TwitterVane

Installation Guide

19 February 2013

Introduction

About TwitterVane

TwitterVane is a tool for leveraging the power of the crowd to select websites for web archiving. It enables curators to define web collections and identify trending Tweets and associated URLs that are relevant to those collections.

About this document

This document is the TwitterVane System Installation Guide. It describes how install, configure and deploy the TwitterVane web application.

Where to find more information

The public resources for TwitterVane, including the source code, binaries and documentation, are available through the TwitterVane Open Source GitHub project:

https://github.com/ukwa/twittervane

Contents of this document

Following this introduction, the guide includes the following sections:

- **Getting Started** (page 4) — covers prerequisites, supported platforms, other platforms, and optional prerequisites for using TwitterVane
- **Setting up the database** (page 5) — procedures for setup using PostgreSQL 8.2
- **Application Server Setup** (page 6) — procedures for deploying TwitterVane to Tomcat, includes configuration options and troubleshooting

Getting Started

The following section explains how to get TwitterVane up and running.

Prerequisites

The following are required to successfully install and run TwitterVane:

- Java 1.6 JDK or above
- Apache Tomcat 6.X or above (the application has been tested on Tomcat 6.0.35)
- A Postgresql 8.2 database server.

Supported platforms

The following platforms have been used during the development of the TwitterVane:

- Red Hat Enterprise Linux Server release 6.3 (Santiago).

Other platforms

The following platform was used during the development of the Web Curator tool but is not explicitly supported:

- Windows XP.

Setting up the database

Setup using PostgreSQL 8.2

This guide assumes you have installed and configured Postgresql 8.2 prior to setting up the TwitterVane database and schema.

1 Create the database and role:

```
CREATE DATABASE "twittervane" WITH ENCODING='UTF8';

\c twittervane

CREATE ROLE usr_tv LOGIN PASSWORD 'password' NOINHERIT
VALID UNTIL 'infinity';

grant all PRIVILEGES on DATABASE twittervane to
usr_tv;
```

2 Generate the database schema:

Run the following scripts using the Postgres SQL command line, psql:

```
  psql -f twittervane -f /twittervane/TweetView/sql/
schema.sql

  psql -f twittervane -f /twittervane/TweetView/sql/
schema_patch1.sql
```

3 The Postgres JDBC driver is included in the GitHub repository (see section: “Where to find more information”) under the /TweetView/lib/ directory.

- The Postgres driver is called postgresql-8.2-507.jdbc4.jar
- The driver will need to be placed into the $TOMCAT_HOME/common/lib/ directory.
- Also required in the $TOMCAT_HOME/common/lib/ directory is the jta.jar

Overview

There are three major components to the deployment of the TwitterVane:

- the **TweetView** component (TweetView.war)
- the **TwitterStreamAgent** component (TweetStreamAgent.war)
- the **TweetAnalyser** component (TweetAnalyser.war).

Each of these three components must be deployed for TwitterVane to be fully functional.

The **TweetView** component provides the management and reporting features that curators use to create and report on Web Collections.

The **TweetStreamAgent** provides the UI and services for managing the inbound Tweet data from Twitter.

The **TweetAnalyser** performs URL expansion on shortened URLs associated with a Tweet, resolves a Tweet to a web collection, and manages the storage of Tweets.

All three components can be deployed to different Tomcat instances on a single server or distributed servers.

Recommended Deployment Configuration

The recommended deployment configuration for TwitterVane consists of using a separate Tomcat instance for each of the three components **TweetView**, **TweetStreamAgent** and **TweetAnalyser**.

This isolates application server resources for each component and balances system resources across all components. This means that when one component experiences high load, the other components are not adversely affected.

This also enables the required security constraints to be applied to each component. For example, the **TweetStreamAgent** and **TweetAnalyser** management UIs may need to be secured for administrator access only.

Deploying TwitterVane to Tomcat

To deploy TwitterVane to Tomcat:

1 Make sure you have installed and configured both Java 1.6 JDK and Apache-Tomcat 6.X successfully.

2 Deploy the WAR files into Tomcat. The simplest deployment is to deploy all three WAR files into the same Tomcat container.

   – You can copy the WAR files into the $TOMCAT_HOME/ webapps/ directory.
   – Provided Tomcat is configured correctly, when you start Tomcat the WAR files will be exploded and the application will start.

3 Shut down Tomcat once the WAR files have been extracted. This will allow you to modify the configuration files in the following steps.

Configure the Application Context

The curator user interface for TwitterVane is provided by the **TweetView** component. To allow curators to access **TweetView** via the URL:

http://servername:port/TwitterVane/

the application context for **TweetView** must be configured.

Edit the server.xml file in $TOMCAT_HOME/conf and add the following docBase and path properties within the <Host …> element:

```
     <?xml version="1.0" encoding="UTF-8"?>
<Host appBase="webapps" … >

      <Context
              docBase="/path/to/tomcat/home/webapps/TweetView"
     path="/twittervane" reloadable="true" >
      </Context>

</Host>
```

Configure the RMI Connection

The **TweetStreamAgent** communicates with the **TweetAnalyser** via Spring RMI and acts as a client. The client connection is configured in the following file:

$TOMCAT_HOME\webapps\TweetStreamAgent\WEB-INF\spring-servlet.xml

```
<!-- RMI client definition for the tweetAnalyserService
--> <bean id="tweetAnalyserService"
class="org.springframework.remoting.rmi.RmiProxyFactory
Bean">
<property name="serviceUrl" value="rmi://
localhost:1099/TweetAnalyserService"/>
<property name="serviceInterface"
value="uk.bl.wap.crowdsourcing.agent.TweetAnalyserServi
ce"/>
</bean>
```

Ensure that the hostname value is correct for the “serviceUrl” property.

The serviceURL includes an RMI registry port which is created by the **TweetAnalyser**. The port is configured in `RmiServiceExporter` bean in the following file:

$TOMCAT_HOME\webapps\TweetAnalyser\WEB-INF\spring-servlet.xml

```
<!-- RMI service definition for the tweetAnalyserService
-->
<bean
class="org.springframework.remoting.rmi.RmiServiceExporte
r">
 <!-- does not necessarily have to be the same name as
 the bean to be exported -->
  <property name="serviceName"
value="TweetAnalyserService"/>
  <property name="service" ref="tweetAnalyserService"/>
  <property name="serviceInterface"
value="uk.bl.wap.crowdsourcing.agent.TweetAnalyserService
"/>
  <!-- defaults to 1099 -->
  <property name="registryPort" value="1099"/>
</bean>
```

Configure the Analyser Options

There are two analyser options that affect how the analysis is processed:

Top Urls

To limit the number of unnecessary URL expansions that are requested from the URL shortener services, a “Top N” optimisation has been implemented. This restricts the number of expansions requested to the "N most frequently occurring" URLs. This limit is controlled by the following configuration option in the above file:

```
<bean id="tweetAnalyserService"
class="uk.bl.wap.crowdsourcing.agent.TweetAnalyserService
Impl">
<!-- defines the most popular n-URLs -->
<property name="topUrls">
   <value type="java.lang.Integer">10</value></property>
    …
</bean>
```

Process Batch Size

When the **TweetAnalyser** runs, it loads unprocessed Tweets into memory to resolve the Tweet to a web collection and coordinate the URL expansion.
If there are a significant number of Tweets to be processed (ie: 50,000+) the application will exhaust its heap space. To avoid this scenario, Tweets are loaded into memory in batches for processing. The batch size is controlled by the following property in the above file:

```
<bean id="tweetAnalyserService"
class="uk.bl.wap.crowdsourcing.agent.TweetAnalyserService
Impl">
…
  <!-- defines the number of tweets that will be loaded
into memory for processing -->
  <property name="processBatchSize">
   <value type="java.lang.Integer">200</value>
  </property>
</bean>
```

WARNING: Setting the `processBatchSize` to a high value will cause **TweetStreamAgent** threads to wait when making the RMI call to trigger the **TweetAnalyser**. These threads will queue and resume when the analysis completes, causing the **TweetStreamAgent** to consume more heap space in the interim. This may cause the **TweetStreamAgent** to exhaust its heap space. It is recommended that the **TweetStreamAnalyser** is stopped via the admin interface (see: "TwitterVane Administrators Guide.doc") if the batch size needs to be set to a large value (eg: if, for some reason, all Tweets need reprocessing).

Configure the Tweet Stream Options

There are two **TweetStreamAgent** options that affect analyser processing and Tweet stream errors:

Analysis Trigger Value

The **TweetStreamAgent** sends a request to the **TweetAnalyser** after receiving a specified number of Tweets. This number is specified in the following file:

$TOMCAT_HOME\webapps\TweetStreamAgent\WEB-INF\spring-servlet.xml

```
<bean id="tweetStreamAgentService"
 class="uk.bl.wap.crowdsourcing.agent.TweetStreamAgentSer
 viceImpl">
   <property name="jsonLogger" ref="jsonLogger" />
   <property name="analysisTriggerValue" value="100" />
   <property name="displayLastStreamErrors" value="5" />
…
</bean>
```

Recent Stream Errors

The number of recent Tweet stream errors that are displayed in the admin web interface (see: the “Twitter Stream” section in the “TwitterVane Administrators Guide.doc”) is also configured in the above file. Its value is set by the `displayLastStreamErrors` property.

Configure Logging

The log files for each of the **TweetView**, **TweetStreamAgent** and **TweetAnalyser** components are configured in the components \WEB-INF\classes\log4j.xml file for each component.

This file controls the location of the log file and the required logging level (INFO, DEBUG, ERROR etc).

The **TweetStreamAgent** also writes the content of a Tweet in Json format to a daily Json logfile. The name and location of the Json logfile is configured by the `JsonLogger` bean in the file:

$TOMCAT_HOME\webapps\TweetStreamAgent\WEB-INF\spring-servlet.xml

```
<!-- logger to write json tweets to a specified file -->
<bean id="jsonLogger"
class="uk.bl.wap.crowdsourcing.logger.JsonLogger" lazy-
init="false">
  <property name="fileName" value="json" />
  <property name="filePath" value="/usr/share/tomcat6/
logs/json/" />
  <property name="fileExtension" value=".log." />
</bean>
```

The configuration above will generate Json log files with the following format:

`json.log.YYYY-MM-DD`

eg:

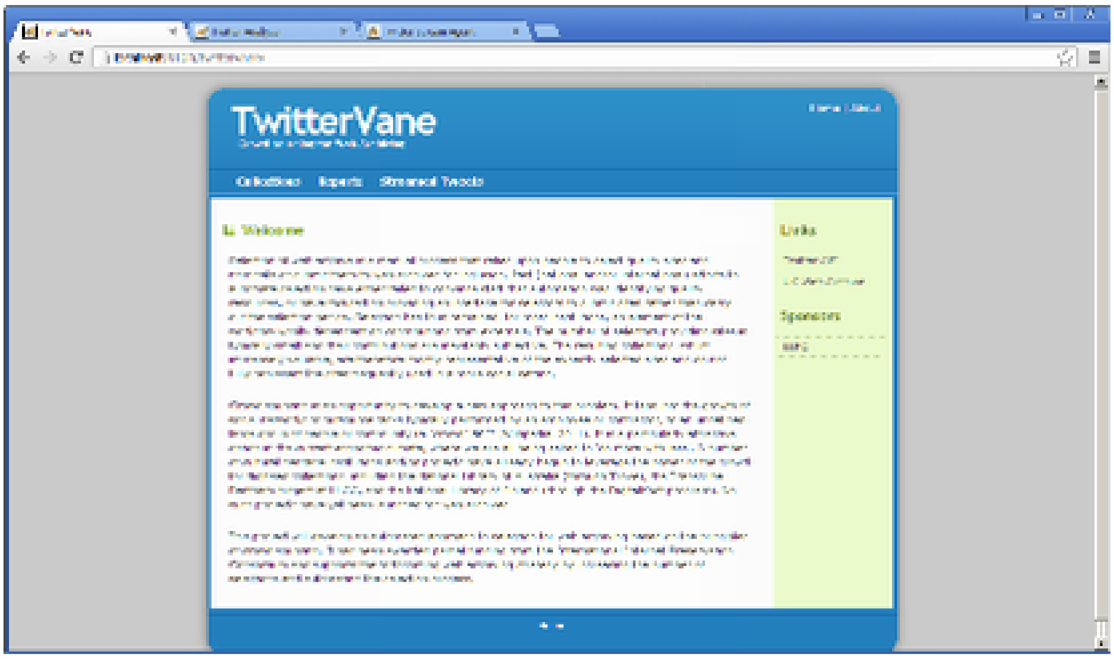
`json.log.2013-02-22`

Start the Application Servers

Once the components are configured and the application servers started, each component's UI will be available at the following URIs:

TweetView:
http://[server]:[port]/TwitterVane/

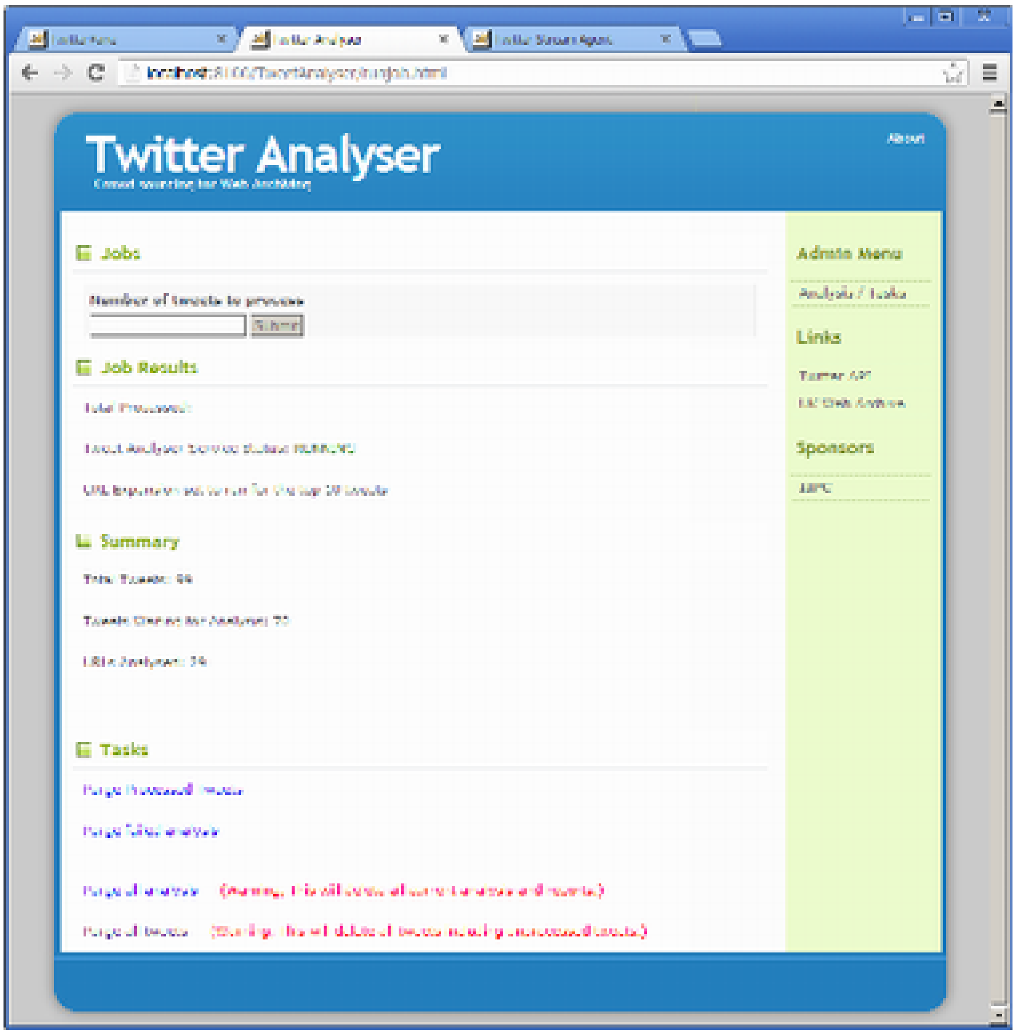
eg: server=localhost, port=8100



TweetAnalyser:

http://[server]:[port]/TweetAnalyser/

eg: server=localhost, port=8100



localhost:8100/TweetAnalyser/runjob.html

Twitter Analyser

Crowd sourcing for Web Archiving

About

Jobs

Number of tweets to process

Submit

Job Results

Total Processed:

Tweet Analyser Service status: RUNNING

URL Expansion set to run for the last 20 tweets [illegible]

Summary

Total Tweets: 96

Tweets Queued for Analysis: 72

URLs Analysed: 24

Tasks

Purge Processed Tweets

Purge Total analysis [illegible]

Purge all analysis (Warning, this will delete all current analysis and reports)

Purge all tweets (Warning: this will delete all tweets including unprocessed tweets)

Admin Menu

Analysis / Tasks

Links

Twitter API

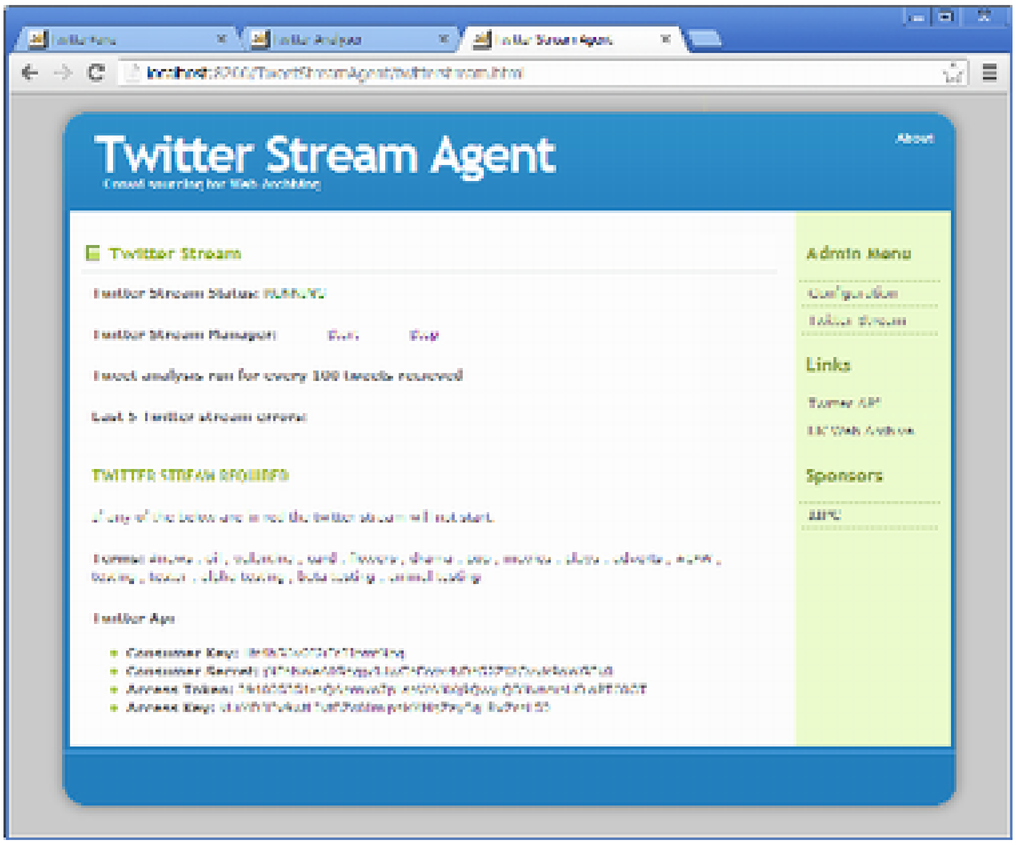
UK Web Archive

Sponsors

JISC

TweetStreamAgent:
http://[server]:[port]/TweetStreamAgent/

eg: server=localhost, port=8200



localhost:8200/TweetStreamAgent/twitterstream.html

Twitter Stream Agent

Crowd sourcing for Web Archiving

About

Twitter Stream

Twitter Stream Status: RUNNING

Twitter Stream Manager: Start Stop

Tweet analysis run for every 100 tweets received

Last 5 Twitter stream errors:

TWITTER STREAM REQUIRED

If any of the below are unset the twitter stream will not start.

Tracked terms: [illegible]

Twitter Api:

- Consumer Key: [illegible]
- Consumer Secret: [illegible]
- Access Token: [illegible]
- Access Key: [illegible]

Admin Menu

Configuration

Twitter Stream

Links

Twitter API

UK Web Archive

Sponsors

JISC

Troubleshooting setup

See the following table to troubleshoot TwitterVane setup.

Problem	Possible solution
Database connection failure	Check that the database connection is defined correctly in the WEB-INF\classes\META-INF\persistence.xml for all three components and that the server can communicate with this host on the specified port.
Connect exception from the TweetStreamAgent	The following exception will be thrown if the **TweetStreamAgent** is running and the **TweetAnalyser** component is unavailable (eg: component is stopped, not started, starting or undeployed): 2013-02-22 11:59:28 - Could not connect to remote service [rmi://localhost:1099/TweetAnalyserService]; nested exception is java.rmi.ConnectException: Connection refused to host: 194.66.231.178; nested exception is: java.net.ConnectException: Connection refused: connect Ensure that the **TweetAnalyser** component is deployed and has started successfully. If the exception persists after the **TweetAnalyser** has started, then the RMI configuration in the "Configure the RMI Connection" section of this guide should be checked for the two components.

Connect exception from the TweetStreamAgent	`The following exception is occasionally encountered during the development of TwitterVane:`
	
	`Invocation of init method failed; nested exception is org.springframework.remoting.RemoteLookupFailureException: Lookup of RMI stub failed; nested exception is java.rmi.NoSuchObjectException: no such object in table`
	
	It is caused by the removal of the RMI service class from the JVM running the **TweetAnalyser** component. This may have resulted from a redeployment of bytecode during debugging, or can be caused by the removal of the class during garbage collection.
	
	In the latter case, disabling garbage collection within the JVM running the **TweetAnalyser** will prevent the exception reoccurring. The application server running the **TweetAnalyser** component should then be restarted.

RMI Stub Error	The following exception appears in the TweetStreamAgent log when multiple versions of the Spring Framework are present on the application server classpath: 2013-02-06 13:54:43,980 ERROR [Twitter4J Async

	Dispatcher[0]] controller.TwitterStreamDaemonController$1 (TwitterStreamDaemonController.java:247) - Transaction error on tweetNo matching RMI stub method found for: public abstract void uk.bl.wap.crowdsourcing.agent.TweetAnalyserService.run(); nested exception is java.lang.NoSuchMethodException: org.springframework.remoting.rmi.RmiInvocationWrapper_Stub.run(), cause: java.lang.NoSuchMethodException: org.springframework.remoting.rmi.RmiInvocationWrapper_Stub.run() To fix the error, remove the incorrect versions of the Spring jars from the classpath. The correct version for TwitterVane is: spring-*-3.2.0.RELEASE.jar The same exception may also be thrown when the following jar is on the classpath: wayback-hadoop-1.*.jar This jar should be removed from the classpath if present.