OUCC

Scala × Play Framework

Hands-on

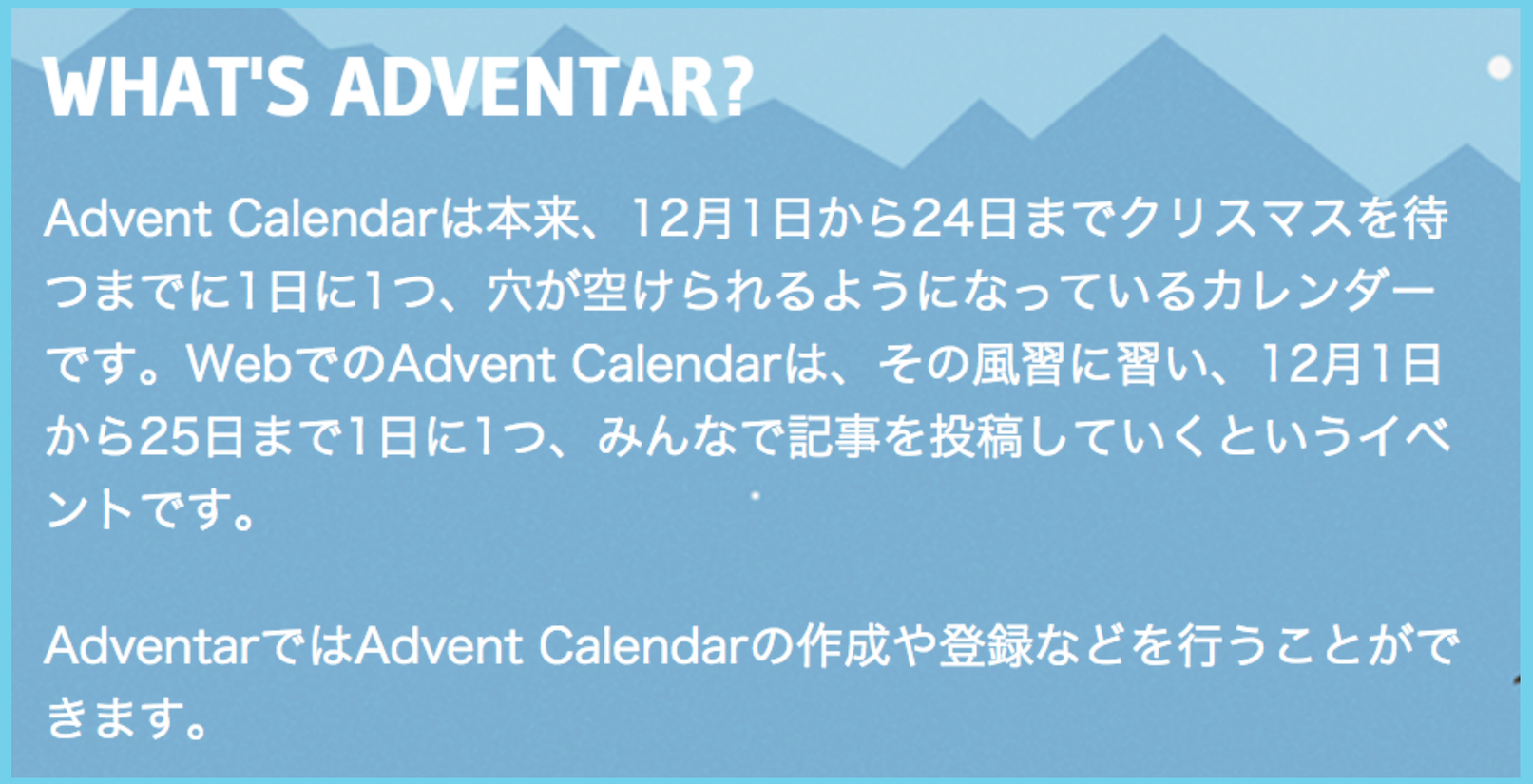
@spring_raining

宣伝

一緒にWebアプリ作ろう！！！！！！！！！

→→→Reply to @spring_raining

宣伝



OUCC Advent Calendar書きませんか

！！！！！？？？！！！！？？

→http://www.adventar.org/calendars/449

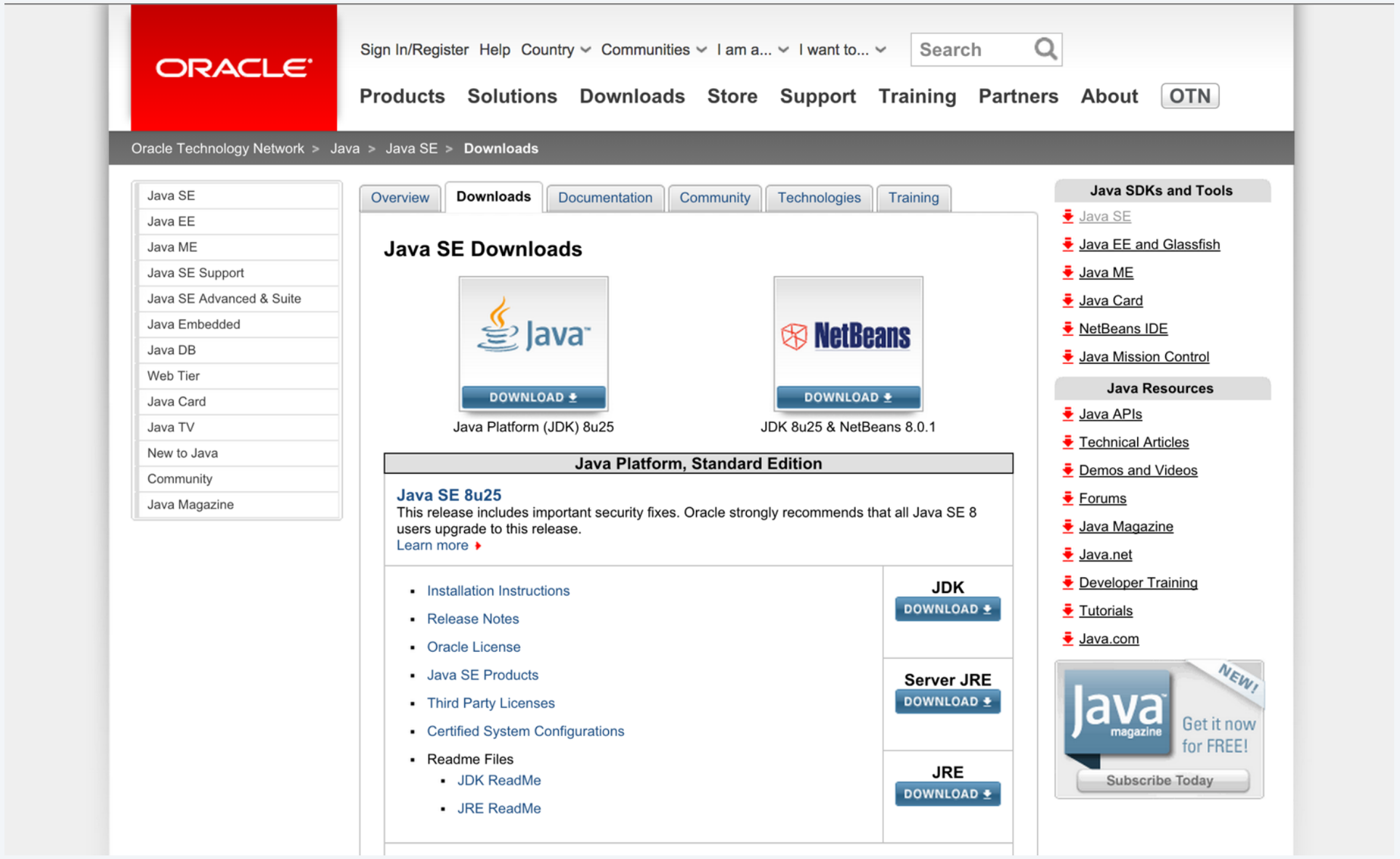
Scalaとは

- Javaと同じ環境で動く
- 簡潔な表記
- 関数型meetsオブジェクト指向

Play Frameworkとは

- Ruby on Rails / Django like
- たくさんの導入事例
- ★5,405

GETTING STARTED



「Java SE」で検索検索

「JDK」をインストール

Java SE Development Kit 8u25

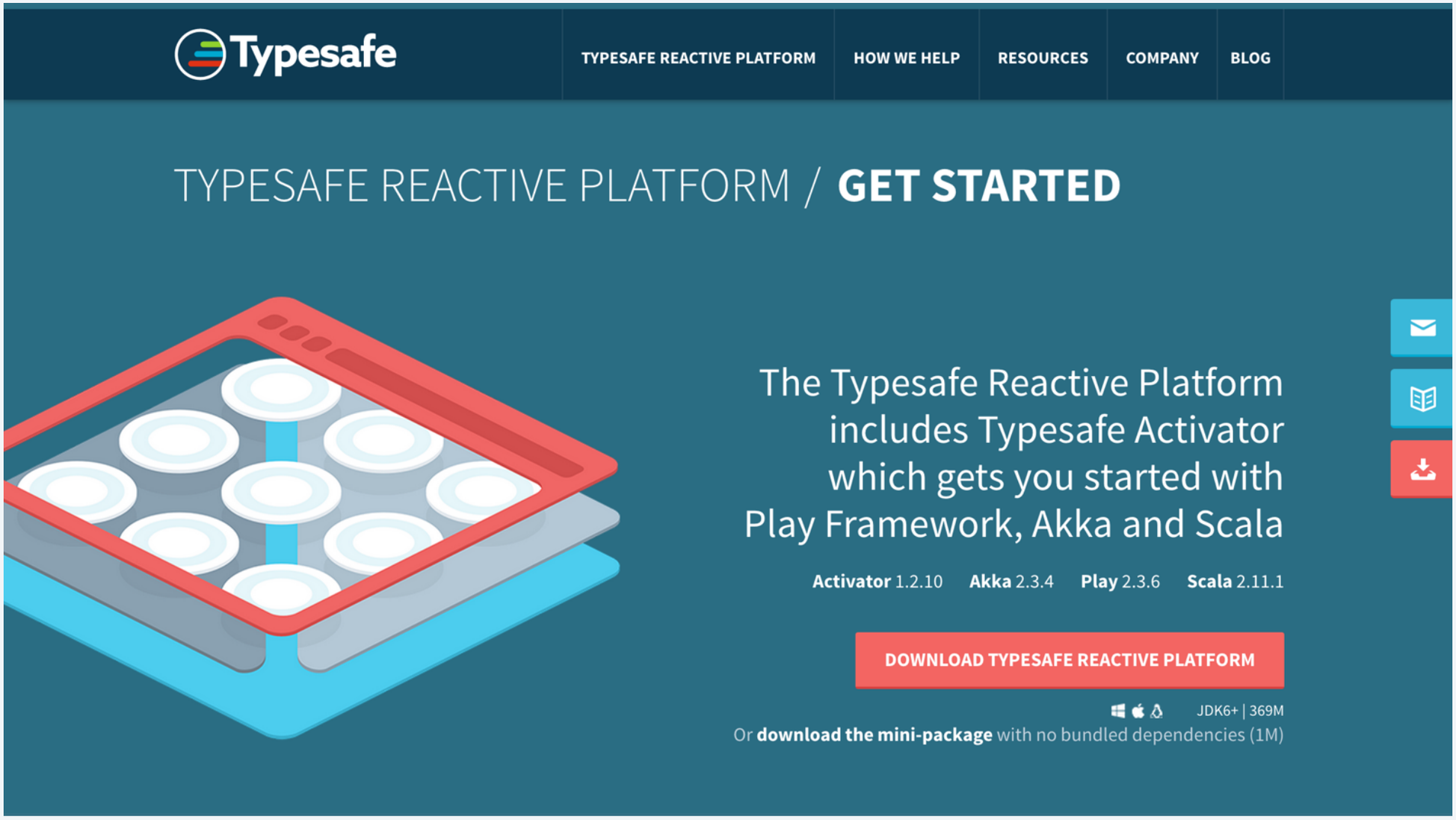
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ◉ **Decline License Agreement**

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u25-linux-i586.rpm
Linux x86	154.88 MB	jdk-8u25-linux-i586.tar.gz
Linux x64	135.6 MB	jdk-8u25-linux-x64.rpm
Linux x64	153.42 MB	jdk-8u25-linux-x64.tar.gz
Mac OS X x64	209.13 MB	jdk-8u25-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	137.01 MB	jdk-8u25-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.14 MB	jdk-8u25-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.11 MB	jdk-8u25-solaris-x64.tar.Z
Solaris x64	94.24 MB	jdk-8u25-solaris-x64.tar.gz
Windows x86	157.26 MB	jdk-8u25-windows-i586.exe
Windows x64	169.62 MB	jdk-8u25-windows-x64.exe

わかりにくい

「Accept License Agreement」をクリック



typesafe.com

Typesafe Activator をダウンロード & インストール

IDEはお好みで

IntelliJ IDEAを想定して進めます

ダウンロードが終わるまで...

Scalaの基本構文

SYNTAX

変数

```
var hoge: Int = 10
var fuga = "100"
println(hoge + fuga)
hoge = 20
fuga = 200
```

変数

```
var hoge: Int = 10
var fuga = "100"
```

```
hoge = 20
fuga = 200
```

var 名前 = 値

var 名前: 型 = 値

型は自動で推測される

変数

```
var hoge: Int = 10
var fuga = "100"
println(hoge + fuga)
```

→10110

```
fuga = 200
```

変数

```
var hoge: Int = 10
var fuga = "100"
```

```
hoge = 20
fuga = 200
```

→コンパイルエラー

fugaはString型なので

Int型は代入できない

定数

```
val teisuu = "konnitiwa"
println(teisuu)
```

val 名前 = 値

val 名前 : 型 = 値

定数は再代入できない

配列

```
val a = Array("yui", "yukari", "yuzuko")
println(a(0))
println(a.length)
```

Array(初期値..)

new Array(配列長)

new Array[型](配列長)

配列

```
val a = Array("yui", "yukari", "yuzuko")
println(a(0))
println(a.length)
```

配列の要素は[ ]ではなく

( )で取得

if

```
var foo: String = ""
foo = if (1 == 1) "yui" else "yuzuko"
val bar = if (false) "yukari"
println(bar)
```

if (条件式) 式

if (条件式) 式 else 式

if (条件式) { 式.. }

if (条件式) { 式.. } else { 式.. }

if

```
foo = if (1 == 1) "yui" else "yuzuko"
println(bar)
```

ifは「式」なので値を返す

返された値は

変数に代入される

if

```
var foo: String = ""
val bar = if (false) "yukari"
println(bar)
```

uzuko"

ifが値を返さない場合...

Unitというクラスを返す

(voidみたいなもの)

for (1)

```
val cast = Array("moffle", "macaron", "tiramy")
for (i <- cast) {
  println(i)
}
```

for (変数名 <- コレクション) { 式.. }

for (1)

```
offle", "macaron", "tiramy")
```

```
for (i <- cast) {
  println(i)
}
```

コレクションから要素を
1つずつ取り出して実行
(foreachみたいなやつ)

for (2)

```
for (i <- 0 until 5) {
  println(i)
}
```

Scalaにはいわゆる

for (〜 ; 〜 ; 〜) は使えない

for (2)

```
fo (i <- 0 until 5)
}
```

整数 until 整数

一旦Rangeクラスで整数の
コレクションを作り
そこから値を取り出す

for (3)

```
0 until 5
```
→ Range(0, 1, 2, 3, 4)

```
0 to 5
```
→ Range(0, 1, 2, 3, 4, 5)

```
0 to 5 by 2
```
→ Range(0, 2, 4)

```
5 to 0 by -1
```
→ Range(5, 4, 3, 2, 1, 0)

while

```
var i = 0
var summer = ""
while (i < 10) {
   summer += "hiji"
   i += 1
}
println(summer)
```

while (条件式) 式

while (条件式) { 式.. }

while

```
var i = 0
var summer = ""
while (i < 10) {
   summer += "hiji"
   i += 1
}
println(summer)
```

breakやcontinueは？

match (1)

```
val somebody = "miyako"
somebody match {
  case "yuno" =>
    println(144.3)
  case "miyako" =>
    println(165)
  case "nori" | "nazuna" =>
    println("?")
  case _ =>
    println("unknown")
}
```

変数 match {

  case パターン => 処理

}

match

=switchのすごいやつ

match (1)

```
val somebody = "miyako"
somebody match {
  case "yuno" =>
    println(144.3)
  case "miyako" =>

  case "nori" | "nazuna"

  case _
          ("unknown")
}
```

→nori またはnazuna

→default

match (2)

```
val animal = ("tanuki", "itachi", "araiguma")
animal match {
  case ("hakubishin", _, triple) =>
    println("hakubishin and " + triple)
  case (_, "itachi", triple) =>
    println("itachi and " + triple)
  case (_, _, triple) =>
    println(triple)
}
```

match (2)

```
val animal = ("tanuki", "itachi", "araiguma")
animal match {
  case ("hakubishin", _, tri
    println("hakubishin and
  case (_, "itachi", triple)
    println("itachi and " +
  case (_, _, triple) =>
    println(triple)
}
```

→タプル

配列に似てるけど

値の変更はできない

match (2)

```
val animal = ("tanuki", "itachi", "araiguma")
animal match {
  case ("hakubishin", _, triple) =>
    println("hakubishin and " + triple)
  case (_, "itachi", triple) =>
    println("itachi and " +
  case (_, _, triple) =>
    println(triple)
}
```

_ はワイルドカード
何でもアリ

match (2)

```
val animal = ("tanuki", "itachi", "araiguma")
animal match {
  case ("hakubishin", _, triple) =>
    println("hakubishin and " + triple)
  case (_, "itachi", triple) =>
    println("itachi and " + triple)
  case (_, _, triple) =>
    println(triple)
}
```

マッチした値は取得できる

match (3)

```
val someType: Any = "Yo"
someType match {
  case _: Int =>
    println("seisuu")
  case _: String =>
    println("mojiretsu")
  case _ =>
    println("unknown")
}
```

型を判定することも可能

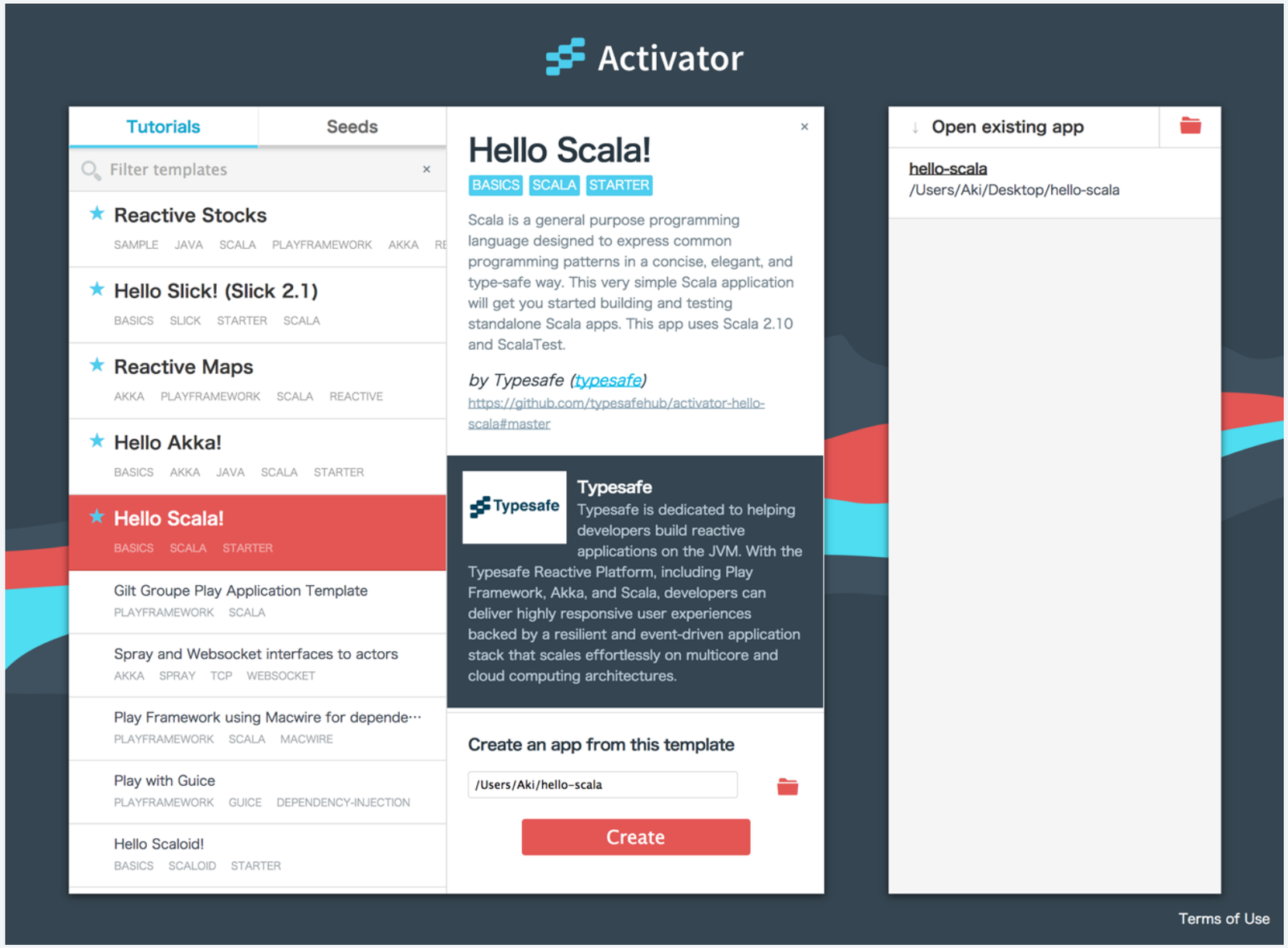
そろそろダウンロード

終わりそう

```
☁  ~  activator ui
Checking for a newer version of Activator (current version 1.2
   ... our current version 1.2.10 looks like the latest.
Found previous process id: 5495
FOUND REPO = activator-local @ file:/usr/local/Cellar/typesafe
Play server process ID is 5585
[info] play - Application started (Prod)
[info] play - Listening for HTTP on /127.0.0.1:8888
[info] a.e.s.Slf4jLogger - Slf4jLogger started
[INFO] [11/22/2014 14:07:24.582] [default-akka.actor.default-d
[warn] e.m.m.MimeDetectorRegistry - MimeDetector [eu.medsea.mi
s already registered.
[warn] e.m.m.MimeDetectorRegistry - MimeDetector [eu.medsea.mi
s already registered.
[warn] e.m.m.MimeDetectorRegistry - MimeDetector [eu.medsea.mi
s already registered.
```

activator ui

起動するとブラウザが立ち上がる



Hello Scala!

作成したらCode view & Open in IDE を選択

Hello world

```
object Hello {
  def main(args: Array[String]): Unit = {
    println("Hello, world!")
  }
}
```

Hello world

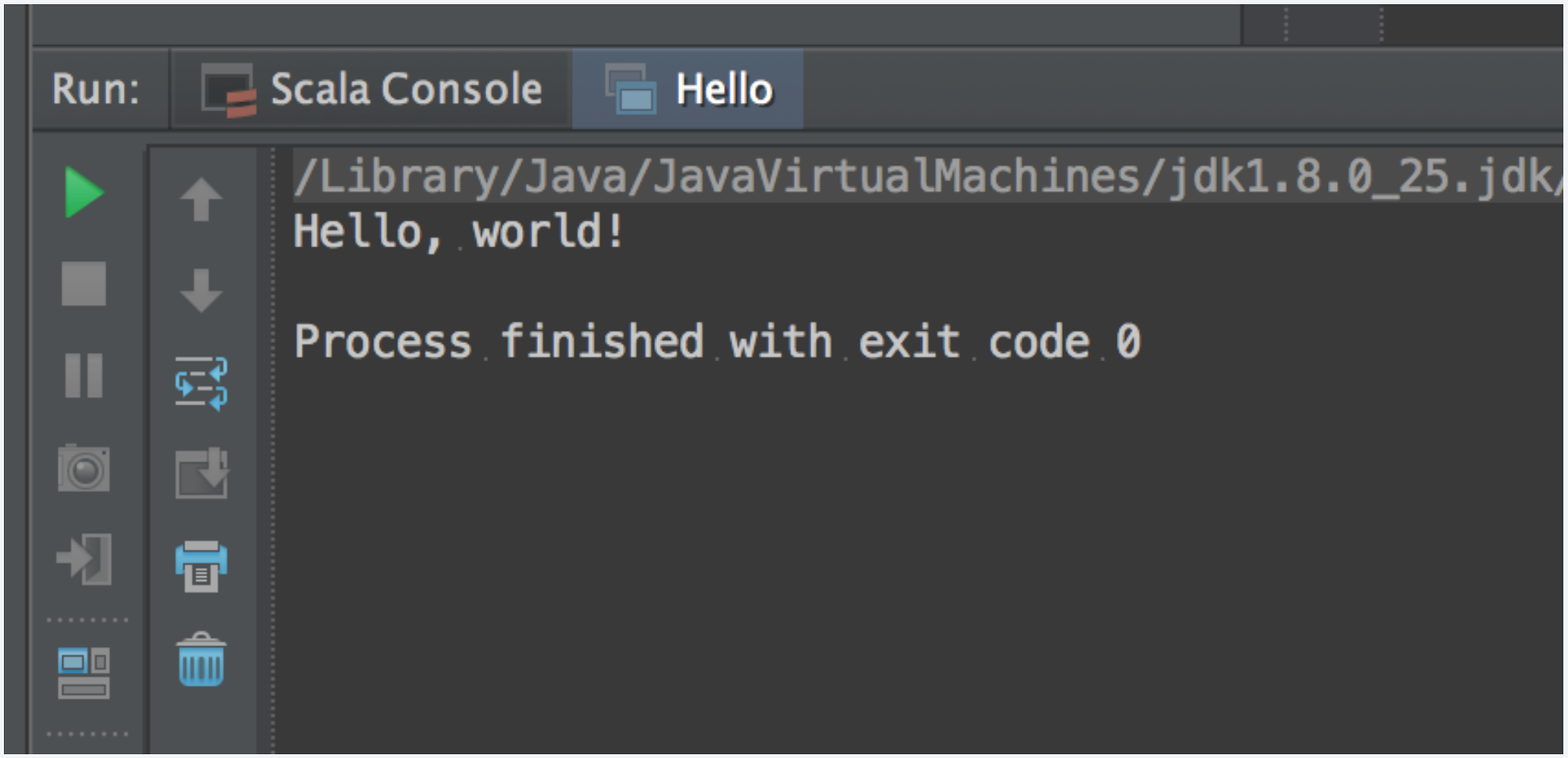
```
def main(args: Array[String]): Unit = {
  println("Hello, world!")
}
```

main関数

Hello world

```
object Hello {
  def main(args: Array[String]): Unit = {
    println("Hello, world!")
  }
}
```

Helloオブジェクト


Run:
Scala Console
Hello
/Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/
Hello, world!
Process finished with exit code 0

FUNCTION

CLASS

OBJECT

Function(1)

```
def pow (a: Int, b: Int): Int = {
  if (b <= 1) a
  else pow(a, b - 1) * a
}

println(pow(2, 3))
```

def 関数名(引数名: 型, ..): 戻型 = { .. }

def 関数名(引数名: 型, ..) = { .. }

Function(1)

```
def pow (a: Int, b: Int): Int = {
  if (b <= 1) a
  else pow(a, b - 1) * a
}
```

```
println(pow(2, 3))
```

returnは省略可

Function(2)

```
val max = (a: Int, b: Int) => {
  if (a > b) a else b
}

println(max(2, 3))
```

関数名(引数名: 型, ..) => { .. }: 戻型

関数名(引数名: 型, ..) => { .. }

Function(2)

```
val max = (a: Int, b: Int) => {
  if (a > b) a else b
}

println(max(2, 3))
```

Scalaは関数自体を変数にしたり

返値にしたりできる

=関数リテラル

Function(3)

```
def multiply(a: Int)(b: Int) = {
  a * b
}
val twice = multiply(2)

println(twice(3))
```

1つの引数リストは複数の
引数に分割することができる
=カリー化

Function(3)

```
def multiply(a: Int)(b: Int) = {
  a * b
}

val twice = multiply(2)

println(twice(3))
```

twiceは(Int)=>Int型の関数

カリー化した関数に
引数を部分的に与えると
新しい関数が作れる

Class(1)

```
class Ship(n: String) {
  val name = n
  val level = 1
  val equipments = new Array[String](4)
}

val takao = new Ship("Takao")
println(takao.level)
```

class クラス名 = { .. }

class クラス名(引数定義) = { .. }

Class(1)

```
class Ship(n: String) {
  val name = n
  val level = 1
  val equipments = new Array[String](4)
}

val takao = new Ship("Takao")
println(ta
```

new クラス名

でインスタンス生成

Class(2)

```
class Ship(n: String) {
  val name = n
}

final class Battleship(n: String) extends Ship(n) {
  def explain = "[Battleship] " + name
}

val kirishima = new Battleship("Kirishima")
println(kirishima explain)
```

継承：元あるクラスから
新しいクラスを作る

Class(2)

```
class Ship(n: String) {
  val name = n
}

final class Battleship(n: String) extends Ship(n)

}

val kirishima = new Battleship("Kirishima")
println(kirishima explain)
```

extendで継承

finalでそれ以上継承させない

Object(1)

```
object RabbitHouse {
  var member = List("Chino", "Rize")
  def invite(person: String) = {
    member = person :: member
  }
}

RabbitHouse.invite("Cocoa")
println(RabbitHouse.member)
```

Scalaにはstaticメソッドはないかわりにobjectでシングルトンオブジェクトが作れる

Object(1)

```
object RabbitHouse {
                                  hino", "Rize")
  def invite(person: String) = {
    member = person :: member
  }
}

RabbitHouse.invite("Cocoa")
println(RabbitHouse.member)
```

object オブジェクト名 = { .. }

シングルトンオブジェクト

=インスタンスが1つしか

作れないクラス

Object(2)

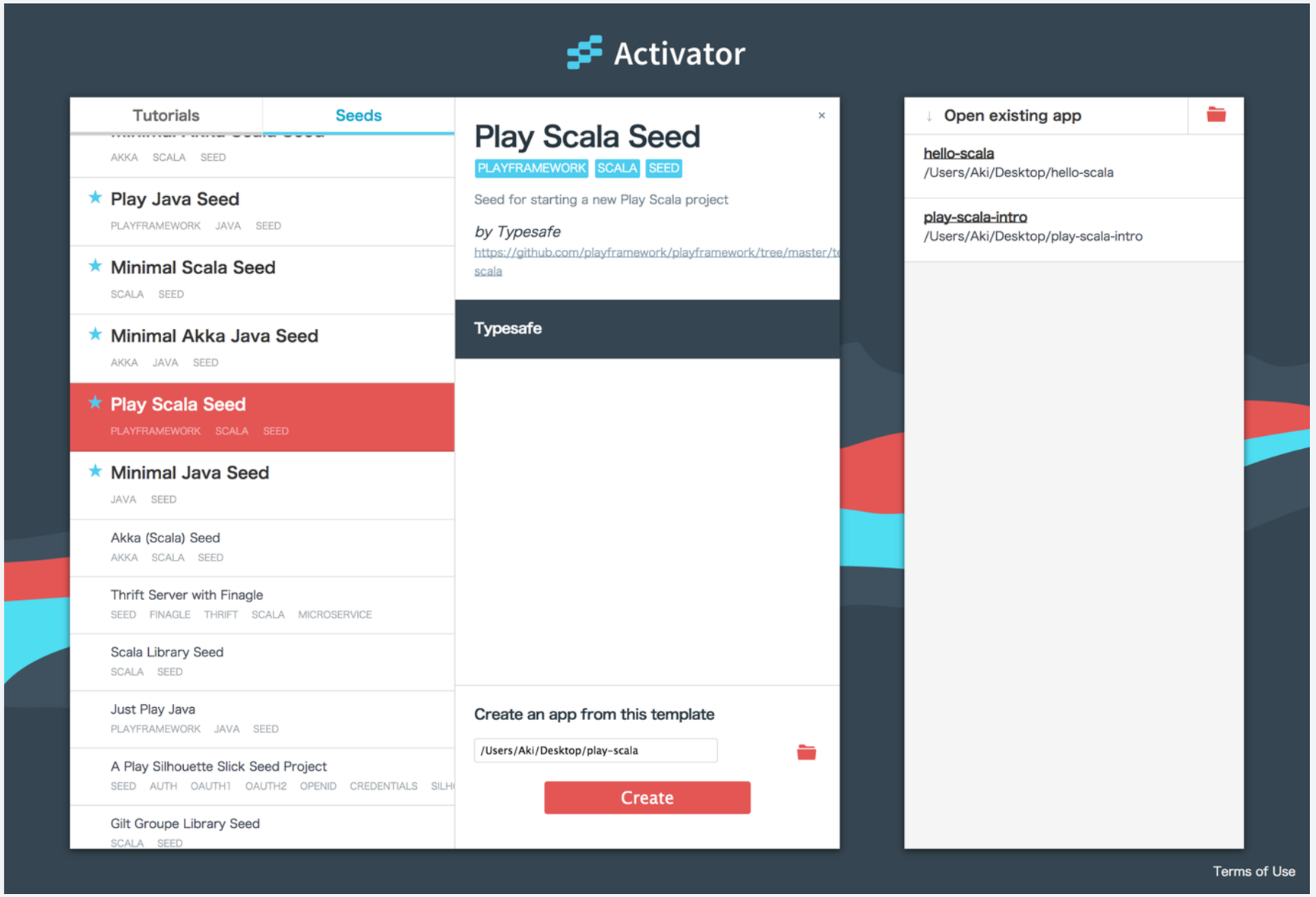
```
class RabbitHouse {
  var member = List("Chino", "Rize", "Cocoa")
}

object RabbitHouse {
  def apply() = new RabbitHouse
}

val rh = RabbitHouse()
println(rh.member)
```

applyメソッドは
呼び出す際名前を省略できる

他にもいろいろあるけど

ひとまず終わり

Hello Play!



Play Scala Seed

ダウンロード時間むっちゃかかる

ダウンロードが終わるまで...

WAFについて

Web application framework?

Webアプリの基本

Webページ
見せて

HTMLとか

URL mapping

Web API

Security

REST

めんどい

Cashing

Ajax

Database

Web template

そこでWAF

Webアプリを制作する際のめんどい部分があらかじめ用意されている

MVC

= Model View Controller

Webアプリでよく使われる設計パターン

Play Frameworkでも使われている

MVC

Model

Controller

View

そろそろダウンロード

終わりそう

```
☁  Desktop  cd play-scala
☁  play-scala  activator run
[info] Loading project definition from /Users/Aki/Desktop/play-scala/
[info] Set current project to play-scala (in build file:/Users/Aki/De

--- (Running the application from SBT, auto-reloading is enabled) ---

[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Ctrl+D to stop and go back to the console...)
```

今度こそHello Play

ダウンロードした場所でactivator run
localhost:9000にアクセス

Welcome to Play

localhost:9000

これは 英語 のページです。翻訳しますか？ いいえ 翻訳

Your new application is ready.

Browse APIs

Welcome to Play

Congratulations, you've just created a new Play application. This page will help you with the next few steps.

You're using Play 2.3.6

Why do you see this page?

The `conf/routes` file defines a route that tells Play to invoke the `Application.index` action whenever a browser requests the `/` URI using the GET method:

```
# Home page
GET     /           controllers.Application.index
```

Play has invoked the `controllers.Application.index` method to obtain the `Action` to execute:

Browse

- Local documentation
- Browse the Scala API

Start here

- Using the Play console
- Setting up your preferred IDE

How it works?

Playのファイル構造

app/	→アプリのソースコード
build.sbt	→ビルドスクリプト
conf/	→設定ファイル
logs/	→ログファイル
project/	→sbt設定ファイル
public/	→アセット(画像ファイルとか)
target/	→自動で生成されるファイル
test/	→テストのソースコード

sbt?

Scalaのファイルをビルドするツール

ライブラリ管理もしてくれるらしい

~~(よく知らない)~~

/conf/routes

```
GET     /                           controllers.Application.index

GET     /assets/*file               controllers.Assets.at(path="/public", file)
```

アクセスしてきた場所によって
どのページを見せるかを決める

/conf/routes

```
GET     /
            controllers.Application.index
```

```
GET     /assets/*file
            controllers.Assets.at(path="/public", file)
```

/ にアクセスしてきたら

controllers.Application.indexを実行

/app/controller/Application.scala

```
object Application extends Controller {
  def index = Action {
    Ok(views.html.index("Your new application is ready."))
  }
}
```

コントローラーの内容は
Controllerを継承したオブジェクトに
書かれている

/app/controller/Application.scala

```
def index = Action {
  Ok(views.html.index("Your new application is ready."))
}
```

→ `def index = Action.apply(Ok(views.html.index("...")))`

indexが実行されたら

Action.apply()を返す

/app/controller/Application.scala

```
object Application extends Controller {

  def index = Action {
    Ok(views.html.index("Your new application is ready."))
  }

}
```

ビューのindexファイルを
OKのステータスで返す

/app/views/index.scala.html

```
@(message: String)

@main("Welcome to Play") {

    @play20.welcome(message)

}
```

.htmlファイルだけど
普通のhtmlファイルではない

Template Engine

データを元に.scala.htmlテンプレートから
出力するWebページを生成する

Template engine(1)

```
<html>
  <head>
    <title>@title</title>
  </head>
  <body>
    @content
  </body>
</html>
```

先頭に@をつけると

その部分はScalaのコード

Template engine(1)

```
<html>
  <head>
    <titl  @title  title>
  </head>
  @content
</html>
```

@変数 で

変数の中身が出力

Template engine(2)

```
<ul>
@for(u <- users) {
  <li>@u.name : @u.age</li>
  @if(u.age >= 20) {
    <div>adult</div>
  }
}
</ul>
```

@for() { .. } とか

@if() { .. } とか

Template engine(3)

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

テンプレートは関数のように
引数を取ることができる

Template engine(3)

```
@(title: String)(content: Html)
```

```
<html>
  <head>
    <title>@title</title>
  </head>
  <body>
    @content
  </body>
</html>
```

先頭行にScalaの関数っぽく

引数の定義をする

Template engine(4)

```
@(message: String)

@main("Welcome to Play") {

   @play20.welcome(message)

}
```

引数を定義すると
別ファイルからテンプレートと
同じ名前で関数が呼べる

Template engine(4)

```
@(message: String)

@main("Welcome to Play") {

  @play20.welcome(message)

}
```

引数として

"Welcome to Play”と

play20.welcome(message)

をmainに与えている

まとめると

Application.scala

routes

index.scala.html

お疲れ様でした

発表は終わりです
各自でファイルの中身を変えて
おもしろWebアプリを作ってみてください