package com.twitter.tweetypie

package backends

import com.twitter.expandodo.thriftscala.AttachmentEligibilityRequest

import com.twitter.expandodo.thriftscala.AttachmentEligibilityResponses

import com.twitter.expandodo.thriftscala.Card2Request

import com.twitter.expandodo.thriftscala.Card2RequestOptions

import com.twitter.expandodo.thriftscala.Card2Responses

import com.twitter.expandodo.thriftscala.CardsResponse

import com.twitter.expandodo.thriftscala.GetCardUsersRequests

import com.twitter.expandodo.thriftscala.GetCardUsersResponses

import com.twitter.expandodo.{thriftscala => expandodo}

import com.twitter.finagle.Backoff

import com.twitter.finagle.service.RetryPolicy

import com.twitter.servo.util.FutureArrow

import com.twitter.tweetypie.util.RetryPolicyBuilder

object Expandodo {

import Backend.\_

type GetCards = FutureArrow[Set[String], collection.Map[String, expandodo.CardsResponse]]

type GetCards2 = FutureArrow[

(Seq[expandodo.Card2Request], expandodo.Card2RequestOptions),

expandodo.Card2Responses

]

type GetCardUsers = FutureArrow[expandodo.GetCardUsersRequests, expandodo.GetCardUsersResponses]

type CheckAttachmentEligibility =

FutureArrow[Seq[

expandodo.AttachmentEligibilityRequest

], expandodo.AttachmentEligibilityResponses]

def fromClient(client: expandodo.CardsService.MethodPerEndpoint): Expandodo =

new Expandodo {

val getCards = FutureArrow(client.getCards \_)

val getCards2 = FutureArrow((client.getCards2 \_).tupled)

val getCardUsers = FutureArrow(client.getCardUsers \_)

val checkAttachmentEligibility = FutureArrow(client.checkAttachmentEligibility \_)

}

case class Config(

requestTimeout: Duration,

timeoutBackoffs: Stream[Duration],

serverErrorBackoffs: Stream[Duration]) {

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

new Expandodo {

val getCards: FutureArrow[Set[String], collection.Map[String, CardsResponse]] =

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

val getCards2: FutureArrow[(Seq[Card2Request], Card2RequestOptions), Card2Responses] =

policy("getCards2", ctx)(svc.getCards2)

val getCardUsers: FutureArrow[GetCardUsersRequests, GetCardUsersResponses] =

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

val checkAttachmentEligibility: FutureArrow[Seq[

AttachmentEligibilityRequest

], AttachmentEligibilityResponses] =

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

}

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

defaultPolicy(name, requestTimeout, retryPolicy, ctx)

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

RetryPolicy.combine[Try[B]](

RetryPolicyBuilder.timeouts[B](timeoutBackoffs),

RetryPolicy.backoff(Backoff.fromStream(serverErrorBackoffs)) {

case Throw(ex: expandodo.InternalServerError) => true

}

)

}

implicit val warmup: Warmup[Expandodo] =

Warmup[Expandodo]("expandodo")(

\_.getCards2((Seq.empty, expandodo.Card2RequestOptions("iPhone-13")))

)

}

trait Expandodo {

import Expandodo.\_

val getCards: GetCards

val getCards2: GetCards2

val getCardUsers: GetCardUsers

val checkAttachmentEligibility: CheckAttachmentEligibility

}