Java VM

07.10.2014

Parser combinators

- Рекурсивный нисходящий парсер
- Парсер реализуется в виде обычной функции, а различные конструкции - в виде функций высшего порядка

«Просто» парсер

```
class Parser {
  abstract class Result
  case class Success(result: Output) extends Result
  case class Fail(message: String) extends Result

  def parse(i: Input): Result = ???
}
```

Комбинатор

```
class Parser {
  abstract class Result
  case class Success(result: Output, rest: Input) extends Result
  case class Fail(message: String, all: Input) extends Result

  def parse(i: Input): Result = ???
}
```

```
def combine(first: Parser, second: Parser): Parser = ???
```

Реализация в Scala

```
package scala.util.parsing.combinator.Parsers //*

abstract class Parser[+T] extends (Input=>ParseResult[T]) {
  type Elem
  type Input = Reader[Elem]

  def apply(in: Input): ParseResult[T]
  //…
}
```

Реализация в Scala (2)

- RegexParsers, JavaTokenParsers
- parseAll(S, Input)

Стандартные комбинаторы

- JavaTokenParsers, RegexParsers
- | - комбинатор «или»
- ~, <~, ~> - комбинаторы «и»
- rep - комбинатор повторения
- ^^ - комбинатор трансформации

Пример

- Парсер арифметических выражений

^^

```
def expr: Parser[Expression] = term~("+"~term | "-"~term) ^^ {
  case ~(a, ~("+", b)) => Addition(a, b)
}
```

Задачки

- Калькулятор (обратная польская запись). Обязательно использовать промежуточное представление
- Парсинг JSON

Контакты

- dmitry.naydanov@jetbrains.com