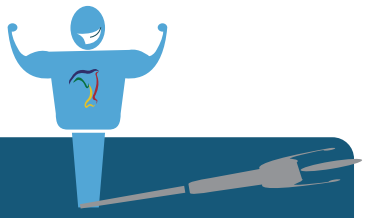
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

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

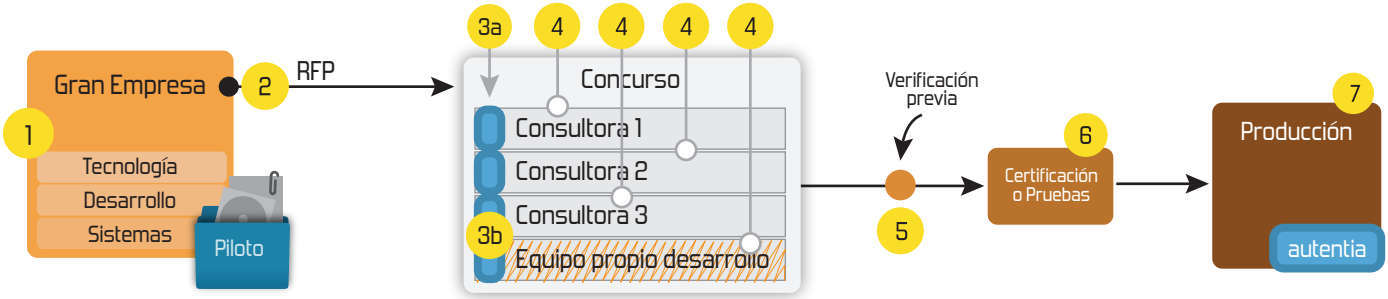
1. Desarrollo de componentes y proyectos a medida



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
- 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
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

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: 2014-05-06

Tutorial visitado 2 veces Descargar en PDF

Implementar una UDF de Apache Pig

0. Índice de contenidos.

1. Introducción.

Este tutorial es una continuación del de primeros pasos con Apache Pig donde vamos a ver cómo crear una función UDF (User-Defined Function) de Pig para extender la funcionalidad que viene por defecto.

Las UDFs de Pig se pueden escribir en Java, Javascript, Jython, Python, Ruby y Groovy. Una librería que contiene un gran variedad de UDFs es la de piggybank. Contiene funciones para manejar fechas, operaciones matemáticas, estadísticas, manejo de strings, etc.

Si el tutorial de primeros pasos de Apache Pig se te quedó corto para afrontar algún cálculo más complejo, a continuación vamos a ver uno de los aspectos más interesentes de esta utilidad.

Puedes descargarte 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
- Apache Pig 0.12.0

3. Crear la UDF

Para crear una User-Defined Function de Pig podemos hacerlo en varios lenguajes. Yo lo haré en Java por lo primeramente crearé un proyecto maven. En el pom.xml añadimos las dependencias de Hadoop y Pig.

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4      <modelVersion>4.0.0</modelVersion>
5
6      <groupId>com.autentia.tutoriales</groupId>
7      <artifactId>pig-udf</artifactId>
8      <version>1.0-SNAPSHOT</version>
9      <packaging>jar</packaging>
10
```

```
    <name>pig-udf</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <encoding>${project.build.sourceEncoding}</encoding>
                    <source>1.6</source>
                    <target>1.6</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.2.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>2.2.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.pig</groupId>
            <artifactId>pig</artifactId>
            <version>0.12.0</version>
        </dependency>
    </dependencies>
</project>
```

Una vez que tenemos la estructura del proyecto Maven creada creamos la clase de la UDF. ésta debe extender la clase **org.apache.pig.EvalFunc** e implementar el método **exec**. Este método será el que llame Pig cuando en el script indiquemos la UDF a utilizar.

Voy a utilizar el mismo ejemplo del tutorial de primeros pasos con Apache Pig. En este script sacábamos el valor medio de las medidas tomadas en el fichero. Pig era el encargado de calcular la media y nos devolvía el valor en decimal con varios decimales. Vamos a hacer una función para redondear a 2 decimales.

```
package com.autentia.tutoriales;

import java.io.IOException;

import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

public class DecimalFormat extends EvalFunc<String> {

    public String exec(Tuple input) throws IOException {
        if (null == input || input.size() != 1) {
            return null;
        }

        final Double number = (Double) input.get(0);

        try {
            return new java.text.DecimalFormat("#.##").format(input.get(0));
        } catch (Exception e) {
            return String.valueOf(number);
        }
    }
}
```

4. Usar la UDF desde el script de Pig

Ahora vamos a hacer carga de la librería y usar la UDF que acabamos de crear. Antes de nada empaquetamos en un jar la clase con **mvn package**. En el script hacemos la carga de la librería con el comando **register**. A continuación creamos un alias para la función DecimalFormat. En la línea 17 cuando se calcula la media del valor se le pasa a la función DecimalFormat que la formateará a 2 decimales devolviendo el resultado que posteriormente se guardará en la salida:

```
-- Registramos la librería de la UDF
register 'target/pig-udf-1.0-SNAPSHOT.jar';

-- Alias para la UDF
define DecimalFormat com.autentia.tutoriales.DecimalFormat();

-- Hacemos la carga del fichero separando por ';' y creamos el schema.
measure = load 'input/calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';') as (date
```

Últimos Tutoriales

» Primeros pasos con Bonita BPM Community 6.2.6

» Lanzamos nuevo site en Autentia.com

» Primeros pasos con Apache Pig

» Android Flavors

» Maven, Jenkins, Sonar y tests de integracion

Últimos Tutoriales del Autor

» Primeros pasos con Apache Pig

» Implementando tu propio Writable en Hadoop

» Primeros pasos de MapReduce con Hadoop

» Primeros pasos con Hadoop: instalación y configuración en Linux

» Trabajando con Mule ESB

Categorías del Tutorial

Big Data

```
9
10 -- Filtramos los resultados, la primera línea no nos vale.
11 filter_measure = filter measure by date != 'DIA';
12
13 -- Agrupamos los datos por provincia.
14 measure_by_province = group filter_measure by province;
15
16 -- Recorremos los registros por provincia y calculamos la media de co.
17 num_measures_by_province = foreach measure_by_province generate group, DecimalFormat(AVG
18
19 -- Ordenamos de menor a mayor índice de co.
20 ordered_measures = order num_measures_by_province by measure;
21
22 -- Almacenamos la salida en un fichero
23 store ordered_measures INTO 'output/measures_by_province.out';
```

Si lanzamos el script podemos observar el resultado

```
1 pig -x local -f udf.pig                                                    ?
```

```
1 SORIA       0.18                                                           ?
2 VALLADOLID  0.69
3 ZAMORA      0.84
4 BURGOS      0.86
5 ÁVILA       0.96
6 LEÓN        0.98
7 SEGOVIA     1.02
8 PALENCIA    1.18
9 SALAMANCA   1.39
```

6. Conclusiones.

Una buena herramienta de software es aquella que te permite multitud de opciones para desarrollar tu código pero es incluso mejor si no está cerrada sino que permite extenderla y enriquecerla con librerías personalizadas. En este tutorial hemos visto cómo crear una UDF para extender la funcionalidad de Apache Pig de una forma muy sencilla.

Si no tienes bastante con la funcionalidad por defecto que viene con la distribución de Apache Pig existen librerías de UDFs ya implementadas con multitud de funciones:

- Twitter Elephant Bird
- PiggyBank
- Amazon PiggyBank

Puedes descargarte el código del tutorial desde mi repositorio de github pinchando aquí.

Espero que te haya sido de ayuda.

Un saludo.

Juan


XHTML 1.0
CSS
RSS
ATOM