



Установка d0sl

Подготовка, установка
SDK, запуск примеров

Предварительные требования

Наши примеры работают с версией MPS **2021.3**, поэтому вы должны сначала установить ее из [здесь](#). Пожалуйста, будьте внимательны и выбирайте версию **2021.3** для ОС вашего компьютера.

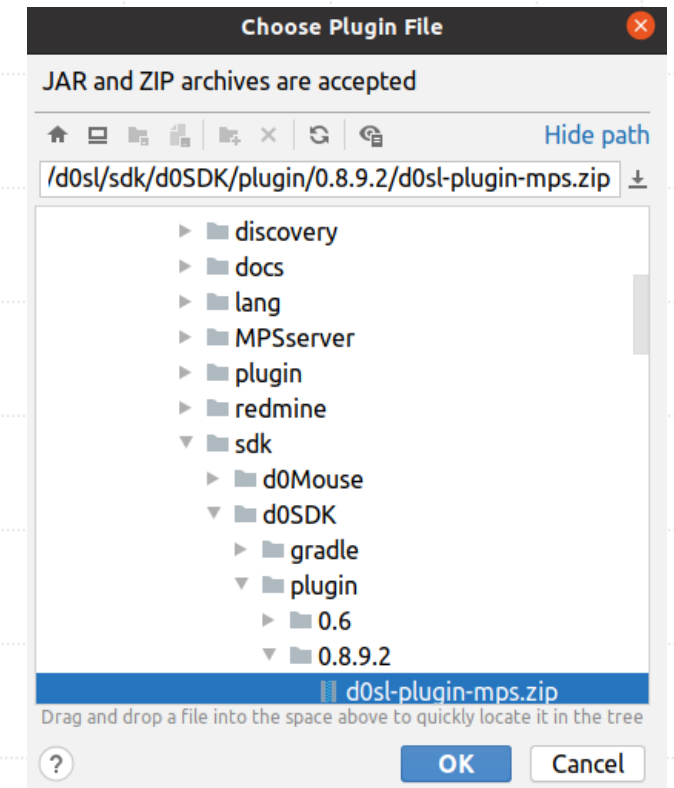
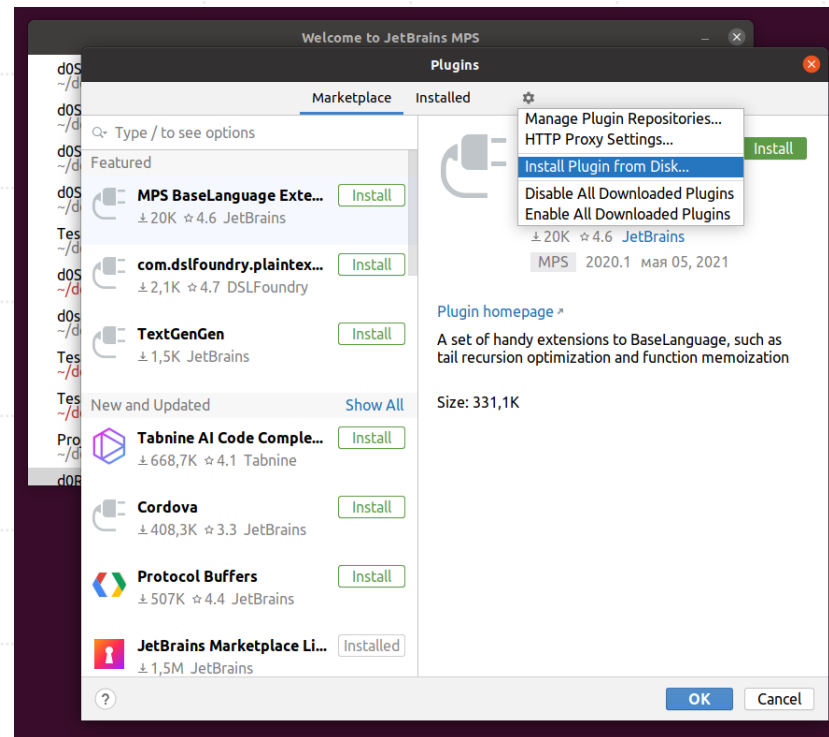
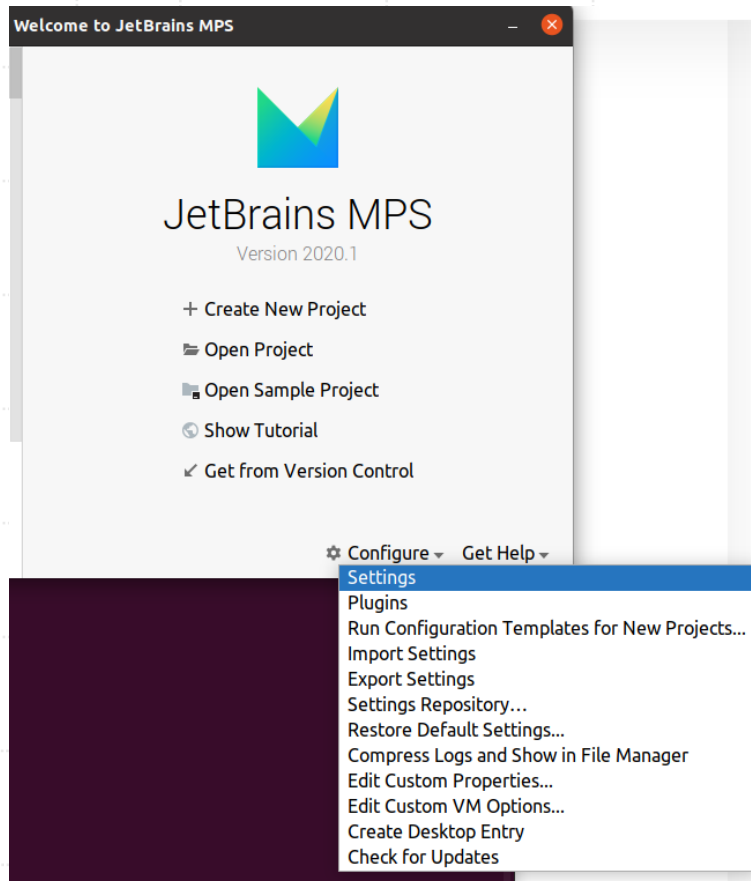


Установка

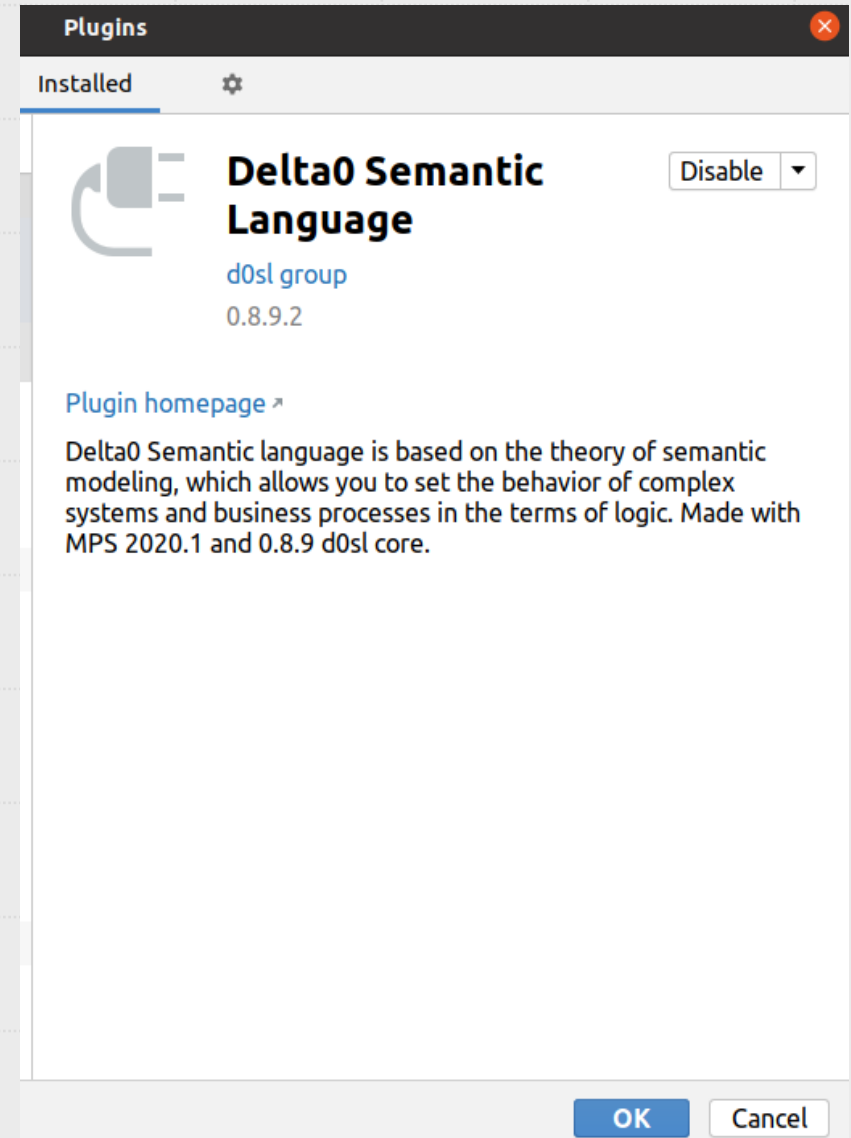
1. Установите систему программирования JetBrains Meta Programming System (MPS) <https://www.jetbrains.com/mps/>.
2. Клонировать git-репо:
 - `git clone https://github.com/d0sl/d0SDK`

Установка плагина d0sl

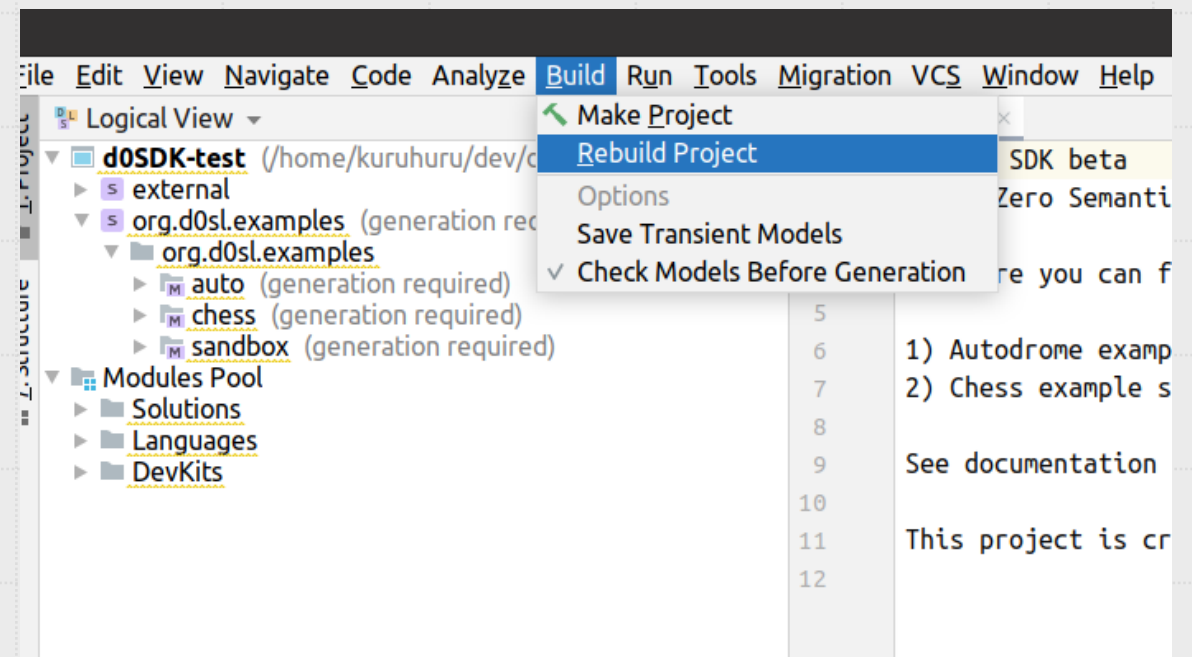
- При первом запуске MPS отображает окно выбора проекта. Но сначала вы выбираете шестеренку в нижней части Configure/plugins.
- Затем выберите установить плагин с диска.
- Затем выберите файл d0SDK/plugin/0.8.9.2/d0sl-plugin-mps.zip



После установки
плагины вы должны
увидеть следующее:



Откройте проект в MPS, выбрав клонированную директорию (например, директорию d0SDK).
Пересоберите проект



Если все прошло
успешно, вы
увидите следующее
окно:

The screenshot shows an IDE window with a project structure on the left and a build log on the right. The project structure is for a project named 'd0SDK-test' located at '/home/kuruhuru/dev/d0sl/sdk/d0SDK-test'. It contains several sub-projects: 'external', 'org.d0sl.examples' (with sub-projects 'auto', 'chess', and 'sandbox'), 'Modules Pool', 'Solutions', 'Languages', and 'DevKits'. The build log shows the following steps and their durations:

- 21:10:36: apply delta changes:2: 2.7 ms
- 21:10:36: weavings:1: 0.0 ms
- 21:10:36: Step 1.2:1: 2.1 ms
- 21:10:36: reductions:1: 2.1 ms (own: 2.0 ms)
- 21:10:36: copy roots:1: 0.0 ms
- 21:10:36: root mappings:1: 0.0 ms
- 21:10:36: weavings:1: 0.0 ms
- 21:10:36: post-processing:6: 2.0 ms
- 21:10:36: pre-processing:6: 1.4 ms
- 21:10:36: analyzing dependencies:1: 10.8 ms
- 21:10:36: generation completed successfully in 898 ms
- 21:10:37: [jetbrains.mps.make.ModuleMaker] Cycle #1: [[org.d0sl.examples [solution]]]
- 21:10:38: [CoreMakeTask] "compile" target execution time: 1371 ms
- 21:10:38: [CoreMakeTask] "textGen" target execution time: 1068 ms
- 21:10:38: [CoreMakeTask] "generate" target execution time: 919 ms
- 21:10:38: [CoreMakeTask] "reloadClasses" target execution time: 123 ms
- 21:10:38: [CoreMakeTask] Other targets execution time: 129 ms; copyTraceInfo: 97 ms, reconcile

The IDE also shows a 'README.md' file with the following content:

```
# d0SL SDK beta
Delta Zero Semantic Lang
Here are you can find th
1) Autodrome example sho
2) Chess example shows h
See documentation (insta
This project is created
```

The IDE status bar at the bottom indicates 'Rebuild successful'.

Головоломка восьми ферзей

Головоломка восьми ферзей – это проблема размещения восьми шахматных ферзей на шахматной доске 8×8 так, чтобы никакие два ферзя не угрожали друг другу.

Проблема поиска всех решений задачи о восьми ферзях может быть достаточно вычислительно затратной, поскольку существует 4 426 165 368 возможных расположений восьми ферзей на доске 8×8, но только 92 решения.

Семантический способ решения этой проблемы заключается в следующем:

Представьте, что у нас есть робот, который может произвольно расставить восемь ферзей на шахматной доске. Но он не знает, правильная это позиция или нет.

Затем, после каждой расстановки, робот спрашивает нас, правильно ли он расположил ферзей на доске. Если мы отвечаем, что правильно, робот считает задание выполненным. Если же мы говорим, что неправильно, робот пытается расположить ферзей по-другому.

Для того чтобы определить, правильно ли расставлены ферзи на шахматной доске, мы напишем семантические правила на языке d0sl. Например: Для любых двух ферзей на шахматной доске должно быть верно, что они не находятся на одной диагонали и не находятся на одной линии.

```
def check board(board : ChessBoard) means
  check all
    var queens = ChessDSL.get queens(board)
    for all q1, q2 in queens
      not ChessDSL.on one line(q1, q2) and
      not ChessDSL.on one diagonal(q1, q2)
    end
end def
```

copy

Чтобы запустить пример с шахматами
Найдите семантическую модель
ChessAll в org.d0sl.examples/sandbox.
Откройте на ней контекстное меню,
нажав правую кнопку мыши. И
выберите Run Node ChessAll.

The screenshot shows an IDE window with a 'Logical View' of a project. The project structure is as follows:

- examples-d0sl (D:\Dev\MPS\examples-d0sl)
 - org.d0sl.examples
 - auto
 - chess
 - sandbox
 - AutodromeAII
 - AutodromeDSL
 - ChessAll**
 - ChessDSL
 - Math
 - MathTest
- Modules Pool

A context menu is open over the 'ChessAll' node, listing various actions. The 'Run' option is highlighted in blue.

```
def check board2(board : ChessBoard) means
check all
  var queens = ChessDSL.get queens(board)
  for all q1, q2 in queens
    not ChessDSL.on one line(q1, q2)
end
end def
```

The context menu items include:

- Preview Generated Text (Ctrl+Alt+Shift+F9)
- Set Virtual Package...
- Expand All
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Cut (Ctrl+X)
- Copy Reference (Ctrl+Shift+C)
- Copy Node QName
- Copy Node Reference as URL
- Delete (Delete)
- Safe Delete (Alt+Delete)
- Clone Root (Shift+F5)
- Show Node in Explorer (Alt+X)
- Language Debug
- Find Usages (Alt+F7)
- Find Usages Settings... (Ctrl+Alt+Shift+F7)
- Refactoring
- Help
- Compare with the Same Repository Version
- Run 'Node ChessAll' (Ctrl+Shift+F10)**
- Debug 'Node ChessAll'

Чтобы изменить семантику

- Семантическая модель определена в песочнице/ChessAll. Когда робот расставляет ферзей, он вызывает предикат `check board`. В семантической модели заранее подготовлено несколько различных предикатов `check board`. Просто измените имя текущего предиката `check board`, а другой (например, `check board1`) переименуйте в `check board`. И вы увидите, как меняется поведение робота в зависимости от заданной семантики.
- Например, следующая модель запретит роботу ставить ферзей на одну линию, но он сможет расположить их по одной диагонали.

```
def check board2(board : ChessBoard) means
  check all
    var queens = ChessDSL.get queens(board)
    for all q1, q2 in queens
      not ChessDSL.on one line(q1, q2)
    end
end def
```

copy

Warning

После каждого изменения модели не забывайте пересобирать решение MPS (Ctrl+F9).