package com.twitter.tweetypie

package backends

import com.twitter.finagle.service.RetryPolicy

import com.twitter.servo.util.FutureArrow

import com.twitter.socialgraph.thriftscala.ExistsRequest

import com.twitter.socialgraph.thriftscala.ExistsResult

import com.twitter.socialgraph.thriftscala.RequestContext

import com.twitter.socialgraph.{thriftscala => sg}

import com.twitter.tweetypie.util.RetryPolicyBuilder

object SocialGraphService {

import Backend.\_

type Exists =

FutureArrow[(Seq[sg.ExistsRequest], Option[sg.RequestContext]), Seq[sg.ExistsResult]]

def fromClient(client: sg.SocialGraphService.MethodPerEndpoint): SocialGraphService =

new SocialGraphService {

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

def ping: Future[Unit] = client.ping().unit

}

case class Config(socialGraphTimeout: Duration, timeoutBackoffs: Stream[Duration]) {

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

new SocialGraphService {

val exists: FutureArrow[(Seq[ExistsRequest], Option[RequestContext]), Seq[ExistsResult]] =

policy("exists", socialGraphTimeout, ctx)(svc.exists)

def ping(): Future[Unit] = svc.ping()

}

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

name: String,

requestTimeout: Duration,

ctx: Context

): Builder[A, B] =

defaultPolicy(name, requestTimeout, retryPolicy, ctx)

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

RetryPolicyBuilder.timeouts[Any](timeoutBackoffs)

}

implicit val warmup: Warmup[SocialGraphService] =

Warmup[SocialGraphService]("socialgraphservice")(\_.ping)

}

trait SocialGraphService {

import SocialGraphService.\_

val exists: Exists

def ping(): Future[Unit]

}