VSCode’s Editor

Abstract

This documentation provides a quick reference to the core features of Visual Studio Code’s Editor. It focusses on aspects that are expected to be used in the context of the prensent template. Reading these pages is not a prerequisite to use the template. However, as it points out the way the template was designed in the aim of exploiting certain of VSCode’s features and extensions, a quick walk through before starting to use the template for the first time should be of interest even to those who are already familiar with VSCode and is recommended to everyone.

▼ **Table of contents**

About This Documentation

Visual Studio Code or, for short *VSCode*, is a cross-platform source code editor developed by Microsoft. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open source and released under the permissive MIT License. The compiled binaries are freeware and free for private or commercial use.

The purpose of the present documentation is not to serve as a tutorial for VSCode, but to provide an insight of the features and extensions that the enclosing template was designed to exploit. It is not a prerequisite to read it before using the template, but it is recommended to do so, even for those who are already familiar with VSCode. The sections dedicated to the extensions, in particular, should be of interest even to VSCode’s experts. The latter are also encouraged to do so and to provide feedbacks and suggestions, as knowledgeable opinions are for sure one of the most valuable assets any project can benefit from.

VSCode’s Usefull Links

The present documentation is not intended to serve as a reference for VSCode. For any question you may have about VSCode itself, please refer to the following links:

- Official website
- Official documentation
- Getting started with Visual Studio Code

Despite it not being the goal of this document to provide a tutorial for VSCode, a few words about its core concepts features are in order. The next section describe how a project is organized in VSCode and how configuration works.

VSCode’s Editor Overview

VSCode’s Workspaces

A `workspace` is a collection of files and folders opened in the editor. A `single-root workspace` contains a single folder, while a `multi-root workspace` has several.

The concept of a workspace enables VS Code to:

- Configure settings applying only to specific folders.
- Persist tasks-link and debugger launch configurations.
- Store and restore UI states.
- Selectively enable or disable extensions only for that workspace.

Creating a `workspace` is as simple as opening a folder or a set of folders in the editor. Once opened, *VSCode* will automatically keep track of its state, which includes the last used layout. *VSCode* saves the configuration of `single-root` workspaces in a folder named `.vscode` in the workspace root directory. It does in a `<the-project-name``>.code-workspace` file for `multi-root` workspaces; that file may lie wherever convenient, including a remote location.

Note

The syntax used by *VSCode* in the case of a `single-root workspace` lacks consistency. Indeed, until a `.code-workspace` file exists, the `Settings Panel` labels `User` and *Workspace* as the available configuration levels. Then, it surprisingly saves the preferences within a `.vscode/settings.json` file, although such location should be home to nothing else than *folder level settings*! This inconsistency is

> enhanced when defining and saving some preferences **first** and creating a `.code-workspace` **later**.
>
> One would expect the *workspace* configuration to be available in the `Workspace` tab of the editor's *Settings Panel*, but they are not. Instead, they now considered a `Folder` configuration. Such behavior makes sense when considering the saving location but makes none when focusing on the editor's UI. Everything happens as if the UI changed the nature of the settings.

It is worth pointing out that multiple versions of a `.code-workspace` file can coexist, which allows one to save different configurations of the same workspace and switch between them easily.

VSCode Configuration

VSCode configuration happens at three levels: `User` , `Workspace` , and `Folder` , which override each other in that order. The `Settings Panel` of the editor lets the user choose the level to edit and updates the corresponding file:

- **User settings** are (typically) defined in the file `settings.json` located in the `.vscode` folder of the *user's home directory*. They then apply globally to all *VSCode* instances with the lowest priority. Note that the *user's settings* themsselves branch into two categories:
    - **Global settings** are defined in the file `settings.json` located in the `.vscode` folder of the *user's home directory*. They apply to all *VSCode* instances with the lowest priority.
    - **Profile settings** can be defined only if a *profile* different from the default one is used and apply only when that profile is active. In that case, they override the *global settings*. Switching from one profile to another is done by clicking on the profile's name in the bottom-left corner of the *VSCode*'s window. The same interface is also used to create, edit, and delete profiles.
- **Workspace settings** lie in the `.code-workspace` that *VSCode* creates when choosing to save a workspace. These settings then apply to the opened workspace, overriding the *user's settings* but not the *folder's settings*.
- **Folder settings** are in the file `settings.json` located in the `.vscode` folder of the selected `<root-folder>` . They apply, with the highest priority, to the whole contents of the enclosing folder.

Going through *VSCode*'s `Settings Panel` is not the only way to configure the editor. An alternative consists in modifying the corresponding `json` files directly. That second method is much faster and, in certain cases, gives access to more options than the graphical interface of the setting panel does. However, it requires a good knowledge of the available settings and/or intensive use of the documentation.