package com.twitter.home\_mixer.product.scored\_tweets.candidate\_pipeline

import com.twitter.tweet\_mixer.{thriftscala => t}

import com.twitter.home\_mixer.model.HomeFeatures.AuthorIdFeature

import com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.TweetypieStaticEntitiesFeatureHydrator

import com.twitter.home\_mixer.product.scored\_tweets.gate.MinCachedTweetsGate

import com.twitter.home\_mixer.product.scored\_tweets.model.ScoredTweetsQuery

import com.twitter.home\_mixer.product.scored\_tweets.param.ScoredTweetsParam.CachedScoredTweets

import com.twitter.home\_mixer.product.scored\_tweets.param.ScoredTweetsParam.CandidatePipeline

import com.twitter.home\_mixer.product.scored\_tweets.response\_transformer.ScoredTweetsTweetMixerResponseFeatureTransformer

import com.twitter.home\_mixer.util.CachedScoredTweetsHelper

import com.twitter.product\_mixer.component\_library.candidate\_source.tweet\_mixer.TweetMixerCandidateSource

import com.twitter.product\_mixer.component\_library.filter.PredicateFeatureFilter

import com.twitter.product\_mixer.component\_library.model.candidate.TweetCandidate

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.BaseCandidateSource

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.BaseCandidateFeatureHydrator

import com.twitter.product\_mixer.core.functional\_component.filter.Filter

import com.twitter.product\_mixer.core.functional\_component.gate.Gate

import com.twitter.product\_mixer.core.functional\_component.marshaller.request.ClientContextMarshaller

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidateFeatureTransformer

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineQueryTransformer

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineResultsTransformer

import com.twitter.product\_mixer.core.model.common.identifier.CandidatePipelineIdentifier

import com.twitter.product\_mixer.core.model.common.identifier.FilterIdentifier

import com.twitter.product\_mixer.core.pipeline.candidate.CandidatePipelineConfig

import com.twitter.timelines.configapi.decider.DeciderParam

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Candidate Pipeline Config that fetches tweets from TweetMixer.

\*/

@Singleton

class ScoredTweetsTweetMixerCandidatePipelineConfig @Inject() (

tweetMixerCandidateSource: TweetMixerCandidateSource,

tweetypieStaticEntitiesFeatureHydrator: TweetypieStaticEntitiesFeatureHydrator)

extends CandidatePipelineConfig[

ScoredTweetsQuery,

t.TweetMixerRequest,

t.TweetResult,

TweetCandidate

] {

override val identifier: CandidatePipelineIdentifier =

CandidatePipelineIdentifier("ScoredTweetsTweetMixer")

val HasAuthorFilterId = "HasAuthor"

override val enabledDeciderParam: Option[DeciderParam[Boolean]] =

Some(CandidatePipeline.EnableTweetMixerParam)

override val gates: Seq[Gate[ScoredTweetsQuery]] = Seq(

MinCachedTweetsGate(identifier, CachedScoredTweets.MinCachedTweetsParam),

)

override val candidateSource: BaseCandidateSource[t.TweetMixerRequest, t.TweetResult] =

tweetMixerCandidateSource

private val MaxTweetsToFetch = 400

override val queryTransformer: CandidatePipelineQueryTransformer[

ScoredTweetsQuery,

t.TweetMixerRequest

] = { query =>

val maxCount = (query.getQualityFactorCurrentValue(identifier) \* MaxTweetsToFetch).toInt

val excludedTweetIds = query.features.map(

CachedScoredTweetsHelper.tweetImpressionsAndCachedScoredTweets(\_, identifier))

t.TweetMixerRequest(

clientContext = ClientContextMarshaller(query.clientContext),

product = t.Product.HomeRecommendedTweets,

productContext = Some(

t.ProductContext.HomeRecommendedTweetsProductContext(

t.HomeRecommendedTweetsProductContext(excludedTweetIds = excludedTweetIds.map(\_.toSet)))),

maxResults = Some(maxCount)

)

}

override val preFilterFeatureHydrationPhase1: Seq[

BaseCandidateFeatureHydrator[ScoredTweetsQuery, TweetCandidate, \_]

] = Seq(tweetypieStaticEntitiesFeatureHydrator)

override val featuresFromCandidateSourceTransformers: Seq[

CandidateFeatureTransformer[t.TweetResult]

] = Seq(ScoredTweetsTweetMixerResponseFeatureTransformer)

override val filters: Seq[Filter[ScoredTweetsQuery, TweetCandidate]] = Seq(

PredicateFeatureFilter.fromPredicate(

FilterIdentifier(HasAuthorFilterId),

shouldKeepCandidate = \_.getOrElse(AuthorIdFeature, None).isDefined

)

)

override val resultTransformer: CandidatePipelineResultsTransformer[

t.TweetResult,

TweetCandidate

] = { sourceResult => TweetCandidate(id = sourceResult.tweetId) }

}