package com.twitter.visibility.interfaces.des

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.visibility.VisibilityLibrary

import com.twitter.visibility.builder.VisibilityResult

import com.twitter.visibility.builder.tweets.StratoTweetLabelMaps

import com.twitter.visibility.builder.tweets.TweetFeatures

import com.twitter.visibility.common.SafetyLabelMapSource

import com.twitter.visibility.features.AuthorId

import com.twitter.visibility.features.FeatureMap

import com.twitter.visibility.interfaces.common.tweets.SafetyLabelMapFetcherType

import com.twitter.visibility.models.ContentId.TweetId

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.models.ViewerContext

case class DESVisibilityRequest(

tweet: Tweet,

visibilitySurface: SafetyLevel,

viewerContext: ViewerContext)

object DESVisibilityLibrary {

type Type = DESVisibilityRequest => Stitch[VisibilityResult]

def apply(

visibilityLibrary: VisibilityLibrary,

getLabelMap: SafetyLabelMapFetcherType,

enableShimFeatureHydration: Any => Boolean = \_ => false

): Type = {

val libraryStatsReceiver = visibilityLibrary.statsReceiver

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

val tweetLabelMap = new StratoTweetLabelMaps(

SafetyLabelMapSource.fromSafetyLabelMapFetcher(getLabelMap))

val tweetFeatures = new TweetFeatures(tweetLabelMap, libraryStatsReceiver)

{ request: DESVisibilityRequest =>

vfEngineCounter.incr()

val contentId = TweetId(request.tweet.id)

val authorId = coreData.userId

val featureMap =

visibilityLibrary.featureMapBuilder(

Seq(

tweetFeatures.forTweet(request.tweet),

\_.withConstantFeature(AuthorId, Set(authorId))

)

)

val isShimFeatureHydrationEnabled = enableShimFeatureHydration()

if (isShimFeatureHydrationEnabled) {

FeatureMap.resolve(featureMap, libraryStatsReceiver).flatMap { resolvedFeatureMap =>

visibilityLibrary.runRuleEngine(

contentId,

resolvedFeatureMap,

request.viewerContext,

request.visibilitySurface

)

}

} else {

visibilityLibrary.runRuleEngine(

contentId,

featureMap,

request.viewerContext,

request.visibilitySurface

)

}

}

}

}