package com.twitter.visibility.builder.users

import com.twitter.finagle.stats.Counter

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.gizmoduck.thriftscala.AdvancedFilters

import com.twitter.gizmoduck.thriftscala.MentionFilter

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.UserSource

import com.twitter.visibility.features.ViewerFiltersDefaultProfileImage

import com.twitter.visibility.features.ViewerFiltersNewUsers

import com.twitter.visibility.features.ViewerFiltersNoConfirmedEmail

import com.twitter.visibility.features.ViewerFiltersNoConfirmedPhone

import com.twitter.visibility.features.ViewerFiltersNotFollowedBy

import com.twitter.visibility.features.ViewerMentionFilter

class ViewerAdvancedFilteringFeatures(userSource: UserSource, statsReceiver: StatsReceiver) {

private[this] val scopedStatsReceiver = statsReceiver.scope("viewer\_advanced\_filtering\_features")

private[this] val requests = scopedStatsReceiver.counter("requests")

private[this] val viewerFiltersNoConfirmedEmail =

scopedStatsReceiver.scope(ViewerFiltersNoConfirmedEmail.name).counter("requests")

private[this] val viewerFiltersNoConfirmedPhone =

scopedStatsReceiver.scope(ViewerFiltersNoConfirmedPhone.name).counter("requests")

private[this] val viewerFiltersDefaultProfileImage =

scopedStatsReceiver.scope(ViewerFiltersDefaultProfileImage.name).counter("requests")

private[this] val viewerFiltersNewUsers =

scopedStatsReceiver.scope(ViewerFiltersNewUsers.name).counter("requests")

private[this] val viewerFiltersNotFollowedBy =

scopedStatsReceiver.scope(ViewerFiltersNotFollowedBy.name).counter("requests")

private[this] val viewerMentionFilter =

scopedStatsReceiver.scope(ViewerMentionFilter.name).counter("requests")

def forViewerId(viewerId: Option[Long]): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

\_.withFeature(ViewerFiltersNoConfirmedEmail, viewerFiltersNoConfirmedEmail(viewerId))

.withFeature(ViewerFiltersNoConfirmedPhone, viewerFiltersNoConfirmedPhone(viewerId))

.withFeature(ViewerFiltersDefaultProfileImage, viewerFiltersDefaultProfileImage(viewerId))

.withFeature(ViewerFiltersNewUsers, viewerFiltersNewUsers(viewerId))

.withFeature(ViewerFiltersNotFollowedBy, viewerFiltersNotFollowedBy(viewerId))

.withFeature(ViewerMentionFilter, viewerMentionFilter(viewerId))

}

def viewerFiltersNoConfirmedEmail(viewerId: Option[Long]): Stitch[Boolean] =

viewerAdvancedFilters(viewerId, af => af.filterNoConfirmedEmail, viewerFiltersNoConfirmedEmail)

def viewerFiltersNoConfirmedPhone(viewerId: Option[Long]): Stitch[Boolean] =

viewerAdvancedFilters(viewerId, af => af.filterNoConfirmedPhone, viewerFiltersNoConfirmedPhone)

def viewerFiltersDefaultProfileImage(viewerId: Option[Long]): Stitch[Boolean] =

viewerAdvancedFilters(

viewerId,

af => af.filterDefaultProfileImage,

viewerFiltersDefaultProfileImage

)

def viewerFiltersNewUsers(viewerId: Option[Long]): Stitch[Boolean] =

viewerAdvancedFilters(viewerId, af => af.filterNewUsers, viewerFiltersNewUsers)

def viewerFiltersNotFollowedBy(viewerId: Option[Long]): Stitch[Boolean] =

viewerAdvancedFilters(viewerId, af => af.filterNotFollowedBy, viewerFiltersNotFollowedBy)

def viewerMentionFilter(viewerId: Option[Long]): Stitch[MentionFilter] = {

viewerMentionFilter.incr()

viewerId match {

case Some(id) =>

userSource.getMentionFilter(id).handle {

case NotFound =>

MentionFilter.Unfiltered

}

case \_ => Stitch.value(MentionFilter.Unfiltered)

}

}

private[this] def viewerAdvancedFilters(

viewerId: Option[Long],

advancedFilterCheck: AdvancedFilters => Boolean,

featureCounter: Counter

): Stitch[Boolean] = {

featureCounter.incr()

val advancedFilters = viewerId match {

case Some(id) => userSource.getAdvancedFilters(id)

case \_ => Stitch.value(AdvancedFilters())

}

advancedFilters.map(advancedFilterCheck)

}

}