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

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.query\_transformer.TimelineRankerUtegQueryTransformer

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

import com.twitter.product\_mixer.component\_library.candidate\_source.timeline\_ranker.TimelineRankerUtegCandidateSource

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.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.pipeline.candidate.CandidatePipelineConfig

import com.twitter.timelineranker.{thriftscala => t}

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

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Candidate Pipeline Config that fetches tweets from the Timeline Ranker UTEG Candidate Source

\*/

@Singleton

class ScoredTweetsUtegCandidatePipelineConfig @Inject() (

timelineRankerUtegCandidateSource: TimelineRankerUtegCandidateSource)

extends CandidatePipelineConfig[

ScoredTweetsQuery,

t.UtegLikedByTweetsQuery,

t.CandidateTweet,

TweetCandidate

] {

override val identifier: CandidatePipelineIdentifier =

CandidatePipelineIdentifier("ScoredTweetsUteg")

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

Some(CandidatePipeline.EnableUtegParam)

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

MinCachedTweetsGate(identifier, CachedScoredTweets.MinCachedTweetsParam)

)

override val candidateSource: BaseCandidateSource[t.UtegLikedByTweetsQuery, t.CandidateTweet] =

timelineRankerUtegCandidateSource

override val queryTransformer: CandidatePipelineQueryTransformer[

ScoredTweetsQuery,

t.UtegLikedByTweetsQuery

] = TimelineRankerUtegQueryTransformer(identifier)

override val featuresFromCandidateSourceTransformers: Seq[

CandidateFeatureTransformer[t.CandidateTweet]

] = Seq(ScoredTweetsUtegResponseFeatureTransformer)

override val resultTransformer: CandidatePipelineResultsTransformer[

t.CandidateTweet,

TweetCandidate

] = { sourceResult => TweetCandidate(id = sourceResult.tweet.get.id) }

}