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

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

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

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

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

import com.twitter.stringcenter.client.StringCenter

import com.twitter.stringcenter.client.core.ExternalString

private[decorator] case class SocialContextIdAndScreenName(

socialContextId: Long,

screenName: String)

object EngagerSocialContextBuilder {

private val UserIdRequestParamName = "user\_id"

private val DirectInjectionContentSourceRequestParamName = "dis"

private val DirectInjectionIdRequestParamName = "diid"

private val DirectInjectionContentSourceSocialProofUsers = "socialproofusers"

private val SocialProofUrl = ""

}

case class EngagerSocialContextBuilder(

contextType: GeneralContextType,

stringCenter: StringCenter,

oneUserString: ExternalString,

twoUsersString: ExternalString,

moreUsersString: ExternalString,

timelineTitle: ExternalString) {

import EngagerSocialContextBuilder.\_

def apply(

socialContextIds: Seq[Long],

query: PipelineQuery,

candidateFeatures: FeatureMap

): Option[SocialContext] = {

val realNames = candidateFeatures.getOrElse(RealNamesFeature, Map.empty[Long, String])

val validSocialContextIdAndScreenNames = socialContextIds.flatMap { socialContextId =>

realNames

.get(socialContextId).map(screenName =>

SocialContextIdAndScreenName(socialContextId, screenName))

}

validSocialContextIdAndScreenNames match {

case Seq(user) =>

val socialContextString =

stringCenter.prepare(oneUserString, Map("user" -> user.screenName))

Some(mkOneUserSocialContext(socialContextString, user.socialContextId))

case Seq(firstUser, secondUser) =>

val socialContextString =

stringCenter

.prepare(

twoUsersString,

Map("user1" -> firstUser.screenName, "user2" -> secondUser.screenName))

Some(

mkManyUserSocialContext(

socialContextString,

query.getRequiredUserId,

validSocialContextIdAndScreenNames.map(\_.socialContextId)))

case firstUser +: otherUsers =>

val otherUsersCount = otherUsers.size

val socialContextString =

stringCenter

.prepare(

moreUsersString,

Map("user" -> firstUser.screenName, "count" -> otherUsersCount))

Some(

mkManyUserSocialContext(

socialContextString,

query.getRequiredUserId,

validSocialContextIdAndScreenNames.map(\_.socialContextId)))

case \_ => None

}

}

private def mkOneUserSocialContext(socialContextString: String, userId: Long): GeneralContext = {

GeneralContext(

contextType = contextType,

text = socialContextString,

url = None,

contextImageUrls = None,

landingUrl = Some(

Url(

urlType = DeepLink,

url = "",

urtEndpointOptions = None

)

)

)

}

private def mkManyUserSocialContext(

socialContextString: String,

viewerId: Long,

socialContextIds: Seq[Long]

): GeneralContext = {

GeneralContext(

contextType = contextType,

text = socialContextString,

url = None,

contextImageUrls = None,

landingUrl = Some(

Url(

urlType = UrtEndpoint,

url = SocialProofUrl,

urtEndpointOptions = Some(UrtEndpointOptions(

requestParams = Some(Map(

UserIdRequestParamName -> viewerId.toString,

DirectInjectionContentSourceRequestParamName -> DirectInjectionContentSourceSocialProofUsers,

DirectInjectionIdRequestParamName -> socialContextIds.mkString(",")

)),

title = Some(stringCenter.prepare(timelineTitle)),

cacheId = None,

subtitle = None

))

))

)

}

}