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

import com.twitter.home\_mixer.functional\_component.filter.ReplyFilter

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

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

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.gate.MinTimeSinceLastRequestGate

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.param.ScoredTweetsParam.EnableBackfillCandidatePipelineParam

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

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

import com.twitter.product\_mixer.component\_library.gate.NonEmptySeqFeatureGate

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

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

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

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.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.CandidateSourceIdentifier

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

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

import com.twitter.timelines.configapi.FSParam

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

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class ScoredTweetsBackfillCandidatePipelineConfig @Inject() (

tweetypieStaticEntitiesHydrator: TweetypieStaticEntitiesFeatureHydrator)

extends CandidatePipelineConfig[

ScoredTweetsQuery,

ScoredTweetsQuery,

Long,

TweetCandidate

] {

override val identifier: CandidatePipelineIdentifier =

CandidatePipelineIdentifier("ScoredTweetsBackfill")

private val HasAuthorFilterId = "HasAuthor"

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

Some(CandidatePipeline.EnableBackfillParam)

override val supportedClientParam: Option[FSParam[Boolean]] =

Some(EnableBackfillCandidatePipelineParam)

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

Seq(

MinTimeSinceLastRequestGate,

NonEmptySeqFeatureGate(TimelineServiceTweetsFeature),

MinCachedTweetsGate(identifier, CachedScoredTweets.MinCachedTweetsParam)

)

override val queryTransformer: CandidatePipelineQueryTransformer[

ScoredTweetsQuery,

ScoredTweetsQuery

] = identity

override def candidateSource: CandidateSource[ScoredTweetsQuery, Long] =

PassthroughCandidateSource(

identifier = CandidateSourceIdentifier("ScoredTweetsBackfill"),

candidateExtractor = { query =>

query.features.map(\_.getOrElse(TimelineServiceTweetsFeature, Seq.empty)).toSeq.flatten

}

)

override val featuresFromCandidateSourceTransformers: Seq[

CandidateFeatureTransformer[Long]

] = Seq(ScoredTweetsBackfillResponseFeatureTransformer)

override val resultTransformer: CandidatePipelineResultsTransformer[Long, TweetCandidate] = {

sourceResult => TweetCandidate(id = sourceResult)

}

override val preFilterFeatureHydrationPhase1: Seq[

BaseCandidateFeatureHydrator[ScoredTweetsQuery, TweetCandidate, \_]

] = Seq(tweetypieStaticEntitiesHydrator)

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

ReplyFilter,

PredicateFeatureFilter.fromPredicate(

FilterIdentifier(HasAuthorFilterId),

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

)

)

}