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

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.bijection.Base64String

import com.twitter.bijection.{Injection => Serializer}

import com.twitter.interests\_mixer.model.request.{HasTopicId => InterestsMixerHasTopicId}

import com.twitter.explore\_mixer.model.request.{HasTopicId => ExploreMixerHasTopicId}

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.functional\_component.decorator.urt.builder.metadata.BaseClientEventDetailsBuilder

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

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

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

import com.twitter.suggests.controller\_data.home\_tweets.thriftscala.HomeTweetsControllerData

import com.twitter.suggests.controller\_data.home\_tweets.v1.thriftscala.{

HomeTweetsControllerData => HomeTweetsControllerDataV1

}

import com.twitter.suggests.controller\_data.thriftscala.ControllerData

import com.twitter.suggests.controller\_data.v2.thriftscala.{ControllerData => ControllerDataV2}

object TopicTweetClientEventDetailsBuilder {

implicit val ByteSerializer: Serializer[ControllerData, Array[Byte]] =

BinaryScalaCodec(ControllerData)

val ControllerDataSerializer: Serializer[ControllerData, String] =

Serializer.connect[ControllerData, Array[Byte], Base64String, String]

}

case class TopicTweetClientEventDetailsBuilder[-Query <: PipelineQuery]()

extends BaseClientEventDetailsBuilder[Query, TweetCandidate] {

import TopicTweetClientEventDetailsBuilder.\_

override def apply(

query: Query,

topicTweetCandidate: TweetCandidate,

candidateFeatures: FeatureMap

): Option[ClientEventDetails] =

Some(

ClientEventDetails(

conversationDetails = None,

timelinesDetails = Some(

TimelinesDetails(

injectionType = None,

controllerData = buildControllerData(getTopicId(query)),

sourceData = None)),

articleDetails = None,

liveEventDetails = None,

commerceDetails = None

))

private def getTopicId(query: Query): Option[Long] = {

query match {

case query: InterestsMixerHasTopicId => query.topicId

case query: ExploreMixerHasTopicId => query.topicId

case \_ => None

}

}

private def buildControllerData(topicId: Option[Long]): Option[String] =

Some(

ControllerData

.V2(ControllerDataV2.HomeTweets(HomeTweetsControllerData.V1(

HomeTweetsControllerDataV1(tweetTypesBitmap = 0L, topicId = topicId)))))

.map(ControllerDataSerializer)

}