package com.twitter.visibility.builder.users

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.gizmoduck.thriftscala.Label

import com.twitter.gizmoduck.thriftscala.Labels

import com.twitter.gizmoduck.thriftscala.Profile

import com.twitter.gizmoduck.thriftscala.Safety

import com.twitter.gizmoduck.thriftscala.User

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tseng.withholding.thriftscala.TakedownReason

import com.twitter.util.Duration

import com.twitter.util.Time

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.UserSource

import com.twitter.visibility.features.\_

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

private[this] val scopedStatsReceiver = statsReceiver.scope("author\_features")

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

private[this] val authorUserLabels =

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

private[this] val authorIsSuspended =

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

private[this] val authorIsProtected =

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

private[this] val authorIsDeactivated =

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

private[this] val authorIsErased =

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

private[this] val authorIsOffboarded =

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

private[this] val authorIsNsfwUser =

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

private[this] val authorIsNsfwAdmin =

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

private[this] val authorTakedownReasons =

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

private[this] val authorHasDefaultProfileImage =

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

private[this] val authorAccountAge =

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

private[this] val authorIsVerified =

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

private[this] val authorScreenName =

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

private[this] val authorIsBlueVerified =

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

def forAuthor(author: User): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

\_.withConstantFeature(AuthorId, Set(author.id))

.withConstantFeature(AuthorUserLabels, authorUserLabels(author))

.withConstantFeature(AuthorIsProtected, authorIsProtected(author))

.withConstantFeature(AuthorIsSuspended, authorIsSuspended(author))

.withConstantFeature(AuthorIsDeactivated, authorIsDeactivated(author))

.withConstantFeature(AuthorIsErased, authorIsErased(author))

.withConstantFeature(AuthorIsOffboarded, authorIsOffboarded(author))

.withConstantFeature(AuthorTakedownReasons, authorTakedownReasons(author))

.withConstantFeature(AuthorHasDefaultProfileImage, authorHasDefaultProfileImage(author))

.withConstantFeature(AuthorAccountAge, authorAccountAge(author))

.withConstantFeature(AuthorIsNsfwUser, authorIsNsfwUser(author))

.withConstantFeature(AuthorIsNsfwAdmin, authorIsNsfwAdmin(author))

.withConstantFeature(AuthorIsVerified, authorIsVerified(author))

.withConstantFeature(AuthorScreenName, authorScreenName(author))

.withConstantFeature(AuthorIsBlueVerified, authorIsBlueVerified(author))

}

def forAuthorNoDefaults(author: User): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

\_.withConstantFeature(AuthorId, Set(author.id))

.withConstantFeature(AuthorUserLabels, authorUserLabelsOpt(author))

.withConstantFeature(AuthorIsProtected, authorIsProtectedOpt(author))

.withConstantFeature(AuthorIsSuspended, authorIsSuspendedOpt(author))

.withConstantFeature(AuthorIsDeactivated, authorIsDeactivatedOpt(author))

.withConstantFeature(AuthorIsErased, authorIsErasedOpt(author))

.withConstantFeature(AuthorIsOffboarded, authorIsOffboarded(author))

.withConstantFeature(AuthorTakedownReasons, authorTakedownReasons(author))

.withConstantFeature(AuthorHasDefaultProfileImage, authorHasDefaultProfileImage(author))

.withConstantFeature(AuthorAccountAge, authorAccountAge(author))

.withConstantFeature(AuthorIsNsfwUser, authorIsNsfwUserOpt(author))

.withConstantFeature(AuthorIsNsfwAdmin, authorIsNsfwAdminOpt(author))

.withConstantFeature(AuthorIsVerified, authorIsVerifiedOpt(author))

.withConstantFeature(AuthorScreenName, authorScreenName(author))

.withConstantFeature(AuthorIsBlueVerified, authorIsBlueVerified(author))

}

def forAuthorId(authorId: Long): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

\_.withConstantFeature(AuthorId, Set(authorId))

.withFeature(AuthorUserLabels, authorUserLabels(authorId))

.withFeature(AuthorIsProtected, authorIsProtected(authorId))

.withFeature(AuthorIsSuspended, authorIsSuspended(authorId))

.withFeature(AuthorIsDeactivated, authorIsDeactivated(authorId))

.withFeature(AuthorIsErased, authorIsErased(authorId))

.withFeature(AuthorIsOffboarded, authorIsOffboarded(authorId))

.withFeature(AuthorTakedownReasons, authorTakedownReasons(authorId))

.withFeature(AuthorHasDefaultProfileImage, authorHasDefaultProfileImage(authorId))

.withFeature(AuthorAccountAge, authorAccountAge(authorId))

.withFeature(AuthorIsNsfwUser, authorIsNsfwUser(authorId))

.withFeature(AuthorIsNsfwAdmin, authorIsNsfwAdmin(authorId))

.withFeature(AuthorIsVerified, authorIsVerified(authorId))

.withFeature(AuthorScreenName, authorScreenName(authorId))

.withFeature(AuthorIsBlueVerified, authorIsBlueVerified(authorId))

}

def forNoAuthor(): FeatureMapBuilder => FeatureMapBuilder = {

\_.withConstantFeature(AuthorId, Set.empty[Long])

.withConstantFeature(AuthorUserLabels, Seq.empty)

.withConstantFeature(AuthorIsProtected, false)

.withConstantFeature(AuthorIsSuspended, false)

.withConstantFeature(AuthorIsDeactivated, false)

.withConstantFeature(AuthorIsErased, false)

.withConstantFeature(AuthorIsOffboarded, false)

.withConstantFeature(AuthorTakedownReasons, Seq.empty)

.withConstantFeature(AuthorHasDefaultProfileImage, false)

.withConstantFeature(AuthorAccountAge, Duration.Zero)

.withConstantFeature(AuthorIsNsfwUser, false)

.withConstantFeature(AuthorIsNsfwAdmin, false)

.withConstantFeature(AuthorIsVerified, false)

.withConstantFeature(AuthorIsBlueVerified, false)

}

def authorUserLabels(author: User): Seq[Label] =

authorUserLabels(author.labels)

def authorIsSuspended(authorId: Long): Stitch[Boolean] =

userSource.getSafety(authorId).map(safety => authorIsSuspended(Some(safety)))

def authorIsSuspendedOpt(author: User): Option[Boolean] = {

authorIsSuspended.incr()

author.safety.map(\_.suspended)

}

private def authorIsSuspended(safety: Option[Safety]): Boolean = {

authorIsSuspended.incr()

safety.exists(\_.suspended)

}

def authorIsProtected(author: User): Boolean =

authorIsProtected(author.safety)

def authorIsDeactivated(authorId: Long): Stitch[Boolean] =

userSource.getSafety(authorId).map(safety => authorIsDeactivated(Some(safety)))

def authorIsDeactivatedOpt(author: User): Option[Boolean] = {

authorIsDeactivated.incr()

author.safety.map(\_.deactivated)

}

private def authorIsDeactivated(safety: Option[Safety]): Boolean = {

authorIsDeactivated.incr()

safety.exists(\_.deactivated)

}

def authorIsErased(author: User): Boolean = {

authorIsErased.incr()

author.safety.exists(\_.erased)

}

def authorIsOffboarded(authorId: Long): Stitch[Boolean] = {

userSource.getSafety(authorId).map(safety => authorIsOffboarded(Some(safety)))

}

def authorIsNsfwUser(author: User): Boolean = {

authorIsNsfwUser(author.safety)

}

def authorIsNsfwUser(authorId: Long): Stitch[Boolean] = {

userSource.getSafety(authorId).map(safety => authorIsNsfwUser(Some(safety)))

}

def authorIsNsfwUser(safety: Option[Safety]): Boolean = {

authorIsNsfwUser.incr()

safety.exists(\_.nsfwUser)

}

def authorIsNsfwAdminOpt(author: User): Option[Boolean] = {

authorIsNsfwAdmin.incr()

author.safety.map(\_.nsfwAdmin)

}

def authorTakedownReasons(authorId: Long): Stitch[Seq[TakedownReason]] = {

authorTakedownReasons.incr()

userSource.getTakedownReasons(authorId)

}

def authorHasDefaultProfileImage(authorId: Long): Stitch[Boolean] =

userSource.getProfile(authorId).map(profile => authorHasDefaultProfileImage(Some(profile)))

def authorAccountAge(authorId: Long): Stitch[Duration] =

userSource.getCreatedAtMsec(authorId).map(authorAccountAgeFromTimestamp)

def authorIsVerified(authorId: Long): Stitch[Boolean] =

userSource.getSafety(authorId).map(safety => authorIsVerified(Some(safety)))

def authorIsVerifiedOpt(author: User): Option[Boolean] = {

authorIsVerified.incr()

author.safety.map(\_.verified)

}

private def authorIsVerified(safety: Option[Safety]): Boolean = {

authorIsVerified.incr()

safety.exists(\_.verified)

}

def authorScreenName(author: User): Option[String] = {

authorScreenName.incr()

author.profile.map(\_.screenName)

}

def authorScreenName(authorId: Long): Stitch[String] = {

authorScreenName.incr()

userSource.getProfile(authorId).map(profile => profile.screenName)

}

}