Scala

Gedanken zu einer Programmiersprache

Sebastian Eidecker

16. März 2016

sieht in jedem Problem einen Nagel.

Paul Watzlawick

Wer als Werkzeug nur einen Hammer hat,

Inhalt IT im Wandel

IT im Wandel

Inhalt

Herausforderungen

IT im Wandel

Inhalt

Herausforderungen

Manifeste











IT im Wandel

Manifeste

Herausforderungen

Scala

IT im Wandel

Herausforderungen

Manifeste

Scala

Management Summary

IT im Wandel

Herausforderungen

Manifeste

Scala

Management Summary

Ein wenig Code

IT im Wandel

Herausforderungen

Manifeste

Scala

Management Summary

Ein wenig Code

Spannendes

Software Engineering

Software	Engineering

IT im Wandel







Herausforderungen

· Stabilität und Resilienz



- · Stabilität und Resilienz
- Wertbeitrag

- · Stabilität und Resilienz
- Wertbeitrag
- Businesstreiber

- · Stabilität und Resilienz
- Wertbeitrag
- Businesstreiber

Matthias Magnor – CEO Surface und Contract Logistics

IT im Wandel

Manifeste

•

 Antwortbereit, Widerstandsfähig, Elastisch, Nachrichtenorientiert (2013)

• Gut gefertigt, Stets Mehrwert, Gemeinschaft aus Experten, Produktive Partnerschaften (2009)

 Individuen und Interaktionen, Funktionierende Software, Zusammenarbeit mit dem Kunden, Reagieren auf Veränderung (2001)

- Antwortbereit, Widerstandsfähig, Elastisch, Nachrichtenorientiert (2013)
- Gut gefertigt, Stets Mehrwert, Gemeinschaft aus Experten, Produktive Partnerschaften (2009)
- Individuen und Interaktionen, Funktionierende Software, Zusammenarbeit mit dem Kunden, Reagieren auf Veränderung (2001)

Wo stehen wir?

Wo stehen wir im Wettbewerb?

Scala

Management Summary

Scalable Language

Objektorientiert

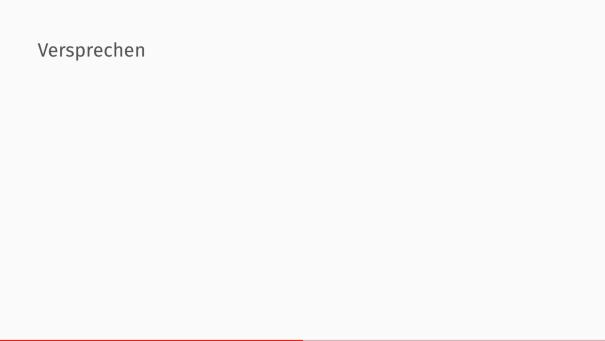
- Objektorientiert
- Funktional

- Objektorientiert
- Funktional
- Statisch typisiert

- Objektorientiert
- Funktional
- Statisch typisiert
- Gewohnte Syntax ("Java ohne Semikolon")

- Objektorientiert
- Funktional
- Statisch typisiert
- Gewohnte Syntax ("Java ohne Semikolon")
- Ausdrucksstark (APIs/DSLs schreiben)

- Objektorientiert
- Funktional
- Statisch typisiert
- Gewohnte Syntax ("Java ohne Semikolon")
- Ausdrucksstark (APIs/DSLs schreiben)
- Immutable Objects als Konzept



Versprechen

Produktivitätssteierung

Versprechen

- Produktivitätssteierung
- · Höhere Codequalität

Versprechen

- Produktivitätssteierung
- · Höhere Codequalität
- durch
- Weniger Code

Versprechen

- Produktivitätssteierung
- · Höhere Codequalität
- durch
- Weniger Code
- · Höheres Abstraktionsniveau

Versprechen

- Produktivitätssteierung
- · Höhere Codequalität
- durch
- Weniger Code
- · Höheres Abstraktionsniveau
- Mehr Spaß

• Java-Bytecode, läuft auf JVM

Scala und die Java-Plattform

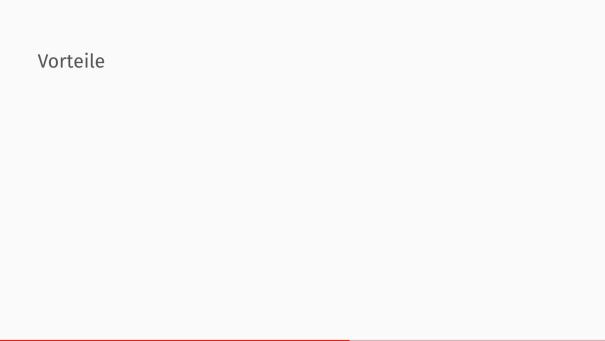
· Java-Bytecode, läuft auf JVM

· Java-Bibliotheken nutzbar

- · Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar

Bekannte IDFs

- · Java-Bytecode, läuft auf JVM
- · Iava-Bibliotheken nutzbar
- Bekannte IDFs
- · Ähnlicher Paketierungs- und Buildprozess (sbt)



Modern

- Modern
- Verständlich funktional

- Modern
- · Verständlich funktional
- · Java-Ökosystem

- Modern
- · Verständlich funktional
- · Java-Ökosystem
- Macht Spaß

- Modern
- · Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

- Modern
- · Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

Nachteile

- Modern
- · Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

Komplex

- Modern
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

- Komplex
- · Zukunftssicher?

- Modern
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

- Komplex
- · Zukunftssicher?
- Zahl an Entwicklungssklaven

- Modern
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

- Komplex
- · Zukunftssicher?
- Zahl an Entwicklungssklaven
- Binärkompatibilität nicht in alle Ewigkeit

Scala

Ein wenig Code



Eine Java-Klasse

5

7

8

10

11

12 13

```
public class Person {
  private final String firstName;
  private final String lastName:
  public Person(String firstName, String lastName) {
      this.firstName = firstName:
      this.lastName = lastName:
  public String getFirstName() {
      return firstName:
  public String getLastName() {
      return lastName:
```

Eine Java-Klasse

5

8

9

10

11

12 13

```
aOverride
public boolean equals(Object o) {
    if (this == 0) return true:
    if (o == null || getClass() != o.getClass()) return false;
    Person person = (Person) o:
    if (firstName != null ?
        !firstName.equals(person.firstName) :
        person.firstName != null) return false:
    if (lastName != null ?
        !lastName.equals(person.lastName) :
        person.lastName != null) return false:
    return true:
```

Eine Java-Klasse

```
public int hashCode() {
   int result = firstName != null ? firstName.hashCode() : 0;
   result =
        31 * result + (lastName != null ? lastName.hashCode() :
        0);
   return result;
}
```

Businesslogik?



Dasselbe in Scala

case class Person(firstName:String, lastName:String)

Dasselbe in Scala

Scala

Spannendes



Akka

Scalable real-time transaction processing

Akka

- Scalable real-time transaction processing
- · Will die aktuellen Probleme lösen

Akka

- · Scalable real-time transaction processing
- · Will die aktuellen Probleme lösen





ScalaTest ·

ScalaTest		

•

Scala

Mehr für Nerds



Mehr für Nerds

Sprecht mich an









Mehr für Nerds

- Sprecht mich an

- · Hands on-Termin bei Interesse

Mehr für Nerds

- Sprecht mich an
- · Hands on-Termin bei Interesse
- Heiko Seeberger: "Durchstarten mit Scala. Tutorial für Einsteiger (2. Aufl.)"

