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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.metadata.WhoToFollowFeedbackActionInfoBuilder

import com.twitter.product\_mixer.component\_library.model.candidate.UserCandidate

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

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.metadata.BaseFeedbackActionInfoBuilder

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

import com.twitter.stringcenter.client.StringCenter

import javax.inject.Inject

import javax.inject.Provider

import javax.inject.Singleton

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

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

import com.twitter.timelines.service.{thriftscala => tl}

import com.twitter.timelines.util.FeedbackRequestSerializer

import com.twitter.timelineservice.suggests.thriftscala.SuggestType

import com.twitter.timelineservice.thriftscala.FeedbackType

object HomeWhoToSubscribeFeedbackActionInfoBuilder {

private val FeedbackMetadata = tl.FeedbackMetadata(

injectionType = Some(SuggestType.WhoToSubscribe),

engagementType = None,

entityIds = Seq.empty,

ttlMs = None

)

private val FeedbackRequest =

tl.DefaultFeedbackRequest2(FeedbackType.SeeFewer, FeedbackMetadata)

private val EncodedFeedbackRequest =

FeedbackRequestSerializer.serialize(tl.FeedbackRequest.DefaultFeedbackRequest2(FeedbackRequest))

}

@Singleton

case class HomeWhoToSubscribeFeedbackActionInfoBuilder @Inject() (

feedbackStrings: FeedbackStrings,

@ProductScoped stringCenterProvider: Provider[StringCenter])

extends BaseFeedbackActionInfoBuilder[PipelineQuery, UserCandidate] {

private val whoToSubscribeFeedbackActionInfoBuilder = WhoToFollowFeedbackActionInfoBuilder(

seeLessOftenFeedbackString = feedbackStrings.seeLessOftenFeedbackString,

seeLessOftenConfirmationFeedbackString = feedbackStrings.seeLessOftenConfirmationFeedbackString,

stringCenter = stringCenterProvider.get(),

encodedFeedbackRequest =

Some(HomeWhoToSubscribeFeedbackActionInfoBuilder.EncodedFeedbackRequest)

)

override def apply(

query: PipelineQuery,

candidate: UserCandidate,

candidateFeatures: FeatureMap

): Option[FeedbackActionInfo] =

whoToSubscribeFeedbackActionInfoBuilder.apply(query, candidate, candidateFeatures)

}