package com.twitter.visibility.builder

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.Gate

import com.twitter.stitch.Stitch

import com.twitter.visibility.features.\_

import com.twitter.visibility.common.stitch.StitchHelpers

import scala.collection.mutable

object FeatureMapBuilder {

type Build = Seq[FeatureMapBuilder => FeatureMapBuilder] => FeatureMap

def apply(

statsReceiver: StatsReceiver = NullStatsReceiver,

enableStitchProfiling: Gate[Unit] = Gate.False

): Build =

fns =>

Function

.chain(fns).apply(

new FeatureMapBuilder(statsReceiver, enableStitchProfiling)

).build

}

class FeatureMapBuilder private[builder] (

statsReceiver: StatsReceiver,

enableStitchProfiling: Gate[Unit] = Gate.False) {

private[this] val hydratedScope =

statsReceiver.scope("visibility\_result\_builder").scope("hydrated")

val mapBuilder: mutable.Builder[(Feature[\_], Stitch[\_]), Map[Feature[\_], Stitch[\_]]] =

Map.newBuilder[Feature[\_], Stitch[\_]]

val constantMapBuilder: mutable.Builder[(Feature[\_], Any), Map[Feature[\_], Any]] =

Map.newBuilder[Feature[\_], Any]

def build: FeatureMap = new FeatureMap(mapBuilder.result(), constantMapBuilder.result())

def withConstantFeature[T](feature: Feature[T], value: T): FeatureMapBuilder = {

val anyValue: Any = value.asInstanceOf[Any]

constantMapBuilder += (feature -> anyValue)

this

}

def withFeature[T](feature: Feature[T], stitch: Stitch[T]): FeatureMapBuilder = {

val profiledStitch = if (enableStitchProfiling()) {

val featureScope = hydratedScope.scope(feature.name)

StitchHelpers.profileStitch(stitch, Seq(hydratedScope, featureScope))

} else {

stitch

}

val featureStitchRef = Stitch.ref(profiledStitch)

mapBuilder += FeatureMap.rescueFeatureTuple(feature -> featureStitchRef)

this

}

def withConstantFeature[T](feature: Feature[T], option: Option[T]): FeatureMapBuilder = {

option.map(withConstantFeature(feature, \_)).getOrElse(this)

}

}