package com.twitter

import com.twitter.mediaservices.commons.thriftscala.MediaKey

import com.twitter.snowflake.id.SnowflakeId

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.gizmoduck.thriftscala.QueryFields

package object tweetypie {

// common imports that many classes need, will probably expand this list in the future.

type Logger = com.twitter.util.logging.Logger

val Logger: com.twitter.util.logging.Logger.type = com.twitter.util.logging.Logger

type StatsReceiver = com.twitter.finagle.stats.StatsReceiver

val TweetLenses: com.twitter.tweetypie.util.TweetLenses.type =

com.twitter.tweetypie.util.TweetLenses

type Future[A] = com.twitter.util.Future[A]

val Future: com.twitter.util.Future.type = com.twitter.util.Future

type Duration = com.twitter.util.Duration

val Duration: com.twitter.util.Duration.type = com.twitter.util.Duration

type Time = com.twitter.util.Time

val Time: com.twitter.util.Time.type = com.twitter.util.Time

type Try[A] = com.twitter.util.Try[A]

val Try: com.twitter.util.Try.type = com.twitter.util.Try

type Throw[A] = com.twitter.util.Throw[A]

val Throw: com.twitter.util.Throw.type = com.twitter.util.Throw

type Return[A] = com.twitter.util.Return[A]

val Return: com.twitter.util.Return.type = com.twitter.util.Return

type Gate[T] = com.twitter.servo.util.Gate[T]

val Gate: com.twitter.servo.util.Gate.type = com.twitter.servo.util.Gate

type Effect[A] = com.twitter.servo.util.Effect[A]

val Effect: com.twitter.servo.util.Effect.type = com.twitter.servo.util.Effect

type FutureArrow[A, B] = com.twitter.servo.util.FutureArrow[A, B]

val FutureArrow: com.twitter.servo.util.FutureArrow.type = com.twitter.servo.util.FutureArrow

type FutureEffect[A] = com.twitter.servo.util.FutureEffect[A]

val FutureEffect: com.twitter.servo.util.FutureEffect.type = com.twitter.servo.util.FutureEffect

type Lens[A, B] = com.twitter.servo.data.Lens[A, B]

val Lens: com.twitter.servo.data.Lens.type = com.twitter.servo.data.Lens

type Mutation[A] = com.twitter.servo.data.Mutation[A]

val Mutation: com.twitter.servo.data.Mutation.type = com.twitter.servo.data.Mutation

type User = com.twitter.gizmoduck.thriftscala.User

val User: com.twitter.gizmoduck.thriftscala.User.type = com.twitter.gizmoduck.thriftscala.User

type Safety = com.twitter.gizmoduck.thriftscala.Safety

val Safety: com.twitter.gizmoduck.thriftscala.Safety.type =

com.twitter.gizmoduck.thriftscala.Safety

type UserField = com.twitter.gizmoduck.thriftscala.QueryFields

val UserField: QueryFields.type = com.twitter.gizmoduck.thriftscala.QueryFields

type Tweet = thriftscala.Tweet

val Tweet: com.twitter.tweetypie.thriftscala.Tweet.type = thriftscala.Tweet

type ThriftTweetService = TweetServiceInternal.MethodPerEndpoint

type TweetId = Long

type UserId = Long

type MediaId = Long

type AppId = Long

type KnownDeviceToken = String

type ConversationId = Long

type CommunityId = Long

type PlaceId = String

type FieldId = Short

type Count = Long

type CountryCode = String // ISO 3166-1-alpha-2

type CreativesContainerId = Long

def hasGeo(tweet: Tweet): Boolean =

TweetLenses.placeId.get(tweet).nonEmpty ||

TweetLenses.geoCoordinates.get(tweet).nonEmpty

def getUserId(tweet: Tweet): UserId = TweetLenses.userId.get(tweet)

def getText(tweet: Tweet): String = TweetLenses.text.get(tweet)

def getCreatedAt(tweet: Tweet): Long = TweetLenses.createdAt.get(tweet)

def getCreatedVia(tweet: Tweet): String = TweetLenses.createdVia.get(tweet)

def getReply(tweet: Tweet): Option[Reply] = TweetLenses.reply.get(tweet)

def getDirectedAtUser(tweet: Tweet): Option[DirectedAtUser] =

TweetLenses.directedAtUser.get(tweet)

def getShare(tweet: Tweet): Option[Share] = TweetLenses.share.get(tweet)

def getQuotedTweet(tweet: Tweet): Option[QuotedTweet] = TweetLenses.quotedTweet.get(tweet)

def getUrls(tweet: Tweet): Seq[UrlEntity] = TweetLenses.urls.get(tweet)

def getMedia(tweet: Tweet): Seq[MediaEntity] = TweetLenses.media.get(tweet)

def getMediaKeys(tweet: Tweet): Seq[MediaKey] = TweetLenses.mediaKeys.get(tweet)

def getMentions(tweet: Tweet): Seq[MentionEntity] = TweetLenses.mentions.get(tweet)

def getCashtags(tweet: Tweet): Seq[CashtagEntity] = TweetLenses.cashtags.get(tweet)

def getHashtags(tweet: Tweet): Seq[HashtagEntity] = TweetLenses.hashtags.get(tweet)

def getMediaTagMap(tweet: Tweet): Map[MediaId, Seq[MediaTag]] = TweetLenses.mediaTagMap.get(tweet)

def isRetweet(tweet: Tweet): Boolean = tweet.coreData.flatMap(\_.share).nonEmpty

def isSelfReply(authorUserId: UserId, r: Reply): Boolean =

r.inReplyToStatusId.isDefined && (r.inReplyToUserId == authorUserId)

def isSelfReply(tweet: Tweet): Boolean = {

getReply(tweet).exists { r => isSelfReply(getUserId(tweet), r) }

}

def getConversationId(tweet: Tweet): Option[TweetId] = TweetLenses.conversationId.get(tweet)

def getSelfThreadMetadata(tweet: Tweet): Option[SelfThreadMetadata] =

TweetLenses.selfThreadMetadata.get(tweet)

def getCardReference(tweet: Tweet): Option[CardReference] = TweetLenses.cardReference.get(tweet)

def getEscherbirdAnnotations(tweet: Tweet): Option[EscherbirdEntityAnnotations] =

TweetLenses.escherbirdEntityAnnotations.get(tweet)

def getCommunities(tweet: Tweet): Option[Communities] = TweetLenses.communities.get(tweet)

def getTimestamp(tweet: Tweet): Time =

if (SnowflakeId.isSnowflakeId(tweet.id)) SnowflakeId(tweet.id).time

else Time.fromSeconds(getCreatedAt(tweet).toInt)

}