Frontend Interface

Methoden zur Minimierung von blockenden Aufrufen im Routing

TECHNISCHE UNIVERSITÄT DARMSTADT

Matthias Jahn, Frederick Schäfer

Scala Futures

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstrukt zur nebenläufigen Ausführung von Code
- abhängige Berechnungen einfach Erstellbar

Beispiel:

```
val piFuture = Future { calcPI() }
val resultFuture = piFuture map { pi => 2 * pi * r }
val result = Await.result(resultFuture, 5 seconds)
```

For Comprehension

TECHNISCHE UNIVERSITÄT DARMSTADT

Verschachtelung zweier Futures mit flatMap:

```
val futureA = Future { calcA() }
val futureB = Future { calcB() }
val result = futureA flatMap {
  a => futureB map { b => a * b }
}
```

Verschachtelung mit For:

```
val futureA = Future { calcA() }
val futureB = Future { calcB() }
for(
  a <- futureA
  b <- futureB
) yield a * b
```

nebenläufig

```
for(
  a <- Future { calcA() }
  b <- Future { calcB() }
) yield a * b
```

sequenziell

Futures und Akka

TECHNISCHE UNIVERSITÄT DARMSTADT

- Akka Ask („?“) Methode liefert ein Future
- Weiterleitung eines Ergebnisses mit *pipeTo* möglich
- Kapselung des internen Aktorzustandes beachten
- **Achtung**: "sender" ist ein *def*

Beispiel:

```
val from = sender
val result = (actorA ? GetInfo).mapTo[Info]
result.map(info => InfoRequestResult(info, from) pipeTo actorB
```

Futures und Spray-Routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- direkte Unterstützung für Futures in complete()
- seit Version 1.2-M8 zusätzlich mit onSuccess, onFailure und onComplete Direktiven

Vorher:

```
val future = (subjectProviderManager ? GetAllProcessInstances(userId))
  .mapTo[AllProcessInstancesAnswer]
val result = Await.result(future, timeout.duration)
complete(result.processInstanceInfo)
```

Nachher:

```
val future = (subjectProviderManager ? GetAllProcessInstances(userId))
  .mapTo[AllProcessInstancesAnswer]
complete(future.map(result => result.processInstanceInfo))
```

Beispiel für onSuccess

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorher:

```
val userFuture = (persistenceActor ? Users.Read.ByIdWithIdentities(id)).mapTo[…]
val user = Await.result(userFuture, timeout.duration)
if (user.isDefined) {
  complete(UserWithMail(…))
} else {
  complete(StatusCodes.NotFound)
}
```

Nachher:

```
val userFuture = (persistenceActor ? Users.Read.ByIdWithIdentities(id)).mapTo[…]
onSuccess(userFuture) {
  user =>
    if (user.isDefined) {
      complete(UserWithMail(…))
    } else {
      complete(StatusCodes.NotFound)
    }
}
```

Anweisungen innerhalb des Routingpfades

TECHNISCHE UNIVERSITÄT DARMSTADT

- Anweisungen können bei der Erstellung der Route oder pro Anfrage ausgeführt werden

```
val route: Route = get {
  val time = currentTime()
  complete("Current time: " + time)
}
```

bedeutet in etwa:

```
val route: Route = {
  val inner = {
    val time = currentTime()
    { ctx => ctx.complete("Current time: " + time) }
  }
  get.apply(inner)
}
```

Wird bei der Erstellung der Route ausgeführt

http://spray.io/documentation/1.2-M8/spray-routing/advanced-topics/understanding-dsl-structure/

Aufrufe innerhalb des Routingpfades

TECHNISCHE UNIVERSITÄT DARMSTADT

Lösungsmöglichkeit 1: alle Anweisungen innerhalb von *complete*

```
get {
  complete {
    val time = currentTime()
    "Current time: " + time
  }
}
```

Lösungsmöglichkeit 2: Nutzung der *dynamic* Direktive

```
get {
  dynamic {
    val time = currentTime()
    complete("Current time: " + time)
  }
}
```