

# Модерни програмни езици за JVM

## Groovy, Scala и Clojure

инж. Божидар Бацов

OpenFest 2010

21.11.2010

# Кой е инж. Божидар Бацов?

# Кой е инж. Божидар Бацов?

- Батман

# Кой е инж. Божидар Бацов?

- Батман
- Хакер

# Кой е инж. Божидар Бацов?

- Батман
- Хакер
- Технически директор на Drow Ltd.

# Кой е инж. Божидар Бацов?

- Батман
- Хакер
- Технически директор на Drow Ltd.
- Ентусиаст на тема свободни технологии, качествен алкохол и добра музика

# Кой е инж. Божидар Бацов?

- Батман
- Хакер
- Технически директор на Drow Ltd.
- Ентузиаст на тема свободни технологии, качествен алкохол и добра музика
- Кинокритик на свободна практика

# Кой е инж. Божидар Бацов?

- Батман
- Хакер
- Технически директор на Drow Ltd.
- Ентусиаст на тема свободни технологии, качествен алкохол и добра музика
- Кинокритик на свободна практика
- Преподавател на още по-свободна практика

# Кой е инж. Божидар Бацов?

- Батман
- Хакер
- Технически директор на Drow Ltd.
- Ентусиаст на тема свободни технологии, качествен алкохол и добра музика
- Кинокритик на свободна практика
- Преподавател на още по-свободна практика
- Контакти
  - Twitter - bbatsov
  - Ел. поща - bozhidar@drow.bg
  - Jabber/GTalk - bozhidar.batsov@gmail.com
  - irc.freenode.net - bozhidar #emacs #clojure #scala #lisp

## Забележка #1

Аз **НЕ** съм експерт в областта на езиците, за които ще си говорим.

## Забележка #2

Тази презентация ще представи само най-общо и повърхностно разглежданите езици.

# Java

- Програмен език

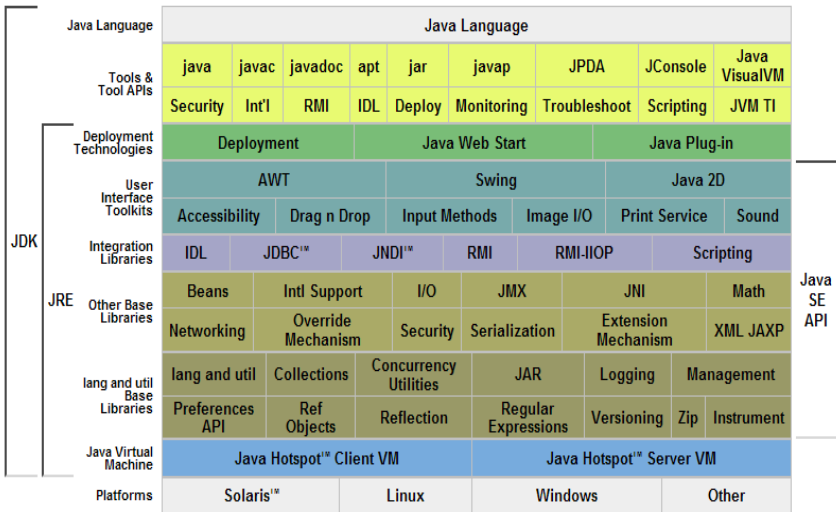
# Java

- Програмен език
- Виртуална машина

# Java

- Програмен език
- Виртуална машина
- Стандартна библиотека

# Платформата Java



# Езикът Java

- Създаден да замени C++

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp

# ЕЗИКЪТ Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тромава и консервативна

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тровава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тровава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност
  - проповядва императивен стил на програмиране

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тремава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност
  - проповядва императивен стил на програмиране
  - липсват closures

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тремава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност
  - проповядва императивен стил на програмиране
  - липсват closures
  - езикът не е чисто обектно ориентиран

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тровава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност
  - проповядва императивен стил на програмиране
  - липсват closures
  - езикът не е чисто обектно ориентиран
  - checked exceptions са наказание от Гореп

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тровава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност
  - проповядва императивен стил на програмиране
  - липсват closures
  - езикът не е чисто обектно ориентиран
  - checked exceptions са наказание от Гореп
  - няма възможност за презареждане на оператори

# Езикът Java

- Създаден да замени C++
- Отвежда C++ програмистите до средата на пътя към Lisp
- Разработката му е тровава и консервативна
- Основни критики към Java в момента
  - ниско ниво на изразителност
  - проповядва императивен стил на програмиране
  - липсват closures
  - езикът не е чисто обектно ориентиран
  - checked exceptions са наказание от Гореп
  - няма възможност за презареждане на оператори
  - и мнооого други ;-)

# Алтернативните езици

- Java 1.0 излиза през януари 1996

# Алтернативните езици

- Java 1.0 излиза през януари 1996
- през 1997 вече има 40 езика с имплементации за JVM

# Алтернативните езици

- Java 1.0 излиза през януари 1996
- през 1997 вече има 40 езика с имплементации за JVM
- през 2004 те са вече 169

# Алтернативните езици

- Java 1.0 излиза през януари 1996
- през 1997 вече има 40 езика с имплементации за JVM
- през 2004 те са вече 169
- понастоящем са около 240+

# Алтернативните езици

- Java 1.0 излиза през януари 1996
- през 1997 вече има 40 езика с имплементации за JVM
- през 2004 те са вече 169
- понастоящем са около 240+

# Предимства на разработка на езици за JVM

- Rock solid, hearth touching ;-)

# Предимства на разработка на езици за JVM

- Rock solid, hearth touching ;-)
- JVM е изключително надеждна, многоплатформена и притежава висока производителност и отлични оптимизации

# Предимства на разработка на езици за JVM

- Rock solid, hearth touching ;-)
- JVM е изключително надеждна, многоплатформена и притежава висока производителност и отлични оптимизации
- Имате на разположение директно огромната база от съществуващ Java код

# Предимства на разработка на езици за JVM

- Rock solid, hearth touching ;-)
- JVM е изключително надеждна, многоплатформена и притежава висока производителност и отлични оптимизации
- Имате на разположение директно огромната база от съществуващ Java код
- Купица отлични инструменти за разработка и привеждане в експлоатация

# Не всичко е ток и жица

- Виртуалната машина по настоящем не е оптимизирана за динамични езици и за някои стилове на програмиране (например функционално)

# Не всичко е ток и жица

- Виртуалната машина по настоящем не е оптимизирана за динамични езици и за някои стилове на програмиране(например функционално)
- Виртуалната машина има сравнително голям период на студено стартиране(cold start)

# Не всичко е ток и жица

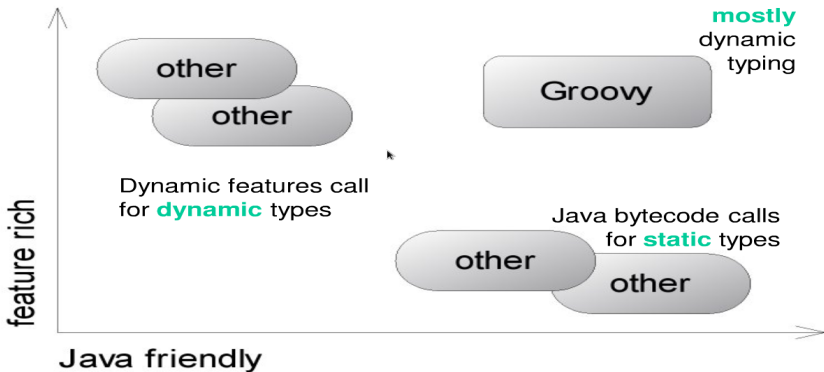
- Виртуалната машина по настоящем не е оптимизирана за динамични езици и за някои стилове на програмиране(например функционално)
- Виртуалната машина има сравнително голям период на студено стартиране(cold start)
- Java имплементациите на някои езици като Python(Jython) не са съвместими с native имплементациите

# Groovy

## Хвала на Groovy

Groovy is like a super version of Java. It can leverage Java's enterprise capabilities but also has cool productivity features like closures, builders and dynamic typing. If you are a developer, tester or script guru, you have to love Groovy.

# Пейзажът на JVM езиците



# Groovy е...

- Динамичен език за JVM
  - Програмите на Groovy директно се компилират до байт код
- Вдъхновен от езици като Ruby, Python и Smalltalk
- Създаден за да улесни живота на (Java) програмистите
- Проект с отворен код, лицензиран под Apache Public License
- С граматика директно извлечена от Java 5
  - поддържа анотации, generics, изброени типове(enums), статични импорти и т.н.

# Ключови характеристики

- Напълно обектно-ориентиран
- Типизирането не е задължително
- Няма нужда да чакате Java 7/8/9:
  - Closures - използвани / присвоими блокове код
  - Атрибути - автоматично генерирани getter/setter методи
- Duck typing - типът на обектите се определя от техния интерфейс, а не от техния клас
- Аритметика базирана на BigDecimal
- Улеснена работа с SQL, XML, Swing и т.н.

# Java интеграция

- Groovy генерира Java байткод за JVM
  - Същите низове, регулярни изрази и т.н.
  - Същите API
  - Същия модел на сигурност, същия нишков модел
  - Същите OO концепции
  - Комбинирана компилация

# Пример 1

```
1 System.out.println("Hello , World!");
2 println 'Hello , World!'
3
4 def name = 'Bozhidar'
5 println "$name, I'll get the car." // GString
6 String longer = ""${name}, the car
7 is in the next row.""
8
9 def printSize(obj) {
10     print obj?.size() // safe dereferencing
11 }
12
13 def pets = [ 'ant', 'bee', 'cat' ]
14
15 pets.each { pet ->
16     assert pet < 'dog'
17 }
```

## Пример 2

```
1 import static java.util.Calendar.getInstance as now
2 import java.util.GregorianCalendar as D
3 import org.codehaus.groovy.runtime.TimeCategory
4
5 println new D(2007,11,25).time
6 println now().time
7
8 assert "Hello World!" =~ /Hello/
9 // Find operator
10 assert "Hello World!" ==~ /Hello\b.*/ // Match
    operator
11 def p = ~/Hello\b.*/
12 // Pattern operator
13 assert p.class.name == 'java.util.regex.Pattern'
```



# Groovy Dev Kit(GDK)

- Groovy „декорира" съществуващите JDK API
  - Не можем да разширим `java.lang.String` или `java.io.File`?

```
1 new File('/x.txt').eachLine {  
2     println it  
3 }
```

## Обработка на данни от външни източници

- Вместо по-добро SQL API имаме стандартен достъп до данни (Uniform Data Access)
- Достъпът до данни представени като обектни, XML или релационни бази данни има повече общи черти, отколкото различни
- Едно API независимо от източника на данни
- Разликите са само в установяването на изходната постановка
- Приликите са в обработката на данните в резултатните структури
- Резултатът е подобен на LINQ в .NET

# Работа с XML

```
1 <records>
2   <car name='HSV Maloo' make='Holden' year='2006'>
3     <country>Australia</country>
4     <record type='speed'>Production Pickup Truck with speed of 271kph
5       </record>
6   </car>
7   <car name='P50' make='Peel' year='1962'>
8     <country>Isle of Man</country>
9     <record type='size'>Smallest Street-Legal Car at 99cm wide and 59
10      kg weight</record>
11   </car>
12   <car name='Royale' make='Bugatti' year='1931'>
13     <country>France</country>
14     <record type='price'>Most Valuable Car at $15 million</record>
15   </car>
16 </records>
```

# Работа с XML

```
1 def records = new XmlParser().parse("records.  
   xml")  
2 records.car.each {  
3     println "year = ${it.@year}"  
4 }
```

# Swing

```
1 import java.awt.FlowLayout
2 builder = new groovy.swing.SwingBuilder()
3 langs = ["Groovy", "Ruby", "Python", "Pnuts"]
4 gui = builder.frame(size: [290, 100],
5 title: 'Swinging with Groovy!') {
6 panel(layout: new FlowLayout()) {
7     panel(layout: new FlowLayout()) {
8         for (lang in langs) {
9             checkBox(text: lang)
10        }
11    }
12    button(text: 'Groovy Button', actionPerformed: {
13        builder.optionPane(message: 'Indubitably Groovy!').
14        createDialog(null, 'Zen Message').show()
15    })
16    button(text: 'Groovy Quit',
17    actionPerformed: {System.exit(0)})
18    }
19    }
20    gui.show()
```

# Инструментите на занаята

- Част от дистрибуцията на Groovy
  - groovysh
  - groovyconsole
  - groovyc
- Среди за разботка
  - IntelliJ IDEA is the King!
  - NetBeans
  - Eclipse
- Строителни инструменти
  - Apache Maven
  - Gradle
  - Apache Buildr

# Scala

James Gosling, създател на Java

If I were to pick a language to use today other than Java, it would be Scala...

# Scala е ...

- SCAlable LAnguage
- Чист OO език
- Функционален език
- Език със статично типизиране
- Голяма общност
- Прекрасна интеграция със съществуващ Java код

# Експресивен

```
1 val phonebook = Map("Pesho" -> 123456,  
2                       "Vankata" -> 654321)  
3 phonebook += ("Marmota" -> 987654)  
4 println(phonebook("Pesho"))
```

## Високо ниво

```
1 boolean hasUpperCase = false;
2 for (int i = 0; i < name.length(); i++) {
3     if (Character.isUpperCase(name.charAt(i)))
4         {
5         hasUpperCase = true;
6         break;
7     }
8 }
```

```
1 val hasUpperCase = name.exists(_.isUpperCase)
```

## Компактен

```
1 // Java
2 public class Person {
3     private String name;
4     private int age;
5     public Person(String name,
6         int age) {
7         this.name = name;
8         this.age = age;
9     }
10    public String getName() {
11        return name;
12    }
13    public int getAge() {
14        return age;
15    }
16    public void setName(String name)
17        {
18        this.name = name;
19    }
20    public void setAge(Int age) {
21        this.age = age;
22    }
23 }
```

```
1 // Scala
2 class Person(
3     var name: String,
4     var age: Int)
```

# Чиста проба ООП

- Всичко е обект
- Няма оператори, има само методи
  - $1 + 2$
  - $1.(+)(2)$
- Няма статични методи/полета - има companion обекти

# Мощ без граници

```
1 val service = actor {
2   loop {
3     receive {
4       case Add(x,y) => reply(x+y)
5       case Sub(x,y) => reply(x-y)
6     }
7   }
8 }
9 service ! Add(4, 2) // => 6
```

## По-гъвкаво наследяване с Traits

- Интерфейси на стероиди
- Възможности отвъд границите на въображението
- Отворете Google и напишете scala traits

# Завръщането на глаголите(функциите)

- Функциите са пълноправни обекти
  - `val inc = (x: Int) => x + 1`
  - `inc(1) // => 2`
- Higher-order functions
  - `List(1, 2, 3).map((x: Int) => x + 1)`
  - `=> List(2, 3, 4)`
- Подсладени функции ;-)
  - `List(1, 2, 3).map(x => x + 1)`
  - `List(1, 2, 3).map(_ + 1)`

# Къде са ми Closures?

```
1 var more = 7
2 val addMore = (x: Int) => x + more
3 addMore(3) // => 10
4 more = 8
5 addMore(3) // => 11
```

# Функционални структури от данни

- List
- Maps
- Sets
- Trees
- Stacks

# На колене пред всемогъщия List

```
1 val list = List(1, 2, 3)
2 list.map(_ + 1) // => List(2,3,4)
3 list.filter(_ < 2) // => List(3)
4 list.exists(_ == 3) // => true
5 list.drop(2) // => List(3)
6 list.reverse // => List(3,2,1)
7 list.sort(_ > _) // => List(3,2,1)
8 list.flatten(list) // => List(1, 2, 3)
9 list.slice(2, 3) // => List(3)
```

# Pattern Matching

```
1 def matchAny(a: Any): Any = a match {  
2   case 1 => "one"  
3   case "two" => 2  
4   case i: Int => "scala.Int"  
5   case <tag>{ t }</tag> => t  
6   case head :: tail => head  
7   case _ => "default"  
8 }
```

# Инструментите на занаята

- Част от дистрибуцията на Scala
  - scala
  - scalac
  - fsc
  - scaladoc
  - sbaz
- Среди за разботка
  - IntelliJ IDEA is still the King!
  - ENSIME - Enhanced Scala Emacs Integraction
  - NetBeans
  - Eclipse
- Строителни инструменти
  - Apache Maven
  - Simple Build Tool
  - Apache Buildr

# Clojure

Rich Hickey, създател на Clojure

Time is the New Memory!  
Mutable objects are the new spaghetti code!

# Clojure

- Clojure е динамичен език за JVM
- Clojure е елегантен
- Clojure е еволюцията на Lisp
- Clojure е функционален
- Clojure е паралелен
- Clojure работи директно с Java
- Clojure е бърз

# Clojure е елегантен

- Премахва инцидентната сложност в огромна степен
- Програмите на Clojure са кратки и експресивни
- Кратките програми по правило са по-лесни и по-евтини за поддръжка

## Пример - Java

```
1 public class StringUtils {
2     public static boolean isBlank(String str) {
3         int strLen;
4         if (str == null || (strLen = str.length()
5             ) == 0) {
6             return true;
7         }
8         for (int i = 0; i < strLen; i++) {
9             if ((Character.isWhitespace(str.charAt(
10                i))) == false)) {
11                 return false;
12             }
13         }
14     }
15 }
```

# Пример - Clojure

```
1 (defn blank? [s] (every? #(Character/  
    isWhitespace %) s))
```

## Пример - Java

```
1 public class Book {
2     private String title;
3     private String author;
4
5     public String getTitle() {
6         return title;
7     }
8     public void setTitle(String title) {
9         this.title = title;
10    }
11    public String getAuthor() {
12        return author;
13    }
14    public void setAuthor(String author) {
15        this.author = author;
16    }
17 }
```

# Пример - Clojure

```
1 (defstruct Book :name :author)
```

# Clojure e Lisp

- Динамичен
- Кодът е данни(code is data)
- Reader
- Малко ядро
- Sequences
- Синтактични абстракции

# Динамична разработка

- REPL (Read-Print-Eval-Loop)
- Дефиниране на функции в реално време
- Зареждане и компилиране на код по време на изпълнение
- Интроспекция
- Интерактивна среда

# Clojure е функционален

- Функциите са обекти
- Данните са непроменими(immutable)
- Повечето функции са чисти(нямат странични ефекти)
- Clojure не е чист функционален език, а практичен такъв

# Clojure е паралелен

- Clojure реализира паралелен програмен модел, който не е организиран около заключване на секции код
- Паралелна инфраструктура
  - STM(Software Transactional Memory) - за синхронни координирани промени на няколко стойности
  - атоми - за синхронна промяна на една стойност
  - агенти - за асинхронна промяна на стойности
  - vars - променливи, локални за дадена нишка с динамичен scoping
  - Java threading model

# Пример

```
1 (def visitors (ref #{}))
2
3 (defn hello
4   "Writes hello message to *out*. Calls you by
5   username.
6   Knows if you have been here before."
7   [username]
8   (dosync
9     (let [past-visitor (@visitors username)]
10        (if past-visitor
11          (str "Welcome back, " username)
12          (do
13            (alter visitors conj username)
14            (str "Hello, " username)))))))
```

# Clojure не отрича Java

- Обръщанията от Clojure към Java класове са директни и не минават през междинен слой
- Използването на Java класове е идиоматично

# Никой не е по-голям от мързела

- Унифициране на структурите от данни със seq API
- Повечето seqs в Clojure са lazy - стойностите им се изчисляват само, когато има нужда от тях
- Предимства
  - Може да отложите скъпи изчисления, докато не са необходими
  - Може да работите с огромни(безкрайни) структури от данни без да ликвидирате всичката си памет
  - Може да забавяте входно/изходни операции, докато не станат абсолютно наложителни

# Мързел в действие

```
1 (defn lazy-seq-fibo
2   ([]
3     (concat [0 1] (lazy-seq-fibo 0 1)))
4   ([a b]
5     (let [n (+ a b)]
6         (lazy-seq
7           (cons n (lazy-seq-fibo b n))))))
8
9 (take 10 (lazy-seq-fibo))
10 => (0 1 1 2 3 5 8 13 21 34)
11
12 (nth (lazy-seq-fibo) 1000000)
```

# Инструментите на занаята

- Част от дистрибуцията на Clojure
  - REPL
- Среди за разботка
  - SLIME - Superior Lisp Interaction Mode for Emacs
  - vimclojure
  - IntelliJ IDEA
  - NetBeans
  - Eclipse
- Строителни инструменти
  - Apache Maven
  - Leiningen

# Заклучение

- езикът Java е новият C++ (Cobol)
- Дори и никога да не ползвате някоя от съвременните алтернативи на Java в ежедневноста си работа те ще ви направят по-добри програмисти по принцип
- Ако, обаче, ги използвате - те ще направят работата ви едновременно много по-забавна и продуктивна :-)

# Въпроси

Тук е момента да зададете вашите  
въпроси! :-)

# Край

Благодаря Ви за вниманието!