package com.twitter.follow\_recommendations.common.predicates.user\_activity

import com.twitter.core\_workflows.user\_model.thriftscala.UserState

import com.twitter.decider.Decider

import com.twitter.decider.RandomRecipient

import com.twitter.finagle.Memcached.Client

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.base.Predicate

import com.twitter.follow\_recommendations.common.base.PredicateResult

import com.twitter.follow\_recommendations.common.base.StatsUtil

import com.twitter.follow\_recommendations.common.clients.cache.MemcacheClient

import com.twitter.follow\_recommendations.common.clients.cache.ThriftEnumOptionBijection

import com.twitter.follow\_recommendations.common.models.CandidateUser

import com.twitter.follow\_recommendations.common.models.FilterReason

import com.twitter.follow\_recommendations.configapi.deciders.DeciderKey

import com.twitter.product\_mixer.core.model.marshalling.request.HasClientContext

import com.twitter.stitch.Stitch

import com.twitter.strato.generated.client.onboarding.UserRecommendabilityWithLongKeysOnUserClientColumn

import com.twitter.timelines.configapi.HasParams

import javax.inject.Inject

import javax.inject.Singleton

abstract case class UserStateActivityPredicate(

userRecommendabilityClient: UserRecommendabilityWithLongKeysOnUserClientColumn,

validCandidateStates: Set[UserState],

client: Client,

statsReceiver: StatsReceiver,

decider: Decider = Decider.False)

extends Predicate[(HasParams with HasClientContext, CandidateUser)] {

private val stats: StatsReceiver = statsReceiver.scope(this.getClass.getSimpleName)

// client to memcache cluster

val bijection = new ThriftEnumOptionBijection[UserState](UserState.apply)

val memcacheClient = MemcacheClient[Option[UserState]](

client = client,

dest = "/s/cache/follow\_recos\_service:twemcaches",

valueBijection = bijection,

ttl = UserActivityPredicateParams.CacheTTL,

statsReceiver = stats.scope("twemcache")

)

override def apply(

targetAndCandidate: (HasParams with HasClientContext, CandidateUser)

): Stitch[PredicateResult] = {

val userRecommendabilityFetcher = userRecommendabilityClient.fetcher

val (\_, candidate) = targetAndCandidate

val deciderKey: String = DeciderKey.EnableExperimentalCaching.toString

val enableDistributedCaching: Boolean = decider.isAvailable(deciderKey, Some(RandomRecipient))

val userStateStitch: Stitch[Option[UserState]] =

enableDistributedCaching match {

case true => {

memcacheClient.readThrough(

// add a key prefix to address cache key collisions

key = "UserActivityPredicate" + candidate.id.toString,

underlyingCall = () => queryUserRecommendable(candidate.id)

)

}

case false => queryUserRecommendable(candidate.id)

}

val resultStitch: Stitch[PredicateResult] =

userStateStitch.map { userStateOpt =>

userStateOpt match {

case Some(userState) => {

if (validCandidateStates.contains(userState)) {

PredicateResult.Valid

} else {

PredicateResult.Invalid(Set(FilterReason.MinStateNotMet))

}

}

case None => {

PredicateResult.Invalid(Set(FilterReason.MissingRecommendabilityData))

}

}

}

StatsUtil.profileStitch(resultStitch, stats.scope("apply"))

.rescue {

case e: Exception =>

stats.scope("rescued").counter(e.getClass.getSimpleName).incr()

Stitch(PredicateResult.Invalid(Set(FilterReason.FailOpen)))

}

}

def queryUserRecommendable(

userId: Long

): Stitch[Option[UserState]] = {

val userRecommendabilityFetcher = userRecommendabilityClient.fetcher

userRecommendabilityFetcher.fetch(userId).map { userCandidate =>

userCandidate.v.flatMap(\_.userState)

}

}

}

@Singleton

class MinStateUserActivityPredicate @Inject() (

userRecommendabilityClient: UserRecommendabilityWithLongKeysOnUserClientColumn,

client: Client,

statsReceiver: StatsReceiver)

extends UserStateActivityPredicate(

userRecommendabilityClient,

Set(

UserState.Light,

UserState.HeavyNonTweeter,

UserState.MediumNonTweeter,

UserState.HeavyTweeter,

UserState.MediumTweeter

),

client,

statsReceiver

)

@Singleton

class AllTweeterUserActivityPredicate @Inject() (

userRecommendabilityClient: UserRecommendabilityWithLongKeysOnUserClientColumn,

client: Client,

statsReceiver: StatsReceiver)

extends UserStateActivityPredicate(

userRecommendabilityClient,

Set(

UserState.HeavyTweeter,

UserState.MediumTweeter

),

client,

statsReceiver

)

@Singleton

class HeavyTweeterUserActivityPredicate @Inject() (

userRecommendabilityClient: UserRecommendabilityWithLongKeysOnUserClientColumn,

client: Client,

statsReceiver: StatsReceiver)

extends UserStateActivityPredicate(

userRecommendabilityClient,

Set(

UserState.HeavyTweeter

),

client,

statsReceiver

)

@Singleton

class NonNearZeroUserActivityPredicate @Inject() (

userRecommendabilityClient: UserRecommendabilityWithLongKeysOnUserClientColumn,

client: Client,

statsReceiver: StatsReceiver)

extends UserStateActivityPredicate(

userRecommendabilityClient,

Set(

UserState.New,

UserState.VeryLight,

UserState.Light,

UserState.MediumNonTweeter,

UserState.MediumTweeter,

UserState.HeavyNonTweeter,

UserState.HeavyTweeter

),

client,

statsReceiver

)