DSL быстро и просто

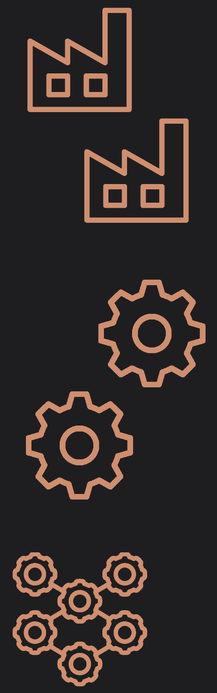
Опыт создания языка с JetBrains MPS

Луговой Сергей, 2026

Луговой Сергей

- C/C++, АРМы, СУБД
- Java с первой версии
- Telecom: SMSC/USSDC, сервисы
- Системы обработки данных

Предыстория



Kafka

RabbitMQ

:// REST

Cassandra

Ignite

Hadoop

Декларативный подход?

```
{
  "selectors": [
    {"type":"SchemaSelector", "knownSteps":["SEND", "RECEIVE", "CANCELLED"]}
  ],
  "schemes" : {
   "SEND" : {
      "transformers": [
        {
          "type":"SaveOperation", "store":"sample", "collection":"operative",
          "id":"/uin", "kind":"/schema", "timestamp":"/cDate",
        }
      ],
      "plan":[
        {"type":"id", "value":"SEND-${/uin}"},
        {"type":"timestamp", "value":"/cDate", "pattern":"yyyy-MM-dd'T'HH:mm:ssXXX"},
        {"type":"eval", "to": "/time", "class": "long", "exp": "#now()"},
        {"type":"archive", "path":"/", "archive": "hadoop", "parameters": {"pretty": false} },
        {"type":"include",  "domain":"other", "schema":"send"}
      ]
    },
    "RECEIVE" : {
```

Императивный подход!

```
accounting plan processing
{
  collect event id from @/uin
  collect event time from @/cDate
  set @/accountingTime value #instant.now
  switch on @/step {
    in case 'SEND' save operation @/ to 'sample.operations' id @/uin kind 'SEND'
    in case 'RECEIVE' load operation from 'sample.operations' as saved id @/uin
        on complete exec transferReceived
    in case 'CANCEL' load operation from 'sample.operations' as saved id @/uin
        on complete exec transferCancelled
    by default throw InvalidData
  }
  archive @/ to 'hadoop'
}
```

Предметно-ориентированный язык

Внутренний DSL

Java, Kotlin, Groovy

Расширяемый синтаксис + своя семантика

```
dependencies {
    implementation "org.apache.logging.log4j:log4j-api:${log4jVersion}"
    compileOnly "org.projectlombok:lombok:${lombokVersion}"
    annotationProcessor "org.projectlombok:lombok:${lombokVersion}"
}

tasks.register('copyLanguageStructure', Copy) {
    from(rootDir) { include "languages/Plang/models/Plang.structure.mps" }
    into "${sourceSets.main.output.resourcesDir}"
}

processResources {
    dependsOn 'copyLanguageStructure'
}

jar {
    manifest {
        attributes "Language-Concepts-Packages": ["com.example.plang"]
        attributes 'Language-Structure-File': structureFile
    }
}
```

```
val queryBuilder = QueryBuilder
    .selectFrom(keyspaceId, tableId)
    .columns("first", "second", "last")
    .limit(100)
    .whereColumn("from").isGreaterThanOrEqualTo(literal(begin))
    .whereColumn("from").isLessThan(literal(end))
    .orderBy("last", ClusteringOrder.DESC)
    .allowFiltering();

for ((day, lesson, teacher, student) in scheduledEvents) {
    val teacherSchedule: Schedule = teacher.schedule
    teacherSchedule[day, lesson] shouldNotEqual null
    teacherSchedule[day, lesson]!!.student shouldEqual student
    val studentSchedule = student.schedule
    studentSchedule[day, lesson] shouldNotEqual null
    studentSchedule[day, lesson]!!.teacher shouldEqual teacher
}
```

Предметно-ориентированный язык

Внутренний DSL

Java, Kotlin, Groovy

Расширяемый синтаксис + своя семантика

Текстовый DSL

Lexer, Parser, Interpretator, Generator, IDE plugin

Сложно и долго в создании языка и инструментов

```
SELECT first, second, last
FROM some_table
WHERE from >= ? and to < ?
ORDER BY last DESC
FETCH FIRST 100 ROWS ONLY
```

```
SELECT:                 S E L E C T;
FROM:                   F R O M;
UNSIGNED_INTEGER:       [0-9]+;
CHAR_STRING:            '\'' (~('\'' | '\r' | '\n') | '\'' '\'' | NEWLINE)* '\'';

select_statement
    : (WITH factoring_element (',' factoring_element)*)?
      subquery (for_update_clause | order_by_clause)*;

subquery
    : subquery_basic_elements subquery_operation_part*;

subquery_basic_elements
    : query_block | '(' subquery ')';

query_block
    : SELECT (DISTINCT | UNIQUE | ALL)? selected_fields into_clause?
      from_clause where_clause? hierarchical_query_clause?
      group_by_clause? model_clause?;
```

Предметно-ориентированный язык

Внутренний DSL

Java, Kotlin, Groovy

Расширяемый синтаксис + своя семантика

Текстовый DSL

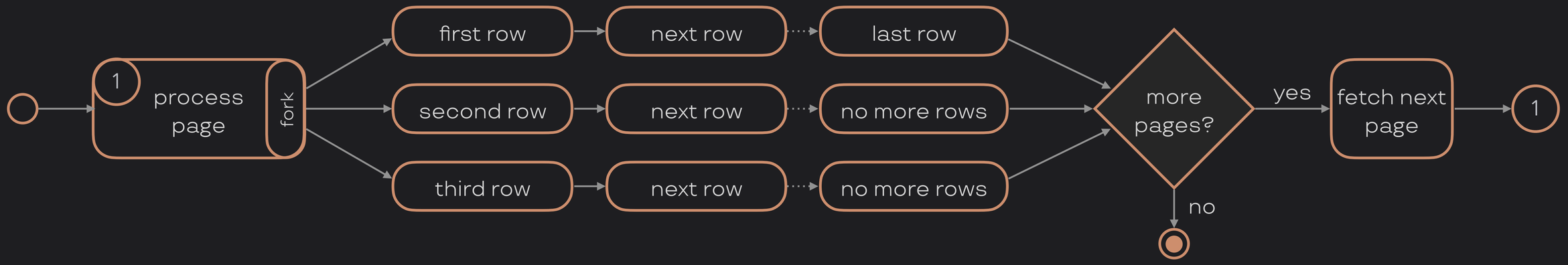
Lexer, Parser, Interpretator, Generator, IDE plugin

Сложно и долго в создании языка и инструментов

Графический DSL

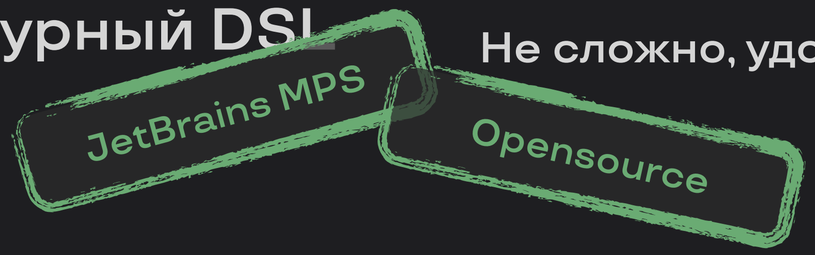
Visual modeling

Очень не удобен для императивного подхода



Предметно-ориентированный язык

Внутренний DSL	Java, Kotlin, Groovy Расширяемый синтаксис + своя семантика
Текстовый DSL	Lexer, Parser, Interpretator, Generator, IDE plugin Сложно и долго в создании языка и инструментов
Графический DSL	Visual modeling Очень не удобен для императивного подхода
Структурный DSL	Tree/graph modeling Не сложно, удобно. Autocompletion практически из коробки.



JetBrains MPS

IDE для создания и использования DSL

```
concept SwitchStatement extends    BaseConcept
                        implements Statement

  alias: switch
  short description: Switch statement

  children:
  expression : Expression[1]
  cases      : SwitchCase[1..n]
  null       : SwitchNull[0..1]
  default    : SwitchDefault[0..1]


concept SwitchCase extends    StatementsBlock
                   implements SwitchElement

  alias: in case
  short description: Case branch

  children:
  case : ConstantExpression[1..n]
```

```
switch on quantity {
  in case 1 do println "one"
  in case 2 do println "two"
  in case 3 4 5 do println "few"
  in case 6 7 8 9 do println "some"
  for null do println "nothing"
  by default do println "many"
}
```

Проекционный редактор

```
list<Integer> list = new arraylist<Integer>{1, 2, 3, 4};

System.out.println(String.valueOf((∑ [[1 k 0] [0 1.0 0] [0 0 1]] )));

matrix<Double> r = [[cos(a) sin(1) 1] [2/3 1 3 + 1.0/2] [MAX 3 * ln(e) 7 - a/2 + b exp(1)] [4 0 ∏ n]];
```

Генерация

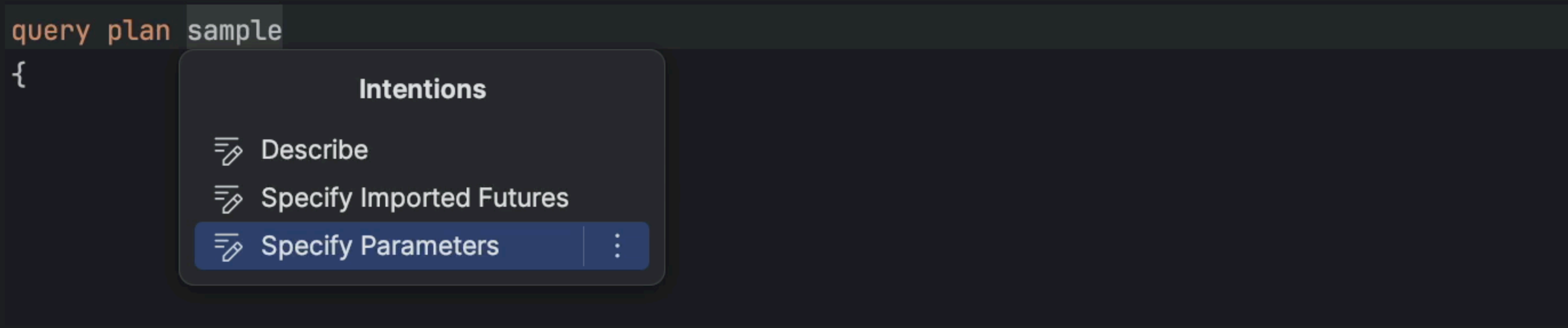
```
switch on quantity {
  in case 1 do println "one"
  in case 2 do println "two"
  in case 3 4 5 do println "few"
  in case 6 7 8 9 do println "some"
  for null do println "nothing"
  by default do println "many"
}
```

```
{
  "plan": [
    {"type": "switch", "expression": "quantity",
      "cases": [
        {"constants": [1], "do": [
            {"type": "println", "expression": "\"one\""}
          ]},
        {"constants": [2], "do": [
          {"type": "println", "expression": "\"two\""}
        ]},
        {"constants": [3, 4, 5], "do": [
          {"type": "println", "expression": "\"few\""}
        ]},
        {"constants": [6, 7, 8, 9], "do": [
          {"type": "println", "expression": "\"some\""}
        ]}
      ],
      "null": [
        {"type": "println", "expression": "\"nothing\""}
      ],
      "default": [
        {"type": "println", "expression": "\"many\""}
      ]
    }
  ]
}
```

Помощь при редактировании

```
query plan sample
{
```

Помощь при редактировании



Помощь при редактировании

```
query plan sample
parameters [fio // full name]
{
```

Помощь при редактировании

```
query plan sample
parameters [ fio // full name ]
{
  query
```

```
N query accumulator      Query accumulated event data
N query activity             Query subject activities
N query counters             Query collected counters
N query document                Query stored document
N query profile  Query profile by uid or unique index
N query relations             Query subject relations
```

Press ⌥⌘B to Show item trace

Помощь при редактировании

```
query plan sample
parameters [ fio // full name ]
{
  query profile customer using
    retrieve [      : * ]
```

Помощь при редактировании

```
query plan sample
parameters [ fio // full name ]
{
  query profile customer using
    retrieve [      : * ]
```

```
N search  Profile search criteria
N uid   Profile unique identifier

Press ⌥⌘B to Show item trace
```

Помощь при редактировании

```
query plan sample
parameters[fio // full name]
{
  query profile customer using search in person properties [fullName : fio autosense]
    retrieve ['person' : *]
```

Помощь при редактировании

```
query plan sample
parameters[fio // full name]
{
  query profile customer using search in person properties [fullName : fio autosense]
    retrieve ['person' : *]
    on complete set @/profile value customer
```

Помощь при редактировании

```
query plan sample
parameters[fio // full name]
{
  query profile customer using search in person properties [fullName : fio autosense]
    retrieve ['person' : *]
    on complete {
      set @/profile value customer
      query counters 'sent' as cnt from 'transfers' limited by this month
        where['id': @customer/uid]
        on complete set @/sent value cnt
    }
```

Элементы MPS

Элементы MPS

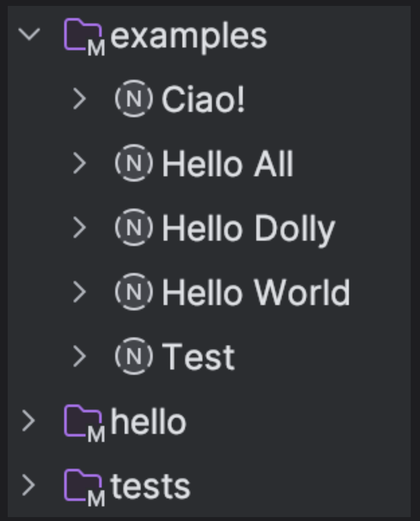
Concept

```
concept IfStatement extends    BaseConcept
                    implements Statement

  instance can be root: false
  alias: if
  short description: if-then-else statement

  properties:
  << ... >>

  children:
  condition : BooleanExpression[1]
  then      : StatementsBlock[1]
  else      : StatementsBlock[0..1]

  references:
  << ... >>
```

Элементы MPS

Concept

Model

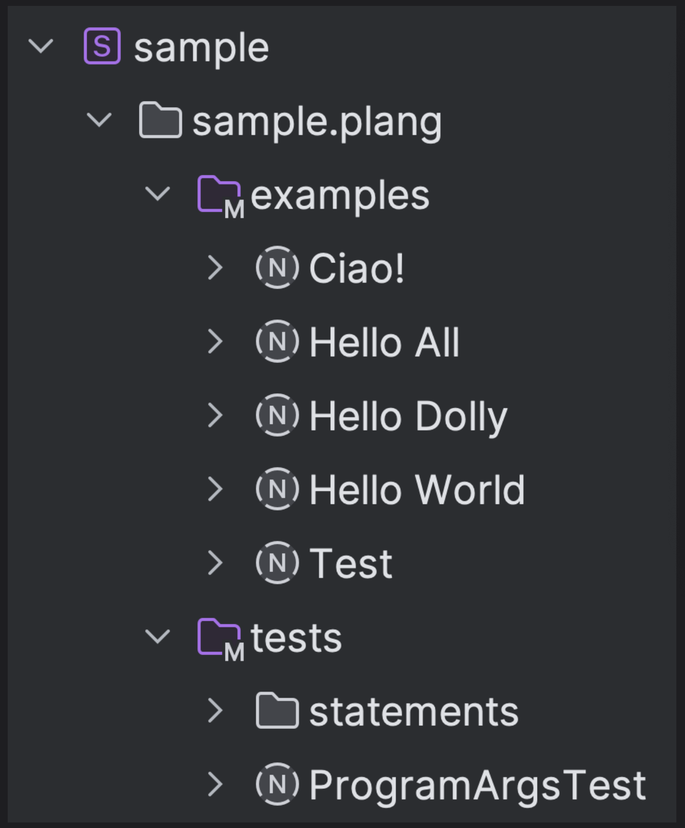


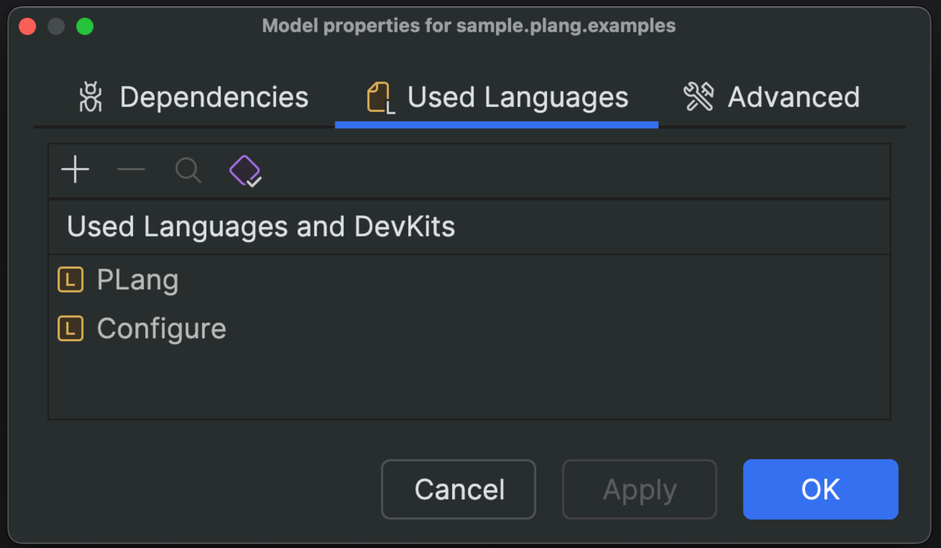

```
program Hello Dolly {
  println "Hello Dolly"
}
```

```
program Test
args[some // something boolean]
{
  if some then println "true"
  else println "false"
}
```

Элементы MPS

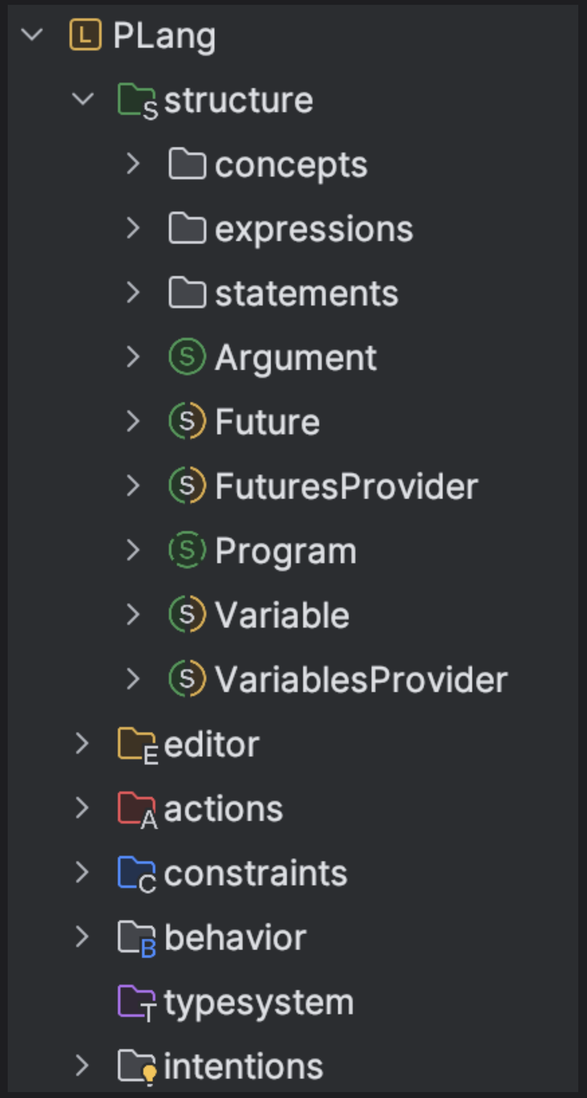
- Concept
- Model
- Solution





Элементы MPS

- Concept
- Model
- Solution
- Language




```
concept Program extends    StatementsBlock
                implements INamedConcept

  instance can be root: true
  alias: <no alias>
  short description: <no short description>

  properties:
  << ... >>

  children:
  arguments : Argument[0..n]

  references:
  << ... >>
```

Language aspects

structure	Структура языка: Concept, Interface concept их свойства, ссылки и потомки
editor	Представление концептов и способы создания и модификации их
actions	Кастомизация создания узлов, копирования и вставки
constraints	Дополнительные условия применимости концептов
behavior	Конструкторы, методы, утилитарные функции концептов
typesystem	Правила касающиеся типов элементов языка
intentions	Быстрый доступ к наиболее часто используемым операциям с концептами
findUsages	Кастомизация поиска связанных элементов
dataFlow	Поиск недостижимых инструкций, неиспользуемых элементов, и т.д.
test	Тестирование структуры языка
textGen	Кастомизация текстовой генерации
migration	Миграции версий языка
generator	Генерация кода

Структура языка
Выражения и инструкции

Выражения

```
interface concept Expression extends <none>
```

```
  properties:
  << ... >>

  children:
  << ... >>

  references:
  << ... >>
```

Выражения

```
interface concept BinaryOperation extends Expression
```

```
  properties:
  << ... >>

  children:
  left : Expression[1]
  right : Expression[1]

  references:
  << ... >>
```

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    <choose cell model>

  inspected cell layout:
    <choose cell model>
```

Cell Collection | What's this?

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    <choose cell model>

  in

Cell
```

Intentions

- Surround with Horizontal Collection
- Surround with Indent Collection
- Surround with Vertical Collection
- Generate Default (Expression-like)
- Generate Default (Statement-like)
- Extract Component ⌥⌘C

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    [> <choose cell model> <]

  inspected cell layout:
    <choose cell model>
```

Concept Alias | left Link | What's this?

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    [> % left % <choose cell model> <]

  inspected cell layout:
    <choose cell model>
```

Concept Alias | right Link | What's this?

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    [> % left % # alias # <choose cell model> <]

  inspected cell layout:
    <choose cell model>

right Link What's this?
```

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    [> % left % # alias # %ri <]

  inspected cell layout:
    <choose cell model>

New Cell | What's this?
```

```
N %ri                                          text constant
l %right%    ^linkDeclaration (PLang.structure.BinaryOperation)
l %right%->  ^linkDeclaration (PLang.structure.BinaryOperation)
Press ⌥⌘B to Show item trace
```

Выражения

```
<default> editor for concept BinaryOperation
  node cell layout:
    [> % left % # alias # % right % <]
```

Style:

```
Operator {
  editable : true
}
```

Common:

action map	<default>
keymap	<default>
menu	<none>
transformation menu	<none>
attracts focus	noAttraction
show if	<no condition>

Выражения

```
@doc  Суммирует числа, склеивает строки, добавляет в коллекции
concept PlusOperation extends    <default>
                      implements BinaryOperation

  instance can be root: false
  alias: +
  short description: Add/concat/append
```

Инструкции

```
interface concept Statement extends <none>
```

properties:
<< ... >>

children:
<< ... >>

references:
<< ... >>

Инструкции

```
concept IfStatement extends    BaseConcept
                    implements Statement

  instance can be root: false
  alias: if
  short description: if-then-else statement

  properties:
  << ... >>

  children:
  condition : BooleanExpression[1]
  then      : StatementsBlock[1]
  else      : StatementsBlock[0..1]
```

Инструкции

```
<default> editor for concept IfStatement
  node cell layout:
    [- if % condition % then % then % ?[- else % else % -] -]
```

```
show if (editorContext, node)->boolean {
    node.else != null;
}
```

Инструкции

```
concept StatementsBlock extends    BaseConcept

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  showBraces : boolean

  children:
  statements : Statement[0..n]
```

Инструкции

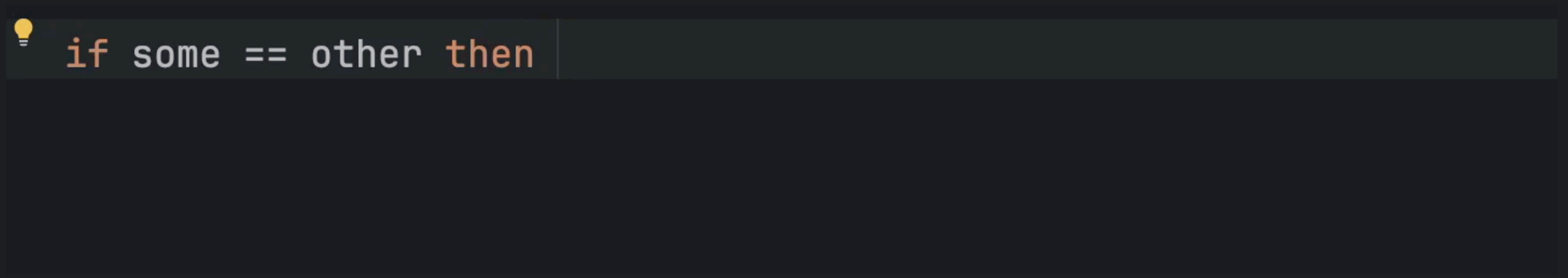
```
<default> editor for concept StatementsBlock
  node cell layout:
    [-
    ? {
    (- % statements % ↵ /empty cell: <default> -)
    ? }
    -]
```

Style:

```
<no base style> {
  indent-layout-indent : true
  indent-layout-on-new-line : (editorContext, node)->boolean {
                                node.showBraces || node.statements.size > 1;
                              }
  indent-layout-new-line-children : true
}
```

Intentions

Intentions

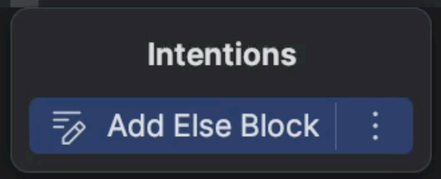


Intentions

```
if some == other then println "Equal"
else
```

Intentions

```
if some == other then println "Equal"
```



Intensions

```
intention IfElse for concept IfStatement {

  description(node, editorContext)->string {
    "Add Else Block";
  }

  isApplicable(node, editorContext)->boolean {
    node.else == null;
  }

  execute(node, editorContext)->void {
    node.else = new node<StatementsBlock>();
  }
```

Подстановки

Подстановки

```
concept EmptyLine extends     BaseConcept
                  implements Statement


<default> editor for concept EmptyLine
  node cell layout:
    <constant>


Style:
Placeholder {
  << ... >>
}
```

Подстановки

```
substitute menu for concept EmptyLine : default

substitute action(output concept: Statement)
```

Подстановки

```
substitute menu for concept EmptyLine : default

substitute action(output concept: Statement)
  create node (parentNode, currentTargetNode, link, editorContext, model, pattern)->node<Statement> {
    node<IfStatement> ifStatement = currentTargetNode.prev-sibling:IfStatement;
    ifStatement.else = new node<StatementsBlock>();
    ifStatement.else.select[in: editorContext, cell: LAST_EDITABLE];
    return ifStatement;
  }
```

Подстановки

```
substitute menu for concept EmptyLine : default

substitute action(output concept: Statement)
  create node (parentNode, currentTargetNode, link, editorContext, model, pattern)->node<Statement> {
    node<IfStatement> ifStatement = currentTargetNode.prev-sibling:IfStatement;
    ifStatement.else = new node<StatementsBlock>();
    ifStatement.else.select[in: editorContext, cell: LAST_EDITABLE];
    return ifStatement;
  }
  can substitute (parentNode, currentTargetNode, link, editorContext, model, pattern, strictly)->boolean {
    if (pattern :ne: "else") { return false; }
    node<> prev = currentTargetNode.prev-sibling;
    return prev.isInstanceOf(IfStatement) && prev:IfStatement.else == null;
  }
```

Подстановки

```
substitute menu for concept EmptyLine : default

substitute action(output concept: Statement)
  create node (parentNode, currentTargetNode, link, editorContext, model, pattern)->node<Statement> {
    node<IfStatement> ifStatement = currentTargetNode.prev-sibling:IfStatement;
    ifStatement.else = new node<StatementsBlock>();
    ifStatement.else.select[in: editorContext, cell: LAST_EDITABLE];
    return ifStatement;
  }
  can substitute (parentNode, currentTargetNode, link, editorContext, model, pattern, strictly)->boolean {
    if (pattern :ne: "else") { return false; }
    node<> prev = currentTargetNode.prev-sibling;
    return prev.isInstanceOf(IfStatement) && prev:IfStatement.else == null;
  }
include default substitute menu for (parentNode, currentTargetNode, link, editorContext, model)->concept< > {
  concept/Statement/;
}
```

Подстановки

```
<default> editor for concept StatementsBlock
  node cell layout:
    [-
    ? {
    (- % statements % ↵ /empty cell: <default> -)
    ? }
    -]
```

```
element factory (node, prevNode, nextNode, index)->node<> {
                  new node<EmptyLine>();
                }
```

Подстановки

```
if some == other then println "Equal"
```

Transformations

Transformations

```
var abc = "Some text"
println "Abc = "
```

Transformations

```
transformation menu for concept Expression : default

section({ side transform : right }) {
```

Transformations

```
transformation menu for concept Expression : default

section({ side transform : right }) {
  wrap substitute menu
    target node <default>
    menu to wrap default substitute menu for BinaryOperation
```

Transformations

```
transformation menu for concept Expression : default

section({ side transform : right }) {
  wrap substitute menu
    target node <default>
    menu to wrap default substitute menu for BinaryOperation
    handler
      (editorContext, node, model, pattern, wrappedItem, createdNode, targetNode)->void {
        targetNode.replace with(createdNode);
        createdNode.left = targetNode:Expression;
        createdNode.right.select[in: editorContext, cell: MOST_RELEVANT, selectionStart: -1];
      }
    <no additional features>
}
```

jetbrains.mps.baseLanguage.PrecedenceUtil

Переменные
и область их видимости

Переменные

```
interface concept Variable extends INamedConcept
```

properties:
<< ... >>

children:
<< ... >>

references:
<< ... >>

Переменные

```
concepts constraints Variable {

  default scope
    scope (referenceNode, contextNode, containmentLink, position, linkTarget)->Scope {
      ListScope.forNamedElements(concept/VariablesProvider/.collectVariables(contextNode));
    }
    presentation <no presentation>
```

Переменные

```
interface concept behavior VariablesProvider {
```

Переменные

```
interface concept behavior VariablesProvider {

public virtual abstract void provideVariables(nlist<Variable> list, node<> requestor);
```

Переменные

```
concept VarStatement extends    BaseConcept
                     implements Statement
                                Variable
                                VariablesProvider

  alias: var
  short description: Variable declaration

  children:
  value : Expression[1]

<default> editor for concept VarStatement
  node cell layout:
    [- var { name } = % value % -]
```

Переменные

```
public virtual void provideVariables(nlist<Variable> list, node<> requestor)
  overrides VariablesProvider.provideVariables {
  list.add(this);
}
```

Переменные

```
concept behavior StatementsBlock {

  public virtual void provideVariables(nlist<Variable> list, node<> requestor)
    overrides VariablesProvider.provideVariables {
    node<Statement> stopNode = requestor.ancestor<concept = Statement, +>;
    for (node<Statement> statement : this.statements) {
      if (statement == stopNode) { break; }
      if (statement.isInstanceOf(VariablesProvider)) {
        statement:VariablesProvider.provideVariables(list, this);
      }
    }
  }
```

Переменные

```
concept VarValue extends    BaseConcept
                 implements Expression

  properties:
  << ... >>

  children:
  << ... >>

  references:
  var : Variable[1]

<default> editor for concept VarValue
  node cell layout:
    ( | % | var | % | -> | { | name | } | )
```

Переменные

```
substitute menu for concept VarValue : default

parameterized
  parameter type: node<Variable>
  parameter values: (parentNode, currentTargetNode, link, editorContext, model)->sequence<node<Variable>> {
    node<> requestedNode = currentTargetNode == null ? parentNode : currentTargetNode;
    concept/VariablesProvider/.collectVariables(requestedNode);
  }
  substitute action(output concept: default)
    create node (parameterObject, parentNode, currentTargetNode, link, editorContext, model, pattern)->node<VarValue> {
      node<VarValue> node = new node<VarValue>();
      node.var = parameterObject;
      return node;
    }
    matching text (parameterObject, parentNode, currentTargetNode, link, editorContext, model, pattern)->string {
      parameterObject.name;
    }
```

Переменные

```
var abc = "text"
if some then var bca = "other"
println "abc = " + abc + "bca = " + bca
```

Future

Future

```
interface concept Future extends Variable

default scope
  scope (referenceNode, contextNode, containmentLink, position, linkTarget)->Scope {
    ListScope.forNamedElements(concept/FuturesProvider/.collectFutures(contextNode))
  }

interface concept FuturesProvider extends <none>

public virtual abstract void provideFutures(nlist<Future> list, node<> requestor);

public static nlist<Future> collectFutures(node<> requestor) {
  nlist<Future> collected = new nlist<Future>;
  // ...
  return collected;
}
```

Future

```
concept RestGet extends     BaseConcept
                implements  Statement
                            Future
                            VariablesProvider
                            FuturesProvider
  alias: get rest
  short description: REST GET request

  children:
  url        : StringExpression[1]
  onResponse : RestSuccessHandler[0..1]
  onError    : RestErrorHandler[0..1]
```

Future

```
concept behavior RestGet {
public virtual void provideFutures(nlist<Future> list, node<> requestor)
  overrides FuturesProvider.provideFutures {
  if (Utils.isChildOf(this, requestor)) {
    list.add(this);
  }
}
public static boolean isChildOf(node<> node, node<> parentNode) {
  for (node<> parent = node.parent; parent != null; parent = parent.parent) {
    if (parent == parentNode) { return true; }
  }
  return false;
}
```

Future

```
concept behavior RestGet {
public virtual void provideVariables(nlist<Variable> list, node<> requestor)
  overrides VariablesProvider.provideVariables {
  if (Utils.isSameOrParentFor(this.onResponse, requestor)) {
    list.add(this);
  }
}
```

Future

```
get some from uri
```

Future

```
get some from uri
  on success status println "Success response: " + some + " status: " + status
```

Future

```
get some from uri
  on success status println "Success response: " + some + " status: " + status
  on error code and reason do {
    println "Error '" + reason + "' occured with code: " + code
  }
```

Future

```
get some from uri
  on success status println "Success response: " + some + " status: " + status
  on error code and reason do {
    println "Error '" + reason + "' occured with code: " + code
    println some + status
  }
println some + status + code + reason
```

Future

```
get some from uri1
```

Future

```
get some from uri1
get other from uri2
```

Future

```
get some from uri1
get other from uri2
  on success println "Result: " + other + some
```

Future

```
get some from uri1
get other from uri2

when all some, other futures complete do {
  println "Got " + some + " and " + other
}
```

Исполнение

Интерпретация или компиляция

Интерпретация

Уже были наработки

Легко реализовали версионирование моделей

Исправления и изменения не требуют перекомпиляции моделей

Можно иметь разные варианты движков на проде и тесте

XML представление модели

```
<?xml version="1.0" encoding="UTF-8"?>
<model ref="r:54e11540-5661-4188-a61b-120eae2814f8(sample.plang.hello)">
 <languages>
    <use id="36ab000b-5cc9-4dfb-a803-f0ddb17df645" name="PLang" version="0" />
  </languages>
 <registry>
   <language id="36ab000b-5cc9-4dfb-a803-f0ddb17df645" name="PLang">
      <concept id="76800883154347674 29" name="Println" flags="ng" index="2ByNf$">
        <child id="7680088315434767558" name="text" index="2ByNdB" />
      </concept>
      <concept id="7680088315433104261" name="StringLiteral" flags="ng" index="2B$pg$">
        <property id="7680088315433104264" name="value" index="2B$pgD" />
      </concept>
      <concept id="7680088315432946623" name="Program" flags="ng" index="2B_QKu" />
      <concept id="7680088315432946626" name="StatementsBlock" flags="ng" index="2B_QLz">
        <child id="7680088315432946627" name="statements" index="2B_QLy" />
      </concept>
    </language>
  </registry>
  ...
```

XML представление модели

```
<?xml version="1.0" encoding="UTF-8"?>
<model ref="r:54e11540-5661-4188-a61b-120eae2814f8(sample.plang.hello)">
  <languages> ... </languages>
  <registry> ...  </registry>

  <node concept="2B_QKu" id="3yKqvyftbWj">
    <property role="TrG5h" value="Hello" />
    <node concept="2ByNf$" id="3yKqvyftbWl" role="2B_QLy">
      <node concept="2B$pg$" id="3yKqvyftbWq" role="2ByNdB">
        <property role="2B$pgD" value="Hello World!" />
      </node>
    </node>
  </node>

</model>
```

XML + JAXB = легко

```
@XmlRootElement(name = "property")
public class MPSNode {
  @XmlAttribute
  String concept;
  @XmlAttribute
  String id;
  @XmlAttribute
  String role;
  @XmlTransient
  MPSNode parent;

  @XmlElements({@XmlElement(name = "property", type = MPSNodeProperty.class)})
  List<MPSNodeProperty> properties = new LinkedList<>();
  @XmlElements({@XmlElement(name = "node", type = MPSNode.class)})
  List<MPSNode> children = new LinkedList<>();
  @XmlElements({@XmlElement(name = "ref", type = MPSRef.class)})
  List<MPSRef> refs = new LinkedList<>();
```

Реализация концептов

```
@Concept(language = "PLang")
@RequiredArgsConstructor
public class IfStatement<C extends ExecuteContext<C>> implements Statement<C> {
  @ConceptChild
  private final Expression<?, C> condition;
  @ConceptChild
  private final StatementsBlock<C> then;
  @ConceptChild @Nullable
  private final StatementsBlock<C> else;


  @Override
  public CompletionStage<?> execute(C context) { … }
}
```

Фабрика концептов

```
public class IfStatementFactory<C extends ExecuteContext<C>>
  implements ConceptFactory<IfStatement<C>>
{
  public String conceptName() { return "PLang.structure.IfStatement"; }

  public IfStatement<C> create(BuilderContext ctx, MPSNode node, MPSConcept meta) {
    BuilderContext context = ctx.fork(() -> "if");
    Expression<?,C> condition_ = context.createChildConcept(node, "condition");
    StatementsBlock<C> then_ = context.createChildConcept(node, "then");
    StatementsBlock<C> else_ = context.createChildConceptIfExists(node, "else");
    try {
      return new IfStatement(condition_, then_, else_);
    } catch (Exception ex) {
      throw ConceptFactory.error(ex, context);
    }
  }
}
```

Реализация концептов

```
public CompletionStage<?> execute(C context) {
  var res = condition.evaluate(context);
  boolean result = res != null && (res instanceof Boolean bool ? bool : false);
  if (result) {
    var ctx = context.fork(() -> "if " + condition.digest() + " then»;
    return then.execute(ctx);
  }
  if (else != null) {
    var ctx = context.fork(() -> "if " + condition.digest() + " else";
    return else.execute(ctx);
  }
  context.trace(() -> "if " + condition.digest() + " = false");
  return VOID_FUTURE;
}
```

Контекст исполнения

Входные данные

Компоненты окружения

var

Локальные переменные

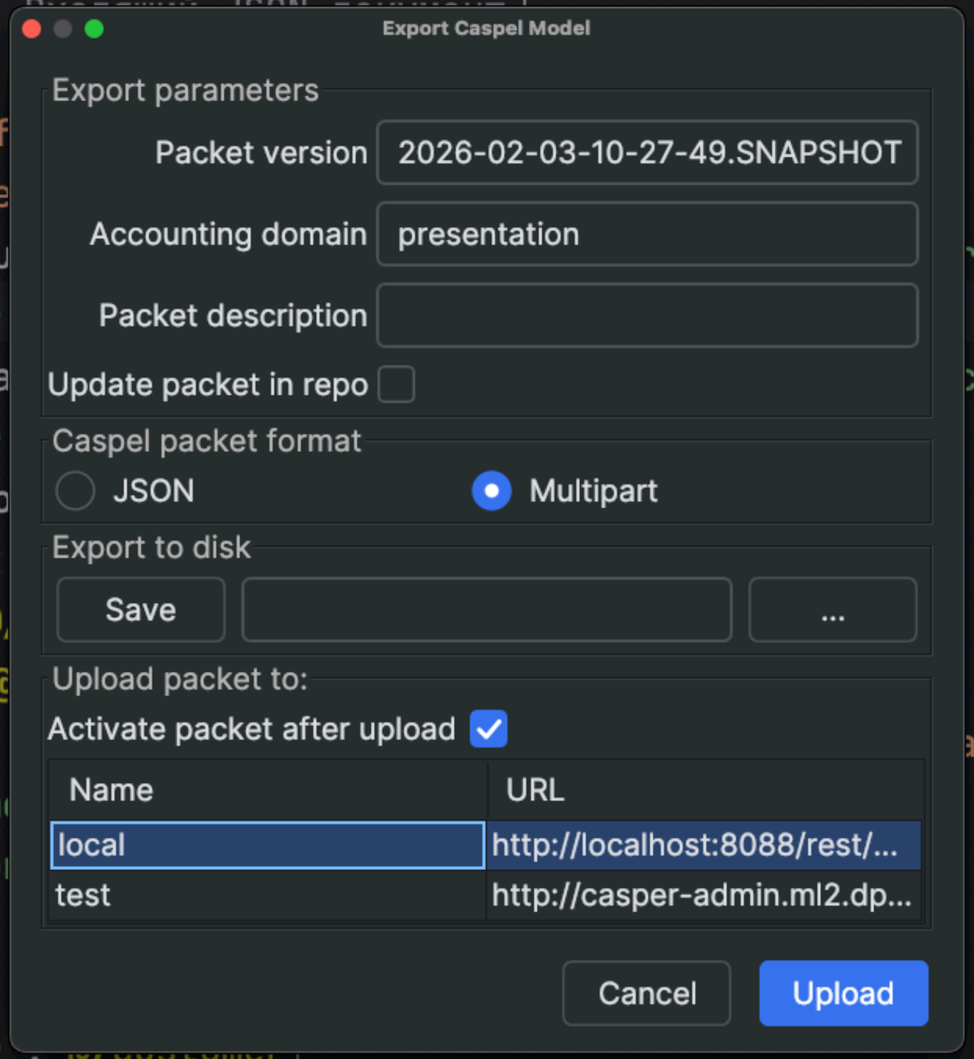
Реестр ресурсов

Информация трасировки

```
public interface ExecuteContext<C extends ExecuteContext<C>> {
    void setVariable(String name, Object value);
    Object getVariable(String name);

    void trace(Supplier<String> messageSupplier);

    C fork(Supplier<String> stackInfo);
}
```

Export plugin

```
default accounting plan shopping
parameters[input //
{
  collect event id f
  collect event time
  collect counter pu                                    rchases'
    using['customer
  collect counter ba                                    chases'
    using['customer
  collect counter so
    using['merchant
          'shop': @
  for each item in @
    collect counter                                     additive
      using['custom
            'categor
  }
  register relation
    using['customer
```



Итого

- Всего за 2 месяца получили рабочий вариант
- Поддержка асинхроной обработки
- Диалекты языка для разных задач
- Удобный инструмент для специалистов
- Уменьшение затрат
- Сокращение сроков изменения и создания сценариев
- Мы спокойно занимаемся развитием платформы и диалектов

Спасибо за внимание

Луговой Сергей, Koronatech

@JavaTalksMeetup

koronatech.ru