package com.twitter.home\_mixer.product.for\_you.feature\_hydrator

import com.twitter.home\_mixer.marshaller.timelines.DeviceContextMarshaller

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

import com.twitter.home\_mixer.model.request.DeviceContext

import com.twitter.home\_mixer.model.request.HasDeviceContext

import com.twitter.home\_mixer.service.HomeMixerAlertConfig

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

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

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

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.QueryFeatureHydrator

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

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

import com.twitter.stitch.Stitch

import com.twitter.stitch.timelineservice.TimelineService

import com.twitter.timelineservice.{thriftscala => t}

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class TimelineServiceTweetsQueryFeatureHydrator @Inject() (

timelineService: TimelineService,

deviceContextMarshaller: DeviceContextMarshaller)

extends QueryFeatureHydrator[PipelineQuery with HasDeviceContext] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("TimelineServiceTweets")

override val features: Set[Feature[\_, \_]] = Set(TimelineServiceTweetsFeature)

private val MaxTimelineServiceTweets = 200

override def hydrate(query: PipelineQuery with HasDeviceContext): Stitch[FeatureMap] = {

val deviceContext = query.deviceContext.getOrElse(DeviceContext.Empty)

val timelineQueryOptions = t.TimelineQueryOptions(

contextualUserId = query.clientContext.userId,

deviceContext = Some(deviceContextMarshaller(deviceContext, query.clientContext))

)

val timelineServiceQuery = t.TimelineQuery(

timelineType = t.TimelineType.Home,

timelineId = query.getRequiredUserId,

maxCount = MaxTimelineServiceTweets.toShort,

cursor2 = None,

options = Some(timelineQueryOptions),

timelineId2 = query.clientContext.userId.map(t.TimelineId(t.TimelineType.Home, \_, None)),

)

timelineService.getTimeline(timelineServiceQuery).map { timeline =>

val tweets = timeline.entries.collect {

case t.TimelineEntry.Tweet(tweet) => tweet.statusId

}

FeatureMapBuilder().add(TimelineServiceTweetsFeature, tweets).build()

}

}

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(99.7)

)

}