

JBatch

... или не очень большие данные



Дмитрий Александров

- Ведущий эксперт-программист в T-Systems
- 10 лет коду на энтерпрайз
- Bulgarian JUG co-lead
- Балуюсь организацией конференций
- Люблю самолетики

Немного о сегодняшнем докладе

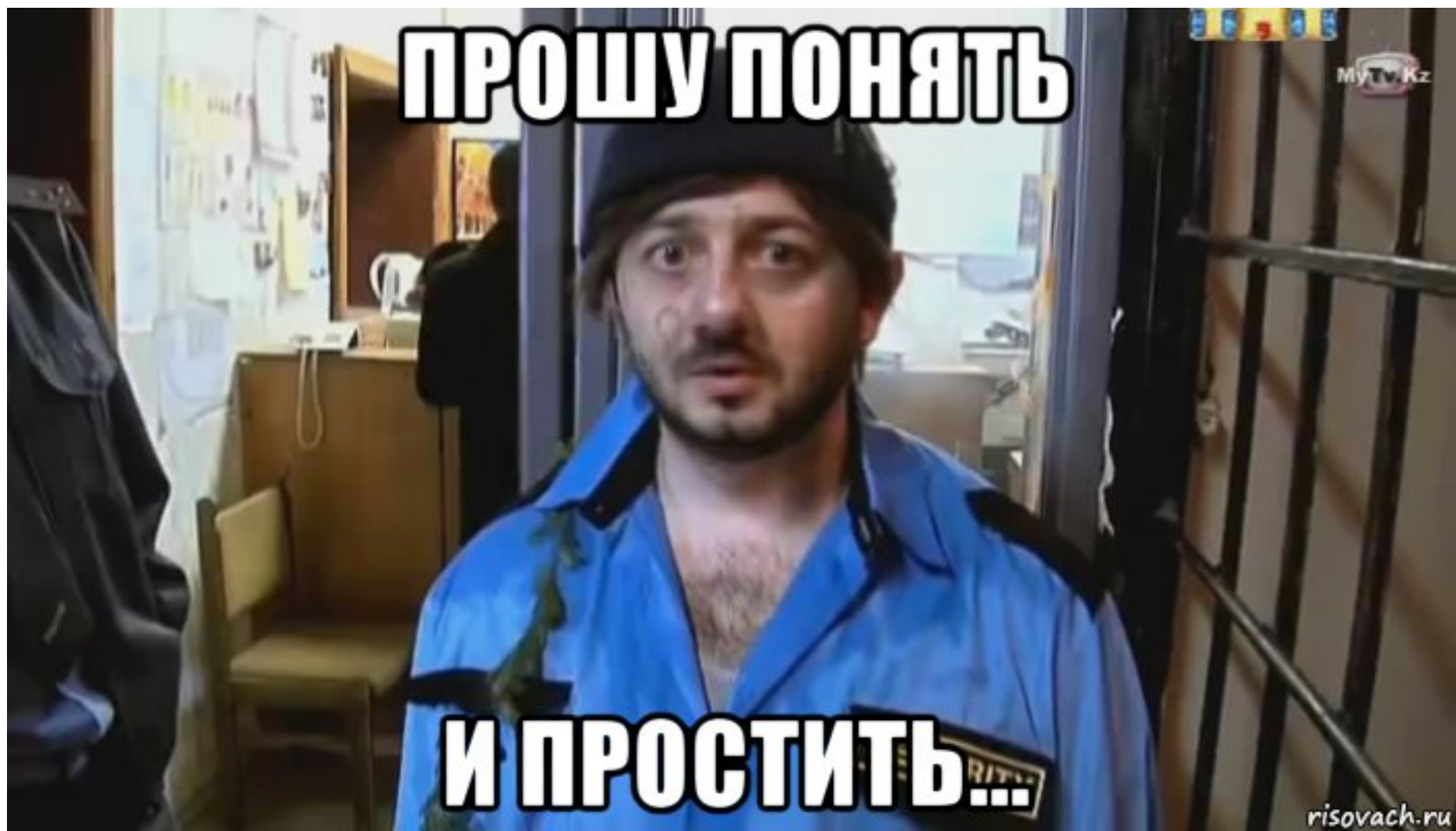
- Чаще живу не в России, могу странно произносить слова

Немного о сегодняшнем докладе

- Чаще живу не в России, могу странно произносить слова
- По умолчанию буду рассуждать в рамках Java EE

Немного о сегодняшнем докладе

- Чаще живу не в России, могу странно произносить слова
- По умолчанию буду рассуждать в рамках Java EE
- Для демо буду использовать Arquillian



Joker<?>



ПОЕХАЛИ!

а вообще зачем нужны
компьютеры?

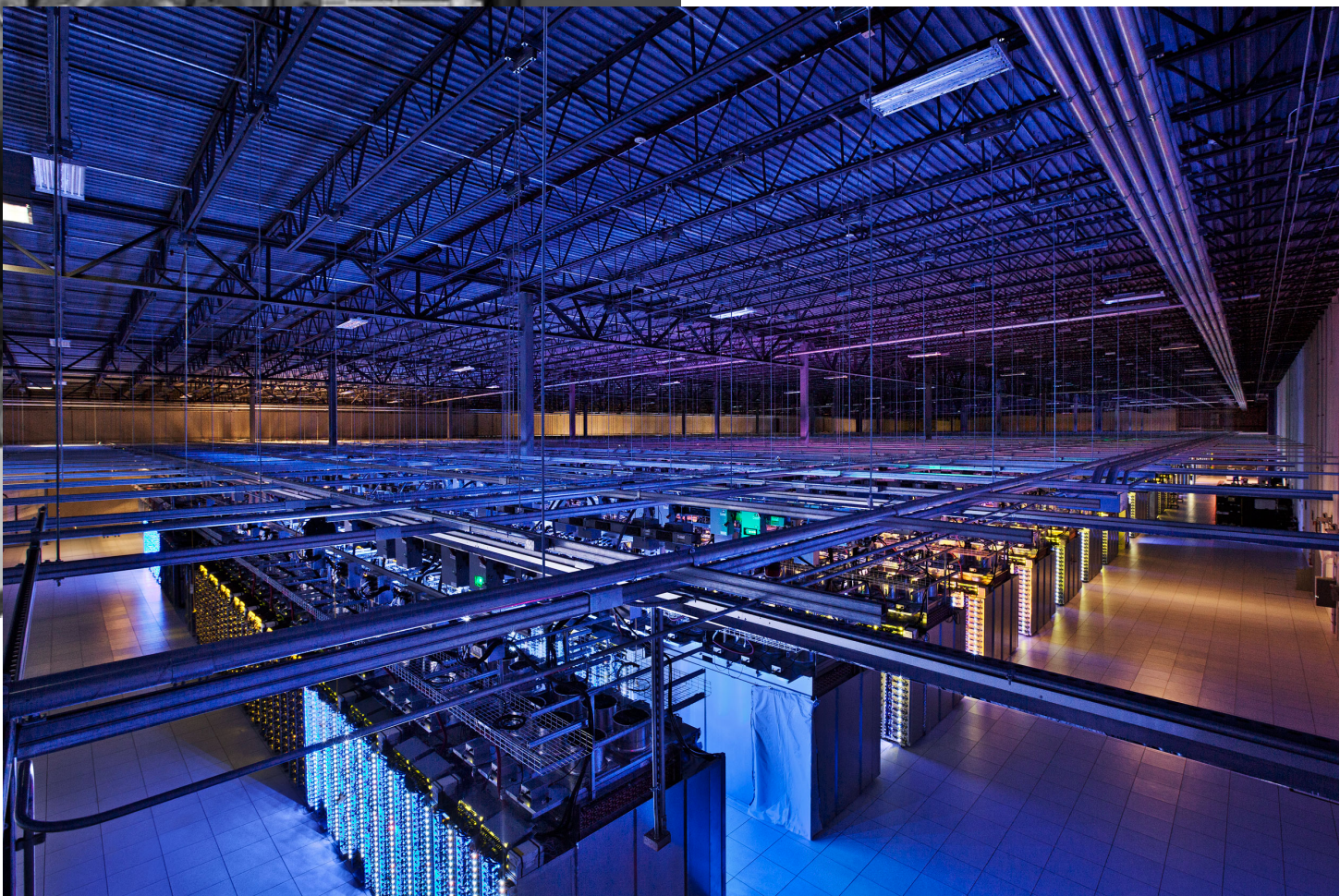
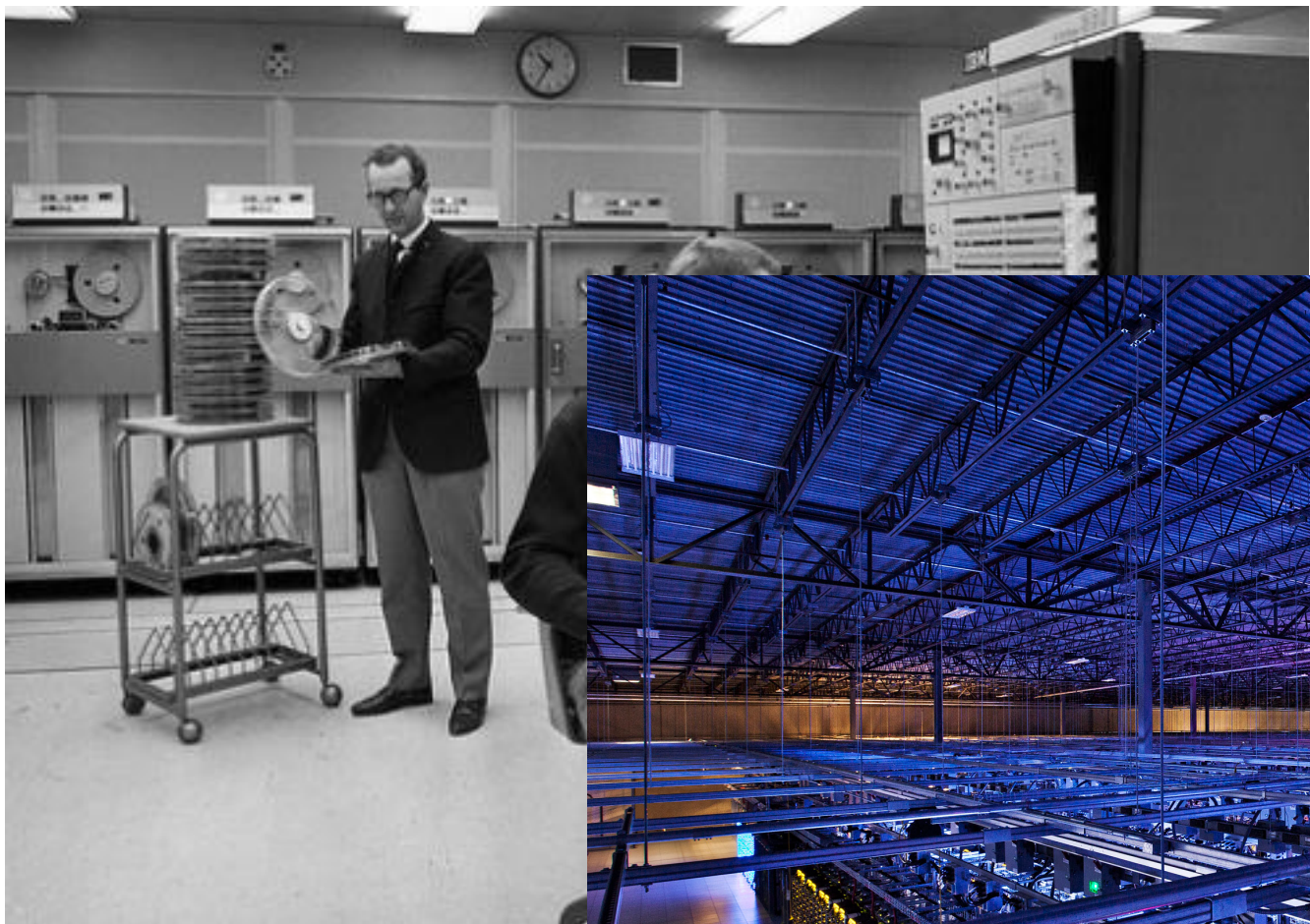


Правильный ответ:
Смотреть котиков!



... но вообще-то они помогают
в автоматизации
обработки данных





... а вы помните свою
первую задачу на работе?

Я более чем уверен:

1. Взять CSV файл
2. Распарсить его
3. Сохранить в базе



Миром энтерпрайза до сих пор правят CSV/XML файлы!



А что вообще такое Batch?

А что вообще такое Batch?

- *Пакетная обработка.*

А что вообще такое Batch?

- *Пакетная обработка.*
- *(почти) не участвует оператор.*

А что вообще такое Batch?

- *Пакетная обработка.*
- *(почти) не участвует оператор.*
- *(часто) как фоновый процесс.*

..наяндексил

Пакетная обработка в жизни

- Общественный транспорт



Пакетная обработка в жизни

- Общественный транспорт
- Оптовая торговля



Пакетная обработка в жизни

- Общественный транспорт
- Оптовая торговля
- Общепит ☺



А зачем вообще в программировании?

- Целый подкласс задач

А зачем вообще в программировании?

- Целый подкласс задач
- Особенно в энтерпрайзе

А зачем вообще в программировании?

- Целый подкласс задач
- Особенно в энтерпрайзе
- Исполняются долго

А зачем вообще в программировании?

- Целый подкласс задач
- Особенно в энтерпрайзе
- Исполняются долго (... ~41 час)

А зачем вообще в программировании?

- Целый подкласс задач
- Особенно в энтерпрайзе
- Исполняются долго (... ~41 час)
- За ними можно наблюдать и ими управлять

Чем полезна пакетная обработка

Чем полезна пакетная обработка

- Эффективность

Чем полезна пакетная обработка

- Эффективность
- Лучшая утилизация ресурсов

Чем полезна пакетная обработка

- Эффективность
- Лучшая утилизация ресурсов
- Автоматизация

Чем полезна пакетная обработка

- Эффективность
- Лучшая утилизация ресурсов
- Автоматизация
- Концентрация внимания на одной задаче

Что может прийти в голову при слове “Батч” в Java

Что может прийти в голову при слове “Батч” в Java

- Spring batch

Что может прийти в голову при слове “Батч” в Java

- Spring batch
- Hadoop?

Что может прийти в голову при слове “Батч” в Java

- Spring batch
- Hadoop?
- Spark?

Что может прийти в голову при слове “Батч” в Java

- Spring batch
- Hadoop?
- Spark?
- Websphere datagrid?

Что может прийти в голову при слове “Батч” в Java

Ну что-нибудь свое:

там пару циклов, прочитать файл .. записать в базу данных!

А в чем собственно проблема?

- Обычно сводится к циклу:
 - Прочитать из одного места
 - Обработать
 - Положить в другое место/записать результат

А в чем собственно проблема?

- Обычно сводится к циклу:
 - Прочитать из одного места
 - Обработать
 - Положить в другое место/записать результат
- Таких циклов может быть несколько, необходимо управлять

А в чем собственно проблема?

- Обычно сводится к циклу:
 - Прочитать из одного места
 - Обработать
 - Положить в другое место/записать результат
- Таких циклов может быть несколько, необходимо управлять
- А что если надо выполнить что-то параллельно?

А в чем собственно проблема?

- Обычно сводится к циклу:
 - Прочитать из одного места
 - Обработать
 - Положить в другое место/записать результат
- Таких циклов может быть несколько, необходимо управлять
- А что если надо выполнить что-то параллельно?
- А вдруг что-то пойдет не так?

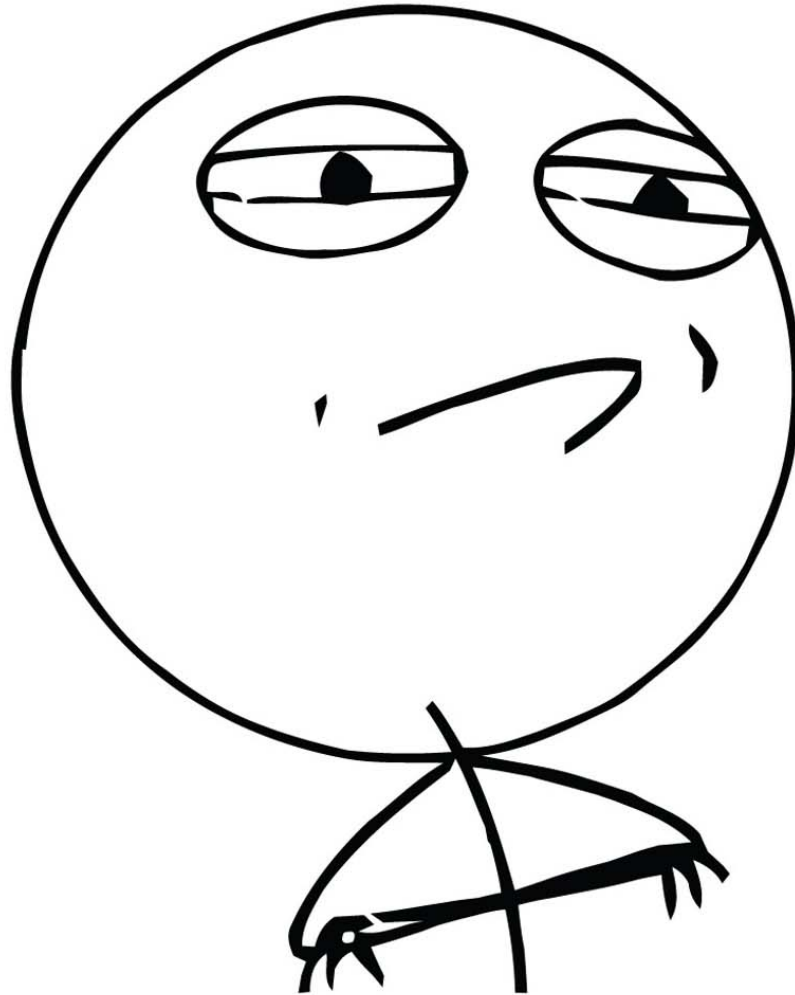
А в чем собственно проблема?

- Обычно сводится к циклу:
 - Прочитать из одного места
 - Обработать
 - Положить в другое место/записать результат
- Таких циклов может быть несколько, необходимо управлять
- А что если надо выполнить что-то параллельно?
- А вдруг что-то пойдет не так?
- А когда и с какой периодичностью запускать?

А в чем собственно проблема?

- Обычно сводится к циклу:
 - Прочитать из одного места
 - Обработать
 - Положить в другое место/записать результат
- Таких циклов может быть несколько, необходимо управлять
- А что если надо выполнить что-то параллельно?
- А вдруг что-то пойдет не так?
- А когда и с какой периодичностью запускать?
- А может нужна статистика?

тут в голову приходит мысль о
«не очень больших данных»

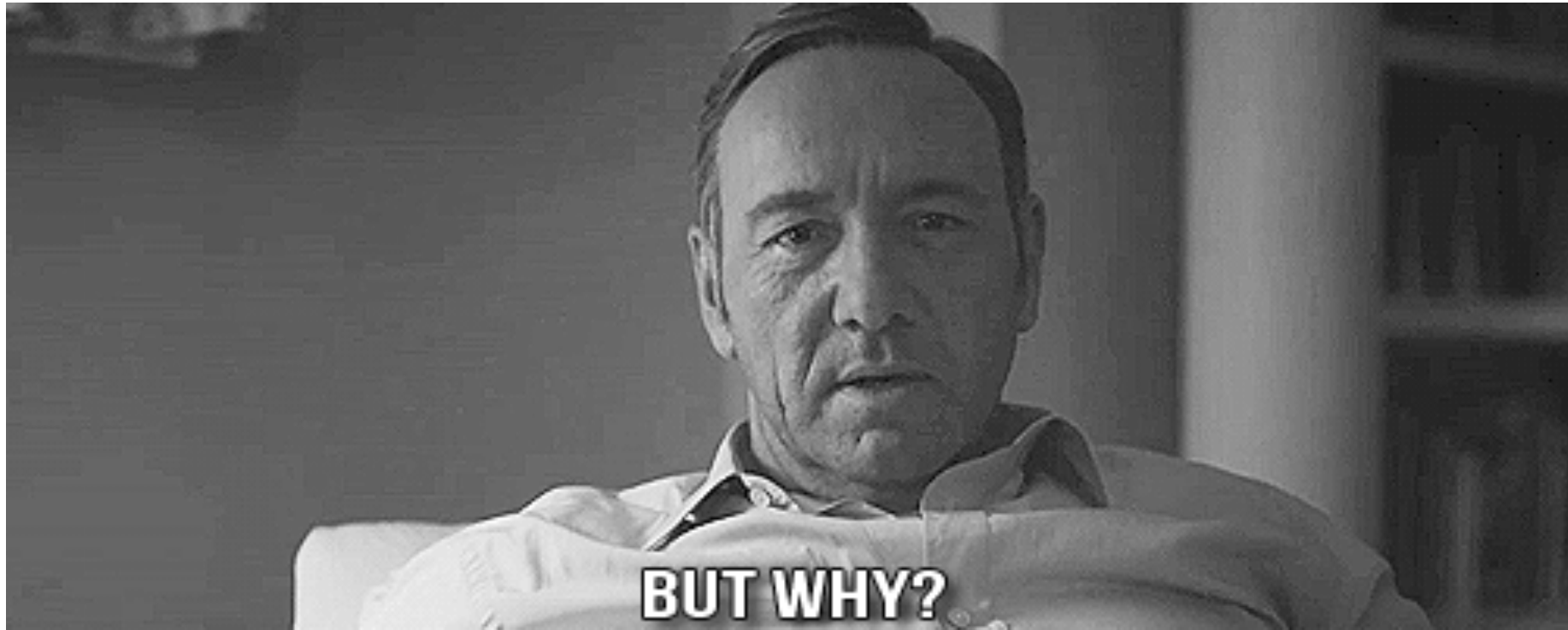


CHALLENGE ACCEPTED









I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself

DON'T REPEAT YOURSELF

Repetition is the root of all software evil

JSR-352

- Экспертная группа создана в 2011
 - IBM, VMware, RedHat, Oracle, Credit Suisse и др.



Joker

JSR-352

- Экспертная группа создана в 2011
 - IBM, VMware, RedHat, Oracle, Credit Suisse и др.
- Сделана под сильным впечатлением Spring Batch

JSR-352

- Экспертная группа создана в 2011
 - IBM, VMware, RedHat, Oracle, Credit Suisse и др.
- Сделана под сильным впечатлением Spring Batch
- Релиз спецификации 24.11.2013

JSR-352

- Экспертная группа создана в 2011
 - IBM, VMware, RedHat, Oracle, Credit Suisse и др.
- Сделана под сильным впечатлением Spring Batch
- Релиз спецификации 24.11.2013
- Часть Java EE 7!

JSR-352

- Экспертная группа создана в 2011
 - IBM, VMware, RedHat, Oracle, Credit Suisse и др.
- Сделана под сильным впечатлением Spring Batch
- Релиз спецификации 24.11.2013
- Часть Java EE 7!
- Но работает и на Java SE)

JSR-352 куул фичеры

- Обработка данных «Кусками»
или полностью

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Последовательная или параллельная обработка

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Последовательная или параллельная обработка
- Чекпоинты

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Последовательная или параллельная обработка
- Чекпоинты
- Управление потоком работы

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Остановка/восстановление
- Последовательная или параллельная обработка
- Чекпоинты
- Управление потоком работы

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Последовательная или параллельная обработка
- Чекпоинты
- Управление потоком работы
- Остановка/восстановление
- Управление исключениями

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Последовательная или параллельная обработка
- Чекпоинты
- Управление потоком работы
- Остановка/восстановление
- Управление исключениями
- Транзакционность

JSR-352 куул фичеры

- Обработка данных «Кусками» или полностью
- Последовательная или параллельная обработка
- Чекпоинты
- Управление потоком работы
- Остановка/восстановление
- Управление исключениями
- Транзакционность
- Метрики

Вместо тысячи слов



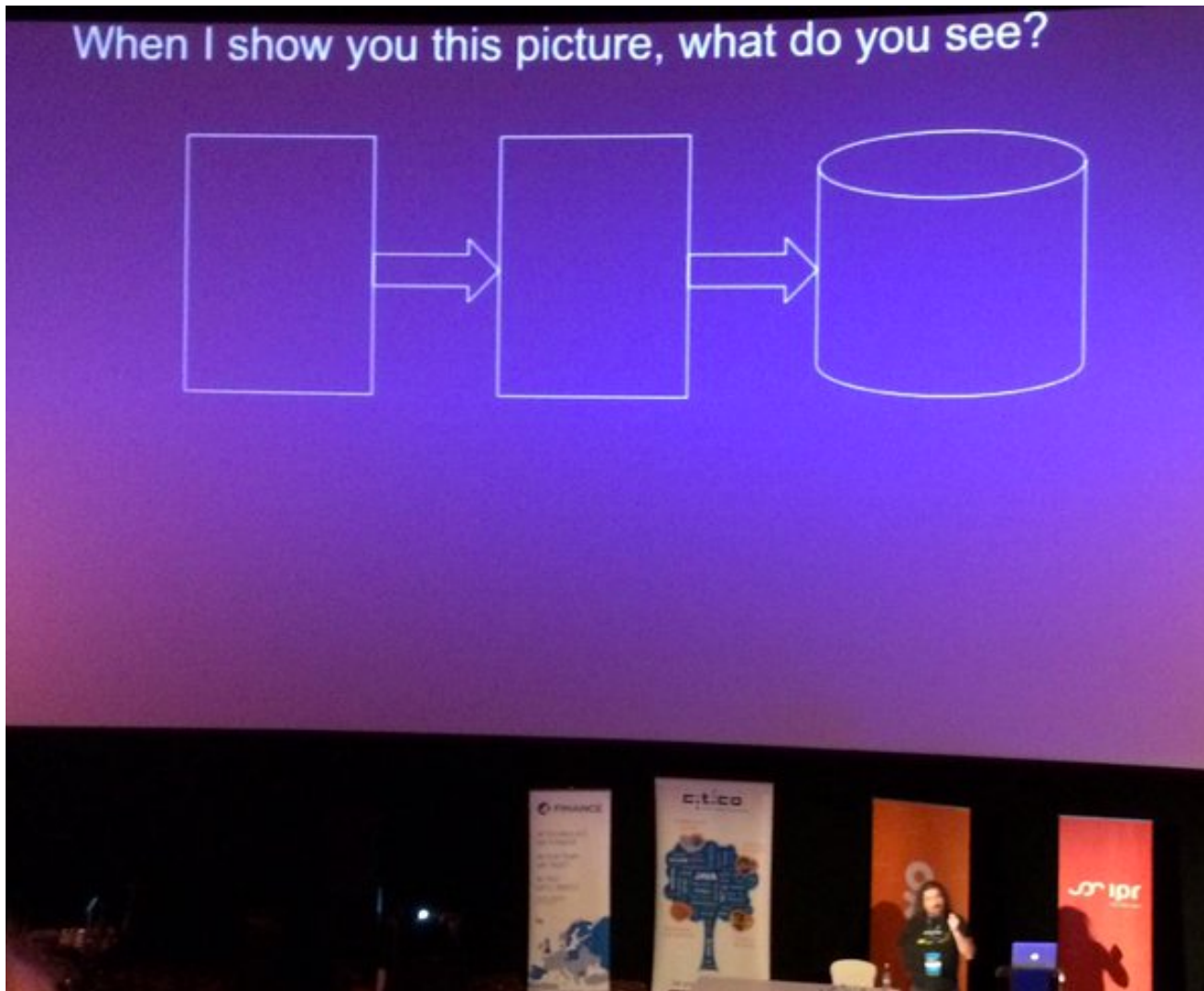
Joker<?>

А пока я тут говорю..

Код можете взять тут:

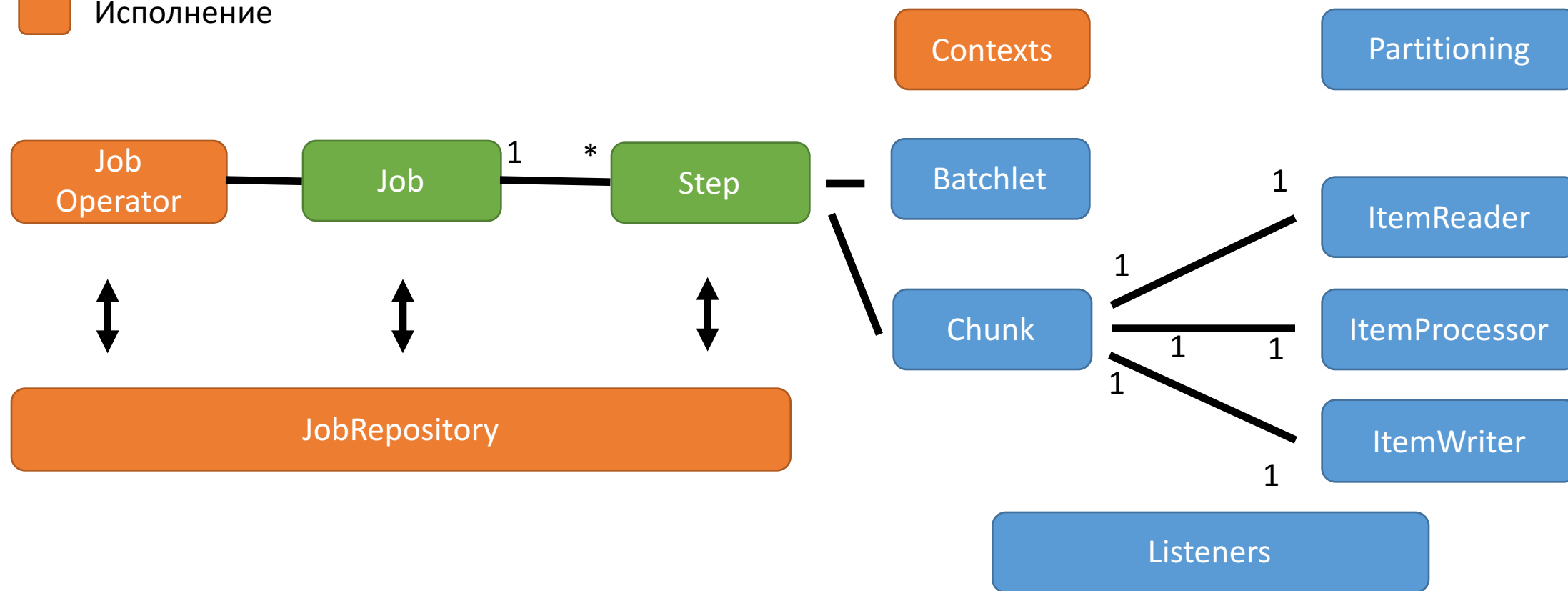
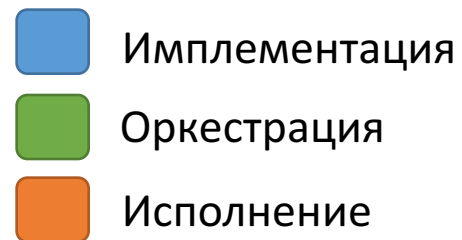
<https://github.com/dalexandrov/jbatch-demo>

Ничего не напоминает?

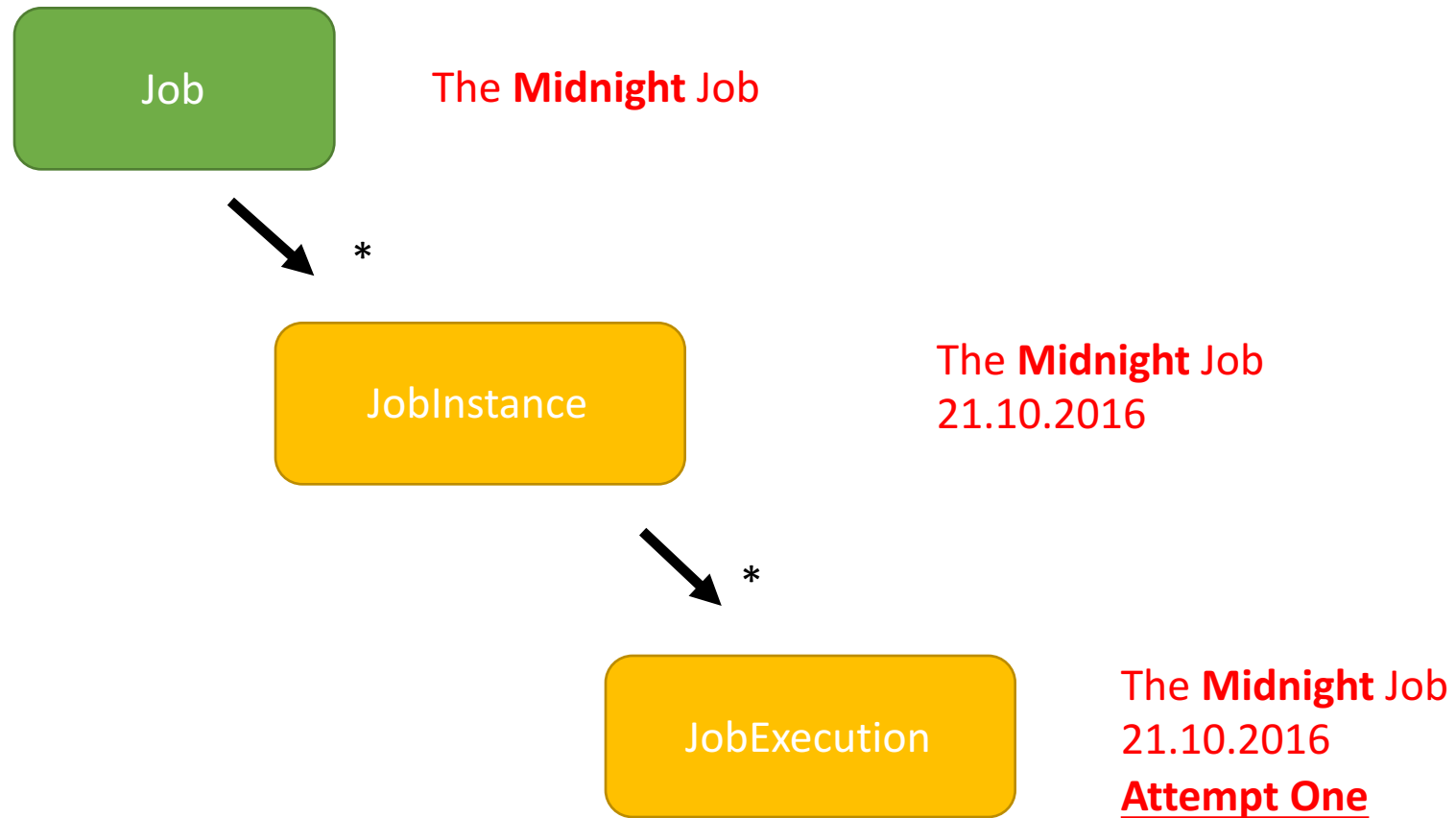


Joker<?>

Концепции



Работа штука сложная



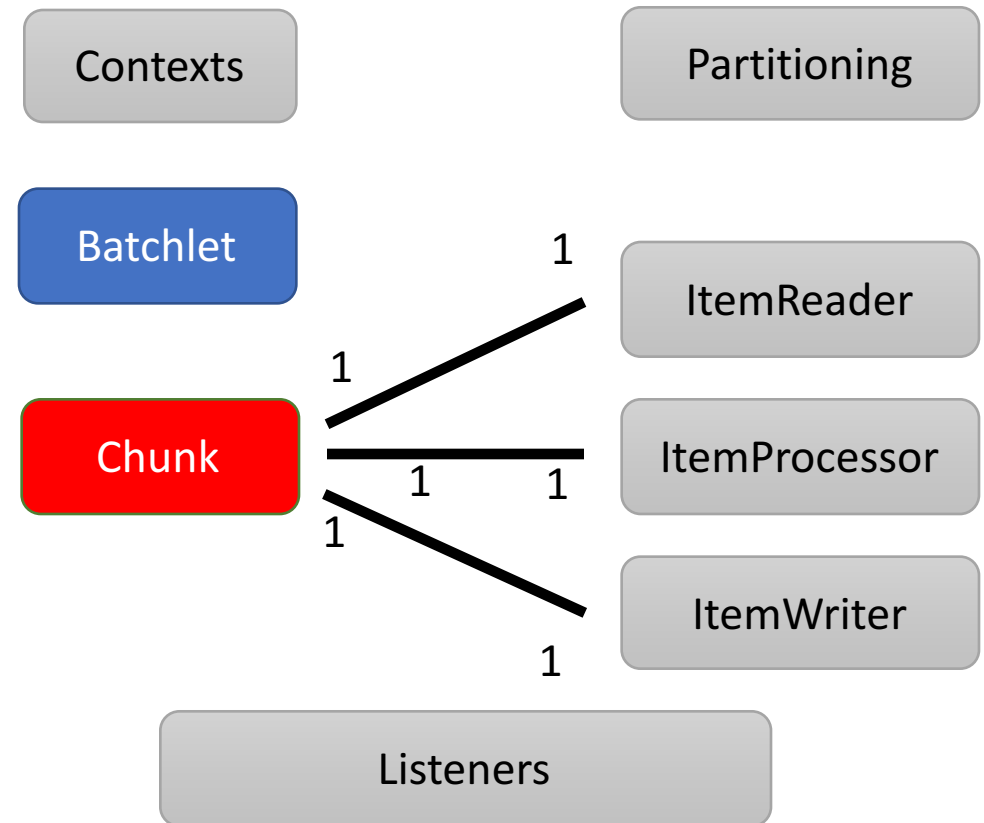
Chunk vs. batchlet

- **Chunk**

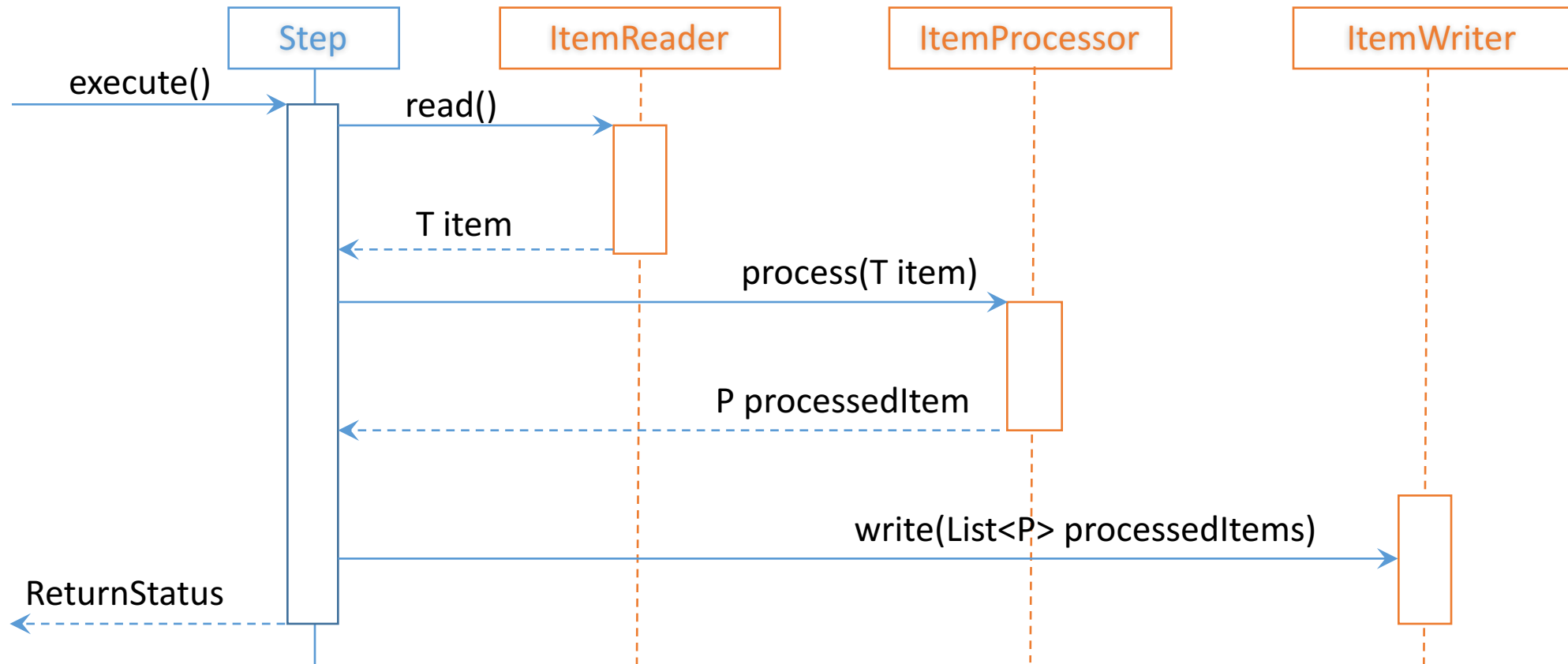
- ETL Pattern
- Содержит по одному Reader, Processor, Writer
- Reader/Processor вызываются пока обрабатывается «кусок»
- Далее записывается весь «кусок»

- **Batchlet**

- Вызывается, исполняется, возвращает return code на выходе



Внутри Step - chunk



Определение Step

Оркестрация

```
<step id="jokerStep">  
  <chunk checkpoint-policy="item" item-count="3">  
    <reader ref="myItemReader"/>  
    <processor ref="myItemProcessor"/>  
    <writer ref="myItemWriter"/>  
  </chunk>  
</step>
```

Joker<?>

ItemReader

Имплементация

@Named

```
public class MyItemReader extends AbstractItemReader {
```

```
    @Override
```

```
    public Object readItem() throws Exception {...}
```

```
}
```


ItemProcessor

Имплементация

@Named

```
public class MyItemProcessor implements ItemProcessor {
```

```
    @Override
```

```
    public Object processItem(Object item) throws Exception {...}
```

```
}
```

ItemWriter

Имплементация

@Named

```
public class MyItemWriter extends AbstractItemWriter {
```

```
    @Override
```

```
    public void writeItems(List<Object> items) throws Exception {...}
```

```
}
```

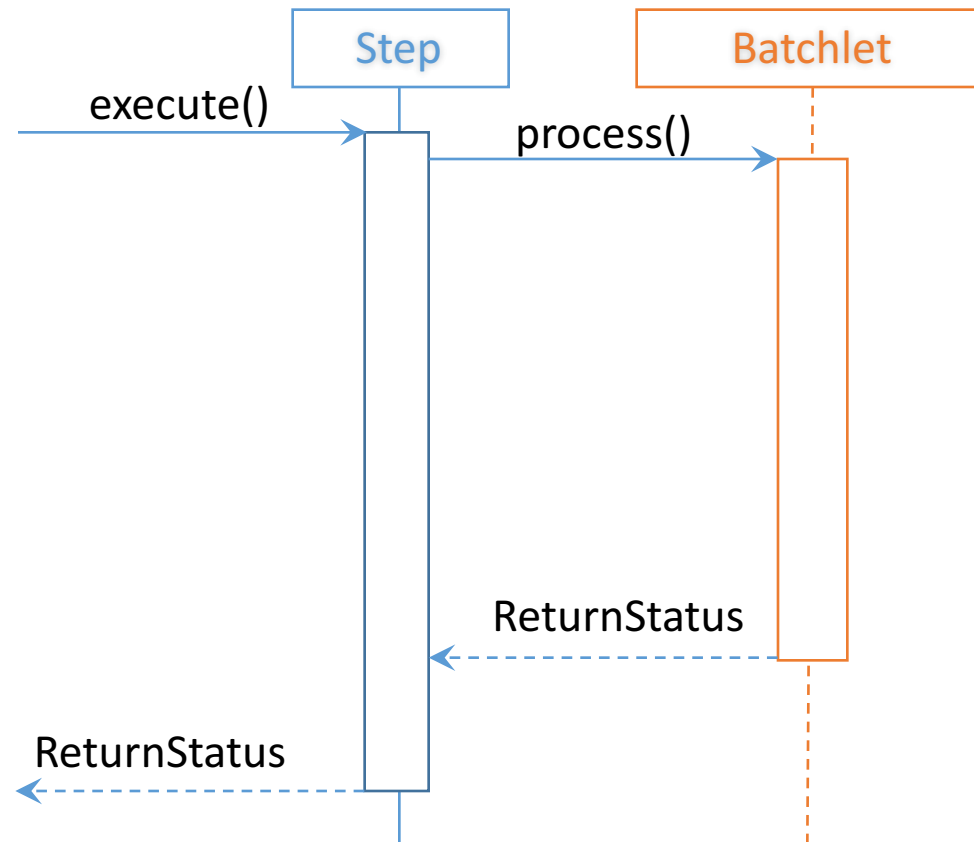
Joker<?>



... а если не все
вписывается в ETL pattern?



A batchlet вообще норм



Примерно так

@Named

```
public class MyBatchlet extends AbstractBatchlet {  
    @Override  
    public String process() {  
        System.out.println("Joker");  
        return BatchStatus.COMPLETED.toString();  
    }  
}
```

Joker<?>

Теперь попытаемся со всем этим взлететь

Исполнение

```
JobOperator jobOperator =  
    BatchRuntime.getJobOperator();
```

```
Long executionId = jobOperator.start("jokerJob", new Properties());
```

```
JobExecution jobExecution = jobOperator.getJobExecution(executionId);
```

Batch Exit Status

- Job и Step имеют exit status
- Для управления workflow
- STARTING, STARTED, STOPPING, STOPPED, FAILED, COMPLETED, ABANDONED

На этом все!
Можно на обед!

И мы собирались только
ради этого?

Усложним!

Listeners

Оркестрация

...

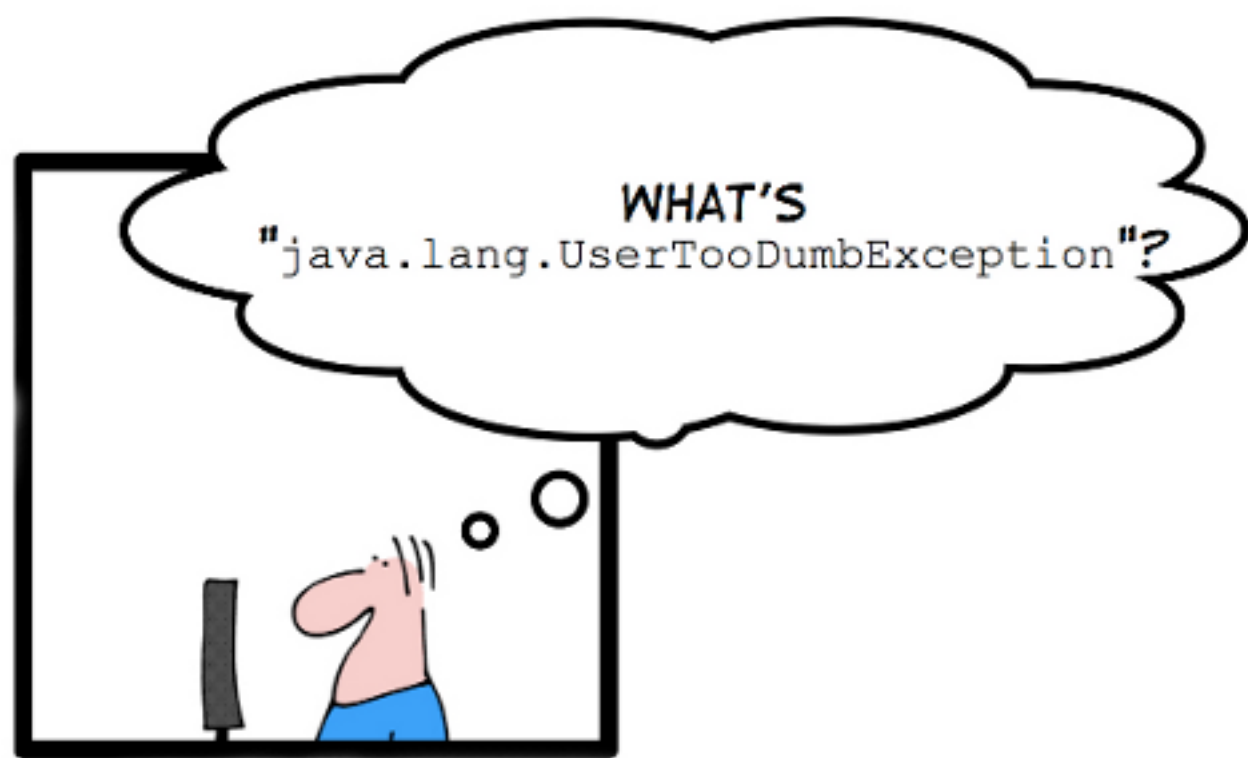
<listeners>

<listener ref="myJobListener"/>

</listeners>

...

Joker<?>



Опаньки...

Exception хэндлинг

Оркестрация

```
<chunk checkpoint-policy="item" item-count="3"  
        skip-limit="3" retry-limit="3">
```

...

```
<skippable-exception-classes>  
  <include class="java.lang.RuntimeException"/>  
  <exclude class="java.lang.IllegalArgumentException"/>  
</skippable-exception-classes>  
<retryable-exception-classes>  
  <exclude class="java.lang.IllegalArgumentException"/>  
</retryable-exception-classes>  
</chunk>
```

Joker<?>

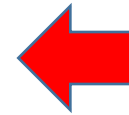
Exception хэндлинг

Оркестрация

```
<chunk checkpoint-policy="item" item-count="3"  
        skip-limit="3" retry-limit="3">
```

...

```
<skippable-exception-classes>  
  <include class="java.lang.RuntimeException"/>  
  <exclude class="java.lang.IllegalArgumentException"/>  
</skippable-exception-classes>  
<retryable-exception-classes>  
  <exclude class="java.lang.IllegalArgumentException"/>  
</retryable-exception-classes>  
</chunk>
```



Joker<?>

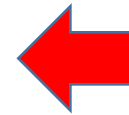
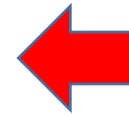
Exception хэндлинг

Оркестрация

```
<chunk checkpoint-policy="item" item-count="3"  
        skip-limit="3" retry-limit="3">
```

...

```
<skippable-exception-classes>  
  <include class="java.lang.RuntimeException"/>  
  <exclude class="java.lang.IllegalArgumentException"/>  
</skippable-exception-classes>  
<retryable-exception-classes>  
  <exclude class="java.lang.IllegalArgumentException"/>  
</retryable-exception-classes>  
</chunk>
```



Joker<?>

Checkpoint



Checkpoint

- Бывает 2х типов:
 - Item
 - Custom

Checkpoint

Оркестрация

```
<step id="jokerStep">  
  <chunk checkpoint-policy="custom">  
    <reader ref="myItemReader"/>  
    <processor ref="myItemProcessor"/>  
    <checkpoint-algorithm ref="MyCheckpointAlgorithm"/>  
  </chunk>  
</step>
```



Алгоритм!

Имплементация

@Named

```
public class MyCheckpointAlgorithm extends AbstractCheckpointAlgorithm {
```

```
....
```

@Override

```
public boolean isReadyToCheckpoint() throws Exception {
```

```
    return count % 5 == 0;
```

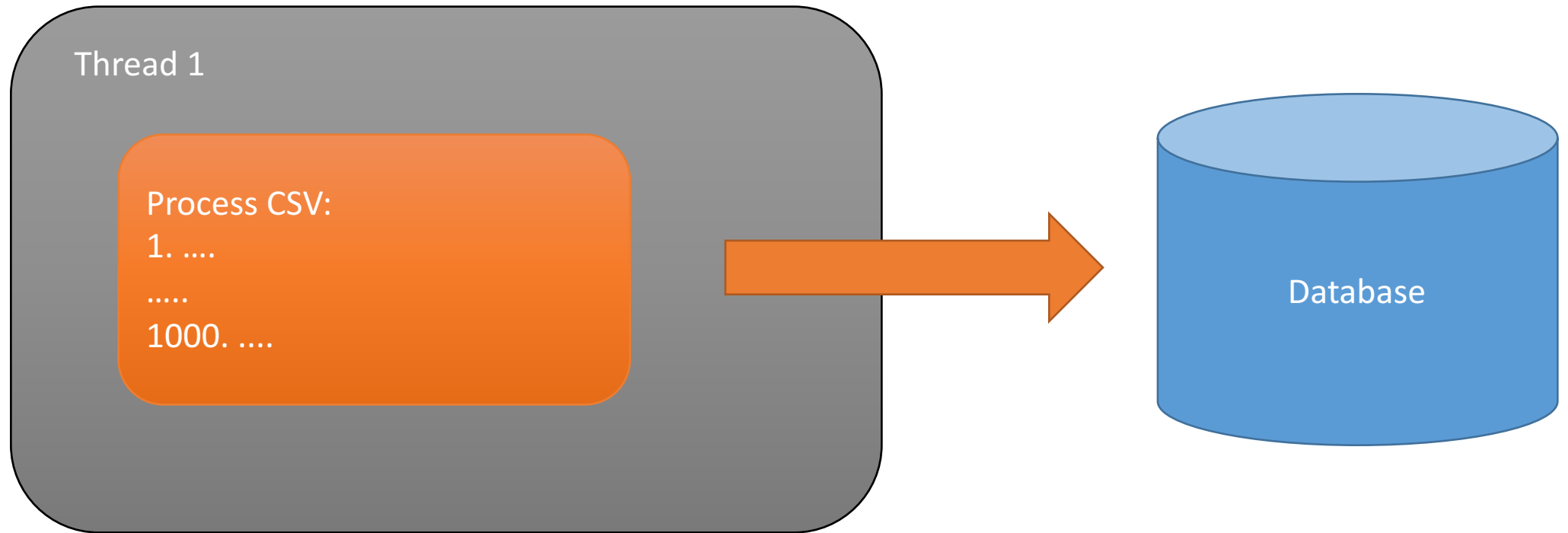
```
}
```

```
}
```

Partition



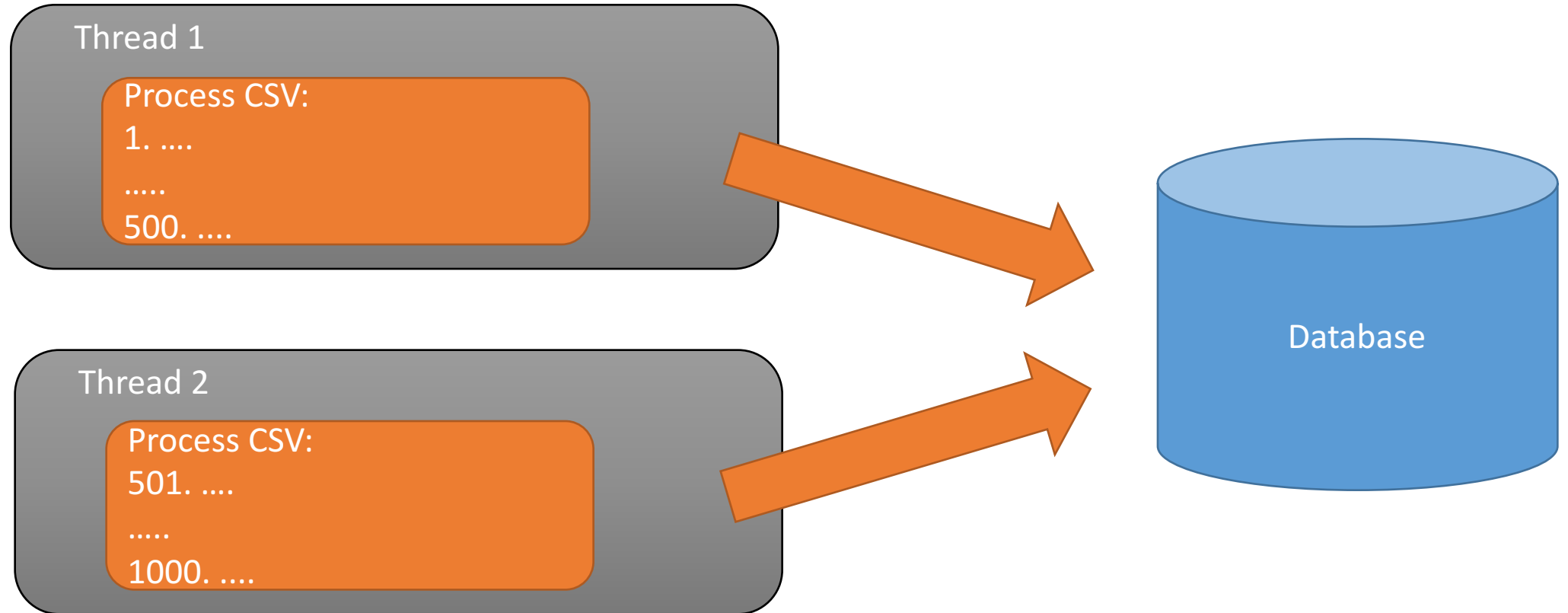
Partition



Partition

- Что-то типа «шардинга»
- Легко оркестрируется
- Легко можно написать свой алгоритм распараллеливания

Partition



Partition

Оркестрация


```
<partition>  
  <plan partitions="2">  
    <properties partition="0">  
      <property name="start" value="0"/>  
      <property name="end" value="500"/>  
    </properties>  
    <properties partition="1">  
      <property name="start" value="501"/>  
      <property name="end" value="1000"/>  
    </properties>  
  </plan>  
</partition>
```

Joker<?>

Partition

Оркестрация

```
<chunk item-count="3">
  <reader ref="myItemReader">
    <properties>
      <property name="start" value="#{partitionPlan['start']}" />
      <property name="end" value="#{partitionPlan['end']}" />
    </properties>
  </reader>
  <processor ref="myItemProcessor"/>
  <writer ref="myItemWriter"/>
</chunk>
```

Two red arrows originate from the right side of the slide. One arrow points to the expression `#{partitionPlan['start']}` within the `<property name="start" value="..." />` tag. The other arrow points to the expression `#{partitionPlan['end']}` within the `<property name="end" value="..." />` tag.

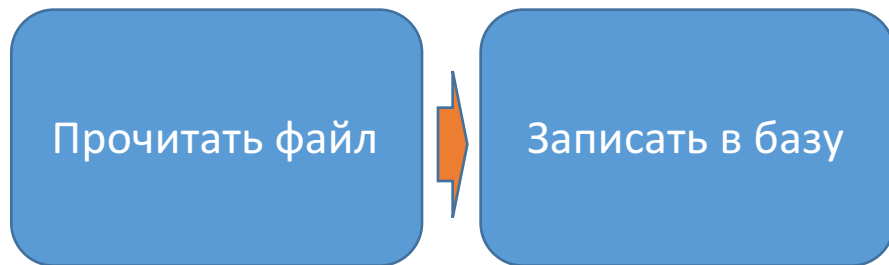
Partition

Реально можно ускориться!

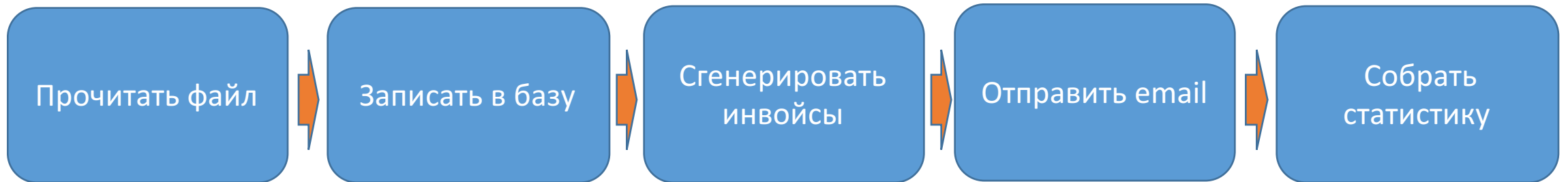
Flow/Split



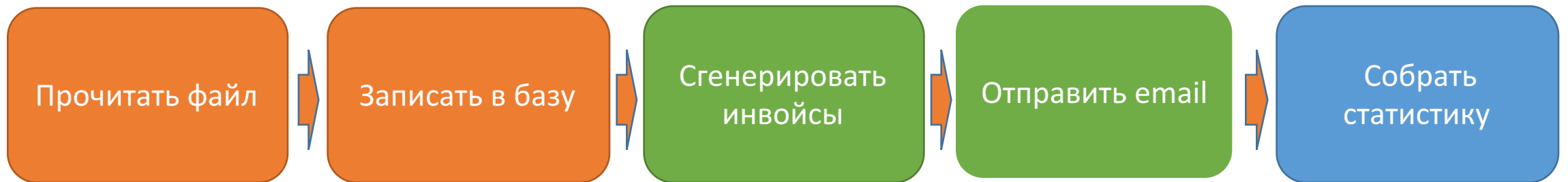
Постановка задачи



Постановка задачи



Постановка задачи



Flow

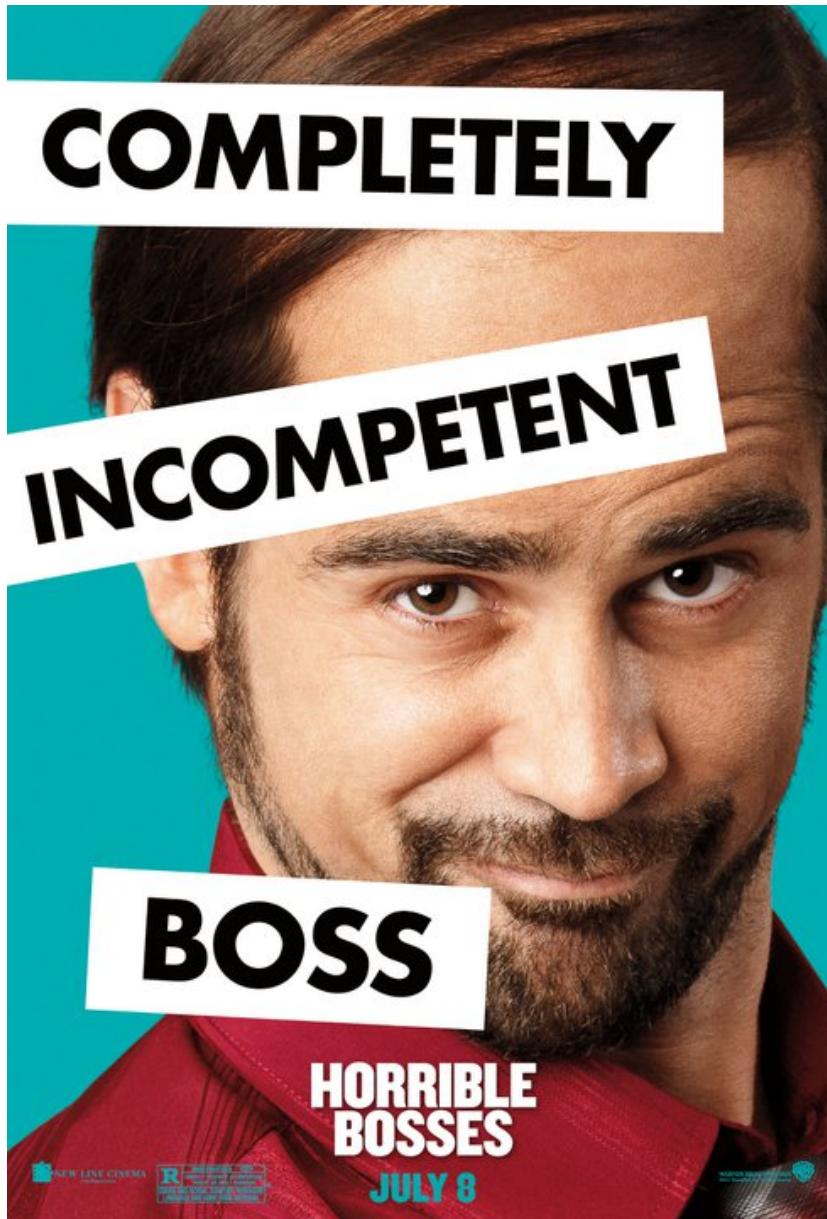
- Это касается бизнес процессов

Flow

- Это касается бизнес процессов
- Логическое разделение процессов

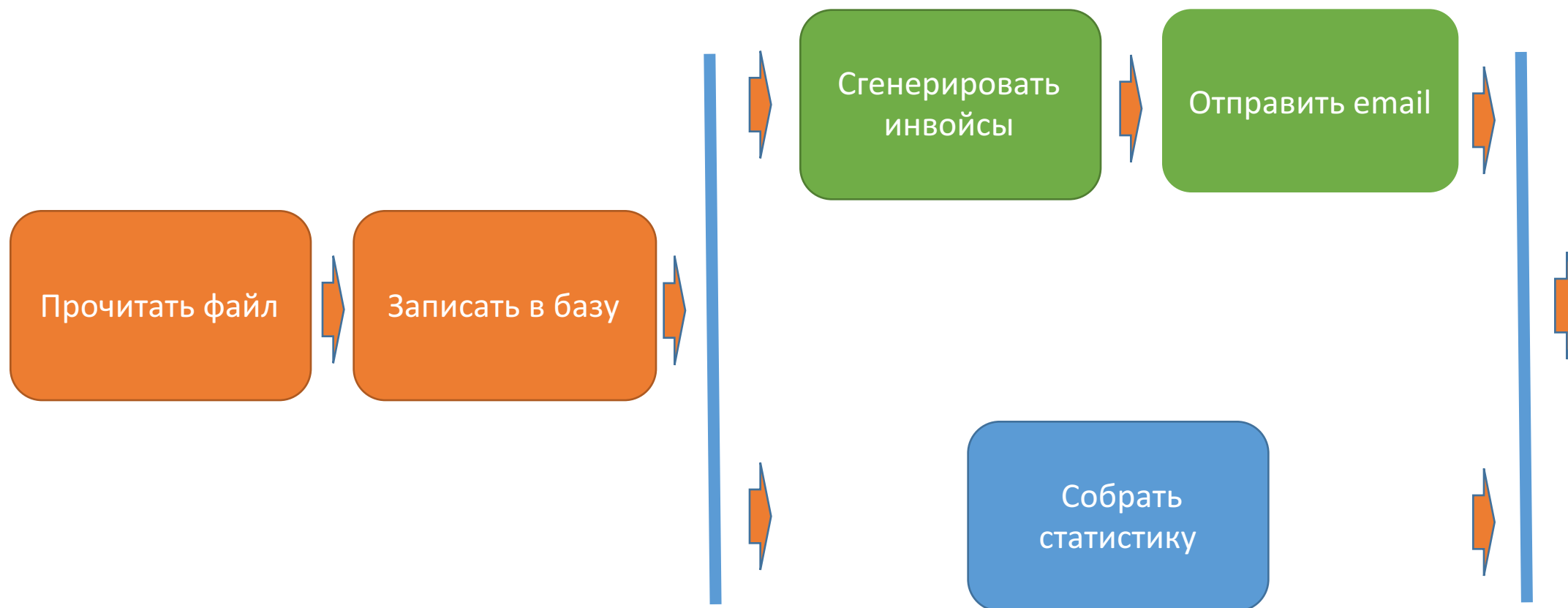
Flow

- Это касается бизнес процессов
- Логическое разделение процессов
- Атомарная последовательность шагов



Joker<?>

Постановка задачи



Split

- Параллельное исполнение нескольких Flow

Split

- Параллельное исполнение нескольких Flow
- Не путать с Partition

Split

- Параллельное исполнение нескольких Flow
- Не путать с Partition
- Внутри могут быть только Flow

Split

```
<split id="mySplit">  
  <flow id="flow1">  
    <step id="myChunk" next="myBatchlet">...</step>  
    <step id="myBatchlet">...</step>  
  </flow>  
  <flow id="flow2">  
    <step id="otherChunk" next="otherBatchlet">...</step>  
    <step id="otherBatchlet">...</step>  
  </flow>  
</split>
```

Joker<?>

А в чем собственно разница?

- Partition это про данные
- Flow/Split это про бизнес процессы

Decision



Joker<?>

Decision

- Решать что делать дальше
- Применяется для Step, Flow и Split
- По сути это if/else
- Необходимо имплементировать

Decision

Оркестрация

```
<step id="myStep" next="myDecider">  
  <batchlet ref="myBatchlet"/>  
</step>  
<decision id="myDecider" ref="MyDecider">  
  <next on="foo" to="myFooStep"/>  
  <next on="bar" to="myBarStep"/>  
</decision>
```

Joker<?>

Decision

Имплементация

```
public class MyDecider implements Decider {  
  
    public String decide(StepExecution[] executions)  
        throws Exception {  
        return "decision";  
    }  
}
```



Метрики

- `readCount` – сколько успешно прочитали.
- `writeCount` – сколько успешно записали.
- `filterCount` – сколько отфильтровано `ItemProcessor`.
- `commitCount` – сколько транзакций закомичено.
- `rollbackCount` – сколько транзакций заролбечино.
- `readSkipCount` – сколько «skippable» исключений кинуто `ItemReader`.
- `processSkipCount` – сколько «skippable» исключений кинуто `ItemProcessor`.
- `writeSkipCount` – сколько «skippable» исключений кинуто `ItemWriter`.

Транзакционность

- JBatch должен работать как в SE так и в EE
- Глобальный режим транзакций в режиме EE
 - Коммит происходит при записи чанка
- Локальный режим транзакций в режиме SE
 - По таймауту

А там что-то про шедюлер?
Или скеджулер?

Ну так вот..

его нет.



demotivation.me

ДА. УЖ.

Но не надо огорчаться

- Это просто не в компетенции JSR-352

Но не надо огорчаться

- Это просто не в компетенции JSR-352
- В среде EE решается обыкновенным @Scheduler

Но не надо огорчаться

@Singleton

```
public class BatchJobRunner {  
    @Schedule(dayOfWeek = "Sun")  
    public void scheduleJob() {  
        JobOperator jobOperator =  
            BatchRuntime.getJobOperator();  
        jobOperator.start("myJob", new Properties());  
    }  
}
```

Joker<?>

Но не надо огорчаться

- Это просто не в компетенции JSR-352
- В среде EE решается обыкновенным @Scheduler
- В среде SE можно прикрутить например Quartz*

**<http://www.quartz-scheduler.org/>*

И еще ложки дегтя

- Все еще мало стандартных Readers/Writers

И еще ложки дегтя

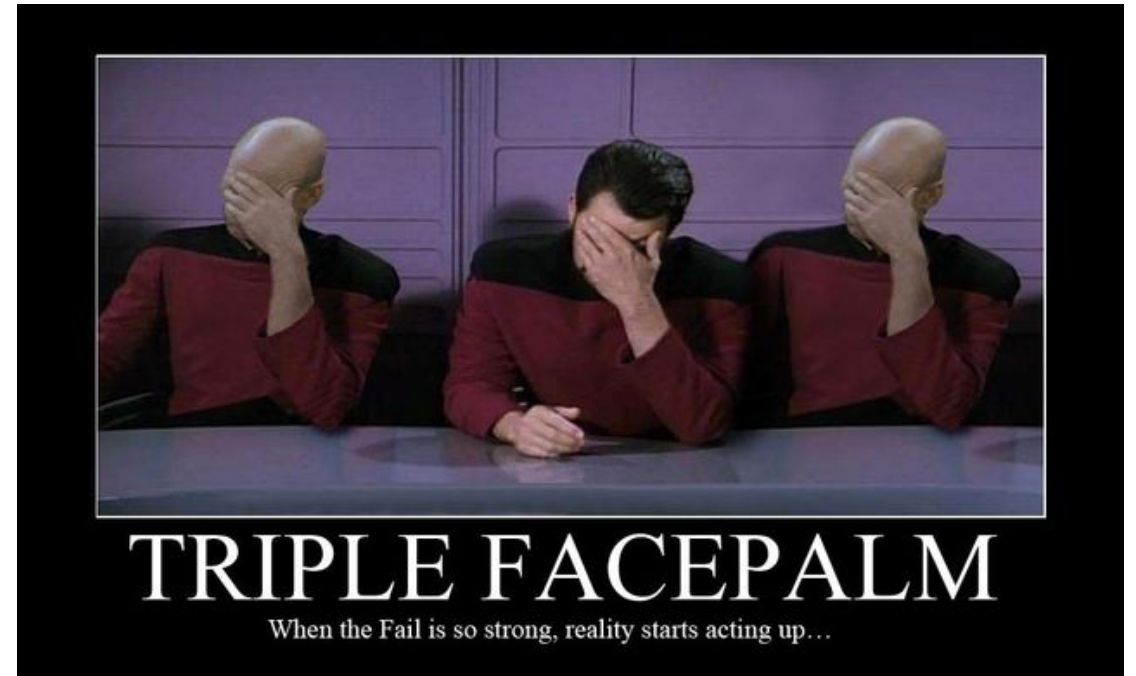
- Все еще мало стандартных Readers/Writers
- Без Generics

И еще ложки дегтя

- Все еще мало стандартных Readers/Writers
- Без Generics
- Только xml конфигурация

И еще ложки дегтя

- Все еще мало стандартных Readers/Writers
- Без Generics
- Только xml оркестрация



Но ведь есть другие батч
движки?

Известные имплементации JSR-352

- JBatch IBM (Glassfish, JEUS)
- JBeret (Wildfly)
- BatchEE
- ну и конечно же Spring Batch*

**(не совсем)*

Spring Batch

- Первый стабильный релиз 2009
- Послужило прототипом для JSR-352
- Сильная интеграция в мире Spring

Spring Batch

Spring Batch	JSR-352
Job	Job
Step	Step
Chunk	Chunk
Item	Item
ItemReader/ItemStream	ItemReader
ItemProcessor	ItemProcessor
ItemWriter/ItemStream	ItemWriter
JobInstance	JobInstance
JobExecution	JobExecution
StepExecution	StepExecution
JobExecutionListener	JobListener
StepExecutionListener	StepListener
Listeners	Listeners

Spring Batch



Spring Batch

Spring Batch:

```
<job id="myJob">
  <step id="myStep">
    <tasklet>
      <chunk
        reader="reader"
        writer="writer"
        processor="processor"
        commit-interval="10"/>
    </tasklet>
  </step>
</job>
```

JSR-352:

```
<job id="myJob">
  <step id="myStep" >
    <chunk item-count="2">
      <reader ref="reader"/>
      <processor ref="processor"/>
      <writer ref="writer"/>
    </chunk>
  </step>
</job>
```

Joker<?>

Spring Batch

С версии 3.0 оркестрацию можно вести в формате JSR-352
😊

EasyBatch

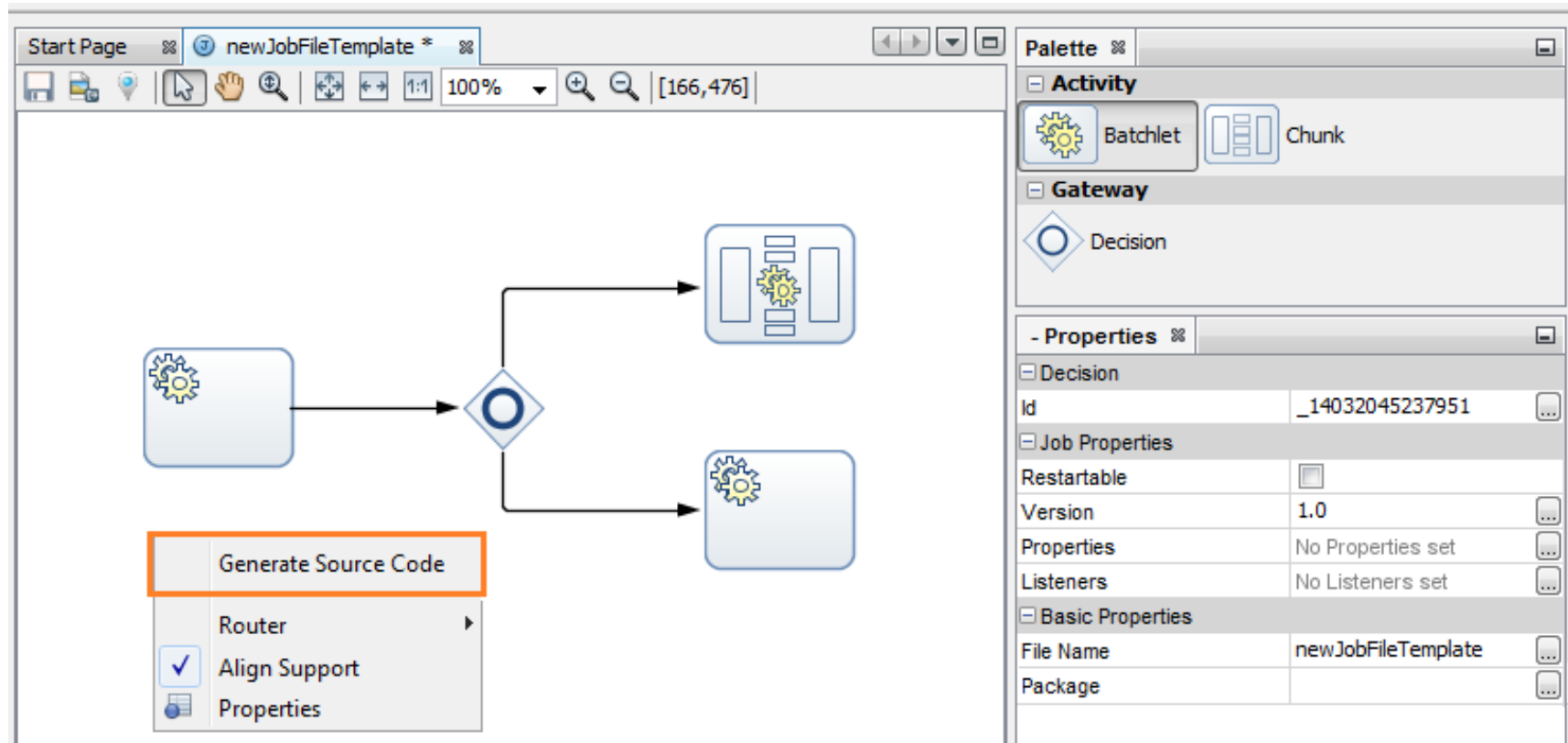
- Относительно молодой проект
- Не стандартизованный
- Позиционируется как очень простая и легко воспринимаемая альтернатива
- Чем-то напоминает стримы
- <https://github.com/EasyBatch/easybatch-framework>

EasyBatch

```
public class EasyBatchHelloWorldLauncher {  
  
    public static void main(String[] args) throws Exception {  
        JobBuilder.aNewJob()  
            .reader(new FlatFileRecordReader(new File("tweets.csv")))  
            .filter(new HeaderRecordFilter())  
            .mapper(new DelimitedRecordMapper(Tweet.class, "id", "user", "message"))  
            .processor(new TweetProcessor())  
            .call();  
    }  
}  
  
public class TweetProcessor implements RecordProcessor<Record<Tweet>, Record<Tweet>> {  
    @Override  
    public Record<Tweet> processRecord(Record<Tweet> record) {  
        System.out.println(record.getPayload());  
        return record;  
    }  
}
```

В ПОМОЩЬ

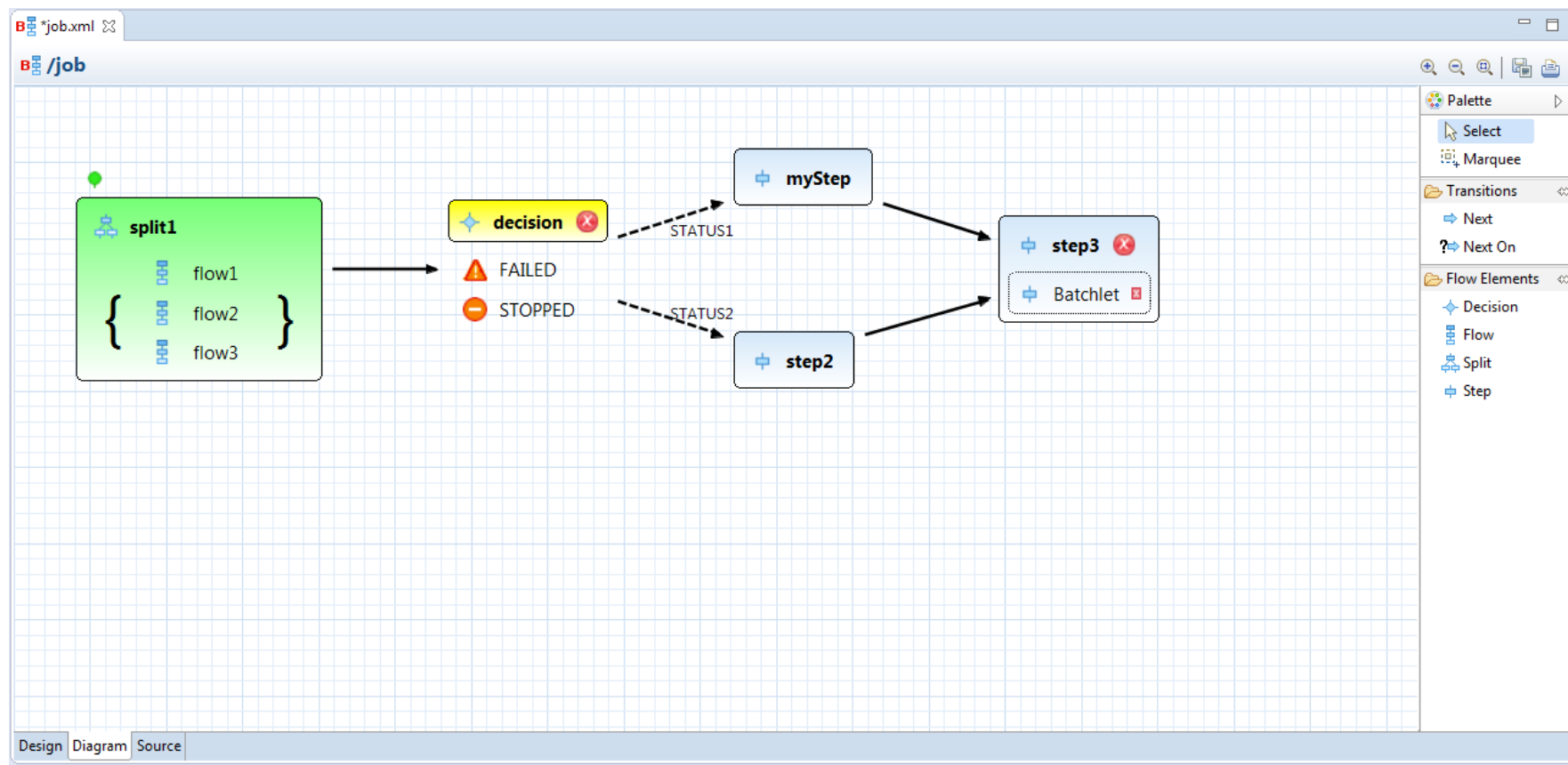
JBatch suite



Joker<?>

В ПОМОЩЬ

JBoss tools



Joker<?>

Выводы

Выводы

- Не стоит недооценивать данный класс задач

Выводы

- Не стоит недооценивать данный класс задач
- Особенно в энтерпрайзе

Выводы

- Не стоит недооценивать данный класс задач
- Особенно в энтерпрайзе
- Почти всегда требования будут усложняться

Выводы

- Не стоит недооценивать данный класс задач
- Особенно в энтерпрайзе
- Почти всегда требования будут усложняться
- Уже почти все придумано! Бери и пользуйся!



Можно уже не (так сильно) велосипедить!

Вопросы есть?



JOKER<?>

ЧТИВО для досуга

- <https://jcp.org/en/jsr/detail?id=352>
- <http://projects.spring.io/spring-batch/>
- <http://www.easybatch.org/>