package com.twitter.product\_mixer.component\_library.pipeline.candidate.who\_to\_follow\_module

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.bijection.Base64String

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

import com.twitter.hermit.internal.thriftscala.HermitTrackingToken

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

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.metadata.BaseClientEventDetailsBuilder

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

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.servo.cache.ThriftSerializer

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

import com.twitter.util.Try

import org.apache.thrift.protocol.TBinaryProtocol

object WhoToFollowClientEventDetailsBuilder {

val InjectionType = "WhoToFollow"

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

BinaryScalaCodec(ControllerData)

private val TrackingTokenSerializer =

new ThriftSerializer[HermitTrackingToken](HermitTrackingToken, new TBinaryProtocol.Factory())

val ControllerDataSerializer: Serializer[ControllerData, String] =

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

def deserializeTrackingToken(token: Option[String]): Option[HermitTrackingToken] =

token.flatMap(t => Try(TrackingTokenSerializer.fromString(t)).toOption)

def serializeControllerData(cd: ControllerData): String = ControllerDataSerializer(cd)

}

case class WhoToFollowClientEventDetailsBuilder[-Query <: PipelineQuery](

trackingTokenFeature: Feature[\_, Option[String]],

) extends BaseClientEventDetailsBuilder[Query, UserCandidate] {

override def apply(

query: Query,

candidate: UserCandidate,

candidateFeatures: FeatureMap

): Option[ClientEventDetails] = {

val serializedTrackingToken = candidateFeatures.getOrElse(trackingTokenFeature, None)

val controllerData = WhoToFollowClientEventDetailsBuilder

.deserializeTrackingToken(serializedTrackingToken)

.flatMap(\_.controllerData)

.map(WhoToFollowClientEventDetailsBuilder.serializeControllerData)

Some(

ClientEventDetails(

conversationDetails = None,

timelinesDetails = Some(

TimelinesDetails(

injectionType = Some(WhoToFollowClientEventDetailsBuilder.InjectionType),

controllerData = controllerData,

sourceData = serializedTrackingToken)),

articleDetails = None,

liveEventDetails = None,

commerceDetails = None

))

}

}