package com.twitter.timelineranker.common

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.FutureArrow

import com.twitter.servo.util.Gate

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.timelineranker.model.RecapQuery

import com.twitter.timelineranker.parameters.recap.RecapParams

import com.twitter.timelineranker.parameters.uteg\_liked\_by\_tweets.UtegLikedByTweetsParams

import com.twitter.timelineranker.util.TweetFilters

import com.twitter.timelines.common.model.TweetKindOption

import com.twitter.util.Future

import scala.collection.mutable

object TweetKindOptionHydratedTweetsFilterTransform {

private[common] val enableExpandedExtendedRepliesGate: Gate[RecapQuery] =

RecapQuery.paramGate(RecapParams.EnableExpandedExtendedRepliesFilterParam)

private[common] val excludeRecommendedRepliesToNonFollowedUsersGate: Gate[RecapQuery] =

RecapQuery.paramGate(

UtegLikedByTweetsParams.UTEGRecommendationsFilter.ExcludeRecommendedRepliesToNonFollowedUsersParam)

}

/\*\*

\* Filter hydrated tweets dynamically based on TweetKindOptions in the query.

\*/

class TweetKindOptionHydratedTweetsFilterTransform(

useFollowGraphData: Boolean,

useSourceTweets: Boolean,

statsReceiver: StatsReceiver)

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

import TweetKindOptionHydratedTweetsFilterTransform.\_

override def apply(envelope: CandidateEnvelope): Future[CandidateEnvelope] = {

val filters = convertToFilters(envelope)

val filterTransform = if (filters == TweetFilters.ValueSet.empty) {

FutureArrow.identity[CandidateEnvelope]

} else {

new HydratedTweetsFilterTransform(

outerFilters = filters,

innerFilters = TweetFilters.None,

useFollowGraphData = useFollowGraphData,

useSourceTweets = useSourceTweets,

statsReceiver = statsReceiver,

numRetweetsAllowed = HydratedTweetsFilterTransform.NumDuplicateRetweetsAllowed

)

}

filterTransform.apply(envelope)

}

/\*\*

\* Converts the given query options to equivalent TweetFilter values.

\*

\* Note:

\* -- The semantic of TweetKindOption is opposite of that of TweetFilters.

\* TweetKindOption values are of the form IncludeX. That is, they result in X being added.

\* TweetFilters values specify what to exclude.

\* -- IncludeExtendedReplies requires IncludeReplies to be also specified to be effective.

\*/

private[common] def convertToFilters(envelope: CandidateEnvelope): TweetFilters.ValueSet = {

val queryOptions = envelope.query.options

val filters = mutable.Set.empty[TweetFilters.Value]

if (queryOptions.contains(TweetKindOption.IncludeReplies)) {

if (excludeRecommendedRepliesToNonFollowedUsersGate(

envelope.query) && envelope.query.utegLikedByTweetsOptions.isDefined) {

filters += TweetFilters.RecommendedRepliesToNotFollowedUsers

} else if (queryOptions.contains(TweetKindOption.IncludeExtendedReplies)) {

if (enableExpandedExtendedRepliesGate(envelope.query)) {

filters += TweetFilters.NotValidExpandedExtendedReplies

} else {

filters += TweetFilters.NotQualifiedExtendedReplies

}

} else {

filters += TweetFilters.ExtendedReplies

}

} else {

filters += TweetFilters.Replies

}

if (!queryOptions.contains(TweetKindOption.IncludeRetweets)) {

filters += TweetFilters.Retweets

}

TweetFilters.ValueSet.empty ++ filters

}

}