package com.twitter.tweetypie.storage

import com.twitter.tweetypie.additionalfields.AdditionalFields

import com.twitter.tweetypie.storage\_internal.thriftscala.StoredTweet

import com.twitter.tweetypie.thriftscala.{Tweet => TpTweet}

/\*\*

\* A field of the stored version of a tweet to read, update, or delete.

\*

\* There is not a one-to-one correspondence between the fields ids of

\* [[com.twitter.tweetypie.thriftscala.Tweet]] and

\* [[com.twitter.tweetypie.storage\_internal.thriftscala.StoredTweet]]. For example, in StoredTweet,

\* the nsfwUser property is field 11; in Tweet, it is a property of the coreData struct in field 2.

\* To circumvent the confusion of using one set of field ids or the other, callers use instances of

\* [[Field]] to reference the part of the object to modify.

\*/

class Field private[storage] (val id: Short) extends AnyVal {

override def toString: String = id.toString

}

/\*\*

\* NOTE: Make sure `AllUpdatableCompiledFields` is kept up to date when adding any new field

\*/

object Field {

import AdditionalFields.isAdditionalFieldId

val Geo: Field = new Field(StoredTweet.GeoField.id)

val HasTakedown: Field = new Field(StoredTweet.HasTakedownField.id)

val NsfwUser: Field = new Field(StoredTweet.NsfwUserField.id)

val NsfwAdmin: Field = new Field(StoredTweet.NsfwAdminField.id)

val TweetypieOnlyTakedownCountryCodes: Field =

new Field(TpTweet.TweetypieOnlyTakedownCountryCodesField.id)

val TweetypieOnlyTakedownReasons: Field =

new Field(TpTweet.TweetypieOnlyTakedownReasonsField.id)

val AllUpdatableCompiledFields: Set[Field] = Set(Geo, HasTakedown, NsfwUser, NsfwAdmin)

def additionalField(id: Short): Field = {

require(isAdditionalFieldId(id), "field id must be in the additional field range")

new Field(id)

}

}