# Creating a Spring Batch Application

**Michael Hoffman**
DEVELOPER AND PLURALSIGHT AUTHOR

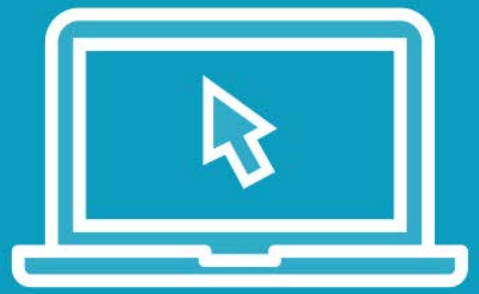
@mhi_inc  github.com/michaelhoffmantech

```
dependencies {

    // Starter to include the Spring Batch
    // dependencies via Spring Boot
    compile "org.springframework.boot:spring-boot-starter-batch"

    // Spring Batch testing support
    testCompile "org.springframework.batch:spring-batch-test"

}
```

# Build.gradle

**Found in root folder of the project**

# Demo



DEMOS.md file in project root directory

Demo 2 – Gradle dependencies

```
@Component
@EnableBatchProcessing
public class BatchConfiguration implements BatchConfigurer {

    private JobRepository jobRepository;
    private JobExplorer jobExplorer;
    private JobLauncher jobLauncher;

}
```

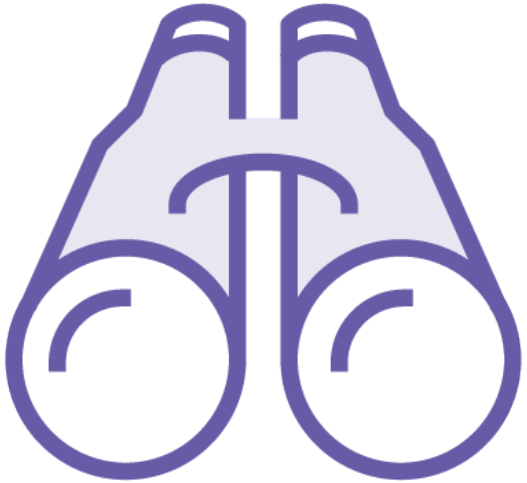
# BatchConfiguration.java

**New class to add in the package**

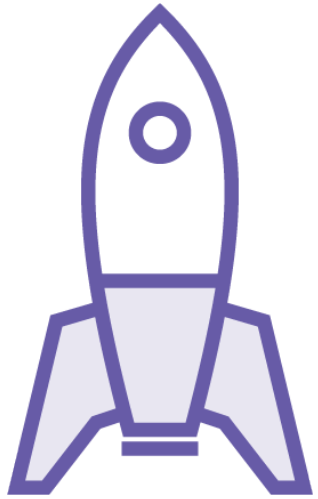
**com.pluralsight.springbatch.patientbatchloader.config**

# Batch Configuration Features

**JobRepository**
Persists meta-data about batch jobs

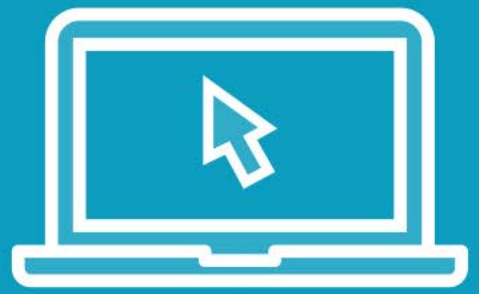


**JobExplorer**
Retrieves meta-data from the JobRepository



**JobLauncher**
Runs jobs with given parameters

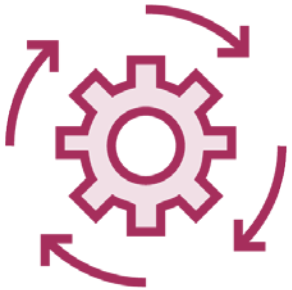
# Demo



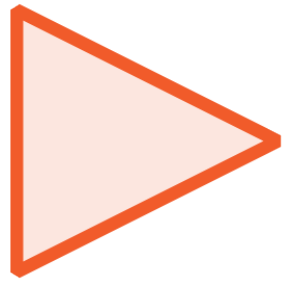
## Demo 3 – Spring Batch configuration

# Spring Batch Terms

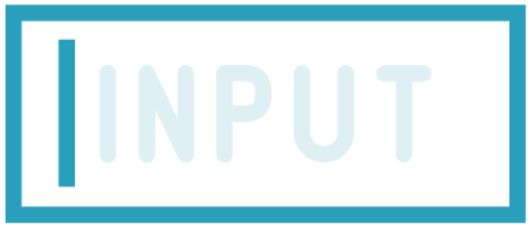
Job



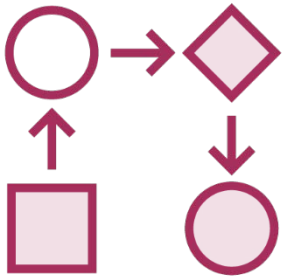
Job Instance



Job Execution




Job Parameters



Step

Step Execution

# JobRepository Database Tables

Batch_Job_Instance

Batch_Job_Execution

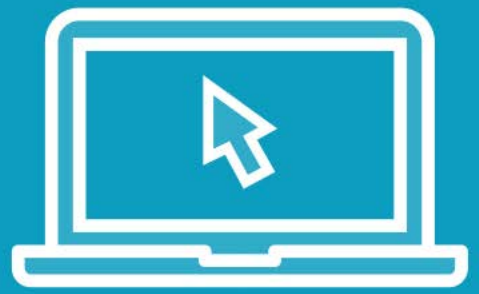
Batch_Job_Execution_Context

Batch_Job_Execution_Params

Batch_Step_Execution

Batch_Step_Execution_Context

# Demo



## Demo 4 – Spring Batch job repository database schema

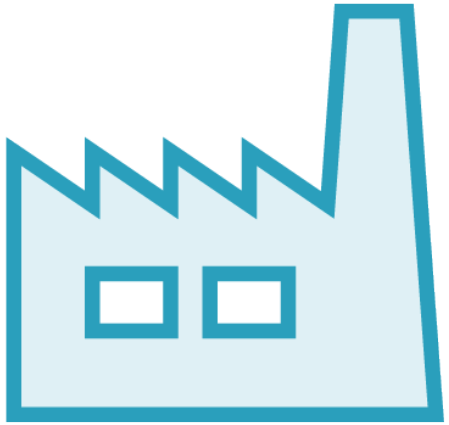
```
@Configuration
public class BatchJobConfiguration {

    @Autowired
    private JobBuilderFactory jobBuilderFactory;

    …

}
```

# BatchJobConfiguration.java

**New class to add in the package**

**com.pluralsight.springbatch.patientbatchloader.config**

# Job Configuration Classes



**JobBuilderFactory**

Factory for getting the type of builder required for job configuration

**SimpleJobBuilder**

Provides a DSL for defining how the job is constructed

```
public class BatchJobConfiguration {

    @Bean
    public Job job(Step step) throws Exception {
        return this.jobBuilderFactory
            .get(Constants.JOB_NAME)
            .validator(validator())
            .start(step)
            .build();
    }
}
```

# BatchJobConfiguration.java

**New class to add in the package**

**com.pluralsight.springbatch.patientbatchloader.config**

# JobParametersValidator

```
@Bean
public JobParametersValidator validator() {
  return new JobParametersValidator() {
    @Override public void validate(JobParameters parameters) throws JobParametersInvalidException {
      String fileName = parameters.getString(Constants.JOB_PARAM_FILE_NAME);
      if (StringUtils.isBlank(fileName)) { throw new JobParametersInvalidException(…); }
      try {
        Path file = Paths.get(applicationProperties.getBatch().getInputPath() +
          File.separator + fileName);
        if (Files.notExists(file) || !Files.isReadable(file)) { throw new Exception(…); }
      } catch (Exception e) { … }
    }
  };
}
```

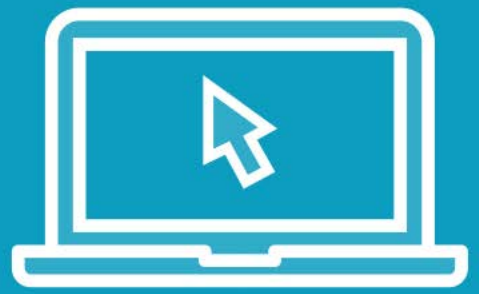
# BatchJobConfiguration.java

**New class to add in the package**

**com.pluralsight.springbatch.patientbatchloader.config**

# JobRegistryBeanPostProcessor

# Demo



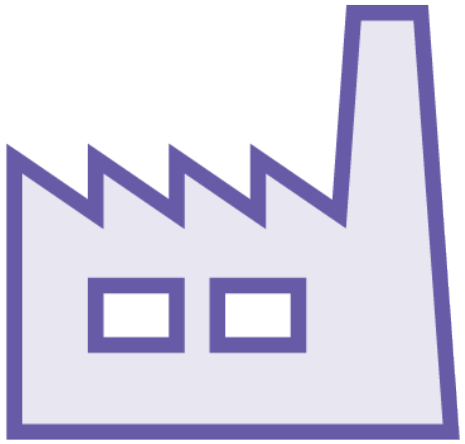
## Demo 5 – Spring Batch job configuration

```
@Configuration
public class BatchJobConfiguration {

    @Autowired
    private StepBuilderFactory stepBuilderFactory;

    …

}
```

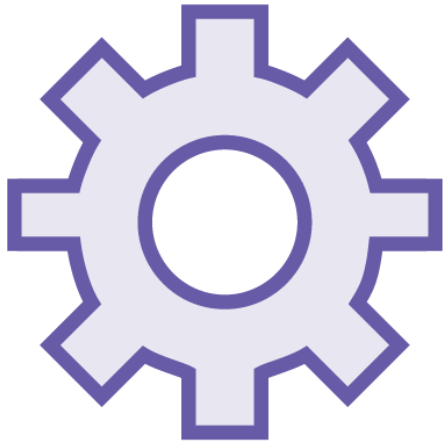
# BatchJobConfiguration.java

**Package com.pluralsight.springbatch.patientbatchloader.config**

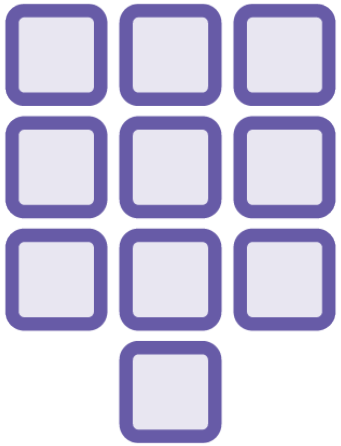
# Step Configuration



**StepBuilderFactory**
Factory for getting the type of builder required for step configuration



**Tasklet**
Defines what task the step will actually perform



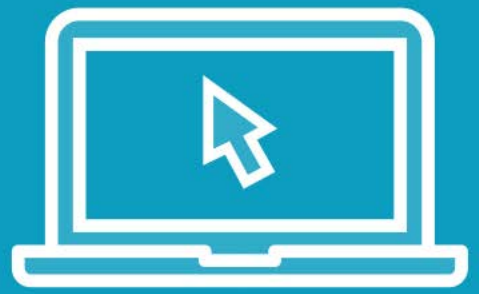
**Chunking**
Data is grouped in chunks to be output to an endpoint

```
@Bean
public Step step() throws Exception {
    return this.stepBuilderFactory
        .get(Constants.STEP_NAME)
        .tasklet(new Tasklet() {
            @Override public RepeatStatus execute(
                StepContribution contribution, ChunkContext chunkContext)
                throws Exception {
                    System.err.println("Hello World!");
                    return RepeatStatus.FINISHED;
                }
            })
            .build();
}
```

# BatchJobConfiguration.java

**Package com.pluralsight.springbatch.patientbatchloader.config**

# Demo



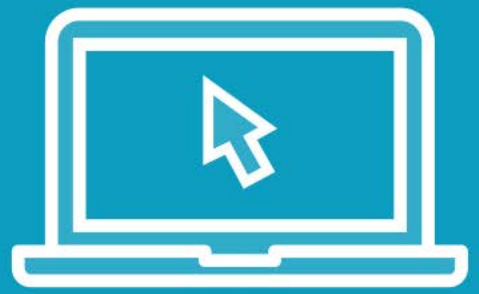
## Demo 6 – Spring Batch step configuration

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = PatientBatchLoaderApp.class)
@ActiveProfiles("dev")
public class BatchJobConfigurationTest {

    @Autowired
    private Job job;

    @Test
    public void test() {
        assertNotNull(job);
        assertEquals(Constants.JOB_NAME, job.getName());
    }
}
```

# BatchJobConfigurationTest.java

**New class in the test folder under the package**
**com.pluralsight.springbatch.patientbatchloader.config**

# Demo



## Demo 7 – Spring Batch job unit test configuration

```
@RestController
@RequestMapping("/job")
public class JobResource {
     private final JobLauncher jobLauncher;
     private final Job job;

     public JobResource(JobLauncher jobLauncher, Job job) {
          this.jobLauncher = jobLauncher;
          this.job = job;
     }

     …
}
```

# JobResource.java

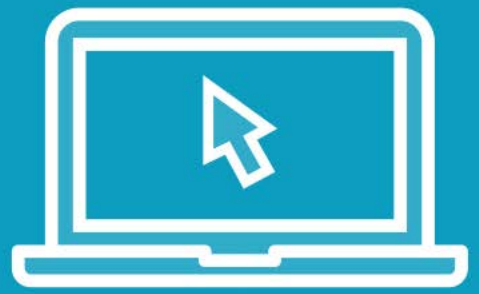
**New class in the package com.pluralsight.springbatch.patientbatchloader.web.rest**

```
@GetMapping("/{fileName:.+}")
public ResponseEntity<String> runJob(@PathVariable String fileName) {
     Map<String, JobParameter> parameterMap = new HashMap<>();
     parameterMap.put(Constants.JOB_PARAM_FILE_NAME, new JobParameter(fileName));
     try {
          jobLauncher.run(job, new JobParameters(parameterMap));
     } catch (Exception e) {
          return new ResponseEntity<String>("Failure: " + e.getMessage(),
               HttpStatus.INTERNAL_SERVER_ERROR);
     }
     return new ResponseEntity<String>("Success", HttpStatus.OK);
}
```

# JobResource.java

**New class in the package com.pluralsight.springbatch.patientbatchloader.web.rest**

# Demo



## Demo 8 – Spring Batch job execution

# Summary

- Added dependencies
- Configured Spring Batch features
- Configured a batch job and step
- Tested and executed the batch job