EBase-Filesystem v0.6使用简介

# 简介

EBase-filesystem是一个文件系统抽象层接口，接口设计采用了Hadoop Filesystem v1.2.0，并继承了其HDFS、本地文件系统、ftp、Amazon S3、InMemory等实现。EBase-filesystem中增加了MongoDB的实现，可以使用相同的接口操作MongoDB、HDFS、本地文件系统等存储，只需修改一处配置即可。

EBase-filesystem v0.6依赖于hadoop-corev1.2.0和mongo-java-driver-2.11.2。

# 主要配置

以下配置要放到java classpath中。

* core-default.xml：

fs.default.name：配置底层实现。

mongodb:/// - MongoDB实现

hdfs:/// - HDFS实现

file:/// - 本地文件系统

后面不需要写具体路径，具体路径在各自实现的配置文件中设置。

* core-site.xml：HDFS实现

fs.defaultFS：HDFS访问路径，默认为hdfs://${hosts}:8020

* hdfs-default.xml：HDFS默认配置，如无特殊需要，不需更改
* hdfs-site.xml：对HDFS进行二次开发时，可以把配置写在这里。
* mongo-site.xml：MongoDB实现配置，请参照样例文件中每个配置项的<description>字段进行修改。其中设置MongoDB库地址的参数是“fs.defaultFS”。

# API

* 文件系统用户

运行此文件系统程序的程序，必定是在操作系统的一个用户下运行的。文件系统会取得这个用户和其所在组，作为其操作文件对象的所属用户和组。

* 创建文件系统对象

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

可以在任意系统中随时创建本地文件系统，这个方法与配置文件中的实现配置无关，如下：

FileSystem fs = FileSystem.*getLocal*(conf);

上面的方法可以创建配置文件中创建的文件系统对象，还可以通过自定义uri创建依赖其他存储系统的文件系统对象：

URI uri1 = URI.*create*("mongodb://127.0.0.1:27017/gaint/storage");

FileSystem fs1 = FileSystem.*get*(uri1, conf);

URI uri2 = URI.*create*("hdfs://127.0.0.1:8020");

FileSystem fs2 = FileSystem.*get*(uri2, conf);

这样，程序中就可以同时存在底层存储是mongodb和hdfs实现的文件系统对象。

* 关闭文件系统操作

文件系统在创建后，会保存在框架的缓存中，下次再使用同一个文件系统对象时，会直接从缓存里去，如果调用了关闭文件系统方法，会把这个文件系统对象从缓存中删除。

fs.close();

fs.*closeAll*();

closeAll方法会删除所有缓存中的操作系统对象。

* 绝对路径和相对路径

文件系统支持绝对路径和相对路径操作，可以通过getWorkingDirectory()来获取当前所在的工作路径，通过setWorkingDirectory()方法设置当前工作路径。

文件系统创建后，默认的当前工作路径是${URI}/user/${username}，如果是本地文件系统，就是当前运行系统所在根路径。

例如，fs是刚创建好的一个文件系统对象。

fs.getWorkingDirectory() == ${URI}/user/${username}

fs.setWorkingDirectory(new Path(“../”)) == ${URI}/user

fs.setWorkingDirectory(new Path(“/test/test”)) == ${URI}/test/test

如果要在系统中复用文件系统对象，那么请使用绝对路径来操作，以免多个线程使用造成混乱。

* Path对象

此对象时文件系统抽象的一个路径对象，文件系统方法只接受这个对象作为参数进行操作。他提供了格式校验、格式转换、获取父路径、路径深度、协议转换等功能。这个类屏蔽了一些协议、格式等于具体实现有关的细节。创建方法：

Path path = new Path(“/test”);

* 文件系统操作方法

下面是文件系统的具体操作方法，其中很多都具有多个重载方法，大部分与普通文件系统操作大同小异，就不一一详细介绍。

mkdirs – 创建文件夹，如果父文件夹不存在，会递归创建

copyFromLocalFile – 从本地文件系统拷贝文件到存储

moveFromLocalFile – 从本地文件系统迁移文件到存储，会删除本地文件

copyToLocalFile – 从存储拷贝数据到本地文件系统

moveToLocalFile – 迁移数据到本地文件系统，删除存储中的文件

getUsed – 返回系统中所有文件的大小之和

getFileStatus – 获取文件状态，包括文件所有者、是否路径、复制数等信息…

listStatus – 如果是文件，获取状态信息；如果是路径，获取直接子节点状态信息

create – 创建文件，返回一个OutputStream，向流写入数据

append – 追加文件内容，返回一个已有文件的OutputStream

delete – 删除文件或路径，如果是路径，第二个参数表示是否删除子目录，如果为true，则删除所有子目录，如果为false，并且有子目录，则不允许删除，并抛出异常。

Exists – 判断文件或文件夹是否存在

getUri – 获取当前文件系统的uri

open – 打开一个文件，返回一个InputStream对象

rename – 对文件或文件夹改名。如果改名的文件夹不存在，会递归创建

checkFilenameValid – 校验一个文件或文件夹名称是否合法

createNewFile – 创建一个空文件，如果文件已存在或创建失败，返回false

deleteOnExit – 把文件或文件夹标记为删除，当文件系统对象被关闭时删除。当JVM宕掉时，也会删除这些文件。

isFile – 是否为文件

getHomeDirectory – 返回用户默认根（Home）路径