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

import com.google.inject.Singleton

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.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 BasePopGeohashSource @Inject() (

popGeoSource: CandidateSource[String, CandidateUser],

statsReceiver: StatsReceiver)

extends CandidateSource[

HasParams with HasClientContext with HasGeohashAndCountryCode,

CandidateUser

]

with BasePopGeohashSourceConfig {

val stats: StatsReceiver = statsReceiver

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

val foundGeohashCounter: Counter = stats.counter("found\_geohash\_value")

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

val missingGeohashCounter: Counter = stats.counter("missing\_geohash\_value")

/\*\* @see [[CandidateSourceIdentifier]] \*/

override val identifier: CandidateSourceIdentifier = CandidateSourceIdentifier(

"BasePopGeohashSource")

override def apply(

target: HasParams with HasClientContext with HasGeohashAndCountryCode

): Stitch[Seq[CandidateUser]] = {

if (!candidateSourceEnabled(target)) {

return Stitch.Nil

}

target.geohashAndCountryCode

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

foundGeohashCounter.incr()

val keys = (minGeohashLength(target) to math.min(maxGeohashLength(target), geohash.length))

.map("geohash\_" + geohash.take(\_)).reverse

if (returnResultFromAllPrecision(target)) {

Stitch

.collect(keys.map(popGeoSource.apply)).map(

\_.flatten.map(\_.withCandidateSource(identifier))

)

} else {

Stitch

.collect(keys.map(popGeoSource.apply)).map(

\_.find(\_.nonEmpty)

.getOrElse(Nil)

.take(maxResults(target)).map(\_.withCandidateSource(identifier))

)

}

}.getOrElse {

missingGeohashCounter.incr()

Stitch.Nil

}

}

}

trait BasePopGeohashSourceConfig {

type Target = HasParams with HasClientContext

def maxResults(target: Target): Int = 200

def minGeohashLength(target: Target): Int = 2

def maxGeohashLength(target: Target): Int = 4

def returnResultFromAllPrecision(target: Target): Boolean = false

def candidateSourceEnabled(target: Target): Boolean = false

}