package com.twitter.tweetypie

package backends

import com.twitter.finagle.service.RetryPolicy

import com.twitter.mediainfo.server.thriftscala.GetTweetMediaInfoRequest

import com.twitter.mediainfo.server.thriftscala.GetTweetMediaInfoResponse

import com.twitter.mediainfo.server.{thriftscala => mis}

import com.twitter.servo.util.FutureArrow

import com.twitter.tweetypie.util.RetryPolicyBuilder

object MediaInfoService {

import Backend.\_

type GetTweetMediaInfo = FutureArrow[mis.GetTweetMediaInfoRequest, mis.GetTweetMediaInfoResponse]

def fromClient(client: mis.MediaInfoService.MethodPerEndpoint): MediaInfoService =

new MediaInfoService {

val getTweetMediaInfo = FutureArrow(client.getTweetMediaInfo)

}

case class Config(

requestTimeout: Duration,

totalTimeout: Duration,

timeoutBackoffs: Stream[Duration]) {

def apply(svc: MediaInfoService, ctx: Backend.Context): MediaInfoService =

new MediaInfoService {

val getTweetMediaInfo: FutureArrow[GetTweetMediaInfoRequest, GetTweetMediaInfoResponse] =

policy("getTweetMediaInfo", ctx)(svc.getTweetMediaInfo)

}

private[this] def policy[A, B](name: String, ctx: Context): Builder[A, B] =

defaultPolicy(name, requestTimeout, retryPolicy, ctx, totalTimeout = totalTimeout)

private[this] def retryPolicy[B]: RetryPolicy[Try[B]] =

RetryPolicyBuilder.timeouts[Any](timeoutBackoffs)

}

}

trait MediaInfoService {

import MediaInfoService.\_

val getTweetMediaInfo: GetTweetMediaInfo

}