package com.twitter.tweetypie.config

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.ExceptionCounter

import com.twitter.tweetypie.serverutil.ActivityUtil

import com.twitter.util.{Activity, Return, Try}

import com.twitter.util.logging.Logger

trait DynamicConfigLoader {

def apply[T](path: String, stats: StatsReceiver, parse: String => T): Activity[Option[T]]

}

object DynamicConfigLoader {

def apply(read: String => Activity[String]): DynamicConfigLoader =

new DynamicConfigLoader {

val logger = Logger(getClass)

private def snoopState[T](stats: StatsReceiver)(a: Activity[T]): Activity[T] = {

val pending = stats.counter("pending")

val failure = stats.counter("failure")

val success = stats.counter("success")

a.mapState {

case s @ Activity.Ok(\_) =>

success.incr()

s

case Activity.Pending =>

pending.incr()

Activity.Pending

case s @ Activity.Failed(\_) =>

failure.incr()

s

}

}

def apply[T](path: String, stats: StatsReceiver, parse: String => T): Activity[Option[T]] = {

val exceptionCounter = new ExceptionCounter(stats)

val rawActivity: Activity[T] =

snoopState(stats.scope("raw"))(

ActivityUtil

.strict(read(path))

.map(parse)

.handle {

case e =>

exceptionCounter(e)

logger.error(s"Invalid config in $path", e)

throw e

}

)

val stableActivity =

snoopState(stats.scope("stabilized"))(rawActivity.stabilize).mapState[Option[T]] {

case Activity.Ok(t) => Activity.Ok(Some(t))

case \_ => Activity.Ok(None)

}

stats.provideGauge("config\_state") {

Try(stableActivity.sample()) match {

case Return(Some(c)) => c.hashCode.abs

case \_ => 0

}

}

stableActivity

}

}

}