package com.twitter.product\_mixer.component\_library.pipeline.candidate.who\_to\_follow\_module

import com.twitter.account\_recommendations\_mixer.{thriftscala => t}

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

import com.twitter.product\_mixer.core.functional\_component.marshaller.request.ClientContextMarshaller

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineQueryTransformer

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

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.BadRequest

import com.twitter.timelines.configapi.Param

object WhoToFollowArmCandidatePipelineQueryTransformer {

val HomeDisplayLocation = "timeline"

val HomeReverseChronDisplayLocation = "timeline\_reverse\_chron"

val ProfileDisplayLocation = "profile\_timeline"

}

case class WhoToFollowArmCandidatePipelineQueryTransformer[-Query <: PipelineQuery](

displayLocationParam: Param[String],

excludedUserIdsFeature: Option[Feature[PipelineQuery, Seq[Long]]],

profileUserIdFeature: Option[Feature[PipelineQuery, Long]])

extends CandidatePipelineQueryTransformer[Query, t.AccountRecommendationsMixerRequest] {

override def transform(input: Query): t.AccountRecommendationsMixerRequest = {

input.params(displayLocationParam) match {

case WhoToFollowArmCandidatePipelineQueryTransformer.HomeReverseChronDisplayLocation =>

t.AccountRecommendationsMixerRequest(

clientContext = ClientContextMarshaller(input.clientContext),

product = t.Product.HomeReverseChronWhoToFollow,

productContext = Some(

t.ProductContext.HomeReverseChronWhoToFollowProductContext(

t.HomeReverseChronWhoToFollowProductContext(

wtfReactiveContext = Some(getWhoToFollowReactiveContext(input))

)))

)

case WhoToFollowArmCandidatePipelineQueryTransformer.HomeDisplayLocation =>

t.AccountRecommendationsMixerRequest(

clientContext = ClientContextMarshaller(input.clientContext),

product = t.Product.HomeWhoToFollow,

productContext = Some(

t.ProductContext.HomeWhoToFollowProductContext(

t.HomeWhoToFollowProductContext(

wtfReactiveContext = Some(getWhoToFollowReactiveContext(input))

)))

)

case WhoToFollowArmCandidatePipelineQueryTransformer.ProfileDisplayLocation =>

t.AccountRecommendationsMixerRequest(

clientContext = ClientContextMarshaller(input.clientContext),

product = t.Product.ProfileWhoToFollow,

productContext = Some(

t.ProductContext.ProfileWhoToFollowProductContext(t.ProfileWhoToFollowProductContext(

wtfReactiveContext = Some(getWhoToFollowReactiveContext(input)),

profileUserId = profileUserIdFeature

.flatMap(feature => input.features.map(\_.get(feature)))

.getOrElse(throw PipelineFailure(BadRequest, "profileUserId not provided")),

)))

)

case displayLocation =>

throw PipelineFailure(BadRequest, s"display location $displayLocation not supported")

}

}

private def getWhoToFollowReactiveContext(

input: Query

): t.WhoToFollowReactiveContext = {

t.WhoToFollowReactiveContext(

excludedUserIds = excludedUserIdsFeature.flatMap(feature =>

input.features

.map(\_.get(feature))),

)

}

}