1. RSS 介绍
   1. RSS
   2. 抓取RSS所用的Java Lib

RSS Lib For Java（rsslib4j-0.2.jar） 从官网抄点他的特性

调用方法：

// 新建一个RSSHandler对象

RSSHandler remoteRSSHandler = **new** RSSHandler();

// 解析RSS数据，获取的数据存入remoteRSSHandler中

RSSParser.*parseXmlFile*(**new** URL(“RSS的地址”), remoteRSSHandler, **false**);

// 获取RSS channel

RSSChannel channel = remoteRSSHandler.getRSSChannel();

1. 程序总体框架
   1. 功能描述

这是一个服务器程序。程序实现了按照配置文件（rss.cfg）的描述连接Cassandra数据库，并按规定的时间间隔抓取网络上的RSS数据，经过去重处理后。将RSS数据写入数据库中。

另外程序同时实现一个基于网络的命令行接口。可以通过一个Client连接此服务器程序，实时监控服务器程序的状态，以及对服务器进行一些日常的管理。

* 1. 框架

程序有两个线程，一个线程负责数据的抓取，另一个负责基于网络的命令行接口。

数据抓取线程在RssDownloader. DownloaderTimerTask.java中实现

命令行接口在RssDownloader. Utility. CmdServer.java中实现。

从下图中可以了解整个程序的所用到的包和类

（图以后发给你）

1. 数据结构
   1. 主要数据结构
      1. RssSource Package

这个包里存在两个类，一个是RssSource.java这个类主要用于描述单个Rss源。另一个是RssSourceList.java这个类用于存储RssSource的列表。在程序启动的时候，会从Cassandra数据库中，读取Rss源到这个类对象中。

* + 1. RssItem Package

这个包里也存在两个类，一个是RssItem.java 这个类主要用于描述单个RssItem。另一个类是RssItemList.java。这个主要用于存放，操作从网上下载下来的Rss Item. 本程序是将每一个从网上下载的RSS Item实时存入数据库中。但是由于众多RSS新闻网站存在大量的重复新闻。那么我们需要暂存N天的RSS Item用于鉴别RSS Item是否有重复。

所以在这个类中有两个重要的数据结构。rssHashTable用于暂存放从网上下载下来的RssItem。rssHashTable是HashMap 数据类型，其Key是Rss Item的SimHash Value，而其Value是每一个RSS Item对象

另一个 rssDuplicated用于存放，通过甄别所发现的重复RssItem项。rssDuplicated也是HashMap数据类型，其Key是一群重复RSS Item中主RSS Item的SimHash值，而其value是一个用于存放一群重复RSS Item的链表。

* 1. 数据流程
     1. 数据基本流程

RSS数据从网络上被下载下来到存入数据库，经历了：下载-> 去重判断-> 存入数据库。下面的伪代码将描述这一过程，这部分代码在DownloaderTimerTask.eachRun()中实现

While （在RSS源列表中还有待处理的RSS源）{

获取RSS源；

If (RSS 源 的pubdate较上次新) {

While (在Rss Item列表中还有待处理RSS Item) {

计算RSS Item的SimHash值.

新建一个Rss Item对象.

ret = 将新的RSS Item添加到Rss Item列表中

If（ret ok） { // RSS 成功添加入RSS Item 列表

更新统计数据

}

}

}

}

* + 1. 数据去重流程

由于RSS的特性，不同的RSS可能会有大量相同或者类似的RSS Item。去重的处理主要是依靠比较两两Rss Item的simHash值。如果其simhash的hamming距离小于设定值，则可判断这是两个类似的Rss Item。那么当每一个新的RSS Item被获取后，就需要对整个rssHashTable进行遍历，比较新RSS Item和rssHashTable中每个rss item的海明距离。这个过程是本算法是本程序最耗时的过程，而且随着rssHashTable的增长，这一过程会变得越来越慢。这部分代码具有可提高的余地。需要对simhash做更有效的查找和比较。Simhash是矢量向量，如果可以对其进行矢量排序，那么将大大减少其查找的时间消耗。这部分工作将在去重算法部分详细讨论。

If（新的RSs Item不在rssHashTable中） {

While (遍历rssHashTable) {

If(computeHammingDistance(newRssItem，eachRssItem) < pre-setValue) {

newRssItem为重复RSS Item，进行重复RSS Item处理

}

}

}

1. 去重算法
   1. simhash算法

在网上找点东西贴上

* 1. 在本程序中的应用

这本程序中，我们借用了xxx项目（具体需要网络才能查）中的simhash算法。通过下面的函数进行simhash的计算：

public static long computeOptimizedSimHashForBytes(byte[] data,int offset,int length)；

然后在rss的去重部分，又通过下面的函数进行hamming距离的计算

public static int hammingDistance(long hash1, long hash2)；

1. 数据库
   1. Cassandra 介绍
   2. 数据库schema

（回头我在想想怎么写，先把它全部贴上）

1. 命令行接口

命令行接口为一个典型的Server/Client TCP/IP通信模型。它独立运行于一个线程中。Server通过Accept函数等待client的连接。当有client向服务器发起请求后，新建一个线程用于对其的服务操作。

下面为其伪代码：

While(1) {

clientSocket = accept(serverSocket);

Thread clientThread = new Thread(clientSocket);

clientThread.start()

}

在每个client线程中，有一个命令行解析接口，负责处理每一条命令。其伪代码如下：

While（inputCmd=inputIO.getLine() != null） {

Foreach(cmd: commandList) { // 在command List中遍历每一个command

If (inputCmd.compare(cmd.cmdstr)) { // 如果输入的cmd和其中一个cmd一致

cmd.exe(); // 执行该command

}

}

}