
# Scala, SBT & Play! for Rapid [Web] Application Development

Антон Кириллов
ZeptoLab

# Привет!

Я **Антон Кириллов**,

- Lead Server Dev @ zeptolab
- к.т.н. «математическое моделирование, численные методы и комплексы программ»
- HighLoad и Distributed системы
- Big Scala fan!

**antonkirillov @**

**akirillov @**

# О чем пойдет речь

О том, что быстрая разработка приложений требует:

- мощный, выразительный язык
- средства автоматизации сборки и тестирования
- веб-фреймворк

А еще о том, как Scala стек отвечает этим потребностям

И, конечно же, success story!

# Frameworks & SDKs



















# Build tools






rake



make

# Ожидания от языка

- быстрый
- выразительный
- статически-типизированный
- объектно-ориентированный
- функциональный
- возможность параллельных вычислений
- изящный

# Что мы имеем?

- ~~C~~
- ~~C++~~
- ~~Ruby~~
- ~~PHP~~
- ~~Python~~
- ~~Erlang~~
- ~~Java~~
- ~~[динамические JVM языки]~~

# Scala: Scalable Language

ag;le days | Москва 2013

# Scala

# WAT.

# Scala и RAD

- Сложная
- Неэффективная
- Нет веб-фреймворков
- Нет автосборщиков
- Зачем, если есть Java
- У нас серьезный бизнес, а не Twitter

# Scala и RAD

# Scala.{ Overview }

- Функциональный и объектно-ориентированный
- Исполняется в JVM
- Java – совместимый
- Некоторые плюшки:
  - Actors concurrency model
  - immutability
  - выведение типов
  - функции высшего порядка
  - traits
  - pattern matching
  - … и многое другое

# Scala.{ OOP }

# Scala.{ Case Classes }

```
case class Person(firstName: String = "Jamie", lastName: String = "Allen")

val jamieDoe = Person(lastName = "Doe")
res0: Person = Person(Jamie,Doe)
```

- Превосходные Data Transfer Objects
- По умолчанию, поля класса immutable & public
- Не могут быть унаследованы
- Предоставляют **`equals()`**, **`copy()`**, **`hashCode()`** and **`toString()`**
- Не нужно использовать **`new`** для создания экземпляров
- Именованные параметры и значения по умолчанию дают нам семантику Builder pattern

# Scala.{ Tuples }

```
def firstPerson = (1, Person(firstName = “Barbara”))
val (num: Int, person: Person) = firstPerson
```

- Отлично подходят для группировки объектов без DTO
- Оборачивают несколько значений в один контейнер
- Максимум – 22 элемента

# Scala.{ Objects }

```
object Bootstrapper extends App { Person.createJamieAllen }

object Person {
  def createJamieAllen = new Person("Jamie", "Allen")
  def createJamieDoe = new Person("Jamie", "Doe")
  val aConstantValue = "A constant value”
}

class Person(val firstName: String, val lastName: String)
```

- Singletons within a JVM process
- Отсутствуют пляски с private конструктором
- Companion Objects, используются в качестве фабрик и для хранения констант

# Scala.{ Pattern Matching }

```
name match {
  case "Lisa" => println("Found Lisa”)
  case Person("Bob") => println("Found Bob”)
  case "Karen" | "Michelle" => println("Found Karen or Michelle”)
  case Seq("Dave", "John") => println("Got Dave before John”)
  case Seq("Dave", "John", _*) => println("Got Dave before John”)
  case ("Susan", "Steve") => println("Got Susan and Steve”)
  case x: Int if x > 5 => println("got value greater than 5: " + x)
  case x => println("Got something that wasn't an Int: " + x)
  case _ => println("Not found”)
}
```

- С этого начинается зависимость от Scala =)
- Крайне мощная и удобочитаемая конструкция

# Scala.{ Functional Programming }


MOAR
FUNCTIONS!

# Scala.{ Rich Collection Functionality }

```
val numbers = 1 to 20 // Range(1, 2, 3, ... 20)

numbers.head // Int = 1
numbers.tail // Range(2, 3, 4, ... 20)
numbers.take(5) // Range(1, 2, 3, 4, 5)
numbers.drop(5) // Range(6, 7, 8, ... 20)

scala> numbers.par map(_ + 1)
res2: s.c.parallel.immutable.ParSeq[Int] = ParVector(2, 3, 4,...

scala> res2.seq
res3: s.c.immutable.Range = Range(2, 3, 4,...
```

- Предоставляют большое количество удобных методов
- Совет: тратьте 5 минут ежедневно на прочтение ScalaDoc для отдельной имплементации

# Scala.{ Higher Order Functions }

```
val names = List("Barb", "May", "Jon")

names map(_.toUpperCase)
res0: List[java.lang.String] = List(BARB, MAY, JON)

names flatMap(_.toUpperCase)
res1: List[Char] = List(B, A, R, B, M, A, Y, J, O, N)

names filter (_.contains("a"))
res2: List[java.lang.String] = List(Barb, May)

val numbers = 1 to 20 // Range(1, 2, 3, ... 20)

numbers.groupBy(_ % 3)
res3: Map[Int, IndexedSeq[Int]] = Map(1 -> Vector(1, 4, 7,
10, 13, 16, 19), 2 -> Vector(2, 5, 8, 11, 14, 17, 20), 0 ->
Vector(3, 6, 9, 12, 15, 18))
```

- На самом деле просто методы
- Applying closures to collections

# Scala.{ Currying }

```
def product(i: Int)(j: Int) = i * j
val doubler = product(2)_
doubler(3) // Int = 6
doubler(4) // Int = 8

val tripler = product(3)_
tripler(4) // Int = 12
tripler(5) // Int = 15
```

- Рассмотрим функцию, которая принимает n параметров как список отдельных аргументов
- «Каррируем» ее для создания новой функции, которая принимает только один параметр
- Сконцентрируемся на значении и используем его для специальных реализаций **product()** в зависимости от семантики
- В Scala такие функции должны быть явно определены
- _ - то, что явно указывает на каррированную функцию

# Scala.{ Concurrency }

# Scala.{ Actors }

```
import akka.actor._

class MyActor extends Actor {
  def receive = {
     case x => println(“Got value: “ + x)
  }
}
```

- Основаны на концептах из Erlang/OTP
- Акторы перенесены из core-библиотеки Scala в Akka начиная с версии 2.10
- Данная парадигма параллелизации использует сети независимых объектов, которые взаимодействуют при помощи сообщений и почтовых ящиков

# Scala.{ Futures }

```
import scala.concurrent._

val costInDollars = Future {
  webServiceProxy.getCostInDollars.mapTo[Int]
}

costInDollars map (myPurchase.setCostInDollars(_))
```

- Позволяют вам писать асинхронный код, который более производителен, чем последовательный
- При объединении с **lazy vals** дают еще большую гибкость

# Scala.{ m0ar }

- Lazy Definitions
- Implicits
  - Implicit Conversions
  - Implicit Parameters
  - Implicit Classes
- Type theory
  - Type inference
  - Type classes
  - Higher Kinded Types
  - Algebraic Data Types
- Macros
- Category Theory
  - Morphism
  - Functor
  - Monad

# Внезапно!

Переходя с Java на Scala, будьте готовы к **двукратному уменьшению** количества **строк кода**.

*Какое это имеет значение?*
*Разве Eclipse не допишет эти строки за меня?*

Эксперименты* показали, что для понимания программы среднее время затрачиваемое на одно слово исходного кода постоянно.

Другими словами: **в два раза меньше кода** значит **в два раза меньше времени на его понимание.**

*G. Dubochet. Computer Code as a Medium for Human Communication: Are Programming Languages Improving? In 21st Annual Psychology of Programming Interest Group Conference, pages 174-187, Limerick, Ireland, 2009.

# SBT: Simple Build Tool

# SBT.{ Overview }

- Простая настройка для простых проектов
- .sbt дескриптор использует Scala-based DSL
- Инкрементальная перекомпиляция
- Непрерывная компиляция и тестирование с triggered execution
- Поддержка смешанных Scala/Java проектов
- Тестирование с ScalaCheck, specs и ScalaTest
- Scala REPL
- Поддержка внешних проектов (git репозиторий как зависимость)
- Параллельное исполнение задач (в т.ч. тестов)
- Гибкое управление зависимостями (Ivy, Maven, manual)

# SBT.{ Directory Layout }

```
project/
  Build.scala
src/
  main/
    resources/
       <files to include in main jar here>
    scala/
       <main Scala sources>
    java/
       <main Java sources>
  test/
    resources
       <files to include in test jar here>
    scala/
       <test Scala sources>
    java/
       <test Java sources>
target/
 < compiled classes, packaged jars, managed files, and documentation >
build.sbt
```

# SBT.{ Build Definitions }.{ build.sbt }

```
name := "My Project"

version := "1.0"

libraryDependencies += "junit" % "junit" % "4.8" % "test"

libraryDependencies ++= Seq(
            "net.databinder" %% "dispatch-google" % "0.7.8",
            "net.databinder" %% "dispatch-meetup" % "0.7.8"
)

defaultExcludes ~= (filter => filter || "*~")

publishTo <<= version { (v: String) =>
  if(v endsWith "-SNAPSHOT")
    Some(ScalaToolsSnapshots)
  else
    Some(ScalaToolsReleases)
}
```

# SBT.{ Build Definitions }.{ build.sbt }

```
// define the repository to publish to
publishTo := Some("name" at "url")

parallelExecution in Test := true

// add SWT to the unmanaged classpath
unmanagedJars in Compile += Attributed.blank(file("/usr/share/java/swt.jar"))

javacOptions ++= Seq("-source", "1.7", "-target", "1.7")
initialCommands in console := "import myproject._"

watchSources <+= baseDirectory map { _ / "input" }

libraryDependencies +=
  "log4j" % "log4j" % "1.2.15" excludeAll(
    ExclusionRule(organization = "com.sun.jmx"),
    ExclusionRule(organization = "javax.jms")
  )
```

# SBT.{ Build Definitions }.{ Build.scala}

```
import sbt._
import Keys._

object AnyBuild extends Build {
   lazy val root = Project(id = "hello", base = file(".")) aggregate(foo, bar)

   lazy val foo = Project(id = "hello-foo",  base = file("foo"))

   lazy val bar = Project(id = "hello-bar", base = file("bar"))
}
```

- Ref:  Akka, Scalaz

# SBT.{ Usage }

**Interactive mode**

$ sbt
compile

**Batch mode**

$ sbt clean compile "test-only TestA TestB"

**Continuous build and test**

$ sbt
> ~ compile

# SBT.{ Commands }

## Common commands

- clean
- compile
- test
- console
- run <argument>*
- package
- help <command>
- reload

## History Commands

- !! - выполнить последнюю команду еще раз
- !:n - показать последние n команд
- !n - выполнить команду с индексом n, как из !:
- !string - выполнить последнюю команду, начинающуюся с 'string'

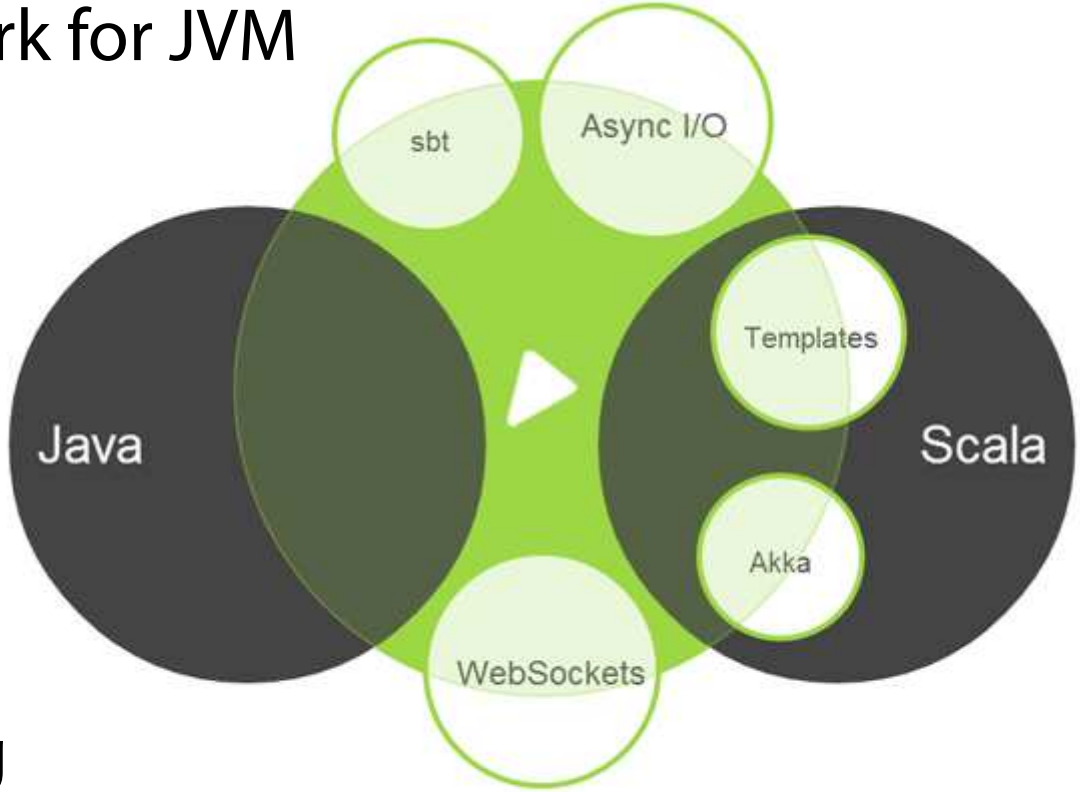
# SBT.{ m0ar }

- Custom Keys
- Scopes: configuration, project, and task
- Plugins
- Super-configurable Build.scala
- :=, +=, ++=, <+=, <++=, ~=

# Play! Framework

# Play.{ Overview }

- full stack web framework for JVM
- high-productive
- asynch & reactive
- stateless
- HTTP-centric
- typesafe
- scalable
- open source
- browser error reporting
- db evolutions
- integrated test framework




**Play 2.0**

# Play.{ Sample }

1. download Play 2.0 binary package

2. add play to your PATH:

```
export PATH=$PATH:/path/to/play20
```

3. create new project:

```
$ play new appName
```

4. run the project:

```
$ cd appName
$ play run
```

# Play.{ Directory Layout}

**app** → Application sources
- assets → Compiled asset sources
- controllers → Application controllers
- models → Application business layer
- views → Templates

**conf** → Configurations files
- application.conf → Main configuration file
- routes → Routes definition

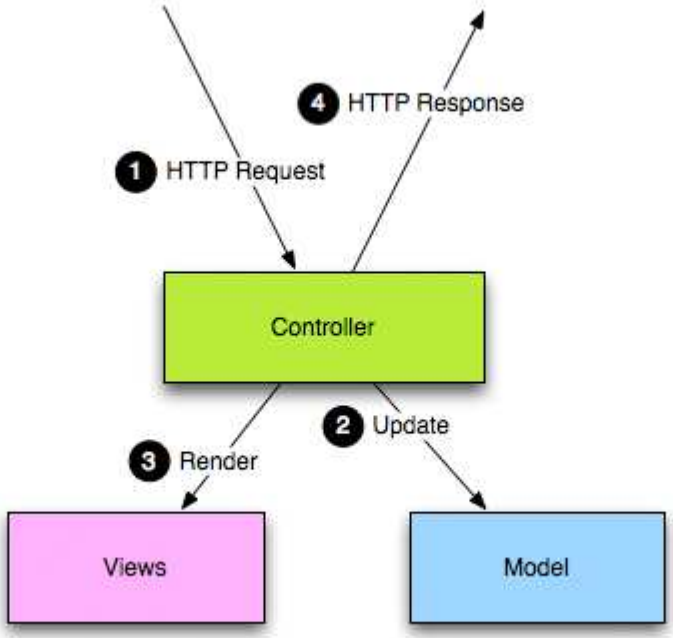
**public** → Public assets (/stylesheets, /javascripts, /images)

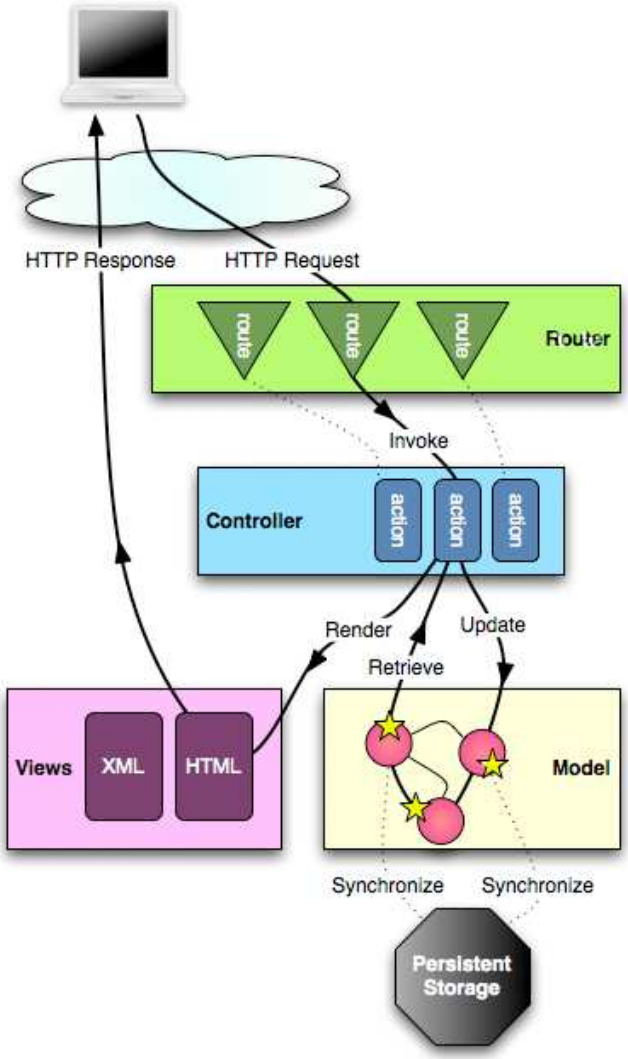
**project** → sbt configuration files
- build.properties → Marker for sbt project
- Build.scala → Application build script
- plugins.sbt → sbt plugins

**lib** → Unmanaged libraries dependencies

**test** → source folder for unit or functional tests

# Play.{ Request Life Cycle }



# Play.{ Templates }

views/**main**.scala.html:

```
@(title: String)(content: Html)
<!DOCTYPE html>
<html>
 <head>
  <title>@title</title>
 </head>
 <body>
  <section class="content">@content</section>
 </body>
</html>
```

views/**hello**.scala.html:

```
@(name: String = “Guest”)

@main(title = "Home") {
 <h1>Welcome @name! </h1>
}
```

then from Scala class:

```
val html = views.html.Application.hello(name)
```

# Play.{ Database Evolutions }

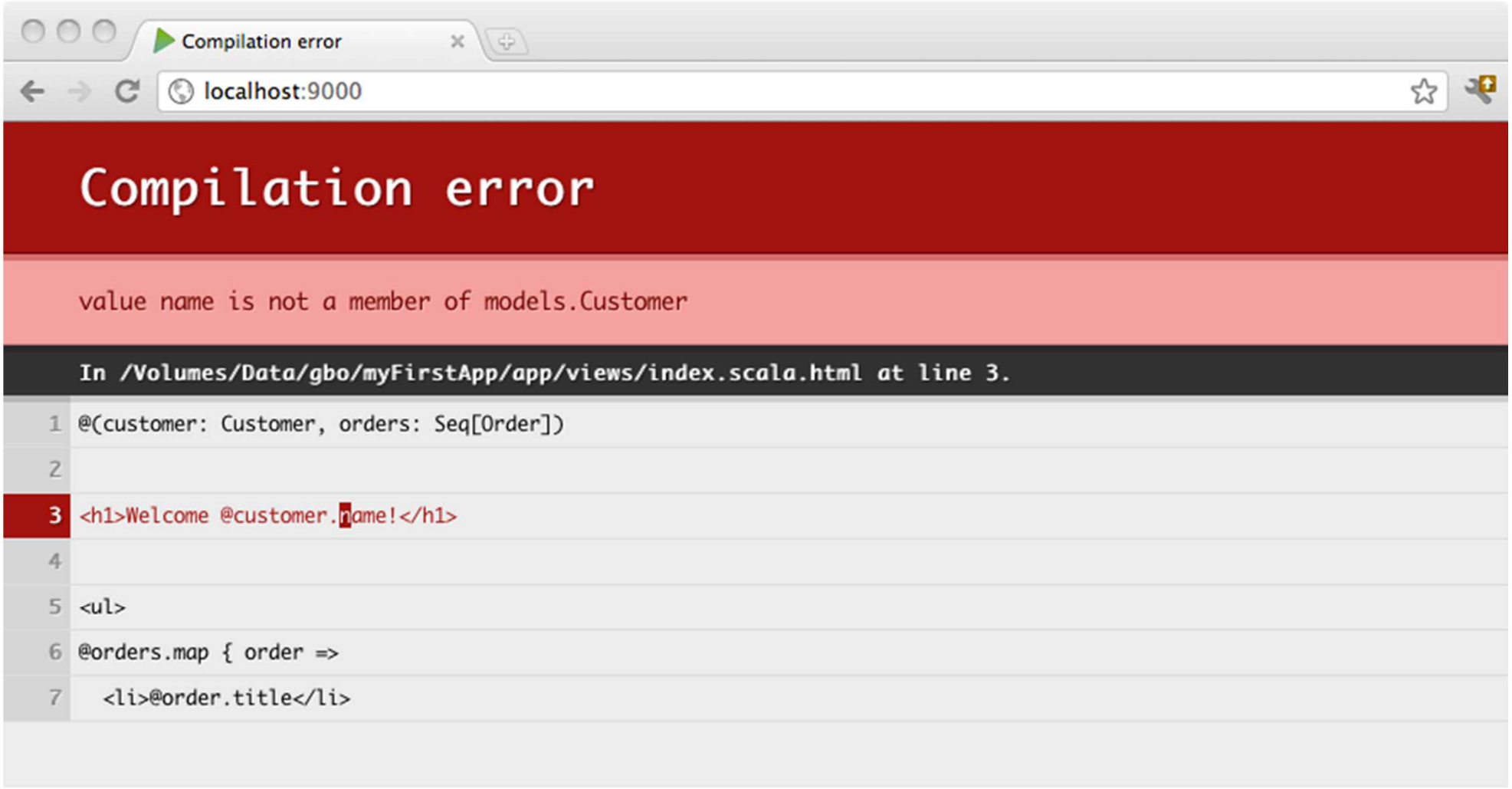
conf/evolutions/**${x}**.sql:

```
# Add Post

# --- !Ups
CREATE TABLE Post (
  id bigint(20) NOT NULL AUTO_INCREMENT,
  title varchar(255) NOT NULL,
  content text NOT NULL,
  postedAt date NOT NULL,
  author_id bigint(20) NOT NULL,
  FOREIGN KEY (author_id) REFERENCES User(id),
  PRIMARY KEY (id)
);

# --- !Downs
DROP TABLE Post;
```

# Play.{ Browser Error Reporting }



# Play.{ Database Access with Anorm }

```
import anorm._

DB.withConnection { implicit c =>

 val selectCountries = SQL("Select * from Country")

 // Transform the resulting Stream[Row] to a List[(String,String)]
 val countries = selectCountries().map(row =>
  row[String]("code") -> row[String]("name")
 ).toList
}
```

```
// using Parser API
import anorm.SqlParser._

val count: Long = SQL("select count(*) from Country").as(scalar[Long].single)

val result:List[String~Int] = {
 SQL("select * from Country")
 .as(get[String]("name")~get[Int]("population") map { case n~p => (n,p) } *)
}
```

# Play.{ m0ar}

- Streaming response
- Chunked results
- Comet
- WebSockets
- Caching API
- I18n
- Testing:
  - Templates
  - Controllers routes
  - Server
  - With browser
- IDE support: eclipsify, idea

# Success Story

# Success Story.{ Synopsis }

- Сервис сбора статистики по рейтингам мобильных приложений в сторах
- Параллельная выгрузка данных из источников и сохранение в БД
- Парсинг html-страниц с данными
- JSON-RPC 2.0
- Клиентская библиотека
- Reuse компонентов

- Ref:  /akirillov/rateslap

# Success Story.{ Solution Layout }

```
.
└ project
    └ Build.scala              // contains all dependency information
    └ plugins.sbt
└  service                     // can't get dependencies from parent project via play.sbt
    └ app
    └ conf
    └ logs
    └ project
        └ build.properties
        └ Build.scala          // shoulda specify via external dependency in local Ivy repo
        └ plugins.sbt
    └ public
    └ test
└ rs-client
    └ build.sbt
    └ src
└ rs-commons
└ rs-core
```

# Success Story.{ JSON Handlers }

```
# JSON-RPC Handler
POST    /rpc.json              controllers.JsonHandler.handleJsonRequest
```

```
object JsonHandler extends  Controller{

 def handleJsonRequest =  Action(parse.json) {
    request =>
     (request.body \ "method").asOpt[String].map { method => method match {
       case "getGamesStats" => Ok(AppHandler.getGameStats(request.body))
       case _ =>  Ok(ErrorConstructor.constructError((request.body \ "id").asOpt[String], -32601,
                                                                          "Method  not found"))
     }
     }.getOrElse {
       BadRequest(ErrorConstructor.constructError((request.body \ "id").asOpt[String], -32601,
                                                                          "Method not found"))
     }
 }
}
```

# Success Story.{ JSON Parsing }

```
def buildJsonResponse(request: StatsResponse, id: String): JsValue = {
  JsObject(
    Seq(
      "jsonrpc" -> JsString("2.0"),
      "result" -> JsObject(
        Seq(
          "application" -> JsString(request.application),
          "store" -> JsString(request.store),
          "rankType" -> JsString(request.rankType),
          if(request.rankings != null) {
                "rankings" -> Json.toJson (request.rankings)}
          } else {"rankings" -> JsNull},
          "error" -> JsString(request.error)
        )
      ),
      "id" -> JsString(id)
    )
  )
}
```

# Success Story.{ Error Reporting }

```
curl -v -H "Content-Type: application/json" -X GET -d '{"jsonrpc": "2.0", "method": "getGamesStats","params": {"application":"Cut The Rope", "store":"appstore", "dates":["2012-01-01","20120-01-02"],"rankType":"inapp", "countries":["USA","Canada"], "authObject":{"username":"anton", "password":"secret"}}, "id": "1"}' http://localhost:9000/rpc.json
```

```
<!DOCTYPE html>
<html>
[тонны html-ерунды]
        <body>
                <h1>Action not found</h1>
                <p id="detail">
                        For request 'GET /rpc.json'
                </p>
        </body>
</html>
```

**~compile рулит!**

# Success Story.{ XML Parsing }

```
override def parse(input: String) = {
   val source = XML.loadString(input)

   val countries = (source \\ "tr" filter(p  =>
            (p \ "@class").text.equals("ranks") && !((p \\ "a" text).equals(""))
   )).toList

   val ranks = (source \\ "tr" filter(p  =>  (p \ "@class").text.equals("ranks")))
    .filter (n => ((n \\ "td").size > 0) && ((n \\ "td").head \ "@title" text).equals("Rank #"))
    .map(r => (r \\ "td").head)
}
```

# Success Story.{ Actors }

```
lazy val crawler = new AppAnnieCrawler(request.AuthInfo)

  loopWhile(!jobFinished){
    react{

      case statsRequest: StatsRequest => {

        val ranks = Rank.find(request.application, request.rankType, date, request.countries)

        if (ranks.size == 0){
              new ParserActor(crawler).start ! SingleDateRequestWithData(date)
        } else {
          ...
        }
    }
```

# Success Story.{ Result }

- Стек содержит библиотеки для реализации большинства типовых задач
- Недостаточно развиты средства разработки и отладки, хотя есть определенные надежды
- Многопоточность с Actors Model гораздо более удобна, хотя и непривычна
- Отладка сервисов без UI несколько проблематична
- Применение evolutions без броузера проблематично
- Play приложение как часть комплекса влечет ряд проблем с автоматизированной сборкой

# Resources

- Fast Track:
  - Programming Scala by V. Subramanian
  - Zeroturnaround Scala Adoption Guide
- In-Depth:
  - Structure and Interpretation of Computer Programs
  - Programming in Scala by Odersky & Co
  - Coursera: Functional Programming Principles in Scala
- Bonus: Twitter Scala School

# Scala и RAD

- Уровень выбираете сами
- Крайне эффективный и гибкий язык
- Отличный веб-фреймворк с широкими возможностями
- SBT: сложность выбираете сами
- Не вместо, а вместе с Java
- У нас серьезный бизнес, а не Twitter –

**/twitter** (>100 opensource библиотек)

# Scala и RAD

```
ThisRoom getPeople foreach( person => {
        shakeHand(person)
        thanks(person)
}

> ~questions?
```