springone 2GX

DALLAS

2014

Getting Started with Spring for Apache Hadoop

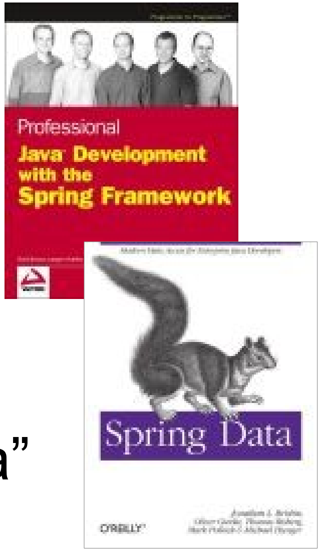
Thomas Risberg

springone 2GX

Who am I?

Thomas Risberg

- Working on the Spring Data engineering team at Pivotal
- Lead for the Spring for Apache Hadoop project
- Joined Spring Framework team in 2003 working on JDBC support
- co-author of “Professional Java Development with Spring Framework” from Wrox 2005 and “Spring Data” book from O'Reilly 2012



So, Hadoop, we're all in love?

Image credit: Martine Lemmens
http://www.freeimages.com/photo/1328068

Many organizations are using or evaluating Hadoop

- So, I assume you have some basic familiarity
- For the examples, you need access to an Hadoop cluster
  - (check resources slide at the end for instructions how to set up dev cluster)

Hadoop challenges

- *Hadoop has a challenging out-of-the-box programming model*
  - *Low level API*
  - *Uses checked exceptions (IOException)*
  - *Old and New MapReduce APIs*
  - *Hadoop v1 vs. v2 APIs*
- *Many different Hadoop ecosystem projects with diverging APIs and configuration styles*
- *Difficulty getting dependencies and Hadoop ecosystem project versions to work in harmony*
- *Non trivial applications often become a collection of shell scripts calling Hadoop command line applications*

Spring for Apache Hadoop

“

Spring for Apache Hadoop provides extensions to Spring, Spring Batch, and Spring Integration to build manageable and robust pipeline solutions around Hadoop.

A quick example ...

- Display a directory listing of the Hadoop Distributed File System (HDFS)
- We need:
  - Spring Boot
  - URL for the Hadoop Namenode
  - An instance of the File System Shell

Let's get Groovy ...

```groovy
@Grab("org.springframework.data:spring-data-hadoop:2.0.2.RELEASE-hadoop24")

import org.apache.hadoop.fs.FileStatus
import org.springframework.data.hadoop.fs.FsShell

public class Application implements CommandLineRunner {

    @Autowired FsShell fsShell;

    void run(String... strings) throws Exception {
        println "*** HDFS content:"
        for (FileStatus fs : fsShell.ls("/")) {
          println "${fs.owner} ${fs.group} : /${fs.path?.name}"
        }
    }

    @Bean FsShell fsShell() {
        org.apache.hadoop.conf.Configuration hadoopConfiguration =
                new org.apache.hadoop.conf.Configuration()
        hadoopConfiguration.set("fs.defaultFS", "hdfs://borneo:8020")
        return new FsShell(hadoopConfiguration);
    }
}
```

Let's try Java ...

```java
@ComponentScan
@EnableAutoConfiguration
public class Application implements CommandLineRunner {

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

    @Autowired
    FsShell fsShell;

    @Override
    public void run(String... strings) throws Exception {
        System.out.println("*** HDFS content:");
        for (FileStatus fs : fsShell.ls("/")) {
            System.out.println(fs.getOwner() +
                    " " +  fs.getGroup() + ": /" + fs.getPath().getName());
        }
    }

    @Bean
    FsShell fsShell() {
        org.apache.hadoop.conf.Configuration hadoopConfiguration =
                new org.apache.hadoop.conf.Configuration();
        hadoopConfiguration.set("fs.defaultFS", "hdfs://borneo:8020");
        return new FsShell(hadoopConfiguration);
    }
}
```

Running the app ...

```
$ spring run app.groovy

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.1.4.RELEASE)

*** HDFS content:
hdfs supergroup : /
hdfs supergroup : /tmp
hdfs supergroup : /user
trisberg supergroup : /xd
```

How do I build it?

- Use Maven or Gradle
- Use Spring Boot for running app
- Use Spring IO Platform BOM for dependency management
  - Override the versions for Hadoop if needed:
    - `spring-data-hadoop.version`
    - `hadoop.version`

Add Spring IO Platform

```
<parent>
  <groupId>io.spring.platform</groupId>
  <artifactId>platform-bom</artifactId>
  <version>1.0.1.RELEASE</version>
  <relativePath/>
</parent>

<properties>
  <spring-data-hadoop.version>2.0.2.RELEASE-hadoop24</spring-data-hadoop.version>
  <hadoop.version>2.4.1</hadoop.version>
  <java.version>1.7</java.version>
</properties>
```

Add Spring Boot

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-hadoop</artifactId>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

Build and then Run ...

```
$ mvn clean package
...
$ java -jar target/hello-hadoop-0.1.0.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.1.4.RELEASE)

*** HDFS content:
hdfs supergroup : /
hdfs supergroup : /tmp
hdfs supergroup : /user
trisberg supergroup : /xd
```

Spring for Apache Hadoop - Features

- Consistent programming and declarative configuration model
  - Create, configure, and parameterize Hadoop connectivity and all job types
  - Support for running MapReduce jobs, streaming, tool, jars
  - Configure Hadoop's distributed cache
  - Environment profiles – easily move application from dev to qa to prod
  - Support for working with Hive, Pig and Hbase
  - Writing to HDFS – partitioning, many data formats
  - Support for YARN programming

Spring for Apache Hadoop - Features

- Developer productivity
  - Create well-formed applications, not spaghetti script applications
  - Simplify HDFS access and FsShell API with support for JVM scripting
  - Helper “Template” classes for Pig/Hive/Hbase
  - Runner classes for MR/Pig/Hive for small workflows
  - Tasklet implementations for larger Spring Batch flows

Spring For Apache Hadoop - Use Cases

- Apply across a wide range of use cases
  - Ingest: Events/JDBC/NoSQL/Files to HDFS
  - Orchestrate: Hadoop Jobs
  - Export: HDFS to JDBC/NoSQL
- Spring Integration and Spring Batch make this possible
- Spring XD makes it even easier

Spring For Apache Hadoop - History

- Project started by Dave Syer and Costin Leau in 2011
- First 1.0 GA release in February 2013
- Current versions:
  - 1.1.0 supports Hadoop v1 & v2 –> 1.0.4 - 2.2.0
  - 2.0.2 supports Hadoop v1 & v2 –> 1.2.1 - 2.4.0
  - 2.1.0.M1 Hadoop v2 only –> 2.2.0 - ...

Versions, versions, versions ...

- This presentation and all accompanying examples uses:
  - Hadoop v2 APIs
  - Apache Hadoop version 2.4.1
  - Spring for Apache Hadoop 2.0.2.RELEASE-hadoop24
  - Hive version 0.13.1, hiveserver2
  - Spring XD version 1.0.0.RELEASE

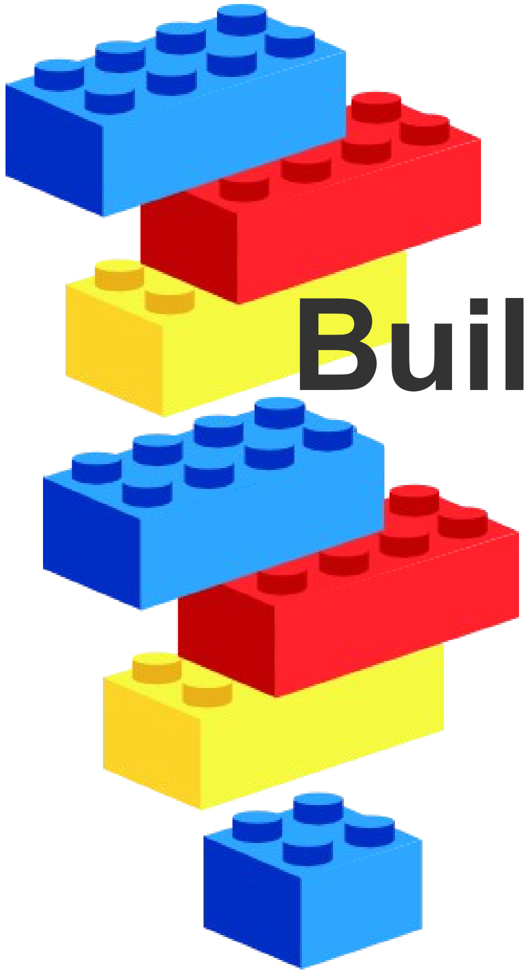
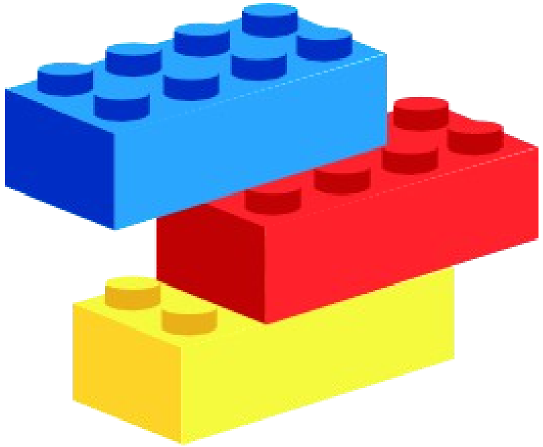
So, what about the distributions?

- Spring for Apache Hadoop provides several “flavors” to match dependencies with Hadoop distributions from:
  - Apache
  - Cloudera
  - Hortonworks
  - Pivotal

Supported distributions for 2.0

- `hadoop22 hadoop24 hadoop12`
- `cdh5 cdh4`
- `hdp21 hdp20 hdp13`
- `phd20 phd1`
  - Specified like this:
    - ➔ `2.0.2.RELEASE-hadoop24`

Building Blocks

Spring For Apache Hadoop - Configuration

- XML namespace

```xml
<hadoop:configuration>
  fs.defaultFS=${spring.hadoop.fsUri}
  yarn.resourcemanager.address=${spring.hadoop.resourceManagerHost}
</hadoop:configuration>
```

- @Bean

```java
@Value("${spring.hadoop.fsUri}")
String defaultFS;
@Value("${spring.hadoop.resourceManagerAddress}")
String resourceManager;

@Bean
Configuration hadoopConfiguration() {
    Configuration hadoopConfiguration = new Configuration();
    hadoopConfiguration.set("fs.defaultFS", defaultFS);
    hadoopConfiguration.set("yarn.resourcemanager.address", resourceManager);
    return hadoopConfiguration;
}
```

Considering providing improved support in version 2.1.x

Spring For Apache Hadoop - Jobs

- Runners for common jobs

  job-runner, jar-runner, tool-runner, pig-runner, hive-runner

```
<job id="tweetCountJob"
  input-path="${hdfs.input.path}"
  output-path="${hdfs.output.path}"
  jar="file:#{systemProperties['user.dir']}/target/lib/tweets-mapreduce.jar"
  mapper="com.springdeveloper.hadoop.TweetCountMapper"
  reducer="com.springdeveloper.hadoop.IntSumReducer"/>

<job-runner id="runner" run-at-startup="true"
  pre-action="setupScript"
  job-ref="tweetCountJob"
  post-action="resultsScript"/>
```

```
hdfs.input.path=/tweets/input/workflow
hdfs.output.path=/tweets/results
```

Spring For Apache Hadoop - Jobs

- Scripting support for File System Shell commands
  - mkdir, cp, chmod, rm
  - JavaScript, Groovy

```
if (!fsh.test(inputDir)) {
   fsh.mkdir(inputDir);
   fsh.copyFromLocal(localFile, inputDir);
   fsh.chmod(700, inputDir)
}
if (fsh.test(outputDir)) {
   fsh.rmr(outputDir)
}
```

```
<script id="setupScript" location="file-prep.groovy">
  <property name="localFile" value="#{systemProperties['user.dir']}/${local.file}"/>
  <property name="inputDir" value="${hdfs.input.path}"/>
  <property name="outputDir" value="${hdfs.output.path}"/>
</script>
```

```
local.file=../data/hadoop-tweets_2014-09-02.txt
hdfs.input.path=/tweets/input/workflow
hdfs.output.path=/tweets/results
```

MapReduce Example – Count Hashtags

```
{
  "created_at":"Mon Aug 11 13:43:13 +0000 2014",
  "entities": {
    "hashtags":[{
      "indices":[81,87],
      "text":"cisco"}
    ],
  ...
  }
  ...
  "text": "RT @cisco_dp: A platform for Hadoop as a Service (HaaS): ... #cisco ...",
  "user":{
    ...
    "screen_name":"John_Foxworth",
    "statuses_count":2576,
    "time_zone":"Eastern Time (US & Canada)"
    ...
  }
}
```

Find the number of times each #hashtag is used

The data file has the entire JSON document for each tweet on a single line

MapReduce Example – a Mapper

```java
public class TweetCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

      private final static IntWritable ONE = new IntWritable(1);
      private final ObjectMapper jsonMapper = new ObjectMapper(new JsonFactory());

      @Override
      protected void map(LongWritable key, Text value, Context context)
                  throws IOException, InterruptedException {

            Map<String, Object> tweet = jsonMapper.readValue(value.toString(),
                        new TypeReference<HashMap<String,  Object>>(){});
            Map<String, Object> entities = (Map<String, Object>) tweet.get("entities");
            List<Map<String, Object>> hashTagEntries = null;
            if (entities != null) {
                  hashTagEntries = (List<Map<String, Object>>) entities.get("hashtags");
            }
            if (hashTagEntries != null && hashTagEntries.size() > 0) {
                  for (Map<String, Object> hashTagEntry : hashTagEntries) {
                        String hashTag = hashTagEntry.get("text").toString();
                        context.write(new Text(hashTag), ONE);
                  }
            }
      }
}
```

MapReduce Example – and Reducer

```
public class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

     @Override
     protected void reduce(Text key, Iterable<IntWritable> values, Context context)
               throws IOException, InterruptedException {
          int sum = 0;
          for (IntWritable value : values) {
               sum += value.get();
          }
          context.write(key, new IntWritable(sum));
     }

}
```

MapReduce Example – the Boot App

```
@Configuration
@ImportResource("META-INF/spring/application-context.xml")
public class Application {

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

}
```

The app doesn't do much, it simply loads the app context and relies on the job getting started on context start-up

MapReduce Example – the app-context

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/hadoop"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/hadoop http://www.springframework.org/schema/hadoop/spring-hadoop.xsd">

        <configuration>
                fs.defaultFS=${spring.hadoop.fsUri}
                yarn.resourcemanager.hostname=${spring.hadoop.resourceManagerHost}
                mapreduce.framework.name=yarn
                mapreduce.jobhistory.address=${spring.hadoop.jobHistoryAddress}
        </configuration>

        <job id="tweetCountJob"
                input-path="${hdfs.input.path}"
                output-path="${hdfs.output.path}"
                jar="file:#{systemProperties['user.dir']}/target/lib/tweets-mapreduce.jar"
                mapper="com.springdeveloper.hadoop.TweetCountMapper"
                reducer="com.springdeveloper.hadoop.IntSumReducer"/>

        <job-runner id="runner" run-at-startup="true"
                job-ref="tweetCountJob" />

</beans:beans>
```

Use 'maven-dependency-plugin' to copy and strip the version from the tweets-mapreduce.jar

DEMO

MapReduce Example

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/tree/master/simple-mapreduce

Hive Example – Find Popular Tweeters

```
{
  ...
  "created_at": "Mon Aug 11 13:43:13 +0000 2014",
  "text": "RT @cisco_dp: A platform for Hadoop as a Service (HaaS): ... #cisco ...",
  "user": {
    ...
    "followers_count": 214,
    ...
    "screen_name": "John_Foxworth",
    ...
  }
}
```

Find the top 10 tweeters based on their number of followers

The data file has the entire JSON document for each tweet on a single line

Hive Example – a Boot App

```
@EnableAutoConfiguration
@Configuration
public class Application implements CommandLineRunner {

    @Inject
    JdbcTemplate hive2;

    @Value("${tweets.input}")
    String input;

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

...
```

Hive Example – a Boot App #2

```
    public void run(String... strings) throws Exception {
        System.out.println("Running Hive task using data from '" + input + "' ...");
        String ddl = "create external table if not exists tweetdata (value STRING) LOCATION '" + input + "'";
        hive2.execute(ddl);
        String query =
                "select tweets.username, tweets.followers " +
                "from " +
                "  (select distinct " +
                "    get_json_object(t.value, '$.user.screen_name') as username, " +
                "    cast(get_json_object(t.value, '$.user.followers_count') as int) as followers " +
                "    from tweetdata t" +
                "  ) tweets " +
                "order by tweets.followers desc limit 10";
        List<Map<String, Object>> results = hive2.queryForList(query);
        System.out.println("Results: ");
        for (Map<String, Object> r : results) {
            System.out.println(r.get("tweets.username") + " : " + r.get("tweets.followers"));
        }
    }

}
```

DEMO

Hive Example

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/tree/master/simple-hive

HDFS Shell Commands - FsShell

```
//requires three variables, localSourceFile and inputDir, outputDir
// use the shell (made available under variable fsh)
if (!fsh.test(inputDir)) {
   fsh.mkdir(inputDir);
   fsh.copyFromLocal(localSourceFile, inputDir);
   fsh.chmod(700, inputDir)
}
if (fsh.test(outputDir)) {
   fsh.rmr(outputDir)
}
```

```
<script id="setupScript" location="copy-files.groovy">
     <property name="localSourceFile" value="${basedir}/${localSourceFile}"/>
     <property name="inputDir" value="${wordcount.input.path}"/>
     <property name="outputDir" value="${wordcount.output.path}"/>
</script>
```

Small Workflow Example

- We are getting daily files containing tweets from a twittersearch for #hadoop
- Files are in a /var/hadoop-data/{date} directory
- We need to:
  - Copy them to HDFS directory
  - Run a MapReduce Job

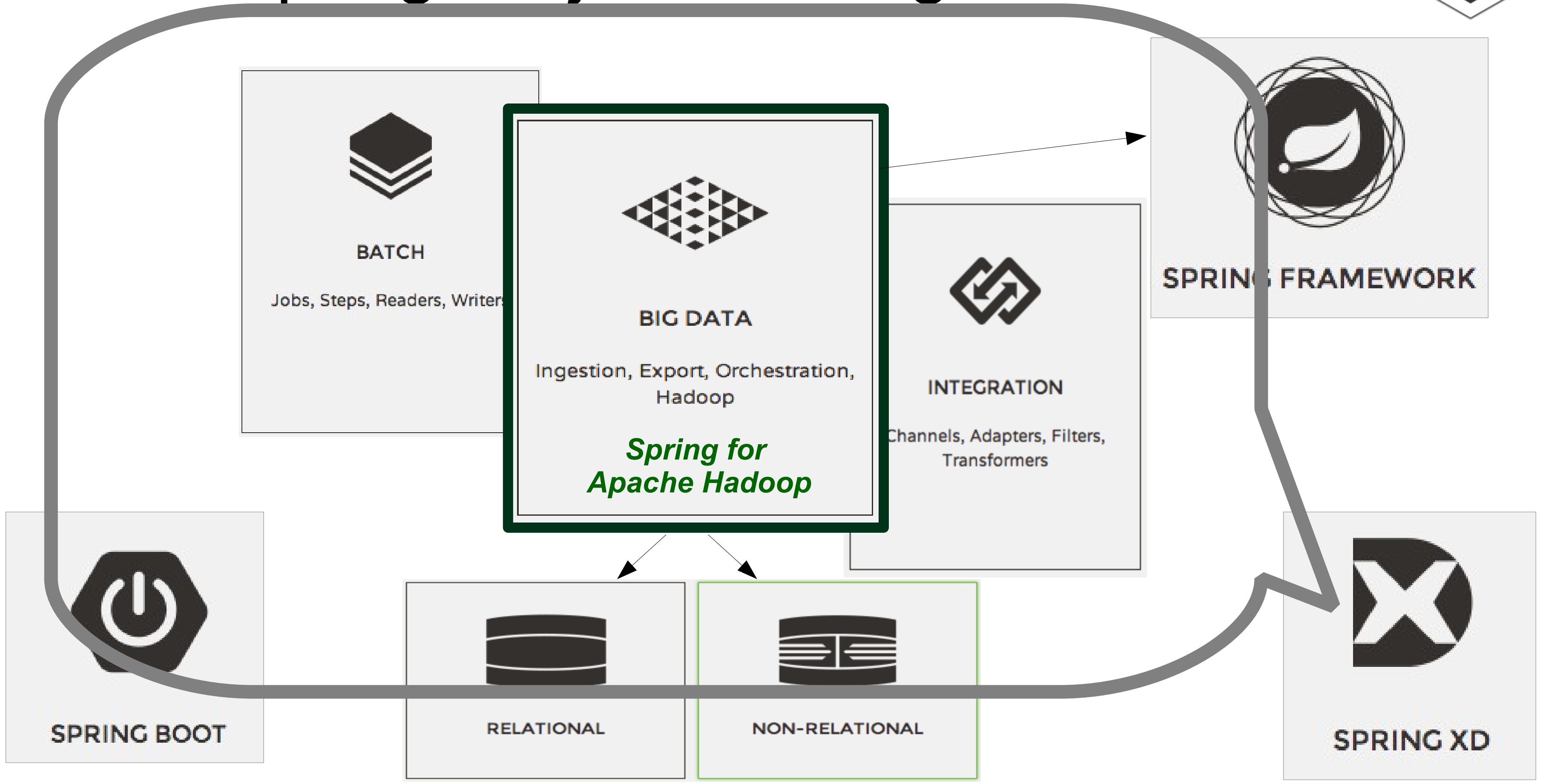
DEMO
Small Workflow Example

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/tree/master/simple-workflow

Bigger

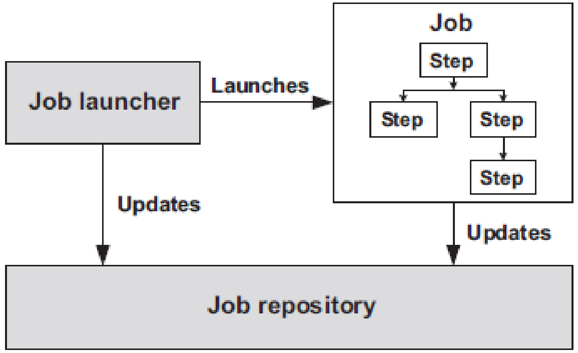
Workflows

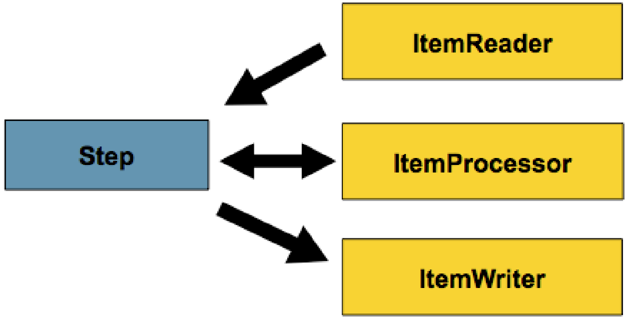
Spring Projects for Big Data


BATCH
Jobs, Steps, Readers, Writers
BIG DATA
Ingestion, Export, Orchestration, Hadoop
Spring for Apache Hadoop
INTEGRATION
Channels, Adapters, Filters, Transformers
SPRING FRAMEWORK
SPRING BOOT
RELATIONAL
NON-RELATIONAL
SPRING XD

Time for Batch

- Framework for batch processing
  - Basis for JSR-352
- Born out of collaboration with Accenture in 2007
- Features
  - parsers, mappers, readers, writers
  - automatic retries after failure
  - periodic commits
  - synchronous and asynch processing
  - parallel processing
  - partial processing (skipping records)
  - non-sequential processing
  - job tracking and restart





Spring XD Batch Job Packaging

```
modules/
├── job
│   └── tweets-hashtags
│   │   ├── config
│   │   │   └── tweets-hashtags.xml
│   │   └── lib
│   │       ├── tweets-mapreduce.jar
│   │       └── xd-batch-mapreduce-0.1.0.jar
```

Spring, Batch, XD, Hadoop etc. jars provided by Spring XD runtime

DEMO

Batch MapReduce

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/tree/master/xd-batch-mapreduce

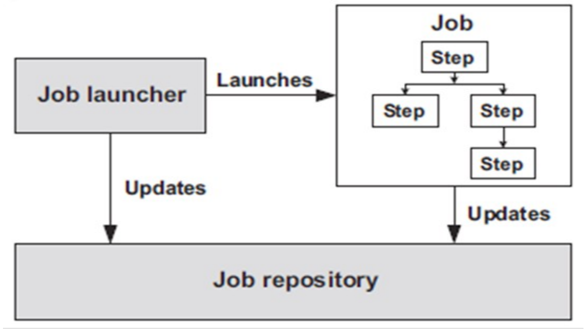
Spring Batch Workflows for Hadoop

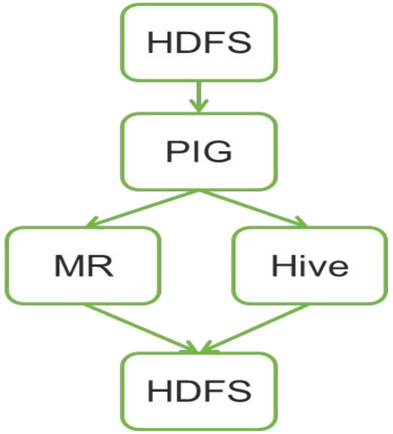
- Batch Ingest/Export
  - Examples
    - Read log files on local file system, transform and write to HDFS
    - Read from HDFS, transform and write to JDBC, HBase, MongoDB,…
- Batch Analytics
  - Orchestrate Hadoop based workflows with Spring Batch
  - Also orchestrate non-hadoop based workflows

Hadoop Analytical Workflow with Spring Batch

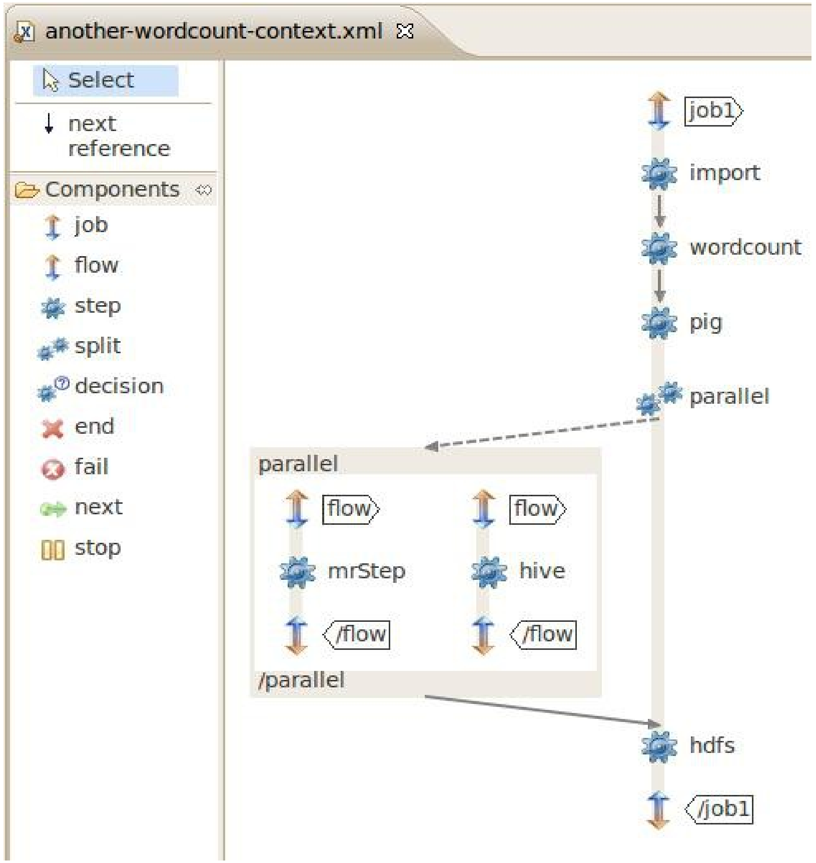
- Reuse same Batch infrastructure and knowledge to manage Hadoop workflows

- Step can be any Hadoop job type or HDFS script





Spring Batch Configuration for Hadoop



```
<job id="job1">
  <step id="import" next="wordcount">
    <tasklet ref="import-tasklet"/>
  </step>
  <step id="wc" next="pig">
    <tasklet ref="wordcount-tasklet"/>
  </step>
  <step id="pig">
    <tasklet ref="pig-tasklet"></step>
  <split id="parallel" next="hdfs">
    <flow><step id="mrStep">
        <tasklet ref="mr-tasklet"/>
      </step></flow>
    <flow><step id="hive">
        <tasklet ref="hive-tasklet"/>
      </step></flow>
  </split>
  <step id="hdfs">
    <tasklet ref="hdfs-tasklet"/></step>
</job>
```

Exporting HDFS to JDBC

- Use Spring Batch's
  - MutliFileItemReader
  - JdbcItemWriter

```
<step id="step1">
   <tasklet>
      <chunk reader="flatFileItemReader" processor="itemProcessor"
             writer="jdbcItemWriter"
             commit-interval="100" retry-limit="3"/>
      </chunk>
   </tasklet>
</step>
```

Batch Workflow Example

- We are getting daily files containing tweets from a twittersearch for #hadoop
- Files are in a /var/hadoop-data/{date} directory
- We need to:
  - Copy them to HDFS directory
  - Run MapReduce and Hive Jobs
  - Run some steps in parallel
  - Export data to RDBMS

DEMO

Batch Workflow

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/tree/master/xd-batch-workflow

Future Plans

- Spring for Apache Hadoop
  - Version 2.1 for any new features
    - Hadoop v2 only
    - Improved YARN support
    - Improved support for writing to HDFS
      - Avro, Parquet, HAWQ ...
  - Version 2.0 and 1.1 *in maintenance mode*
    - 2.0: hadoop 1.2.1 - 2.x
    - 1.1: hadoop 1.0.4 - 2.2.0 (HBase 0.94.x / Pig 0.11 / Hive 0.11)

Resources

Demo Source: https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop

Hadoop Install: https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/blob/master/InstallingHadoop.asciidoc

Project: http://projects.spring.io/spring-hadoop/

Questions: http://stackoverflow.com/questions/tagged/spring-data-hadoop

Twitter: https://twitter.com/springcentral

https://twitter.com/trisberg

YouTube: http://youtube.com/user/SpringSourceDev

Tips and Tricks

1. Copy dependency jar without version number

```
<plugin>
  <artifactId>maven-dependency-plugin</artifactId>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <outputDirectory>${project.build.directory}/xd-lib</outputDirectory>
        <includeArtifactIds>tweets-mapreduce</includeArtifactIds>
        <stripVersion>true</stripVersion>
        <excludeTransitive>true</excludeTransitive>
      </configuration>
    </execution>
  </executions>
</plugin>
```

2. Create an XD batch job module

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <configuration>
    <descriptors>
      <descriptor>src/main/assembly/assembly.xml</descriptor>
    </descriptors>
  </configuration>
  <executions>
    <execution>
      <id>package</id>
      <phase>package</phase>
      <goals>
        <goal>assembly</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

```
<fileSet>
  <directory>${project.basedir}/src/main/resources</directory>
  <outputDirectory>/modules/job</outputDirectory>
  <includes>
    <include>**/*.xml</include>
  </includes>
</fileSet>
```

```
<fileSet>
  <directory>${project.build.directory}/xd-lib</directory>
  <outputDirectory>/lib</outputDirectory>
  <includes>
    <include>**/*.jar</include>
  </includes>
</fileSet>
```

```
<dependencySet>
  <outputDirectory>/lib</outputDirectory>
  <useProjectArtifact>true</useProjectArtifact>
  <includes>
    <include>${project.groupId}:${project.artifactId}</include>
  </includes>
</dependencySet>
```

3. Use job parameters in config

```
<util:map id="stepExpr" map-class="java.util.HashMap" scope="step">
  <entry key="inputPath" value="#{jobParameters['input.path']?:'/tweets/input/'}"/>
  <entry key="tweetDate" value="#{(jobParameters['local.file'].split('_')[1]).substring(0,10)}"/>
  <entry key="outputPath" value="#{jobParameters['output.path']?:'/tweets/output'}"/>
  <entry key="hivePath" value="#{jobParameters['hive.path']?:'/tweets/hive'}"/>
</util:map>
```

```
<hadoop:property name="localFile" value="#{jobParameters['local.file']}"/>
```

```
<property name="dataPath" value="#{stepExpr['inputPath']+stepExpr['tweetDate']}"/>
```

```
output-path="#{stepExpr['outputPath']}/hashtags"
```

```
insert overwrite directory '#{stepExpr['hivePath']}/influencers'
```

4. Create a JDBC Tasklet for Batch

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/blob/master/
xd-batch-workflow/src/main/java/com/springdeveloper/data/jdbc/batch/JdbcTasklet.java

Dave Syer's example:

https://src.springframework.org/svn/spring-batch-admin/sandbox/cloud-
sample/src/main/java/org/springframework/batch/admin/sample/job/JdbcTasklet.java

Christian Tzolov's version:

https://github.com/tzolov/spring-xd-jdbc-
job/blob/master/src/main/java/com/gopivotal/spring/xd/module/jdbc/JdbcTasklet.java

Let's not forget YARN!

Dedicated YARN talk after lunch today 9/9:

Painless Build and Deploy for YARN Applications with Spring

- **Janne Valkealahti**
- **12:45 PM - 2:15 PM Trinity 6-7**

DEMO

YARN Example

https://github.com/SpringOne2GX-2014/Intro-to-Spring-Hadoop/tree/master/hello-yarn