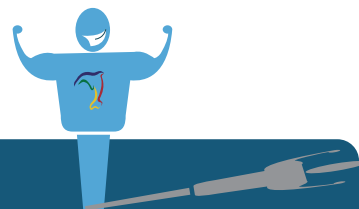


¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

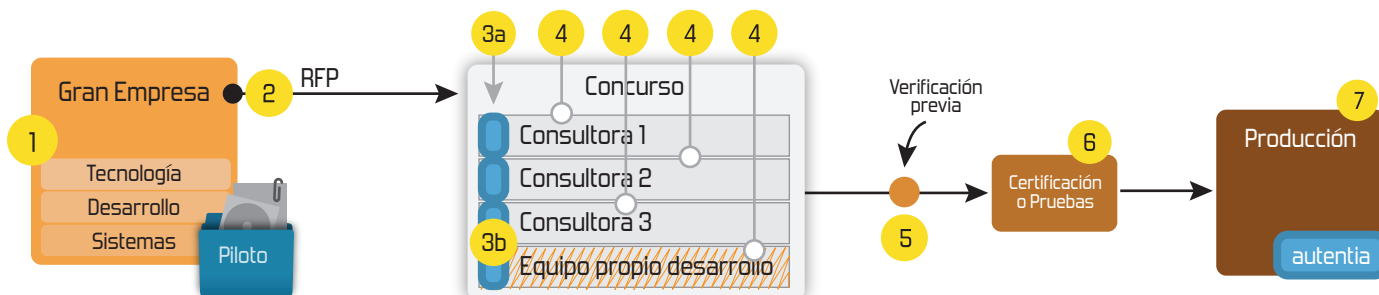
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

Control de autenticación y
acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)



Entra en Adictos a través de



E-mail

Contraseña

Entrar

Registrarme
Olvidé mi contraseña[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestros libros](#) [Más](#)» Estás en: [Inicio](#) [Tutoriales](#) [Testing de Hadoop con MRUnit](#)

Juan Alonso Ramos

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en [Autentia](#): Ofrecemos de servicios soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

[Ver todos los tutoriales del autor](#)

Fecha de publicación del tutorial: 2014-06-30

Tutorial visitado 1 veces [Descargar en PDF](#)

Testing de Hadoop con MRUnit

0. Índice de contenidos.

1. Introducción.
2. Entorno.
3. Configuración del proyecto.
4. Test unitarios.
5. Test de integración.
6. Conclusiones.

1. Introducción.

Para hacer test unitarios de algoritmos Map Reduce disponemos de la librería Apache MRUnit, una librería orientada a probar algoritmos Map Reduce de Hadoop que se integra con JUnit. Para asegurarnos que los cambios futuros en el software no rompen nada de lo que ya funciona, debemos hacer test automáticos de manera que el ciclo de vida asegure que se ponen en verde la batería de tests con cada cambio nuevo que se introduzca.

Por ello no debería ser una recomendación sino una obligación dejar probado con tests unitarios que nuestro código hace lo que debe ya que si mañana alguien o nosotros mismos metemos cualquier cambio en el algoritmo, nos aseguremos que todo sigue funcionando perfectamente.

En este tutorial vamos a hacer test unitarios y de integración para probar cómo funciona MRUnit.

Puedes descargar el código del tutorial desde mi repositorio de github pinchando [aquí](#).

2. Entorno.

El tutorial se ha realizado con el siguiente entorno:

- Ubuntu 12.04 64 bits
- Oracle Java SDK 1.6.0_27
- Apache Hadoop 2.2.0
- MRUnit 1.0.0
- Apache Maven 3.1.1

3. Configuración del proyecto

Partiremos del proyecto que creamos en [este](#) tutorial. Teníamos un algoritmo MapReduce que procesaba un fichero con datos climatológicos donde el resultado era la agrupación de los datos históricos por uno de los índices de polución por cada provincia de Castilla y León. Ahora vamos a hacer tests con MRUnit para asegurar que nuestro algoritmo funciona correctamente. El desarrollo lógico habría sido hacer el test junto al código y no después aunque nunca es tarde :P

Aunque lógicamente podemos testear las clases con JUnit y mockear las clases de Hadoop con Mockito por ejemplo, MRUnit nos facilita mucho el trabajo ya que nos proporciona drivers específicos para el mapper y el reducer. También dispone de un driver para hacer tests de integración y poder probar las clases en conjunto.

Para empezar añadimos al pom.xml las librerías de JUnit y MRUnit.

```
1 <dependencies>
2   <dependency>
3     <groupId>junit</groupId>
4     <artifactId>junit</artifactId>
5     <version>4.11</version>
```

Catálogo de servicios Autentia



Síguenos a través de:



Últimas Noticias

» Screencasts de programación narrados en Español

» Sorteo de entradas para APIdays Mediterranea

» Concurso del Día de la Madre:

» Aprende gratis ReactiveCocoa

» Checklist de Scrum de Autentia

[Histórico de noticias](#)

Últimos Tutoriales

» Dart, el lenguaje de programación web del futuro creado por Google

» Cómo integrar en Gradle un servidor Jetty o Tomcat

» Taller de Ventas

» Introducción a Spring Data Hadoop

» Desarrollando una aplicación de detección de iBeacons

```

6         <scope>test</scope>
7     </dependency>
8
9     <dependency>
10         <groupId>org.apache.mrunit</groupId>
11         <artifactId>mrunit</artifactId>
12         <version>1.0.0</version>
13         <classifier>hadoop2</classifier>
14         <scope>test</scope>
15     </dependency>
16 </dependencies>

```

Importante!! Hay que indicarle en el classifier la versión de Hadoop que tenemos instalada.

4. Tests unitarios.

Para las pruebas del mapper, MRUnit nos proporciona la clase MapDriver. En nuestro test debemos crear una instancia de esta clase pasándole en el constructor el mapper que queremos probar. Para el reducer existe un driver similar.

En el test debemos configurar la entrada y salida del MapDriver con los métodos **withInput** indicando los datos de entrada al mapper y **withOutput** con la salida esperada. Lanzamos el test llamando al método **runTest** el cuál llamará al método **map** de nuestro mapper pasándole en la entrada los datos que metimos en el **withInput**. Si la salida del mapper no corresponde con lo que metimos en el método **withOutput** se lanzará una **AssertionError** haciendo fallar el test.

```

1 package com.autentia.tutoriales;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5
6 import org.apache.hadoop.io.DoubleWritable;
7 import org.apache.hadoop.io.LongWritable;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapreduce.Mapper;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.ReduceDriver;
12 import org.junit.Assert;
13 import org.junit.Before;
14 import org.junit.Test;
15
16 public class AirQualityTest {
17
18     private MapDriver<LongWritable, Text, Text, DoubleWritable> mapDriver;
19     private ReduceDriver<Text, DoubleWritable, Text, Text> reduceDriver;
20     private MapReduceDriver mapReduceDriver;
21
22     private final AirQualityMapper mapper = new AirQualityMapper();
23     private final AirQualityReducer reducer = new AirQualityReducer();
24
25     @Before
26     public void setUp() {
27         mapDriver = new MapDriver(mapper);
28         reduceDriver = new ReduceDriver(reducer);
29         mapReduceDriver = new MapReduceDriver(mapper, reducer);
30     }
31
32     @Test
33     public void shouldReturnCOValueForProvinceOnlyWhenValueIsANumber() throws IOException {
34         mapDriver.withInput(new LongWritable(0), new Text("01/01/1997;1.2;12;33;63;56;;;");
35         mapDriver.withOutput(new Text("ÁVILA"), new DoubleWritable(1.2));
36
37         mapDriver.runTest();
38     }
39
40     @Test
41     public void shouldReturnEmptyResultMapWhenValueIsNotNumber() throws IOException {
42         mapDriver.withInput(new LongWritable(0), new Text("DIA;CO (mg/m3);NO (ug/m3);NO2
43
44         Assert.assertTrue(mapDriver.run().isEmpty());
45     }
46 }

```

Otra opción si queremos recuperar la salida del mapper es utilizar el método **run()** del **mapDriver**. Este método ejecutará la tarea **Map** y nos devolverá el resultado para que seamos nosotros quienes evaluemos el resultado. Los tests del reducer son muy similares.

```

1 ...
2
3 @Test
4 public void shouldReturnTheAverageOfValuesByProvince() throws IOException {
5     reduceDriver.withInput(new Text("ÁVILA"), new ArrayList<DoubleWritable>() {
6         {
7             add(new DoubleWritable(1.5));
8             add(new DoubleWritable(1.4));
9             add(new DoubleWritable(1.6));
10         }
11     });
12     reduceDriver.withOutput(new Text("ÁVILA"), new Text("1.5"));
13     reduceDriver.runTest();
14 }
15
16 @Test
17 public void shouldReturnEmptyResultReduceWhenValuesListIsEmpty() throws IOException {
18     reduceDriver.withInput(new Text("ÁVILA"), new ArrayList<DoubleWritable>());
19
20     Assert.assertTrue(reduceDriver.run().isEmpty());
21 }
22 }

```

5. Tests de integración.

Últimos Tutoriales del Autor

- » Introducción a Spring Data Hadoop
- » Implementar una función UDF de Apache Pig
- » Primeros pasos con Apache Pig
- » Implementando tu propio Writable en Hadoop
- » Primeros pasos de MapReduce con Hadoop

Una vez probados el mapper y el reducer por separado vamos a hacer un test de integración que pruebe todo en conjunto. El funcionamiento es similar al anterior, creamos un `mapReduceDriver` configurando la entrada y el resultado esperado para la salida. Probaremos varios casos, con datos buenos y malos y varios tipos de salida.

```

1  @Test
2  public void shouldMapInputLinesAndReduceByCOAndProvince() throws IOException {
3      mapReduceDriver
4          .withInput(new LongWritable(1), new Text("01/01/1997;1.2;12;33;63;56;;;19;ÁVILA"))
5          .withInput(new LongWritable(2), new Text("22/04/1997;1.1;45;49;75;No cumple el anexo IV..."))
6
7          .withInput(new LongWritable(3), new Text("24/12/2003;1.6;27;22;45;25;4;;;6;BURGOS"))
8          .withInput(new LongWritable(4), new Text("25/12/2003;2.2;35;13;53;28;21;;;7;BURGOS"))
9          .withInput(new LongWritable(5), new Text("26/12/2003;No cumple el anexo IV..."))
10         .withInput(new LongWritable(6), new Text("27/12/2003; 0.1;18;15;45;30;24;;;7;BURGOS"))
11
12         .withOutput(new Text("BURGOS"), new Text("1.3"))
13         .withOutput(new Text("ÁVILA"), new Text("1.15"))
14
15         .runTest();
16     }

```

Si todo ha ido bien se pondrán en verde nuestros tests y lo mejor, que al quedar integrados en el ciclo de vida de nuestro proyecto nos aseguramos que todo seguirá funcionando.

```

1  -----
2  T E S T S
3  -----
4  Running com.autentia.tutoriales.AirQualityTest
5  18:06:53,514 DEBUG main util.Shell:326 - setuid exited with exit code 0
6  18:06:53,988 DEBUG main mrunit.MapDriverBase:296 - Mapping input (0, 01/01/1997;1.2;12;33)
7  18:06:54,094 DEBUG main mrunit.TestDriver:651 - Matched expected output (ÁVILA, 1.2) at p
8  18:06:54,188 DEBUG main mapreduce.MapReduceDriver:252 - Starting map phase with mapper: co
9  18:06:54,208 DEBUG main mapreduce.MapReduceDriver:264 - Starting reduce phase with reducer:
10 18:06:54,238 DEBUG main mapreduce.MapReduceDriver:220 - Reducing input (BURGOS, (1.6, 2.2,
11 18:06:54,239 DEBUG main mapreduce.MapReduceDriver:220 - Reducing input (ÁVILA, (1.2, 1.1))
12 18:06:54,270 DEBUG main mrunit.TestDriver:651 - Matched expected output (ÁVILA, 1.15) at g
13 18:06:54,279 DEBUG main mrunit.TestDriver:651 - Matched expected output (BURGOS, 1.3) at g
14 18:06:54,458 DEBUG main mrunit.TestDriver:338 - Reducing input (ÁVILA, (1.5, 1.4, 1.6))
15 18:06:54,473 DEBUG main mrunit.TestDriver:651 - Matched expected output (ÁVILA, 1.5) at p
16 Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.84 sec

```

6. Conclusiones.

Como véis, hacer tests de algoritmos MapReduce es muy sencillo si tienes las herramientas adecuadas. Conociendo JUnit te resultará muy sencillo utilizar MRUnit ya que su funcionamiento es similar.

Puedes descargar el código del tutorial desde mi repositorio de github pinchando [aquí](#).

Espero que te haya sido de ayuda.

Un saludo.

Juan

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share |  0

¡Anímate y coméntanos lo que pienses sobre este **TUTORIAL**!

» **Regístrate** y accede a esta y otras ventajas «



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

IMPULSA

Impulsores

Comunidad

¿Ayuda?

.....

0 personas han traído clicks a esta página

sin clicks + + + + + + + +

powered by [karmacray](#)

Copyright 2003-2014 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

