Exercises for the Backend Tutorial

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Problem 1 Backend Tutorial

The Backend Tutorial provides a simplified version of the actual Backend and should help in understanding the concepts.

Problem 1.1 Setup

The simplified implementation is in the git repository in the folder exercises/BackendTutorial.

We suggest using the tools HTTPRequester for Firefox
https://addons.mozilla.org/de/firefox/addon/httprequester/
or Advanced REST Client for Chrome
https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjel
to solve the exercises.

The structure and an explanation of each component is provided in our wiki:
https://sbpm-groupware.atlassian.net/wiki/display/SBPM/SBPM+-+Tutorial

Copy the folder BackendTutorial to exercises/tutorial/<lastname>/ and execute the sbt eclipse script. Import the project into Eclipse and run Boot.scala.

Try to fetch the status of the first subject (there are two subjects) via
GET http://localhost:8080/subject/1

Problem 1.2 Bugfixing

An ActState is a branching state pointing to multiple following states. However, the current implementation just uses the first transition when an ExecuteAction message arrives. Extend the ExecuteAction message and the ActStateActor so that it is possible to choose between multiple branches.

Notice: the ActStateActor is located in the package ”de.tkip.sbpm.application.state”. You can find an description of the structure when you follow the link provided in Problem 1.1.

Problem 1.3 Extend the REST interface

Currently the two subjects are instantiated from the first TestPair. Now you have to implement a solution to instantiate the second TestPair. There is already a function in the ProcessInstanceActor to change the TestPair.

It should be called by **PUT** http://localhost:8080/subject with the content `{"instance": n}` where the instance parameter is an Int and refers to the TestPair that should be loaded.

Extend the REST interface and create a new message.

Problem 1.4 Extend the execution

This task is about implementing a simple communication between two subjects. The communication consists of SendStates that send a message to the other subject and wait for an acknowledgement[1] (Ack)

[1] implemented as **case object** Ack

to change the state. The ReceiveStates[2] wait for incoming messages and acknowledge them, it then can change to the next state.

In this task you should use the second TestPair, that (in contrast to the first one) contains also Send- and ReceiveStates.

[2] The ReceiveState is already implemented