package com.twitter.visibility.rules.generators

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.models.SafetyLevelGroup

import com.twitter.visibility.rules.Action

import com.twitter.visibility.rules.FreedomOfSpeechNotReachActions.FreedomOfSpeechNotReachActionBuilder

class TweetVisibilityPolicy(

rules: Map[SafetyLevel, FreedomOfSpeechNotReachActionBuilder[\_ <: Action]] = Map()) {

def getRules(): Map[SafetyLevel, FreedomOfSpeechNotReachActionBuilder[\_ <: Action]] = rules

}

object TweetVisibilityPolicy {

private[generators] val allApplicableSurfaces =

SafetyLevel.List.toSet --

SafetyLevelGroup.Special.levels --

Set(

SafetyLevel.SearchPeopleTypeahead,

SafetyLevel.UserProfileHeader,

SafetyLevel.UserScopedTimeline,

SafetyLevel.SpacesParticipants,

SafetyLevel.GryphonDecksAndColumns,

SafetyLevel.UserSettings,

SafetyLevel.BlockMuteUsersTimeline,

SafetyLevel.AdsBusinessSettings,

SafetyLevel.TrustedFriendsUserList,

SafetyLevel.UserSelfViewOnly,

SafetyLevel.ShoppingManagerSpyMode,

)

def builder(): TweetVisibilityPolicyBuilder = TweetVisibilityPolicyBuilder()

}

case class TweetVisibilityPolicyBuilder(

rules: Map[SafetyLevel, FreedomOfSpeechNotReachActionBuilder[\_ <: Action]] = Map()) {

def addGlobalRule[T <: Action](

actionBuilder: FreedomOfSpeechNotReachActionBuilder[T]

): TweetVisibilityPolicyBuilder =

copy(rules =

rules ++ TweetVisibilityPolicy.allApplicableSurfaces.map(\_ -> actionBuilder))

def addSafetyLevelRule[T <: Action](

safetyLevel: SafetyLevel,

actionBuilder: FreedomOfSpeechNotReachActionBuilder[T]

): TweetVisibilityPolicyBuilder = {

if (TweetVisibilityPolicy.allApplicableSurfaces.contains(safetyLevel)) {

copy(rules = rules ++ Map(safetyLevel -> actionBuilder))

} else {

this

}

}

def addSafetyLevelGroupRule[T <: Action](

group: SafetyLevelGroup,

actionBuilder: FreedomOfSpeechNotReachActionBuilder[T]

): TweetVisibilityPolicyBuilder =

copy(rules =

rules ++ group.levels.collect {

case safetyLevel if TweetVisibilityPolicy.allApplicableSurfaces.contains(safetyLevel) =>

safetyLevel -> actionBuilder

})

def addRuleForAllRemainingSafetyLevels[T <: Action](

actionBuilder: FreedomOfSpeechNotReachActionBuilder[T]

): TweetVisibilityPolicyBuilder =

copy(rules =

rules ++ (TweetVisibilityPolicy.allApplicableSurfaces -- rules.keySet)

.map(\_ -> actionBuilder).toMap)

def build: TweetVisibilityPolicy = {

new TweetVisibilityPolicy(rules)

}

}