Part 4: Create your First Action

This tutorial continues on from the previous tutorials Part 1: Create your First Project, Part 2: Create your First Configuration and Part 3: Create your First Repository and it is recommended that these are completed before reading this tutorial.

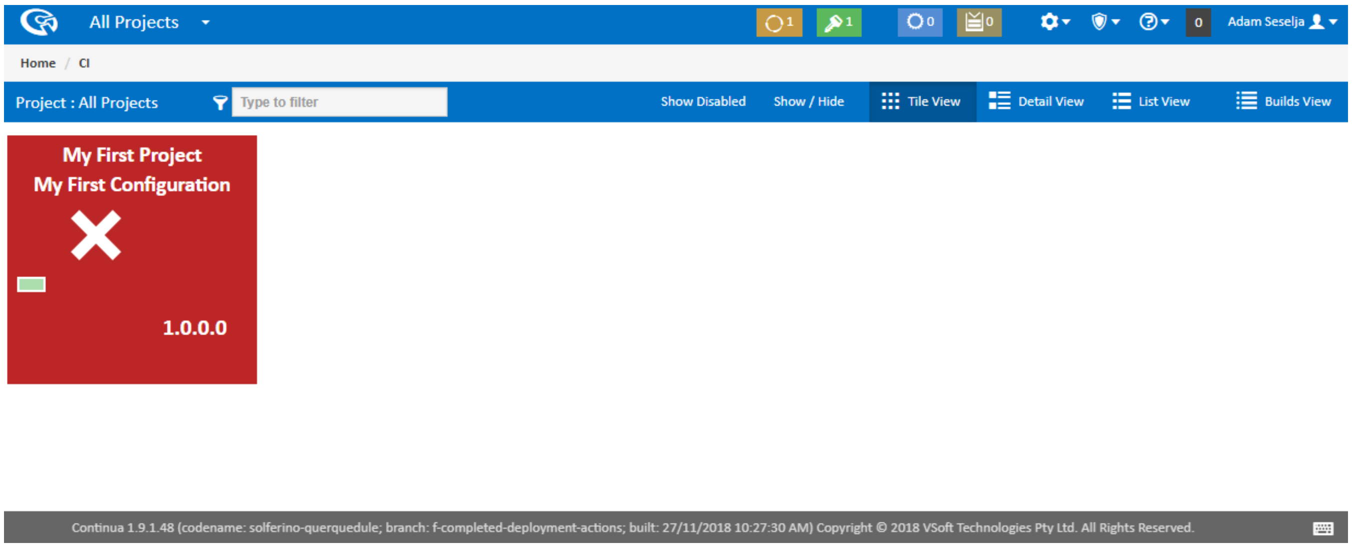
So far we have created a project, attached a configuration and created a repository, however we still cannot run a build! Fear not, by the end of this tutorial you will be ready to run your first build.

In this tutorial we will be creating a MSBuild action that will build our Fluent NHibernate source repository. This tutorial assumes you have the .NET 4.0 Framework installed on your server.

Attempting to Run a Build

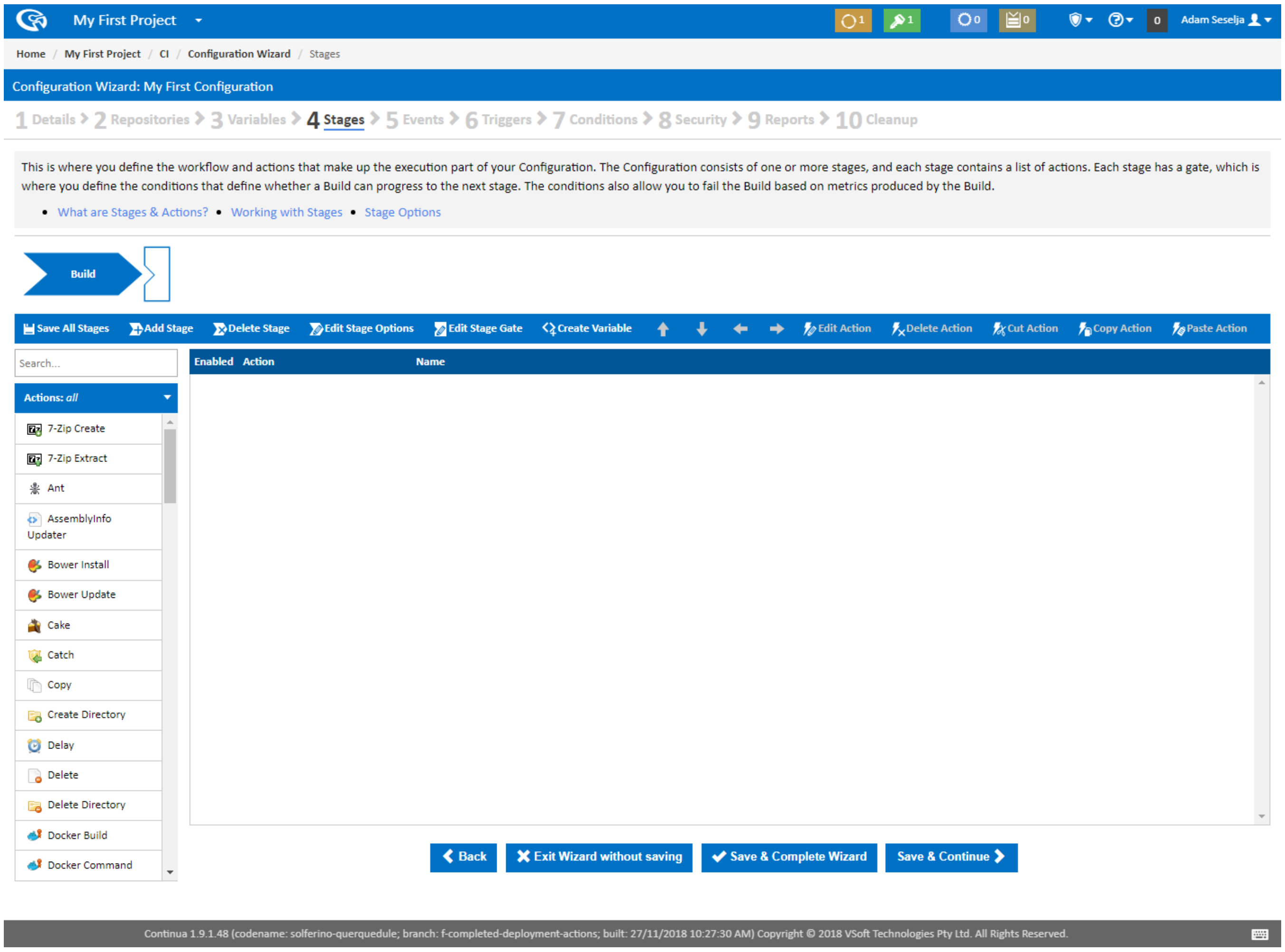
Go ahead and try running a build. The easiest way to run a build is by using the **Quick Build button (Fast forward icon)** on the configuration tile. The quick build action will run a build with default parameters passed into the build process.

Once the build has run, it will turn red, as shown below. The reason the build failed is that there are no actions for the build to complete. Well this is easily fixed by adding some actions to the **Configuration Workflow** in the **Configuration Wizard**. Click the **Edit Configuration button (Edit pencil icon)** on the configuration tile to enter the Configuration Wizard then navigate to the **Stages section** of the Configuration Wizard.


All Projects
1
1
0
0
0
Adam Seselja
Home / CI
Project : All Projects
Type to filter
Show Disabled
Show / Hide
Tile View
Detail View
List View
Builds View
My First Project
My First Configuration
1.0.0.0
Continua 1.9.1.48 (codename: solferino-querquedule; branch: f-completed-deployment-actions; built: 27/11/2018 10:27:30 AM) Copyright © 2018 VSoft Technologies Pty Ltd. All Rights Reserved.

Stages & Workflow Editor

Welcome to the Stage & Workflow Editor. This is where you define the individual actions that will run when a build is executed. For more detailed information on working with workflows, check out the Stages section.



This is where you define the workflow and actions that make up the execution part of your Configuration. The Configuration consists of one or more stages, and each stage contains a list of actions. Each stage has a gate, which is where you define the conditions that define whether a Build can progress to the next stage. The conditions also allow you to fail the Build based on metrics produced by the Build.

Continua 1.9.1.48 (codename: solferino-querquedule; branch: f-completed-deployment-actions; built: 27/11/2018 10:27:30 AM) Copyright © 2018 VSoft Technologies Pty Ltd. All Rights Reserved.

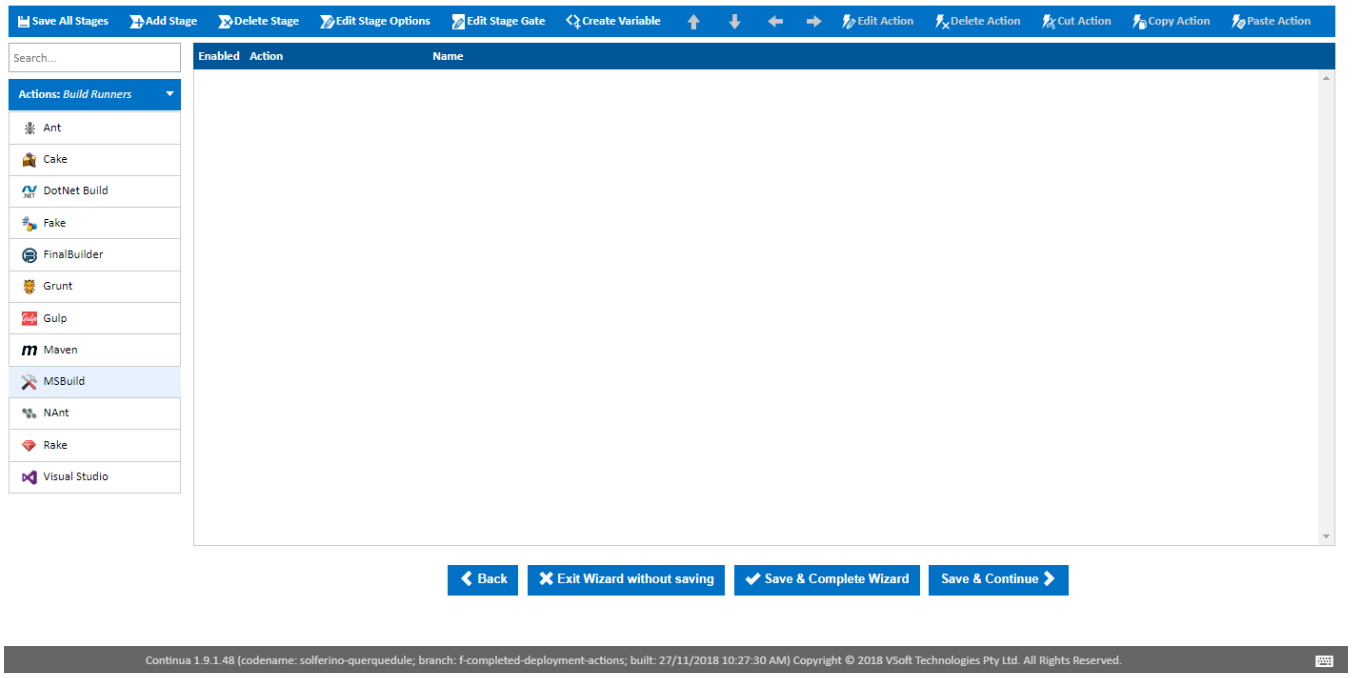
Stages

Continua CI creates a default 'Build' Stage when a new configuration is created and this stage is displayed near the top of the page in the Stage Editor. Each stage is represented as a blue chevron and is accompanied by a stage gate, which is represented as a box with a triangle cut out. Stages can be used as a way to break a complex build process into smaller, logical pieces. For example, your build may consist of a build stage, a test stage and a deployment stage. For this tutorial however the default 'Build' stage will be all we need.

Actions

Actions are the individual build steps that are executed when a build is run. These actions range from running a batch file to running a build runner and executing unit tests. All actions are defined and controlled through the bottom section of the stages page. On the left you can find all available actions and to the right is the Workflow Editor where the action execution order is defined.

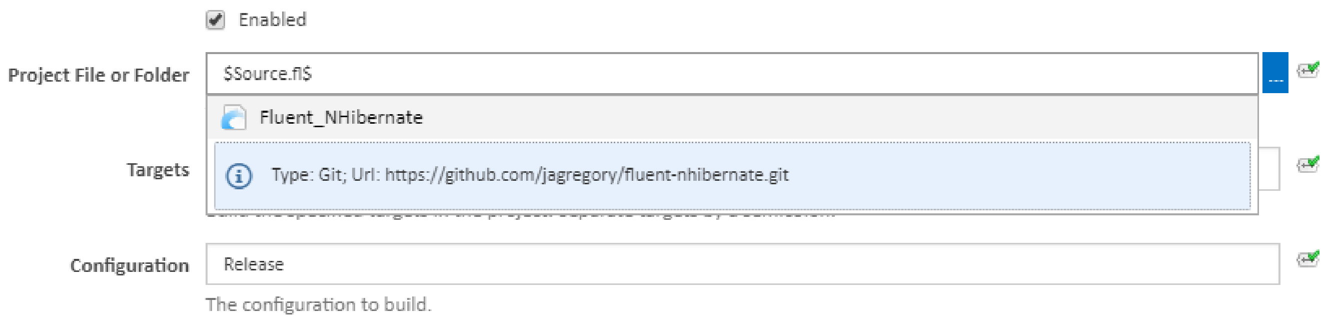
So lets create our first action by navigating to the **Build Runners category** as seen below. Selecting the Build Runners category will display all available build runner actions that can be executed within Continua. For this tutorial we want to build our Fluent NHibernate source code using MSBuild, so lets add a MSBuild Action by clicking **MSBuild**.



Continua 1.9.1.48 (codename: solferino-querquedule; branch: f-completed-deployment-actions; built: 27/11/2018 10:27:30 AM) Copyright © 2018 VSoft Technologies Pty Ltd. All Rights Reserved.

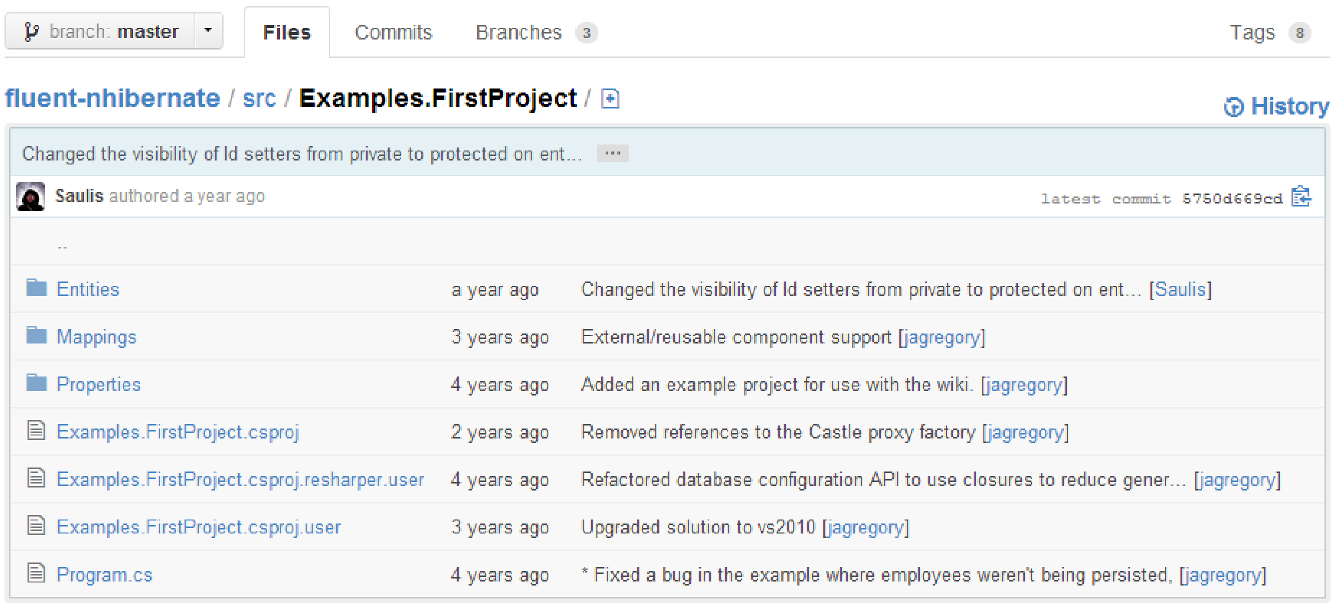
Selecting an action will bring up the action dialog where you can set all the properties for your MSBuild action. In this tutorial we will need to point this action to the Fluent NHibernate Example .csproj file, so lets set the Project File property to $Source.Fluent_Nhibernate$/src/Examples.FirstProject /Examples.FirstProject.csproj. Continua CI supports Variables, Objects & Expressions which we are using in this field to point to our repository. You can see that we are using $Source.Fluent_Nhibernate$ to point to our repository rather then providing a concrete file path. This is done so that you do not need to worry about where repositories are located as Continua CI looks after all repository management for you. So lets breakdown what is happening when we define our project file as $Source.Fluent_Nhibernate$/src/Examples.FirstProject/Examples.FirstProject.csproj.

Continua allows you to specify dynamic objects and variables that are then given values when a build is executed. Dynamic objects and expressions are registered with surrounding $ symbols. In the image below you can see that specifying $Source.$ displays a dropdown that lists all the repositories that can be accessed from this configuration. At the moment we only have our Fluent NHibernate repository, however any other repositories that may have been created will also display in this list. So by setting the project file to $Source.Fluent_Nhibernate$, we are pointing Continua to our Fluent_NHibernate repository. If you called your repository a different name then this property will need to reflect the new name. ie. $Source.<repo_name>$



	✓ Enabled
Project File or Folder	$Source.fl$
	Fluent_NHibernate
	Type: Git; Url: https://github.com/jagregory/fluent-nhibernate.git
Targets	
Configuration	Release
	The configuration to build.

Anything that follows $Source.Fluent_Nhibernate$ refers to the file structure of the repository itself. So by specifying /src/Examples.FirstProject/Examples. FirstProject.csproj we are telling Continua to get the Examples.FirstProject.csproj file from the src/Examples.FirstProject folder of the repository. On the Git Hub Fluent NHibernate page you can see the project's repository structure. In the two images below, you can see how we point to the .csproj file in Continua CI and how the project's file structure looks over at GitHub.


branch: master | Files | Commits | Branches 3 | Tags 8

fluent-nhibernate / src / **Examples.FirstProject** /

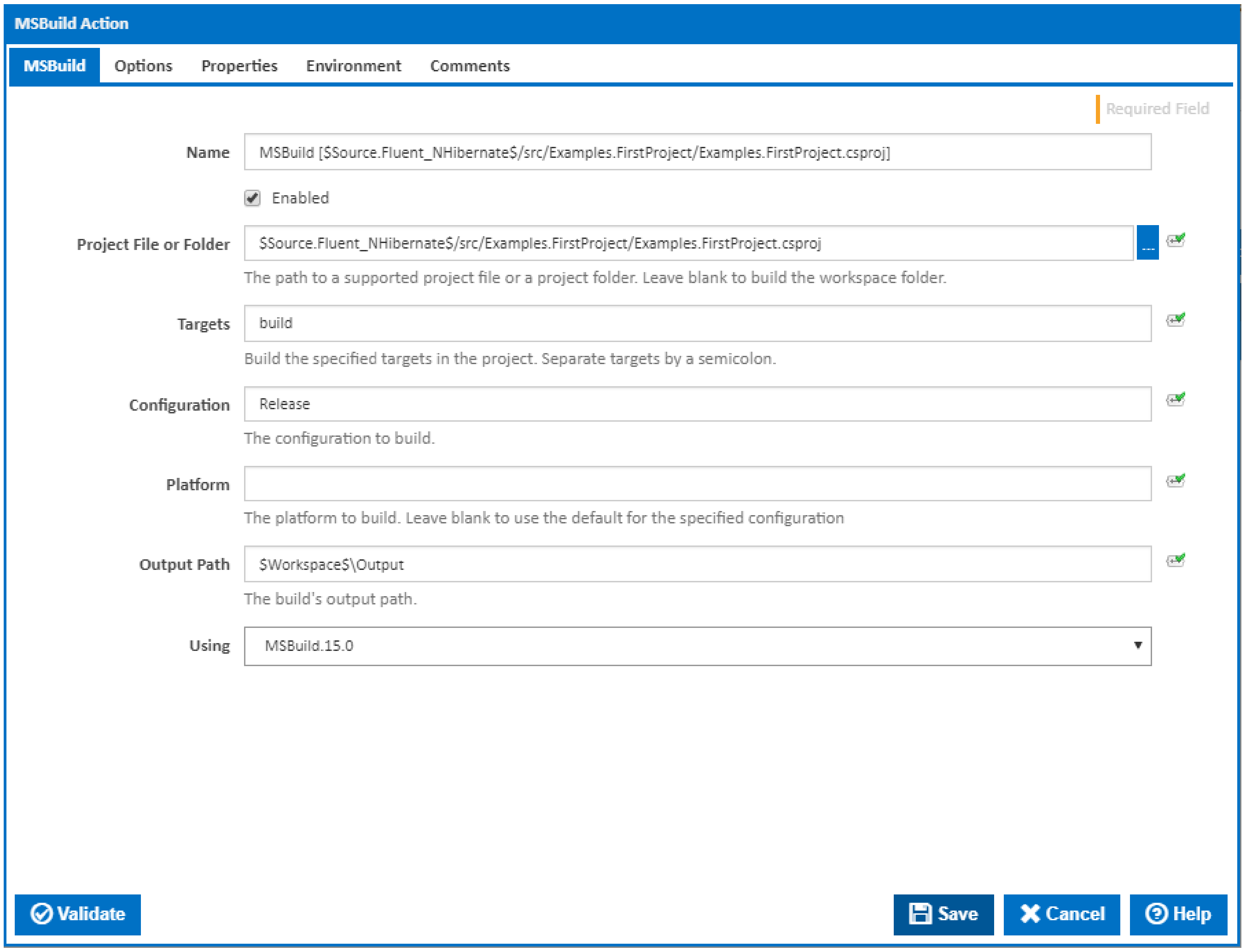
History

Changed the visibility of Id setters from private to protected on ent…

Saulis authored a year ago — latest commit 5750d669cd

..

Entities	a year ago	Changed the visibility of Id setters from private to protected on ent… [Saulis]
Mappings	3 years ago	External/reusable component support [jagregory]
Properties	4 years ago	Added an example project for use with the wiki. [jagregory]
Examples.FirstProject.csproj	2 years ago	Removed references to the Castle proxy factory [jagregory]
Examples.FirstProject.csproj.resharper.user	4 years ago	Refactored database configuration API to use closures to reduce gener… [jagregory]
Examples.FirstProject.csproj.user	3 years ago	Upgraded solution to vs2010 [jagregory]
Program.cs	4 years ago	* Fixed a bug in the example where employees weren't being persisted. [jagregory]


MSBuild Action

MSBuild | Options | Properties | Environment | Comments

Required Field

Name	MSBuild [$Source.Fluent_NHibernate$/src/Examples.FirstProject/Examples.FirstProject.csproj]
	☑ Enabled
Project File or Folder	$Source.Fluent_NHibernate$/src/Examples.FirstProject/Examples.FirstProject.csproj The path to a supported project file or a project folder. Leave blank to build the workspace folder.
Targets	build Build the specified targets in the project. Separate targets by a semicolon.
Configuration	Release The configuration to build.
Platform	 The platform to build. Leave blank to use the default for the specified configuration
Output Path	$Workspace$\Output The build's output path.
Using	MSBuild.15.0

Validate | Save | Cancel | Help

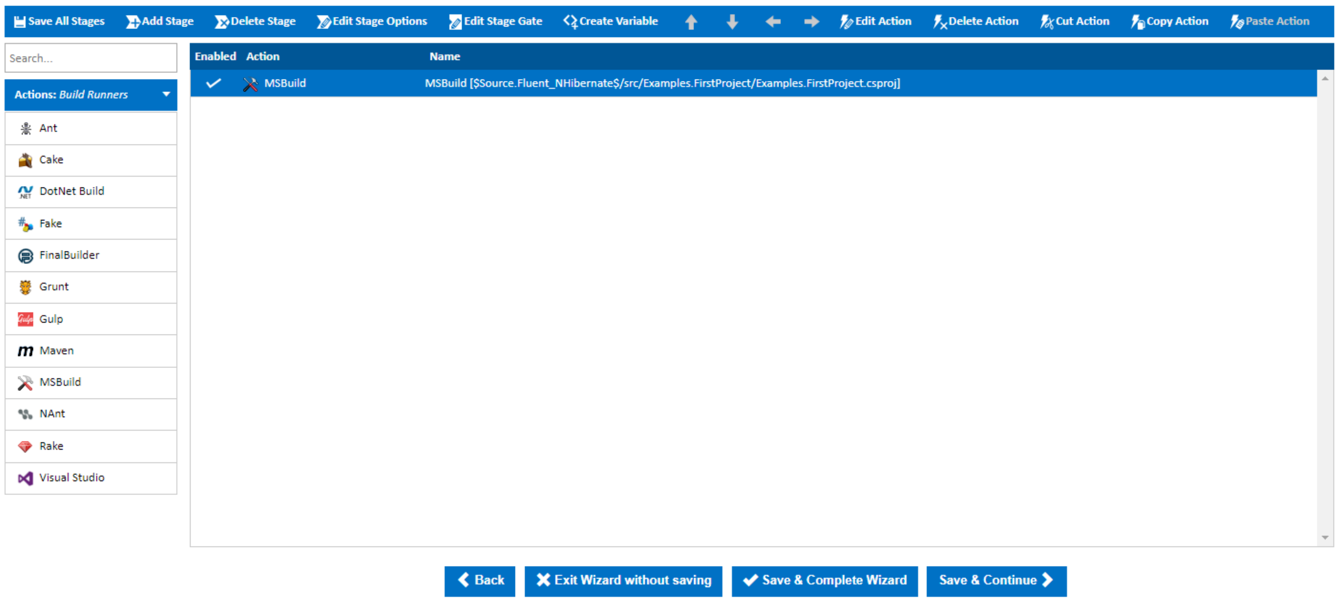
For this tutorial we need to set **Project File**, set the action to **enabled**, set **Configuration** to **Release**, set **Output Path** to **$Workspace$/Output** and set the **Using property to whichever .NET Framework version that is installed on the server.**

You should notice that we also used a dynamic expression when setting the output path. In this case we used the expression $Workspace$. This expression is extremely important in Continua as it references the workspace folder on the agent where all your build actions will be executed. As each build is executed it dynamically creates new temporary directories on the agent while the build is executing. These directories will always be created in the ContinuaAgent folder which you specified when you installed the agent. The reason you should always refer to your build files using $Workspace$ is that you do not know the directory name or structure at runtime. By using $Workspace$, you will always be referencing the root folder for a specific build on the agent.

You may also notice that we are specifying that this project should be built in the subfolder **'Output'.** By default, every stage will copy all the files from the output folder back to the Continua server once the stage has completed. This copying of data between the agent and server is defined on the stage itself using Workspace Rules. For this tutorial however, the default workspace rule is all we need, as long as our project is built in **$Workspace$/Output**.

The **Using** property is used to tell Continua which version of the .NET Framework it should use when executing the MSBuild action. This Using property uses property collectors to point Continua to the correct application it should use when performing a task. Unless your .NET framework was installed in a non-standard directory, the default Property Collector should find MSBuild. If Continua cannot find MSBuild then a new property collector will need to be specified in the administration section.

Once your MSBuild action has been configured, click Save which will add your action to the Workflow editor, as shown below.



Enabled	Action	Name
✓	MSBuild	MSBuild [$Source.Fluent_NHibernate$/src/Examples.FirstProject/Examples.FirstProject.csproj]

If you were to create additional actions then these would also be added to the Workflow editor underneath your MSBuild action. In Continua, actions toward the top of the list are executed before actions that are towards the bottom. For now though, the MSBuild action is all we need to build our project so lets save our workflow by clicking **Save & Complete Wizard**.

Congratulations, you have created your first action workflow! Continue on to Part 5: Using Builds where we demonstrate how to run your build and access all the build information.

Continue to Part 5: Using Builds