

Spring Batch Workshop

Hello World

- Problem: getting started with Spring Batch
- Solution: writing a simple “Hello world” job

Hello World

- A Spring Batch job is made of steps
- The Hello World job has one step
- The processing is implemented in a *Tasklet*

Hello World

- The Hello World Tasklet

```
public class HelloWorldTasklet implements Tasklet {  
  
    @Override  
    public RepeatStatus execute(  
        StepContribution contribution,  
        ChunkContext chunkContext) throws Exception {  
        System.out.println("Hello world!");  
        return RepeatStatus.FINISHED;  
    }  
}
```

Hello World

- The configuration of the Hello World job
 - Notice the <batch /> namespace

```
<batch:job id="helloWorldJob">  
  <batch:step id="helloWorldStep">  
    <batch:tasklet>  
      <bean class="com.zenika.workshop.springbatch.HelloWorldTasklet" />  
    </batch:tasklet>  
  </batch:step>  
</batch:job>
```

Hello World

- Spring Batch needs some infrastructure beans
 - Let's use the typical test configuration

```
<bean id="transactionManager"
      class="o.s.b.support.transaction.ResourcelessTransactionManager" />

<bean id="jobRepository"
      class="o.s.b.core.repository.support.MapJobRepositoryFactoryBean" />

<bean id="jobLauncher"
      class="o.s.b.core.launch.support.SimpleJobLauncher">
  <property name="jobRepository" ref="jobRepository" />
</bean>
```

Hello World

- Let's test!

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/hello-world-job.xml")
public class HelloWorldJobTest {

    @Autowired
    private Job job;

    @Autowired
    private JobLauncher jobLauncher;

    @Test public void helloWorld() throws Exception {
        JobExecution execution = jobLauncher.run(job, new JobParameters());
        assertEquals(ExitStatus.COMPLETED, execution.getExitStatus());
    }
}
```

Flat file reading

- Problem: reading lines from a flat file and sending them to another source (e.g. database)
- Solution: using the FlatFileItemReader


Flat file reading

- Spring Batch has built-in support for flat files
 - Through the FlatFileItemReader for reading
- The FlatFileItemReader handles I/O
- 2 main steps:
 - Configuring the FlatFileItemReader
 - Providing a line – object mapping strategy

Flat file reading

- The usual suspects:

```
De-Anna,Raghunath,2010-03-04  
Susy,Hauerstock,2010-03-04  
Kiam,Whitehurst,2010-03-04  
Alecia, Van Holst,2010-03-04  
Hing,Senecal,2010-03-04
```



```
public class Contact {  
  
    private Long id;  
    private String firstname,lastname;  
    private Date birth;  
  
    (...)  
}
```

Flat file reading

- What do we need to read a flat file?
 - How to tokenize a line
 - How to map the line with a Java object
 - Where to find the file to read

Flat file reading

Tokenization

```
<bean id="reader"
      class="org.springframework.batch.item.file.FlatFileItemReader">
  <property name="lineMapper">
    <bean class="org.springframework.batch.item.file.mapping.DefaultLineMapper">
      <property name="lineTokenizer">
        <bean
          class="org.springframework.batch.item.file.transform.DelimitedLineTokenizer">
            <property name="names" value="firstname,lastname,birth" />
          </bean>
        </property>
        <property name="fieldSetMapper">
          <bean class="com.zenika.workshop.springbatch.ContactFieldSetMapper" />
        </property>
      </bean>
    </property>
    <property name="resource" value="classpath:contacts.csv" />
  </bean>
```

File to read

Line – object mapping

Flat file reading

- A FieldSetMapper to map a line with an object
- More about business logic, so typically implemented by developer
 - Spring Batch provides simple implementations

Flat file reading

```
package com.zenika.workshop.springbatch;

import org.springframework.batch.item.file.mapping.FieldSetMapper;
import org.springframework.batch.item.file.transform.FieldSet;
import org.springframework.validation.BindException;

public class ContactFieldSetMapper implements FieldSetMapper<Contact> {

    @Override
    public Contact mapFieldSet(FieldSet fieldSet) throws BindException {
        return new Contact(
            fieldSet.readString("firstname"),
            fieldSet.readString("lastname"),
            fieldSet.readDate("birth", "yyyy-MM-dd")
        );
    }
}
```