package com.twitter.tweetypie

package repository

import com.twitter.dataproducts.enrichments.thriftscala.\_

import com.twitter.gizmoduck.thriftscala.UserResponseState.\_

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

import com.twitter.tweetypie.backends.GnipEnricherator

import com.twitter.tweetypie.thriftscala.GeoCoordinates

case class ProfileGeoKey(tweetId: TweetId, userId: Option[UserId], coords: Option[GeoCoordinates]) {

def key: TweetData =

TweetData(

tweetId = tweetId,

userId = userId,

coordinates = coords.map(ProfileGeoRepository.convertGeo)

)

}

object ProfileGeoRepository {

type Type = ProfileGeoKey => Stitch[ProfileGeoEnrichment]

case class UnexpectedState(state: EnrichmentHydrationState) extends Exception(state.name)

def convertGeo(coords: GeoCoordinates): TweetyPieGeoCoordinates =

TweetyPieGeoCoordinates(

latitude = coords.latitude,

longitude = coords.longitude,

geoPrecision = coords.geoPrecision,

display = coords.display

)

def apply(hydrateProfileGeo: GnipEnricherator.HydrateProfileGeo): Type = {

import EnrichmentHydrationState.\_

val emptyEnrichmentStitch = Stitch.value(ProfileGeoEnrichment())

val profileGeoGroup = SeqGroup[TweetData, ProfileGeoResponse] { keys: Seq[TweetData] =>

// Gnip ignores writePath and treats all requests as reads

LegacySeqGroup.liftToSeqTry(

hydrateProfileGeo(ProfileGeoRequest(requests = keys, writePath = false))

)

}

(geoKey: ProfileGeoKey) =>

Stitch

.call(geoKey.key, profileGeoGroup)

.flatMap {

case ProfileGeoResponse(\_, Success, Some(enrichment), \_) =>

Stitch.value(enrichment)

case ProfileGeoResponse(\_, Success, None, \_) =>

// when state is Success enrichment should always be Some, but default to be safe

emptyEnrichmentStitch

case ProfileGeoResponse(

\_,

UserLookupError,

\_,

Some(DeactivatedUser | SuspendedUser | NotFound)

) =>

emptyEnrichmentStitch

case r =>

Stitch.exception(UnexpectedState(r.state))

}

}

}