package com.twitter.servo.gate

import com.twitter.decider

import com.twitter.servo.util.Gate

import scala.annotation.tailrec

object DeciderGate {

/\*\*

\* Create a Gate[Unit] with a probability of returning true

\* that increases linearly with the availability of feature.

\*/

def linear(feature: decider.Feature): Gate[Unit] =

Gate(\_ => feature.isAvailable, "DeciderGate.linear(%s)".format(feature))

/\*\*

\* Create a Gate[Unit] with a probability of returning true

\* that increases exponentially with the availability of feature.

\*/

def exp(feature: decider.Feature, exponent: Int): Gate[Unit] = {

val gate = if (exponent >= 0) linear(feature) else !linear(feature)

@tailrec

def go(exp: Int): Boolean = if (exp == 0) true else (gate() && go(exp - 1))

Gate(\_ => go(math.abs(exponent)), "DeciderGate.exp(%s, %s)".format(feature, exponent))

}

/\*\*

\* Create a Gate[Long] that returns true if the given feature is available for an id.

\*/

def byId(feature: decider.Feature): Gate[Long] =

Gate(id => feature.isAvailable(id), "DeciderGate.byId(%s)".format(feature))

}