Spring Batch Hello World Example

By mkyong (https://www.mkyong.com/author/mkyong/) | July 31, 2013 | Viewed : 250,283 times +1,755 pv/w

Spring Batch is a framework for batch processing – execution of a series of jobs. In Spring Batch, A job consists of many steps and each step consists of a `READ-PROCESS-WRITE` task or `single operation` task (tasklet).

1. For "READ-PROCESS-WRITE" process, it means "read" data from the resources (csv, xml or database), "process" it and "write" it to other resources (csv, xml and database). For example, a step may read data from a CSV file, process it and write it into the database. Spring Batch provides many made Classes to read/write CSV, XML and database.
2. For "single" operation task (tasklet), it means doing single task only, like clean up the resources after or before a step is started or completed.
3. And the steps can be chained together to run as a job.

```
1 Job = Many Steps.
1 Step = 1 READ-PROCESS-WRITE or 1 Tasklet.
Job = {Step 1 -> Step 2 -> Step 3} (Chained together)
```

Spring Batch Examples

Consider following batch jobs :

1. Step 1 – Read CSV files from folder A, process, write it to folder B. "READ-PROCESS-WRITE"
2. Step 2 – Read CSV files from folder B, process, write it to the database. "READ-PROCESS-WRITE"
3. Step 3 – Delete the CSB files from folder B. "Tasklet"
4. Step 4 – Read data from a database, process and generate statistic report in XML format, write it to folder C. "READ-PROCESS-WRITE"
5. Step 5 – Read the report and send it to manager email. "Tasklet"

In Spring Batch, we can declare like the following :

```
<job id="abcJob" xmlns="http://www.springframework.org/schema/batch">
<step id="step1" next="step2">
  <tasklet>
    <chunk reader="cvsItemReader" writer="cvsItemWriter"
                    processor="itemProcesser" commit-interval="1" />
  </tasklet>
</step>
<step id="step2" next="step3">
  <tasklet>
    <chunk reader="cvsItemReader" writer="databaseItemWriter"
                    processor="itemProcesser" commit-interval="1" />
  </tasklet>
</step>
<step id="step3" next="step4">
  <tasklet ref="fileDeletingTasklet" />
</step>
<step id="step4" next="step5">
  <tasklet>
    <chunk reader="databaseItemReader" writer="xmlItemWriter"
                    processor="itemProcesser" commit-interval="1" />
  </tasklet>
</step>
<step id="step5">
    <tasklet ref="sendingEmailTasklet" />
</step>
  </job>
```

The entire jobs and steps execution are stored in database, which make the failed step is able to restart at where it was failed, no need start over the entire job.

1. Tutorial

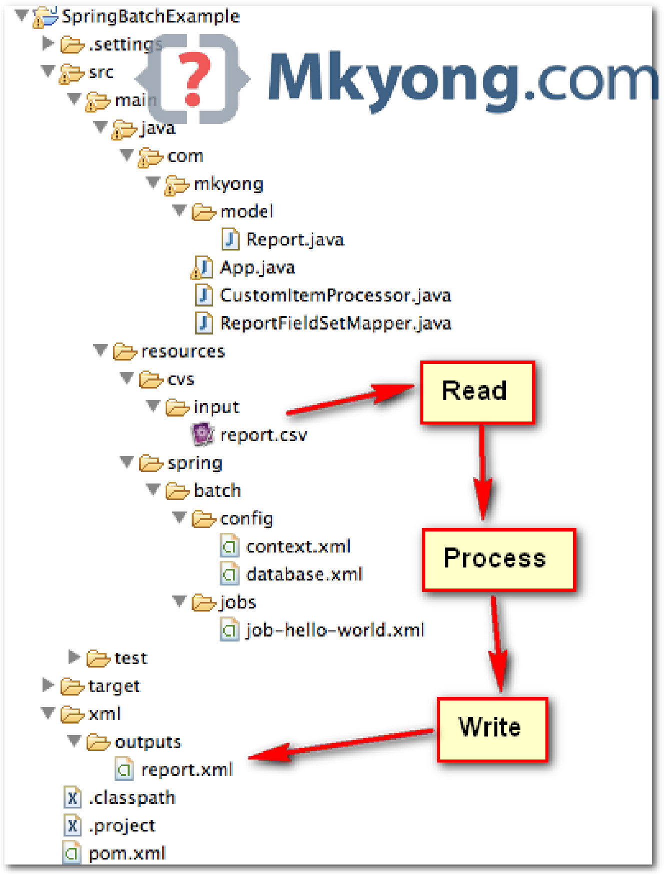
In this Spring Batch tutorial, we will show you how to create a job, read a CSV file, process it, write the output to an XML file.

Tools and libraries used

1. Maven 3
2. Eclipse 4.2
3. JDK 1.6
4. Spring Core 3.2.2.RELEASE
5. Spring OXM 3.2.2.RELEASE
6. Spring JDBC 3.2.2.RELEASE
7. Spring Batch 2.2.0.RELEASE

2. Project Directory

Review final project directory, a standard Maven project.



3. Project Dependencies

They must have dependencies are just Spring Core, Spring Batch and JDK 1.5. Read comments for self-explanatory.

```
pom.xml
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mkyong</groupId>
    <artifactId>SpringBatchExample</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>SpringBatchExample</name>
    <url>http://maven.apache.org</url>

    <properties>
        <jdk.version>1.6</jdk.version>
        <spring.version>3.2.2.RELEASE</spring.version>
        <spring.batch.version>2.2.0.RELEASE</spring.batch.version>
        <mysql.driver.version>5.1.25</mysql.driver.version>
        <junit.version>4.11</junit.version>
    </properties>

    <dependencies>

        <!-- Spring Core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- Spring jdbc, for database -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- Spring XML to/back object -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-oxm</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- MySQL database driver -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql.driver.version}</version>
        </dependency>

        <!-- Spring Batch dependencies -->
        <dependency>
            <groupId>org.springframework.batch</groupId>
            <artifactId>spring-batch-core</artifactId>
            <version>${spring.batch.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.batch</groupId>
            <artifactId>spring-batch-infrastructure</artifactId>
            <version>${spring.batch.version}</version>
        </dependency>
        <!-- Spring Batch unit test -->
        <dependency>
            <groupId>org.springframework.batch</groupId>
            <artifactId>spring-batch-test</artifactId>
            <version>${spring.batch.version}</version>
        </dependency>

        <!-- Junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>

    </dependencies>
    <build>
        <finalName>spring-batch</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-eclipse-plugin</artifactId>
                <version>2.9</version>
                <configuration>
                    <downloadSources>true</downloadSources>
                    <downloadJavadocs>false</downloadJavadocs>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
                <configuration>
                    <source>${jdk.version}</source>
                    <target>${jdk.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

4. Spring Batch Jobs

A CSV file.

`report.csv`

```
1001,"213,100",980,"mkyong", 29/7/2013
1002,"320,200",1080,"staff 1", 30/7/2013
1003,"342,197",1200,"staff 2", 31/7/2013
```

A Spring batch job, to read above csv file with `FlatFileItemReader`, process the data with `itemProcessor` and write it to an XML file
with `StaxEventItemWriter`.

`job-hello-world.xml`

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:batch="http://www.springframework.org/schema/batch"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/batch
        http://www.springframework.org/schema/batch/spring-batch-2.2.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    ">

    <import resource="../config/context.xml" />
    <import resource="../config/database.xml" />

    <bean id="report" class="com.mkyong.model.Report" scope="prototype" />
    <bean id="itemProcessor" class="com.mkyong.CustomItemProcessor" />

    <batch:job id="helloWorldJob">
      <batch:step id="step1">
        <batch:tasklet>
            <batch:chunk reader="cvsFileItemReader" writer="xmlItemWriter"
                          processor="itemProcessor" commit-interval="10">
            </batch:chunk>
        </batch:tasklet>
      </batch:step>
    </batch:job>

    <bean id="cvsFileItemReader" class="org.springframework.batch.item.file.FlatFileItemReader">

        <property name="resource" value="classpath:cvs/input/report.csv" />

        <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="id,sales,qty,staffName,date" />
                </bean>
            </property>
            <property name="fieldSetMapper">
                <bean class="com.mkyong.ReportFieldSetMapper" />

                 <!-- if no data type conversion, use BeanWrapperFieldSetMapper to map by name
                <bean
                    class="org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper">
                    <property name="prototypeBeanName" value="report" />
                </bean>
                 -->
            </property>
            </bean>
        </property>

    </bean>

    <bean id="xmlItemWriter" class="org.springframework.batch.item.xml.StaxEventItemWriter">
        <property name="resource" value="file:xml/outputs/report.xml" />
        <property name="marshaller" ref="reportMarshaller" />
        <property name="rootTagName" value="report" />
    </bean>

    <bean id="reportMarshaller" class="org.springframework.oxm.jaxb.Jaxb2Marshaller">
       <property name="classesToBeBound">
        <list>
            <value>com.mkyong.model.Report</value>
        </list>
        </property>
    </bean>

</beans>
```

Map CSV value to `Report` object and write it to XML file (via jaxb annotations).

```
Report.java
```

```
package com.mkyong.model;

import java.math.BigDecimal;
import java.util.Date;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "record")
public class Report {

    private int id;
    private BigDecimal sales;
    private int qty;
    private String staffName;
    private Date date;

    @XmlAttribute(name = "id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @XmlElement(name = "sales")
    public BigDecimal getSales() {
        return sales;
    }

    public void setSales(BigDecimal sales) {
        this.sales = sales;
    }

    @XmlElement(name = "qty")
    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }

    @XmlElement(name = "staffName")
    public String getStaffName() {
        return staffName;
    }

    public void setStaffName(String staffName) {
        this.staffName = staffName;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    @Override
    public String toString() {
        return "Report [id=" + id + ", sales=" + sales
                    + ", qty=" + qty + ", staffName=" + staffName + "]";
    }

}
```

To convert a `Date`, you need a custom `FieldSetMapper`. If no data type conversion, just use `BeanWrapperFieldSetMapper` to map the values by name automatically.

`ReportFieldSetMapper.java`

```
package com.mkyong;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import org.springframework.batch.item.file.mapping.FieldSetMapper;
import org.springframework.batch.item.file.transform.FieldSet;
import org.springframework.validation.BindException;

import com.mkyong.model.Report;

public class ReportFieldSetMapper implements FieldSetMapper<Report> {

    private SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");

    @Override
    public Report mapFieldSet(FieldSet fieldSet) throws BindException {

        Report report = new Report();
        report.setId(fieldSet.readInt(0));
        report.setSales(fieldSet.readBigDecimal(1));
        report.setQty(fieldSet.readInt(2));
        report.setStaffName(fieldSet.readString(3));

        //default format yyyy-MM-dd
        //fieldSet.readDate(4);
        String date = fieldSet.readString(4);
        try {
            report.setDate(dateFormat.parse(date));
        } catch (ParseException e) {
            e.printStackTrace();
        }

        return report;

    }

}
```

A itemProcessor will be fired before itemWriter.

`CustomItemProcessor.java`

```java
package com.mkyong;

import org.springframework.batch.item.ItemProcessor;
import com.mkyong.model.Report;

public class CustomItemProcessor implements ItemProcessor<Report, Report> {

    @Override
    public Report process(Report item) throws Exception {

        System.out.println("Processing..." + item);
        return item;
    }

}
```

Spring context and database configuration.

context.xml

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">

    <!-- stored job-meta in memory -->
    <!--
    <bean id="jobRepository"
        class="org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean">
        <property name="transactionManager" ref="transactionManager" />
    </bean>
     -->

     <!-- stored job-meta in database -->
    <bean id="jobRepository"
        class="org.springframework.batch.core.repository.support.JobRepositoryFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="transactionManager" ref="transactionManager" />
        <property name="databaseType" value="mysql" />
    </bean>

    <bean id="transactionManager"
        class="org.springframework.batch.support.transaction.ResourcelessTransactionManager" />

    <bean id="jobLauncher"
        class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
        <property name="jobRepository" ref="jobRepository" />
    </bean>

</beans>
```

database.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/jdbc
        http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd">

        <!-- connect to MySQL database -->
    <bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="" />
    </bean>

    <bean id="transactionManager"
        class="org.springframework.batch.support.transaction.ResourcelessTransactionManager" />

    <!-- create job-meta tables automatically -->
    <jdbc:initialize-database data-source="dataSource">
        <jdbc:script location="org/springframework/batch/core/schema-drop-mysql.sql" />
        <jdbc:script location="org/springframework/batch/core/schema-mysql.sql" />
    </jdbc:initialize-database>

</beans>
```

5. Run It

The most simplest way to run a batch job.

```
package com.mkyong;

import org.springframework.batch.core.Job;
import org.springframework.batch.core.JobExecution;
import org.springframework.batch.core.JobParameters;
import org.springframework.batch.core.launch.JobLauncher;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
  public static void main(String[] args) {

    String[] springConfig  =
        {
            "spring/batch/jobs/job-hello-world.xml"
        };

    ApplicationContext context =
            new ClassPathXmlApplicationContext(springConfig);

    JobLauncher jobLauncher = (JobLauncher) context.getBean("jobLauncher");
    Job job = (Job) context.getBean("helloWorldJob");

    try {

        JobExecution execution = jobLauncher.run(job, new JobParameters());
        System.out.println("Exit Status : " + execution.getStatus());

    } catch (Exception e) {
        e.printStackTrace();
    }

    System.out.println("Done");

  }
}
```

Output

```
report.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<report>
    <record id="1001">
        <date>2013-07-29T00:00:00+08:00</date>
        <qty>980</qty>
        <sales>213100</sales>
        <staffName>mkyong</staffName>
    </record>
    <record id="1002">
        <date>2013-07-30T00:00:00+08:00</date>
        <qty>1080</qty>
        <sales>320200</sales>
        <staffName>staff 1</staffName>
    </record>
    <record id="1003">
        <date>2013-07-31T00:00:00+08:00</date>
        <qty>1200</qty>
        <sales>342197</sales>
        <staffName>staff 2</staffName>
    </record>
</report>
```

In console

```
Jul 30, 2013 11:52:00 PM org.springframework.batch.core.launch.support.SimpleJobLauncher$1 run
INFO: Job: [FlowJob: [name=helloWorldJob]] launched with the following parameters: [{}]
Jul 30, 2013 11:52:00 PM org.springframework.batch.core.job.SimpleStepHandler handleStep
INFO: Executing step: [step1]
Processing...Report [id=1001, sales=213100, qty=980, staffName=mkyong]
Processing...Report [id=1002, sales=320200, qty=1080, staffName=staff 1]
Processing...Report [id=1003, sales=342197, qty=1200, staffName=staff 2]
Jul 30, 2013 11:52:00 PM org.springframework.batch.core.launch.support.SimpleJobLauncher$1 run
INFO: Job: [FlowJob: [name=helloWorldJob]] completed with the following parameters: [{}] and the following status: [COMPL
Exit Status : COMPLETED
Done
```

Download Source Code

Download it – SpringBatch-Hello-World-Example.zip (http://www.mkyong.com/wp-content/uploads/2013/07/SpringBatch-Hello-World-Example.zip) (27 kb)

References

1. What is Batch Processing (http://en.wikipedia.org/wiki/Batch_processing)
2. Spring Batch Frameworks (http://static.springsource.org/spring-batch/)
3. Spring Batch – Reference Documentation (http://static.springsource.org/spring-batch/reference/html/)

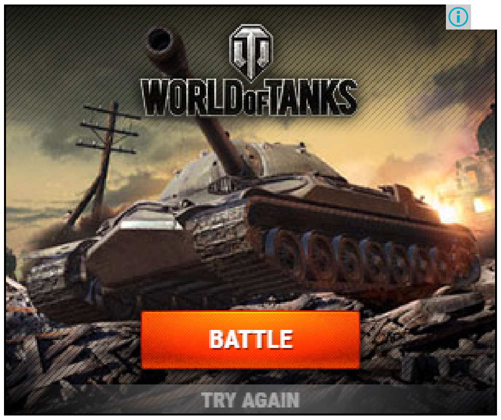
Tags : hello world (https://www.mkyong.com/tag/hello-world/) spring batch (https://www.mkyong.com/tag/spring-batch/)

Share this article on

Twitter (https://twitter.com/intent/tweet?text=Spring Batch Hello World Example&url=https://www.mkyong.com/spring-batch/spring-batch-hello-world-example/&via=mkyong) Facebook (https://www.facebook.com/sharer/sharer.php?u=https://www.mkyong.com/spring-batch/spring-batch-hello-world-example/) Google+ (https://plus.google.com/share?url=https://www.mkyong.com/spring-batch/spring-batch-hello-world-example/)

Related Posts

65k	Html Tutorial Hello World (/html/html-tutorial-hello-world/)
707k	JAX-WS Hello World Example - RPC Style (/webservices/jax-ws/jax-ws-hello-world-example/)
539k	Maven + Spring hello world example (/spring/quick-start-maven-spring-example/)
55k	TestNG Hello World Example (/unittest/testng-hello-world-example/)
326k	Struts 2 Hello World Example (/struts2/struts-2-hello-world-example/)
205k	Gradle - Spring 4 MVC Hello World Example (/spring-mvc/gradle-spring-mvc-web-project-example/)
512k	Spring Security hello world example (/spring-security/spring-security-hello-world-example/)
223k	Android hello world example (/android/android-hello-world-example/)
76k	Google app engine Java hello world example using Eclipse (/google-app-engine/google-app-engine-hello-world-example-using-eclipse/)
2k	GAE + Python hello world on Mac OS X (/google-app-engine/gae-python-hello-world-on-mac-os-x/)



Powered by Google

Spring Security Form Login Using Database

Spring Batch + Spring TaskScheduler example

Spring Batch metadata tables are not created au...

Spring Security Limit Login Attempts example

Spring Batch Example – XML File To CSV File

Spring Batch MultiResourceItemReader...

Spring Batch Example – MySQL Database To XML

Spring Batch Example – CSV File To MySQL Database

About the Author

mkyong

Founder of Mkyong.com (http://mkyong.com) and HostingCompass.com (http://hostingcompass.com), love Java and open source stuff. Follow him on Twitter (https://twitter.com/mkyong), or befriend him on Facebook (http://www.facebook.com/java.tutorial) or Google Plus (https://plus.google.com/110948163568945735692?rel=author). If you like my tutorials, consider make a donation to these charities (http://www.mkyong.com/blog/donate-to-charity/).

Popular Posts

1m
How to send HTTP request GET/POST in Java (/java/how-to-send-http-request-getpost-in-java/)

627k
Spring MVC hello world example (/spring-mvc/spring-mvc-hello-world-example/)

649k
How to create a Web Application Project with Maven (/maven/how-to-create-a-web-application-project-with-maven/)

704k
JAX-WS Hello World Example - RPC Style (/webservices/jax-ws/jax-ws-hello-world-example/)

969k
Android Tutorial (/tutorials/android-tutorial/)

660k
How to read XML file in Java - (SAX Parser) (/java/how-to-read-xml-file-in-java-sax-parser/)

558k
Spring JdbcTemplate Querying examples (/spring/spring-jdbctemplate-querying-examples/)

702k
Java XML Tutorial (/tutorials/java-xml-tutorials/)

1.2m
How to read file in Java - BufferedReader (/java/how-to-read-file-from-java-bufferedreader-example/)

603k
How to create XML file in Java - (DOM Parser) (/java/how-to-create-xml-file-in-java-dom/)

591k
Android ListView example (/android/android-listview-example/)
695k
Hibernate Query examples (HQL) (/hibernate/hibernate-query-examples-hql/)

968k
How to write to file in Java - BufferedWriter (/java/how-to-write-to-file-in-java-bufferedwriter-example/)
921k
Spring Security Tutorial (/tutorials/spring-security-tutorials/)

1.3m
Java Properties file examples (/java/java-properties-file-examples/)
2m
Spring Tutorial (/tutorials/spring-tutorials/)

909k
Java object sorting example (Comparable and Comparator) (/java/java-object-sorting-example-comparable-and-comparator/)
715k
Spring 3 MVC hello world example (/spring3/spring-3-mvc-hello-world-example/)

935k
How to convert Java object to / from JSON (Gson) (/java/how-do-convert-java-object-to-from-json-format-gson-api/)
902k
JAX-WS Tutorial (/tutorials/jax-ws-tutorials/)

Comments

← Older Comments (https://www.mkyong.com/spring-batch/spring-batch-hello-world-example/comment-page-2/#comments)
→

- Pingback: Лучшие фильмы2 (http://hd1080hd.ru/)()
- Pingback: HDKINOONLINE (http://www.google.com.mx/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCYQFjAA&url=https://www.facebook.com/HDKINOONLINE)()
- Pingback: clic (http://blog.i.ua/user/8129882)()
- Pingback: ukraine girl kiev lviv 2016 (http://bbs.hfut.edu.cn/forum/home/link.php?url=http://necessary-gifts.ru)()
- Pingback: ukraine girl kiev lviv (http://l-www.sitepal.com/affiliates/entry/?spdirect=1&affId=75895&promotionId=17691&link=http://necessary-gifts.ru)()
- Pingback: more (http://2.warcraft4.kl.com.ua/)()
- Pingback: лучшие фильмы года 2016 (http://www.etracker.de/rdirect.php?et=tMEVNm&et_cid=63&et_lid=6944&et_sub=backlinks&et_src=search&et_placement=&et_url=http%3A//hd2017hd.ru/)()
- Pingback: Дивергент, глава 3: За стеной / Allegiant, Высотка / High-Rise, (http://hd2017.ru)()
- Pingback: film online 2016 free ua ru kz (http://hd2017.ru/rss.php)()
- Pingback: zootopia 2016 (http://hd2017.ru/)()
- Pingback: rialudi (http://ria-lydi.ru/)()
- Pingback: my blog gp 2016 (http://i96619a2.bget.ru/)()
- Pingback: Zveropolis / Zootropolis smotret onlayn v horoshem kachestve. (http://hdkinopoisk.ru)()
- Pingback: èìÿ áëîãà (http://geometria.ru/users/4406801/blog/172118)()

← Older Comments (https://www.mkyong.com/spring-batch/spring-batch-hello-world-example/comment-page-2/#comments)
→

Rising Posts (100k-500k pv)

+11.4k	Jackson 2 - Convert Java Object to / from JSON (/java/jackson-2-convert-java-object-to-from-json/)
+5.4k	Java 8 forEach examples (/java8/java-8-foreach-examples/)
+4.2k	RESTful Java client with java.net.URL (/webservices/jax-rs/restfull-java-client-with-java-net-url/)
+4.2k	Log4j hello world example (/logging/log4j-hello-world-example/)
+4.1k	Java - Read a file from resources folder (/java/java-read-a-file-from-resources-folder/)
+4k	Apache HttpClient Examples (/java/apache-httpclient-examples/)
+3.6k	Spring @PropertySource example (/spring/spring-propertysources-example/)
+3.5k	Spring 4 MVC Ajax Hello World Example (/spring-mvc/spring-4-mvc-ajax-hello-world-example/)
+3.4k	Spring MVC form handling example (/spring-mvc/spring-mvc-form-handling-example/)
+3.4k	Spring Batch Tutorial (/tutorials/spring-batch-tutorial/)

VOLTDB
Transform your apps to FAST
with the world's fastest operational database
Free white paper >>

Rising Posts (10k-99k pv)

+2.9k	Java - Convert String to int (/java/java-convert-string-to-int/)
+2.8k	How to set java_home on Windows 10? (/java/how-to-set-java_home-on-windows-10/)
+2.7k	Java 8 Streams filter examples (/java8/java-8-streams-filter-examples/)
+2.5k	Spring embedded database examples (/spring/spring-embedded-database-examples/)
+2.1k	Java 8 Stream - Read a file line by line (/java8/java-8-stream-read-a-file-line-by-line/)
+2.1k	Java - How to split a string (/java/java-how-to-split-a-string/)
+1.9k	Java 8 Lambda : Comparator example (/java8/java-8-lambda-comparator-example/)
+1.8k	Java - Generate random integers in a range (/java/java-generate-random-integers-in-a-range/)
+1.8k	logback.xml Example (/logging/logback-xml-example/)

+1.5k	Java 8 - Filter a Map examples (/java8/java-8-filter-a-map-examples/)

Rising Posts (<10k pv)

+1.2k	Python - How to loop a dictionary (/python/python-how-to-loop-a-dictionary/)
+989	Java 8 - Collectors groupingBy and mapping example (/java8/java-8-collectors-groupingby-and-mapping-example/)
+843	Java 8 flatMap example (/java8/java-8-flatmap-example/)
+696	JUnit + Spring integration example (/unittest/junit-spring-integration-example/)
+520	Spring MVC - Refactoring a jQuery Ajax Post example (/spring-mvc/spring-mvc-refactoring-a-jquery-ajax-post-example/)
+474	Unit Test - What is Mocking? and Why? (/unittest/unit-test-what-is-mocking-and-why/)
+423	How to use Gradle Wrapper (/gradle/how-to-use-gradle-wrapper/)
+413	Java - How to convert Array to Stream (/java8/java-how-to-convert-array-to-stream/)
+400	Maven and JUnit example (/unittest/maven-and-junit-example/)
+362	How to view HTTP headers in Google Chrome? (/computer-tips/how-to-view-http-headers-in-google-chrome/)

Favorites Links

Android Getting Started (http://developer.android.com/training/index.html)
Google App Engine – Java (https://cloud.google.com/appengine/docs/java/)

Spring 2.5.x Documentation (http://docs.spring.io/spring/docs/2.5.x/reference/index.html)
Spring 3.2.x Documentation (http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/)
Spring 4.1.x Documentation (http://docs.spring.io/spring/docs/4.1.x/spring-framework-reference/html/)
Java EE 5 Tutorial (http://docs.oracle.com/javaee/5/tutorial/doc/docinfo.html)
Java EE 6 Tutorial (http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html)
Java EE 7 Tutorial (https://docs.oracle.com/javaee/7/tutorial/index.html)
Java 6 API (http://docs.oracle.com/javase/6/docs/api/overview-summary.html)
Java 7 API (http://docs.oracle.com/javase/7/docs/api/overview-summary.html)
Java 8 API (http://docs.oracle.com/javase/8/docs/api/overview-summary.html)
JSF Home Page (https://javaserverfaces.java.net/)
JSP Home Page (https://jsp.java.net/)
Maven Central Repository (http://search.maven.org/)
Hibernate ORM (http://hibernate.org/orm/)
JAX-WS Home Page (https://jax-ws.java.net/)
JAX-RS Home Page (Jersey) (https://jax-ws.java.net/)

Partners & Bookmarks

Java Code Geeks (http://www.javacodegeeks.com/)
TestNG Founder (http://beust.com/weblog/)
DZone (https://dzone.com)

About Mkyong.com

Mkyong.com is for Java and J2EE developers, all examples are simple and easy to understand, and well tested in my development environment.

Mkyong.com is created, written by, and maintained by Yong Mook Kim, aka Mkyong. It is built on WordPress (https://wordpress.org/), hosted by Liquid Web (//www.liquidweb.com/?RID=mkyong), and the caches are served by CloudFlare CDN.