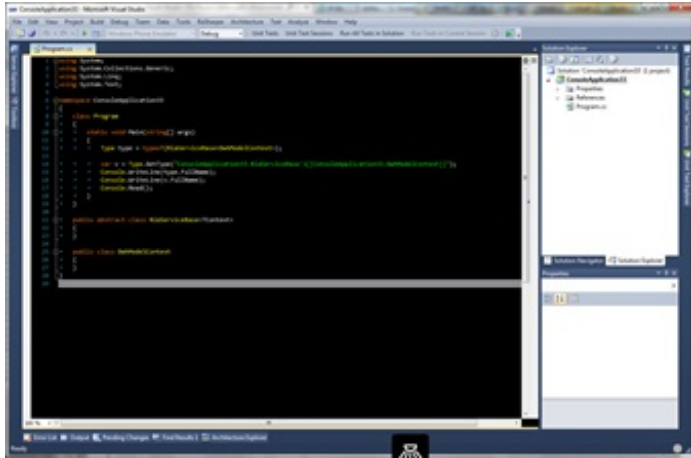


Original source - Sergey Zvezdin: personal blog (<http://blog.zvezdin.com/en/49>)

Using Visual Studio 2010 in different environments



Do you like to [customize Visual Studio](#)? Code formatting. [Code coloring](#). [Extensions](#). Hot keys. Windows and toolbars layout. I think that each developer who cares about his efficiency is engaged in it. As a result each of us have base package of settings, which is specific for current environment. You are like a duck to water in this environment. And the Visual Studio helps you to be more productive with automatic code formatting and refactoring. And it would seem, can already break nothing this harmony.

Unfortunately, banal vital circumstances can break this harmony. For example, you have decided to participate in the external project in which the rules of code formatting are different from your. Or you are speaking at event and listeners have

asked to change color and to increase font size because of singularities of a local projector. Familiarly, isn't that so?

What we often do in these situations? Certainly we open a Visual Studio settings and we change them according to our needs. In it there is nothing bad if only you don't need to do it periodically. And how to be, if you, for example, with constant periodicity give lectures to students? Or switch between projects with various rules of code formatting? In these situations you will need to change development environment settings frequently that finally can become strongly ineffective. And in this case the logical decision to automate these actions.

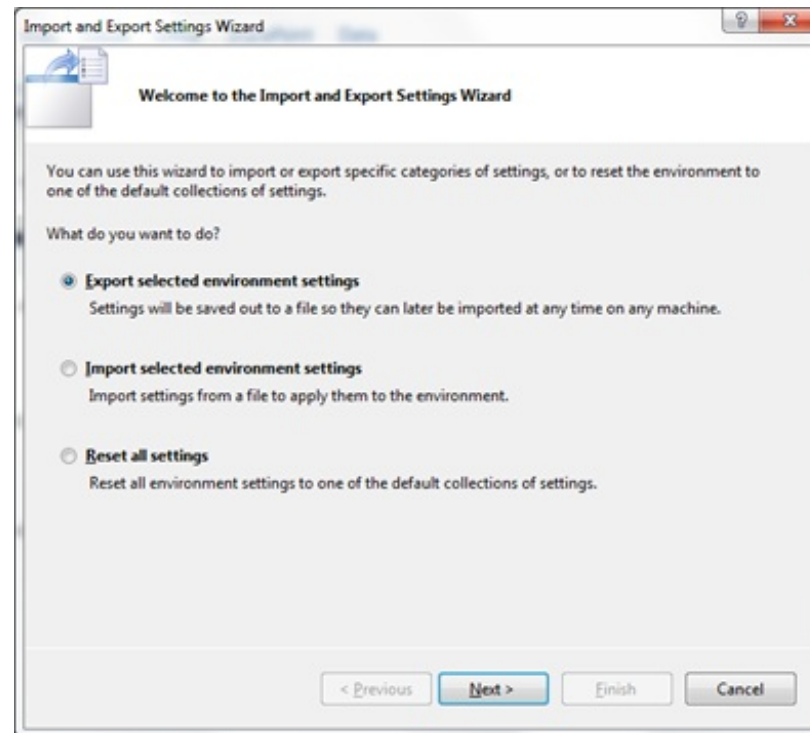
Two approaches can be the solution of this problem:

1. Usage of prepared settings for IDE for each case.
2. Usage of several isolated instances of IDE.


Let's consider each of them.

Prepared settings for IDE for each case

This approach assumes to store on a hard disk of several settings files. Each file corresponds to a different situation. To generation such configuration file very simply – simply enough to use the standard “Import and Export Settings Wizard” dialog. With its help it is possible how to save current settings in a configuration file, and to recover an environment state by an existing file.



Configuration files are the usual XML files that containing all settings of the environment that pointed at export. [Theoretically](#) you can automate setuping of Visual Studio according to any own algorithms. For this purpose it is possible to change contents of a configuration file. However it is very difficult to me to imagine, in what situations it is really necessary. After you have prepared some various configurations, you can import them when the appropriate surrounding is needed for you. But to use the mentioned “Import and Export Settings Wizard” dialog every time when you need to change an environment – superfluous expenditure of time. Instead it is possible to create some shortcut on a desktop for start of a Visual Studio and to specify for each of them a path to a configuration file:

```
 devenv.exe /ResetSettings "D:\VSConfig\settings1.vssettings"
```

It is easy to guess that in the example resulted above the “D:\VSConfig\settings1.vssettings” path just specifies a configuration file. Now

each shortcut of a Visual Studio on your desktop launches a development environment with an appropriate settings.

Benefit of this solution is automation of loading process of given configuration. But such solution possesses also a bunch of problems:

- start of a Visual Studio with reboot of settings occupies more durable time, than normal start of a development environment;
- internal caches of Visual Studio are cleared when you use /ResetSettings option. It slow down performance of Visual Studio.;
- configuration files always should be stored in an actual state (if you change IDE settings when you work and you want to save them for the future, you should update a configuration file on a hard disk);
- the logic of starting of several instances of Visual Studio with various settings are unobvious.

Usage of several isolated instances of IDE

Alternative method is usage of experimental isolated instances of a Visual Studio. Actually isolated instances have been created for the purpose of debugging of extensions for a Visual Studio – you develop own extension and launch debugging. At this moment the isolated instance of the Visual Studio which have independent setting is launched.

We can use this singularity for solving our task. For this purpose we need to have some isolated instances of a Visual Studio. Possibility of creation of isolated instances appears after installing [Visual Studio 2010 SDK](#). After installing SDK you will have [CreateExpInstance.exe utility](#). It controls isolated instances of Visual Studio.

For creation of a new instance it is necessary to use the following command:



```
CreateExpInstance.exe /VSInstance=10.0 /RootSuffix=Presentation
```

Actually this utility [simply creates](#) the necessary keys in Windows registry. RootSuffix parameter points a name of a created instance. This name is required for start of this instance. To similarly usage of configuration files, it is possible to create some shortcuts for Visual Studio starting and to specify parameters:



```
devenv.exe /RootSuffix Presentation
```

Now, when we have created some isolated instances of the Visual Studio, we can launch each of them and setup an appropriate environment which will be saved for this case.

Benefits of this approach are obvious – we get rid of all minuses of the previous method, creating for each environment.

Conclusion

The problem decided in this post can be actual not for all. But if for you this problem is actual, it is better to reduce an amount of necessary additional actions to a minimum. In my opinion the decision with usage of isolated instances of the Visual Studio is more useful, as allows to accelerate start IDE in comparison with the first method.

I wish successful experiments!