package com.twitter.tweetypie

package hydrator

import com.twitter.tweetypie

import com.twitter.tweetypie.core.TweetData

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.\_

import org.apache.thrift.protocol.TField

/\*\*

\* Encapsulates basic, immutable details about a tweet to be hydrated, along with the

\* `TweetQuery.Options`. Only tweet data that are not affected by hydration should be

\* exposed here, as a single `TweetCtx` instance should be usable for the entire hydration

\* of a tweet.

\*/

trait TweetCtx {

def opts: TweetQuery.Options

def tweetId: TweetId

def userId: UserId

def text: String

def createdAt: Time

def createdVia: String

def isRetweet: Boolean

def isReply: Boolean

def isSelfReply: Boolean

def sourceUserId: Option[UserId]

def sourceTweetId: Option[TweetId]

def inReplyToTweetId: Option[TweetId]

def geoCoordinates: Option[GeoCoordinates]

def placeId: Option[String]

def hasTakedown: Boolean

def quotedTweet: Option[QuotedTweet]

def completedHydrations: Set[HydrationType]

def isInitialInsert: Boolean = opts.cause.initialInsert(tweetId)

def tweetFieldRequested(field: TField): Boolean = tweetFieldRequested(field.id)

def tweetFieldRequested(fieldId: FieldId): Boolean = opts.include.tweetFields.contains(fieldId)

def mediaFieldRequested(field: TField): Boolean = mediaFieldRequested(field.id)

def mediaFieldRequested(fieldId: FieldId): Boolean = opts.include.mediaFields.contains(fieldId)

}

object TweetCtx {

def from(td: TweetData, opts: TweetQuery.Options): TweetCtx = FromTweetData(td, opts)

trait Proxy extends TweetCtx {

protected def underlyingTweetCtx: TweetCtx

def opts: TweetQuery.Options = underlyingTweetCtx.opts

def tweetId: TweetId = underlyingTweetCtx.tweetId

def userId: UserId = underlyingTweetCtx.userId

def text: String = underlyingTweetCtx.text

def createdAt: Time = underlyingTweetCtx.createdAt

def createdVia: String = underlyingTweetCtx.createdVia

def isRetweet: Boolean = underlyingTweetCtx.isRetweet

def isReply: Boolean = underlyingTweetCtx.isReply

def isSelfReply: Boolean = underlyingTweetCtx.isSelfReply

def sourceUserId: Option[UserId] = underlyingTweetCtx.sourceUserId

def sourceTweetId: Option[TweetId] = underlyingTweetCtx.sourceTweetId

def inReplyToTweetId: Option[TweetId] = underlyingTweetCtx.inReplyToTweetId

def geoCoordinates: Option[GeoCoordinates] = underlyingTweetCtx.geoCoordinates

def placeId: Option[String] = underlyingTweetCtx.placeId

def hasTakedown: Boolean = underlyingTweetCtx.hasTakedown

def completedHydrations: Set[HydrationType] = underlyingTweetCtx.completedHydrations

def quotedTweet: Option[QuotedTweet] = underlyingTweetCtx.quotedTweet

}

private case class FromTweetData(td: TweetData, opts: TweetQuery.Options) extends TweetCtx {

private val tweet = td.tweet

def tweetId: MediaId = tweet.id

def userId: UserId = getUserId(tweet)

def text: String = getText(tweet)

def createdAt: Time = getTimestamp(tweet)

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

def isRetweet: Boolean = getShare(tweet).isDefined

def isSelfReply: Boolean = tweetypie.isSelfReply(tweet)

def isReply: Boolean = getReply(tweet).isDefined

def sourceUserId: Option[MediaId] = getShare(tweet).map(\_.sourceUserId)

def sourceTweetId: Option[MediaId] = getShare(tweet).map(\_.sourceStatusId)

def inReplyToTweetId: Option[MediaId] = getReply(tweet).flatMap(\_.inReplyToStatusId)

def geoCoordinates: Option[GeoCoordinates] = TweetLenses.geoCoordinates.get(tweet)

def placeId: Option[String] = TweetLenses.placeId.get(tweet)

def hasTakedown: Boolean = TweetLenses.hasTakedown(tweet)

def completedHydrations: Set[HydrationType] = td.completedHydrations

def quotedTweet: Option[QuotedTweet] = getQuotedTweet(tweet)

}

}