package com.twitter.home\_mixer.candidate\_pipeline

import com.twitter.home\_mixer.functional\_component.feature\_hydrator.InNetworkFeatureHydrator

import com.twitter.home\_mixer.functional\_component.feature\_hydrator.NamesFeatureHydrator

import com.twitter.home\_mixer.functional\_component.feature\_hydrator.TweetypieFeatureHydrator

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

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

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

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

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

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

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

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

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

import com.twitter.home\_mixer.service.HomeMixerAlertConfig

import com.twitter.product\_mixer.component\_library.candidate\_source.tweetconvosvc.ConversationServiceCandidateSource

import com.twitter.product\_mixer.component\_library.candidate\_source.tweetconvosvc.ConversationServiceCandidateSourceRequest

import com.twitter.product\_mixer.component\_library.candidate\_source.tweetconvosvc.TweetWithConversationMetadata

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

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.decorator.CandidateDecorator

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.BaseGate

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

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

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

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.PipelineQuery

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

/\*\*

\* Candidate Pipeline Config that fetches tweets from the Conversation Service Candidate Source

\*/

class ConversationServiceCandidatePipelineConfig[Query <: PipelineQuery](

conversationServiceCandidateSource: ConversationServiceCandidateSource,

tweetypieFeatureHydrator: TweetypieFeatureHydrator,

namesFeatureHydrator: NamesFeatureHydrator,

invalidSubscriptionTweetFilter: InvalidSubscriptionTweetFilter,

override val gates: Seq[BaseGate[Query]],

override val decorator: Option[CandidateDecorator[Query, TweetCandidate]])

extends DependentCandidatePipelineConfig[

Query,

ConversationServiceCandidateSourceRequest,

TweetWithConversationMetadata,

TweetCandidate

] {

override val identifier: CandidatePipelineIdentifier =

CandidatePipelineIdentifier("ConversationService")

private val TweetypieHydratedFilterId = "TweetypieHydrated"

private val QuotedTweetDroppedFilterId = "QuotedTweetDropped"

override val candidateSource: BaseCandidateSource[

ConversationServiceCandidateSourceRequest,

TweetWithConversationMetadata

] = conversationServiceCandidateSource

override val queryTransformer: DependentCandidatePipelineQueryTransformer[

Query,

ConversationServiceCandidateSourceRequest

] = { (\_, candidates) =>

val tweetsWithConversationMetadata = candidates.map { candidate =>

TweetWithConversationMetadata(

tweetId = candidate.candidateIdLong,

userId = candidate.features.getOrElse(AuthorIdFeature, None),

sourceTweetId = candidate.features.getOrElse(SourceTweetIdFeature, None),

sourceUserId = candidate.features.getOrElse(SourceUserIdFeature, None),

inReplyToTweetId = candidate.features.getOrElse(InReplyToTweetIdFeature, None),

conversationId = None,

ancestors = Seq.empty

)

}

ConversationServiceCandidateSourceRequest(tweetsWithConversationMetadata)

}

override val featuresFromCandidateSourceTransformers: Seq[

CandidateFeatureTransformer[TweetWithConversationMetadata]

] = Seq(ConversationServiceResponseFeatureTransformer)

override val resultTransformer: CandidatePipelineResultsTransformer[

TweetWithConversationMetadata,

TweetCandidate

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

override val preFilterFeatureHydrationPhase1: Seq[

BaseCandidateFeatureHydrator[Query, TweetCandidate, \_]

] = Seq(

tweetypieFeatureHydrator,

InNetworkFeatureHydrator,

)

override def filters: Seq[Filter[Query, TweetCandidate]] = Seq(

RetweetDeduplicationFilter,

FeatureFilter.fromFeature(FilterIdentifier(TweetypieHydratedFilterId), IsHydratedFeature),

PredicateFeatureFilter.fromPredicate(

FilterIdentifier(QuotedTweetDroppedFilterId),

shouldKeepCandidate = { features => !features.getOrElse(QuotedTweetDroppedFeature, false) }

),

invalidSubscriptionTweetFilter,

InvalidConversationModuleFilter

)

override val postFilterFeatureHydration: Seq[

BaseCandidateFeatureHydrator[Query, TweetCandidate, \_]

] = Seq(namesFeatureHydrator)

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(),

HomeMixerAlertConfig.BusinessHours.defaultEmptyResponseRateAlert()

)

}