package com.twitter.follow\_recommendations.controllers

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.models.ClientContextConverter

import com.twitter.follow\_recommendations.common.models.DisplayLocation

import com.twitter.follow\_recommendations.models.DebugParams

import com.twitter.follow\_recommendations.models.DisplayContext

import com.twitter.follow\_recommendations.models.RecommendationRequest

import com.twitter.follow\_recommendations.{thriftscala => t}

import com.twitter.gizmoduck.thriftscala.UserType

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class RecommendationRequestBuilder @Inject() (

requestBuilderUserFetcher: RequestBuilderUserFetcher,

statsReceiver: StatsReceiver) {

private val scopedStats = statsReceiver.scope(this.getClass.getSimpleName)

private val isSoftUserCounter = scopedStats.counter("is\_soft\_user")

def fromThrift(tRequest: t.RecommendationRequest): Stitch[RecommendationRequest] = {

requestBuilderUserFetcher.fetchUser(tRequest.clientContext.userId).map { userOpt =>

val isSoftUser = userOpt.exists(\_.userType == UserType.Soft)

if (isSoftUser) isSoftUserCounter.incr()

RecommendationRequest(

clientContext = ClientContextConverter.fromThrift(tRequest.clientContext),

displayLocation = DisplayLocation.fromThrift(tRequest.displayLocation),

displayContext = tRequest.displayContext.map(DisplayContext.fromThrift),

maxResults = tRequest.maxResults,

cursor = tRequest.cursor,

excludedIds = tRequest.excludedIds,

fetchPromotedContent = tRequest.fetchPromotedContent,

debugParams = tRequest.debugParams.map(DebugParams.fromThrift),

userLocationState = tRequest.userLocationState,

isSoftUser = isSoftUser

)

}

}

}