package com.twitter.tweetypie.core

import com.twitter.servo.util.ExceptionCategorizer

import com.twitter.spam.rtf.thriftscala.FilteredReason

import scala.util.control.NoStackTrace

sealed trait FilteredState

object FilteredState {

/\*\*

\* The tweet exists and the filtered state was due to business rules

\* (e.g. safety label filtering, or protected accounts). Note that

\* Suppress and Unavailable can both have a FilteredReason.

\*/

sealed trait HasFilteredReason extends FilteredState {

def filteredReason: FilteredReason

}

/\*\*

\* The only FilteredState that is not an exception. It indicates that

\* the tweet should be returned along with a suppress reason. This is

\* sometimes known as "soft filtering". Only used by VF.

\*/

case class Suppress(filteredReason: FilteredReason) extends FilteredState with HasFilteredReason

/\*\*

\* FilteredStates that cause the tweet to be unavailable are modeled

\* as an [[Exception]]. (Suppressed filtered states cannot be used as

\* exceptions because they should not prevent the tweet from being

\* returned.) This is sometimes known as "hard filtering".

\*/

sealed abstract class Unavailable extends Exception with FilteredState with NoStackTrace

object Unavailable {

// Used for Tweets that should be dropped because of VF rules

case class Drop(filteredReason: FilteredReason) extends Unavailable with HasFilteredReason

// Used for Tweets that should be dropped and replaced with their preview because of VF rules

case class Preview(filteredReason: FilteredReason) extends Unavailable with HasFilteredReason

// Used for Tweets that should be dropped because of Tweetypie business logic

case object DropUnspecified extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.UnspecifiedReason(true)

}

// Represents a Deleted tweet (NotFound is represented with stitch.NotFound)

case object TweetDeleted extends Unavailable

// Represents a Deleted tweet that violated Twitter Rules (see go/bounced-tweet)

case object BounceDeleted extends Unavailable

// Represents both Deleted and NotFound source tweets

case class SourceTweetNotFound(deleted: Boolean) extends Unavailable

// Used by the [[ReportedTweetFilter]] to signal that a Tweet has a "reported" perspective from TLS

case object Reported extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.ReportedTweet(true)

}

// The following objects are used by the [[UserRepository]] to signal problems with the Tweet author

object Author {

case object NotFound extends Unavailable

case object Deactivated extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.AuthorIsDeactivated(true)

}

case object Offboarded extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.AuthorAccountIsInactive(true)

}

case object Suspended extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.AuthorIsSuspended(true)

}

case object Protected extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.AuthorIsProtected(true)

}

case object Unsafe extends Unavailable with HasFilteredReason {

val filteredReason: FilteredReason = FilteredReason.AuthorIsUnsafe(true)

}

}

}

/\*\*

\* Creates a new ExceptionCategorizer which returns an empty category for any

\* Unavailable value, and forwards to `underlying` for anything else.

\*/

def ignoringCategorizer(underlying: ExceptionCategorizer): ExceptionCategorizer =

ExceptionCategorizer {

case \_: Unavailable => Set.empty

case t => underlying(t)

}

}