -> …\SimpleGitVersion.Core.dll

| | | | | | <StdOut> Successfully created package '…\ZeroBuild\SimpleGitVersion.Core.0.0.0-0.nupkg'.

| | | | | - Trace: Added new Shalias for SimpleGitVersion.Core.

| | | | < Success.

| | | | > Info: Publishing SimpleGitVersion.Cake.

| | | | | - Info: Deleting C:/Users/olivi/.nuget/packages/SimpleGitVersion.Cake/0.0.0-0.

| | | | | Removed SimpleGitVersion.Cake package in version 0.0.0-0 from local NuGet cache.

| | | | | Removing Project references:

| | | | | <ProjectReference Include="…/SimpleGitVersion.Core/SimpleGitVersion.Core.csproj" />

| | | | | - Trace: Added unconditional package reference

| | | | | SimpleGitVersion.Core -> 0.0.0-0 for develop/SGV-Net/SimpleGitVersion.Cake.

| | | | | Update in develop/SGV-Net/SimpleGitVersion.Cake: Code.Cake from 1.0.1 to 0.0.0-0.

| | | | | > Info: Replacing Projects/………/SimpleGitVersion.Cake/SimpleGitVersion.Cake.csproj.

| | | | | - Info: Deleting C:\Dev\CKEnvTest\A1Test\Projects\SGV-Net\SimpleGitVersion.Cake\bin.

| | | | | Deleting C:\Dev\CKEnvTest\A1Test\Projects\SGV-Net\SimpleGitVersion.Cake\obj.

| | | | | > Info: Replacing Projects/SGV-Net/branches/develop/NuGet.config.

| | | | | > Trace: dotnet pack --output "C:/Dev/CKEnvTest/A1Test/LocalFeed/ZeroBuild" --no-dependencies --configuration Debug /p:Version="0.0.0-0" /p:AssemblyVersion="0.0.0" /p:FileVersion="0.0.0.0" /p:InformationalVersion="0.0.0-0 (0.0.0-0) - SHA1: 0000000000000000000000000000000000000000 - CommitDate: 0001-01-01 00:00:00Z"

| | | | | | - Info: <StdOut> Microsoft (R) Build Engine version 15.9.20+g88f5fadfbe pour .NET Core

| | | | | | <StdOut> Copyright (C) Microsoft Corporation. Tous droits réservés.

| | | | | | <StdOut>

| | | | | | <StdOut> Restauration des packages pour ………\SimpleGitVersion.Cake.csproj...

| | | | | | <StdOut> Installation de SimpleGitVersion.Core 0.0.0-0.

| | | | | | <StdOut> Génération du fichier MSBuild …\SimpleGitVersion.Cake.csproj.nuget.g.props.

| | | | | | <StdOut> Génération du fichier MSBuild …\SimpleGitVersion.Cake.csproj.nuget.g.targets.

| | | | | | <StdOut> Restauration effectuée dans 778,79 ms pour …\SimpleGitVersion.Cake.csproj.

| | | | | | <StdOut> SimpleGitVersion.Cake -> …\netstandard2.0\SimpleGitVersion.Cake.dll

| | | | | | <StdOut> Successfully created package

| | | | | | 'C:/…/A1Test/LocalFeed/ZeroBuild\SimpleGitVersion.Cake.0.0.0-0.nupkg'.

| | | | < Success.

| | | | > Info: Building CK-ActivityMonitor/CodeCakeBuilder.

| | | | | - Trace: Primary solution 'Projects/CK-ActivityMonitor/branches/develop/develop.sln'

| | | | | refreshed due to successful reload.

| | | | | Update in develop/CK-ActivityMonitor/CodeCakeBuilder: CK.Text from 8.0.2 to 0.0.0-0.

| | | | | Update in develop/CK-ActivityMonitor/CodeCakeBuilder: Code.Cake from 1.0.1 to 0.0.0-0.

| | | | | Update in …/CK-ActivityMonitor/CodeCakeBuilder: SimpleGitVersion.Cake from 0.37.3 to 0.0.0-0.

| | | | | > Info: Replacing Projects/CK-ActivityMonitor/branches/develop/CodeCakeBuilder/CodeCakeBuilder.csproj.

| | | | | - Info: Deleting C:\Dev\CKEnvTest\A1Test\Projects\CK-ActivityMonitor\CodeCakeBuilder\bin.

| | | | | Deleting C:\Dev\CKEnvTest\A1Test\Projects\CK-ActivityMonitor\CodeCakeBuilder\obj.

| | | | | > Info: Replacing Projects/CK-ActivityMonitor/branches/develop/NuGet.config.

| | | | | > Trace: dotnet publish --output "C:/Dev/CKEnvTest/A1Test/LocalFeed/ZeroBuild/Builders/CK-ActivityMonitor" --no-dependencies --configuration Debug /p:Version="0.0.0-0" /p:AssemblyVersion="0.0.0" /p:FileVersion="0.0.0.0" /p:InformationalVersion="0.0.0-0 (0.0.0-0) - SHA1: 0000000000000000000000000000000000000000 - CommitDate: 0001-01-01 00:00:00Z"

| | | | | | - Info: <StdOut> Microsoft (R) Build Engine version 15.9.20+g88f5fadfbe pour .NET Core

| | | | | | <StdOut> Copyright (C) Microsoft Corporation. Tous droits réservés.

| | | | | | <StdOut>

| | | | | | <StdOut> Restauration des packages pour …\CodeCakeBuilder\CodeCakeBuilder.csproj...

| | | | | | <StdOut> Installation de SimpleGitVersion.Cake 0.0.0-0.

| | | | | | <StdOut> Génération du fichier MSBuild …\obj\CodeCakeBuilder.csproj.nuget.g.props.

| | | | | | <StdOut> Génération du fichier MSBuild …\obj\CodeCakeBuilder.csproj.nuget.g.targets.

| | | | | | <StdOut> Restauration effectuée dans 1,26 sec pour …\CodeCakeBuilder.csproj.

| | | | | | <StdOut> CodeCakeBuilder -> …\Debug\netcoreapp2.1\CodeCakeBuilder.dll

| | | | | | <StdOut> CodeCakeBuilder -> …\A1Test\LocalFeed\ZeroBuild\Builders\CK-ActivityMonitor\

| | | | < Success.

| | | | ……(same for all the other CodeCakeBuilder)……

| | | - Trace: Closing protected scopes.

| | | Saving 13 entries in file 'C:/Dev/CKEnvTest/A1Test/LocalFeed/ZeroBuild/CacheZeroVersion.txt'.

| | | > Info: Reloading solutions for branch develop.

| < Success => WorkStatus = Idle, GlobalGitStatus = Develop

Only what needs to be built is built among the Build projects and recall that you should barely directly use the ZeroBuild: this is done and maintained automatically by global build commands.

# The Workflows

## Checking out a World

This is the first and simplest action. As soon as a World (for instance the “Customer” world) is selected (see above), the target directory is created as well as all folders that structure the solutions, Git repositories are cloned (from their urls) and checked out on the “develop” branch.

The World folder contains 3 files that are initialized by this process:

* CK-World.htm

It is currently only a marker file (it is empty but it is planned to be an actual html file with information regarding the world and, ideally, tools or actions). This file marks the World root and supports the “Reproducible Builds” (see <http://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html>).

* Customer-World-State.xml  
  This file (that should not be manually edited) is the local state of the world. It contains data like Build results with the versions and other required state and information.
* CKEnv-Customer-KeyVault.txt

This file contains the secrets that required by the World. It is password protected and by opening it once (with the password), all required secrets are available to CK-Env whenever it needs them without further user interactions.

When a new (not yet known) secret is required, the user is prompted for it and the secret is stored in the vault.

This is an easy way to provide the full context of a complex customer code base to a new developer without leaking any secret. The new developer must either:

* Obtain an existing CKEnv-XXX-KeyVault.txt file with its password (that must be saved in the developer’s KeePass, PasswordSafe or any other secret manager tool).
* Know (or obtain) the different secrets that are required by the solutions (API keys and other secrets) and iteratively build its own local vault.

## Repositories status and Global state overview

A checked-out world can be in multiple states. Most of the possible actions require that:

* All repositories are on the same branch: either “develop” or “develop-local”.
* All working folder repositories must be clean: the files are the same as the committed ones (no new, deleted or modified files).
* The World is in a specific state, the default one being “Idle”.

This state is defined by 2 enumerations:

* GlobalWorkStatus

Defines the current state of a World in terms of current operation. The enumeration below lists the different operational states.

* StandardGitStatus

Defines the 3 standard git status that applies to one or multiple repositories.

A repository (or a world of repositories) is either on "Develop" (the default), on "Local" or the "Master". Any other configurations results in a "Unknown" status. When a world or a repository is in this unknown status some operations cannot be done.

|  |  |
| --- | --- |
| public enum GlobalWorkStatus  {  /// No operation are currently in progress.  /// The StandardGitStatus reflects the repository status.  Idle,  /// Switch from develop to local branches.  SwitchingToLocal,  /// Switching back to develop.  SwitchingToDevelop,  /// Releasing the stack. Once done, the release can be  /// published or canceled.  Releasing,  /// Waiting for release to be canceled or publisehd.  WaitingReleaseConfirmation,  /// Release is being canceled.  CancellingRelease,  /// Release is being published.  PublishingRelease,  } | [Flags]  public enum StandardGitStatus  {  Unknown = 0,  Local = 1,  Develop = 2,  Master = 4,  DevelopOrLocal = Local | Develop,  MasterOrDevelop = Master | Develop,  MasterOrLocal = Master | Local,  KnownBranches = Master | Develop | Local  } |

CKli displays these statuses right after the load of a World and at any moment by entering an empty line at the prompt:

> World: CK - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

>

> Trace: Executing World\Initialize.

| > Info: World Status.

| | ………(truncated)………

| | > Info: CK-Core-Projects\CK-ActivityMonitor - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | > Info: CK-Core-Projects\CK-Auth-Abstractions - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | > Info: CK-Core-Projects\CK-Core - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | ………(truncated)………

| < 0 dirty (out of 32).

| - Info: All 32 git folders are on 'develop-local' branch.

< Success => WorkStatus = Idle, GlobalGitStatus = Local

Any working folder that has local modifications is displayed:

| > Info: World Status.

| | ………(truncated)………

| | > Info: CK-Core-Projects\CK-Auth-Abstractions - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | > Info: CK-Core-Projects\CK-Core - branch: develop-local.

| | | - Error: Repository 'CK-Core-Projects\CK-Core' has uncommited changes (develop-local).

| | < Dirty. (11 plugins)

| | > Info: CK-Core-Projects\CK-SqlServer - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | ………(truncated)………

| | > Info: SimpleGitVersion\SGV-Net - branch: develop-local.

| | < Up-to-date. (11 plugins)

| < 1 dirty (out of 32).

| - Info: Dirty: CK-Core-Projects\CK-Core

| All 32 git folders are on 'develop-local' branch.

< Failed => WorkStatus = Idle, GlobalGitStatus = Local

And when current branches differ:

| > Info: World Status.

| | > Info: CK-AspNet-Projects\CK-AspNet - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | ………(truncated)………

| | > Info: CK-Core-Projects\CK-Core - branch: develop.

| | | - Error: Repository 'CK-Core-Projects\CK-Core' has uncommited changes (develop).

| | < Dirty. (11 plugins)

| | > Info: CK-Core-Projects\CK-SqlServer - branch: transaction-support.

| | < Up-to-date. (11 plugins)

| | > Info: CK-Core-Projects\CK-Monitoring - branch: develop.

| | < Up-to-date. (11 plugins)

| | > Info: CK-Core-Projects\CK-Reflection - branch: develop-local.

| | < Up-to-date. (11 plugins)

| | ………(truncated)………

| < 1 dirty (out of 32).

| - Info: Dirty: CK-Core-Projects\CK-Core

| > Info: 3 different branches:

| | - Info: Branch 'develop-local': CK-AspNet-Projects\CK-AspNet, CK-AspNet-Projects\CK-AspNet-Auth, CK-AspNet-Projects\CK-AspNet-Tester, CK-Core-Projects\CK-ActivityMonitor, CK-Core-Projects\CK-Auth-Abstractions,

………(truncated)……… CK-Core-Projects\CK-Reflection, CK-Core-Projects\CK-Testing, SimpleGitVersion\SGV-Net

| | Branch 'develop': CK-Core-Projects\CK-Core, CK-Core-Projects\CK-Monitoring

| | Branch 'transaction-support': CK-Core-Projects\CK-SqlServer

< Failed => WorkStatus = Idle, GlobalGitStatus = Unknown

## Homogenization of configurations: ApplySettings

Once a world is checked out, the ApplyingSettings commands are in charge of the configuration of the Solutions.

ApplySettings commands are argument-free methods that are typically implemented by plugins and drive the content of some important files and folders in a Solution. Thanks to the “globbing”, these commands can be applied globally or locally. Applying all the available settings to one project for instance:

> run \*\*CK-Core/\*\*ApplySettings

Confirm execution of:

CK-Core-Projects\CK-Core\branches\develop-local\CodecakeBuilder\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\CodecakeBuilder\CodeCakeBuilder.csproj\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\CodecakeBuilder\CodeCakeBuilderKeyVault.txt\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\Common\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\RemoteStore.TestHelper.config\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\RepositoryInfo.xml\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\NuGet.config\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\.gitignore\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\Appveyor.yml\ApplySettings

Y/N?

Or, applying only the AppVeyor.yml settings to all the solutions of the CK-World:

> World: CK - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Appveyor\*\*Apply\*

Confirm execution of:

CK-AspNet-Projects\CK-AspNet\branches\develop-local\Appveyor.yml\ApplySettings

CK-AspNet-Projects\CK-AspNet-Auth\branches\develop-local\Appveyor.yml\ApplySettings

CK-AspNet-Projects\CK-AspNet-Tester\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-ActivityMonitor\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-Auth-Abstractions\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-Core\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-SqlServer\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-Monitoring\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-Reflection\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-Testing\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-Text\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-MicroBenchmark\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-UnitsOfMeasure\branches\develop-local\Appveyor.yml\ApplySettings

CK-Core-Projects\CK-WeakAssemblyNameResolver\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-CodeGen\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-Database\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-DB\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-DB-GitHub\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-DB-Actor-ActorEMail\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-DB-SqlCKTrait\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-DB-User-SimpleInvitation\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-DB-User-UserPassword\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-Setup\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-Setup-Dependency\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CKSetupRemoteStore\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-Sqlite\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-SqlServer-Parser\branches\develop-local\Appveyor.yml\ApplySettings

CK-Database-Projects\CK-SqlServer-Parser-Model\branches\develop-local\Appveyor.yml\ApplySettings

Yodii-Projects\Yodii-Script\branches\develop-local\Appveyor.yml\ApplySettings

SimpleGitVersion\CodeCake\branches\develop-local\Appveyor.yml\ApplySettings

SimpleGitVersion\CSemVer-Net\branches\develop-local\Appveyor.yml\ApplySettings

SimpleGitVersion\SGV-Net\branches\develop-local\Appveyor.yml\ApplySettings

Y/N?

Cuurent ApplySettings implementations are what they are: they do their current job but:

* it could certainly be better
* and a LOT of other ApplySettings should be developed, like, for instance, handling the project license: an attribute License="…" in the <SolutionSettings> should ensure that the proper license is defined in the csproj (and everywhere it should be).

Applying settings does not commit anything, it just alters the files/folder of the currently checked out branch.

## Global operations on the Git repositories

Committing the changes globally is simple:

> World: A1Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Commit

[required] - commitMessage: Applied new settings (changed the world).

[default value: False] - amendIfPossible:

Confirm execution of:

Projects\CK-UnitsOfMeasure\Commit

Projects\CodeCake\Commit

Projects\CK-ActivityMonitor\Commit

Projects\CK-Monitoring\Commit

Projects\CK-Core\Commit

Projects\SGV-Net\Commit

Projects\CK-Text\Commit

Projects\CSemVer-Net\Commit

Projects\CK-Setup\Commit

With payload:

[required] - commitMessage - Value = 'Applied new seetings (changed the world).'

[default value: False] - amendIfPossible - <use default value: False>

Y/N?

Y> Trace: Executing Projects\CK-UnitsOfMeasure\Commit.

| > Info: Committing changes in 'Projects\CK-UnitsOfMeasure' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CodeCake\Commit.

| > Info: Committing changes in 'Projects\CodeCake' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CK-ActivityMonitor\Commit.

| > Info: Committing changes in 'Projects\CK-ActivityMonitor' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CK-Monitoring\Commit.

| > Info: Committing changes in 'Projects\CK-Monitoring' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CK-Core\Commit.

| > Info: Committing changes in 'Projects\CK-Core' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\SGV-Net\Commit.

| > Info: Committing changes in 'Projects\SGV-Net' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CK-Text\Commit.

| > Info: Committing changes in 'Projects\CK-Text' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CSemVer-Net\Commit.

| > Info: Committing changes in 'Projects\CSemVer-Net' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

> Trace: Executing Projects\CK-Setup\Commit.

| > Info: Committing changes in 'Projects\CK-Setup' (branch 'develop').

| | - Info: Working Folder is dirty. Committing changes.

When committing, it is possible to amend the current commit only if the local branch is ahead of the remote one.

Other supported operations are:

1. FetchBranches

To fetch all remote branches from ‘origin’ only (or all remotes if there is multiple remotes):

> run \*\*\*Measure\*\*Fetch\*\*

[default value: True] - originOnly:

Confirm execution of:

Projects\CK-UnitsOfMeasure\FetchBranches

With payload:

[default value: True] - originOnly - <use default value: True>

Y/N?

Y> Trace: Executing Projects\CK-UnitsOfMeasure\FetchBranches.

| > Info: Fetching origin in repository 'Projects\CK-UnitsOfMeasure'.

| | - Info: Fetching remote 'origin'.

1. Push  
   To push the current branch to the ‘origin’ remote.

> run \*\*\*Measure\*\*Push

Confirm execution of:

Projects\CK-UnitsOfMeasure\Push

Y/N?

(I won’t confirm this one 😊)

1. Pull

To pull-merge the current branch (this executes a FetchBranches first).

> run \*\*\*Measure\*\*Pull

Confirm execution of:

Projects\CK-UnitsOfMeasure\branches\develop\SolutionDriver\Pull

Y/N?

Y> Trace: Executing Projects\CK-UnitsOfMeasure\branches\develop\SolutionDriver\Pull.

| > Info: Pulling branch 'develop' in 'Projects\CK-UnitsOfMeasure'.

| | > Info: Fetching origin in repository 'Projects\CK-UnitsOfMeasure'.

| | | - Info: Fetching remote 'origin'.

When pulling, if merge doesn’t succeed because of merge conflicts), the repository is left as-is and the merge must be manually done.

## CI Building globally: Introducing packages resolution and propagation

The “normal” state of a World is to be “Idle” in terms of operations and “Develop” regarding the Git status (all repositories are on the ‘develop’ branch). In this state the “AllBuild” command is available. This command handles a full build of the whole World, propagating package version dependencies across all the Solutions, and generating the full set of artifacts in CI build versions.

### “AllBuild” command overview

The 3 main steps of this command are:

1. Preparation
   * We ensure that ZeroBuild projects are up-to-date.
   * All the solutions are loaded and topologically sorted.
2. Prebuild: For each solution, in increasing rank order:
   * Project dependencies are upgraded (with the packages built and produced by the previous solutions).
   * A commit point is created with a message that describes the upgraded dependency versions.
   * Based on the maximal number of commit points between the current head and the last release, the CI build version number is computed and associated to the packages produced by the current solution.

Note that a commit is not necessarily created here if the dependencies have not changed (since working folder is up-to-date).

1. Build: For each solution, in increasing rank order:
   * We upgrade any dependencies of the Build projects (ie. the CodeCakeBuilder project).
   * We amend the current commit if it is possible.
     + If not (we are on a fresh check-out or the branch has been pushed and no upgrade has been done during the pre-build phase), this triggers a retry of the build process. This could be called an edge case and we won’t detail this more here.
   * Solution’s CodeCakeBuilder is run.
     + At its level, CodeCakeBuilder checks the packages with the version it has to build are not already present in its NuGet feeds. In such case build is skipped.

Before building a Solution, CKli displays the ordered list of Solutions with a star \* in front of the one being processed:

| | | - Info: -- Rank 0

| | | 0 - develop/CK-WeakAssemblyNameResolver

| | | 1 - develop/CodeCake

| | | 2 - develop/CK-AspNet-Tester

| | | 3 - develop/CK-Auth-Abstractions

| | | 4 - develop/CK-MicroBenchmark

| | | 5 - develop/CK-Reflection

| | | 6 - develop/CK-UnitsOfMeasure

| | | 7 - develop/CSemVer-Net

| | | 8 - develop/Yodii-Script

| | | -- Rank 1

| | | 9 - develop/CK-Text

| | | 10 - develop/SGV-Net

| | | -- Rank 2

| | | 11 - develop/CK-Core

| | | -- Rank 3

| | | 12 - develop/CK-ActivityMonitor

| | | -- Rank 4

| | | 13 - develop/CK-SqlServer-Parser-Model

| | | 14 - develop/CK-CodeGen

| | | 15 - develop/CK-Monitoring

| | | 16 - develop/CK-Setup-Dependency

| | | -- Rank 5

| | | 17 - develop/CK-AspNet

| | | 18 - develop/CK-Testing

| | | -- Rank 6

| | | 19 - develop/CK-SqlServer

| | | 20 - develop/CK-Setup

| | | 21 - develop/CK-SqlServer-Parser

| | | -- Rank 7

| | | 22 - develop/CKSetupRemoteStore

| | | \* 23 - develop/CK-Database

| | | -- Rank 8

| | | 24 - develop/CK-DB-SqlCKTrait

| | | 25 - develop/CK-Sqlite

| | | 26 - develop/CK-DB

| | | -- Rank 9

| | | 27 - develop/CK-DB-Actor-ActorEMail

| | | 28 - develop/CK-DB-User-SimpleInvitation

| | | 29 - develop/CK-DB-User-UserPassword

| | | 30 - develop/CK-DB-GitHub

| | | -- Rank 10

| | | 31 - develop/CK-AspNet-Auth

Once all is build (locally), the CI Build can be published.

### The “PublishCI” command

Once fully built, a CI build can be published thanks to the “PublishCI” command. This command is available as long as the system is in “Idle” state and the last build is a CI Build.

* NuGet packages are pushed to the remote feed after having checked for their existence.
* On Azure Feed, packages are added to the ‘CI’ view.
* CKSetup components that are new to the remote store are pushed into it.
* The “develop” branch is pushed to the remote repository.

This action is idempotent: publishing existing artifacts does not raise any error and, more importantly, if a run fails (lost of internet connection for example), it can safely be restarted until it succeeds.

## Working across the World: The ‘develop-local’ branch.

The ‘develop-local’ branch enables to locally work on the World. As its names states, it MUST NOT be pushed to any remote. From the “normal” state (“Idle”/ ”Develop”), one can switch the whole world to the ‘develop-local’ branch.

> World: A1Test - [run <globbed command name> | list [<globbed command name>] | cls | restart | exit]

> run \*\*Swi\*\*

Confirm execution of:

World/SwitchToLocal

Y/N?

Y> Trace: Executing World/SwitchToLocal.

| > Info: Switching to develop-local.

| | > Info: Switching 'Projects/CK-ActivityMonitor' to branch 'develop-local').

| | | > Info: Committing changes in 'Projects/CK-ActivityMonitor' (branch 'develop').

| | | < Working folder is up-to-date.

| | | - Info: Creating the develop-local.

| | | > Trace: Initializing plugins for 'Projects/CK-ActivityMonitor' branch 'develop-local'.

| | | > Trace: C:/Dev/…/Projects/CK-ActivityMonitor (develop-local).: Raising OnLocalBranchEntered event.

| | | | > Info: Replacing Projects/CK-ActivityMonitor/branches/develop-local/RepositoryInfo.xml.

| | | | > Info: Replacing Projects/CK-ActivityMonitor/branches/develop-local/NuGet.config.

| | | > Info: Amending Commit in 'Projects/CK-ActivityMonitor' (branch 'develop-local').

| | | | - Info: Working Folder is dirty. Committing changes.

| | > Info: Switching 'Projects/CK-Monitoring' to branch 'develop-local').

| | | > Info: Committing changes in 'Projects/CK-Monitoring' (branch 'develop').

| | | < Working folder is up-to-date.

| | | - Info: Creating the develop-local.

| | | > Trace: Initializing plugins for 'Projects/CK-Monitoring' branch 'develop-local'.

| | | > Trace: …CKEnvTest/A1Test/Projects/CK-Monitoring (develop-local).: Raising OnLocalBranchEntered event.

| | | | > Info: Replacing Projects/CK-Monitoring/branches/develop-local/RepositoryInfo.xml.

| | | | > Info: Replacing Projects/CK-Monitoring/branches/develop-local/NuGet.config.

| | | > Info: Amending Commit in 'Projects/CK-Monitoring' (branch 'develop-local').

| | | | - Info: Working Folder is dirty. Committing changes.

| | > Info: Computing SolutionDependencyContext for branch develop-local.

| | > Info: Building ZeroVersion projects.

| | | - Info: File 'C:/Dev/CKEnvTest/A1Test/LocalFeed/ZeroBuild/CacheZeroVersion.txt' contains 2 entries.

| | | > Trace: Resolving drivers and reading Sha signatures.

| | | > Trace: Analysing dependencies.

| | | | > Info: CK-ActivityMonitor/CodeCakeBuilder <= (no dependency).

| | | | < Project 'CK-ActivityMonitor/CodeCakeBuilder' is up to date. Build skipped.

| | | | > Info: CK-Monitoring/CodeCakeBuilder <= (no dependency).

| | | | < Project 'CK-Monitoring/CodeCakeBuilder' is up to date. Build skipped.

| | | - Info: Nothing to build. Build projects are up-to-date.

| < Success => WorkStatus = Idle, GlobalGitStatus = Local

When in ‘develop -local’, solutions files of all the working folders are modified:

1. NuGet.config is modified so that the 3 local feeds are available as sources:

<configuration>

<packageSources>

<clear />

<add key="Signature-OpenSource"

value="https://pkgs.dev.azure.com/Signature-OpenSource/\_packaging/Default/nuget/v3/index.json" />

<add key="LocalFeed-Local" value="C:\Dev\CK\LocalFeed\Local" />

<add key="LocalFeed-Develop" value="C:\Dev\CK\LocalFeed\CI" />

<add key="LocalFeed-Master" value="C:\Dev\CK\LocalFeed\Release" />

</packageSources>

<packageSourceCredentials>

<Signature-OpenSource>

<add key="Username" value="SignatureOpenSource" />

<add key="ClearTextPassword" value="c56geu5acyhfgeiqhdl3t2xo5xumlztnjf736zdbogyxtaulydlq" />

</Signature-OpenSource>

</packageSourceCredentials>

</configuration>

1. Repository.xml file is modified so that the branch uses CI Zero timed based versions:

<RepositoryInfo xmlns="http://csemver.org/schemas/2015">

<StartingVersionForCSemVer>v6.0.0-alpha</StartingVersionForCSemVer>

<Branches>

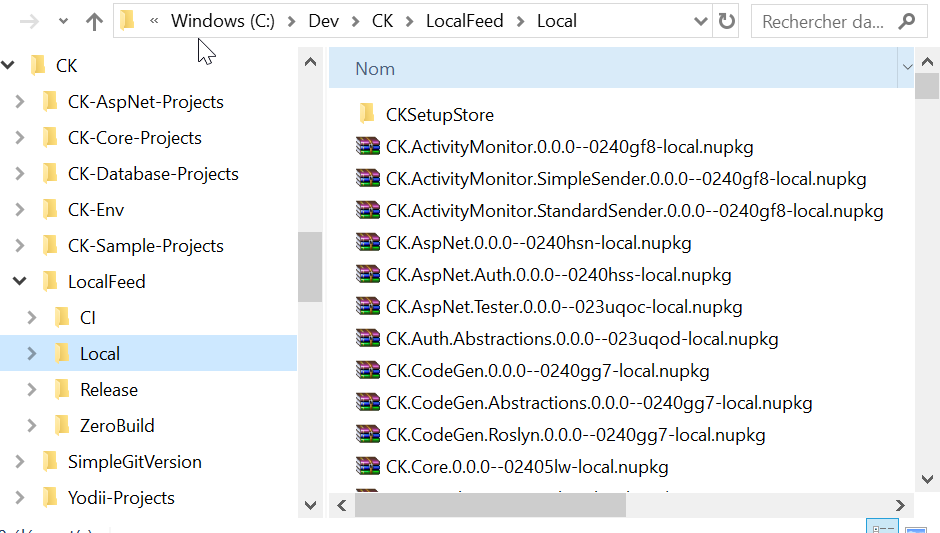
<Branch Name="develop" VersionName="develop" CIVersionMode="LastReleaseBased" />

<Branch Name="develop-local" VersionName="local" CIVersionMode="ZeroTimed" />

</Branches>

</RepositoryInfo>

In this mode, produced artifacts use ZeroTimed versions:



Once a World has switched to local, one can more easily work across solutions:

1. CodeCakeBuilder generates the artifacts only into the “LocalFeed/Local” folder
2. The NuGet.config file is plugged on this local feed.
3. The CKSetup stores are also rerouted to “LocalFeed/Local/CKSetupStore” folder.

The “AllBuild” command is available and does its job a little bit differently than when on “develop” branch because commits are always “amended” (it is the commit time that is used to compute the version number). This allows work in “develop-local” to avoid polluting the Git history as much as possible.

Working in this purely local branch is easier than in “develop” as soon as the job implies multiple solutions across the World. Moreover, since this branch is never pushed and uses ZeroTimed based version number, commits are amended by default: this enables “dirty cross solution fixes” to not pollute too much the published Git history (of course, commits can be created to record changes that should be visible).

One rule to follow: if there is any breaking change in projects that participates in the Build projects (projects that are ultimately used by at least one CodeCakeBuilder of the World), then propagating these changes across the Solutions SHOULD be done in “develop-local”.

## Releasing: From roadmap to publication

TBD

# A World is Multiple: Long-Term-Support branches

TBD

# Ensure publish configuration

You can configure the project to avoid errors like publishing by accident the unit tests.

By default, all the projects are published except the CodeCakeBuilder project and all the projects in the Tests folder.

You can allow to publish the Tests projects in a solution: In the world config file, you just need to add the attribute TestProjectsArePublished="True" to the Element SolutionSpec of your Solution.

You can explicitly exclude projects thanks to the <NotPublishedProjects> list:

<NotPublishedProjects>

<add Folder="CKSetupRemoteStore" />

</NotPublishedProjects>

Any dotnet project that are not published must have in the .csproj the element IsPackable=”false” in the PropertyGroup.