基于Morphlines的ETL开发流程

作者：费英林

目录

[1. 系统概要 2](#_Toc423457321)

[1.1. 背景介绍 2](#_Toc423457322)

[1.2. 系统架构 3](#_Toc423457323)

[1.3. UDH Search与Morphline的关系 3](#_Toc423457324)

[2. 系统安装与配置 4](#_Toc423457325)

[2.1. UDH Search服务安装 4](#_Toc423457326)

[3. Morphline命令开发 4](#_Toc423457327)

[3.1. CommandBuilder 4](#_Toc423457328)

[3.2. Command 5](#_Toc423457329)

[4. Morphline配置文件 7](#_Toc423457330)

[5. Morphline部署运行 8](#_Toc423457331)

[5.1. MapReduce作业 8](#_Toc423457332)

[5.2. Lily NRT Indexer 8](#_Toc423457333)

Morphlines是一个开源框架，它降低了Hadoop平台上开发和维护ETL流程的时间和成本，通过Morphlines可以抽取、转换并加载数据到Apache Solr、HBase、HDFS、企业数据仓库或者在线分析应用。我们可以很方便的通过配置文件对morphline的流程进行配置，不需要进行Java编码工作。

Morphlines本身是一个类库，可以嵌入到任何Java代码里。一个morphline是一个转换命令集合，这些命令对记录进行加载、转换等等。记录是一个内存数据结构，由键值对组成，可能含有blob或POJO附件。

# 系统概要

## 背景介绍

在Hadoop平台上开发ETL应用是一项费时费力的工作，要编写一些复杂的处理程序，如MapReduce作业、Oozie工作流等等。为了适应多样化的应用需求，缩短应用开发周期，降低开发维护成本，Cloudera公司开发了Morphlines框架。这个框架通过简单的配置文件来定义一个ETL流程，不需要任何Java编码即可满足大部分应用需求。如果既有的命令无法满足应用需求，程序员也可以写一些命令，这些命令可以方便的整合到Morphlines框架中。

## 系统架构



1）输入源：ETL数据源，Morphlines可以从Flume、HBase和HDFS中读取数据。

2）Morphline：一个自定义的工作流配置文件。

3）输出：Morphline可以输出数据到多个系统，如HBase、HDFS、Solr及企业数据仓库等等。

## UDH Search与Morphline的关系

Morphline最初是作为Search的一部分来开发的，后来为了有更多的开发者和使用者能够了解和扩展Morphline的功能，Morphline形成了一个独立的开源项目。利用Morphline可以实现Search中的ETL阶段工作，也可以实现Search之外的一些ETL工作。本文以Morphline在Search中的应用为例，讲述代码开发和部署，以及一些需要注意的问题。下图是Morphline在Search整体架构中的角色应用。



# 系统安装与配置

## UDH Search服务安装

 在集群上部署UDH Search服务，参见“专题-UDH Search系统构建及其应用”一文。需要配置Lily HBase Batch Indexer和Lily HBase NRT Indexer，分别用于索引的批量创建和实时创建，其中索引的实时创建是利用HBase的Replication机制实现的。

# Morphline命令开发

 Morhpline命令需要实现Java接口Command或它的子类AbstractCommand，另外还需要一个CommandBuilder接口的实现类，在这个CommandBuilder里定义Morhpline命令的名字。下面以删除某字段里Html标签为例，给出代码的具体实现。

## CommandBuilder

在Builder里指定命令名称并创建命令实例（工厂方法）。

package **com.yonyou.udhsearch.morphlines;**

import org.kitesdk.morphline.api.Command;

import org.kitesdk.morphline.api.CommandBuilder;

import org.kitesdk.morphline.api.MorphlineContext;

import com.typesafe.config.Config;

public final class DropHtmlTags implements CommandBuilder {

 @Override

 public Collection<String> getNames() {

 return Collections.singletonList("**htmlDropper**");

 }

 @Override

 public Command build(Config config, Command parent, Command child, MorphlineContext context) {

 return **new HtmlDropper(this, config, parent, child, context);**

 }

}

## Command

在命令实现类里实现doProcess(Record record)方法，在这个方法加入我们需要的处理逻辑，在本例中是删除Html标签及回车换行符。

package **com.yonyou.udhsearch.morphlines**;

import java.util.Collection;

import java.util.Collections;

import java.util.ListIterator;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.kitesdk.morphline.api.Record;

import org.kitesdk.morphline.base.AbstractCommand;

import com.typesafe.config.Config;

private static final class HtmlDropper extends AbstractCommand {

 private static final String regEx\_script = "<script[^>]\*?>[\\s\\S]\*?<\\/script>";

 private static final String regEx\_style = "<style[^>]\*?>[\\s\\S]\*?<\\/style>"; //

 private static final String regEx\_html = "<[^>]+>";

 private static final String regEx\_space = "\\s\*|\t|\r|\n";

 private final String fieldName;

 public HtmlDropper(CommandBuilder builder, Config config, Command parent, Command child, MorphlineContext context) {

 super(builder, config, parent, child, context);

 this.fieldName = getConfigs().getString(config, "**field**");

 validateArguments();

 }

 @Override

 protected boolean doProcess(Record record) {

 ListIterator iter = record.get(fieldName).listIterator();

 while (iter.hasNext()) {

 iter.set(transformFieldValue(iter.next()));

 }

 // pass record to next command in chain:

 return super.doProcess(record);

 }

 /\*\* Transforms the given input value to some output value \*/

 private Object transformFieldValue(Object value) {

 if (value == null) return "";

 String htmlStr = value.toString();

 try {

 Pattern p\_script = Pattern.compile(regEx\_script, Pattern.CASE\_INSENSITIVE);

 Matcher m\_script = p\_script.matcher(htmlStr);

 htmlStr = m\_script.replaceAll("");

 Pattern p\_style = Pattern.compile(regEx\_style, Pattern.CASE\_INSENSITIVE);

 Matcher m\_style = p\_style.matcher(htmlStr);

 htmlStr = m\_style.replaceAll("");

 Pattern p\_html = Pattern.compile(regEx\_html, Pattern.CASE\_INSENSITIVE);

 Matcher m\_html = p\_html.matcher(htmlStr);

 htmlStr = m\_html.replaceAll("");

 Pattern p\_space = Pattern.compile(regEx\_space, Pattern.CASE\_INSENSITIVE);

 Matcher m\_space = p\_space.matcher(htmlStr);

 htmlStr = m\_space.replaceAll("");

 if (htmlStr.indexOf("&nbsp;") != -1)

 htmlStr = htmlStr.replaceAll("&nbsp;", "");

 } catch (Exception e) {

 htmlStr = value.toString();

 }

 return htmlStr;

 }

 @Override

 protected void doNotify(Record notification) {

 LOG.debug("myNotification: {}", notification);

 super.doNotify(notification);

 }

}

# Morphline配置文件

在Command的构造函数里，我们指定了一个属性’ field’，也就是说，我们可以通过field来指定对哪个字段进行操作。除了field属性，还需要在Morphline的配置文件里引入新开发的命令类。在这个例子里，对字段data\_f2进行Html标签删除操作。如下所示：

morphlines : [

 {

 id : morphline1

 importCommands : ["org.kitesdk.morphline.\*\*", "com.ngdata.\*\*", "**com.yonyou.udhsearch.morphlines.\*\***"]

 commands : [

 {

 extractHBaseCells {

 mappings : [

 {

 inputColumn : "data:f1"

 outputField : "data\_f1"

 type : string

 source : value

 }

 {

 inputColumn : "data:f2"

 outputField : "data\_f2"

 type : string

 source : value

 }

 ]

 }

 }

 ***{***

 ***htmlDropper {***

 ***field : data\_f2***

 ***}***

 ***}***

 { logTrace { format : "output record: {}", args : ["@{}"] } }

 ]

 }

]

# Morphline部署运行

运行时，需要保证新开发的命令类在MapReduce作业或Lily NRT Indexer服务的类路径里。

## MapReduce作业

对于MR作业，可直接将打好的jar包包含在命令行里，如下所示：

HADOOP\_CLASSPATH=/usr/lib/hbase/hbase-protocol.jar \

hadoop --config /etc/hadoop/conf jar \

/usr/lib/hbase-solr/tools/hbase-indexer-mr-\*-job.jar --conf \

/etc/hbase/conf/hbase-site.xml --libjars ***/root/tools/MyMorphlineCommands/bin/DropHtmlTags.jar*** \

-D 'mapred.child.java.opts=-Xmx500m' \

--hbase-indexer-file /root/search/hbase\_html\_tags\_dropper/conf/hbase\_html\_tags\_dropper\_mapper.xml --zk-host \

127.0.0.1/solr --collection hbase\_html\_tags\_dropper --go-live --log4j \

/etc/hbase/conf.dist/log4j.properties

## Lily NRT Indexer

将jar包放在服务的lib目录下，即/usr/lib/hbase-solr/lib/，重启服务。