Agenda

- MKT insights
- Editor
- Generator
- Internal project
MKT insights

- 2017
- 2018
- Overview
History

MPS project was started ~ at 2002
MPS 1.0 was released at 2009
MPS 2018.1 was released on April 2018
MPS Team

- 12 Developers
- 1 Tester
- 1 Product MKT Manager
- Located in 3 offices
  - St-Petersburg
  - Prague
  - Munich
mps-support@jetbrains.com
MPS Users
2017

- Communication
- Promotion
2017

- Communication
- Promotion
Language Designer
Language Designer

End-user
MPS video
2017

- Communication
- Promotion
Events 2017

- PLDI-WCIRE in Barcelona.
- Siemens Day in Munich.
- EclipseCon in Stuttgart.
- SECR event in Saint Petersburg.
- JetBrains Meetup in Moscow.
- GeeCON in Prague.
- Java Days in Prague.

+100 events sponsored by JetBrains
SEO insights

Domain-specific languages  10th-20th place
Language workbench       Top 10
Projectional editor       Top 10
Social networks

Subreddit: /nosyntax
Twitter analytics

![Twitter analytics chart]

@JetBrains. All rights reserved
MPS Downloads
MPS Downloads
2018

- Promotion
- Building community
- Education
2018

- Promotion
- Building community
- Education
Events 2018

- LangDev Meetup, Amsterdam (March 2018)
- Java Days Istanbul (May 2018)
- ...

Calendars for releases and events.
Empowering the Community

http://mps.rocks
2018

- Promotion
- Building community
- Education
Building Community

- MPS Community Meetup
- Everything you need to know about MPS
2018

- Promotion
- Building community
- Education
JetBrains MPS Introductory online course
By Vaclav Pech
Education

ProgrammingBasics
How to think like a programmer.

What is this?
This is a tutorial on how to think like a programmer, and to learn some programming along the way. It teaches you fundamental ideas and concepts present in all programming systems, from "real" programming languages over scripting languages and configuration files to domain-specific languages.

Table of Contents
Part 1: The Basics
1. Variables and Expressions
2. Defining Programs
3. Types
4. Functions
What can you do to help MPS?
Help:

- Ask/Answer questions on Stack Overflow and Forum
- Share your projects/knowledge
- Contribute
- Contact us:
Help:

• Ask/Answer questions on Stack Overflow and Forum
• Share your projects/knowledge
• Contribute
• Contact us:
Help:

- Ask/Answer questions on Stack Overflow and Forum
- Share your projects/knowledge
- Contribute
- Contact us:
Help:

• Ask/Answer questions on Stack Overflow and Forum
• Share your projects/knowledge
• Contribute
• Contact us

MPS Blog

mps-support@jetbrains.com
What to expect in the near future

• New type system
• Generation plans
• Generated languages
• Improve RCP story
MPS Editor Features
From 2017.1 to 2018.1

Simon Alperovich
Node URL (inspired by Itemis)

Example use case:

`interpolatorInstances` in mbeddr tutorial (node URL http://127.0.0.1:63320/node?ref=r%3Abd5ec23c-c294-47cc-a078-675c03abdb88%28mbeddr.tutorial.main.defaultExtensions%29%2F1731059994647378529)
Shiftless code completion

![Diagram of code completion]

*JetBrains. All rights reserved*
Two step deletion

On button: 1 --> Technical Department
On button: 2 --> Marketing Department
On button: 3 --> Others

Activity: Technical Department on button: 1
On button: 1 --> Hardware
On button: * --> Previous Menu

Activity: Hardware on button: 1
① Get info √

Activity: Previous Menu on button: *
① Get info √
Completion item trace

Implicit transformation menu for BuildTextStringPart: include menu for the superconcepts
Implicit transformation menu for BuildStringPart: include menu for the superconcepts
default transformation menu for BaseConcept
include substitute menu for the link target concept: BuildStringPart
default substitute menu for BuildStringPart
parameterized substitute menu part

Substitute item: $build.number. Parameter object: build.number
Generic placeholders and free floating comments

On button: 1  -->  Technical Department
On button: 2  -->  Marketing Department
On button: 3  -->  Others

[ We should add button 4 here according to the specification ]
Customize font family for cell

```html
style Heading {
    font-style: bold
    font-size: 100
    font-family: Comfortoo
}
```

On button: 1 --> Internet
On button: 2 --> Payment
Reflective editor for the subtree

```java
build variable macro date {
    initial value:
    build variable macro init with date {
        pattern: yyyyMMdd
    }
}
```
Our goals

- Improving the editor experience
- Integrating the MPS extensions such as tables, grammar cells and linear selection
Thank you for your attention —
The goal of the project

Intellectual assistance in the biological research space

- Use DSLs to express knowledge in the area of biological research
- Establish known facts lookup (query language)
  - Show the most important / relevant facts first
- Link most of the available facts with source of information
- Integrate it with the research data sources
The Biological Expression Language (BEL) is a language for representing biological observations in a computable form, along with contextual information.
Biological Expression Language (BEL)

Other BEL frameworks:
- Biological Expression Language Information Extraction WorkFlow (BELIEF) fraunhofer.de
- Open BEL openbel.org
• Custom model persistence using the BEL syntax
• Projectional editor for BEL
• Intentions & necessary validations helping non-experts to use BEL
• No code generation
MPS BEL Editor

• We started with BEL implementation in MPS
• Existing BEL editors are far behind the modern IDEs
• Partners gave us positive feedback on our prototype
• BEL is covering only some limited space of biological facts area
• We are thinking about another language, BetterThanBel (BTB)
• BTB is not even prototyped yet
Query language

• Should be a core DSL of the knowledge base
• There should be a query engine executing queries
• Probably we will use some AI techniques in order to process queries
• We will start working on this language as soon as we have a prototype of BTB language
- Create a plan of a future experiment
- Log all experimental data during the experiment execution
- Later use this data for some analysis
- Provide assistance in making some conclusions from the experimental data (derive some facts on BTB)
- Introduce additional facts into the knowledge base
- Link new facts with the experiment plan & log (data)
LabBook DSL ontology library

```plaintext
ontology_library CommonCratology
    ontology_type fluid super-types: << ... >>

ontology_type blood super-types: fluid

ontology_type cellular organism super-types: << ... >>

ontology_type mammal super-types: cellular organism

ontology_type homo sapiens super-types: mammal, cellular organism
```
LabBook DSL

system of units

---

library SystemOfUnits

- Should we call it "primitive/property/data types"?...
  - value float - measurement unit
  - basic physical units
    - motion length
    - measured in: km, m, dm, cm, mm, μm, nm, in, ft, mi
    - motion weight
    - measured in: Nt, t, kg, g, mg, μg, n
    - motion volume
    - measured in: km³, m³, l
    - motion temperature
    - measured in: K, °C, °F
    - motion time
    - measured in: y, h, min, s, ms, μs, ns
    - motion Line of day
    - measured in: HH, HH:mm, HH:mm:ss
    - motion density
    - measured in: kg/m³, kg/l, g/ml, g/cm³
    - motion probability
    - measured in: percent, prob

- Enumerations
  - enumerable motion sex
    - values: male, female, hermaphrodite
  - enumerable motion color
    - values: white, blue, green, yellow, brown
  - enumerable motion blood kind
    - values: umbilical cord, arterial, venous, capillary

- DataTypes
  - motion string
    - measured in: << ... >>
  - motion boolean
    - measured in: << ... >>
  - motion integer
    - measured in: << ... >>
LabBook DSL
properties

property library CommonProperties

[ physical properties of physical objects ]
<derived name> : volume
<derived name> : temperature
<derived name> : weight
height : length
width : length
<derived name> : length
<derived name> : color

[ Fluids ]
density : density
<derived name> : blood kind

[ Livings ]
age : time
gender : sex
hair color : color
foot size : length
name : string
LabBook DSL
sample declaration

```
sample_library MyFirstExperimentSamples

sample Person (person) extending: <no extendedSample>
ontology elements: homo sapiens

identified by property name

properties:
  age : time
  gender : sex
  weight : weight
  name : string

references:
  << ... >>

sample Blood (blood) extending: <no extendedSample>
ontology elements: blood

identified by reference to owner

properties:
  blood kind : blood kind
  volume : volume

references:
  owner : Person
```
LabBook DSL
action declaration

```
action_library MyFirstExperimentActions

action extractBloodSample(Person person, volume vol) -> Blood

// Set the blood sample of the specified person (specified amount of ml) & fix a
result.owner = person
result.bloodKind = venous
result.volume = vol

action splitUpBloodSample(Blood sample) -> [Blood, 4]

// Split up each blood sample onto 4 samples in a proportion 1/4, 1/4, 1/8, 3/8
result.size = 4
result[*].bloodKind = sample.bloodKind
result[*].owner = sample.owner
result[0].volume = "0.05"
result[1].volume = "0.05"
result[2].volume = "0.025"
result[3].volume = "0.075"

action freeze(Blood sample, temperature t) -> FrozenBlood

// Freeze passed blood sample down to the specified temperature
```
plan of experiment MyFirstExperimentPlan

description: In this experiment we will try to use as much features of MPS labbook language as it possible

input samples:
  [Person] oldPersons
  [Person, 5] youngPersons

output samples: <todo....>

steps:

  /*Extract blood samples*/
  step Step1:
    input: oldPersons, youngPersons

  /*Split up blood samples*/
  step Step2:
    input: oldBloodSamples, youngBloodSamples
    output: [Blood] oldBloodSamplesArray, [Blood] youngBloodSamplesArray

conditions:
  << ... >>

measurements:
  << ... >>

actions:
  oldBloodSamplesArray = oldBloodSamples.splitUpBloodSample(tableElement)
  youngBloodSamplesArray = youngBloodSamples.splitUpBloodSample(tableElement)
LabBook DSL experiment instance
LabBook DSL prototype

- Comfortable editing environment guiding the user through entering data
- Simple expressions language
- Interpreter for expressions
- Not using code generation
- The prototype has no type system
- Underlying models used for:
  - Future analysis
  - Extracting higher-level facts (dependencies)
  - Revisiting experiment results in case of future questions
X reasons to develop DSL in MPS

• Provide the team with positive feedback
• Better understand the requirements of DSL designers
• Find usability problems
• Submit bug reports
• Formulate feature requests
MPS-extensions

- The MPS-extensions project has been separated from the mbeddr code base
- The project is available on GitHub
- We have a feature branch of LabBook with the dependency on this project
- We should create a build of MPS-extensions on MPS build server
Thank you for your attention