package com.twitter.tweetypie

package handler

import com.twitter.geoduck.backend.hydration.thriftscala.HydrationContext

import com.twitter.geoduck.common.thriftscala.Constants

import com.twitter.geoduck.common.thriftscala.PlaceQuery

import com.twitter.geoduck.common.thriftscala.PlaceQueryFields

import com.twitter.geoduck.service.common.clientmodules.GeoduckGeohashLocate

import com.twitter.geoduck.service.thriftscala.LocationResponse

import com.twitter.geoduck.util.primitives.LatLon

import com.twitter.geoduck.util.primitives.{Geohash => GDGeohash}

import com.twitter.geoduck.util.primitives.{Place => GDPlace}

import com.twitter.servo.util.FutureArrow

import com.twitter.tweetypie.repository.GeoduckPlaceConverter

import com.twitter.tweetypie.{thriftscala => TP}

object ReverseGeocoder {

val log: Logger = Logger(getClass)

private def validatingRGC(rgc: ReverseGeocoder): ReverseGeocoder =

FutureArrow {

case (coords: TP.GeoCoordinates, language: PlaceLanguage) =>

if (LatLon.isValid(coords.latitude, coords.longitude))

rgc((coords, language))

else

Future.None

}

/\*\*

\* create a Geo backed ReverseGeocoder

\*/

def fromGeoduck(geohashLocate: GeoduckGeohashLocate): ReverseGeocoder =

validatingRGC(

FutureArrow {

case (geo: TP.GeoCoordinates, language: PlaceLanguage) =>

if (log.isDebugEnabled) {

log.debug("RGC'ing " + geo.toString() + " with geoduck")

}

val hydrationContext =

HydrationContext(

placeFields = Set[PlaceQueryFields](

PlaceQueryFields.PlaceNames

)

)

val gh = GDGeohash(LatLon(lat = geo.latitude, lon = geo.longitude))

val placeQuery = PlaceQuery(placeTypes = Some(Constants.ConsumerPlaceTypes))

geohashLocate

.locateGeohashes(Seq(gh.toThrift), placeQuery, hydrationContext)

.onFailure { case ex => log.warn("failed to rgc " + geo.toString(), ex) }

.map {

(resp: Seq[Try[LocationResponse]]) =>

resp.headOption.flatMap {

case Throw(ex) =>

log.warn("rgc failed for coords: " + geo.toString(), ex)

None

case Return(locationResponse) =>

GDPlace.tryLocationResponse(locationResponse) match {

case Throw(ex) =>

log

.warn("rgc failed in response handling for coords: " + geo.toString(), ex)

None

case Return(tplaces) =>

GDPlace.pickConsumerLocation(tplaces).map { place: GDPlace =>

if (log.isDebugEnabled) {

log.debug("successfully rgc'd " + geo + " to " + place.id)

}

GeoduckPlaceConverter(language, place)

}

}

}

}

}

)

}