package com.twitter.home\_mixer.functional\_component.feature\_hydrator

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

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

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

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.user\_session\_store.ReadRequest

import com.twitter.user\_session\_store.ReadWriteUserSessionStore

import com.twitter.user\_session\_store.UserSessionDataset

import com.twitter.user\_session\_store.UserSessionDataset.UserSessionDataset

import com.twitter.util.Time

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class LastNonPollingTimeQueryFeatureHydrator @Inject() (

userSessionStore: ReadWriteUserSessionStore)

extends QueryFeatureHydrator[PipelineQuery] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("LastNonPollingTime")

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

FollowingLastNonPollingTimeFeature,

LastNonPollingTimeFeature,

NonPollingTimesFeature

)

private val datasets: Set[UserSessionDataset] = Set(UserSessionDataset.NonPollingTimes)

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

userSessionStore

.read(ReadRequest(query.getRequiredUserId, datasets))

.map { userSession =>

val nonPollingTimestamps = userSession.flatMap(\_.nonPollingTimestamps)

val lastNonPollingTime = nonPollingTimestamps

.flatMap(\_.nonPollingTimestampsMs.headOption)

.map(Time.fromMilliseconds)

val followingLastNonPollingTime = nonPollingTimestamps

.flatMap(\_.mostRecentHomeLatestNonPollingTimestampMs)

.map(Time.fromMilliseconds)

val nonPollingTimes = nonPollingTimestamps

.map(\_.nonPollingTimestampsMs)

.getOrElse(Seq.empty)

FeatureMapBuilder()

.add(FollowingLastNonPollingTimeFeature, followingLastNonPollingTime)

.add(LastNonPollingTimeFeature, lastNonPollingTime)

.add(NonPollingTimesFeature, nonPollingTimes)

.build()

}

}

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(99.9)

)

}