

Play Framework 2

Play Goals

- DX - Developer Experience
 - Fast turnaround
- Scalability
- (Really) Support HTTP

Play Background

- Web application framework
- Open source, Apache 2 license
- Guillaume Bort - creator and lead developer
- Initial release: 2008-02, 1.0: 2009-10
- <http://www.playframework.org>

Play 2

- Play 2.0 released 2012-03
- Re-written in Scala
- Templates in Scala instead of Groovy
- “Everything is compiled” - Including routes
- SBT as build tool (instead of Ivy)

Play

- `play new myapp`
- `play ~run myapp`
- **app**
 - controllers
 - models
 - views

Exercise I

- Create a new Play-app
- Make some change that can be observed at <http://localhost:9000>

Routing

- `conf/routes`

```
GET    /                controllers.Application.index(name : String ?= "Unknown")
GET    /guest           controllers.Application.index(name : String = "My Guest")
GET    /$id<[0-9]+>    controllers.Application.indexId(id : Long)
GET    /:name           controllers.Application.index(name : String)
```

<http://www.playframework.org/documentation/2.0.1/ScalaRouting>

Test

- Specs2
- Fluentlenium

<http://www.playframework.org/documentation/2.0.1/ScalaTest>

Exercise 2

- Create a lookup service where persons can be found using personnummer *or* last name
- Fix a Specs2 test and a Browser test
- Hints:
 - `val idMap = Map[String, String]("Strandberg" -> "Mats Strandberg")`
 - `Ok("This text is sent as text/plain")`

Templates

- Scala Code - a function
- Type Checked
- Templates can call Templates

[http://www.playframework.org/documentation/2.0.1/
ScalaTemplates](http://www.playframework.org/documentation/2.0.1/ScalaTemplates)

Forms

- POST
 - application/x-www-form-urlencoded
 - JSON
- Form Helpers
 - Play Standard
 - Twitter Bootstrap

Exercise 3 - Create users

- A user has first name, last name and email
- email is optional but must be valid when entered
- Create a simple form to fill in data
 - GET `/users/new`
- Create a simple action to receive data
 - POST `/users`

Exercise 3 - Create users

- A user is “saved” by e.g. a debug printout or by `Ok("This text is sent as text/plain")`
- Use `play.api.data.Form`

Stateless

- No State kept by web server!
- Play “Sessions” (cookies)
 - Session scope
 - Flash scope
- Cache
- Databases

Cache

```
Cache.set("item.key", connectedUser)
```

```
val maybeUser: Option[User] = Cache.getAs[User]("item.key")
```

```
val user: User = Cache.getOrElseAs[User]("item.key") {  
    User.findById(connectedUser)  
}
```

Sessions scope = signed cookies

```
Ok("Hello World!").withSession(  
  "connected" -> "user@gmail.com"  
)  
  
def index = Action { request =>  
  request.session.get("connected").map { user =>  
    Ok("Hello " + user)  
  }.getOrElse {  
    Unauthorized("Oops, you are not connected")  
  }  
}
```


Flash Scope

- Used to transport simple success messages for non-Ajax connections
- Cookies (not signed)
- Data kept for only on request

Databases

- Ebean
- Anorm
- Evolution

Modules

- <https://github.com/playframework/Play20/wiki/Modules>

Twitter Bootstrap

Websockets

Interceptors

- Login
- Google login
- OAuth

Heroku

- Create a local git repo

```
git init
git add .
git commit -m init
```

Heroku

- Sign up for Heroku
- Install Heroku Toolbelt
- Push to Heroku

```
heroku create -s cedar myapp  
git push heroku master
```


Distribution

- play dist

Coffeescript

Less

Akka

Extra

- Generera router till JavaScript från Template