package com.twitter.tweetypie

package backends

import com.twitter.finagle.service.RetryPolicy

import com.twitter.servo.util.FutureArrow

import com.twitter.tweetypie.util.RetryPolicyBuilder

import com.twitter.user\_image\_service.thriftscala.ProcessTweetMediaRequest

import com.twitter.user\_image\_service.thriftscala.ProcessTweetMediaResponse

import com.twitter.user\_image\_service.thriftscala.UpdateProductMetadataRequest

import com.twitter.user\_image\_service.thriftscala.UpdateTweetMediaRequest

import com.twitter.user\_image\_service.thriftscala.UpdateTweetMediaResponse

import com.twitter.user\_image\_service.{thriftscala => uis}

object UserImageService {

import Backend.\_

type ProcessTweetMedia = FutureArrow[uis.ProcessTweetMediaRequest, uis.ProcessTweetMediaResponse]

type UpdateProductMetadata = FutureArrow[uis.UpdateProductMetadataRequest, Unit]

type UpdateTweetMedia = FutureArrow[uis.UpdateTweetMediaRequest, uis.UpdateTweetMediaResponse]

def fromClient(client: uis.UserImageService.MethodPerEndpoint): UserImageService =

new UserImageService {

val processTweetMedia = FutureArrow(client.processTweetMedia)

val updateProductMetadata: FutureArrow[UpdateProductMetadataRequest, Unit] = FutureArrow(

client.updateProductMetadata).unit

val updateTweetMedia = FutureArrow(client.updateTweetMedia)

}

case class Config(

processTweetMediaTimeout: Duration,

updateTweetMediaTimeout: Duration,

timeoutBackoffs: Stream[Duration]) {

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

new UserImageService {

val processTweetMedia: FutureArrow[ProcessTweetMediaRequest, ProcessTweetMediaResponse] =

policy("processTweetMedia", processTweetMediaTimeout, ctx)(svc.processTweetMedia)

val updateProductMetadata: FutureArrow[UpdateProductMetadataRequest, Unit] =

policy("updateProductMetadata", processTweetMediaTimeout, ctx)(svc.updateProductMetadata)

val updateTweetMedia: FutureArrow[UpdateTweetMediaRequest, UpdateTweetMediaResponse] =

policy("updateTweetMedia", updateTweetMediaTimeout, ctx)(svc.updateTweetMedia)

}

private[this] def policy[A, B](

name: String,

requestTimeout: Duration,

ctx: Context

): Builder[A, B] =

defaultPolicy(

name = name,

requestTimeout = requestTimeout,

retryPolicy = retryPolicy,

ctx = ctx,

exceptionCategorizer = {

case \_: uis.BadRequest => Some("success")

case \_ => None

}

)

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

RetryPolicyBuilder.timeouts[Any](timeoutBackoffs)

}

}

trait UserImageService {

import UserImageService.\_

val processTweetMedia: ProcessTweetMedia

val updateProductMetadata: UpdateProductMetadata

val updateTweetMedia: UpdateTweetMedia

}