package com.twitter.product\_mixer.component\_library.feature\_hydrator.query.impressed\_tweets

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

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

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.storehaus.ReadableStore

import com.twitter.timelines.impressionstore.thriftscala.ImpressionList

import com.twitter.util.Future

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Query Feature to store ids of the tweets impressed by the user.

\*/

case object ImpressedTweets extends FeatureWithDefaultOnFailure[PipelineQuery, Seq[Long]] {

override val defaultValue: Seq[Long] = Seq.empty

}

/\*\*

\* Enrich the query with a list of tweet ids that the user has already seen.

\*/

@Singleton

case class ImpressedTweetsQueryFeatureHydrator @Inject() (

tweetImpressionStore: ReadableStore[Long, ImpressionList])

extends QueryFeatureHydrator[PipelineQuery] {

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

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

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

query.getOptionalUserId match {

case Some(userId) =>

val featureMapResult: Future[FeatureMap] = tweetImpressionStore

.get(userId).map { impressionListOpt =>

val tweetIdsOpt = for {

impressionList <- impressionListOpt

impressions <- impressionList.impressions

} yield {

impressions.map(\_.tweetId)

}

val tweetIds = tweetIdsOpt.getOrElse(Seq.empty)

FeatureMapBuilder().add(ImpressedTweets, tweetIds).build()

}

Stitch.callFuture(featureMapResult)

// Non-logged-in users do not have userId, returns empty feature

case None =>

val featureMapResult = FeatureMapBuilder().add(ImpressedTweets, Seq.empty).build()

Stitch.value(featureMapResult)

}

}

}