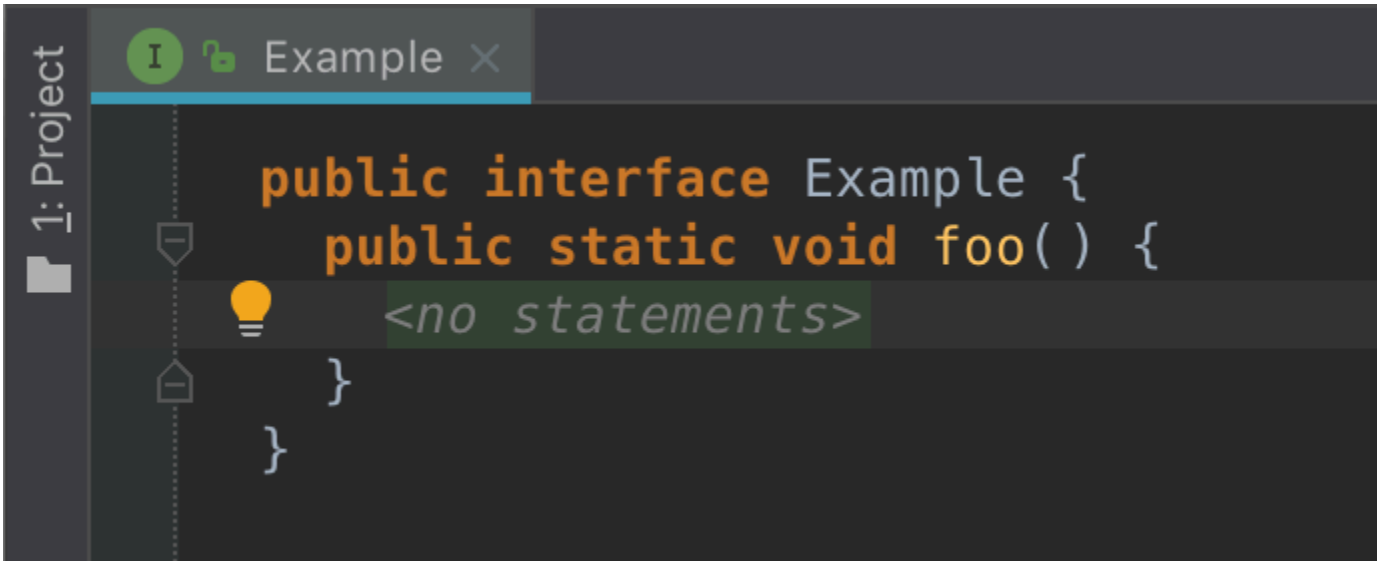
What's new in MPS 2019.1 (draft)

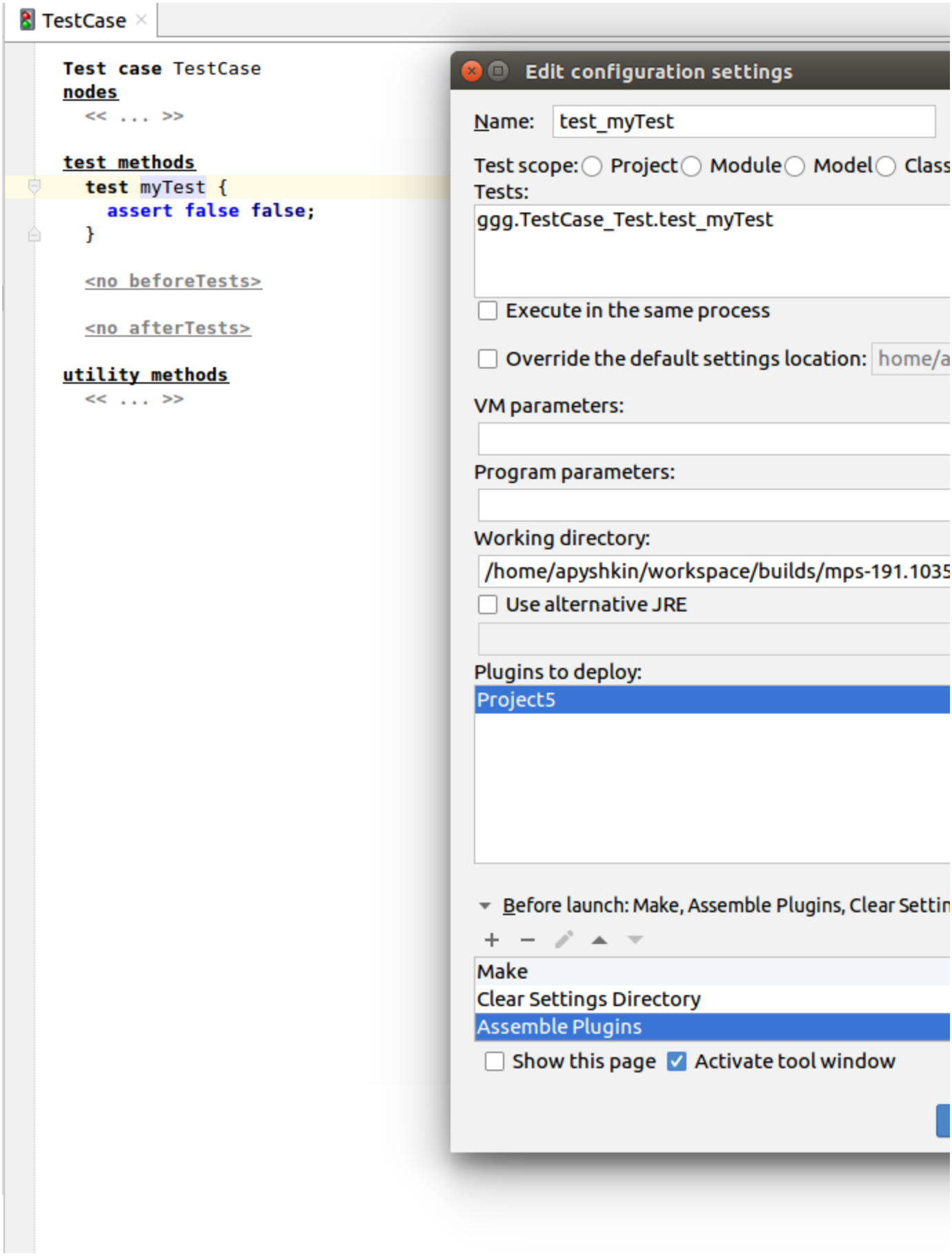
Static methods in baseLanguage interfaces

MPS allows creating static methods in interfaces. In order to use this feature you only need to import baseLanguage.



JUnit run configuration now accepts plugins to deploy

Similar to MPS Instance run configuration one is able to provide the list of idea plugins to be deployed on the test execution.

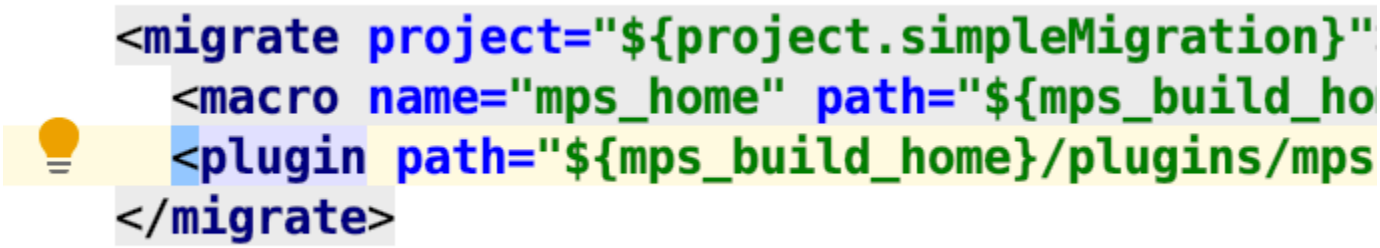

TestCase
Test case TestCase
nodes
<< ... >>
test methods
test myTest {
assert false false;
}
<no beforeTests>
<no afterTests>
utility methods
<< ... >>
Edit configuration settings
Name: test_myTest
Test scope: Project Module Model Class
Tests:
ggg.TestCase_Test.test_myTest
Execute in the same process
Override the default settings location: home/a
VM parameters:
Program parameters:
Working directory:
/home/apyshkin/workspace/builds/mps-191.1035
Use alternative JRE
Plugins to deploy:
Project5
Before launch: Make, Assemble Plugins, Clear Settin
Make
Clear Settings Directory
Assemble Plugins
Show this page
Activate tool window

The before task 'Assemble Plugins' is available in JUnit run configuration as well. It automatically builds the given plugins and copies the artifacts to the settings directory.

"migrate" ant task accepts plugins to deploy

Now, if a plugin is needed for a project to migrate, this can be specified in a <migrate> ant task. The corresponding plugin will be enabled, together with its dependencies.

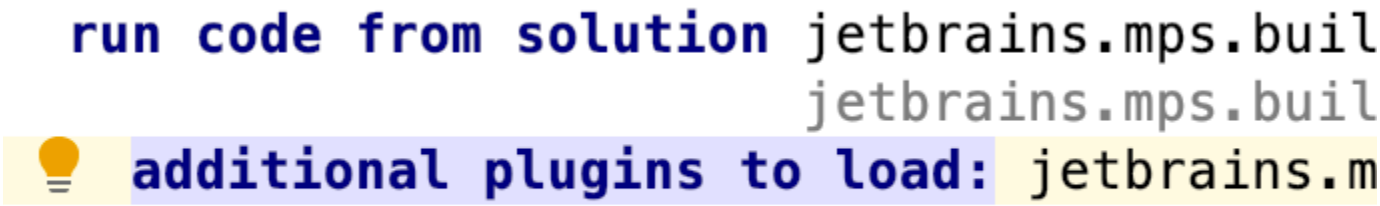
This feature was also integrated into 2018.3.5 by a support request.



Plugin deploy support in "run code from solution" build script instruction

The "run code from solution" instruction allows to enable plugins in the MPS instance that will run the code. The corresponding plugin will be enabled, together with its dependencies.

This feature was also integrated into 2018.3.5 by a support request.



New meaning of 'Compile in MPS' flag

Unchecked 'Compile in MPS' flag in module properties no longer implies there's connected IDEA instance to perform code compilation (MPS-29671)

Now MPS detects aggregated languages

Structure aspects now manifest languages they incorporate by aggregation (i.e. using a foreign concept in a child role), there's no longer need to import such languages explicitly into a model that uses aggregating language or to 'extend' language modules (MPS-28205, MPS-23122).

Support for automatic type inference has been introduced for new VAR macro

Generator language, VAR macro facelift - now multiple variable declarations per macro; type deduced from declaration query unless overridden (MPS-23598 and MPS-24911)

Custom style and priority of completion items (EXPERIMENTAL)

Language designers can set the style and the priority for the items of completion menu.

In order to do so the language designer should create the Completion Styling root in the editor aspect of the language.

In order to specify the style of completion menu item the language designer should first specify the item.

We currently provide with the two possible selectors of the completion items:

1) Items which modify instances of specific concept.This selector selects items which appear in the completion menu when user puts caret on the node of the concept (or it's properties, references, or children) and presses ctrl+space

2) Items which create instances of specific concept. Those are mainly the substitute actions. For example, when user selects the ReturnStatement which is in the StatementList and presses ctrl+space, items in the completion menu items are creating the instances of Statement and replaces the ReturnStatement with those instances. So selector for actions creating instances of Statement will select those items.

It is important that one item could be selected by many selectors. E.g in the situation above several selectors will match one completion item:

1) for actions creating instances of Statement (because the item will replace current node with another Statement)

2) for actions modifying instances of Statement (because the current node is ReturnStatement which is also Statement)

3) for actions modifying instances of StatementList:statement (because the item will paste new node into the statement role of parent StatementList)

So you should prefer the most specific selector possible.

For example selector of actions modifying instances of BaseConcept is the least specific selector and will affect all completion menu items

Once the selector matches the menu item, the language designer could customize it's style. There is the style object for that purpose.

Currently it is possible to:

- make item's font bold
- make item's font italic
- strike out item's text (useful for deprecating the concepts)
- set items' background and text color
- hide item
- set item's priority
- update item's description text

Note that all the styles accumulate: if at least one styler set bold/italic/hide, item will be bold/italic/hidden. If background or text colors conflict, the first will be used.

The priorities are used to sort the items. The most of priorities set for item by stylers is used.

Items are firstly sorted by priorities and then by level of matching the user's text. Currently there is no way to override this level.

There are several parameters the language designer could use, they are stored in the itemInformation parameter:

- matchingText - the text appearing on the left side of the item. This text is used to filter the items when user types text in the completion menu.
- descriptionText - the text appearing on the right side of the item.
- parameterObject - some of the items are parameterized with the object. For example, items which changes the node's reference are parameterized with the target of the reference which they will set when executed.
- outputConcept - some actions, like substitute actions create new nodes, the output concept is the concept of the new node.

The notable examples of stylers in baseLanguage:

1) LastReturnStatementStyling which makes return item bold and sets its priority to non-zero if it is last in current statement list

2) VariableReferencePriority which sets items priority to non-zero if the item references the variable declaration.

IMPORTANT NOTE:

This feature is experimental and it's design could be changed a lot in the future. The feedback regarding this feature will be very appreciated and will help us to improve it.

There is possibility to turn this off: Settings>Editor>General>Use completion styling

Sorting in ProjectPane (Client feature)

Added TreeNodeSortService to manage ChildComparatorProvider contributions that get full control over MPSTreeNode children in ProjectPane tree. Clients shall register/unregister providers from their plugin’s application/project parts.