package com.twitter.visibility.interfaces.tweets

import com.twitter.decider.Decider

import com.twitter.stitch.Stitch

import com.twitter.visibility.VisibilityLibrary

import com.twitter.visibility.builder.VisibilityResult

import com.twitter.visibility.features.TweetDeleteReason

import com.twitter.visibility.features.TweetIsInnerQuotedTweet

import com.twitter.visibility.features.TweetIsRetweet

import com.twitter.visibility.generators.TombstoneGenerator

import com.twitter.visibility.models.ContentId.DeleteTweetId

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.models.TweetDeleteReason.TweetDeleteReason

import com.twitter.visibility.models.ViewerContext

object DeletedTweetVisibilityLibrary {

type Type = DeletedTweetVisibilityLibrary.Request => Stitch[VisibilityResult]

case class Request(

tweetId: Long,

safetyLevel: SafetyLevel,

viewerContext: ViewerContext,

tweetDeleteReason: TweetDeleteReason,

isRetweet: Boolean,

isInnerQuotedTweet: Boolean,

)

def apply(

visibilityLibrary: VisibilityLibrary,

decider: Decider,

tombstoneGenerator: TombstoneGenerator,

): Type = {

val vfEngineCounter = visibilityLibrary.statsReceiver.counter("vf\_engine\_requests")

(request: Request) => {

vfEngineCounter.incr()

val contentId = DeleteTweetId(request.tweetId)

val language = request.viewerContext.requestLanguageCode.getOrElse("en")

val featureMap =

visibilityLibrary.featureMapBuilder(

Seq(

\_.withConstantFeature(TweetIsInnerQuotedTweet, request.isInnerQuotedTweet),

\_.withConstantFeature(TweetIsRetweet, request.isRetweet),

\_.withConstantFeature(TweetDeleteReason, request.tweetDeleteReason)

)

)

visibilityLibrary

.runRuleEngine(

contentId,

featureMap,

request.viewerContext,

request.safetyLevel

)

.map(tombstoneGenerator(\_, language))

}

}

}