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

import com.twitter.bijection.Base64String

import com.twitter.bijection.scrooge.BinaryScalaCodec

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

import com.twitter.finagle.tracing.Trace

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

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

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

import com.twitter.joinkey.context.RequestJoinKeyContext

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.common.UniversalNoun

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

import com.twitter.suggests.controller\_data.TweetTypeGenerator

import com.twitter.suggests.controller\_data.home\_tweets.v1.{thriftscala => v1ht}

import com.twitter.suggests.controller\_data.home\_tweets.{thriftscala => ht}

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

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

object HomeClientEventDetailsBuilder {

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

BinaryScalaCodec(ControllerData)

val ControllerDataSerializer: Serializer[ControllerData, String] =

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

/\*\*

\* define getRequestJoinId as a method(def) rather than a val because each new request

\* needs to call the context to update the id.

\*/

private def getRequestJoinId(): Option[Long] =

RequestJoinKeyContext.current.flatMap(\_.requestJoinId)

}

case class HomeClientEventDetailsBuilder[-Query <: PipelineQuery, -Candidate <: UniversalNoun[Any]](

) extends BaseClientEventDetailsBuilder[Query, Candidate]

with TweetTypeGenerator[FeatureMap] {

import HomeClientEventDetailsBuilder.\_

override def apply(

query: Query,

candidate: Candidate,

candidateFeatures: FeatureMap

): Option[ClientEventDetails] = {

val tweetTypesBitmaps = mkTweetTypesBitmaps(

Home.TweetTypeIdxMap,

HomeTweetTypePredicates.PredicateMap,

candidateFeatures)

val tweetTypesListBytes = mkItemTypesBitmapsV2(

Home.TweetTypeIdxMap,

HomeTweetTypePredicates.PredicateMap,

candidateFeatures)

val candidateSourceId =

candidateFeatures.getOrElse(CandidateSourceIdFeature, None).map(\_.value.toByte)

val homeTweetsControllerDataV1 = v1ht.HomeTweetsControllerData(

tweetTypesBitmap = tweetTypesBitmaps.getOrElse(0, 0L),

tweetTypesBitmapContinued1 = tweetTypesBitmaps.get(1),

candidateTweetSourceId = candidateSourceId,

traceId = Some(Trace.id.traceId.toLong),

injectedPosition = candidateFeatures.getOrElse(PositionFeature, None),

tweetTypesListBytes = Some(tweetTypesListBytes),

requestJoinId = getRequestJoinId(),

)

val serializedControllerData = ControllerDataSerializer(

ControllerData.V2(

ControllerDataV2.HomeTweets(ht.HomeTweetsControllerData.V1(homeTweetsControllerDataV1))))

val clientEventDetails = ClientEventDetails(

conversationDetails = None,

timelinesDetails = Some(

TimelinesDetails(

injectionType = candidateFeatures.getOrElse(SuggestTypeFeature, None).map(\_.name),

controllerData = Some(serializedControllerData),

sourceData = None)),

articleDetails = None,

liveEventDetails = None,

commerceDetails = None

)

Some(clientEventDetails)

}

}