package com.twitter.home\_mixer.product.for\_you.feature\_hydrator

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

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

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

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

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

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

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

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

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

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

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

import com.twitter.product\_mixer.core.feature.Feature

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

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

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

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

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

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

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Social context for convo modules is hydrated on the root Tweet but needs info about the focal

\* Tweet (e.g. author) to render the banner. This hydrator copies focal Tweet data into the root.

\*/

@Singleton

class FocalTweetFeatureHydrator @Inject() ()

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate] {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier("FocalTweet")

override val features: Set[Feature[\_, \_]] = Set(

FocalTweetAuthorIdFeature,

FocalTweetInNetworkFeature,

FocalTweetRealNamesFeature,

FocalTweetScreenNamesFeature

)

private val DefaultFeatureMap = FeatureMapBuilder()

.add(FocalTweetAuthorIdFeature, None)

.add(FocalTweetInNetworkFeature, None)

.add(FocalTweetRealNamesFeature, None)

.add(FocalTweetScreenNamesFeature, None)

.build()

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = {

// Build a map of all the focal tweets to their corresponding features

val focalTweetIdToFeatureMap = candidates.flatMap { candidate =>

val focalTweetId = candidate.features.getOrElse(ConversationModuleFocalTweetIdFeature, None)

if (focalTweetId.contains(candidate.candidate.id)) {

Some(candidate.candidate.id -> candidate.features)

} else None

}.toMap

val updatedFeatureMap = candidates.map { candidate =>

val focalTweetId = candidate.features.getOrElse(ConversationModuleFocalTweetIdFeature, None)

val conversationId = candidate.features.getOrElse(ConversationModuleIdFeature, None)

// Check if the candidate is a root tweet and ensure its focal tweet's features are available

if (conversationId.contains(candidate.candidate.id)

&& focalTweetId.exists(focalTweetIdToFeatureMap.contains)) {

val featureMap = focalTweetIdToFeatureMap.get(focalTweetId.get).get

FeatureMapBuilder()

.add(FocalTweetAuthorIdFeature, featureMap.getOrElse(AuthorIdFeature, None))

.add(FocalTweetInNetworkFeature, Some(featureMap.getOrElse(InNetworkFeature, true)))

.add(

FocalTweetRealNamesFeature,

Some(featureMap.getOrElse(RealNamesFeature, Map.empty[Long, String])))

.add(

FocalTweetScreenNamesFeature,

Some(featureMap.getOrElse(ScreenNamesFeature, Map.empty[Long, String])))

.build()

} else DefaultFeatureMap

}

Stitch.value(updatedFeatureMap)

}

}