

# IntelliJ Rust Intentions pack

Владимир Харитонов

Computer Science Center  
Руководитель: Алексей Кладов

December 19, 2016

# Язык Rust

Разрабатывается с 2009 года компанией Mozilla.

## Особенности

- Абстракция без накладных расходов
- Безопасность памяти без GC (RAII + lifetimes)
- Отсутствие гонок данных при конкурентном программировании



# IDE для Rust

- IntelliJ IDEA
- Eclipse
- Visual Studio
- Sublime
- Emacs
- Vim
- ...

# Постановка задачи

Возможности плагина:

- Вывод типов
- Разрешение ссылок(go to definition)
- Автоформатирование
- Поддержка Cargo(система сборки и менеджер пакетов)

Пополнить плагин стандартными для IDEA "умными фичами":

- Postfix templates
- Surround with...
- Intentions

## Пример. Postfix template "match"















```
1 mod m1 {  
2     enum Action {  
3         Quite,  
4         ChangeColor(i32, i32, i32),  
5         Move { x: i32, y: i32 },  
6     }  
7 }  
8  
9 fn get_action() -> m1::Action {...}  
12  
13 fn main() {  
14     use m1::Action;  
15     get_action().|  
16 }
```

lambda	expr
match	match expr {...}
par	(expr)

Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>π

# Psi-дерево

```
3
4
5 mod m1 {
6     enum Action {
7         Quite,
8         ChangeColor(i32, i32, i32),
9         Move { x: i32, y: i32 },
10    }
11 }
12
13 fn get_action() -> m1::Action {...}
14
15
16
17 fn main() {
18     use m1::Action;
19     get_action()
20 }
```

- PsiWhiteSpace
- ▶  RustModItemElementImpl(MOD\_ITEM)
- PsiWhiteSpace
- ▶  RustFnItemElementImpl(FN\_ITEM)
- PsiWhiteSpace
- ▼  RustFnItemElementImpl(FN\_ITEM)
  -  PsiElement(fn)
  - PsiWhiteSpace
  -  PsiElement(<IDENTIFIER>)
- ▶  RustParametersElementImpl(PARAMETERS)
- PsiWhiteSpace
- ▼  RustBlockElementImpl(BLOCK)
  -  PsiElement({)
  - PsiWhiteSpace
  - ▶  RustUseItemElementImpl(USE\_ITEM)
  - PsiWhiteSpace
  - ▼  RustCallExprElementImpl(CALL\_EXPR)
    - ▶  RustPathExprElementImpl(PATH\_EXPR)
    - ▼  RustArgListElementImpl(ARG\_LIST)
      -  PsiElement(())
      -  PsiElement(())

# Psi-дерево

```
2
3
4
5 mod m1 {
6     enum Action {
7         Quite,
8         ChangeColor(i32, i32, i32),
9         Move { x: i32, y: i32 },
10    }
11 }
12
13 fn get_action() -> m1::Action {...}
14
15
16
17 fn main() {
18     use m1::Action;
19     get_action()
20 }
```

- ▼ RustEnumItemElementImpl(ENUM\_ITEM)
  - ▼ PsiElement(enum)
    - PsiWhiteSpace
    - ▼ PsiElement(<IDENTIFIER>)
      - PsiWhiteSpace
  - ▼ RustEnumBodyElementImpl(ENUM\_BODY)
    - ▼ PsiElement({)
      - PsiWhiteSpace
    - ▶ RustEnumVariantElementImpl(ENUM\_VARIANT)
      - ▼ PsiElement(,)
        - PsiWhiteSpace
    - ▼ RustEnumVariantElementImpl(ENUM\_VARIANT)
      - ▼ PsiElement(<IDENTIFIER>)
        - ▶ RustTupleFieldsElementImpl(TUPLE\_FIELDS)
          - ▼ PsiElement(,)
            - PsiWhiteSpace
      - ▶ RustEnumVariantElementImpl(ENUM\_VARIANT)
        - ▼ PsiElement(,)
          - PsiWhiteSpace
        - ▼ PsiElement({)
          - PsiWhiteSpace

## Postfix template "match". Результат

```
1  mod m1 {  
2      enum Action {  
3          Quite,  
4          ChangeColor(i32, i32, i32),  
5          Move { x: i32, y: i32 },  
6      }  
7  }  
8  
9  fn get_action() -> m1::Action {...}  
12  
13  fn main() {  
14      use m1::Action;  
15      match get_action() {  
16          Action::Quite => {},  
17          Action::ChangeColor(v0, v1, v2) => {},  
18          Action::Move { x, y } => {},  
19      };  
20  }
```



# Match to let if intentions

```
1  fn divide(numerator: f64, denominator: f64) -> Option<f64> {...}
4
5  fn main() {
6      match divide(2, 2) {
7          Some(v0) => {println!("success")},
8          None => {},
9      };
10 }
11
```

Convert math statement to if let ▶



```
1  fn divide(numerator: f64, denominator: f64) -> Option<f64> {...}
4
5  fn main() {
6      if let Some(v0) = divide(2, 2) { println!("success") };
7  }
8
```

# Все фишки

## Postfix templates

- if / else
- while / while not
- match
- paren
- lambda
- assert, assert\_eq
- let, val, var
- for

## Expression surround with

- if, while
- скобки ()
- отрицание !

## Statements surround with

- if
- while
- loop
- for
- скобки

## Intentions

- Добавление ветви else
- Добавление/удаление {}
- Match to let if
- Un-elide lifetimes

## Технологии

- Rust
- Idea plugin API
- Kotlin
- GitHub

## Plugin

<https://intellij-rust.github.io/>

<https://github.com/intellij-rust/intellij-rust.git>