package com.twitter.tweetypie

package repository

import com.twitter.geoduck.common.{thriftscala => Geoduck}

import com.twitter.geoduck.service.thriftscala.GeoContext

import com.twitter.geoduck.service.thriftscala.Key

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

import com.twitter.geoduck.util.service.GeoduckLocate

import com.twitter.geoduck.util.service.LocationResponseExtractors

import com.twitter.geoduck.util.{primitives => GDPrimitive}

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

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

object GeoduckPlaceConverter {

def LocationResponseToTPPlace(lang: String, lr: LocationResponse): Option[TP.Place] =

GDPrimitive.Place

.fromLocationResponse(lr)

.headOption

.map(apply(lang, \_))

def convertPlaceType(pt: Geoduck.PlaceType): TP.PlaceType = pt match {

case Geoduck.PlaceType.Unknown => TP.PlaceType.Unknown

case Geoduck.PlaceType.Country => TP.PlaceType.Country

case Geoduck.PlaceType.Admin => TP.PlaceType.Admin

case Geoduck.PlaceType.City => TP.PlaceType.City

case Geoduck.PlaceType.Neighborhood => TP.PlaceType.Neighborhood

case Geoduck.PlaceType.Poi => TP.PlaceType.Poi

case Geoduck.PlaceType.ZipCode => TP.PlaceType.Admin

case Geoduck.PlaceType.Metro => TP.PlaceType.Admin

case Geoduck.PlaceType.Admin0 => TP.PlaceType.Admin

case Geoduck.PlaceType.Admin1 => TP.PlaceType.Admin

case \_ =>

throw new IllegalStateException(s"Invalid place type: $pt")

}

def convertPlaceName(gd: Geoduck.PlaceName): TP.PlaceName =

TP.PlaceName(

name = gd.name,

language = gd.language.getOrElse("en"),

`type` = convertPlaceNameType(gd.nameType),

preferred = gd.preferred

)

def convertPlaceNameType(pt: Geoduck.PlaceNameType): TP.PlaceNameType = pt match {

case Geoduck.PlaceNameType.Normal => TP.PlaceNameType.Normal

case Geoduck.PlaceNameType.Abbreviation => TP.PlaceNameType.Abbreviation

case Geoduck.PlaceNameType.Synonym => TP.PlaceNameType.Synonym

case \_ =>

throw new IllegalStateException(s"Invalid place name type: $pt")

}

def convertAttributes(attrs: collection.Set[Geoduck.PlaceAttribute]): Map[String, String] =

attrs.map(attr => attr.key -> attr.value.getOrElse("")).toMap

def convertBoundingBox(geom: GDPrimitive.Geometry): Seq[TP.GeoCoordinates] =

geom.coordinates.map { coord =>

TP.GeoCoordinates(

latitude = coord.lat,

longitude = coord.lon

)

}

def apply(queryLang: String, geoplace: GDPrimitive.Place): TP.Place = {

val bestname = geoplace.bestName(queryLang).getOrElse(geoplace.hexId)

TP.Place(

id = geoplace.hexId,

`type` = convertPlaceType(geoplace.placeType),

name = bestname,

fullName = geoplace.fullName(queryLang).getOrElse(bestname),

attributes = convertAttributes(geoplace.attributes),

boundingBox = geoplace.boundingBox.map(convertBoundingBox),

countryCode = geoplace.countryCode,

containers = Some(geoplace.cone.map(\_.hexId).toSet + geoplace.hexId),

countryName = geoplace.countryName(queryLang)

)

}

def convertGDKey(key: Key, lang: String): PlaceKey = {

val Key.PlaceId(pid) = key

PlaceKey("%016x".format(pid), lang)

}

}

object GeoduckPlaceRepository {

val context: GeoContext =

GeoContext(

placeFields = Set(

Geoduck.PlaceQueryFields.Attributes,

Geoduck.PlaceQueryFields.BoundingBox,

Geoduck.PlaceQueryFields.PlaceNames,

Geoduck.PlaceQueryFields.Cone

),

placeTypes = Set(

Geoduck.PlaceType.Country,

Geoduck.PlaceType.Admin0,

Geoduck.PlaceType.Admin1,

Geoduck.PlaceType.City,

Geoduck.PlaceType.Neighborhood

),

includeCountryCode = true,

hydrateCone = true

)

def apply(geoduck: GeoduckLocate): PlaceRepository.Type = {

val geoduckGroup = LegacySeqGroup((ids: Seq[Key.PlaceId]) => geoduck(context, ids))

placeKey =>

val placeId =

try {

Stitch.value(

Key.PlaceId(java.lang.Long.parseUnsignedLong(placeKey.placeId, 16))

)

} catch {

case \_: NumberFormatException => Stitch.exception(NotFound)

}

placeId

.flatMap(id => Stitch.call(id, geoduckGroup))

.rescue { case LocationResponseExtractors.Failure(ex) => Stitch.exception(ex) }

.map { resp =>

GDPrimitive.Place

.fromLocationResponse(resp)

.headOption

.map(GeoduckPlaceConverter(placeKey.language, \_))

}

.lowerFromOption()

}

}