package com.twitter.follow\_recommendations.common.candidate\_sources.geo

import com.google.inject.Singleton

import com.twitter.core\_workflows.user\_model.thriftscala.UserState

import com.twitter.finagle.stats.Counter

import com.twitter.finagle.stats.StatsReceiver

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

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

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

import com.twitter.hermit.model.Algorithm

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSource

import com.twitter.product\_mixer.core.model.common.identifier.CandidateSourceIdentifier

import com.twitter.product\_mixer.core.model.marshalling.request.HasClientContext

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.HasParams

import javax.inject.Inject

@Singleton

class PopCountrySource @Inject() (

popGeoSource: PopGeoSource,

statsReceiver: StatsReceiver)

extends CandidateSource[

HasClientContext with HasParams with HasUserState with HasGeohashAndCountryCode,

CandidateUser

] {

override val identifier: CandidateSourceIdentifier = PopCountrySource.Identifier

val stats: StatsReceiver = statsReceiver.scope("PopCountrySource")

// counter to check if we found a country code value in the request

val foundCountryCodeCounter: Counter = stats.counter("found\_country\_code\_value")

// counter to check if we are missing a country code value in the request

val missingCountryCodeCounter: Counter = stats.counter("missing\_country\_code\_value")

override def apply(

target: HasClientContext with HasParams with HasUserState with HasGeohashAndCountryCode

): Stitch[Seq[CandidateUser]] = {

target.geohashAndCountryCode

.flatMap(\_.countryCode).map { countryCode =>

foundCountryCodeCounter.incr()

if (target.userState.exists(PopCountrySource.BlacklistedTargetUserStates.contains)) {

Stitch.Nil

} else {

popGeoSource("country\_" + countryCode)

.map(\_.take(PopCountrySource.MaxResults).map(\_.withCandidateSource(identifier)))

}

}.getOrElse {

missingCountryCodeCounter.incr()

Stitch.Nil

}

}

}

object PopCountrySource {

val Identifier: CandidateSourceIdentifier = CandidateSourceIdentifier(

Algorithm.PopCountry.toString)

val MaxResults = 40

val BlacklistedTargetUserStates: Set[UserState] = Set(

UserState.HeavyTweeter,

UserState.HeavyNonTweeter,

UserState.MediumTweeter,

UserState.MediumNonTweeter)

}