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.PerspectiveFilteredLikedByUserIdsFeature

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

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

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

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

import com.twitter.home\_mixer.product.following.model.HomeMixerExternalStrings

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

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

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

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

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

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

import com.twitter.product\_mixer.core.product.guice.scope.ProductScoped

import com.twitter.stringcenter.client.StringCenter

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class NotInterestedTopicFeedbackActionBuilder @Inject() (

@ProductScoped stringCenter: StringCenter,

externalStrings: HomeMixerExternalStrings) {

def apply(

candidateFeatures: FeatureMap

): Option[FeedbackAction] = {

val isOutOfNetwork = !candidateFeatures.getOrElse(InNetworkFeature, true)

val validFollowedByUserIds =

candidateFeatures.getOrElse(SGSValidFollowedByUserIdsFeature, Nil)

val validLikedByUserIds =

candidateFeatures

.getOrElse(SGSValidLikedByUserIdsFeature, Nil)

.filter(

candidateFeatures.getOrElse(PerspectiveFilteredLikedByUserIdsFeature, Nil).toSet.contains)

if (isOutOfNetwork && validLikedByUserIds.isEmpty && validFollowedByUserIds.isEmpty) {

val topicIdSocialContext = candidateFeatures.getOrElse(TopicIdSocialContextFeature, None)

val topicContextFunctionalityType =

candidateFeatures.getOrElse(TopicContextFunctionalityTypeFeature, None)

(topicIdSocialContext, topicContextFunctionalityType) match {

case (Some(topicId), Some(topicContextFunctionalityType))

if topicContextFunctionalityType == RecommendationTopicContextFunctionalityType ||

topicContextFunctionalityType == RecWithEducationTopicContextFunctionalityType =>

Some(

FeedbackAction(

feedbackType = RichBehavior,

prompt = None,

confirmation = None,

childFeedbackActions = None,

feedbackUrl = None,

hasUndoAction = Some(true),

confirmationDisplayType = None,

clientEventInfo = None,

icon = None,

richBehavior =

Some(RichFeedbackBehaviorMarkNotInterestedTopic(topicId = topicId.toString)),

subprompt = None,

encodedFeedbackRequest = None

)

)

case \_ => None

}

} else {

None

}

}

}