package com.twitter.tsp.common

import com.twitter.decider.Decider

import com.twitter.decider.RandomRecipient

import com.twitter.util.Future

import javax.inject.Inject

import scala.util.control.NoStackTrace

/\*

Provides deciders-controlled load shedding for a given displayLocation

The format of the decider keys is:

enable\_loadshedding\_<display location>

E.g.:

enable\_loadshedding\_HomeTimeline

Deciders are fractional, so a value of 50.00 will drop 50% of responses. If a decider key is not

defined for a particular displayLocation, those requests will always be served.

We should therefore aim to define keys for the locations we care most about in decider.yml,

so that we can control them during incidents.

\*/

class LoadShedder @Inject() (decider: Decider) {

import LoadShedder.\_

// Fall back to False for any undefined key

private val deciderWithFalseFallback: Decider = decider.orElse(Decider.False)

private val keyPrefix = "enable\_loadshedding"

def apply[T](typeString: String)(serve: => Future[T]): Future[T] = {

/\*

Per-typeString level load shedding: enable\_loadshedding\_HomeTimeline

Checks if per-typeString load shedding is enabled

\*/

val keyTyped = s"${keyPrefix}\_$typeString"

if (deciderWithFalseFallback.isAvailable(keyTyped, recipient = Some(RandomRecipient)))

Future.exception(LoadSheddingException)

else serve

}

}

object LoadShedder {

object LoadSheddingException extends Exception with NoStackTrace

}