

java.next

stuart halloway
<http://thinkrelevance.com>

Copyright 2007-2009 Relevance, Inc. This presentation is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. See <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>

1

most java code is bad

repetitive, bureaucratic code

untested/untestable

stuck on an old version of _____

miles and miles of crap

2

java can also
be part of the
solution



3

4

java, circa 2003

```
1 public ActionForward edit(ActionMapping mapping,  
2                           ActionForm form,  
3                           HttpServletRequest request,  
4                           HttpServletResponse response)  
5     throws Exception {  
6     PersonForm personForm = (PersonForm) form;  
7     if (personForm.getId() != null) {  
8         PersonManager mgr =  
9             (PersonManager) getBean("personManager");  
10        Person person = mgr.getPerson(personForm.getId());  
11        personForm = (PersonForm) convert(person);  
12        updateFormBean(mapping, request, personForm);  
13    }  
14    return mapping.findForward("edit");  
15 }
```

5

6

java.next: 2005-?

```
def edit  
  @person = Person.find(params[:id])  
end  
  
def new  
  @person = Person.new  
end
```

7

8

evolving style

convention over configuration

reasonable defaults

no checked exceptions

YAGNI

domain nouns, not language nouns

DSLs

9



Photo credit: <http://www.flickr.com/photos/uhduh/92437037/sizes/l/>

11

how java.next can help

10

common java.next features

everything is an object

easy beans

higher-order functions

unchecked exceptions

open APIs

DSLs

12

everything is an object

```
1 ; is java enterprise-ready?  
2 2000000 * 2000000  
3 -> 1385447424
```

```
1 # ruby is  
2 2_000_000 * 2_000_000  
3 => 4000000000000
```

easy beans

immutable

convenient constructor

```
1 //scala  
2 case class Person(firstName: String,  
3                      lastName: String) {}
```

static typing

13

14

higher-order functions

define a function

fn name

fn args

```
(defn whole-numbers []  
  (iterate inc 1))
```

call a fn

pass a fn as an argument

15

16

anonymous function

```

(map (fn [x] (* x x))
  (whole-numbers))
-> (1 4 9 16 25 36 49 64 81 100 ...)

```

sequence API

```

(filter odd? (whole-numbers))
-> (1 3 5 7 9 11 13 15 17 19 ...)

```

lazy evaluation

closed APIs



Ceremony is: checked exceptions.
Photo credit: <http://www.flickr.com/photos/marinegirl/2036373729>

closed

```

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

```

open APIs

groovy metaclass

```
1 // Groovy
2 String.metaClass.isBlank = {
3     length() == 0 || every{ Character.isWhitespace(it.charAt(0)) }
4 }
```

21

22

ruby open class

```
1 # Ruby (from Rails)
2 class String
3     def blank?
4         empty? || strip.empty?
5     end
6 end
```

scala implicit coercion

```
1 // Scala
2 class CharWrapper(ch: Char) {
3     def isWhitespace = Character.isWhitespace(ch)
4 }
5 implicit def charWrapper(ch: Character) = new CharWrapper(ch)
6 class BlankWrapper(s: String) {
7     def isBlank = s.isEmpty || s.forall(ch => ch.isWhitespace)
8 }
9 implicit def stringWrapper(s: String) = new BlankWrapper(s)
```

23

24

clojure generic dispatch

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

25

DSLs

26

ruby (from rails)

methods feel
like keywords

key/value options

```
class Coach < ActiveRecord::Base
  belongs_to :team
  has_many :sponsors, :as => :spokesperson
  validates_presence_of :first_name, :last_name
end
```

flexible punctuation

groovy

```
1 //from EasyB, www.easyb.org
2 given "an invalid zip code", {
3   invalidzipcode = "221o1"
4 }
5
6 and
7
8 given "the zipcodevalidator is initialized", {
9   zipvalidate = new ZipCodeValidator()
10 }
11
12 when "validate is invoked", {
13   value = zipvalidate.validate(invalidzipcode)
14 }
15
16 then "the validator should return false", {
17   ensure(!value)
18 }
```

flexible punctuation

27

28

scala

```
1 // From Programming in Scala (PrePrint Edition)
2 import scala.util.parsing.combinator._
3 class Arith extends JavaTokenParsers {
4   def expr: Parser[Any] = term~rep("+~term | "-~term)
5   def term: Parser[Any] = factor~rep("*~factor | "/"~factor)
6   def factor: Parser[Any] = floatingPointNumber | "("~expr~")"
7 }
```

extremely
flexible punctuation

clojure

data is syntax

```
1 ; blog.thinkrelevance.com/2008/9/16/pcl-clojure-chapter-3
2 (filter (where {:artist "Dixie Chicks" :rating 8}) (init-db))
3
4 (defn where [criteria]
5   (fn [m]
6     (every? (fn [[k v]] (= (k m) v)) criteria)))
```

everything is a sequence

29

30

which language
should I learn?

every time you
start a green-field
Java project,
God kills a kitten

31

32

clojure considerations

- + functional
- + multimethods
- + concurrency (STM et al)
- + lisp
- + a la carte
- youngest java.next language

33

groovy considerations

- + easiest to learn
- + easiest bi-di interop
- ? more committed to reusing Java libs
- worst Java baggage
- no concurrency/multicore story

34

ruby considerations

- + biggest community
- + commercial support: ~~Sun~~ Oracle EngineYard
- + Rails
- + multiple platforms
- no concurrency/multicore story

35

scala considerations

- + functional
- + high performance
- + pattern matching
- + actor model
- ? hybrid object/functional
- hardest to learn

36

contact stu

Email: stu@thinkrelevance.com
Office: 919-442-3030
Twitter: twitter.com/stuarthalloway
Talks: <http://blog.thinkrelevance.com/talks>
Blog: <http://blog.thinkrelevance.com>
Java.next: <http://blog.thinkrelevance.com/2008/9/24/java-next-overview>
Book: <http://www.pragprog.com/titles/shcloj/programming-clojure>