package com.twitter.follow\_recommendations.common.clients.gizmoduck

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.gizmoduck.thriftscala.LookupContext

import com.twitter.gizmoduck.thriftscala.PerspectiveEdge

import com.twitter.gizmoduck.thriftscala.QueryFields

import com.twitter.stitch.Stitch

import com.twitter.stitch.gizmoduck.Gizmoduck

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class GizmoduckClient @Inject() (gizmoduckStitchClient: Gizmoduck, statsReceiver: StatsReceiver) {

val stats = statsReceiver.scope("gizmoduck\_client")

val getByIdStats = stats.scope("get\_by\_id")

val getUserById = stats.scope("get\_user\_by\_id")

def isProtected(userId: Long): Stitch[Boolean] = {

// get latency metrics with StatsUtil.profileStitch when calling .getById

val response = StatsUtil.profileStitch(

gizmoduckStitchClient.getById(userId, Set(QueryFields.Safety)),

getByIdStats

)

response.map { result =>

result.user.flatMap(\_.safety).map(\_.isProtected).getOrElse(true)

}

}

def getUserName(userId: Long, forUserId: Long): Stitch[Option[String]] = {

val queryFields = GizmoduckClient.GetUserByIdUserNameQueryFields

val lookupContext = LookupContext(

forUserId = Some(forUserId),

perspectiveEdges = Some(GizmoduckClient.DefaultPerspectiveEdges)

)

// get latency metrics with StatsUtil.profileStitch when calling .getUserById

val response = StatsUtil.profileStitch(

gizmoduckStitchClient.getUserById(userId, queryFields, lookupContext),

getUserById

)

response.map(\_.profile.map(\_.name))

}

}

object GizmoduckClient {

// Similar to GizmoduckUserRepository.DefaultPerspectiveEdges

val DefaultPerspectiveEdges: Set[PerspectiveEdge] =

Set(

PerspectiveEdge.Blocking,

PerspectiveEdge.BlockedBy,

PerspectiveEdge.DeviceFollowing,

PerspectiveEdge.FollowRequestSent,

PerspectiveEdge.Following,

PerspectiveEdge.FollowedBy,

PerspectiveEdge.LifelineFollowing,

PerspectiveEdge.LifelineFollowedBy,

PerspectiveEdge.Muting,

PerspectiveEdge.NoRetweetsFrom

)

// From GizmoduckUserRepository.DefaultQueryFields

val GetUserByIdQueryFields: Set[QueryFields] = Set(

QueryFields.Account,

QueryFields.Counts,

QueryFields.ExtendedProfile,

QueryFields.Perspective,

QueryFields.Profile,

QueryFields.ProfileDesign,

QueryFields.ProfileLocation,

QueryFields.Safety,

QueryFields.Roles,

QueryFields.Takedowns,

QueryFields.UrlEntities,

QueryFields.DirectMessageView,

QueryFields.MediaView

)

val GetUserByIdUserNameQueryFields: Set[QueryFields] = Set(

QueryFields.Profile

)

}