XenoTemplates Manual

(for version 1.2)

Table of Contents

Overview

Thank you for purchasing XenoTemplates!

XenoTemplates is a Unity editor extension which allows you to create new code/script files from custom templates. You can use XenoTemplates to create any type of text file including C#, Unityscript, and Boo scripts. By default, if you want to create e.g. a new C# script file in Unity you must use the “Create ▸ C# Script” menu option to create a simple boilerplate file and then modify it to suit your needs. Script files created in this way are completely static and so must be modified each time they are created if you are making anything other than a Monobehaviour, or

if you wish to include any dynamic data such as date stamps, authorship, correctly dated copyright lines, etc.

XenoTemplates improves this workflow in two key ways:

1. XenoTemplates provides an easy mechanism to make and use as many different templates as you like. This means that if you're creating e.g. a StateMachineBehaviour your newly created file can already be set up with all your standard boilerplate code for StateMachineBehaviours.

2. XenoTemplates allows for templates to include substitution markers which will be replaced with dynamic data when new files are created from the templates. Substitution markers can be replaced with any string that can be generated in C# code or fetched from an environment variable. Common uses are file creation dates, dated copyright lines, and authorship attributions.

Template and mapping files can be stored anywhere in a project hierarchy, so they are easy to share with everyone working on a project. Templates are a handy way to standardize file layouts and encourage consistent coding practices. Templates and mappings are all fully customizable from within the editor, and are easy to create and organize.

We hope you have an easy, trouble free experience with XenoTemplates, but if you ever run into any difficulties please get in touch with us at support@xenobrain.com and we'll be happy to help you out.

Installation

Installation of XenoTemplates is easy. Simply import the package as you would any other (either through the Asset Store window, via "Assets ▸ Import Package ▸ Custom Package…", or by dragging the Unity package file to your project window). Odds are that if you're reading this you've already successfully installed XenoTemplates.

Upon installation, two directories will be created in your project:

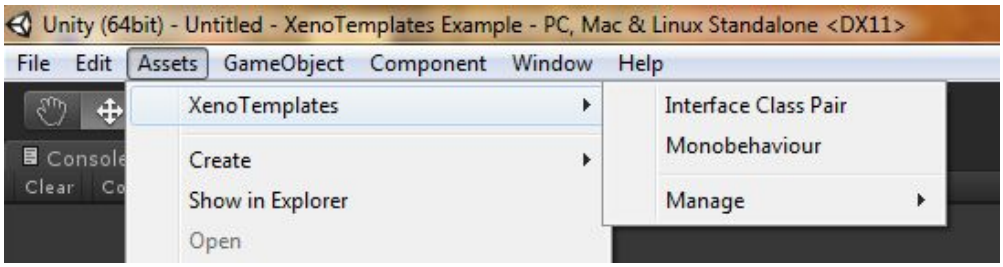
Xeno
This directory holds all the editor code for XenoTemplates.

XenoTemplateFiles
This is where the default templates and mappings live.

You are entirely free to relocate these directories anywhere you'd like in your project.

Accessing and Using XenoTemplates

Creating new files using XenoTemplates is easy! When it's installed XenoTemplates adds a new sub-menu item to the "Assets" menu, which can be reached either in the main menu bar or by right clicking in the project window. This new sub-menu contains a list of the available templates and a "Manage" sub-menu which is used to configure and customize XenoTemplates.



To create a new file (or group of files) from a template just open the "Assets ▸ XenoTemplates" menu and select the template you'd like to use. The Template Instantiation Window will pop up and you'll be prompted to enter a name for your new file(s):

The Template Instantiation Window tells you how many files will be created and shows you how they will be named. The name of each file in the group of created files is determined by the name that you enter and a format provided by the template. Please see the section on Template Elements below for more details.

Once you're happy with the name of your file(s) just click "Proceed" or hit Enter and your files will be created in the active directory of your project. Please note that if you summon the "Assets" menu by right clicking in the project window then the directory in which you right clicked is where the new files will be created. This works just the same way as Unity's built in "Assets ▸ Create" menu items.

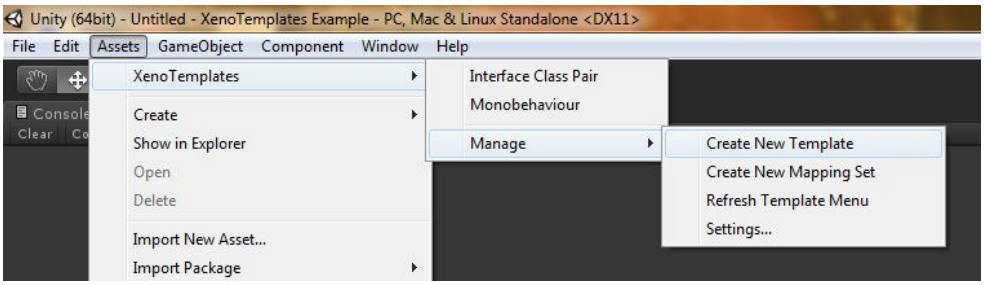
That's it. Your new files are now ready to be edited in your favorite code editor.

Templates

Templates are special assets which control how new files will be created by XenoTemplates. While the XenoTemplates package contains some default templates to get you started, the real power of XenoTemplates comes from making and customizing your own.

Creating a New Template

To create a new template, select “Assets ▸ XenoTemplates ▸ Manage ▸ Create New Template”.



This will create a new asset called “NewXenoTemplate” in your current directory.

IMPORTANT NOTE: If your current directory path does not include a directory called “XenoTemplateFiles” then a new directory called “XenoTemplateFiles” will be created in your current directory and the new template asset will be created inside that new directory.
So, if your current directory is “Assets” then “Create New Template” will create a new template asset in “Assets/XenoTemplateFiles/Assets”.

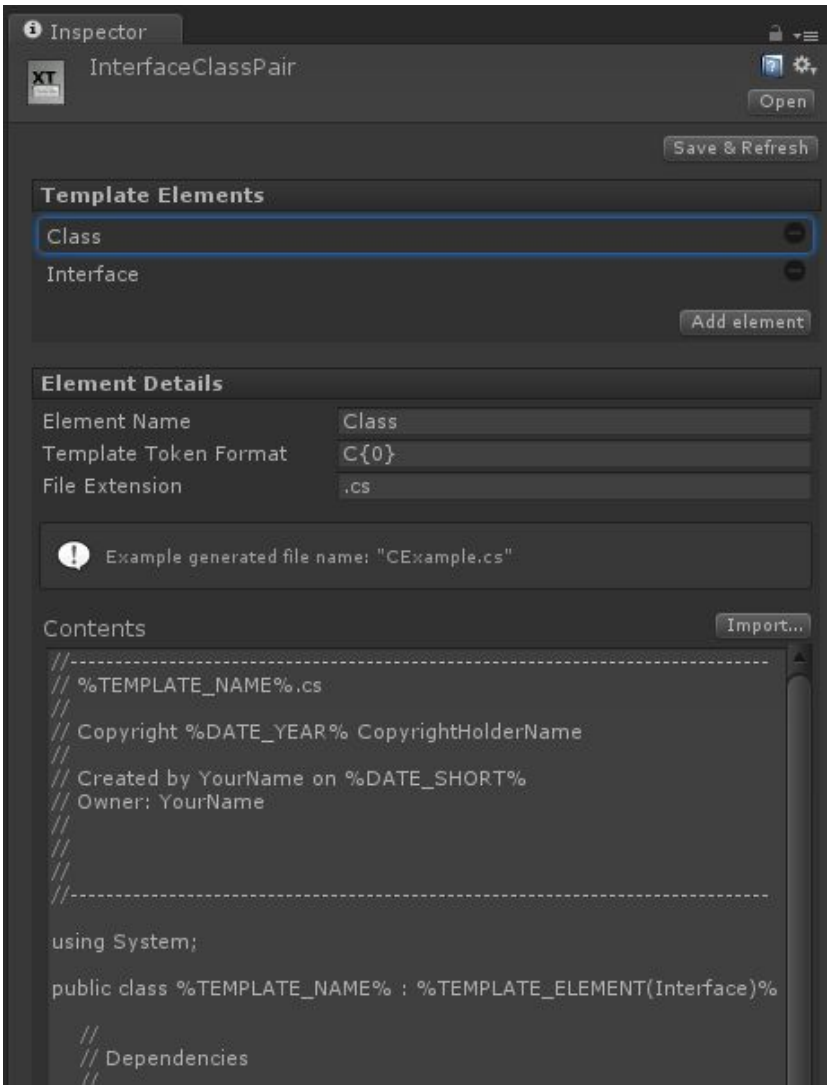
Your new template asset’s name will appear in the “Assets ▸ XenoTemplates” menu, so we strongly recommend renaming the template asset to something meaningful. Further, the directory path of your new template asset will determine how that template is organized in the XenoTemplates menu. Any template in a directory called “XenoTemplateFiles” will appear as a top-level menu item in the XenoTemplates menu. Template assets in subdirectories of “XenoTemplateFiles” will appear in correspondingly named sub-menus of the XenoTemplates menu.

Example: If you create a new template called “Foo” in the “Assets/XenoTemplateFiles/Silly” directory, then your new template will appear as “Assets ▸ XenoTemplates ▸ Silly ▸ Foo”. It’s worth noting that a new template called “Foo” created in the directory “Assets/Some/Directory/Path/XenoTemplateFiles/Silly” will appear in exactly the same way in the XenoTemplates menu. The relative path to the nearest XenoTemplateFiles directory is used to create the sub menus.

Your new template won't be very exciting at first as it will be completely empty. The next step is to add one or more Template Elements.

Template Elements

Each Template is made up of a number of Template Elements. Each element corresponds to a file that will be created when the template is instantiated. Most templates will only use a single element, but multi-element templates have a variety of uses. XenoTemplates comes with a template called "InterfaceClassPair" which consists of two elements - one for the interface file that will be generated and another for the class file that will be generated. Since template elements can reference each other in their substitutions, multi-element templates make creating groups of related files easy. See Substitutions below for more details.



Template elements may be selected by clicking on their names in the Template Elements section of the template inspector. Details of the currently selected element are displayed in the Element Details section of the template inspector.

New elements can be added by dragging an existing file from the project window into the Template Elements section of the inspector or by clicking the “Add Element” button. If a template has more than one element, then elements may be removed by clicking the minus button to the right of the element name. Note that if a template has only one element that element can not be removed. Either modify the element directly or add a new one before removing the old element. Adding new elements is discussed in more detail below.

Element Details

Each Template element has a Name, a Template Token Format, a File Extension, and Contents.

The **Name** is used to distinguish this element from any others in the template and to reference this element for substitutions in multi-element templates. It is not seen in the files produced by this template so feel free to name it whatever you like, though we highly recommend using descriptive names.

The **Element Token Format** is a C# style format string which is used to transform the name supplied by the user when instantiating a template into both the file name of this element and when making substitutions for this template name (see “Substitutions” below for more details on substitutions). If you’re not familiar with C# format strings don’t worry! The most important thing to know is that any time you want the name provided by the user when instantiating this template to appear in the file name simply use “{0}” (without quotes). For more information on C# format strings please see the online documentation for String.Format().

The **File Extension** is optional and is used to generate the name of the file generated by this template.

HINT: In the template inspector there’s an info box that will always tell you what file name will be generated when a user instantiates this template and provides the name “Example”.

Finally, the **Contents** are simply the text contents of the file that will be created by this template. Any substitution markers in the contents (delineated by % symbols, as in %TEMPLATE_NAME%) will be replaced by their appropriate substitutions based upon the available mappings. See the “Substitutions” section below for more details. Note that if there is no mapping defined for a substitution marker it will remain in the generated file.

Adding New Elements

Clicking the “Add Element” button in the template inspector will add a new, empty element to the template. At this point the details can be updated and the Contents can either be typed in directly or imported from an existing text file (and subsequently edited in the inspector if needed).

If you wish to create a template element based upon an existing script the easiest thing to do is to drag that script from the project window onto the Template Elements portion of the template inspector. This will add a new element which is pre-populated with the contents of the script and will make some guesses about the other element details.

When you’re done adding elements to your new template use the “Save & Refresh” button at the top of the inspector to ensure that your template changes are saved and the “Assets ▸ XenoTemplates” menu is up to date. Note that this is equivalent to “File ▸ Save Project” followed by “Assets ▸ XenoTemplates ▸ Manage ▸ Refresh Template Menu”.

Substitutions

Substitutions are how your XenoTemplate contents get populated with dynamic data, such as the current date, when they are instantiated. Special tokens in the contents of each template element are replaced with data based upon the current mappings (see the “Mappings” section below for further details on mappings).

Substitution tokens are delineated by % symbols. There are several built-in substitution tokens defined by XenoTemplates and these are extended by adding additional mappings using a XenoTemplate Mappings asset.

Built-in Substitution Tokens

%TEMPLATE_TOKEN%
This token will be replaced by the name provided by the user when instantiating the template. So, if the user typed “Foo” into the Template Instantiation Window then %TEMPLATE_TOKEN% will be replaced with “Foo” in the created file.

%TEMPLATE_NAME%
This token will be replaced by the name provided by the user when instantiating the template as transformed by the Template Token Format in the template element details. So, if the user typed “Foo” into the Template Instantiation Window and the Template Token Format is “Example{0}” then %TEMPLATE_NAME% will be replaced with “ExampleFoo” in the created file.

%TEMPLATE_NAME(*element_name*)%

This token is exactly like %TEMPLATE_NAME% except that it uses the value from a different template element (referenced by element_name). So, if this template has another element called “bar” which has a Template Token Format of “{0}Bar” and the user typed “Foo” into the Template Instantiation Window then %TEMPLATE_NAME(bar)% will be replaced with “FooBar” in the created file. See the included template InterfaceClassPair for an example of this behavior.

Organizing Templates

Finding Templates
All template assets are labeled as “XenoTemplateFile” using Unity’s built in asset labeling. You can always find all the templates in your project by using the “Search by label” button in the project window.

Moving Templates

You may move your templates anywhere that you like, rename them, or delete them at any time. If you do so, you must select the “Assets ▸ XenoTemplates ▸ Manage ▸ Refresh Template Menu” menu item (or click the “Save & Refresh” button in the template inspector) to update the XenoTemplates menu. In certain circumstances this can leave spurious, empty sub-menus in the XenoTemplates menu. These extra menus are an artifact of the way that Unity updates its editor menus and will disappear when you next restart the editor.

Mappings

Mappings allow you to create new substitution tokens to supplement the built-in ones provided. Mappings are defined in XenoTemplate Mappings assets which are created in a similar fashion to new templates.

IMPORTANT NOTE: While it is possible to have multiple mappings assets, it is important to keep the mappings uniquely named. If there is a conflict across mappings assets one of the mappings will win but which one is undefined.

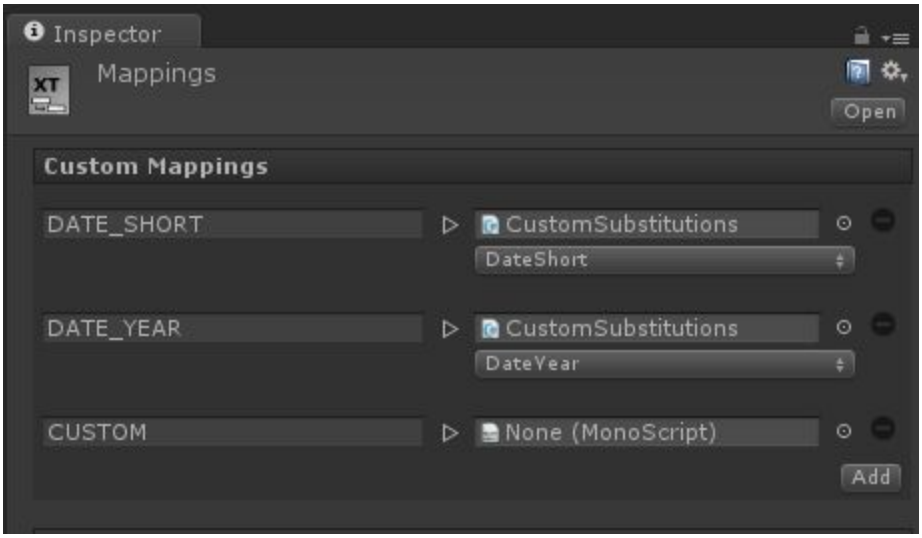
Mappings may be added by using the “Add” button in the mappings inspector and may be removed by clicking the minus button to the right of the mapping. There are two varieties of mapping which are described below.

Custom Mappings

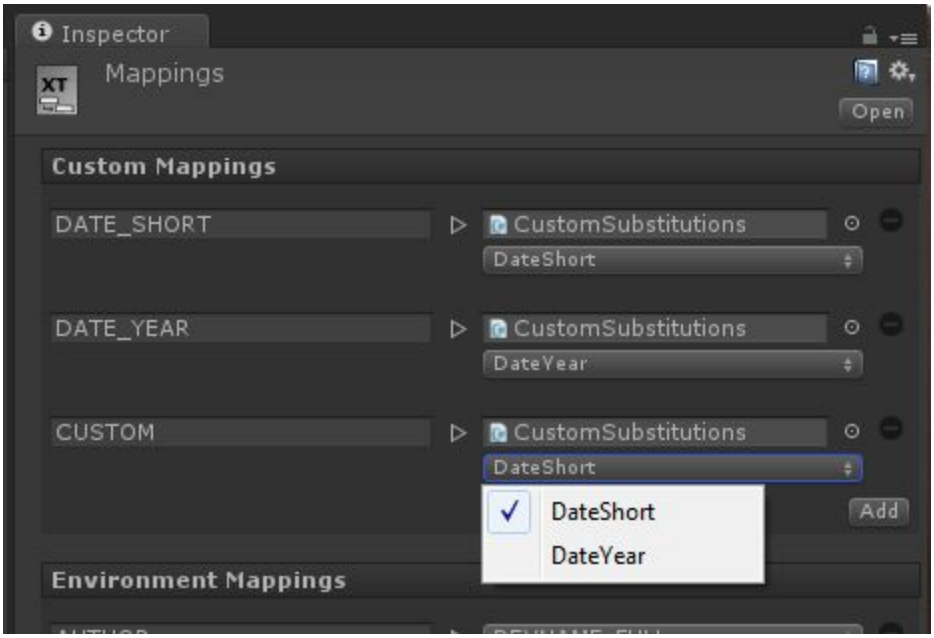
Custom mappings use functions you define to generate their substitution text. To implement a custom mapping you must create a C# script containing a class which exposes a method with the following signature:

public static String CustomMappingMethod();

This may be a bare C# class and the script does not need to be referenced by your project. Once the script is prepared you may drag it directly from the project window into the MonoScript field of a mapping in the mappings inspector. Or, you may use the object selector to find the script in your project.



Once you have selected an appropriate script object you will be presented with a drop-down menu of valid functions to use for the substitution. Select the one you wish to use and whenever a template is instantiated with the given substitution marker your function will execute and the result will be substituted into your new file.

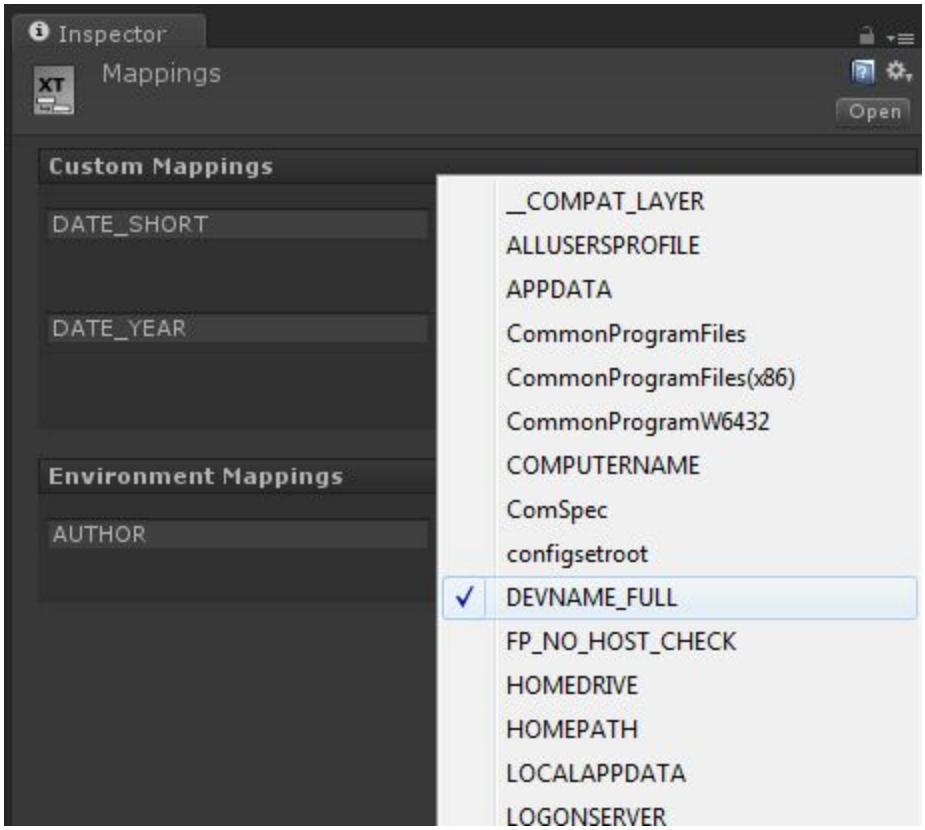


Note that if the chosen script file does not contain any methods which are appropriate for a Custom Mapping an error message will be displayed in place of the drop-down menu.

IMPORTANT NOTE: Because the code in your custom mapping will execute whenever you instantiate a template it is very important that you write simple, reliable code! It is entirely possible for you to write code that will e.g. hang the Unity editor (an infinite loop will do this trivially) so please be careful. If you run into problems with the editor after adding a custom mapping the first thing to check is the code in your mapped method.

Environment Mappings

Environment mappings allow you to embed the value of environment variables into your templates. Use the “Add” button to add a new mapping, then click the drop-down menu to select which environment variable to use.



IMPORTANT NOTE: Because mappings are portable between machines it is entirely possible that there may be a mapping for an environment variable that doesn't exist in the current editor environment. In that case the mapping will be preserved and you will be presented with a warning in the console window when instantiating the template.

Miscellaneous

Included Templates

The XenoTemplates installation package comes with two templates, a mapping set, and a custom substitution script. They provide examples of both single element and multi-element templates and show the use of each of the substitution types. You may use, extend, or delete these assets at your discretion.

Before using these templates you should edit the "CopyrightHolderName" (used in the copyright line) to either your name or your company's name, as appropriate.

While it is possible to simply edit the the templates to replace “YourName” (used in the authorship and ownership attribution lines) with your name, if multiple team members will be using the same template we recommend instead creating an environment variable substitution like %AUTHOR% which is mapped to an environment variable containing each developer’s name. This allows one template to be used by all members of the team while still having correct authorship attributions.

End of Line Normalization

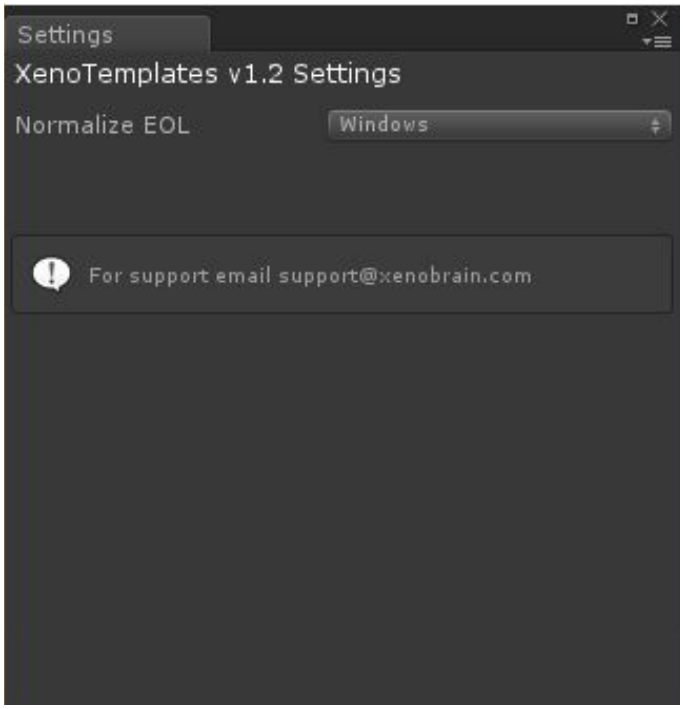
Different operating systems and file editors (including Unity) have different representations for the End of Line (EOL) in a file. Since you may be using a template that was created on an operating system other than the one you are currently using it is entirely possible that the EOLs in the template will be encoded in a way that your system or editor doesn’t like. All modern systems and software can handle this situation, but they usually complain a bit about it. In order to remove the irritation of constantly needing to normalize the EOLs encoded in your files XenoTemplates automatically normalizes EOLs when instantiating templates. Which encoding to use when normalizing EOLs can be configured in the Settings window.

If you don’t know which setting to use feel free to just leave this alone. If you later find that one or more of your editors are complaining about mismatched line endings you can come back and switch to the other option.

TIP: Unity and MonoDevelop use Unix style line endings by default, even on Windows.

Settings

XenoTemplates gets most of its configuration from the Template and Mappings assets, however there is a settings window to hold the one global setting which currently exists.



The settings window also contains the current version number of your XenoTemplates install, as well as a reminder of our support contact information.