package com.twitter.home\_mixer.functional\_component.decorator.urt.builder

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

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

import com.twitter.home\_mixer.param.HomeGlobalParams.EnableSocialContextParam

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.functional\_component.decorator.urt.builder.social\_context.BaseSocialContextBuilder

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.SocialContext

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

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class HomeTweetSocialContextBuilder @Inject() (

likedBySocialContextBuilder: LikedBySocialContextBuilder,

listsSocialContextBuilder: ListsSocialContextBuilder,

followedBySocialContextBuilder: FollowedBySocialContextBuilder,

topicSocialContextBuilder: TopicSocialContextBuilder,

extendedReplySocialContextBuilder: ExtendedReplySocialContextBuilder,

receivedReplySocialContextBuilder: ReceivedReplySocialContextBuilder,

popularVideoSocialContextBuilder: PopularVideoSocialContextBuilder,

popularInYourAreaSocialContextBuilder: PopularInYourAreaSocialContextBuilder)

extends BaseSocialContextBuilder[PipelineQuery, TweetCandidate] {

def apply(

query: PipelineQuery,

candidate: TweetCandidate,

features: FeatureMap

): Option[SocialContext] = {

if (query.params(EnableSocialContextParam)) {

features.getOrElse(ConversationModuleFocalTweetIdFeature, None) match {

case None =>

likedBySocialContextBuilder(query, candidate, features)

.orElse(followedBySocialContextBuilder(query, candidate, features))

.orElse(topicSocialContextBuilder(query, candidate, features))

.orElse(popularVideoSocialContextBuilder(query, candidate, features))

.orElse(listsSocialContextBuilder(query, candidate, features))

.orElse(popularInYourAreaSocialContextBuilder(query, candidate, features))

case Some(\_) =>

val conversationId = features.getOrElse(ConversationModuleIdFeature, None)

// Only hydrate the social context into the root tweet in a conversation module

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

extendedReplySocialContextBuilder(query, candidate, features)

.orElse(receivedReplySocialContextBuilder(query, candidate, features))

} else None

}

} else None

}

}