



FH MÜNSTER
University of Applied Sciences

MSB

FB Wirtschaft
Münster School of Business

Batch- verarbeitung

Mit Spring Batch

Felix Schulze Sindern



Einleitung

Lernziele

Ihr kennt...

- die Merkmale einer Batch-Anwendung
- die Grundkonzepte von Spring Batch
- Stärken und Schwächen von Spring Batch

Ihr könnt...

- Euch in bestehenden Spring Batch Anwendungen zurecht finden
- Eine einfache Batchanwendung mit Spring Batch aufsetzen

Einleitung

Batchverarbeitung - Definition

- Ohne Benutzerinteraktion
- lang laufend (rechenintensiv) oder Massendatenverarbeitung (datenintensiv)
- Ausführung von geschäftskritischen Aufgaben
- Zeit- oder Ereignisgesteuert

In Anlehnung an: [\[TM14\]](#)

Einleitung

Batchverarbeitung - Beispiele

- Automatisierte Massendatenverarbeitung – typischer Weise zeitbasiert
 - Berechnungen am Monatsende, Mitteilungen oder Korrespondenz
- Regelmäßige Anpassung von Geschäftsregeln in Systemen
 - Tarifierungen, Zinsfußänderung, Ermittlung von Versicherungsleistungen
- Integration von Informationen aus externen Quellen – erfordert häufig Formatierung, Transformierung, Validierung der Informationen
 - Erneutes Trainieren von recommender systems

In Anlehnung an: [\[SI20\]](#)

Einleitung

Batchverarbeitung – typischer Ablauf

- Abstrakt gesehen besteht ein Batch Job aus drei Schritten:



1. Daten einlesen



2. Daten verarbeiten



3. Daten persistieren

Bildquellen: <https://undraw.co/illustrations>

Spring Batch

Einführung

- Java Framework zur Erstellung von Batchprogrammen
- Bietet wiederverwendbare Funktionen zur Batchverarbeitung
 - Logging und Tracing
 - Transaktionsmanagement
 - Job Statistiken
- Liefert Schnittstellen zur Verbesserung der Fehlertoleranz/Robustheit
 - Job Restart
 - Retry/Skip
- Bietet Möglichkeiten zur parallelen und verteilten Verarbeitung [PP20]



Bildquellen:

https://pbs.twimg.com/profile_images/1235943430519435264/fgg5R6sl_400x400.png

https://upload.wikimedia.org/wikipedia/commons/thumb/4/44/Spring_Framework_Logo_2018.svg/1920px-Spring_Framework_Logo_2018.svg.png

Spring Batch

Warum Spring Batch?

- Eine simple Batchanwendung ist vergleichsweise einfach umsetzbar
- Eigene Lösungen sind häufig nur schlecht wartbar, da nur der Autor den Überblick hat (Wartungshölle)
- Spring Batch bietet eine feste Struktur wie Batchanwendungen erstellt werden können
 - Struktur ist gut skalierbar
 - In Umgebungen mit niedriger Fehlertoleranz bewährt
 - Struktur von Spring Batch folgt dem Java Standard JSR-352 [[JS20](#), [TF13](#)]
- Synergieeffekte können genutzt werden wenn eine Domänenverwantes Spring Projekt bereits existiert

Spring Batch

Reader, Writer und Processor

- Das Lesen, Verarbeiten und Schreiben von Daten wird in Spring Batch anhand von Interfaces definiert



Item Reader



Item Processor



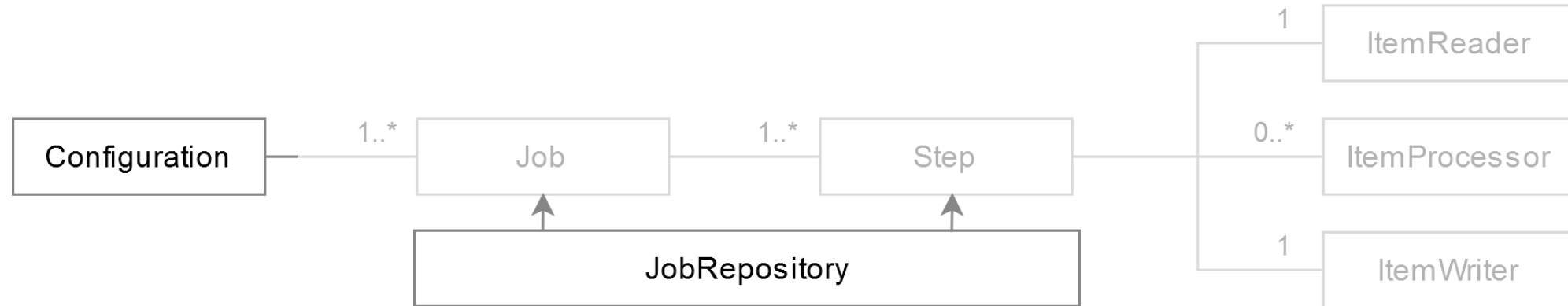
Item Writer

- Spring Batch liefert vorgefertigte Reader und Writer für beliebige Dateiformate bzw. Schnittstellen
 - Unter Anderem: JDBC, Mongo, JSON, Flatfile, Kafka, AMQP [[RW20](#)]

Bildquellen: <https://undraw.co/illustrations>

Spring Batch

Konzepte



- Definiert die Jobs und ihre Bestandteile
- Fassen Steps zusammen
- Werden sequentiell ausgeführt
- Werden parallel ausgeführt
- Ergebnisse werden weitergereicht
- Beliebige viele Processors möglich
- Reader und Writer sind vorgeschrieben

Spring Batch

Hello World

- Liest eine Datei input.txt die einen Namen enthalten soll
 - z. B.: „World“
- Batch Anwendung liest den Namen und schreibt eine Begrüßung in eine neue Datei
 - z. B.: „Hello World!“
- Wenn mehrere Namen in der Inputdatei angegeben werden, werden auch mehrere Begrüßungen ausgegeben

Hinweis: Das Hello World Beispiel ist im Repository auf dem Branch „hello-world“ zu finden

```

@Configuration
@EnableBatchProcessing
public class BatchConfiguration {
    @Autowired
    public JobBuilderFactory jobBuilderFactory;

    @Autowired
    public StepBuilderFactory stepBuilderFactory;

    @Bean
    public Job helloWorldJob() {
        return jobBuilderFactory.get("helloWorldJob")
            .start(helloWorldStep())
            .build();
    }

    @Bean
    public Step helloWorldStep() {
        return stepBuilderFactory.get("helloWorldStep")
            .<String, String>chunk(1)
            .reader(nameReader())
            .processor(new GreetingProcessor())
            .writer(greetingWriter())
            .build();
    }

    @Bean
    public ItemReader<String> nameReader() {
        return new FlatFileItemReaderBuilder<String>()
            .name("nameReader")
            .resource(new FileSystemResource("input.txt"))
            .lineMapper(new PassThroughLineMapper())
            .build();
    }
}

```

```

@Bean
public FlatFileItemWriter<String> greetingWriter() {
    return new FlatFileItemWriterBuilder<String>()
        .name("greetingWriter")
        .resource(new FileSystemResource("output.txt"))
        .lineAggregator(new PassThroughLineAggregator<String>())
        .build();
}
}

```

```

public class GreetingProcessor implements ItemProcessor<String, String> {

    @Override
    public String process(String name) throws Exception {
        return "Hello " + name + "!";
    }

}

```

```

@SpringBootApplication
public class HelloworldApplication {

    public static void main(String[] args) {
        SpringApplication.run(HelloworldApplication.class, args);
    }

}

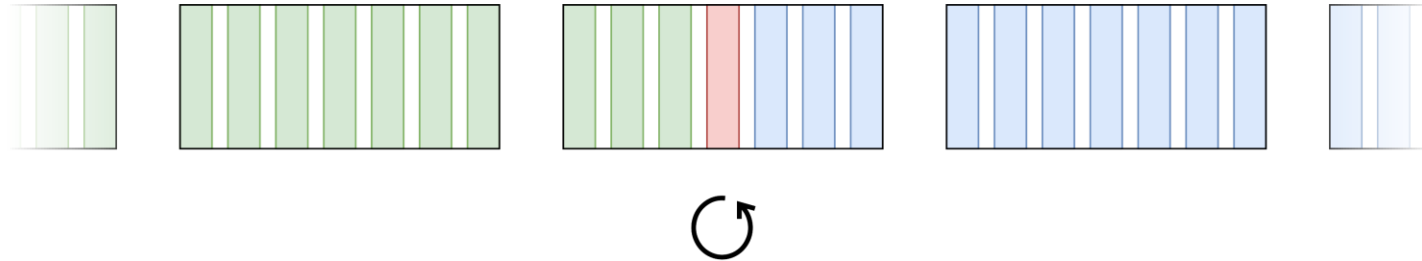
```

Hinweis: Das Hello World Beispiel ist im Repository auf dem Branch „hello-world“ zu finden

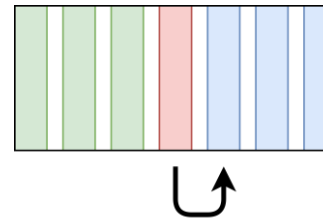
Spring Batch

Weitere Features

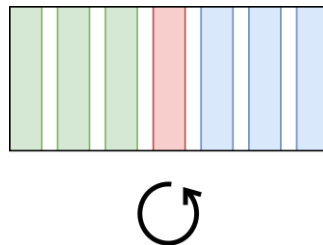
- Restart



- Skip



- Retry



In Anlehnung an: [\[TF12\]](#)

Spring Batch

Retry mit Codebeispiel

```
@Bean
public Step retryStep(ItemWriter<String> writer) {
    return stepBuilderFactory.get("retryStep")
        .<String, String>chunk(10)
        .reader(reader)
        .processor(processor)
        .writer(writer)
        .faultTolerant()
        .retryLimit(3)
        .retry(ConnectTimeoutException.class)
        .retry(DeadlockLoserDataAccessException.class)
        .build();
}
```

- **faultTolerant** gibt an, dass der Step nicht abbricht, wenn eine Exception auftritt
 - Wird benötigt, damit retry greifen kann
- **retryLimit** gibt an, wie häufig ein Step beim auftreten einer Exception wiederholt werden soll
- Mit **retry(Exception)** kann spezifiziert werden, bei welchen Exceptions der Step wiederholt werden soll



Übung

Batcherstellung von Kontoauszügen



Übung: Erstellung von Kontoauszügen

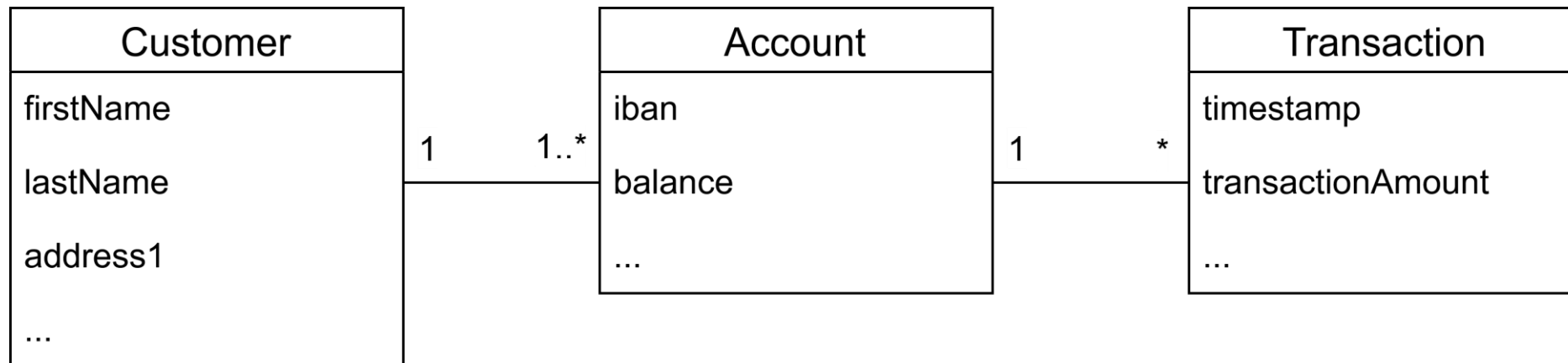
Problemstellung

- Du arbeitest bei der Münster Bank und wurdest beauftragt eine Batchanwendung zu erstellen, die jeden Monat druckbare Kontoauszüge erstellt.
- Pro Kunde soll jeweils ein Kontoauszug erstellt werden
- Ein Kontoauszug listet die Kontostände und Buchungen aller Konten eines Kunden auf



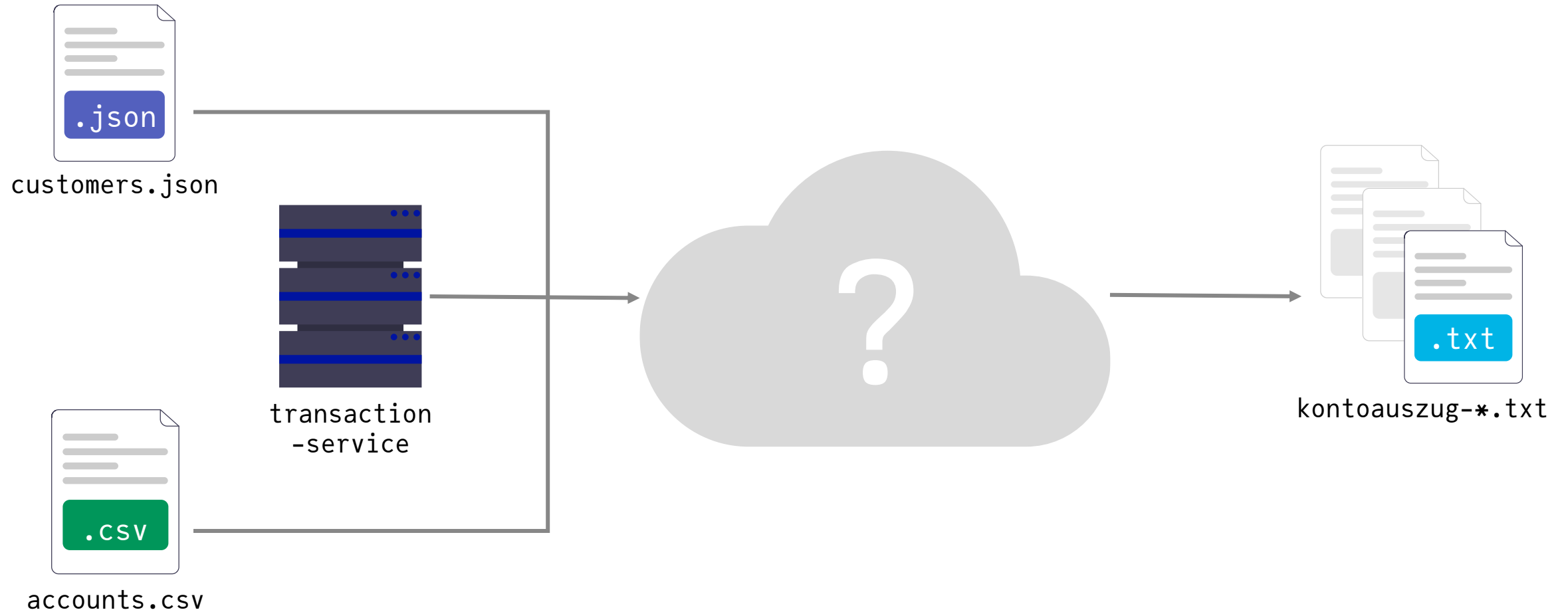
Übung: Erstellung von Kontoauszügen

Klassendiagramm



Übung: Erstellung von Kontoauszügen

Datenquellen und Senken



Übung: Erstellung von Kontoauszügen

Ausgangssituation



- Die Batchanwendung ist nur teilweise implementiert
- Aktuell werden unvollständige Kontoauszüge erstellt
 - Die Kunden und deren Konten werden richtig eingelesen und verarbeitet
 - Allerdings werden die Buchungen der Konten noch nicht beim Transactionservice abgefragt

Übung: Erstellung von Kontoauszügen

Ausgangssituation



Kundenservice Hotline

(0800) 12345

Rund um die Uhr für Sie erreichbar

Amanda Dudmarsh
127 Tomscot Park
Kansas City, Missouri 64179

Münster Bank
Corrensstraße 25
48149 Münster

Kontoauszug für Ihr Konto mit der IBAN: DE95500105174715376938

Kontostand am 2020-12-29: 15209,25 €

Es wurden keine Transaktionen im Zeitraum getätigt.

Kontostand am 2021-01-05: 0,00 €

Kontoauszug für Ihr Konto mit der IBAN: DE59500105178486977952

Kontostand am 2020-12-29: 18240,75 €

Es wurden keine Transaktionen im Zeitraum getätigt.

Kontostand am 2021-01-05: 0,00 €

Übung: Erstellung von Kontoauszügen

Umsetzung mit Spring Batch – Step 1



```
@Bean
public Job createStatementsJob() {
    return jobBuilderFactory.get("createStatementsJob")
        .start(importCustomersStep())
        .next(importAccountsStep())
        .next(fetchTransactionsStep())
        .next(calculateNewBalancesStep())
        .next(generateStatementsStep())
        .build();
}
```



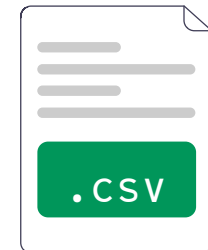
customers.json

Übung: Erstellung von Kontoauszügen

Umsetzung mit Spring Batch – Step 2



```
@Bean
public Job createStatementsJob() {
    return jobBuilderFactory.get("createStatementsJob")
        .start(importCustomersStep())
        .next(importAccountsStep())
        .next(fetchTransactionsStep())
        .next(calculateNewBalancesStep())
        .next(generateStatementsStep())
        .build();
}
```



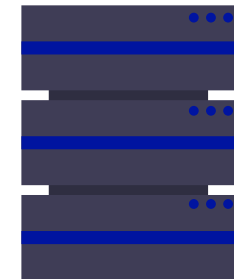
accounts.csv

Übung: Erstellung von Kontoauszügen

Umsetzung mit Spring Batch – Step 3



```
@Bean
public Job createStatementsJob() {
    return jobBuilderFactory.get("createStatementsJob")
        .start(importCustomersStep())
        .next(importAccountsStep())
        .next(fetchTransactionsStep())
        .next(calculateNewBalancesStep())
        .next(generateStatementsStep())
        .build();
}
```



transaction
-service

Übung: Erstellung von Kontoauszügen

Umsetzung mit Spring Batch – Step 4



```
@Bean
public Job createStatementsJob() {
    return jobBuilderFactory.get("createStatementsJob")
        .start(importCustomersStep())
        .next(importAccountsStep())
        .next(fetchTransactionsStep())
        .next(calculateNewBalancesStep())
        .next(generateStatementsStep())
        .build();
}
```



Übung: Erstellung von Kontoauszügen

Umsetzung mit Spring Batch – Step 5



```
@Bean
public Job createStatementsJob() {
    return jobBuilderFactory.get("createStatementsJob")
        .start(importCustomersStep())
        .next(importAccountsStep())
        .next(fetchTransactionsStep())
        .next(calculateNewBalancesStep())
        .next(generateStatementsStep())
        .build();
}
```



kontoauszug-*.txt

Übung: Erstellung von Kontoauszügen

Aufgabe 1



- a) Buchungen vom transaction-service abfragen
- Binde den bereits vorliegenden `HttpTransactionProcessor` im `fetchTransactionsStep` ein.
 - Kontrolliere deine Ergebnisse indem du die Batchanwendung mit `mvn spring-boot:run` ausführst.

Übung: Erstellung von Kontoauszügen

Aufgabe 2 - Ausgangssituation



- Der transaction-service ist unzuverlässig geworden und liefert gelegentlich Antworten mit dem Statuscode „500: Internal Server Error“
- Vorbereitung: Zum „task-2“ Branch wechseln
 - Unstaged changes verwerfen mit `git checkout .`
 - Zum neuen Branch wechseln mit `git checkout task-2`

Übung: Erstellung von Kontoauszügen

Aufgabe 2



- a) Führe die Batchanwendung mit `mvn spring-boot:run` aus. Wie geht die Anwendung mit den 500-Antworten um?
- b) Verbessere die Fehlertoleranz der Batchanwendung indem du für den `fetchTransactionsStep` das Retry-Verfahren (vgl. Folie 11) anwendest.
 - Der Schritt soll erneut ausgeführt werden, wenn die `org.springframework.web.client.HttpServerErrorException` geschmissen wird
 - Der Schritt soll maximal drei Mal wiederholt werden

Übung: Erstellung von Kontoauszügen

Aufgabe 2 - Ergebnis



Kundenservice Hotline
(0800) 12345
Rund um die Uhr für Sie erreichbar

Amanda Dudmarsh
127 Tomscot Park
Kansas City, Missouri 64179

Münster Bank
Correnstraße 25
48149 Münster

Kontoauszug für Ihr Konto mit der IBAN: DE95500105174715376938

| | | | |
|------------|--------------|---------------------------|------------|
| 2021-01-05 | Yabox | Kontostand am 2020-12-29: | 15209,25 € |
| 2021-01-15 | Blogtags | | 14004,00 € |
| 2021-01-25 | Twitterworks | | -3064,00 € |
| | | | -1324,00 € |
| | | Summe Soll: | -4388,00 € |
| | | Summe Haben: | 14004,00 € |
| | | Kontostand am 2021-01-20: | 24825,25 € |

Kontoauszug für Ihr Konto mit der IBAN: DE59500105178486977952

| | | | |
|------------|-----------|---------------------------|------------|
| 2021-01-19 | Bubblemix | Kontostand am 2020-12-29: | 18240,75 € |
| | | | 9940,17 € |
| | | Summe Soll: | 0,00 € |
| | | Summe Haben: | 9940,17 € |
| | | Kontostand am 2021-01-20: | 28180,92 € |

Fazit

Spring Batch

- Spring Batch gibt Batch Anwendungen eine feste Struktur
 - Bessere Wartbarkeit
- Bietet Schnittstellen zur Skalierung und Fehlerbehandlung
 - Parallele und verteilte Verarbeitung
 - Restart, Retry, Skip
- Kommt mit vorgefertigten Readern und Writern
- Die fest vorgeschriebenen Strukturen können störend sein
 - Bei Anwendungsfällen, die nicht von den Features von Spring Batch profitieren
 - Bei kleinen Anwendungen

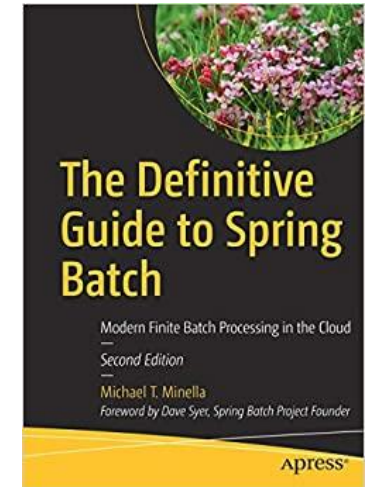
Literaturverzeichnis

Seite 1 von 2



FH MÜNSTER
University of Applied Sciences

| | |
|--------|--|
| [MM19] | The Definitive Guide to Spring Batch: Modern Finite Batch Processing in the Cloud Michael T. Minella Chicago, IL, USA ISBN-13: 978-1-4842-3723-6 |
| [TF13] | Spring Batch and JSR-352 (Batch Applications for the Java Platform) – Differences Tobias Flohre https://blog.codecentric.de/en/2013/07/spring-batch-and-jsr-352-batch-applications-for-the-java-platform-differences/ (abgerufen am: 02.01.21) |
| [PP20] | Scaling and Parallel Processing - Spring Batch - Reference Documentation Spring Batch Documentation Authors https://docs.spring.io/spring-batch/docs/current/reference/html/scalability.html (abgerufen am: 02.01.21) |
| [TM14] | Java-Batch JSR-352 - Der neue Standard in Java EE 7 Thomas Much https://www.muchsoft.com/presentations/jughh-javabatch.pdf (abgerufen am: 02.01.21) |
| [TF12] | Transaktionen in Spring Batch: Massenverarbeitung mit Restart, Skip und Retry Tobias Flohre https://www.codecentric.de/wissen/publikation/transaktionen-in-spring-batch-massenverarbeitung-mit-restart-skip-und-retry (abgerufen am: 02.01.21) |



Bildquelle: https://images-na.ssl-images-amazon.com/images/I/51+4Bnv2GSL._SY344_BO1,204,203,200_.jpg

Literaturverzeichnis

Seite 2 von 2



| | |
|--------|---|
| [JS20] | JSR-352 Support - Spring Batch - Reference Documentation Spring Batch Documentation Authors https://docs.spring.io/spring-batch/docs/current/reference/html/jsr-352.html (abgerufen am: 02.01.21) |
| [SI20] | Spring Batch Introduction - Spring Batch - Reference Documentation Spring Batch Documentation Authors https://docs.spring.io/spring-batch/docs/current/reference/html/spring-batch-intro.html (abgerufen am: 02.01.21) |
| [RW20] | List of ItemReaders and ItemWriters - Spring Batch - Reference Documentation Spring Batch Documentation Authors https://docs.spring.io/spring-batch/docs/current/reference/html/appendix.html#listOfReadersAndWriters (abgerufen am: 03.01.21) |

Alle Illustrationen wurden von <https://undraw.co/search> am 11.01.21 entnommen



Vielen Dank für Ihre Aufmerksamkeit!

Felix Schulze Sindern

