package com.twitter.search.feature\_update\_service;

import scala.runtime.AbstractPartialFunction;

import com.twitter.finagle.service.ReqRep;

import com.twitter.finagle.service.ResponseClass;

import com.twitter.finagle.service.ResponseClassifier;

import com.twitter.search.feature\_update\_service.thriftjava.FeatureUpdateResponse;

import com.twitter.search.feature\_update\_service.thriftjava.FeatureUpdateResponseCode;

import com.twitter.util.Try;

public class FeatureUpdateResponseClassifier

extends AbstractPartialFunction<ReqRep, ResponseClass> {

@Override

public boolean isDefinedAt(ReqRep tuple) {

return true;

}

@Override

public ResponseClass apply(ReqRep reqRep) {

Try<Object> finagleResponse = reqRep.response();

if (finagleResponse.isThrow()) {

return ResponseClassifier.Default().apply(reqRep);

}

FeatureUpdateResponse response = (FeatureUpdateResponse) finagleResponse.apply();

FeatureUpdateResponseCode responseCode = response.getResponseCode();

switch (responseCode) {

case TRANSIENT\_ERROR:

case SERVER\_TIMEOUT\_ERROR:

return ResponseClass.RetryableFailure();

case PERSISTENT\_ERROR:

return ResponseClass.NonRetryableFailure();

// Client cancellations don't necessarily mean failures on our end. The client decided to

// cancel the request (for example we timed out, so they sent a duplicate request etc.),

// so let's treat them as successes.

case CLIENT\_CANCEL\_ERROR:

default:

// The other response codes are client errors, and success, and in those cases the server

// behaved correctly, so we classify it as a success.

return ResponseClass.Success();

}

}

}