



# Exemple SPARK API

(C)Hurence

Version v1.0, 18.02.2020: First draft

# Table des matières

Setup environnement .....	1
Installation de l'historian .....	2
Démarrer Apache Solr .....	3
Création du projet .....	4
Modification du fichier pom.xml .....	5
Exemple test de SPARK API .....	10

# Setup environnement

Pour commencer veuillez créer un dossier `hdh_workspace` par exemple qu'on va appeler `$HDH_HOME`.

```
# create the workspace anywhere you want
mkdir ~/hdh_workspace
export HDH_HOME=~/hdh_workspace
```

# Installation de l'historian

Hurence Data Historian est composé de scripts et fichiers binaires qui permettent de travailler avec les time series et les chunks. Téléchargez la dernière version de l'historian à l'adresse suivante [historian-1.3.5.tgz](#). Décompressez l'archive et entrez dedans, ensuite commencez l'installation en tapant la commande suivante :

```
sudo ./bin/install.sh
```

# Démarrer Apache Solr

Apache SolR est la base de donnée utilisée par l'historian, elle peut être remplacée par un autre moteur de recherche. Rendez-vous dans le dossier solr-8.2.0 et appliquez ces commandes :

```
# démarre un core Solr localement ainsi qu'un serveur zookeeper standalone.  
bin/solr start -cloud -s $SOLR_HOME/data/solr/node1 -p 8983  
# démarre un second core Solr localement qui va utiliser le serveur zookeeper  
précédamment créer.  
bin/solr start -cloud -s $SOLR_HOME/data/solr/node2/ -p 7574 -z localhost:9983
```

Vérifiez que votre instance solr fonctionne correctement en allant sur l'interface graphique à l'adresse suivante : "http://<solrhost>:8983/solr/#/~cloud"

# Création du projet

Pour tester notre api vous aurez besoin d'installer un IDE, dans notre cas nous allons utiliser IntelliJ IDEA: Créez un nouveau projet:

File > New > Project

Il faut spécifier Maven : un outil de gestion et automatisation de production des projets logiciels JAVA. Il faut choisir aussi la version 1.8 de java dans "project SDK" (il faut télécharger et installer le Java SE Development Kit).

Ensuite nommez votre projet et valider (dans notre cas on va l'appeller SPARK API tutorial).

Créez un dossier sous le nom "scala" sous le chemin SPARK API tutorial > src > main

et le convertir en un dossier "source root" pour qu'il soit compilable : clic droit sur le dossier, mark directory as puis source root.

# Modification du fichier pom.xml

Maven nous offre la possibilité d'ajouter des bibliothèques tierces à notre application. il suffit alors d'ajouter une balise `<dependency>` sous la section `<dependencies>`. On commence par l'ajout de scala car notre SPARK API est codé en scala.

```
<dependency>
  <groupId>org.scala-lang</groupId>
  <artifactId>scala-library</artifactId>
  <version>2.11.12</version>
</dependency>
```

On aura besoin aussi de plusieurs autres bibliothèques comme :

spark-core:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>2.3.2</version>
  <exclusions>
    <exclusion>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

Apache spark-sql:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.11</artifactId>
  <version>2.3.2</version>
</dependency>
```

Apache spark mllib:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-mllib_2.11</artifactId>
  <version>2.3.2</version>
  <scope>runtime</scope>
</dependency>
```

Apache solr:

```
<dependency>
  <groupId>org.apache.solr</groupId>
  <artifactId>solr-core</artifactId>
  <version>8.2.0</version>
  <type>jar</type>
  <scope>compile</scope>
</dependency>
```

Spark solr:

```
<dependency>
  <groupId>com.lucidworks.spark</groupId>
  <artifactId>spark-solr</artifactId>
  <version>3.6.6</version>
</dependency>
```

log4j:

```
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.16</version>
</dependency>
```

le jar du loader (obtenu lors de l'installation de l'historian) :

```
<dependency>
  <groupId>org.example</groupId>
  <artifactId>loader</artifactId>
  <version>1.3.5</version>
  <scope>system</scope>
  <systemPath>${HDDH_HOME}/historian-1.3.5/lib/loader-1.3.5.jar</systemPath>
</dependency>
```

puis créez une section `<build>` et au-dessous d'elle créez une section `<plugins>` pour ajouter les balises de plugins:

maven-enforcer-plugin:



```

<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-enforcer-plugin</artifactId>
<version>1.4.1</version>
<executions>
  <execution>
    <goals>
      <goal>enforce</goal>
    </goals>
    <configuration>
      <rules>
        <bannedDependencies>
          <excludes>
            <!--implementation binding-->
            <exclude>org.slf4j:slf4j-jdk14</exclude>
            <exclude>org.slf4j:slf4j-nop</exclude>
            <exclude>org.slf4j:slf4j-simple</exclude>
            <exclude>org.slf4j:slf4j-jcl</exclude>
            <exclude>org.slf4j:logback-classic</exclude>
            <exclude>org.slf4j:log4j-over-slf4j</exclude>
          </excludes>
        </bannedDependencies>
      </rules>
    </configuration>
  </execution>
</executions>
</plugin>

```

maven-surefire-plugin:

```

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.22.2</version>
</plugin>

```

scala-maven-plugin:

```

<plugin>
  <!-- see http://davidb.github.com/scala-maven-plugin -->
  <groupId>net.alchim31.maven</groupId>
  <artifactId>scala-maven-plugin</artifactId>
  <executions>
    <execution>
      <id>scala-compile-first</id>
      <phase>process-resources</phase>
      <goals>
        <goal>add-source</goal>
        <goal>compile</goal>
      </goals>
    </execution>
    <execution>
      <id>scala-test-compile</id>
      <phase>process-test-resources</phase>
      <goals>
        <goal>testCompile</goal>
      </goals>
    </execution>
  </executions>
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <executions>
    <execution>
      <phase>compile</phase>
      <goals>
        <goal>compile</goal>
      </goals>
    </execution>
  </executions>
</plugin>

```

maven-compiler-plugin:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <executions>
    <execution>
      <phase>compile</phase>
      <goals>
        <goal>compile</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Après avoir ajouté les bibliothèques que Inetllij IDEA les a téléchargés.

Créez un package sous le dossier scala et créez un object scala sous ce package. dans l'exemple suivant le package est nommé spark et l'objet scala est nommé JustSimpleTest.

# Exemple test de SPARK API

L'exemple suivant montre comment vous pouvez lire un fichier csv contenant des timeseries, le transformer (chunkifier) et puis l'injecter dans solr.

```
package spark

import com.hurence.historian.model.ChunkRecordV0
import com.hurence.historian.spark.ml.Chunkyfier
import com.hurence.historian.spark.sql
import com.hurence.historian.spark.sql.reader.MeasuresReaderType
import com.hurence.historian.spark.sql.reader.ReaderFactory
import com.hurence.historian.spark.sql.writer.{WriterFactory, WriterType}
import org.apache.spark.sql.SparkSession

object JustSimpleTest {

  def main(args: Array[String]): Unit = {
    val origpath = "$HDDH_HOME/historian/loader/src/test/resources/it-data-4metrics.csv.gz"

    val spark = SparkSession.builder
      .config("spark.master", "local[1]")
      .getOrCreate()

    import spark.implicits._

    val brutData = spark.read.format("csv").option("inferSchema",
      "true").option("header", "true").load(origpath)
```

metric_id	timestamp	value	metric_name	warn	crit	min	max
091c334c-a90a-4d8...	1575157723	13.375	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575157423	13.5	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575157123	13.375	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575156823	13.5	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575156523	13.75	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575156223	2.125	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575155923	2.25	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575155623	1.875	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575155323	8.5	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575155023	17.375	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575154723	15.375	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575154424	18.5	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575154124	18.75	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575153824	24.0	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575153524	18.125	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575153224	17.875	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575152924	14.375	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575152624	20.0	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575152324	12.0	cpu_prct_used	85.0	95.0	null	null
091c334c-a90a-4d8...	1575152025	14.25	cpu_prct_used	85.0	95.0	null	null

```

val reader = ReaderFactory.getMeasuresReader(MeasuresReaderType.GENERIC_CSV)
val measuresDS = reader.read(sql.Options(
  origpath,
  Map(
    "inferSchema" -> "true",
    "delimiter" -> ",",
    "header" -> "true",
    "nameField" -> "metric_name",
    "timestampField" -> "timestamp",
    "timestampDateFormat" -> "ms",
    "valueField" -> "value",
    "tagsFields" -> "metric_id,warn,crit"
  )))
//measuresDS.show(20,200)

```

name	value	timestamp	tags	year	month	hour	day
cpu_prct_used	13.375	1575157723	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	13.5	1575157423	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	13.375	1575157123	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	13.5	1575156823	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	13.75	1575156523	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	2.125	1575156223	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	2.25	1575155923	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	1.875	1575155623	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	8.5	1575155323	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	17.375	1575155023	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	15.375	1575154723	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	18.5	1575154424	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	18.75	1575154124	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	24.0	1575153824	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	18.125	1575153524	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	17.875	1575153224	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	14.375	1575152924	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	20.0	1575152624	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	12.0	1575152324	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19
cpu_prct_used	14.25	1575152025	[metric_id -> 091c334c-a90a-4d8f-ba75-2c936220cd64, warn -> 85.0, crit -> 95.0]	1970	1	6	1970-01-19

```
val chunkyfier = new Chunkyfier().setGroupByCols(Array("name", "tags.metric_id"))
val chunksDS = chunkyfier.transform(measuresDS).as[ChunkRecordV0]
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|   day|          name|          metric_id|          tags|          start|          end| |
|count|   min|         max|   first|   last|   stddev|          avg|
| chunk|         sax|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|1970-01-19|      consumers|adf35232-156a-4fa...|[metric_id -> adf...|1574640025|1575157911|
|1726|      0.0|      3.0|      0.0|      1.0| 0.515856681916497| 0.4084588644264195|H4sIAAAAAAAAO
Pi1...|bbbbbbbbbbbeeeeeee|
|1970-01-19|    memoryConsumed|ed071ac9-71ae-4d7...|[metric_id -> ed0...|1574640225|1575157855|
|853|19464.0| 1374248.0|41832.0|27744.0| 207274.612686945| 165339.23563892144|H4sIAAAAAAAAK
VYC...|bccbbcbcbdbdeebbbb|
|1970-01-19|      messages|98126546-eeec-486...|[metric_id -> 981...|1574640137|1575157736|
|1695| 159.0| 1335.0| 693.0| 471.0| 174.17302956224785| 586.1221238938053|H4sIAAAAAAAAF
1aC...|ddeeeddbbbbcabbab|
|1970-01-19|    messages_ready|38f9d677-f26b-4fd...|[metric_id -> 38f...|1574640030|1575157891|
|1725| 0.0| 5432860.0| 0.0| 0.0| 517905.0306762279| 70756.59768115942|H4sIAAAAAAAAO
Pi1...|cccccccccccccccccc|
|1970-01-19|      ack|1d0c7108-36ed-4a2...|[metric_id -> 1d0...|1574640034|1575157788|
|1724| 0.0| 0.0| 0.0| 0.0| 0.0| 0.0|H4sIAAAAAAAAO
Pi1...|cccccccccccccccccc|
|1970-01-19|    messages_ready|45fb8e90-62db-483...|[metric_id -> 45f...|1574640109|1575157860|
|1722| 0.0|2.5264905E7| 0.0| 0.0| 3749626.935959318| 951618.7543554007|H4sIAAAAAAAAO
2Ye...|bbbbbbbbbceebbbbbb|
|1970-01-19|    memoryConsumed|6f25a6a6-ffa9-4cb...|[metric_id -> 6f2...|1574640503|1575157940|
|848|23832.0| 36768.0|36768.0|36768.0| 627.8567316448054| 36737.49056603773|H4sIAAAAAAAAO
Pi1...|cccccccccccccccccc|
|1970-01-19|    messages_ready|5fb97183-9be6-4ab...|[metric_id -> 5fb...|1574640148|1575157959|
|1726| 0.0| 251.0| 0.0| 0.0| 9.84607564646363| 0.694090382387022|H4sIAAAAAAAAO
```

```
chunksDS.show()
```

```
val writer = WriterFactory.getChunksWriter(WriterType.SOLR)
writer.write(sql.Options("historian", Map(
  "zkhost" -> "localhost:9983",
  "collection" -> "historian",
  "tag_names" -> "metric_id,warn,crit"
)), chunksDS)
```

```
spark.stop()
```

```
}
}
```

```

{
  "responseHeader": {
    "zkConnected": true,
    "status": 0,
    "QTime": 302,
    "params": {
      "q": "*",
      "_": "1594806762314"}},
  "response": {
    "numFound": 220, "start": 0, "maxScore": 1.0, "docs": [
      {
        "chunk_day": "1970-01-19",
        "chunk_start": 1574640025,
        "chunk_end": 1575157911,
        "chunk_count": 1726,
        "chunk_avg": 0.4084588644264195,
        "chunk_stddev": 0.515856681916497,
        "chunk_min": 0.0,
        "chunk_max": 3.0,
        "chunk_first": 0.0,
        "chunk_last": 1.0,
        "chunk_sax": "bbbbbbbbbbbbbbbb",
        "chunk_value": "H4sIAAAAAAAAAOPi1GSAAi5WhTVMBgxc03gUD3XMqrAaTAlQLUWTZyLpuqjvcvJMpp07Bt422tg8cH4Yeq4axYMTD93UQtjLw9lvtFY9+MNU0N0gA2vTYDIdu2rquhC3aa5WJawCgyfkaIkpCbHb",
        "name": "consumers",
        "metric_id": "adf35232-156a-4fad-bde2-12467b76ca57",
        "id": "ad0ab3f35655f50f71b74049ae8a135314ba4129d0d1075b5e8587c9501fc7a2",
        "_version_": 1672270699597660160},
      {
        "chunk_day": "1970-01-19",
        "chunk_start": 1574640030,
        "chunk_end": 1575157891,
        "chunk_count": 1725,
        "chunk_avg": 70756.59768115942,
        "chunk_stddev": 517905.0306762279,
        "chunk_min": 0.0,
        "chunk_max": 5432860.0,
        "chunk_first": 0.0,
        "chunk_last": 0.0,
        "chunk_sax": "cccccccccccccccc",
        "chunk_value": "H4sIAAAAAAAAAOPi1GSAAi5WhTVMBgxc1MSsCqvB1ADVTR5Ym+nrl/rZRj2bME0auLRA31AYeH809bRPuQ24TRgKrh/0eGiEztBw5dByKK+Iey/oR8CQ98Hg9eX1LV14G0KfBcMvNsH14uGeiuP",
        "name": "messages_ready",
        "metric_id": "38f9d677-f26b-4fda-b489-29b9e426a9c9",
        "warn": "100000.0",
        "crit": "500000.0",
        "id": "f9bedb67d4bb82258e5eac26e7042ad6b8f54a7b234d6d811c6523893b5865b8",
        "_version_": 167227070039110656},
    ]
  }
}

```