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

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

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

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

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.model.marshalling.response.urt.metadata.TopicContext

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

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class TopicSocialContextBuilder @Inject() ()

extends BaseSocialContextBuilder[PipelineQuery, TweetCandidate] {

def apply(

query: PipelineQuery,

candidate: TweetCandidate,

candidateFeatures: FeatureMap

): Option[SocialContext] = {

val inNetwork = candidateFeatures.getOrElse(InNetworkFeature, true)

if (!inNetwork) {

val topicIdSocialContextOpt = candidateFeatures.getOrElse(TopicIdSocialContextFeature, None)

val topicContextFunctionalityTypeOpt =

candidateFeatures.getOrElse(TopicContextFunctionalityTypeFeature, None)

(topicIdSocialContextOpt, topicContextFunctionalityTypeOpt) match {

case (Some(topicId), Some(topicContextFunctionalityType)) =>

Some(

TopicContext(

topicId = topicId.toString,

functionalityType = Some(topicContextFunctionalityType)

))

case \_ => None

}

} else {

None

}

}

}