scalac

Lift - wprowadzenie

... Lift just rocks !

Lift - historia

Start w 2007
60 commiterów
Twórca to David Pollack
Inspirowane przez Ruby on Rails,
 Django, Wicket, Seaside

Lift - funkcjonalność

- Lazy loading
- Parallel page rendering
- Comet and Ajax
- Wiring
- Designer friendly templates
- Wizard
- Security

Lift – quick start

Łatwizna!

wget https://github.com/lift/lift_25_sbt/tarball/master
tar -xvzf lift.tar.gz
cd lift-YYY/scalaX/lift_basic
./sbt ~container:start

localhost:8080

Lift - Boot

Startup scripts
LiftRules
Run modes and Properties
SiteMap

*Boot.scala

```scala
class Boot {
  def boot {
    LiftRules.addToPackages("code")

    // Build SiteMap
    val entries = List(Menu.i("Home") / "index")

    //Show the spinny image when an Ajax call starts
    LiftRules.ajaxStart =
      Full(() => LiftRules.jsArtifacts.show("ajax-loader").cmd)

    //our custom rule
    LiftRules.dispatch.append(UserApi.dispatch)

    //connect to MongoDB
    val srvr = new ServerAddress("127.0.0.1", 27017)
    val mo = new MongoOptions
    mo.socketTimeout = 10
    MongoDB.defineDb(DefaultMongoIdentifier, new Mongo(srvr, mo), "lift-test")
  }
}
```

Lift - Rest

LiftRules – dispatch vs statelessDispatchTable
Bloki serve
lift-json

*UserApi.scala

```scala
object UserApi extends Logger {
  def dispatch: LiftRules.DispatchPF = {
    case r @ Req("api" :: "user" :: id :: Nil, _, GetRequest) =>
      debug("Requested user " + id)
      () => {
        User.find(id) match {
          case Full(user) => Full(JsonResponse(user.asJValue))
          case _ => Full(BadResponse())
        }
      }
  }
}
```

Lift – Async Rest

Comet – long polling
Prosty sposób na obsługę długotrwałych zadań
Kontynuacje
Nie blokuje wątku
RestContinuation.async (reply => ...

Lift – Async Rest

UserApi.scala

```
30      case r @ Req("api" :: "user" :: "find" :: Nil, _, PostRequest) =>
31        RestContinuation.async {
32          satisfyRequest => {
33            // schedule a "Null" return if there's no other answer
34            // after 110 seconds
35            Schedule.schedule(() => satisfyRequest(BadResponse()), 110 seconds)
36
37            // register for an "onChange" event.  When it
38            // fires, return the changed item as a response
39            Item.onChange(item => satisfyRequest(item: JValue))
40          }
41        }
```

Demo

Q & A


scalac