package com.twitter.visibility.models

import com.twitter.spam.rtf.thriftscala.SafetyResultReason

import java.util.regex.Pattern

sealed trait LabelSource {

val name: String

}

object LabelSource {

val BotRulePrefix = "bot\_id\_"

val AbusePrefix = "Abuse"

val HSEPrefix = "hse"

val AgentSourceNames = Set(

SafetyResultReason.OneOff.name,

SafetyResultReason.VotingMisinformation.name,

SafetyResultReason.HackedMaterials.name,

SafetyResultReason.Scams.name,

SafetyResultReason.PlatformManipulation.name

)

val Regex = "\\|"

val pattern: Pattern = Pattern.compile(Regex)

def fromString(name: String): Option[LabelSource] = Some(name) collect {

case \_ if name.startsWith(BotRulePrefix) =>

BotMakerRule(name.substring(BotRulePrefix.length).toLong)

case \_ if name == "A" || name == "B" || name == "AB" =>

SmyteSource(name)

case \_ if name.startsWith(AbusePrefix) =>

AbuseSource(name)

case \_ if name.startsWith(HSEPrefix) =>

HSESource(name)

case \_ if AgentSourceNames.contains(name) =>

AgentSource(name)

case \_ =>

StringSource(name)

}

def parseStringSource(source: String): (String, Option[String]) = {

pattern.split(source, 2) match {

case Array(copy, "") => (copy, None)

case Array(copy, link) => (copy, Some(link))

case Array(copy) => (copy, None)

}

}

case class BotMakerRule(ruleId: Long) extends LabelSource {

override lazy val name: String = s"${BotRulePrefix}${ruleId}"

}

case class SmyteSource(name: String) extends LabelSource

case class AbuseSource(name: String) extends LabelSource

case class AgentSource(name: String) extends LabelSource

case class HSESource(name: String) extends LabelSource

case class StringSource(name: String) extends LabelSource

}