package com.twitter.search.earlybird\_root.filters;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.common.util.FinagleUtil;

import com.twitter.search.earlybird.thrift.EarlybirdRequest;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

import com.twitter.search.earlybird.thrift.EarlybirdResponseCode;

import com.twitter.search.earlybird\_root.common.ClientErrorException;

import com.twitter.search.earlybird\_root.common.EarlybirdRequestType;

import com.twitter.util.Function;

import com.twitter.util.Future;

/\*\* Converts exceptions into EarlybirdResponses with error codes. \*/

public class EarlybirdResponseExceptionHandler {

private static final Logger LOG =

LoggerFactory.getLogger(EarlybirdResponseExceptionHandler.class);

private final Map<EarlybirdRequestType, SearchCounter> requestTypeToCancelledExceptions

= new HashMap<>();

private final Map<EarlybirdRequestType, SearchCounter> requestTypeToTimeoutExceptions

= new HashMap<>();

private final Map<EarlybirdRequestType, SearchCounter> requestTypeToPersistentErrors

= new HashMap<>();

private final SearchCounter cancelledExceptions;

private final SearchCounter timeoutExceptions;

private final SearchCounter persistentErrors;

/\*\*

\* Creates a new top level filter for handling exceptions.

\*/

public EarlybirdResponseExceptionHandler(String statPrefix) {

this.cancelledExceptions = SearchCounter.export(

statPrefix + "\_exception\_handler\_cancelled\_exceptions");

this.timeoutExceptions = SearchCounter.export(

statPrefix + "\_exception\_handler\_timeout\_exceptions");

this.persistentErrors = SearchCounter.export(

statPrefix + "\_exception\_handler\_persistent\_errors");

for (EarlybirdRequestType requestType : EarlybirdRequestType.values()) {

String requestTypeNormalized = requestType.getNormalizedName();

requestTypeToCancelledExceptions.put(requestType,

SearchCounter.export(

statPrefix + "\_exception\_handler\_cancelled\_exceptions\_"

+ requestTypeNormalized));

requestTypeToTimeoutExceptions.put(requestType,

SearchCounter.export(

statPrefix + "\_exception\_handler\_timeout\_exceptions\_"

+ requestTypeNormalized));

requestTypeToPersistentErrors.put(requestType,

SearchCounter.export(

statPrefix + "\_exception\_handler\_persistent\_errors\_"

+ requestTypeNormalized));

}

}

/\*\*

\* If {@code responseFuture} is wraps an exception, converts it to an EarlybirdResponse instance

\* with an appropriate error code.

\*

\* @param request The earlybird request.

\* @param responseFuture The response future.

\*/

public Future<EarlybirdResponse> handleException(final EarlybirdRequest request,

Future<EarlybirdResponse> responseFuture) {

return responseFuture.handle(

new Function<Throwable, EarlybirdResponse>() {

@Override

public EarlybirdResponse apply(Throwable t) {

if (t instanceof ClientErrorException) {

ClientErrorException clientExc = (ClientErrorException) t;

return new EarlybirdResponse()

.setResponseCode(EarlybirdResponseCode.CLIENT\_ERROR)

.setDebugString(clientExc.getMessage());

} else if (FinagleUtil.isCancelException(t)) {

requestTypeToCancelledExceptions.get(EarlybirdRequestType.of(request))

.increment();

cancelledExceptions.increment();

return new EarlybirdResponse()

.setResponseCode(EarlybirdResponseCode.CLIENT\_CANCEL\_ERROR)

.setDebugString(t.getMessage());

} else if (FinagleUtil.isTimeoutException(t)) {

requestTypeToTimeoutExceptions.get(EarlybirdRequestType.of(request))

.increment();

timeoutExceptions.increment();

return new EarlybirdResponse()

.setResponseCode(EarlybirdResponseCode.SERVER\_TIMEOUT\_ERROR)

.setDebugString(t.getMessage());

} else {

// Unexpected exception: log it.

LOG.error("Caught unexpected exception.", t);

requestTypeToPersistentErrors.get(EarlybirdRequestType.of(request))

.increment();

persistentErrors.increment();

return new EarlybirdResponse()

.setResponseCode(EarlybirdResponseCode.PERSISTENT\_ERROR)

.setDebugString(t.getMessage());

}

}

});

}

}