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

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

import javax.annotation.Nullable;

import scala.Option;

import com.google.common.base.Preconditions;

import com.google.common.collect.ImmutableSet;

import com.google.common.collect.Sets;

import com.twitter.common.util.Clock;

import com.twitter.context.thriftscala.Viewer;

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

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

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

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

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

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

/\*\*

\* A class that wraps a request and additional per-request data that should be passed to services.

\*

\* This class should be immutable. At the very least, it must be thread-safe. In practice, since

\* EarlybirdRequest is a mutable Thrift structure, the users of this class need to make sure that

\* once a request is used to create a RequestContext instance, it is not modified.

\*

\* If the request needs to be modified, a new RequestContext with the modified EarlybirdRequest

\* should be created.

\*/

public final class EarlybirdRequestContext {

private static final String OVERRIDE\_TIER\_CONFIGS\_DECIDER\_KEY = "use\_override\_tier\_configs";

/\*\*

\* Creates a new context with the provided earlybird request, and using the given decider.

\*/

public static EarlybirdRequestContext newContext(

EarlybirdRequest request,

SearchDecider decider,

Option<Viewer> twitterContextViewer,

Clock clock) throws QueryParserException {

// Try to capture created time as early as possible. For example, we want to account for query

// parsing time.

long createdTimeMillis = clock.nowMillis();

boolean useOverrideTierConfig = decider.isAvailable(OVERRIDE\_TIER\_CONFIGS\_DECIDER\_KEY);

Query parsedQuery = QueryParsingUtils.getParsedQuery(request);

return new EarlybirdRequestContext(

request,

parsedQuery,

useOverrideTierConfig,

createdTimeMillis,

twitterContextViewer);

}

/\*\*

\* Intersection of the userID and the flock response, which is set in the followedUserIds field.

\* This is used for protected cluster.

\*/

public static EarlybirdRequestContext newContextWithRestrictFromUserIdFilter64(

EarlybirdRequestContext requestContext) {

Preconditions.checkArgument(requestContext.getRequest().isSetFollowedUserIds());

EarlybirdRequest request = requestContext.getRequest().deepCopy();

List<Long> toIntersect = request.getFollowedUserIds();

ThriftSearchQuery searchQuery = request.getSearchQuery();

if (!searchQuery.isSetFromUserIDFilter64()) {

searchQuery.setFromUserIDFilter64(new ArrayList<>(toIntersect));

} else {

Set<Long> intersection = Sets.intersection(

Sets.newHashSet(searchQuery.getFromUserIDFilter64()),

Sets.newHashSet(toIntersect));

searchQuery.setFromUserIDFilter64(new ArrayList<>(intersection));

}

return new EarlybirdRequestContext(requestContext, request, requestContext.getParsedQuery());

}

/\*\*

\* Makes an exact copy of the provided request context, by cloning the underlying earlybird

\* request.

\*/

public static EarlybirdRequestContext copyRequestContext(

EarlybirdRequestContext requestContext,

Query parsedQuery) {

return new EarlybirdRequestContext(requestContext, parsedQuery);

}

/\*\*

\* Creates a new context with the provided request, context and reset both the feature schemas

\* cached in client and the feature schemas cached in the local cache.

\*/

public static EarlybirdRequestContext newContext(

EarlybirdRequest oldRequest,

EarlybirdRequestContext oldRequestContext,

List<ThriftSearchFeatureSchemaSpecifier> featureSchemasAvailableInCache,

List<ThriftSearchFeatureSchemaSpecifier> featureSchemasAvailableInClient) {

EarlybirdRequest request = oldRequest.deepCopy();

request.getSearchQuery().getResultMetadataOptions()

.setFeatureSchemasAvailableInClient(featureSchemasAvailableInCache);

ImmutableSet<ThriftSearchFeatureSchemaSpecifier> featureSchemaSetAvailableInClient = null;

if (featureSchemasAvailableInClient != null) {

featureSchemaSetAvailableInClient = ImmutableSet.copyOf(featureSchemasAvailableInClient);

}

return new EarlybirdRequestContext(

request,

EarlybirdRequestType.of(request),

oldRequestContext.getParsedQuery(),

oldRequestContext.useOverrideTierConfig(),

oldRequestContext.getCreatedTimeMillis(),

oldRequestContext.getTwitterContextViewer(),

featureSchemaSetAvailableInClient);

}

public EarlybirdRequestContext deepCopy() {

return new EarlybirdRequestContext(request.deepCopy(), parsedQuery, useOverrideTierConfig,

createdTimeMillis, twitterContextViewer);

}

private final EarlybirdRequest request;

// EarlybirdRequestType should not change for a given request. Computing it once here so that we

// don't need to compute it from the request every time we want to use it.

private final EarlybirdRequestType earlybirdRequestType;

// The parsed query matching the serialized query in the request. May be null if the request does

// not contain a serialized query.

// If a request's serialized query needs to be rewritten for any reason, a new

// EarlybirdRequestContext should be created, with a new EarlybirdRequest (with a new serialized

// query), and a new parsed query (matching the new serialized query).

@Nullable

private final Query parsedQuery;

private final boolean useOverrideTierConfig;

private final long createdTimeMillis;

private final Option<Viewer> twitterContextViewer;

@Nullable

private final ImmutableSet<ThriftSearchFeatureSchemaSpecifier> featureSchemasAvailableInClient;

private EarlybirdRequestContext(

EarlybirdRequest request,

Query parsedQuery,

boolean useOverrideTierConfig,

long createdTimeMillis,

Option<Viewer> twitterContextViewer) {

this(request,

EarlybirdRequestType.of(request),

parsedQuery,

useOverrideTierConfig,

createdTimeMillis,

twitterContextViewer,

null);

}

private EarlybirdRequestContext(

EarlybirdRequest request,

EarlybirdRequestType earlybirdRequestType,

Query parsedQuery,

boolean useOverrideTierConfig,

long createdTimeMillis,

Option<Viewer> twitterContextViewer,

@Nullable ImmutableSet<ThriftSearchFeatureSchemaSpecifier> featureSchemasAvailableInClient) {

this.request = Preconditions.checkNotNull(request);

this.earlybirdRequestType = earlybirdRequestType;

this.parsedQuery = parsedQuery;

this.useOverrideTierConfig = useOverrideTierConfig;

this.createdTimeMillis = createdTimeMillis;

this.twitterContextViewer = twitterContextViewer;

this.featureSchemasAvailableInClient = featureSchemasAvailableInClient;

}

private EarlybirdRequestContext(EarlybirdRequestContext otherContext, Query otherParsedQuery) {

this(otherContext, otherContext.getRequest().deepCopy(), otherParsedQuery);

}

private EarlybirdRequestContext(EarlybirdRequestContext otherContext,

EarlybirdRequest otherRequest,

Query otherParsedQuery) {

this(otherRequest,

otherContext.earlybirdRequestType,

otherParsedQuery,

otherContext.useOverrideTierConfig,

otherContext.createdTimeMillis,

otherContext.twitterContextViewer,

null);

Preconditions.checkState(request.isSetSearchQuery());

this.request.getSearchQuery().setSerializedQuery(otherParsedQuery.serialize());

}

public EarlybirdRequest getRequest() {

return request;

}

public boolean useOverrideTierConfig() {

return useOverrideTierConfig;

}

public EarlybirdRequestType getEarlybirdRequestType() {

return earlybirdRequestType;

}

@Nullable

public Query getParsedQuery() {

return parsedQuery;

}

public long getCreatedTimeMillis() {

return createdTimeMillis;

}

public Option<Viewer> getTwitterContextViewer() {

return twitterContextViewer;

}

@Nullable

public Set<ThriftSearchFeatureSchemaSpecifier> getFeatureSchemasAvailableInClient() {

return featureSchemasAvailableInClient;

}

}