package com.twitter.tsp.common

import com.twitter.abdecider.LoggingABDecider

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.TargetUser

import com.twitter.frigate.common.candidate.TargetABDecider

import com.twitter.frigate.common.util.ABDeciderWithOverride

import com.twitter.gizmoduck.thriftscala.User

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi.Params

import com.twitter.tsp.thriftscala.TopicSocialProofRequest

import com.twitter.util.Future

case class DefaultRecTopicSocialProofTarget(

topicSocialProofRequest: TopicSocialProofRequest,

targetId: UserId,

user: Option[User],

abDecider: ABDeciderWithOverride,

params: Params

)(

implicit statsReceiver: StatsReceiver)

extends TargetUser

with TopicSocialProofRecRequest

with TargetABDecider {

override def globalStats: StatsReceiver = statsReceiver

override val targetUser: Future[Option[User]] = Future.value(user)

}

trait TopicSocialProofRecRequest {

tuc: TargetUser =>

val topicSocialProofRequest: TopicSocialProofRequest

}

case class RecTargetFactory(

abDecider: LoggingABDecider,

userStore: ReadableStore[UserId, User],

paramBuilder: ParamsBuilder,

statsReceiver: StatsReceiver) {

type RecTopicSocialProofTarget = DefaultRecTopicSocialProofTarget

def buildRecTopicSocialProofTarget(

request: TopicSocialProofRequest

): Future[RecTopicSocialProofTarget] = {

val userId = request.userId

userStore.get(userId).map { userOpt =>

val userRoles = userOpt.flatMap(\_.roles.map(\_.roles.toSet))

val context = request.context.copy(userId = Some(request.userId)) // override to make sure

val params = paramBuilder

.buildFromTopicListingViewerContext(Some(context), request.displayLocation, userRoles)

DefaultRecTopicSocialProofTarget(

request,

userId,

userOpt,

ABDeciderWithOverride(abDecider, None)(statsReceiver),

params

)(statsReceiver)

}

}

}