AARHUS UNIVERSITET

Software Architecture in Practice

Connector Exercise

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Exercise 1

- Get hold of the Zip from the week plan
- Unzip, and ‘gradle tasks’ to see the various demo tasks

```
csdev@m1: ~/proj/evuproject/connectors
csdev@m1: ~/proj/evuproject/connectors 113x18
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Connectors tasks
----------------
mqmonitor - Run the temperature monitor using RabbitMQ
mqserver - Run the temperature sensor using RabbitMQ
restmonitor - Run the temperature monitor using ReST
restserver - Run the temperature sensor using ReST
tuplemonitor - Run the temperature monitor using tuplespace/redis
tupleserver - Run the temperature sensor using tuplespace/redis

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project 'connectors'.
```

- Open the ReadMe.md to get ‘intermediary’ information

```
Open    ReadMe.md
        ~/proj/evuproject/connectors    Save

Messaging
---

Messaging requires an intermediate, and I have choosen RabbitMQ, as
their tutorials are really great, and it is a widely used Messagequeue
(MQ) system.

To run a RabbitMQ server, start it as docker container

    docker run -d -p 5672:5672 -p 15672:15672 --name mq rabbitmq:3-management

(Docker is a container based virtual machine, we will return to it
later in the course).

To see what is going on, open the RabbitMQ dashboard using your
```

Or use the ‘start*.sh’ scripts

```
settings.gradle
startmq.sh*
startredis.sh*
tuplespace/
```

Exercise 2 / Tuplespace

- Run both server and monitor at the same time
  - Note the semantics
    - *is every measured temperature displayed?*
  - Start multiple monitors – what happens?
  - Keep them running, and shut down the server. Any issues?
    - Is this a desirable quality of a chemical plant system?
    - Tactics to mitigate?

Exercise 3 / MQ

- Run both server and monitor at the same time
  - Note the semantics
    - *is every measured temperature displayed?*
  - Start one more monitor – what happens?
    - Then try to shut down the first monitor – what happens? Explain!
  - Shut down the server. What happens?
  - Restart it again. What happens?
  - Play around with the RabbitMQ Dashboard
    - Inspect queues, exchanges, …

Exercise 4 / REST

- Run both server and monitor at the same time
  - Note the semantics
    - *is every measured temperature displayed?*
  - Start one more monitor – what happens?
    - Then try to shut down the first monitor – what happens? Explain!
  - Shut down the server. What happens?
  - Restart it again. What happens?

AARHUS UNIVERSITET

Exercise 5 / Characteristics

- Compare the connector characteristics (same/different time; control/data orientation; event/state oriented) from the W6-1 slides with the concrete observations made from running the three demos.
  - Which concrete behavior underlines the characteristics
    - Ala “Evidence of the ‘event orientation’ is seen in the MQ demo because I observe that when I do X then the connector exhibits Y behavior”

Exercise 6 / MQ

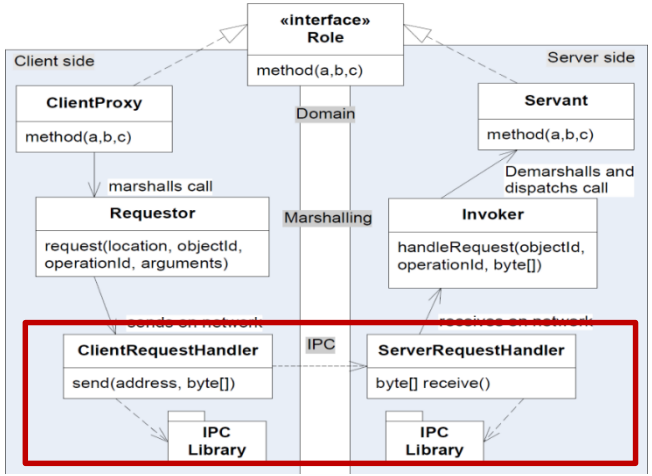
- Rewrite the server so
  - any temperature above 106 Celcius is *also* transmitted with topic 'plant.alert'
- Make an 'alertmonitor' which only displays these temperatures.
  - (Hint: Search for RabbitMQ's 'Topic' tutorial in Java, on WWW).

Exercise 7 / Tuplespace

- Augment the tuple space server, so it implements ‘heartbeat’ in the form of timestamps on each measurement
  - And let the monitor flag ERROR in case T is more than 10 seconds old
- Note:
  - A somewhat bigger exercise, as you need to marshall into Json or some other string representation of (T,date).
  - OffsetDateTime odt = OffsetDateTime.now()
    - Will give current date/time in ISO 8601 format

Exercise 8 / Broker MQ

- The FRDS.Broker library allows any network connector to implement the IPC layer



- Exercise:
  - Write RabbitMq IPC implementations of CRH and SRH, using the RabbitMQ 'RPC tutorial'.
  - Make TeleMed use that instead instead!

- Morale
  - ***This is a 1-2 hour implementation effort, and TeleMed is then refactored into a highly resilient and scalable system !!!***