Day22_Sqoop的安装和操作

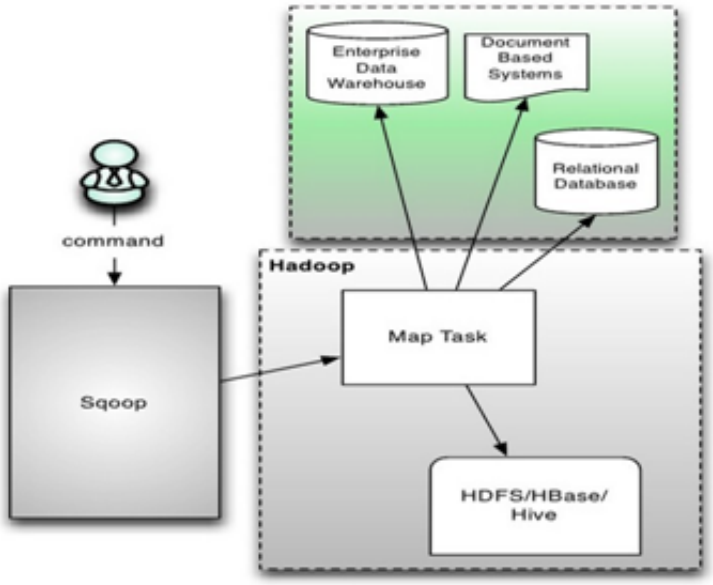
大数据-张军锋 Day22 Sqoop 安装 操作

Sqoop简介

概述

sqoop是Apache顶级项目，主要用来在Hadoop和关系数据库中传递数据。通过sqoop，我们可以方便的将数据从关系数据库导入到HDFS，或者将数据从HDFS导出到关系数据库。

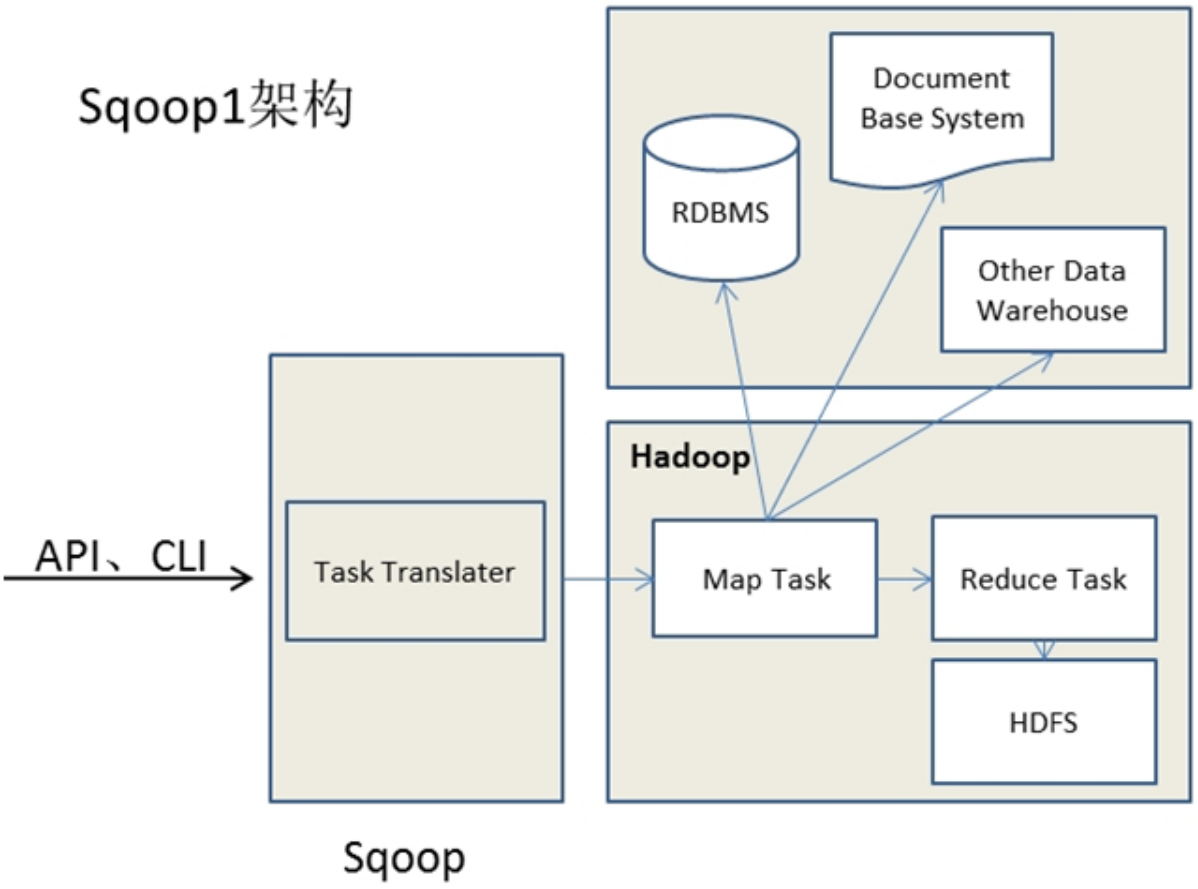
sqoop架构

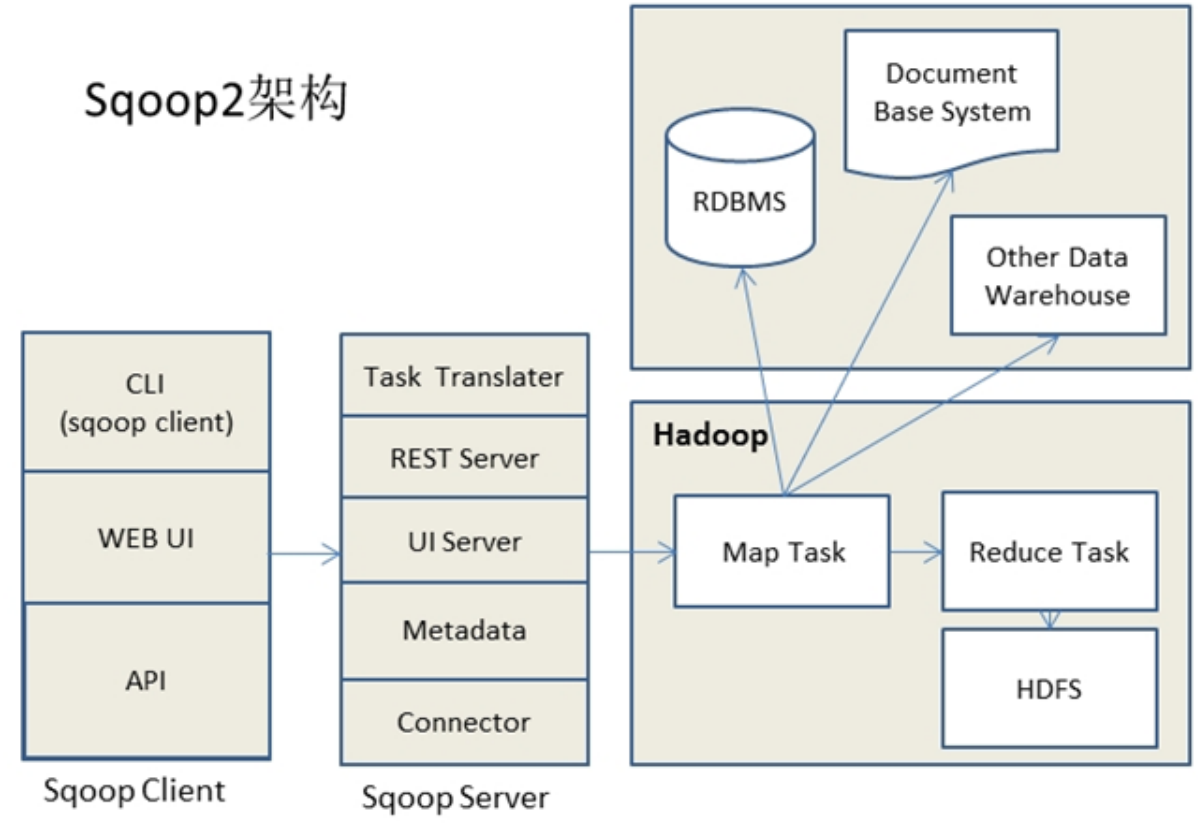


sqoop架构非常简单，其整合了Hive、Hbase和Oozie，通过map-reduce任务来传输数据，从而提供并发特性和容错。

sqoop1和sqoop2区别

这两个版本是完全不兼容的，其具体的版本号区别为1.4.x为sqoop1，1.99x为sqoop2。
sqoop1和sqoop2在架构和用法上已经完全不同。
在架构上，sqoop2引入了sqoop server（具体服务器为tomcat），对connector实现了集中的管理。其访问方式也变得多样化了，其可以通过REST API、JAVA API、WEB UI以及CLI控制台方式进行访问。另外，其在安全性能方面也有一定的改善，在sqoop1中我们经常用脚本的方式将HDFS中的数据导入到mysql中，或者反过来将mysql数据导入到HDFS中，其中在脚本里边都要显示指定mysql数据库的用户名和密码的，安全性做的不是太完善。在sqoop2中，如果是通过CLI方式访问的话，会有一个交互过程界面，你输入的密码信息不被看到。下图是sqoop1和sqoop2简单架构对比：


Sqoop1架构
Document Base System
RDBMS
Other Data Warehouse
Hadoop
API、CLI
Task Translater
Map Task
Reduce Task
HDFS
Sqoop


Sqoop2架构
Document Base System
RDBMS
Other Data Warehouse
Hadoop
CLI (sqoop client)
WEB UI
API
Task Translater
REST Server
UI Server
Metadata
Connector
Map Task
Reduce Task
HDFS
Sqoop Client
Sqoop Server

Sqoop的安装

1. **下载sqoop**
   下载地址：https://mirrors.tuna.tsinghua.edu.cn/apache/sqoop/
2. **解压sqoop安装文件**
   tar -zxvf sqoop-1.99.7-bin-hadoop200.tar.gz
3. **配置sqoop环境变量**

```
#sqoop
export SQOOP_HOME=/opt/Software/Sqoop/sqoop-1.99.7-bin-hadoop200
export PATH=$PATH:$SQOOP_HOME/bin
export LOGDIR=$SQOOP_HOME/logs
export BASEDIR=$SQOOP_HOME/base
```

最后别忘了使 /etc/profile 生效

4. **hadoop的core-site.xml里添加配置**

```
<property>
    <name>hadoop.proxyuser.sqoop2.hosts</name>
    <value>*</value>
</property>

<property>
    <name>hadoop.proxyuser.sqoop2.groups</name>
    <value>*</value>
</property>
```

远程拷贝到另外两个节点

```
scp core-site.xml root@slaver1:/opt/Software/Hadoop/hadoop-2.7.3/et
c/hadoop/
scp core-site.xml root@slaver2:/opt/Software/Hadoop/hadoop-2.7.3/et
c/hadoop/
```

5. **修改conf下的sqoop.properties**

```
143 # Hadoop configuration directory
144 org.apache.sqoop.submission.engine.mapreduce.configuration.directory=/opt/S
```

```
org.apache.sqoop.submission.engine.mapreduce.configuration.director
y=/opt/Software/Hadoop/hadoop-2.7.3/etc/hadoop/
```

6. **初始化Sqoop**
   `sqoop2-tool upgrade`

7. **检测是否初始化成功**
   `sqoop2-tool verify`

```
Verification was successful.
Tool class org.apache.sqoop.tools.tool.VerifyTool has finished correctly.
```

8. **启动Sqoop服务器**
   `sqoop2-server start`
9. **启动客户端**
   `sqoop2-shell`

基本操作 & Sqoop下的object

操作参考文档：http://sqoop.apache.org/docs/1.99.7/

基本信息

`show server -a`

```
sqoop:000> show server -a
Server host: localhost
Server port: 12000
Server webapp: sqoop
```

`show version -a`

```
sqoop:000> show version -a
client version:
  Sqoop 1.99.7 source revision 435d5e61b922a32d7bce567fe5fb1a9c0d9b1bbb
  Compiled by abefine on Tue Jul 19 16:08:27 PDT 2016
server version:
  Sqoop 1.99.7 source revision 435d5e61b922a32d7bce567fe5fb1a9c0d9b1bbb
  Compiled by abefine on Tue Jul 19 16:08:27 PDT 2016
API versions:
  [v1]
```

核心对象

connector

connector是sqoop当前支持的存储系统连接配置类型
connector Name是sqoop默认支持的数据连接类型
Supported Direction中From/to表示连接方式，From表示数据来源(导入)，To表示数据去向(导出)

当前sqoop支持的连接器和连接类型，在此仅作查看

```
show connector
```


```
+------------------------+---------+------------------------------------------------------------+----------------------+
|          Name          | Version |                           Class                            | Supported Directions |
+------------------------+---------+------------------------------------------------------------+----------------------+
| generic-jdbc-connector | 1.99.7  | org.apache.sqoop.connector.jdbc.GenericJdbcConnector       | FROM/TO              |
| kite-connector         | 1.99.7  | org.apache.sqoop.connector.kite.KiteConnector              | FROM/TO              |
| oracle-jdbc-connector  | 1.99.7  | org.apache.sqoop.connector.jdbc.oracle.OracleJdbcConnector | FROM/TO              |
| ftp-connector          | 1.99.7  | org.apache.sqoop.connector.ftp.FtpConnector                | TO                   |
| hdfs-connector         | 1.99.7  | org.apache.sqoop.connector.hdfs.HdfsConnector              | FROM/TO              |
| kafka-connector        | 1.99.7  | org.apache.sqoop.connector.kafka.KafkaConnector            | TO                   |
| sftp-connector         | 1.99.7  | org.apache.sqoop.connector.sftp.SftpConnector              | TO                   |
+------------------------+---------+------------------------------------------------------------+----------------------+
```

driver

驱动配置信息，在此仅作查看

```
show driver
```

```
sqoop:000> show driver
0    [main] WARN  org.apache.hadoop.util.NativeCodeLoader  - Unable to load native
ing builtin-java classes where applicable
Driver specific options:
Persistent id: 8
    Job config 1:
      Name: throttlingConfig
      Label: Throttling resources
      Help: Set throttling boundaries to not overload your systems
      Input 1:
        Name: throttlingConfig.numExtractors
        Label: Extractors
        Help: Number of extractors that Sqoop will use
        Type: INTEGER
        Sensitive: false
        Editable By: ANY
        Overrides:
      Input 2:
        Name: throttlingConfig.numLoaders
        Label: Loaders
        Help: Number of loaders that Sqoop will use
        Type: INTEGER
        Sensitive: false
```

link

数据导入导出配置对象
link里面配置具体的存储连接，它是以connector作为类型的
比方说某个jdbc数据库的连接，某个hdfs集群连接等

job

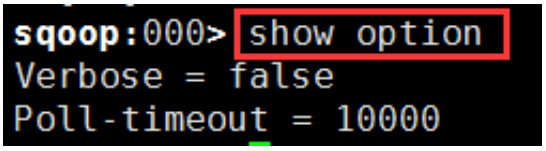
job里面配置一次导入导出过程的全部细节信息，它是以link作为输入和输出的
通常用from link1 to link2 表示把link的数据导入到link2上
导出数据的具体制定在job里面配置

submission

查看当前已提交的sqoop导入导出任务

参数信息

option



权限信息

role
principal|
privilege

JDBC to HDFS

创建mysql link

`create link -c generic-jdbc-connector`

```
sqoop:000> create link -c generic-jdbc-connector
0     [main] WARN  org.apache.hadoop.util.NativeCodeLoader  -
ing builtin-java classes where applicable
Creating link for connector with name generic-jdbc-connector
Please fill following values to create new link object
Name: localmysql

Database connection

Driver class: com.mysql.jdbc.Driver
Connection String: jdbc:mysql://localhost:3306/hive
Username: root
Password: ****
Fetch Size:
Connection Properties:
There are currently 0 values in the map:
entry#

SQL Dialect

Identifier enclose: `
```

connection Properties：数据库的配置参数，可以不写

Identifier enclose：标识符的封装符号，mysql中使用反引号或者空格作为标识符 ，sqoop中默认的是逗号。

创建hdfs link

`create link -c hdfs-connector`

```
sqoop:000> create link -c hdfs-connector
Creating link for connector with name hdfs-connector
Please fill following values to create new link object
Name: bd14hdfs

HDFS cluster

URI: hdfs://master:9000
Conf directory: /opt/Software/Hadoop/hadoop-2.7.3/etc/hadoop
Additional configs::
There are currently 0 values in the map:
entry#
New link was successfully created with validation status OK and name bd14hdfs
```

创建job

`create job -f localmysql -t bd14hdfs`

```
sqoop:000> create job -f localmysql -t bd14hdfs
Creating job for links with from name localmysql and to name bd14hdfs
Please fill following values to create new job object
Name: flocalmysqltobd14hdfs

Database source

Schema name: hive
Table name: COLUMNS_V2
SQL statement:
Column names:
There are currently 0 values in the list:
element#
Partition column:
Partition column nullable:
Boundary query:

Incremental read
```

```
Check column:
Last value:

Target configuration

Override null value:
Null value:
File format:
  0 : TEXT_FILE
  1 : SEQUENCE_FILE
  2 : PARQUET_FILE
Choose: 0
Compression codec:
  0 : NONE
  1 : DEFAULT
  2 : DEFLATE
  3 : GZIP
  4 : BZIP2
  5 : LZO
  6 : LZ4
  7 : SNAPPY
  8 : CUSTOM
Choose: 0
```

```
Custom codec:
Output directory: /bd14/fromSqoop
Append mode: true

Throttling resources

Extractors:
Loaders:

Classpath configuration

Extra mapper jars:
There are currently 0 values in the list:
element#
New job was successfully created with validation status OK  and name flocalmysqltobd14hdfs
```

启动jobhistory

查看提交的状态信息，需要用到jobhistory服务，下面是启动过程

`mr-jobhistory-daemon.sh start historyserver`

```
[root@master sbin]# mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /opt/Software/Hadoop/hadoop-2.7
[root@master sbin]# jps
2832 DataNode
4112 Jps
3922 SqoopShell
3798 SqoopJettyServer
3176 ResourceManager
2730 NameNode
2955 SecondaryNameNode
4075 JobHistoryServer
```

运行job

`start job -n flocalmysqltobd14hdfs`

application_1510121105073_0002	root	Sqoop: flocalmysqltobd14hdfs	MAPREDUCE	default	Wed Nov 8 14:08:02 +0800 2017	Wed Nov 8 14:16:11 +0800 2017	FINISHED	FAILED		History	N/A

运行失败，因为运行在集群上是分布式运行，可能是因为子节点没有安装mysql数据库，所以link中的连接需要修改一下，把`Connection String: jdbc:mysql://localhost:3306/hive`改为`jdbc:mysql://master:3306/hive`

```
sqoop:000> start job -n flocalmysqltobd14hdfs
Exception has occurred during processing command
Exception: org.apache.sqoop.common.SqoopException Message: GENERIC_JDBC_CONNECTOR_0001:Unable to get a connection -
```

如果启动出现上面的错误，删除mysql下user表中的用户，只留localhost和%，如下图所示，删除完之后就`FLUSH PRIVILEGES;`

Host	User	Password	Select_priv	Insert_priv	Update_priv
localhost	root	*81F5E21E35407D884A6C	Y	Y	Y
%	root		Y	Y	Y

> **注意：**如果还是有错，看一下自己的mysql是否设置了免密登录，把link中的密码改为空试试

HDFS to JDBC

```
sqoop:000> create job -f bd14hdfs -t localmysql
Creating job for links with from name bd14hdfs and to name localmysql
Please fill following values to create new job object
Name: hdfstomysqlbd14

Input configuration

Input directory: /bd14/exptomysql
Override null value:
Null value:

Incremental import

Incremental type:
  0 : NONE
  1 : NEW_FILES
Choose: 0
Last imported date:

Database target
```

```
Schema name: from_sqoop
Table name: users
Column names:
There are currently 0 values in the list:
element#
Staging table:
Clear stage table:

Throttling resources

Extractors:
Loaders:

Classpath configuration

Extra mapper jars:
There are currently 0 values in the list:
element#
New job was successfully created with validation status OK
```

我们在hdfs上创建文件导入目录文件夹 hdfs dfs -mkdir /bd14/exptomysql
将数据放入到此此件夹内，就可以完成导入操作了，但是到目前为止sqoop只支持csv格式的文件导入导出，因 此，我们需要将数据转换成csv格式，再放入到目录下
在mysql中创建数据库 create database from_sqoop和表

Java操作Sqoop

创建link

```
package com.bd14.zjf;

import java.util.List;

import org.apache.sqoop.client.SqoopClient;
import org.apache.sqoop.model.MConfig;
import org.apache.sqoop.model.MInput;
import org.apache.sqoop.model.MLink;
import org.apache.sqoop.model.MLinkConfig;
import org.apache.sqoop.validation.Status;

public class SqoopTest {
    // sqoop的服务器url地址
    private final String URL = "http://master:12000/sqoop/";
    // 创建客户端对象
    private SqoopClient client = new SqoopClient(URL);

    // 创建一个link
    public void createLink() {
        // 创建一个generic-jdbc-connector类型的link
        MLink link = client.createLink("generic-jdbc-connector");
        // link.getConnectorLinkConfig()获取connector的link配置信息
        link.setName("window_mysql");
        MLinkConfig linkConfig = link.getConnectorLinkConfig();
        // 根据配置项名称获取配置项
        // 取出所有的配置项
        List<MConfig> list = linkConfig.getConfigs();
        for (MConfig mConfig : list) {
            List<MInput<?>> inputs = mConfig.getInputs();
            for (MInput<?> mInput : inputs) {
                System.out.println(mInput);
            }
        }
        // MLinkConfig的相关配置项并且设置上相应配置值
        linkConfig.getStringInput("linkConfig.connectionString").se
tValue("jdbc:mysql://192.168.89.1:3306/test");
        linkConfig.getStringInput("linkConfig.jdbcDriver").setValu
e("com.mysql.jdbc.Driver");
        linkConfig.getStringInput("linkConfig.username").setValu
e("root");
        linkConfig.getStringInput("linkConfig.password").setValu
e("root");
        linkConfig.getStringInput("dialect.identifierEnclose").setV
alue("`");
        // 保存到sqoop服务器
        Status status = client.saveLink(link);
        if (status.canProceed()) {
            System.out.println("创建link" + link.getName() + "成功");
```

```
            } else {
                System.out.println("创建link" + link.getName() + "失败");
            }

        }

        public static void main(String[] args) {
            SqoopTest sqoopTest = new SqoopTest();
            sqoopTest.createLink();
        }
    }
```

创建job

```
public void createJob() {
    MJob job = client.createJob("hdfslink", "window_mysql");
    job.setName("hdfs2_Window");
    MFromConfig fromjobConfig = job.getFromJobConfig();
    MToConfig toJobConfig = job.getToJobConfig();
    // 列举出配置项信息
    /*List<MConfig> configs = fromjobConfig.getConfigs();
    for (MConfig mConfig : configs) {
        List<MInput<?>> inputs = mConfig.getInputs();
        for (MInput<?> mInput : inputs) {
            System.out.println(mInput);
        }
    }*/
    fromjobConfig.getStringInput("fromJobConfig.inputDirectory").se
tValue("/bd14/exptomy sql");
    // 列举出配置项信息
    /*System.out.println("----------------");
    List<MConfig> configs2 = toJobConfig.getConfigs();
    for (MConfig mConfig : configs2) {
        List<MInput<?>> inputs = mConfig.getInputs();
        for (MInput<?> mInput : inputs) {
            System.out.println(mInput);
        }
    }*/
    toJobConfig.getStringInput("toJobConfig.schemaName").setValu
e("xs");
    toJobConfig.getStringInput("toJobConfig.tableName").setValue("u
sers");
    Status status = client.saveJob(job);
    if (status.canProceed()) {
        System.out.println("创建job " + job.getName() + "成功");
    } else {
        System.out.println("创建job " + job.getName() + "失败，请检查
配置");
    }
}
```

启动job

```
public class SqoopTest {
    // sqoop的服务端url地址
    private final String URL = "http://master:12000/sqoop/";
    // 创建客户端对象
    private SqoopClient client = new SqoopClient(URL);

    // 创建  个link
    public void createLink() {
        MLink link = client.createLink("generic-jdbc-connector");
        link.setName("window_mysql");
        // link.getConnectorLinkConfig()获取connector的link配置信息
        MLinkConfig linkConfig = link.getConnectorLinkConfig();
        // 取出所有的配置项
        /*List<MConfig> list = linkConfig.getConfigs();
        for (MConfig mConfig : list) {
            List<MInput<?>> inputs = mConfig.getInputs();
            for (MInput<?> input : inputs) {
                System.out.println(input);
            }
        }*/
        // MLinkConfig相关配置项名称设置配置项
        linkConfig.getStringInput("linkConfig.jdbcDriver").setValu
e("com.mysql.jdbc.Driver");
        linkConfig.getStringInput("linkConfig.connectionString").se
tValue("jdbc:mysql://192. 168.6.81:3306/test");
        linkConfig.getStringInput("linkConfig.username").setValu
e("root");
        linkConfig.getStringInput("linkConfig.password").setValu
e("root");
        linkConfig.getStringInput("dialect.identifierEnclose").setV
alue(" ");
        Status status = client.saveLink(link);
        if (status.canProceed()) {
            System.out.println("创建link " + link.getName() + "成
功");
        } else {
            System.out.println("创建link " + link.getName() + "失
败,请检查配置项");
        }
    }

    public void createJob() {
        MJob job = client.createJob("hdfslink", "window_mysql");
        job.setName("hdfs2_Window");
        MFromConfig fromjobConfig = job.getFromJobConfig();
        MToConfig toJobConfig = job.getToJobConfig();
        // 列举出配置项信息
        /*List<MConfig> configs = fromjobConfig.getConfigs();
```

```
        for (MConfig mConfig : configs) {
            List<MInput<?>> inputs = mConfig.getInputs();
            for (MInput<?> mInput : inputs) {
                System.out.println(mInput);
            }
        }*/
        fromjobConfig.getStringInput("fromJobConfig.inputDirector
y").setValue("/bd14/exptomy sql");
        // 列举出配置项信息
        /*System.out.println("----------------");
        List<MConfig> configs2 = toJobConfig.getConfigs();
        for (MConfig mConfig : configs2) {
            List<MInput<?>> inputs = mConfig.getInputs();
            for (MInput<?> mInput : inputs) {
                System.out.println(mInput);
            }
        }*/
        toJobConfig.getStringInput("toJobConfig.schemaName").setVal
ue("xs");
        toJobConfig.getStringInput("toJobConfig.tableName").setValu
e("users");
        Status status = client.saveJob(job);
        if (status.canProceed()) {
            System.out.println("创建job " + job.getName() + "成功");
        } else {
            System.out.println("创建job " + job.getName() + "失败，请
检查配置");
        }
    }

    public static void main(String[] args) {
        SqoopTest st = new SqoopTest();
        // st.createLink();
        // st.createJob();
        // 启动job
        st.client.startJob("hdfs2_Window");
    }
}
```