package com.twitter.tweetypie

package decider

import com.google.common.hash.Hashing

import com.twitter.decider.Decider

import com.twitter.decider.Feature

import com.twitter.servo.gate.DeciderGate

import com.twitter.servo.util.Gate

import java.nio.charset.StandardCharsets

import scala.collection.mutable

trait DeciderGates {

def overrides: Map[String, Boolean] = Map.empty

def decider: Decider

def prefix: String

protected val seenFeatures: mutable.HashSet[String] = new mutable.HashSet[String]

private def deciderFeature(name: String): Feature = {

decider.feature(prefix + "\_" + name)

}

def withOverride[T](name: String, mkGate: Feature => Gate[T]): Gate[T] = {

seenFeatures += name

overrides.get(name).map(Gate.const).getOrElse(mkGate(deciderFeature(name)))

}

protected def linear(name: String): Gate[Unit] = withOverride[Unit](name, DeciderGate.linear)

protected def byId(name: String): Gate[Long] = withOverride[Long](name, DeciderGate.byId)

/\*\*

\* It returns a Gate[String] that can be used to check availability of the feature.

\* The string is hashed into a Long and used as an "id" and then used to call servo's

\* DeciderGate.byId

\*

\* @param name decider name

\* @return Gate[String]

\*/

protected def byStringId(name: String): Gate[String] =

byId(name).contramap { s: String =>

Hashing.sipHash24().hashString(s, StandardCharsets.UTF\_8).asLong()

}

def all: Traversable[String] = seenFeatures

def unusedOverrides: Set[String] = overrides.keySet.diff(all.toSet)

/\*\*

\* Generate a map of name -> availability, taking into account overrides.

\* Overrides are either on or off so map to 10000 or 0, respectively.

\*/

def availabilityMap: Map[String, Option[Int]] =

all.map { name =>

val availability: Option[Int] = overrides

.get(name)

.map(on => if (on) 10000 else 0)

.orElse(deciderFeature(name).availability)

name -> availability

}.toMap

}