package com.twitter.visibility.rules.providers

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.timelines.configapi.Params

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.rules.EvaluationContext

import com.twitter.visibility.rules.VisibilityPolicy

sealed abstract class ProvidedEvaluationContext(

visibilityPolicy: VisibilityPolicy,

params: Params,

statsReceiver: StatsReceiver)

extends EvaluationContext(

visibilityPolicy = visibilityPolicy,

params = params,

statsReceiver = statsReceiver)

object ProvidedEvaluationContext {

def injectRuntimeRulesIntoEvaluationContext(

evaluationContext: EvaluationContext,

safetyLevel: Option[SafetyLevel] = None,

policyProviderOpt: Option[PolicyProvider] = None

): ProvidedEvaluationContext = {

(policyProviderOpt, safetyLevel) match {

case (Some(policyProvider), Some(safetyLevel)) =>

new InjectedEvaluationContext(

evaluationContext = evaluationContext,

safetyLevel = safetyLevel,

policyProvider = policyProvider)

case (\_, \_) => new StaticEvaluationContext(evaluationContext)

}

}

}

private class StaticEvaluationContext(

evaluationContext: EvaluationContext)

extends ProvidedEvaluationContext(

visibilityPolicy = evaluationContext.visibilityPolicy,

params = evaluationContext.params,

statsReceiver = evaluationContext.statsReceiver)

private class InjectedEvaluationContext(

evaluationContext: EvaluationContext,

safetyLevel: SafetyLevel,

policyProvider: PolicyProvider)

extends ProvidedEvaluationContext(

visibilityPolicy = policyProvider.policyForSurface(safetyLevel),

params = evaluationContext.params,

statsReceiver = evaluationContext.statsReceiver)