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

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

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

import com.twitter.timelinemixer.clients.manhattan.InjectionHistoryClient

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.timelinemixer.clients.manhattan.DismissInfo

import com.twitter.timelineservice.suggests.thriftscala.SuggestType

import javax.inject.Inject

import javax.inject.Singleton

object DismissInfoQueryFeatureHydrator {

val DismissInfoSuggestTypes = Seq(SuggestType.WhoToFollow)

}

@Singleton

case class DismissInfoQueryFeatureHydrator @Inject() (

dismissInfoClient: InjectionHistoryClient)

extends QueryFeatureHydrator[PipelineQuery] {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier("DismissInfo")

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

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

Stitch.callFuture {

dismissInfoClient

.readDismissInfoEntries(

query.getRequiredUserId,

DismissInfoQueryFeatureHydrator.DismissInfoSuggestTypes).map { response =>

val dismissInfoMap = response.mapValues(DismissInfo.fromThrift)

FeatureMapBuilder().add(DismissInfoFeature, dismissInfoMap).build()

}

}

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(99.8, 50, 60, 60)

)

}