

# Spring Batch Workshop

Arnaud Cogoluègnes

Consultant with Zenika, Co-author Spring Batch in Action

May 23, 2011

# Overview

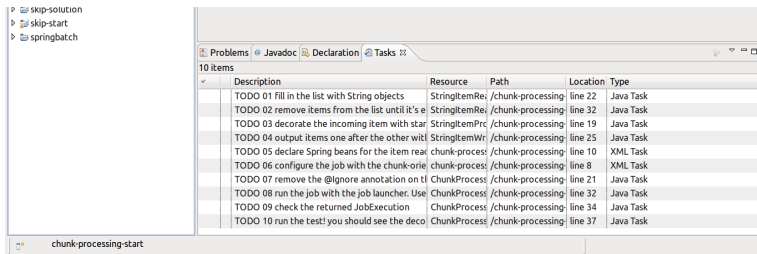
- ▶ This workshop highlights Spring Batch features
- ▶ Problem/solution approach
  - ▶ A few slides to cover the feature
  - ▶ A project to start from, just follow the TODOs
- ▶ Prerequisites
  - ▶ Basics about Java and Java EE
  - ▶ Spring: dependency injection, enterprise support
- ▶ <https://github.com/acogoluegnes/Spring-Batch-Workshop>

# License

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

## Follow the TODOs

- ▶ Track the TODO in the \*-start projects!
- ▶ It's easier with support from the IDE

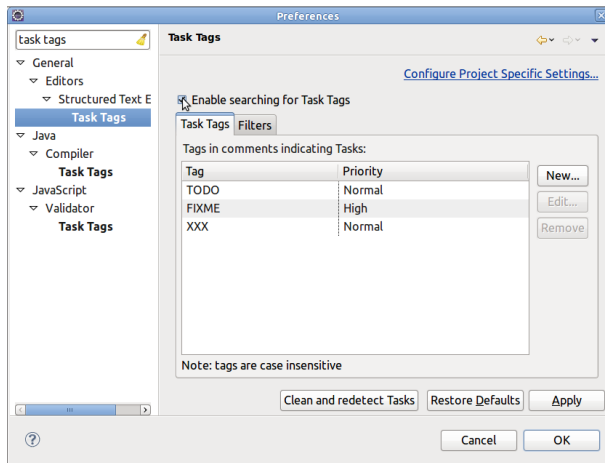


The screenshot shows an IDE window with a project explorer on the left containing 'skip-solution', 'skip-start', and 'springbatch'. The main editor area displays the 'Tasks' tab, which lists 10 items. The table below represents the content of this tab.

	Description	Resource	Path	Location	Type
▼	10 items				
	TODO 01 fill in the list with String objects	StringItemRe	/chunk-processing	line 22	Java Task
	TODO 02 remove items from the list until it's e	StringItemRe	/chunk-processing	line 32	Java Task
	TODO 03 decorate the incoming item with star	StringItemProc	/chunk-processing	line 19	Java Task
	TODO 04 output items one after the other with	StringItemWr	/chunk-processing	line 25	Java Task
	TODO 05 declare Spring beans for the item read	chunk-proces	/chunk-processing	line 10	XML Task
	TODO 06 configure the job with the chunk-ori	chunk-proces	/chunk-processing	line 8	XML Task
	TODO 07 remove the @Ignore annotation on t	ChunkProcess	/chunk-processing	line 21	Java Task
	TODO 08 run the job with the job launcher. Use	ChunkProcess	/chunk-processing	line 32	Java Task
	TODO 09 check the returned JobExecution	ChunkProcess	/chunk-processing	line 34	Java Task
	TODO 10 run the test! you should see the deco	ChunkProcess	/chunk-processing	line 37	Java Task

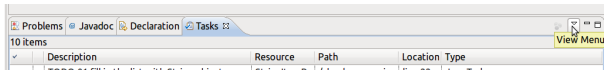
# TODO with Eclipse

- ▶ Window > Preferences > “tasks tag” in filter



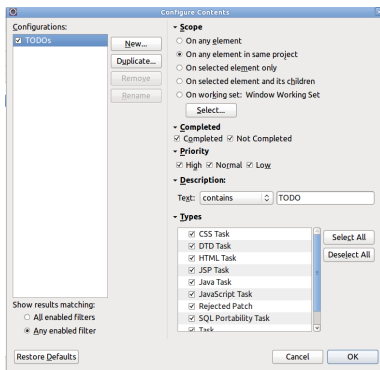
## TODO with Eclipse

- ▶ Open the “Tasks” view
- ▶ click on the down arrow on the right
- ▶ “configure contents”



## TODO with Eclipse

- ▶ Check “TODOs” on the left
- ▶ Check “On any element in the same project” on the right (scope)



## Spring support in IDE is a +

- ▶ e.g. code completion in SpringSource Tool Suite

```
<!-- TODO 03 configure the job with a chunk-oriented step using the reader and the writer -->  
  
<!-- TODO 01 configure the FlatFileItemReader -->  
<bean id="reader" class="FlatFileItemReader"
```



```
<!-- TODO 03 configure the job with a chunk-oriented step using the reader and the writer -->  
  
<!-- TODO 01 configure the FlatFileItemReader -->  
<bean id="reader" class="org.springframework.batch.item.file.FlatFileItemReader"
```



## Basic features for batch applications

- ▶ Read – process – write large amounts of data, efficiently
- ▶ Ready-to-use components to read from/write to
  - ▶ Flat/XML files
  - ▶ Databases (JDBC, Hibernate, JPA, iBatis)
  - ▶ JMS queues
  - ▶ Emails
- ▶ Numerous extension points/hooks

## Advanced features for batch applications

- ▶ Configuration to skip/retry items
- ▶ Execution metadata
  - ▶ Monitoring
  - ▶ Restart after failure
- ▶ Scaling strategies
  - ▶ Local/remote
  - ▶ Partitioning, remote processing

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

# Structure of a job

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

# 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;  
    }  
}
```

# The configuration of the Hello World job

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

## 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>
```

## Running the test in a JUnit 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());
    }
}
```