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

import javax.inject.Inject;

import com.google.common.annotations.VisibleForTesting;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Filter;

import com.twitter.finagle.Service;

import com.twitter.search.common.decider.SearchDecider;

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

import com.twitter.search.earlybird.common.EarlybirdRequestUtil;

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

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

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

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

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

import com.twitter.search.queryparser.query.QueryParserException;

import com.twitter.util.Future;

/\*\*

\* Creates a new RequestContext from an EarlybirdRequest, and passes the RequestContext down to

\* the rest of the filter/service chain.

\*/

public class InitializeRequestContextFilter extends

Filter<EarlybirdRequest, EarlybirdResponse, EarlybirdRequestContext, EarlybirdResponse> {

@VisibleForTesting

static final SearchCounter FAILED\_QUERY\_PARSING =

SearchCounter.export("initialize\_request\_context\_filter\_query\_parsing\_failure");

private final SearchDecider decider;

private final TwitterContextProvider twitterContextProvider;

private final Clock clock;

/\*\*

\* The constructor of the filter.

\*/

@Inject

public InitializeRequestContextFilter(SearchDecider decider,

TwitterContextProvider twitterContextProvider,

Clock clock) {

this.decider = decider;

this.twitterContextProvider = twitterContextProvider;

this.clock = clock;

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequest request,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

EarlybirdRequestUtil.recordClientClockDiff(request);

EarlybirdRequestContext requestContext;

try {

requestContext = EarlybirdRequestContext.newContext(

request, decider, twitterContextProvider.get(), clock);

} catch (QueryParserException e) {

FAILED\_QUERY\_PARSING.increment();

return QueryParsingUtils.newClientErrorResponse(request, e);

}

return service.apply(requestContext);

}

}