

Creating C# Based MPS Program

In this tutorial, you will create an MPS solution project with C# base language code.

First, you will need to install a set of plugins in order to be able to link your project to the C# base language and to the C# standard libraries.

Please, go to *File, Setting, Plugins, Marketplace*. Use the search field to find a plugin called *LangDoc*. Install it. Do the same for plugins *CsBaseLanguage* and *CsStdLibrary*. Then, restart the MPS. Now, when you return to *File, Settings, Plugins, Installed*, you should see what is illustrated in Figure 4.1.

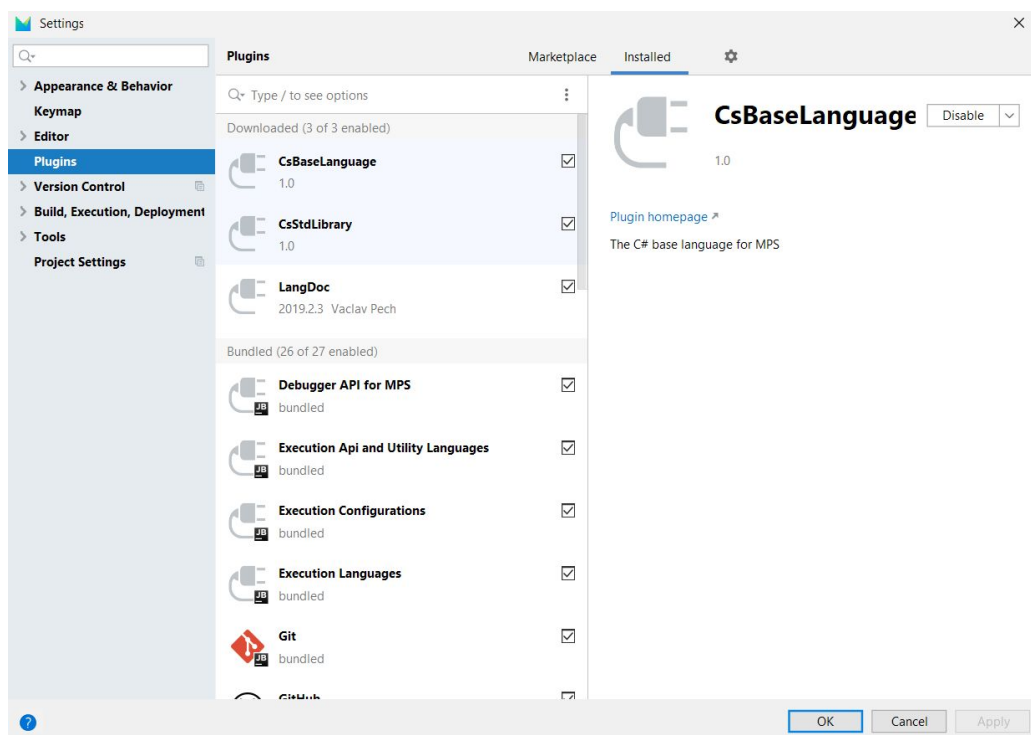


Figure 4.1: Plugins needed for creating a C# base language program

After closing the *Settings* window, please, select *File, New, Project*, as in Figure 4.2.

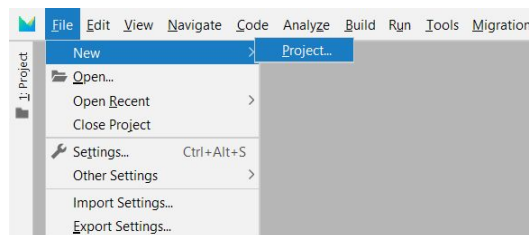


Figure 4.2: Creating a project

Select *Solution Project* and type in the project's and the solution's names. This is illustrated in Figure 4.3. Push the *OK* button.

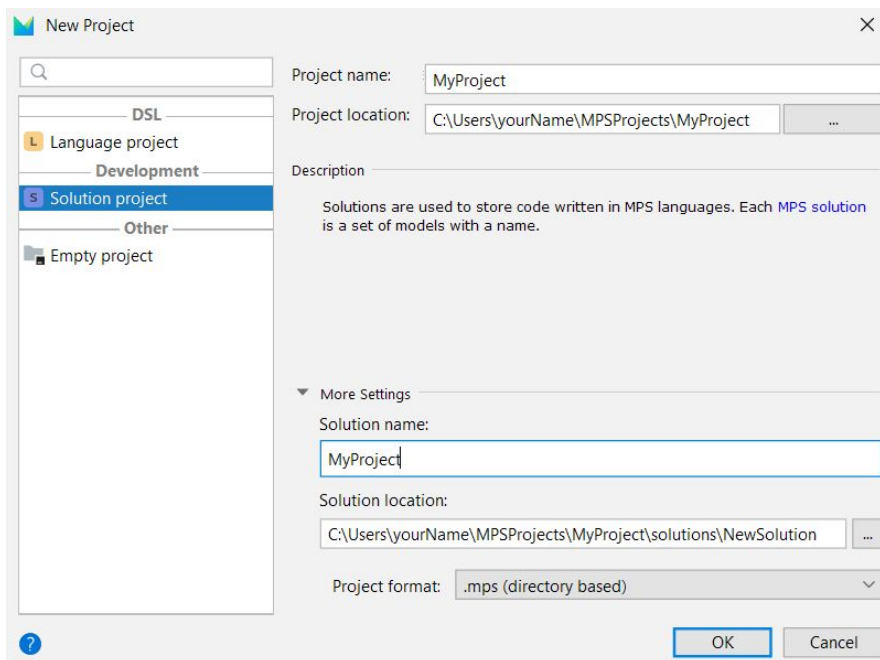


Figure 4.3: Creating a solution project

Then, create a new model (Figure 4.4). You will be asked to fill in the model's name. In this step, do not delete the auto-filled-in prefix and just append your model's name, as in Figure 4.5.

After doing so, you can add the model dependencies in the newly opened sub-window or you can close it and do it later. This tutorial will guide you the way with closing the sub-window and adding the dependencies from scratch, so that you can learn how to add them later than when you are creating a model.

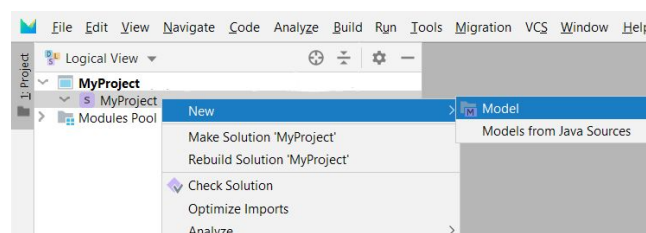


Figure 4.4: Creating a model



Figure 4.5: Naming a model

To add the dependencies after closing the sub-window, right-click your model and select *Model Properties*.

Under the tab *Used languages*, you can add a dependency on the C# base language. Use the plus button on the right, type in *CsBaseLanguage* and select the corresponding item. The result is illustrated in Figure 4.6.

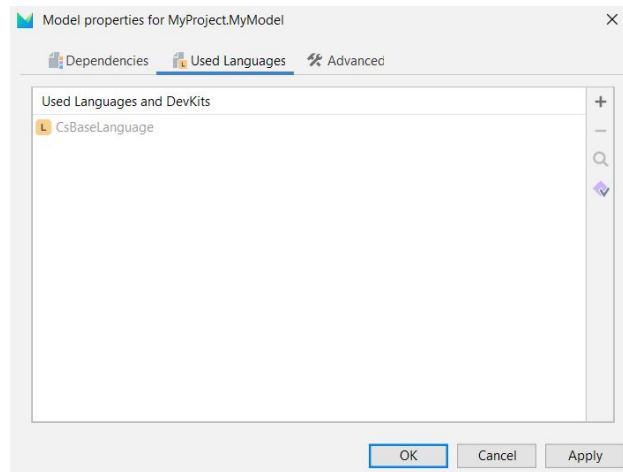


Figure 4.6: Dependency on the C# base language

Under the tab *Dependencies*, you can add dependencies on libraries (more precisely library stubs). For example, we will add a dependency on the *System* namespace of the C# standard library. Hit the plus button on the right and type *System*. Select the *System* item. You should see what is presented in Figure 4.7. Note that you cannot select a dependency on the whole *CsStdLibrary*, you must select particular models (top-most namespaces) in it.

Close the sub-window with the dependency settings.

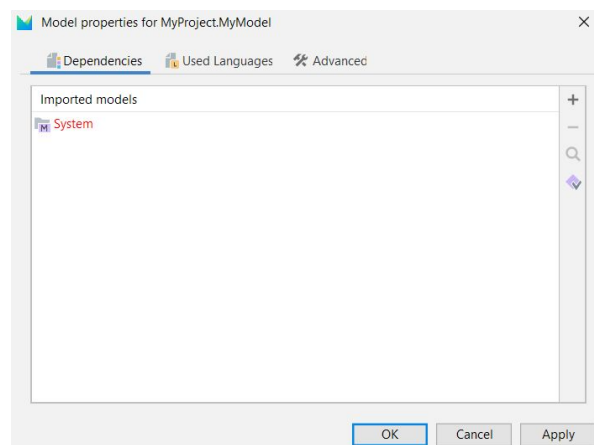


Figure 4.7: Dependency on the System model

Now, add a new root AST node. There is only one root AST node in our base language, called *File*. Right-click the model, select *New* and then *File* (Figure 4.8).

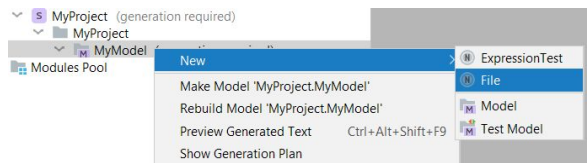


Figure 4.8: Creating a new File AST root node

In the created root AST node, you can write your code. For example, your first class (Figure 4.9). Importantly note that whenever you will experience some difficulties when writing the code, you will probably be able to proceed using the keyboard shortcut *CTRL+Space*. This shortcut offers you all possible AST nodes that can be created in the cursor's location. It is very often used when writing code in MPS because not always is the projectional editing of any selected language fluent as in a text editor.

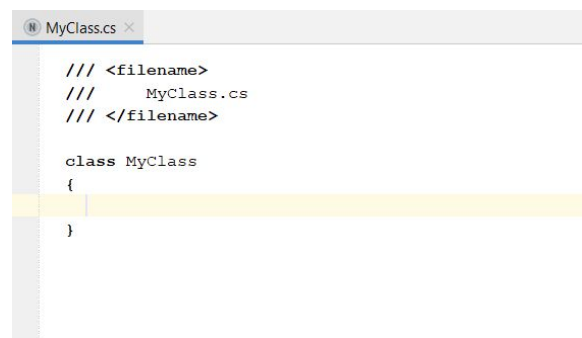


Figure 4.9: Creating the user's first class

If you wish to generate C# source code for your just implemented *File*, right-click the item corresponding to your *File* in the project explorer and trigger *Preview Generated Text* (as in Figure 4.10).

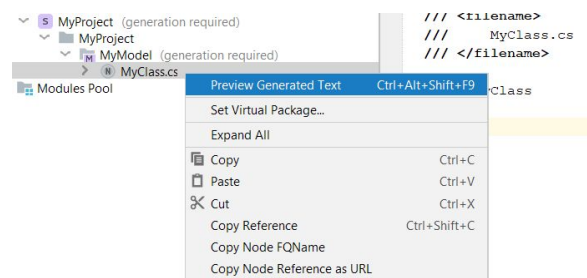
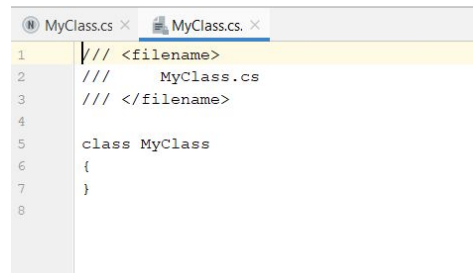


Figure 4.10: Generating the source code

Then you should see that MPS opened another tab with the source code for your *File*. For example, for our program from Figure 4.9, it would look like what is presented in Figure 4.11.



```
1  ///  
2  ///  
3  ///  
4  ///  
5  class MyClass  
6  {  
7  }  
8
```

Figure 4.11: Source code for the created user's first class

From this point on, you are on your own. For more help, remember the keyboard shortcut *CTRL+Space* and see the other tutorials in this document. You will probably also need to see the JetBrains MPS documentation for more general idea of the MPS functionality.