package com.twitter.search.ingester.pipeline.twitter;

import java.util.concurrent.CompletableFuture;

import javax.naming.NamingException;

import org.apache.commons.pipeline.StageException;

import org.apache.commons.pipeline.validation.ConsumedTypes;

import org.apache.commons.pipeline.validation.ProducesConsumed;

import com.twitter.search.ingester.model.IngesterTwitterMessage;

import com.twitter.util.Function;

@ConsumedTypes(IngesterTwitterMessage.class)

@ProducesConsumed

public class RetrieveNamedEntitiesSingleTweetStage extends TwitterBaseStage

<IngesterTwitterMessage, CompletableFuture<IngesterTwitterMessage>> {

private NamedEntityHandler namedEntityHandler;

@Override

protected void doInnerPreprocess() throws StageException, NamingException {

innerSetup();

}

@Override

protected void innerSetup() {

namedEntityHandler = new NamedEntityHandler(

wireModule.getNamedEntityFetcher(), decider, getStageNamePrefix(),

"single\_tweet");

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof IngesterTwitterMessage)) {

throw new StageException(this, "Object is not a IngesterTwitterMessage object: " + obj);

}

IngesterTwitterMessage twitterMessage = (IngesterTwitterMessage) obj;

if (namedEntityHandler.shouldRetrieve(twitterMessage)) {

namedEntityHandler.retrieve(twitterMessage)

.onSuccess(Function.cons(result -> {

namedEntityHandler.addEntitiesToMessage(twitterMessage, result);

emitAndCount(twitterMessage);

}))

.onFailure(Function.cons(throwable -> {

namedEntityHandler.incrementErrorCount();

emitAndCount(twitterMessage);

}));

} else {

emitAndCount(twitterMessage);

}

}

@Override

protected CompletableFuture<IngesterTwitterMessage> innerRunStageV2(IngesterTwitterMessage

message) {

CompletableFuture<IngesterTwitterMessage> cf = new CompletableFuture<>();

if (namedEntityHandler.shouldRetrieve(message)) {

namedEntityHandler.retrieve(message)

.onSuccess(Function.cons(result -> {

namedEntityHandler.addEntitiesToMessage(message, result);

cf.complete(message);

}))

.onFailure(Function.cons(throwable -> {

namedEntityHandler.incrementErrorCount();

cf.complete(message);

}));

} else {

cf.complete(message);

}

return cf;

}

}