Uebung4

Surendiran Sithamparam
Regina Traber

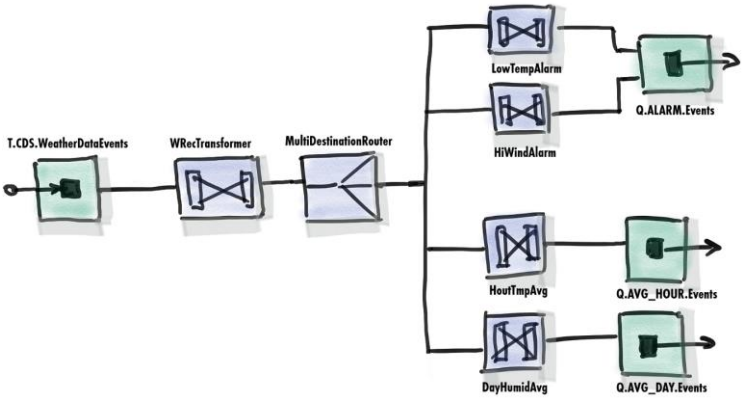
4.4.1.5

Aktivieren Sie nun in ch.ost.mas.cds.integration.processor.IntegratorMain.setupFlows(String[]) die auskommentierte Zeile für einen weiteren Prozess. Überlegen Sie sich zuerst, was die Änderung bewirkt und testen Sie dann damit das Programm erneut (Prozess muss gestoppt und neu gestartet werden). Schreiben Sie ihre Beobachtung auf

Beobachtung: Es wird ein neuer Flow hinzugefügt. Die konvertierten (bearbeiteten) Meldungen von **mConvertedMsgRecordsOut** werden über den den **mConvertedMsgRecordsIn** an den **mConsoleLogger** übergeben der die Meldungen auf der Konsole ausgibt. Die Inbound und Outbound Endpoints werden beider bei der Instanzierung mit derselben queue verbunden.

Alter Ablauf	Neuer Ablauf
• **mInp:** Daten von der Wetterstation werden gehölt • **mWRecTransformer:** Transformieren der Wetterdaten in einen String • **mConvertedMsgRecordsOut**: Schreibt Daten/Meldungen in die message queue	• **mInp:** Daten von der Wetterstation werden gehölt • **mWRecTransformer:** tTansformieren der Wetterdaten in einen String • **mConvertedMsgRecordsOut**: Schreibt Daten/Meldungen in die message queue • **mConvertedMsgRecordsIn:** Holt Daten/Meldungen aus der message queue • **mConsoleLogger:** Daten/Meldungen werden via PrintStream auf der Konsole ausgegeben

4.4.1.7 & 4.4.1.9



Definieren Sie in der IntegratorMain.setupFlows()-Methode den Flow. → siehe Lösung

Bauen sie nach den Alarm-Komponenten die ConsoleLogger-Komponente ein, damit Sie auf der Console sehen, wann eine Alarm-Meldung kommt. Zu Debugging-Zwecken kann das auch schon früher im Flow z.B. vor dem MultiDestinationRouter eingebaut werden → siehe Lösung

```
public class IntegratorMain {

 […]

 private MultiDestinationRouter mMultiDestinationRouter;
 private MsgInboundEndpoint mAlarmIn;

 […]

 /**
  * Setup needed endpoints according to the passed arguments or the defaults otherwise
  *
  * @param pArgs String[] command line arguments to parse
  */
 protected void setupEndpoints(String[] pArgs) {
  String input = mJMSUtil.findArg("-inp", true);
  String avghourout = mJMSUtil.findArg("-avghourout", true);
```

```
  String avgdayout = mJMSUtil.findArg("-avgdayout", true);
  String alarm = mJMSUtil.findArg("-alarmout", true);
  String studmailaddress = mJMSUtil.findArg("-studemail", true);
  mMessageFactory = MessageFactory.getInstance();
  mMessageFactory.setStudentEMail(studmailaddress);
  mInp = new MsgInboundEndpoint(input != null ? input : INP_WRECS, MSGDEST.TOPIC, MSGTYPE.WRECRAW);
  mConvertedMsgRecordsIn = new MsgInboundEndpoint(
    String.join("_", CONV_WRECS, System.getProperty("user.name")), MSGDEST.QUEUE, MSGTYPE.WRECNORM);
  mAvgHourOut = new MsgOutboundEndpoint(
    avghourout != null ? avghourout
      : String.join("_", AVG_HOUR_DEST, System.getProperty("user.name")),
    MSGDEST.QUEUE, MSGTYPE.WAVGMSG);
  mAvgDayOut = new MsgOutboundEndpoint(
    avgdayout != null ? avgdayout : String.join("_", AVG_DAY_DEST, System.getProperty("user.name")),
    MSGDEST.QUEUE, MSGTYPE.WAVGMSG);
  mAlarmOut = new MsgOutboundEndpoint(
    alarm != null ? alarm : String.join("_", ALARM_DEST, System.getProperty("user.name")),
    MSGDEST.QUEUE, MSGTYPE.WALARM);
  mConvertedMsgRecordsOut = new MsgOutboundEndpoint(
    String.join("_", CONV_WRECS, System.getProperty("user.name")), MSGDEST.QUEUE, MSGTYPE.WRECNORM);
  mConsoleLogger = new ConsoleLogger();
  // Alarm inbound endpoint
  mAlarmIn = new MsgInboundEndpoint(
    alarm != null ? alarm : String.join("_", ALARM_DEST, System.getProperty("user.name")),
    MSGDEST.QUEUE, MSGTYPE.WALARM);
 }

 […]

 /**
  * Setup the process(es) to be run and add all processes to the process list
  *
  * @param pArgs
  */
 protected void setupFlows(String[] pArgs) {
  // Multi-destinaton rerouter
  mFlows.add(new ProcessSequence(mInp, mWRecTransformer, mConvertedMsgRecordsOut));
  mMultiDestinationRouter = new MultiDestinationRouter(
    new ProcessSequence(mLowTempFilter, mAlarmOut), new ProcessSequence(mHighWindFilter, mAlarmOut),
    new ProcessSequence(mHourTempAvgColl, mAvgHourOut),
    new ProcessSequence(mDayHumidAvgColl, mAvgDayOut));
  mFlows.add(new ProcessSequence(mConvertedMsgRecordsIn, mMultiDestinationRouter));
  // Alarm to consol logger
  mFlows.add(new ProcessSequence(mAlarmIn, mConsoleLogger));
 }

 […]

}
```

4.4.1.8

Testen Sie den Flow und schauen Sie, ob Ihre Durchschnitts- und Alarm-Werte kommen. Sie können diese mit der ActiveMQ-Admin-Konsole anschauen.

→ in ActiveMQ geprüft

4.4.1.10

Überlegen Sie sich wie man den Prozess definieren würde, wenn die Daten anstatt vom WeatherDataFeeder ins Topic geschrieben würden, vom REST-Service der Wetterstation direkt abgeholt würden? Wie sähe der Prozess aus und was müsste in der vorliegenden Implementation noch zusätzlich implementiert werden

1) Daten abholen: → Neu: vom RESTful Service abholen

Folgende Ansätze ermöglichen es kontinuierlich mit dem RESTful-Service zu interagieren und die neuesten Daten abzurufen. Die Wahl des passenden Ansatzes hängt von der Funktionalität der verwendeten REST-API und die weiteren Anforderungen des Prozesses ab.

- Polling: Ihr Prozess fragt regelmäßig den RESTful-Service nach neuen Daten, indem er wiederholt Anfragen sendet, um zu überprüfen, ob neue Informationen verfügbar sind.
- Webhooks: Der RESTful-Service benachrichtigt aktiv Ihren Prozess, wenn neue Daten vorhanden sind, indem er einen speziellen Endpunkt aufruft, den Sie eingerichtet haben. Dadurch werden die neuen Daten an Ihren Prozess übermittelt.
- Streaming: Ihr Prozess stellt eine Verbindung zum Streaming-Endpunkt her und empfängt den Datenstrom in Echtzeit, sodass Sie immer die neuesten verfügbaren Daten erhalten.

2) Konvertieren der Daten → Neu: Datenformat muss angepasst werden, da neu mit neuer REST API gearbeitet wird; mWRecTransformer muss angepasst werden

3) Alarm und Fachliche Aufbereitung der Daten → kann gleichbleiben

4) Meldungen an Endpunkte senden → kann gleichbleiben

4.5

Zum Schluss lassen sie ihr Programm für 5 Minuten mit folgenden Parametern laufen (neu starten):
-host mq-mas-se.cloud.oeint.ch -studemail <ihre e-mail addresse>@ost.ch -avghourout Q.Abgabe.AVG_HOUR.Events

Feedinterval: 1000ms → gemessen: 24 Q.Abgabe.AVG_HOUR.Events

Browse Q.Abgabe.AVG_HOUR.Ev...

Message ID ↑	Correlation ID	Persistence	Priority	Redelivered	Reply To	Timestamp	Type	Operations
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:1		Persistent	4	false		2023-05-28 18:06:13:793 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:10		Persistent	4	false		2023-05-28 18:08:04:890 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:11		Persistent	4	false		2023-05-28 18:08:17:258 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:12		Persistent	4	false		2023-05-28 18:08:29:591 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:13		Persistent	4	false		2023-05-28 18:08:41:979 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:14		Persistent	4	false		2023-05-28 18:08:54:337 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:15		Persistent	4	false		2023-05-28 18:09:06:668 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:16		Persistent	4	false		2023-05-28 18:09:19:059 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:17		Persistent	4	false		2023-05-28 18:09:31:401 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:18		Persistent	4	false		2023-05-28 18:09:43:767 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:19		Persistent	4	false		2023-05-28 18:09:56:110 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:2		Persistent	4	false		2023-05-28 18:06:26:131 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:20		Persistent	4	false		2023-05-28 18:10:08:657 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:21		Persistent	4	false		2023-05-28 18:10:20:849 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:22		Persistent	4	false		2023-05-28 18:10:33:220 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:23		Persistent	4	false		2023-05-28 18:10:45:572 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:24		Persistent	4	false		2023-05-28 18:10:57:969 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:3		Persistent	4	false		2023-05-28 18:06:38:424 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:4		Persistent	4	false		2023-05-28 18:06:50:736 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:5		Persistent	4	false		2023-05-28 18:07:03:136 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:6		Persistent	4	false		2023-05-28 18:07:15:545 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:7		Persistent	4	false		2023-05-28 18:07:27:869 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:8		Persistent	4	false		2023-05-28 18:07:40:263 UTC	WAVGMSG	Delete
ID:DESKTOP-58KOUCD-60935-1685297147650-1:6:1:1:9		Persistent	4	false		2023-05-28 18:07:52:553 UTC	WAVGMSG	Delete

```java
package ch.ost.mas.cds.integration.processor;

import java.util.ArrayList;
import java.util.List;

import javax.jms.Message;

import ch.ost.mas.cds.integration.base.AVGPERIOD;
import ch.ost.mas.cds.integration.base.IProcessor;
import ch.ost.mas.cds.integration.base.MSGDEST;
import ch.ost.mas.cds.integration.base.MSGTYPE;
import ch.ost.mas.cds.integration.base.WPARAM;
import ch.ost.mas.cds.integration.control.MultiDestinationRouter;
import ch.ost.mas.cds.integration.control.ProcessSequence;
import ch.ost.mas.cds.integration.endpoints.ConsoleLogger;
import ch.ost.mas.cds.integration.endpoints.MsgInboundEndpoint;
import ch.ost.mas.cds.integration.endpoints.MsgOutboundEndpoint;
import ch.ost.mas.cds.integration.translator.AlarmFilter;
import ch.ost.mas.cds.integration.translator.AverageCollector;
import ch.ost.mas.cds.integration.translator.WeatherRecordTransformer;
import ch.ost.mas.cds.integration.util.JMSUtil;
import ch.ost.mas.cds.integration.util.MessageFactory;

public class IntegratorMain {
 private static final String ALARM_DEST = "Q.ALARM.Events";
 private static final String AVG_HOUR_DEST = "Q.AVG_HOUR.Events";
 private static final String AVG_DAY_DEST = "Q.AVG_DAY.Events";
 private static final String CONV_WRECS = "Q.WREC_MAP.Events";
 private static final String INP_WRECS = "T.CDS.WeatherDataEvents";

 private JMSUtil mJMSUtil;
 private MessageFactory mMessageFactory;

 private MsgInboundEndpoint mInp;
 private MsgInboundEndpoint mConvertedMsgRecordsIn;
 private MsgOutboundEndpoint mAvgHourOut;
 private MsgOutboundEndpoint mAvgDayOut;
 private MsgOutboundEndpoint mAlarmOut;
 private MsgOutboundEndpoint mConvertedMsgRecordsOut;

 private AlarmFilter mLowTempFilter;
 private AlarmFilter mHighWindFilter;
 private AverageCollector mHourTempAvgColl;
 private AverageCollector mDayHumidAvgColl;
 private WeatherRecordTransformer mWRecTransformer;
 private ConsoleLogger mConsoleLogger;

 private List<ProcessSequence> mFlows;

 // new objects for exercise
 private MultiDestinationRouter mMultiDestinationRouter;
 private MsgInboundEndpoint mAlarmIn;

 private void setup(String[] pArgs) {
  try {
   JMSUtil.init(pArgs);
   mJMSUtil = JMSUtil.getInstance();
   mFlows = new ArrayList<ProcessSequence>();
   setupEndpoints(pArgs);
   setupTranslators(pArgs);
   setupFilters(pArgs);
   setupFlows(pArgs);
  } catch (Exception pEx) {
   System.err.printf("Error when setting up the system err=%s\n", pEx.getMessage());
  }
 }
```

```
/**
 * Setup needed endpoints according to the passed arguments or the defaults otherwise
 *
 * @param pArgs String[] command line arguments to parse
 */
protected void setupEndpoints(String[] pArgs) {
 String input = mJMSUtil.findArg("-inp", true);
 String avghourout = mJMSUtil.findArg("-avghourout", true);
 String avgdayout = mJMSUtil.findArg("-avgdayout", true);
 String alarm = mJMSUtil.findArg("-alarmout", true);
 String studmailaddress = mJMSUtil.findArg("-studemail", true);
 mMessageFactory = MessageFactory.getInstance();
 mMessageFactory.setStudentEMail(studmailaddress);

 mInp = new MsgInboundEndpoint(input != null ? input : INP_WRECS, MSGDEST.TOPIC, MSGTYPE.WRECRAW);

 mConvertedMsgRecordsOut = new MsgOutboundEndpoint(
   String.join("_", CONV_WRECS, System.getProperty("user.name")), MSGDEST.QUEUE, MSGTYPE.WRECNORM);
 mConvertedMsgRecordsIn = new MsgInboundEndpoint(
   String.join("_", CONV_WRECS, System.getProperty("user.name")), MSGDEST.QUEUE, MSGTYPE.WRECNORM);

 mAvgHourOut = new MsgOutboundEndpoint(
   avghourout != null ? avghourout
     : String.join("_", AVG_HOUR_DEST, System.getProperty("user.name")),
   MSGDEST.QUEUE, MSGTYPE.WAVGMSG);
 mAvgDayOut = new MsgOutboundEndpoint(
   avgdayout != null ? avgdayout : String.join("_", AVG_DAY_DEST, System.getProperty("user.name")),
   MSGDEST.QUEUE, MSGTYPE.WAVGMSG);
 mAlarmOut = new MsgOutboundEndpoint(
   alarm != null ? alarm : String.join("_", ALARM_DEST, System.getProperty("user.name")),
   MSGDEST.QUEUE, MSGTYPE.WALARM);
 mConsoleLogger = new ConsoleLogger();

 // New instanciation for Exercise Alarm input
 mAlarmIn = new MsgInboundEndpoint(
   alarm != null ? alarm : String.join("_", ALARM_DEST, System.getProperty("user.name")),
   MSGDEST.QUEUE, MSGTYPE.WALARM);
}

protected void setupTranslators(String[] pArgs) {
 mWRecTransformer = new WeatherRecordTransformer();
}

protected void setupFilters(String[] pArgs) {
 // create other transformer and filter as needed
 mHighWindFilter = new AlarmFilter(WPARAM.WINDGUST, 30.0, null);
 mLowTempFilter = new AlarmFilter(WPARAM.OUTTEMP, null, 0.0);
 mDayHumidAvgColl = new AverageCollector(WPARAM.OUTHUMIDITY, AVGPERIOD.DAY);
 mHourTempAvgColl = new AverageCollector(WPARAM.OUTTEMP, AVGPERIOD.HOUR);
}

/**
 * Setup the process(es) to be run and add all processes to the process list
 *
 * @param pArgs
 */
protected void setupFlows(String[] pArgs) {
 // flow definition
 mFlows.add(new ProcessSequence(mInp, mWRecTransformer, mConvertedMsgRecordsOut));
 // multi destinaton rerouter
 mMultiDestinationRouter = new MultiDestinationRouter(
   new ProcessSequence(mLowTempFilter, mAlarmOut), new ProcessSequence(mHighWindFilter, mAlarmOut),
   new ProcessSequence(mHourTempAvgColl, mAvgHourOut),
   new ProcessSequence(mDayHumidAvgColl, mAvgDayOut));
 mFlows.add(new ProcessSequence(mConvertedMsgRecordsIn, mMultiDestinationRouter));
```

```
 // alarm to console logger
 mFlows.add(new ProcessSequence(mAlarmIn, mConsoleLogger));
}

private boolean start() {
 boolean success = true;
 try {
  IProcessor activeFlow = null;
  for (IProcessor flow : mFlows) {
   try {
    activeFlow = flow;
    success = flow.start();
    if (!success) {
     break;
    }
   } catch (Exception pEx) {
    System.err.printf("Exception while starting process: flow=%s ex=%s",
      (activeFlow != null) ? activeFlow.getClass().getSimpleName() : "unknown", pEx.getMessage());
    pEx.printStackTrace(System.err);
   }
  }
 } catch (Exception pEx) {
  System.err.printf("Error when starting the producer err=%s\n", pEx.getMessage());
  success = false;
 }
 return success;
}


private void stop() {
 try {
  if (mMessageFactory != null) {
   mMessageFactory.stop();
  }
  mAvgDayOut.stop();
  mAvgHourOut.stop();
  mAlarmOut.stop();
  mConvertedMsgRecordsOut.stop();
  mConsoleLogger.stop();
  mInp.stop();
  mConvertedMsgRecordsIn.stop();
 } catch (Exception pEx) {
  System.err.printf("Error when shutting down err=%s\n", pEx.getMessage());
 }
}


private void run() {

 while (true) {
  IProcessor activeFlow = null;
  try {
   Message inMsg = null; // At begin there is no message
   for (IProcessor flow : mFlows) {
    activeFlow = flow;
    if (flow.canHandle(inMsg)) {
     flow.process(inMsg);
    }
   }
  } catch (Exception pEx) {
   System.err.printf("Exception while running process: flow=%s ex=%s",
     (activeFlow != null) ? activeFlow.getClass().getSimpleName() : "unknown", pEx.getMessage());
   pEx.printStackTrace(System.err);
  }
  try {
   Thread.sleep(100); // Avoid busy wait
```

```
   } catch (Exception pEx) {
    // ignore on purpose
   }
  }
 }


 public static void main(String[] pArgs) {
  IntegratorMain app = new IntegratorMain();
  app.setup(pArgs);
  if (app.start()) {
   app.run();
   app.stop();
   System.exit(0);
  } else {
   System.exit(1);
  }
 }


}
```