package com.twitter.home\_mixer.util

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

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

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

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

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

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

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

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

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

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

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

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

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

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.model.common.presentation.ItemCandidateWithDetails

import com.twitter.product\_mixer.core.model.common.presentation.ModuleCandidateWithDetails

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.UnexpectedCandidateResult

import scala.reflect.ClassTag

object CandidatesUtil {

def getItemCandidates(candidates: Seq[CandidateWithDetails]): Seq[ItemCandidateWithDetails] = {

candidates.collect {

case item: ItemCandidateWithDetails if !item.isCandidateType[CursorCandidate] => Seq(item)

case module: ModuleCandidateWithDetails => module.candidates

}.flatten

}

def getItemCandidatesWithOnlyModuleLast(

candidates: Seq[CandidateWithDetails]

): Seq[ItemCandidateWithDetails] = {

candidates.collect {

case item: ItemCandidateWithDetails if !item.isCandidateType[CursorCandidate] => item

case module: ModuleCandidateWithDetails => module.candidates.last

}

}

def containsType[CandidateType <: UniversalNoun[\_]](

candidates: Seq[CandidateWithDetails]

)(

implicit tag: ClassTag[CandidateType]

): Boolean = candidates.exists {

case ItemCandidateWithDetails(\_: CandidateType, \_, \_) => true

case module: ModuleCandidateWithDetails =>

module.candidates.head.isCandidateType[CandidateType]()

case \_ => false

}

def getOriginalTweetId(candidate: CandidateWithFeatures[TweetCandidate]): Long = {

if (candidate.features.getOrElse(IsRetweetFeature, false))

candidate.features.getOrElse(SourceTweetIdFeature, None).getOrElse(candidate.candidate.id)

else

candidate.candidate.id

}

def getOriginalAuthorId(candidateFeatures: FeatureMap): Option[Long] =

if (candidateFeatures.getOrElse(IsRetweetFeature, false))

candidateFeatures.getOrElse(SourceUserIdFeature, None)

else candidateFeatures.getOrElse(AuthorIdFeature, None)

def isOriginalTweet(candidate: CandidateWithFeatures[TweetCandidate]): Boolean =

!candidate.features.getOrElse(IsRetweetFeature, false) &&

candidate.features.getOrElse(InReplyToTweetIdFeature, None).isEmpty

def getEngagerUserIds(

candidateFeatures: FeatureMap

): Seq[Long] = {

candidateFeatures.getOrElse(FavoritedByUserIdsFeature, Seq.empty) ++

candidateFeatures.getOrElse(RetweetedByEngagerIdsFeature, Seq.empty) ++

candidateFeatures.getOrElse(RepliedByEngagerIdsFeature, Seq.empty)

}

def getMediaUnderstandingAnnotationIds(

candidateFeatures: FeatureMap

): Seq[Long] = {

if (candidateFeatures.get(HasImageFeature))

candidateFeatures.getOrElse(MediaUnderstandingAnnotationIdsFeature, Seq.empty)

else Seq.empty

}

def getTweetIdAndSourceId(candidate: CandidateWithFeatures[TweetCandidate]): Seq[Long] =

Seq(candidate.candidate.id) ++ candidate.features.getOrElse(SourceTweetIdFeature, None)

def isAuthoredByViewer(query: PipelineQuery, candidateFeatures: FeatureMap): Boolean =

candidateFeatures.getOrElse(AuthorIdFeature, None).contains(query.getRequiredUserId) ||

(candidateFeatures.getOrElse(IsRetweetFeature, false) &&

candidateFeatures.getOrElse(SourceUserIdFeature, None).contains(query.getRequiredUserId))

val reverseChronTweetsOrdering: Ordering[CandidateWithDetails] =

Ordering.by[CandidateWithDetails, Long] {

case ItemCandidateWithDetails(candidate: TweetCandidate, \_, \_) => -candidate.id

case ModuleCandidateWithDetails(candidates, \_, \_) if candidates.nonEmpty =>

-candidates.last.candidateIdLong

case \_ => throw PipelineFailure(UnexpectedCandidateResult, "Invalid candidate type")

}

val scoreOrdering: Ordering[CandidateWithDetails] = Ordering.by[CandidateWithDetails, Double] {

case ItemCandidateWithDetails(\_, \_, features) =>

-features.getOrElse(ScoreFeature, None).getOrElse(0.0)

case ModuleCandidateWithDetails(candidates, \_, \_) =>

-candidates.last.features.getOrElse(ScoreFeature, None).getOrElse(0.0)

case \_ => throw PipelineFailure(UnexpectedCandidateResult, "Invalid candidate type")

}

val conversationModuleTweetsOrdering: Ordering[CandidateWithDetails] =

Ordering.by[CandidateWithDetails, Long] {

case ItemCandidateWithDetails(candidate: TweetCandidate, \_, \_) => candidate.id

case \_ => throw PipelineFailure(UnexpectedCandidateResult, "Only Item candidate expected")

}

}