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

import java.util.HashMap;

import java.util.Map;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

import com.twitter.search.common.util.earlybird.EarlybirdResponseMergeUtil;

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

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

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

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

import com.twitter.search.earlybird\_root.validators.FacetsResponseValidator;

import com.twitter.search.earlybird\_root.validators.PassThroughResponseValidator;

import com.twitter.search.earlybird\_root.validators.ServiceResponseValidator;

import com.twitter.search.earlybird\_root.validators.TermStatsResultsValidator;

import com.twitter.search.earlybird\_root.validators.TopTweetsResultsValidator;

import com.twitter.util.Function;

import com.twitter.util.Future;

/\*\*

\* Filter responsible for handling invalid response returned by downstream services, and

\* translating them into EarlybirdResponseExceptions.

\*/

public class ServiceResponseValidationFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

private final Map<EarlybirdRequestType, ServiceResponseValidator<EarlybirdResponse>>

requestTypeToResponseValidators = new HashMap<>();

private final EarlybirdCluster cluster;

/\*\*

\* Creates a new filter for handling invalid response

\*/

public ServiceResponseValidationFilter(EarlybirdCluster cluster) {

this.cluster = cluster;

ServiceResponseValidator<EarlybirdResponse> passThroughValidator =

new PassThroughResponseValidator();

requestTypeToResponseValidators

.put(EarlybirdRequestType.FACETS, new FacetsResponseValidator(cluster));

requestTypeToResponseValidators

.put(EarlybirdRequestType.RECENCY, passThroughValidator);

requestTypeToResponseValidators

.put(EarlybirdRequestType.RELEVANCE, passThroughValidator);

requestTypeToResponseValidators

.put(EarlybirdRequestType.STRICT\_RECENCY, passThroughValidator);

requestTypeToResponseValidators

.put(EarlybirdRequestType.TERM\_STATS, new TermStatsResultsValidator(cluster));

requestTypeToResponseValidators

.put(EarlybirdRequestType.TOP\_TWEETS, new TopTweetsResultsValidator(cluster));

}

@Override

public Future<EarlybirdResponse> apply(

final EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

return service.apply(requestContext).flatMap(

new Function<EarlybirdResponse, Future<EarlybirdResponse>>() {

@Override

public Future<EarlybirdResponse> apply(EarlybirdResponse response) {

if (response == null) {

return Future.exception(new IllegalStateException(

cluster + " returned null response"));

}

if (response.getResponseCode() == EarlybirdResponseCode.SUCCESS) {

return requestTypeToResponseValidators

.get(requestContext.getEarlybirdRequestType())

.validate(response);

}

return Future.value(EarlybirdResponseMergeUtil.transformInvalidResponse(

response,

String.format("Failure from %s (%s)", cluster, response.getResponseCode())));

}

});

}

}