Contributing to JetBrains MPS Project

File a Bug Report

Bug reports are the simplest way you can get involved. Bug reports take little time to file and are very helpful to developers. When you discover a problem, please report it. Make sure you provide information about your environment (OS, JDK and MPS version), steps to reproduce the issue, as well as a textual description of the problem. You can file a bug in JetBrains MPS YouTrack bug database. Before submitting an issue, you may also want to search for already submitted ones describing the same problem - and if you find one, feel free to vote for it.

Create Unit Test reproducing the problem

Bug reports are helpful, but as you probably know, most of the problems can be reproduced/checked by automated JUnit tests. It's much more easy and convenient for our developers to reproduce the problem by running a test case instead of manually going through a list of steps to reproduce the bug. If you invest a little bit more time and create a JUnit test reproducing your problem, we will usually be able to fix/process your bug faster.

Contribute code indirectly

If you would like to improve the MPS code, you can submit a patch by attaching it to corresponding request in the JetBrains MPS YouTrack bug database. You can either file a new issue with the patch attached, or attach a patch to an issue submitted by another user. A developer will review your patch and, if it meets the quality criteria and fits with the rest of the code, you'll be notified about the acceptance of the patch. As it was already mentioned, patches having a unit-test attaches should be processed faster.

Contribute code directly

Although the main Git repository is located at JetBrains (see the Accessing MPS Git Repository section below), we keep a public mirror at GitHub. The mirror is primarily meant to be an easy-to-use repository for external contributors. Here's a cookbook on how to contribute your idea to MPS through GitHub

Building MPS from Source

This section describes how you create a fork of the MPS github repository and open the MPS project in IntelliJ IDEA.

1. The MPS miror repository is at https://github.com/JetBrains/MPS. Fork the jetbrains/MPS repository into your own space using GitHub UI (go to the MPS repo and press the fork button at the top. This will create a fork of the reposistory in your github space)
2. Once you created your own private fork, clone it to your local machine: git clone git@github.com:<yourGitHubName>/MPS.git. Depending on your internet connection, this may take a long time, MPS is big 🙂
3. It is preferrable to work in your own branch, so create a new branch to work in git checkout -b my_branch_name
4. Download IntelliJ IDEA from the JetBrains website. Community edition will work just fine.
5. Open the MPS project using IntelliJ IDEA. In the Open Project Dialg, the folder you cloned MPS into should have the "IDEA project" icon automatically; MPS uses the .idea sub-folder to hold IDEA project definition, so it recognizes the directory as an IDEA project.
6. IDEA may prompt you with an error that complains that no JDK is set up for the project. In the dialog that pops open, please create a new JDK and assign it to the project. Only JDK 6 and JDK 8 are supported.
7. Rebuild the project Menu -> Build -> Rebuild Project. You will need to be connected to the internet, since the process downloads several essential dependencies.
8. Run the MPSLauncher run configuration to start MPS inside IDEA: Menu -> Run -> Run...
9. Alternatively, start a debugger session with the MPSDebug run configuration to start MPS inside IDEA with the option to set breakpoints in IDEA and debug MPS code: Menu -> Run -> Debug...
10. Once MPS starts up, open the MPS project by pointing the Open Project dialog to the root of the cloned project: Menu -> File -> Open Project...
11. As soon as the project is open you should install a handy interoperability plugin into the IntelliJ IDEA instance that you are using for MPS development. Go to Menu -> Tools -> Install IntelliJ IDEA Plugin, point to the right IDEA installation and restart. This will enable joint compilation between MPS and IDEA.

Building MPS from Source on the command line

In order to build MPS from source you need to download the IDEA IntelliJ community edition and open the MPS project in idea. To do so, call "Open Project" and choose the folder in which the git repository is located. After that, trigger a rebuild. Once the rebuild is finished, open a shell and navigate into the build directory. The script to execute is called run.build.sh (assuming you have ant installed).

Making the contribution

This section describes how you make changes to the source code.

1. Now you should be ready to make your changes. Commit as frequently as you like.
2. You should test your code by running the provided automated test suits. These can be run from IntelliJ IDEA: Run -> Run... In particular, the GlobalTestSuite, ModuleTestSuite and ProjectTest run configuration are to be used.
3. There are no published code style guidelines at the moment. You may check out the MPS sources or drop us an email when in doubts.

Pushing your changes to us

This section explains how you can forward your local changes to JetBrains.

1. Push your branch to your private GitHub repository: git push origin my_branch_name
2. Make a pull request at GitHub - open your repository in a browser, make sure you are looking at the branch that you'd like to see pulled and click the Pull Request button in the upper right corner.
3. We should process your request and you'll get notification about the progress through the standard GitHub facilities.

Keeping your fork current

As you change your code and commit, you make changes to your own fork of the MPS github repository. Using pull requests, you can make the MPS developers aware of your changes, and they may or may not integrate your change into the original repository. However, in the meantime, the original MPS repository changes (as a consequence of the MPS team working on the code), and your fork is not kept in sync. To keep your local clone and your remote fork in sync with the changes in the original repository, please do the following:

1. in your local clone, add an additional remote that points to the original MPS github repository. git remote add mpsrepo https://github.com/JetBrains/MPS
2. you can now pull the changes from the original repository by using git pull mpsrepo master
3. if you develop your contribution on a branch as we suggested above, you'll have to merge the updated master into your branch using git merge master (while on your branch)
4. as you do that, you will have to resolve possible merge conflicts locally.
5. you can then push the merged changes to your own fork using git push origin (origin is the default name assigned to a remote by git clone).

In summary, you pull from the original, resolve conflicts and the push to your fork.

Accessing MPS Git Repository

The main MPS repository is located at git://git.jetbrains.com/mps/mps.git and can be used as a base for submitting your patches to MPS. The process using GitHub should, however, be considered to be the preferred way.

To check out from IntelliJ IDEA, select “Version Control | Checkout from Version Control | Git” from the main menu. In the “Git Repository URL” field, enter:

```
git://git.jetbrains.org/mps/mps.git
```

To check out from the command line, use the following command:

```
git clone git://git.jetbrains.org/mps/mps.git
```

In addition you can browse MPS sources using git web access.

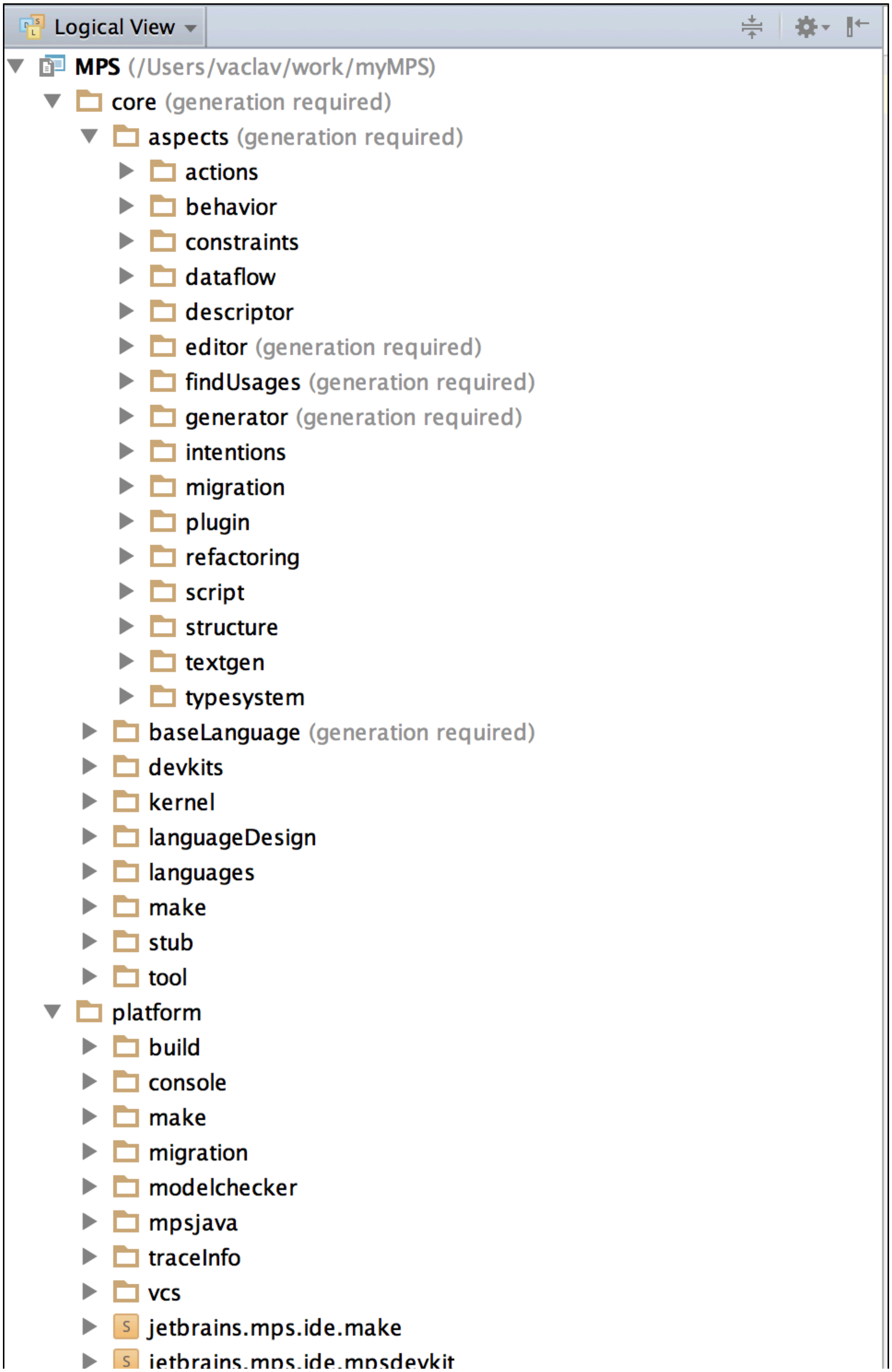
The project structure

Admittedly the over project structure is very difficult to see from the hierarchy of modules and packages in IDEA and MPS and you'll be having hard time understanding how the pieces fit together. We've collected a few guidelines to help you in many cases. For the rest, please get in touch with us directly or through the forum.

The big division

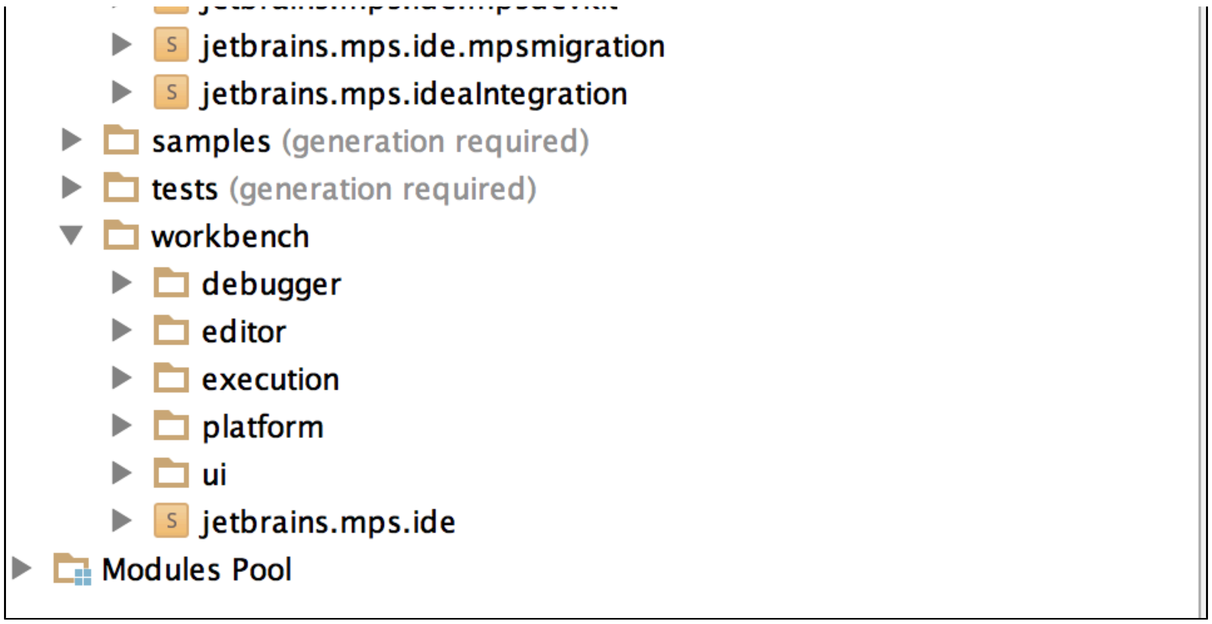
MPS is being developed in both IntelliJ IDEA and MPS itself. So the code base consists of Java sources as well as MPS models. Whenever you open a Java file in IDEA, you can check whether the file contains a comment /*Generated by MPS */, which indicates, that the source was generated and the real source is to be found in MPS, most likely under a similar name. The shortcut Control/Cmd + N for opening files, classes and root nodes by name will certainly come in handy and luckily it is the same shortcut in both IDEA and MPS.

Structure



Logical View

```
▼ MPS (/Users/vaclav/work/myMPS)
    ▼ core (generation required)
        ▼ aspects (generation required)
            ▶ actions
            ▶ behavior
            ▶ constraints
            ▶ dataflow
            ▶ descriptor
            ▶ editor (generation required)
            ▶ findUsages (generation required)
            ▶ generator (generation required)
            ▶ intentions
            ▶ migration
            ▶ plugin
            ▶ refactoring
            ▶ script
            ▶ structure
            ▶ textgen
            ▶ typesystem
        ▶ baseLanguage (generation required)
        ▶ devkits
        ▶ kernel
        ▶ languageDesign
        ▶ languages
        ▶ make
        ▶ stub
        ▶ tool
    ▼ platform
        ▶ build
        ▶ console
        ▶ make
        ▶ migration
        ▶ modelchecker
        ▶ mpsjava
        ▶ traceInfo
        ▶ vcs
        ▶ jetbrains.mps.ide.make
        ▶ jetbrains.mps.ide.mpsdevkit
```


jetbrains.mps.ide.mpsmigration
jetbrains.mps.ideaIntegration
samples (generation required)
tests (generation required)
workbench
debugger
editor
execution
platform
ui
jetbrains.mps.ide
Modules Pool

The Project View in MPS may give you rough guidelines on how the project is structured:

- core - as the name indicates, it holds the essential language-design functionality
    - aspects - holds the functionality and languages of the individual language definition aspects
    - baselanguage - holds the implementation of BaseLanguage and its extensions, such as closures, collections and other
    - languages - contains core languages, such as smodel, quotation, xml and others
    - stubs - stores loaded external libraries, such as JDK, Ant, the IntelliJ platform and the MPS Java sources
- platform - contains the IDE extension functionality, such as the console, vas support, model checking, build language, and other
- samples - groups the sample projects
- tests - contains tests for various aspects of MPS
- workbench - provides integration with the IntelliJ platform and implementation of many UI elements, supports execution and debugging

Committing your work

The MPS project stores generated artefacts in Git alongside the sources and models. Before committing please ensure that all the changed models have been regenerated - the (generation required) note must disappear. May you fail to do so, the TeamCity CI server will reject the commit by a failed build, since it checks all artefacts for being correctly generated. The best way to regenerate is to invoke Rebuild from the model's or module's context menu.