package com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.user

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.user.UserCandidateUrtItemBuilder.UserClientEventInfoElement

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

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

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

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

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.item.user.BaseUserReactiveTriggersBuilder

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

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

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.promoted.BasePromotedMetadataBuilder

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.social\_context.BaseSocialContextBuilder

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.user.User

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.user.UserDisplayType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.user.UserItem

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

object UserCandidateUrtItemBuilder {

val UserClientEventInfoElement: String = "user"

}

case class UserCandidateUrtItemBuilder[Query <: PipelineQuery, UserCandidate <: BaseUserCandidate](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, UserCandidate],

feedbackActionInfoBuilder: Option[

BaseFeedbackActionInfoBuilder[Query, UserCandidate]

] = None,

displayType: UserDisplayType = User,

promotedMetadataBuilder: Option[BasePromotedMetadataBuilder[Query, UserCandidate]] = None,

socialContextBuilder: Option[BaseSocialContextBuilder[Query, UserCandidate]] = None,

reactiveTriggersBuilder: Option[BaseUserReactiveTriggersBuilder[Query, UserCandidate]] = None,

enableReactiveBlending: Option[Boolean] = None)

extends CandidateUrtEntryBuilder[Query, UserCandidate, UserItem] {

override def apply(

query: Query,

userCandidate: UserCandidate,

candidateFeatures: FeatureMap

): UserItem = {

val isMarkUnread = candidateFeatures.getTry(IsMarkUnreadFeature).toOption

UserItem(

id = userCandidate.id,

sortIndex = None, // Sort indexes are automatically set in the domain marshaller phase

clientEventInfo = clientEventInfoBuilder(

query,

userCandidate,

candidateFeatures,

Some(UserClientEventInfoElement)),

feedbackActionInfo =

feedbackActionInfoBuilder.flatMap(\_.apply(query, userCandidate, candidateFeatures)),

isMarkUnread = isMarkUnread,

displayType = displayType,

promotedMetadata =

promotedMetadataBuilder.flatMap(\_.apply(query, userCandidate, candidateFeatures)),

socialContext =

socialContextBuilder.flatMap(\_.apply(query, userCandidate, candidateFeatures)),

reactiveTriggers =

reactiveTriggersBuilder.flatMap(\_.apply(query, userCandidate, candidateFeatures)),

enableReactiveBlending = enableReactiveBlending

)

}

}