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

import com.twitter.conversions.DurationOps.\_

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

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

import com.twitter.home\_mixer.param.HomeGlobalParams.ImpressionBloomFilterFalsePositiveRateParam

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.timelines.clients.manhattan.store.ManhattanStoreClient

import com.twitter.timelines.impressionbloomfilter.{thriftscala => blm}

import com.twitter.timelines.impressionstore.impressionbloomfilter.ImpressionBloomFilter

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class ImpressionBloomFilterQueryFeatureHydrator[

Query <: PipelineQuery with HasSeenTweetIds] @Inject() (

bloomFilterClient: ManhattanStoreClient[

blm.ImpressionBloomFilterKey,

blm.ImpressionBloomFilterSeq

]) extends QueryFeatureHydrator[Query] {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier(

"ImpressionBloomFilter")

private val ImpressionBloomFilterTTL = 7.day

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

private val SurfaceArea = blm.SurfaceArea.HomeTimeline

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

val userId = query.getRequiredUserId

bloomFilterClient

.get(blm.ImpressionBloomFilterKey(userId, SurfaceArea))

.map(\_.getOrElse(blm.ImpressionBloomFilterSeq(Seq.empty)))

.map { bloomFilterSeq =>

val updatedBloomFilterSeq =

if (query.seenTweetIds.forall(\_.isEmpty)) bloomFilterSeq

else {

ImpressionBloomFilter.addSeenTweetIds(

surfaceArea = SurfaceArea,

tweetIds = query.seenTweetIds.get,

bloomFilterSeq = bloomFilterSeq,

timeToLive = ImpressionBloomFilterTTL,

falsePositiveRate = query.params(ImpressionBloomFilterFalsePositiveRateParam)

)

}

FeatureMapBuilder().add(ImpressionBloomFilterFeature, updatedBloomFilterSeq).build()

}

}

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(99.8)

)

}