package com.twitter.visibility.builder.users

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.stitch.Stitch

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.UserId

import com.twitter.visibility.common.UserSearchSafetySource

import com.twitter.visibility.features.ViewerId

import com.twitter.visibility.features.ViewerOptInBlocking

import com.twitter.visibility.features.ViewerOptInFiltering

class ViewerSearchSafetyFeatures(

userSearchSafetySource: UserSearchSafetySource,

statsReceiver: StatsReceiver) {

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

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

private[this] val viewerOptInBlockingRequests =

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

private[this] val viewerOptInFilteringRequests =

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

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

requests.incr()

builder

.withConstantFeature(ViewerId, viewerId)

.withFeature(ViewerOptInBlocking, viewerOptInBlocking(viewerId))

.withFeature(ViewerOptInFiltering, viewerOptInFiltering(viewerId))

}

def viewerOptInBlocking(viewerId: Option[UserId]): Stitch[Boolean] = {

viewerOptInBlockingRequests.incr()

viewerId match {

case Some(userId) => userSearchSafetySource.optInBlocking(userId)

case \_ => Stitch.False

}

}

def viewerOptInFiltering(viewerId: Option[UserId]): Stitch[Boolean] = {

viewerOptInFilteringRequests.incr()

viewerId match {

case Some(userId) => userSearchSafetySource.optInFiltering(userId)

case \_ => Stitch.False

}

}

}