package com.twitter.servo.decider

import com.twitter.decider.{Decider, Feature}

import com.twitter.servo.util.Gate

import com.twitter.servo.gate.DeciderGate

/\*\*

\* Convenience syntax for creating decider gates

\*/

class DeciderGateBuilder(decider: Decider) {

/\*\*

\* idGate should be used when the result of the gate needs to be consistent between repeated

\* invocations, with the condition that consistency is dependent up on passing identical

\* parameter between the invocations.

\*/

def idGate(key: DeciderKeyName): Gate[Long] =

DeciderGate.byId(keyToFeature(key))

/\*\*

\* linearGate should be used when the probability of the gate returning true needs to

\* increase linearly with the availability of feature.

\*/

def linearGate(key: DeciderKeyName): Gate[Unit] =

DeciderGate.linear(keyToFeature(key))

/\*\*

\* typedLinearGate is a linearGate that conforms to the gate of the specified type.

\*/

def typedLinearGate[T](key: DeciderKeyName): Gate[T] =

linearGate(key).contramap[T] { \_ => () }

/\*\*

\* expGate should be used when the probability of the gate returning true needs to

\* increase exponentially with the availability of feature.

\*/

def expGate(key: DeciderKeyName, exponent: Int): Gate[Unit] =

DeciderGate.exp(keyToFeature(key), exponent)

def keyToFeature(key: DeciderKeyName): Feature = decider.feature(key.toString)

}