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

import java.util.List;

import javax.inject.Inject;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.features.thrift.ThriftSearchFeatureSchemaSpecifier;

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

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

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

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

import com.twitter.util.Future;

public class EarlybirdFeatureSchemaAnnotateFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

private final EarlybirdFeatureSchemaMerger schemaMerger;

@Inject

public EarlybirdFeatureSchemaAnnotateFilter(EarlybirdFeatureSchemaMerger merger) {

this.schemaMerger = merger;

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

return service.apply(annotateRequestContext(requestContext));

}

/\*\*

\* Annotate the request to indicate the available features schemas before sending to earlybird.

\*

\* @param requestContext the earlybird request context

\*/

private EarlybirdRequestContext annotateRequestContext(EarlybirdRequestContext requestContext) {

EarlybirdRequest request = requestContext.getRequest();

if (request.isSetSearchQuery()

&& request.getSearchQuery().isSetResultMetadataOptions()

&& request.getSearchQuery().getResultMetadataOptions().isReturnSearchResultFeatures()) {

// Remember the available client side cached features schema in the context and prepare to

// reset it something new.

List<ThriftSearchFeatureSchemaSpecifier> featureSchemasAvailableInClient =

request.getSearchQuery().getResultMetadataOptions().getFeatureSchemasAvailableInClient();

return EarlybirdRequestContext.newContext(

request,

requestContext,

schemaMerger.getAvailableSchemaList(), // Set the available feature schemas based on

// what is cached in the current root.

featureSchemasAvailableInClient);

} else {

return requestContext;

}

}

}