package com.twitter.product\_mixer.component\_library.decorator.urt.builder.social\_context

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

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

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

import com.twitter.product\_mixer.core.model.common.UniversalNoun

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

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

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

/\*\*

\* Use this Builder to create Product Mixer [[SocialContext]] objects when you have a

\* Timeline Service Thrift [[SocialContext]] feature that you want to convert

\*/

case class FeatureSocialContextBuilder(

socialContextFeature: Feature[\_, Option[t.SocialContext]])

extends BaseSocialContextBuilder[PipelineQuery, UniversalNoun[Any]] {

override def apply(

query: PipelineQuery,

candidate: UniversalNoun[Any],

candidateFeatures: FeatureMap

): Option[SocialContext] = {

candidateFeatures.getOrElse(socialContextFeature, None).map {

case t.SocialContext.GeneralContext(context) =>

val contextType = context.contextType match {

case t.ContextType.Like => LikeGeneralContextType

case t.ContextType.Follow => FollowGeneralContextType

case t.ContextType.Moment => MomentGeneralContextType

case t.ContextType.Reply => ReplyGeneralContextType

case t.ContextType.Conversation => ConversationGeneralContextType

case t.ContextType.Pin => PinGeneralContextType

case t.ContextType.TextOnly => TextOnlyGeneralContextType

case t.ContextType.Facepile => FacePileGeneralContextType

case t.ContextType.Megaphone => MegaPhoneGeneralContextType

case t.ContextType.Bird => BirdGeneralContextType

case t.ContextType.Feedback => FeedbackGeneralContextType

case t.ContextType.Topic => TopicGeneralContextType

case t.ContextType.List => ListGeneralContextType

case t.ContextType.Retweet => RetweetGeneralContextType

case t.ContextType.Location => LocationGeneralContextType

case t.ContextType.Community => CommunityGeneralContextType

case t.ContextType.SmartBlockExpiration => SmartblockExpirationGeneralContextType

case t.ContextType.Trending => TrendingGeneralContextType

case t.ContextType.Sparkle => SparkleGeneralContextType

case t.ContextType.Spaces => SpacesGeneralContextType

case t.ContextType.ReplyPin => ReplyPinGeneralContextType

case t.ContextType.NewUser => NewUserGeneralContextType

case t.ContextType.EnumUnknownContextType(field) =>

throw new UnsupportedOperationException(s"Unknown context type: $field")

}

val landingUrl = context.landingUrl.map { url =>

val endpointOptions = url.urtEndpointOptions.map { options =>

UrtEndpointOptions(

requestParams = options.requestParams.map(\_.toMap),

title = options.title,

cacheId = options.cacheId,

subtitle = options.subtitle

)

}

val urlType = url.urlType match {

case t.UrlType.ExternalUrl => ExternalUrl

case t.UrlType.DeepLink => DeepLink

case t.UrlType.UrtEndpoint => UrtEndpoint

case t.UrlType.EnumUnknownUrlType(field) =>

throw new UnsupportedOperationException(s"Unknown url type: $field")

}

Url(urlType = urlType, url = url.url, urtEndpointOptions = endpointOptions)

}

GeneralContext(

text = context.text,

contextType = contextType,

url = context.url,

contextImageUrls = context.contextImageUrls.map(\_.toList),

landingUrl = landingUrl

)

case t.SocialContext.TopicContext(context) =>

val functionalityType = context.functionalityType match {

case t.TopicContextFunctionalityType.Basic =>

BasicTopicContextFunctionalityType

case t.TopicContextFunctionalityType.Recommendation =>

RecommendationTopicContextFunctionalityType

case t.TopicContextFunctionalityType.RecWithEducation =>

RecWithEducationTopicContextFunctionalityType

case t.TopicContextFunctionalityType.EnumUnknownTopicContextFunctionalityType(field) =>

throw new UnsupportedOperationException(s"Unknown functionality type: $field")

}

TopicContext(

topicId = context.topicId,

functionalityType = Some(functionalityType)

)

case t.SocialContext.UnknownUnionField(field) =>

throw new UnsupportedOperationException(s"Unknown social context: $field")

}

}

}