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

import com.twitter.conversions.DurationOps.\_

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

import com.twitter.home\_mixer.param.HomeGlobalParams.EnableNahFeedbackInfoParam

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

import com.twitter.home\_mixer.util.CandidatesUtil

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.model.marshalling.response.urt.icon.Frown

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

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

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

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

import com.twitter.stringcenter.client.StringCenter

import com.twitter.timelines.common.{thriftscala => tlc}

import com.twitter.timelineservice.model.FeedbackInfo

import com.twitter.timelineservice.model.FeedbackMetadata

import com.twitter.timelineservice.{thriftscala => tls}

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class DontLikeFeedbackActionBuilder @Inject() (

@ProductScoped stringCenter: StringCenter,

externalStrings: HomeMixerExternalStrings,

authorChildFeedbackActionBuilder: AuthorChildFeedbackActionBuilder,

retweeterChildFeedbackActionBuilder: RetweeterChildFeedbackActionBuilder,

notRelevantChildFeedbackActionBuilder: NotRelevantChildFeedbackActionBuilder,

unfollowUserChildFeedbackActionBuilder: UnfollowUserChildFeedbackActionBuilder,

muteUserChildFeedbackActionBuilder: MuteUserChildFeedbackActionBuilder,

blockUserChildFeedbackActionBuilder: BlockUserChildFeedbackActionBuilder,

reportTweetChildFeedbackActionBuilder: ReportTweetChildFeedbackActionBuilder) {

def apply(

query: PipelineQuery,

candidate: TweetCandidate,

candidateFeatures: FeatureMap

): Option[FeedbackAction] = {

CandidatesUtil.getOriginalAuthorId(candidateFeatures).map { authorId =>

val feedbackEntities = Seq(

tlc.FeedbackEntity.TweetId(candidate.id),

tlc.FeedbackEntity.UserId(authorId)

)

val feedbackMetadata = FeedbackMetadata(

engagementType = None,

entityIds = feedbackEntities,

ttl = Some(30.days)

)

val feedbackUrl = FeedbackInfo.feedbackUrl(

feedbackType = tls.FeedbackType.DontLike,

feedbackMetadata = feedbackMetadata,

injectionType = candidateFeatures.getOrElse(SuggestTypeFeature, None)

)

val childFeedbackActions = if (query.params(EnableNahFeedbackInfoParam)) {

Seq(

unfollowUserChildFeedbackActionBuilder(candidateFeatures),

muteUserChildFeedbackActionBuilder(candidateFeatures),

blockUserChildFeedbackActionBuilder(candidateFeatures),

reportTweetChildFeedbackActionBuilder(candidate)

).flatten

} else {

Seq(

authorChildFeedbackActionBuilder(candidateFeatures),

retweeterChildFeedbackActionBuilder(candidateFeatures),

notRelevantChildFeedbackActionBuilder(candidate, candidateFeatures)

).flatten

}

FeedbackAction(

feedbackType = DontLike,

prompt = Some(stringCenter.prepare(externalStrings.dontLikeString)),

confirmation = Some(stringCenter.prepare(externalStrings.dontLikeConfirmationString)),

childFeedbackActions =

if (childFeedbackActions.nonEmpty) Some(childFeedbackActions) else None,

feedbackUrl = Some(feedbackUrl),

hasUndoAction = Some(true),

confirmationDisplayType = None,

clientEventInfo = None,

icon = Some(Frown),

richBehavior = None,

subprompt = None,

encodedFeedbackRequest = None

)

}

}

}