Roma
20-23.03.2013
www.codemotionworld.com

Massimiliano Dessì
&
Alberto Quario

Scala On Web
Play - Scalatra - Spray

Google Technology User Group Sardegna

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Speakers

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

Max has more than 13 years of experience in programming. He's a proud father of three.
Manager of GTUG Sardegna, Founder of SpringFramework IT, co-founder of Jug Sardegna.
Author of Spring 2.5 AOP. He works in Energeya and lives in Cagliari, Italy.

Alberto has more than 15 years experience in developing software, he wrote his first programs on a TI-99/4A and hasn't stopped since. Other than languages and development, Alberto's passions include squash, cooking and Monet paintings. He lives and works in Milano, Italy.

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Agenda

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

Scalatra -Playframework -Spray

REST

JSON

Routing

Template

Actors

Sessions

Deploy

Hot Reloading

Test

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra
is a web microframework
written in Scala
inspired to Sinatra
a Ruby DSL to build webapp
http://www.scalatra.org

the BBC, LinkedIn,
the Guardian,
games website IGN,
UK government
rely on Scalatra.

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
class JellyBeans extends ScalatraServlet {

  get("/jellybs/:id") { ... }

  post("/jellybs") { ... }

  put("/jellybs/:id") { ... }

  delete("/jellybs/:id") { ... }

}
```

For browser add PUT & DELETE support client side
`X-HTTP-METHOD-OVERRIDE` or `_method=put` `_method=delete` in the post body

```
class JellyBeansBrowser extends ScalatraFilter with MethodOverride
```

{codemotion}
Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Routes

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class JellyBeans extends ScalatraServlet {

  get("/jellybs/:id") {

        val id:Int = params.getOrElse("id", halt(400)).toInt
        ...
  }

  get("/jellybs/download/*.*") {    //with wildcard
     ...
 }

get("""^\/j(.*)/f(.*)""".or) {    //REGEX
     ...
 }

post("/jellybs", request.getRemoteHost == "127.0.0.1",
    request.getRemoteUser == "admin") {
    ....
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Filter

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class JellyBeans extends ScalatraServlet {

   before() {
      db.acquireConnecion
      contentType="text/html"
   }

   get("/") {
      val menu = db.findWelcome()
      templateEngine.layout("index.ssp", menu)
   }

   after() {
      db.releaseConnection
   }
}
```

Like Servlet Filter
(or Aspect Oriented Programming)
You can add logic before or the routes

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Filter

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class JellyBeans extends ScalatraServlet {

   before("""/api/v1/.*""".r) {
       contentType = "application/json"
   }

   before("/admin/*") { auth }

   after("/admin/*") { user.logout }


}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Handlers

Handlers are top level methods for http routines

```
class JellyBeans extends ScalatraServlet {


   notFound {
     <h1>Not found</h1>
   }

   halt(status = 301, headers =
                Map("Location" -> "http://www.codemotion.com/"))


   get("/jellybs/names/*") {
     "Name not found!"
   }

}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Handlers

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class JellyBeans extends ScalatraServlet {

   get("/jellybs/names/:who") {
     params("who") match {
       case "Cherry" => "Found Cherry!"
       case _ => pass() /* call the next matching route route, routes are
   matched from bottom up*/
     }
   }

 get("/jellybs/download/:id") {
        jellyBeanService.find(params("id")) match {
           case Some(jellybean) => Ok(jellybean)
           case None => NotFound("Sorry, jellybean not found")
        }
 }

}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Request Response & Friends

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class JellyBeans extends ScalatraServlet {

    get("/jellybs/shows/:id") {
       //access to request,response, session and params
       request.body                          //request body as a string
       request.cookies                       // cookie map
       request.isAjax                        // is ajaxRequest
       request.getSession                    // HttpServletSession
       request.locale                        // user locale
       response.getOutputStream              //response outputstream
       servletContext.get("myIntParam")  //servlet context
       val idString = params("id")
       val id = params.getAs[Int]("id")
     //val id = params.getOrElse("id", halt(500)) //another way
        ....
    }
}
```

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
class JellyBeans extends ScalatraServlet {


  get("/jellybs/*/conf/*") {
      // Matches "GET /jellybs/one/conf/two"
      multiParams("splat") // Seq("one", "two")
      //wildcard accessible with key splat
   }

  get("""^\/jelly(.*)/fla(.*)""".r) {
    // Matches "GET /jellybs/flavor"
    multiParams("captures") // == Seq("bs", "vor")
  }


}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

To avoid manual binding from http and our Object Scalatra provide a binding module

```
case class MyClass(id: Integer, name: String)


post("/myroute") {
    val cmd = command[CreateMyClassCommand]
    ...
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Under the hood

```
abstract class MyClassCommand[S](implicit mf: Manifest[S])
                                              extends ModelCommand[S] with JsonCommand {

  implicit def todoStringValidators(b: FieldDescriptor[String]) =
                                                              new MyClassStringValidations(b)
}


class CreateMyClassCommand extends MyClassCommand[MyClass] {

   protected implicit val jsonFormats = DefaultFormats
   val name: Field[String] = asType[String]("name").notBlank.minLength(3)
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Under the hood

```
class MyClassStringValidations(b: FieldDescriptor[String]) {
   def startsWithCap(message: String = "%s must start with a capital letter.") =
   b.validateWith(_ =>
                    _ flatMap {
                                 new PredicateValidator[String](
                                    b.name,
                                    """^[A-Z,0-9]""".r.findFirstIn(_).isDefined,
                                    message).validate(_)
                                 )
                               }
                  )
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Gzip

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class GzipJellyBeans extends ScalatraServlet with GZipSupport{
   get("/") {
    <html>
      <body>
        <h1>This is
          <a href="http://en.wikipedia.org/wiki/Sparta">
            http/gzip
          </a>!
        </h1>
      </body>
    </html>
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Flash message

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class FlashServlet extends ScalatraServlet with FlashMapSupport{

  post("/jellybs/create") {
    flash("notice") = "jellybean created successfully"
    redirect("/home")
  }

  get("/home") {
    ssp("/home")   //Scala Server Pages
  }
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Automatic Serialization and deserialization of any Case class

```
object JSONServlet extends ScalatraServlet with JacksonJsonSupport{

  case class JellyBean(id: Int, name: String, flavor:String)


protected implicit val jsonFormats: Formats = DefaultFormats


before() {
    contentType = formats("json")
}

....
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra JSON

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

Automatic Serialization and deserialization of any Case class

```
object JSONServlet extends ScalatraServlet with JacksonJsonSupport{

  ...

  get("/jellybs/all"){
    jellyBeanRepo.all   //from class to json
  }

  post("/jellybs/create") {
    val jb = parsedBody.extract[JellyBean] //from json to class
    ...
  }

}
```

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Html inline,
Scalate
Twirl (Play2 template)

Scalate
mean
SSP (Scala Server Page)
Scaml
Mustache
Jade

{codemotion}
Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Template

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
class ScalateServlet extends ScalatraServlet with ScalateSupport{

  get("/"jellybeans/ssp) {
    contentType="text/html"
    ssp("/index", "foo" -> "uno", "bar" -> "two")
    // the layout used it's WEB-INF/layouts/default.ssp
  }

 get("/jellybeans/jade") {
    jade("/index", "layout" -> "", "foo" -> "one", "bar" -> "two")
    // render without a layout.
}

get("/jellybeans/direct") {
    templateEngine.layout("/jellybeans/index.ssp")
  //direct invoking of scalate
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
SSP

<%@ var body: String %>
<%@ var title: String = "Some Default Title" %>
<%@ var head: String = "" %>
<html>
<head>
  <title>${title}</title>

  <%-- page specific head goes here --%>
  ${unescape(head)}
</head>
<body>
  <p>layout header goes here...</p>

  ${unescape(body)}

  <p>layout footer goes here...</p>
</body>
</html>
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
JADE

!!! 5
html(lang="en")
  head
    title= pageTitle
    :javascript
      if (foo) {
         bar()
      }
  body
    h1 Jade - node template engine
    #container
      - if (youAreUsingJade)
        p You are amazing
      - else
        p Get on it!
    :coffeescript
      alert "Hello, Coffee!"
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
Mustache

Hello {{name}}
You have just won ${{value}}!
{{#in_ca}}
Well, ${{taxed_value}}, after taxes.
{{/in_ca}}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Template

Massimiliano Dessì & Alberto Quario

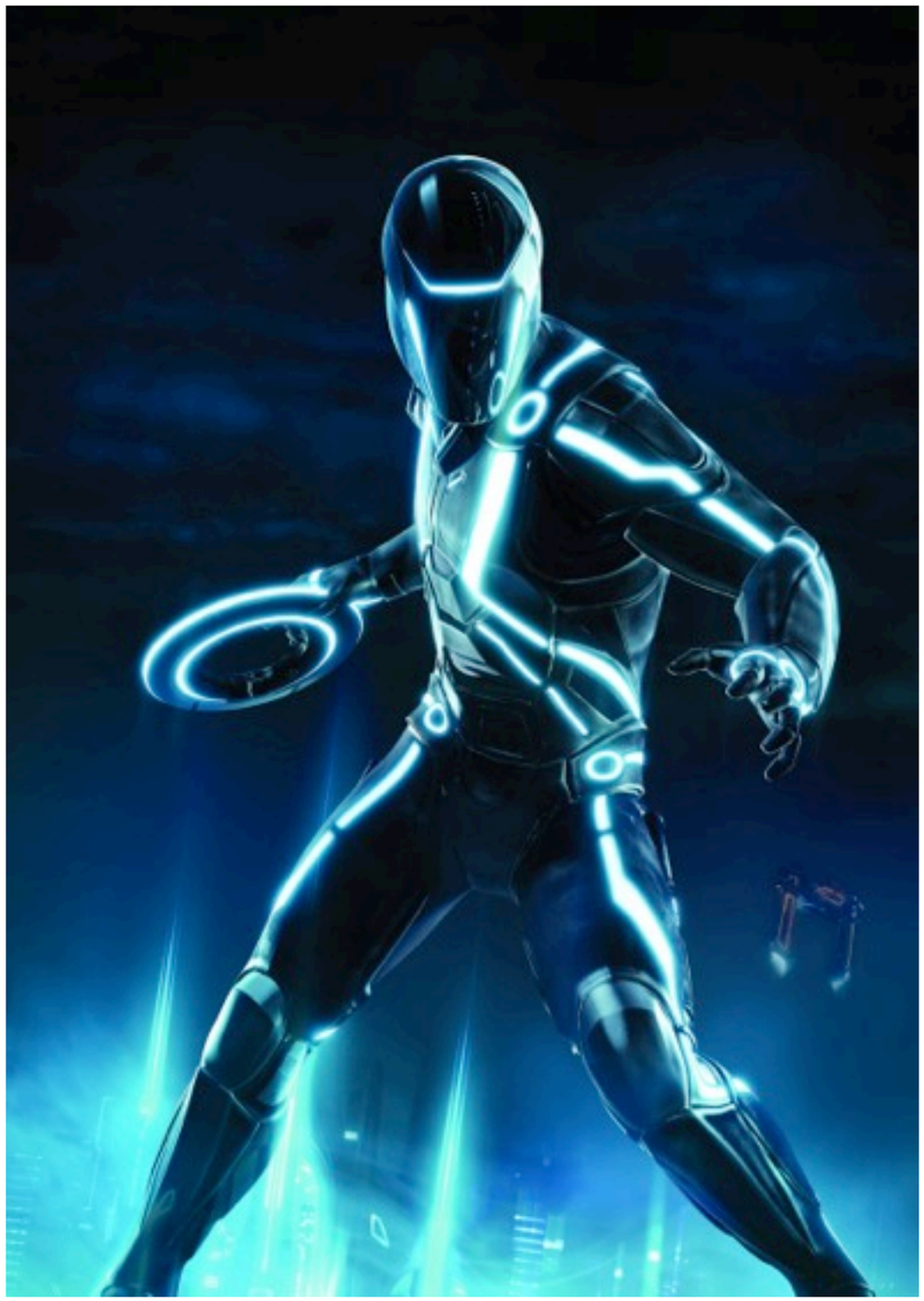
GTUG Sardegna

```
SCAML

!!! XML
!!!
%html
  %head
    %title Myspace
%body
  %h1 I am the international space station
  %p Sign my guestbook
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Non Blocking and Async with Akka and Actors



Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

The routes can returns a Future

```
class MyActorServlet(system:ActorSystem, myActor:ActorRef)
                        extends ScalatraServlet with FutureSupport {

  import _root_.akka.pattern.ask
  implicit val timeout = Timeout(10)
  protected implicit def executor: ExecutionContext = system.dispatcher

  get("/async") {
    myActor ? "Do stuff and give me an answer"
  }

  get("/fire-forget") {
    myActor ! "Hey, you know what?"
    Accepted() //if you do not want return a future
  }


}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
get("/async") {
  myActor ? "Do stuff and give me an answer"
}


get("/fire-forget") {
  myActor ! "Hey, you know what?"
  Accepted() //if you don't want return a future
}

class CodemotionActor extends Actor {
  def receive = {
     case "Do stuff and give me an answer" => sender ! "The answer is 42"
     case "Hey, you know what?" => println("Yeah I know...")
  }
}

}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Akka start in the Bootstrap of Scalatra

```
class ScalatraBootstrap extends LifeCycle {

 val system = ActorSystem()
 val codemotionActor = system.actorOf(Props[CodemotionActor])

  override def init(context: ServletContext) {
    context.mount(new JSONServlet, "/jellybs/*")
    context.mount(new FrontServlet, "/template/*")
    context.mount(new MyActorServlet(system, codemotionActor), "/actors/*")
  }

  override def destroy(context:ServletContext) {
     system.shutdown()
   }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra uses Scentry a porting of Ruby Warden authentication

```
class OurAuthStrategy(protected override val app: ScalatraBase, realm: String)
  extends BasicAuthStrategy[User](app, realm) {

  protected def validate(userName: String, password: String): Option[User] = {
    if(userName == "myusername" && password == "secret") Some(User("myusername"))
    else None
  }

  protected def getUserId(user: User): String = user.id
}



case class User(id: String)
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Now we need to combine our strategy and ScentrySupport

```
trait AuthenticationSupport extends ScentrySupport[User] with BasicAuthSupport[User] {
  self: ScalatraBase =>

  val realm = "Scalatra Basic Auth Example"

  protected def fromSession = { case id: String => User(id) }
  protected def toSession = { case usr: User => usr.id }
  protected val scentryConfig = (new ScentryConfig {}).asInstanceOf[ScentryConfiguration]


}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Now we need to combine our strategy and ScentrySupport

```
trait AuthenticationSupport extends ScentrySupport[User] with BasicAuthSupport[User] {
  ....

  override protected def configureScentry = {
    scentry.unauthenticated {
      scentry.strategies("Basic").unauthenticated()
      scentry.strategies("")
    }
  }

  override protected def registerAuthStrategies = {
    scentry.register("Basic", app => new OurBasicAuthStrategy(app, realm))
  }

  protected def validate(userName: String, password: String): Option[User] = {
     if(userName == "scalatra" && password == "scalatra") Some(User("scalatra"))
     else None
   }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Now we can use the Authentication for all routes defined in our Authenticated Servlets

```
class AuthenticatedServlet extends ScalatraServlet with AuthenticationSupport{

    //every route goes under authentication

}
```

Unauthenticated user will see a browser prompt login

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Deployment

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

The simplest way to deploy
your Scalatra application
is as a
Web application **AR**chive file

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Scalatra Deployment

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

Scalatra is based on regular Java Servlet 3.0
it can start from:
Standalone from jetty embedded
From Servlet container
Heroku
Jelastic
CloudBees
GAE (not out of the box)

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      version="3.0">
    <listener>
      <listener-class>org.scalatra.servlet.ScalatraListener</listener-class>
    </listener>

    <servlet-mapping>
      <servlet-name>default</servlet-name>
      <url-pattern>/img/*</url-pattern>
      <url-pattern>/css/*</url-pattern>
      <url-pattern>/js/*</url-pattern>
      <url-pattern>/assets/*</url-pattern>
    </servlet-mapping>
</web-app>
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Test with Specs2

```
class FrontServletSpec extends ScalatraSpec {
  def is =
    "GET / on FrontServlet" ^
      "should return status 200" ! root200 ^
      end

  addServlet(classOf[FrontServlet], "/*")

  def root200 = get("/") {
    status must_== 200
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Thanks JRebel for free Scala plan !

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play www.playframework.com

- Stateless
- CoC
- Integrated Testing
- Asynchronous I/O
- Java friendly

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play Controller & View

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

Application.scala

```
package controllers
import play.api.mvc.{Controller, Action}

object Application extends Controller {
  def index() = Action {
    Ok(views.html.index("Hello codemotion!"))
  }
}
```

index.scala.html

```
@(message: String)

@main("Welcome to Play 2.1") {
    <h2>@message</h2>
}
```

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play Routing

Massimiliano Dessì & Alberto Quario
GTUG Sardegna

conf/routes

```
GET    /clients/all          controllers.Clients.list()
GET    /clients/:id          controllers.Clients.show(id: Long)
```

Bad request

For request 'GET /clients/hello' [Cannot parse parameter id as Long: For input string: "hello"]

```
GET    /clients/$id<[0-9]+>  controllers.Clients.show(id: Long)
```

Action not found

For request 'GET /clients/hello'

```
GET    /clients              controllers.Clients.list(page: Int ?= 1)
GET    /api/list-all         controllers.Api.list(Option[version])
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
def index = Action { implicit request =>
  session.get("connected").map { user =>
    Ok("Hello " + user)
  }.getOrElse {
    Unauthorized("Oops, you are not connected")
  }
}
```

```
Ok("Welcome!").withSession(
  "connected" -> "user@gmail.com"
)
```

```
Ok("Welcome!").withSession(
  session + ("saidHello" -> "yes") - "theme"
)
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play Authentication

Massimiliano Dessì & Alberto Quario
GTUG Sardegna

```
import play.api.mvc.Security
trait Secured {
  private def username(request: RequestHeader) =
request.session.get("user_id")

  private def onUnauthorized(request: RequestHeader) =
Results.Redirect(routes.Application.login)

  def isAuthenticated(f: => String => Request[AnyContent] =>
Result) = Security.Authenticated(username,onUnauthorized) {
    user => Action(request => f(user)(request))
  }
}
```

```
object Application extends Controller with Secured {
  def index = isAuthenticated { userId => implicit request =>
    Ok(html.index(userId))
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

- compiled as standard Scala functions
- follow a simple naming convention

views/index.scala.html ➡ views.html.index

```
@(customer: Customer, orders: List[Order])

@title(text: String) = @{
  text.split(' ').map(_.capitalize).mkString(" ")
}

<h1>@title(customer.name)</h1>
<ul>
@for(order <- orders) {
  <li>@order.getTitle()</li>
}
</ul>

@defining(customer.name + " " + customer.surname) { fullName =>
  <div>Regards @fullName</div>
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play REST

Massimiliano Dessì & Alberto Quario
GTUG Sardegna

```
GET     /api/reviews                controllers.Reviews.list
GET     /api/reviews/count          controllers.Reviews.count
GET     /api/reviews/:id            controllers.Reviews.show(id: Int)
```

```scala
import play.api.libs.json._
import play.api.libs.functional.syntax._

case class Review(id: Int, text: String)

object Review {
  val reviews: List[Review] =
    List(Review(1, "hello"), Review(2, "codemotion"))

  def findAll: Seq[Review] = reviews

  def count: Int = reviews.size

  def findById(id: Int): Option[Review] =
    reviews.find(_.id == id)

  implicit val reviewReads = (
    (__ \ "id").read[Int] and
    (__ \ "text").read[String]
  )(Review.apply _)

  implicit val creatureWrites = (
    (__ \ "id").write[Int] and
    (__ \ "text").write[String]
  )(unlift(Review.unapply))
}
```

```scala
object Reviews extends Controller {

  def list = Action { request =>
    Ok(toJson(Review.findAll))
  }

  def count = Action { request =>
    Ok(toJson(Review.count))
  }

  def show(id: Int) = Action {
    Review.findById(id).map { review =>
      Ok(toJson(review))
    }.getOrElse(NotFound(
      toJson(Map("error" ->
        "Review with id %s not found".
        format(id)))))
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play Form

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
object Contacts extends Controller {
  val contactForm: Form[Contact] = Form(
    mapping(
      "firstname" -> nonEmptyText,
      "lastname" -> nonEmptyText,
      "company" -> optional(text)
    )(Contact.apply)(Contact.unapply)
  )

  def form = Action {
    Ok(html.contact.form(contactForm))
  }

  def submit = Action { implicit request =>
    contactForm.bindFromRequest.fold(
      errors => BadRequest(html.contact.form(errors)),
      contact => Ok(html.contact.summary(contact))
    )
  }
```

```
case class Contact(
  firstname: String,
  lastname: String,
  company: Option[String]
)
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
@(contactForm: Form[Contact])
    @helper.form(action = routes.Contacts.submit) {

        <fieldset>
            <legend>General informations</legend>

            @inputText(
                contactForm("firstname"),
                '_label -> "First name"
            )

            @inputText(
                contactForm("lastname"),
                '_label -> "Last name"
            )

            @inputText(
                contactForm("company"),
                '_label -> "Company"
            )

        </fieldset>
        <div class="actions">
            <input type="submit" class="btn primary" value="Insert">
            <a href="@routes.Application.index" class="btn">Cancel</a>
        </div>

    }
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
object Application extends Controller {

  def intensiveComputation(): Int = ...

  def index = Action {
    val futureInt = scala.concurrent.Future { intensiveComputation() }
    Async {
      futureInt.map(i => Ok(views.html.index("Got result: " + i)))
    }
  }
}
```

```
play.api.libs.ws.WS

WS.url("http://example.com/feed").get()
WS.url("http://example.com/item").post("content")
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play Non-blocking IO /2

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
def race() = Action {
  Async {
    val start = System.currentTimeMillis()
    def getLatency(r: Any): Long = System.currentTimeMillis() - start

    val googleTime = WS.url("http://www.google.com").get().map(getLatency)
    val yahooTime = WS.url("http://www.yahoo.com").get().map(getLatency)
    val bingTime = WS.url("http://www.bing.com").get().map(getLatency)

    Future.sequence(Seq(googleTime, yahooTime, bingTime)).map {
      case times =>
        Ok(Json.toJson(Map(
          "google" -> times(0),
          "yahoo" -> times(1),
          "bing" -> times(2),
          "total" -> getLatency(0))))
    }
  }
}
```

```
{"google":343,"yahoo":3043,"bing":1608,"total":3048}
```

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
class ApplicationSpec extends Specification {

  "Application" should {

    "send 404 on a bad request" in {
      running(FakeApplication()) {
        route(FakeRequest(GET, "/boum")) must beNone
      }
    }

    "render the index page" in {
      running(FakeApplication()) {
        val home = route(FakeRequest(GET, "/")).get

        status(home) must equalTo(OK)
        contentType(home) must beSome.which(_ == "text/html")
        contentAsString(home) must contain ("Your new application is ready.")
      }
    }
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
class IntegrationSpec extends Specification {

  "Application" should {

    "work in a server" in {
      running(TestServer(3333)) {
        await(WS.url("http://localhost:3333").get).status must equalTo(OK)
      }
    }

    "work from within a browser" in {
      running(TestServer(3333), HTMLUNIT) { browser =>

        browser.goTo("http://localhost:3333/")

        browser.pageSource must contain("Your new application is ready.")
      }
    }
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

play Deploy /2

Massimiliano Dessì & Alberto Quario

GTUG Sardegna

```
Welcome to Play 2.1.0!

These commands are available:
-----------------------------
classpath                      Display the project classpath.
clean                          Clean all generated files.
compile                        Compile the current application.
console                        Launch the interactive Scala console (use :quit to exit).
dependencies                   Display the dependencies summary.
dist                           Construct standalone application package.
exit                           Exit the console.
h2-browser                     Launch the H2 Web browser.
license                        Display licensing informations.
package                        Package your application as a JAR.
play-version                   Display the Play version.
publish                        Publish your application in a remote repository.
publish-local                  Publish your application in the local repository.
reload                         Reload the current application build file.
run <port>                     Run the current application in DEV mode.
test                           Run Junit tests and/or Specs from the command line
eclipse                        generate eclipse project file
idea                           generate Intellij IDEA project file
sh <command to run>            execute a shell command
start <port>                   Start the current application in another JVM in PROD mode.
update                         Update application dependencies.
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

spray is an open-source toolkit for REST/HTTP

and

low-level network IO on top of Scala and Akka.

aka Scala -IKEA

VMWare and Ebay use spray

for some internal projects

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
class PingServiceActor extends Actor {
   def receive = {
        case HttpRequest(GET, "/ping", _, _, _) =>
        sender ! HttpResponse(200, "PONG")
    }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

path with name order => directive route
get and put => are inner route
~ =>route concatenation

```
class MyServiceActor extends Actor with Routing {
 def receive = receiveFromRoute {
   path("order" / HexIntNumber) { id =>
       get {
           completeWith {
               "Received GET for order " + id
           }
       } ~
       put {
           completeWith { "Received PUT for order " + id

           }
       }
   }
 }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
trait LongerService extends HttpService with MyApp {

  val simpleCache = routeCache(maxCapacity = 1000, timeToIdle = Duration("30 min"))

    val route = {
    path("orders") {
      authenticate(BasicAuth(realm = "admin area")) { user =>
        get {
          cache(simpleCache) {
            encodeResponse(Deflate) {
              complete {
                getOrdersFromDB
              }
            }
          }
        } ~
        post {
          (decodeRequest(Gzip) | decodeRequest(NoEncoding)) {
            entity(as[Order]) { order =>
              detachTo(singleRequestServiceActor) {
                complete {
                  // ... write order to DB
                  "Order received"
                }
              }
            }
          }
        }
      }
    } ~
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
pathPrefix("order" / IntNumber) { orderId =>
  path("") {
    // method tunneling via query param
    (put | parameter('method ! "put")) {
      // form extraction from multipart or www-url-encoded forms
      formFields('email, 'total.as[Money]).as(Order) { order =>
        complete {
          // complete with serialized Future result
          (myDbActor ? Update(order)).mapTo[TransactionResult]
        }
      }
    } ~
    get {
      jsonpWithParameter("callback") {
        produce(instanceOf[Order]) { complete => ctx =>
          processOrderRequest(orderId, complete)
        }
      }
    }
  } ~
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

spray Non-blocking IO

Massimiliano Dessì & Alberto Quario
GTUG Sardegna

```
def race() {
  val start = System.currentTimeMillis()
  def getLatency(r: Any): Long = System.currentTimeMillis() - start

  val googleFuture = httpClient.ask(Get("http://www.google.com"))
    .mapTo[HttpResponse].map(getLatency)
  val yahooFuture = httpClient.ask(Get("http://www.yahoo.com"))
    .mapTo[HttpResponse].map(getLatency)
  val bingFuture = httpClient.ask(Get("http://www.example.com"))
    .mapTo[HttpResponse].map(getLatency)

  Future.sequence(Seq(googleFuture, yahooFuture, bingFuture)).map {
    case times =>
      log.info(
        "google" + times(0) +
        " yahoo" + times(1) +
        " bing" + times(2)  +
        " total: "+getLatency(0))
      system.shutdown() // stops all actors
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

```
class DemoServiceSpec extends Specification with Specs2RouteTest with DemoService {
  def actorRefFactory = system

  "The DemoService" should {

    "return a greeting for GET requests to the root path" in {
      Get() ~> demoRoute ~> check { entityAs[String] must contain("Say hello") }
    }

    "return a 'PONG!' response for GET requests to /ping" in {
      Get("/ping") ~> demoRoute ~> check { entityAs[String] === "PONG!" }
    }

    "leave GET requests to other paths unhandled" in {
      Get("/kermit") ~> demoRoute ~> check { handled must beFalse }
    }

    "return a MethodNotAllowed error for PUT requests to the root path" in {
      Put() ~> sealRoute(demoRoute) ~> check {
        status === MethodNotAllowed
        entityAs[String] === "HTTP method not allowed, supported methods: GET"
      }
    }
  }
}
```


Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Massimiliano Dessì & Alberto Quario

Q & A

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/

Thanks for your attention!

Massimiliano: @desmax74

Alberto :@realrealbot

{codemotion}

Except where otherwise noted, this work is licensed under: http://creativecommons.org/licenses/by-nc-sa/3.0/