Debugger features overview

Summary

Debugger allows user to debug programs which are written in languages which are finally generated into Base Language/Java. MPS java debugger offers the following features:

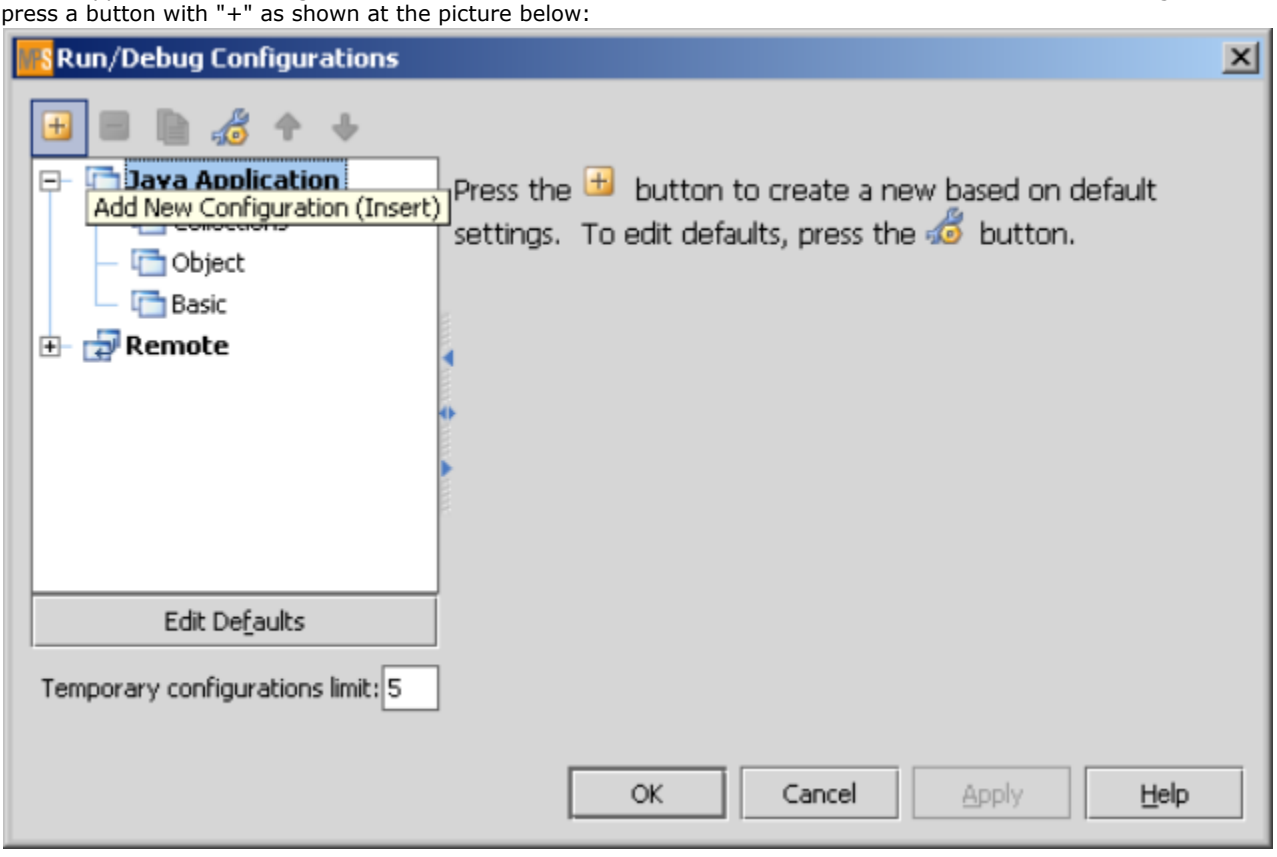
- Execution:
  - executing local run configurations under debugger;
  - connecting to a remote application.
- Breakpoints:
  - line breakpoints;
  - breakpoints viewer.
- Runtime:
  - current debugger position highlighting in editor;
  - viewing and exporting threads state;
  - viewing variables on stack frame;
  - step-by-step execution (stepping over/into/out);
  - low-level expressions evaluation.

Debugging a java application

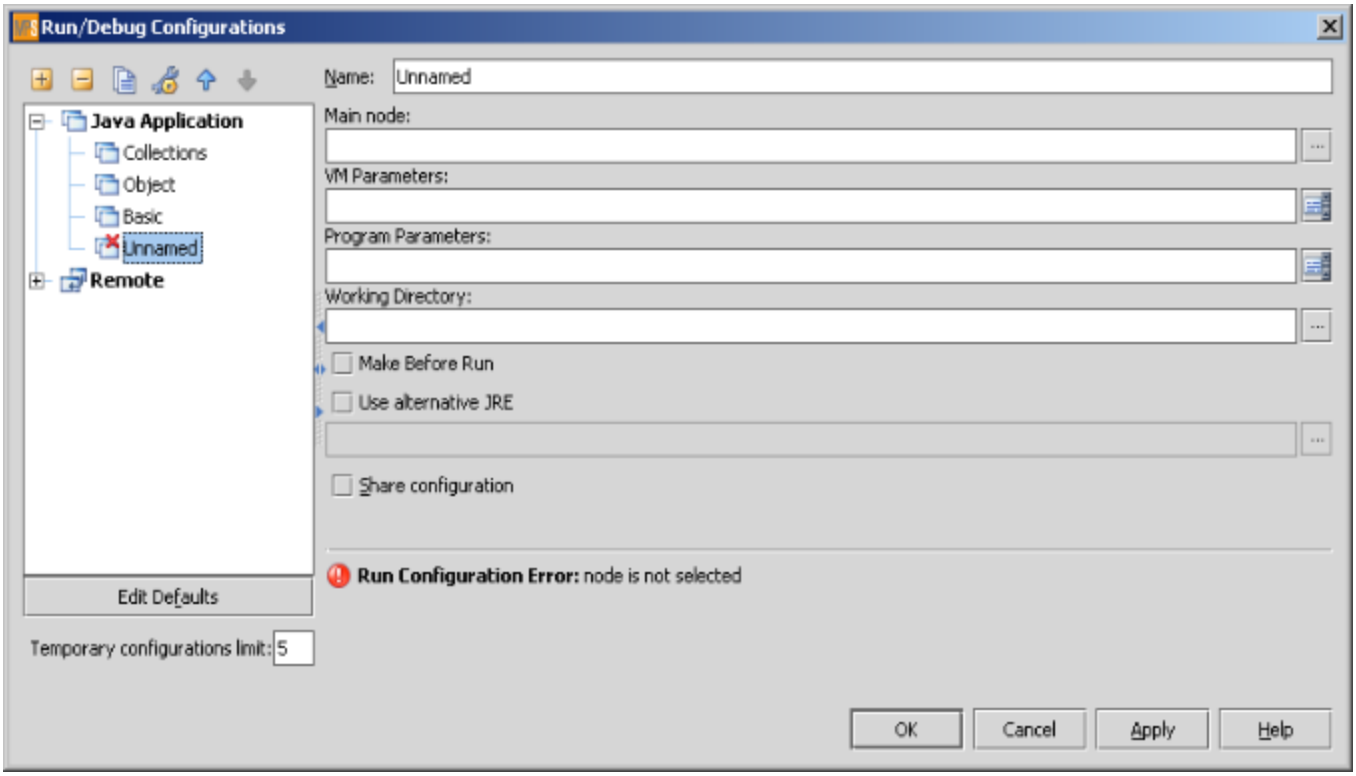
We start with description of how to debug a java application. If a user has a class with main method, a Java Application run configuration should be used to run/debug such a program.

Creating an instance of run configuration

A Java Application run configuration can be created for a class with a main method. Go to Run -> Edit Configurations menu and press a button with "+" as shown at the picture below:



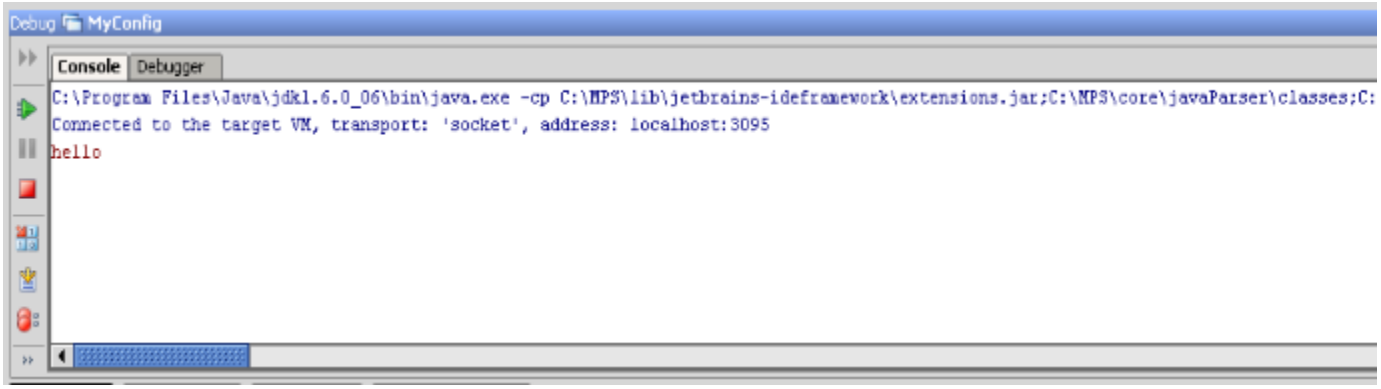
A menu appears, choose Java Application from it and a new Java Application configuration will be created.



A name should be given to this run configuration, and a main node i.e. a class with a main method should be specified. Also VM and program parameters may be specified in a configuration. Look at Run Configuration chapter to learn more about run configurations.

Debugging a configuration

To debug a run configuration, select it from configurations menu and then press Debug button. Debugger starts, and the Debugger tool appears below.



There are two tabs in a tool: one is for the console view and other for the debugger view. In the console an application's output is shown.

Setting a breakpoint

A breakpoint can be set on a statement, field or method. To set or remove a breakpoint, press Ctrl-F8 on a node in the editor or click on a left margin near a node. A breakpoint is marked with a red bubble on the left margin, a pink line inside the editor and a red frame around a node for the breakpoint.

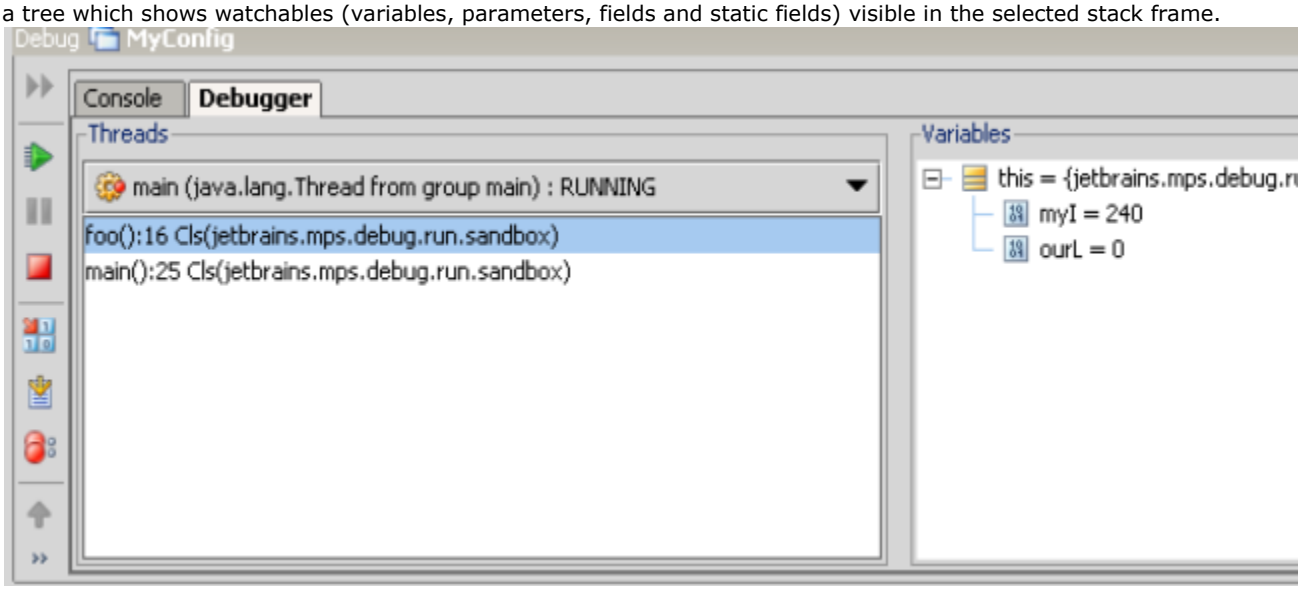
When debugger stops at a breakpoint, the current breakpoint line is marked blue, and the actual node for the breakpoint is decorated with black frame around it.

```
public static void main(string[] args) {
  int i = 1;
  while (i < 6) {
    i++;
  }
  System.err.println("hello");
  new Cls().foo();
}
```

Examining a state of a program at a breakpoint

When at a breakpoint, a Debugger tab can be used to examine a state of a program. There are three controls:

- a combo box to select a thread
- a list of stack frames of the selected thread; the selected frame is marked blue
- a tree which shows watchables (variables, parameters, fields and static fields) visible in the selected stack frame.



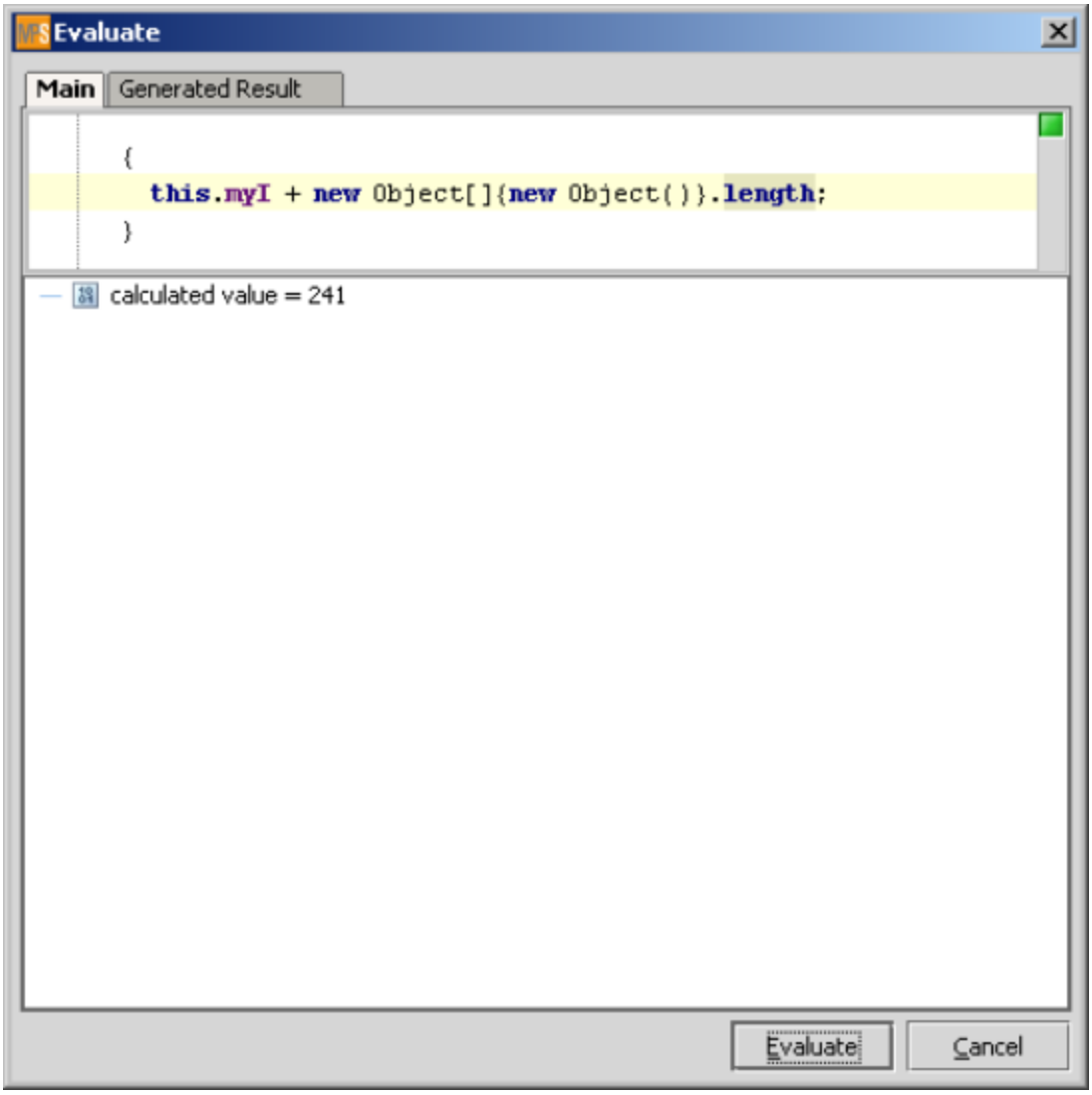
Controlling execution of a program

- To step over, use Run -> Step Over or F8.
- To step out from a method, use Run -> Step Out or Shift-F8.
- To step into a method call, use Run -> Step Into or F7.
- To resume program execution, use Resume button or Run -> Resume or F9.
- To pause a program manually, use Pause button or Run -> Pause. When paused manually i.e. not at a breakpoint, info about variables is unavailable.

Expression evaluation

MPS Java debugger allows user to evaluate expressions during debug, using info from program stack. It is called low-level evaluation, because user is only allowed to use pure java variables/fields/etc from generated code, not entities from high-level source code.

To activate evaluation mode, a program should be stopped at a breakpoint. Press Alt-F8, and a dialog appears.
In a dialog there's a MPS Editor with a statement list inside it. Some code may be written there, which uses variables and fields from stack frame. To evaluate this code, press Evaluate button. The evaluated value will appear in a tree view below.


Evaluate
Main
Generated Result
{
this.myI + new Object[]{new Object()}.length;
}
calculated value = 241
Evaluate
Cancel