package com.twitter.visibility.interfaces.push\_service

import com.twitter.visibility.builder.VisibilityResult

import com.twitter.visibility.rules.Action

import com.twitter.visibility.rules.Allow

import com.twitter.visibility.rules.Drop

import com.twitter.visibility.rules.Rule

import com.twitter.visibility.rules.RuleResult

case class PushServiceVisibilityResponse(

tweetVisibilityResult: VisibilityResult,

authorVisibilityResult: VisibilityResult,

sourceTweetVisibilityResult: Option[VisibilityResult] = None,

quotedTweetVisibilityResult: Option[VisibilityResult] = None,

) {

def allVisibilityResults: List[VisibilityResult] = {

List(

Some(tweetVisibilityResult),

Some(authorVisibilityResult),

sourceTweetVisibilityResult,

quotedTweetVisibilityResult,

).collect { case Some(result) => result }

}

val shouldAllow: Boolean = !allVisibilityResults.exists(isDrop(\_))

def isDrop(response: VisibilityResult): Boolean = response.verdict match {

case \_: Drop => true

case Allow => false

case \_ => false

}

def isDrop(response: Option[VisibilityResult]): Boolean = response.map(isDrop(\_)).getOrElse(false)

def getDropRules(visibilityResult: VisibilityResult): List[Rule] = {

val ruleResultMap = visibilityResult.ruleResultMap

val ruleResults = ruleResultMap.toList

val denyRules = ruleResults.collect { case (rule, RuleResult(Drop(\_, \_), \_)) => rule }

denyRules

}

def getAuthorDropRules: List[Rule] = getDropRules(authorVisibilityResult)

def getTweetDropRules: List[Rule] = getDropRules(tweetVisibilityResult)

def getDropRules: List[Rule] = getAuthorDropRules ++ getTweetDropRules

def getVerdict: Action = {

if (isDrop(authorVisibilityResult)) authorVisibilityResult.verdict

else tweetVisibilityResult.verdict

}

def missingFeatures: Map[String, Int] = PushServiceVisibilityLibraryUtil.getMissingFeatureCounts(

Seq(tweetVisibilityResult, authorVisibilityResult))

}