Xideral

Java Academy

Week 4-Day 3

Spring Batch

Presented by:

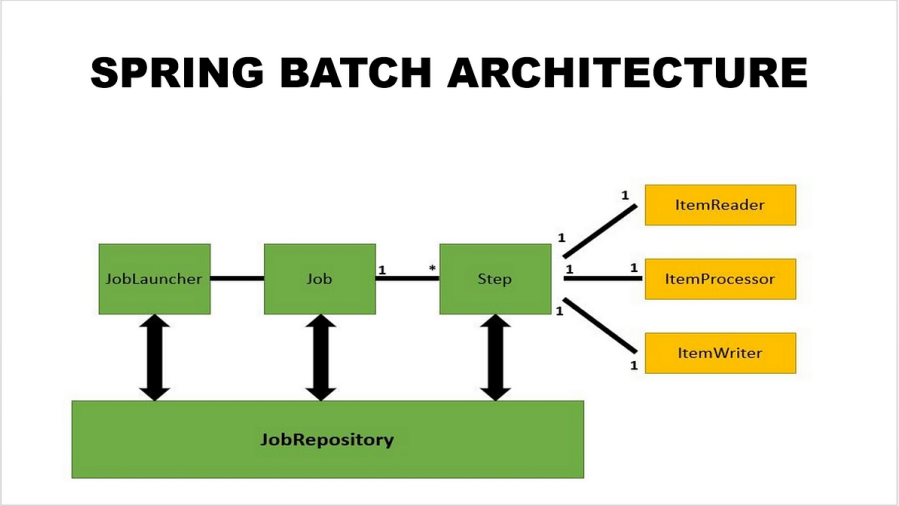
Edgar Itzak Sánchez Rogers

Monterrey Nuevo León México at 6 September 2024

Introduction:

Spring Batch is a lightweight, comprehensive batch framework designed to enable the development of robust batch applications vital for the daily operations of enterprise systems.

Spring Batch provides reusable functions that are essential in processing large volumes of records, including logging/tracing, transaction management, job processing statistics, job restart, skip, and resource management. It also provides more advanced technical services and features that will enable extremely high-volume and high-performance batch jobs through optimization and partitioning techniques.



A job contains 1 or more steps, each step contains a reader, a processor and a writer.

Java example:

The following is a business case where you have a .csv file with a list of 1000 organisation records. It is required to generate a job that filters the organisations that have as a line of business the hospitals and health.

Pom.xml

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-batch</artifactId>
        </dependency>
```

Application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/springbatchorganizations
spring.datasource.username = student
spring.datasource.password = student
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

spring.batch.initialize-schema=ALWAYS
#disabled job run at startup
spring.batch.job.enabled=false
```

Main class:

```java
package com.edgaritzak.springbatch;

import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing;

@SpringBootApplication
public class SpringBatchApplication {

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

Entity:

```java
package com.edgaritzak.springbatch.entity;

import javax.persistence.Column;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="ORGANIZATION_DATA")
public class Organization {

    @Id
    @Column(name="ID")
    private int Id;
    @Column(name="ORGANIZATION_ID")
    private String OrganizationId;
    @Column(name="NAME")
    private String Name;
    @Column(name="WEBSITE")
    private String Website;
    @Column(name="COUNTRY")
    private String Country;
    @Column(name="DESCRIPTION")
    private String Description;
    @Column(name="FOUNDED")
    private int Founded;
    @Column(name="INDUSTRY")
    private String Industry;
    @Column(name="NUMBER_OF_EMPLOYEES")
    private int NumberOfEmployees;
}
```

Repository:

```java
package com.edgaritzak.springbatch.repository;

import com.edgaritzak.springbatch.entity.Organization;

public interface OrganizationRepository extends JpaRepository<Organization,Integer> {

}
```

ItemPorcessor:

```java
package com.edgaritzak.springbatch.config;

import org.springframework.batch.item.ItemProcessor;

public class Processor implements ItemProcessor<Organization,Organization> {

    @Override
    public Organization process(Organization organization) throws Exception {
        if(organization.getIndustry().equals("Hospitality") ||
                organization.getIndustry().equals("Hospital / Health Care")) {
            return organization;
        }else{
            return null;
        }
    }
}
```

Spring Batch Config:

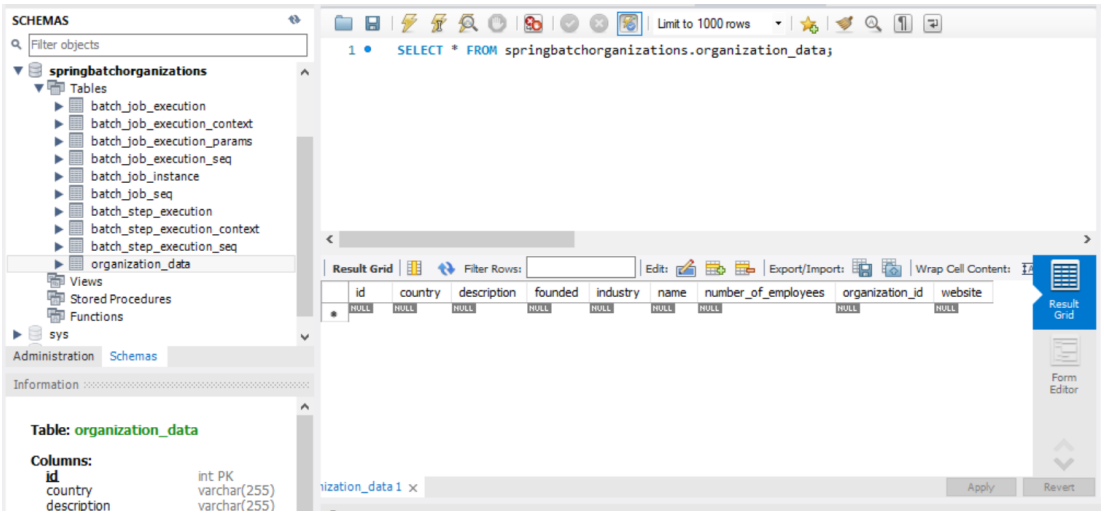
```java
package com.edgaritzak.springbatch.config;

import lombok.AllArgsConstructor;

@Configuration
@EnableBatchProcessing
@AllArgsConstructor
public class SpringBatchConfig {

    private JobBuilderFactory jobBuilderFactory;
    private StepBuilderFactory stepBuilderFactory;
    private OrganizationRepository customerRepository;


    @Bean
    public FlatFileItemReader<Organization> reader() {
        FlatFileItemReader<Organization> itemReader = new FlatFileItemReader<>();
        itemReader.setResource(new FileSystemResource("src/main/resources/organizations-10000.csv"));
        itemReader.setName("csvReader");
        itemReader.setLinesToSkip(1);
        itemReader.setLineMapper(lineMapper());
        return itemReader;
    }

    private LineMapper<Organization> lineMapper() {
        DefaultLineMapper<Organization> lineMapper = new DefaultLineMapper<>();

        DelimitedLineTokenizer lineTokenizer = new DelimitedLineTokenizer();
        lineTokenizer.setDelimiter(",");
        lineTokenizer.setStrict(false);
        lineTokenizer.setNames("Id","OrganizationId","Name","Website","Country","Description","Founded,

        BeanWrapperFieldSetMapper<Organization> fieldSetMapper = new BeanWrapperFieldSetMapper<>();
        fieldSetMapper.setTargetType(Organization.class);

        lineMapper.setLineTokenizer(lineTokenizer);
        lineMapper.setFieldSetMapper(fieldSetMapper);
        return lineMapper;

    }

    @Bean
    public Processor processor() {
        return new Processor();
    }

    @Bean
    public RepositoryItemWriter<Organization> writer() {
        RepositoryItemWriter<Organization> writer = new RepositoryItemWriter<>();
        writer.setRepository(customerRepository);
        writer.setMethodName("save");
        return writer;
    }

    @Bean
    public Step step1() {
        return stepBuilderFactory.get("csv-step").<Organization, Organization>chunk(10)
                .reader(reader())
                .processor(processor())
                .writer(writer())
                .taskExecutor(taskExecutor())
                .build();
    }

    @Bean
    public Job runJob() {
        return jobBuilderFactory.get("importOrganizations")
                .flow(step1()).end().build();
    }

    @Bean
    public TaskExecutor taskExecutor() {
        SimpleAsyncTaskExecutor asyncTaskExecutor = new SimpleAsyncTaskExecutor();
        asyncTaskExecutor.setConcurrencyLimit(10);
        return asyncTaskExecutor;
    }

}
```

REST Controller:

```java
package com.edgaritzak.springbatch.controller;

import org.springframework.batch.core.Job;

@RestController
@RequestMapping("/jobs")
public class JobController {

    @Autowired
    private JobLauncher jobLauncher;
    @Autowired
    private Job job;

    @PostMapping("/importOrganizations")
    public void importCsvToDBJob() {
        JobParameters jobParameters = new JobParametersBuilder()
                .addLong("startAt", System.currentTimeMillis()).toJobParameters();
        try {
            jobLauncher.run(job, jobParameters);
        } catch (JobExecutionAlreadyRunningException | JobRestartException | JobInstance
            e.printStackTrace();
        }
    }
}
```

Explanation:

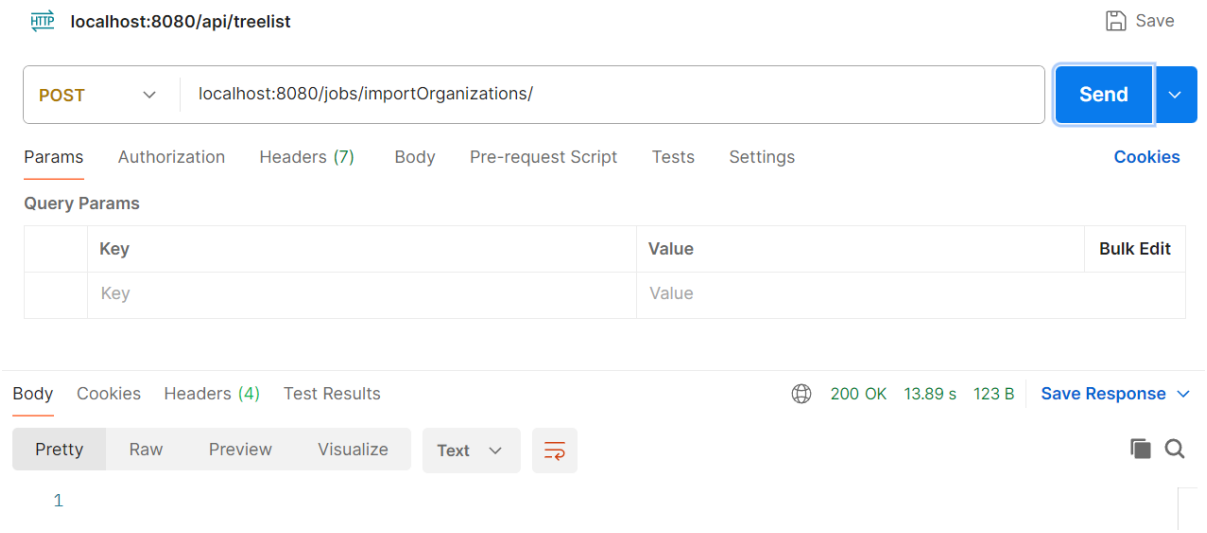
Application is launched and create spring batch creates new tables where it saves processing records



Also creates the entity (organization) table:

```
Hibernate: create table organization_data (id integer not null, country varchar(255), description varchar(255),
```

To execute the defined job is necessary to send a request to: localhost:8080/jobs/importOrganizations/ using the post method



Successful execution:

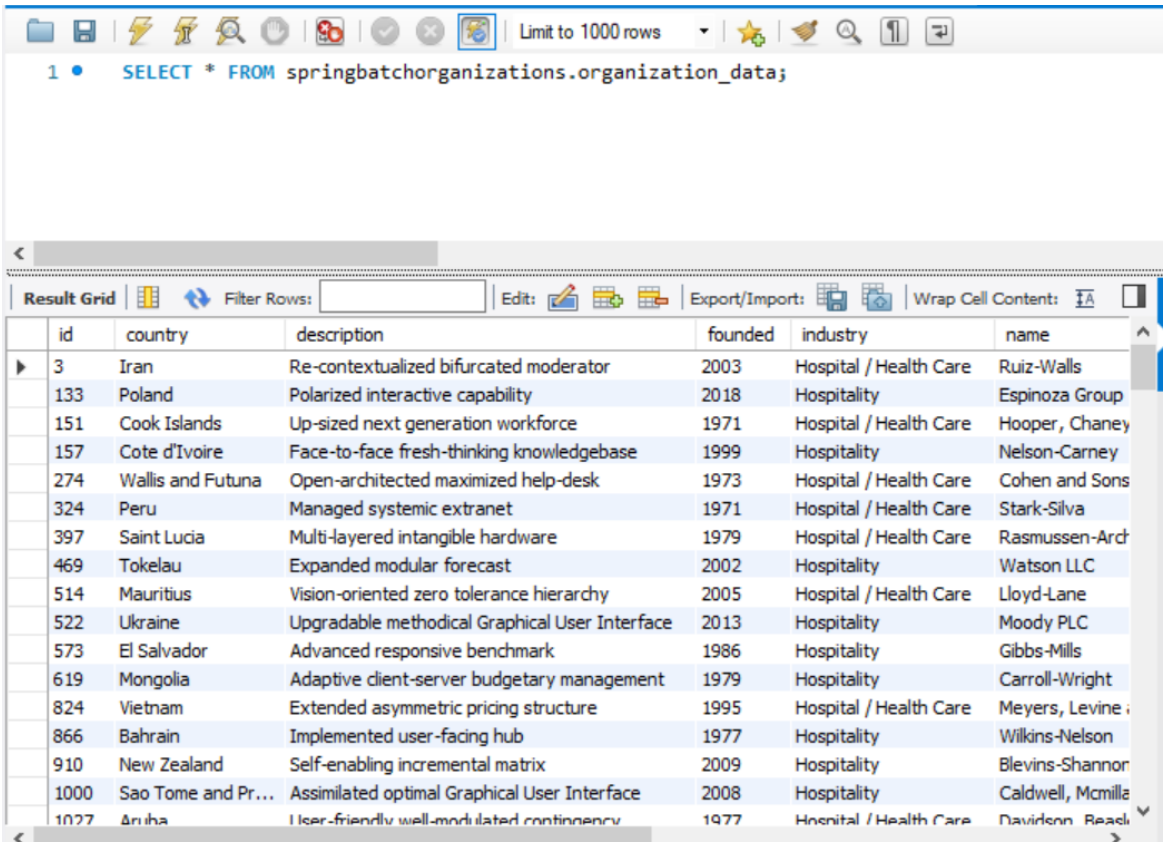
```
Hibernate: select organizati0_.id as id1_0_0_, organizati0_.country as country2_0_0_, organizati0_.description
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: select organizati0_.id as id1_0_0_, organizati0_.country as country2_0_0_, organizati0_.description
Hibernate: select organizati0_.id as id1_0_0_, organizati0_.country as country2_0_0_, organizati0_.description
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: select organizati0_.id as id1_0_0_, organizati0_.country as country2_0_0_, organizati0_.description
Hibernate: select organizati0_.id as id1_0_0_, organizati0_.country as country2_0_0_, organizati0_.description
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: select organizati0_.id as id1_0_0_, organizati0_.country as country2_0_0_, organizati0_.description
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
Hibernate: insert into organization_data (country, description, founded, industry, name, number_of_employees,
2024-09-06 13:32:48.160  INFO 14956 --- [nio-8080-exec-1] o.s.batch.core.step.AbstractStep         : Step: [cs
2024-09-06 13:32:48.186  INFO 14956 --- [nio-8080-exec-1] o.s.b.c.l.support.SimpleJobLauncher      : Job: [Flo
```

Batch processing results:

```
2024-09-06 13:32:48.160 INFO 14956 --- [nio-8080-exec-1]
o.s.batch.core.step.AbstractStep : Step: [csv-step] executed in 13s580ms
```

```
2024-09-06 13:32:48.186 INFO 14956 --- [nio-8080-exec-1]
o.s.b.c.l.support.SimpleJobLauncher : Job: [FlowJob: [name=importOrganizations]]
completed with the following parameters: [{startAt=1725651154400}] and the
following status: [COMPLETED] in 13s660ms
```

Filtered results in organization_data:



```
SELECT * FROM springbatchorganizations.organization_data;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	country	description	founded	industry	name
3	Iran	Re-contextualized bifurcated moderator	2003	Hospital / Health Care	Ruiz-Walls
133	Poland	Polarized interactive capability	2018	Hospitality	Espinoza Group
151	Cook Islands	Up-sized next generation workforce	1971	Hospital / Health Care	Hooper, Chaney
157	Cote d'Ivoire	Face-to-face fresh-thinking knowledgebase	1999	Hospitality	Nelson-Carney
274	Wallis and Futuna	Open-architected maximized help-desk	1973	Hospital / Health Care	Cohen and Sons
324	Peru	Managed systemic extranet	1971	Hospital / Health Care	Stark-Silva
397	Saint Lucia	Multi-layered intangible hardware	1979	Hospital / Health Care	Rasmussen-Arch
469	Tokelau	Expanded modular forecast	2002	Hospitality	Watson LLC
514	Mauritius	Vision-oriented zero tolerance hierarchy	2005	Hospital / Health Care	Lloyd-Lane
522	Ukraine	Upgradable methodical Graphical User Interface	2013	Hospitality	Moody PLC
573	El Salvador	Advanced responsive benchmark	1986	Hospitality	Gibbs-Mills
619	Mongolia	Adaptive client-server budgetary management	1979	Hospitality	Carroll-Wright
824	Vietnam	Extended asymmetric pricing structure	1995	Hospital / Health Care	Meyers, Levine
866	Bahrain	Implemented user-facing hub	1977	Hospitality	Wilkins-Nelson
910	New Zealand	Self-enabling incremental matrix	2009	Hospitality	Blevins-Shannon
1000	Sao Tome and Pr...	Assimilated optimal Graphical User Interface	2008	Hospitality	Caldwell, Mcmilla
1027	Aruba	User-friendly well-modulated contingency	1977	Hospital / Health Care	Davidson, Beasl

Jobs record:

STEP_EXECUTI	VERSION	STEP_NAME	JOB_EXECUTION_ID	START_TIME	END_TIME
1	2	csv-step	1	2024-09-06 13:21:05.556000	2024-09-06 13:21:05.651000
2	2	csv-step	2	2024-09-06 13:23:18.902000	2024-09-06 13:23:18.939000
3	2	csv-step	3	2024-09-06 13:25:12.797000	2024-09-06 13:25:12.881000
4	2	csv-step	4	2024-09-06 13:27:24.400000	2024-09-06 13:27:24.485000
5	2	csv-step	5	2024-09-06 13:27:55.265000	2024-09-06 13:27:55.353000
6	2	csv-step	6	2024-09-06 13:28:58.128000	2024-09-06 13:28:58.208000
7	1007	csv-step	7	2024-09-06 13:32:34.580000	2024-09-06 13:32:48.160000

Conclusion:

Spring Batch is essential for batch processing because it provides a robust and flexible framework for handling complex batch processing tasks. It allows you to handle large volumes of data efficiently, ensures consistency across transactions, and facilitates the implementation of retry and failover strategies. Its ability to define workflows, perform parallel processing and monitor job status makes it an essential tool for enterprise applications that require massive and reliable data processing.

References:

[1] Spring Batch

[2] Spring Batch Overview. Spring batch is a framework that allows... | by Ankitha Gowda | Medium

[3] Getting Started | Creating a Batch Service (spring.io)