

Aufgaben zum Backend Tutorial



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Aufgabe 1 Akka lernen

Im ersten Schritt verweisen wir auf ein externes Beispiel. Folge diesem Link:

<http://doc.akka.io/docs/akka/2.0/intro/getting-started-first-scala.html>

und bearbeite das Tutorial. Im Git liegt unter `exercises/AkkaTutorial` ein leeres Scala-Projekt mit einem `sbt` script, deshalb kann der Teil des Tutorials, der sich mit dem Einrichten des Projekts beschäftigt, übersprungen werden. (Führt das Skript aus und importiert es von der Stelle in Eclipse, an der es liegt. Eure Lösungen sollt ihr nicht hochladen)

Hinweis:

Im Tutorial werden die imports:

```
import akka.util.Duration
```

```
import akka.util.duration._
```

genutzt. Dies funktioniert mit der aktuellen akka Version nicht mehr, nutzt stattdessen den import:

```
import scala.concurrent.duration._
```

Aufgabe 2 Backend Tutorial

Das Backend Tutorial soll dazu dienen, die Konzepte des Backends an Hand einer vereinfachten Version besser zu verstehen.

Aufgabe 2.1 Einrichten

Im Git-Ordner `exercises/BackendTutorial` liegt eine rudimentäre Implementierung des Backends, die den Einstieg in den Backendkernel vereinfachen soll.

Zur Bearbeitung der Aufgaben empfehlen wir den HTTPRequester für Firefox

<https://addons.mozilla.org/de/firefox/addon/httprequester/>

oder Advanced REST Client für Chrome

<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjel>

Die Struktur und eine Erklärung der einzelnen Komponenten ist unter diesem Link verfügbar:

<https://sbpm-groupware.atlassian.net/wiki/display/SBPM/SBPM+-+Tutorial>

Kopiere den Ordner `BackendTutorial` nach `exercises/tutorial/<lastname>/` und führe anschließend das `sbt` Skript aus. Importiere das `BackendTutorial` als Projekt in die Entwicklungsumgebung und starte die Ausführung über die `Boot.scala`.

Versuche über `GET http://localhost:8080/subject/1` den Status des ersten Subjekts auszulesen (es gibt 2).

Aufgabe 2.2 Fehler beheben

Ein `ActState` ist eine Verzweigung, die auf mehrere mögliche Folgestates zeigt. Momentan wird jedoch bei einer `ExecuteAction`-Nachricht nur die erste Transition ausgeführt. Erweitere die `ExecuteAction`-Nachricht und den `ActStateActor` so, dass es möglich ist, sich zwischen verschiedenen Verzweigungen zu entscheiden.

Aufgabe 2.3 REST Schnittstelle erweitern

Bisher wurde immer das erste TestPair aus den Testdaten genommen, um die beiden Subjekte zu instantiieren. Nun soll die Möglichkeit geschaffen werden auch das zweite Testpair zu instantiieren, hierfür existiert bereits eine Funktion im ProcessInstanceActor, um die beiden Subjekte auszutauschen.

Als Aufruf soll der Befehl: **PUT** `http://localhost:8080/subject`

mit dem Argument: `{ "instance": n }` dienen,

wobei `n` ein `Int` ist und das Testbeispiel auswählen soll.

Hierfür muss die REST-Schnittstelle angepasst und eine neue Nachricht eingeführt werden.

Aufgabe 2.4 Ausführung erweitern

In dieser Aufgabe soll eine einfache Kommunikation zwischen zwei Subjekten realisiert werden. Die Kommunikation besteht aus `SendStates`, die eine Nachricht an das andere Subjekt schicken und auf eine Bestätigung¹ (Ack) warten um den State zu wechseln, während `ReceiveStates`² auf Nachrichten warten, den Erhalt bestätigen und nach dem dann in der Lage sind in den nächsten State zu wechseln. Hierzu soll das zweite Testpair verwendet werden, das im Gegensatz zum 1. auch aus `Receive-` und `Sender-States` besteht.

¹ implementiert als **case object** Ack

² Der `ReceiveState` ist bereits implementiert