package com.twitter.tweetypie

package service

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.servo.forked.Forked

import com.twitter.tweetypie.service.ReplicatingTweetService.GatedReplicationClient

/\*\*

\* Wraps an underlying ThriftTweetService, transforming external requests to replicated requests.

\*/

object ReplicatingTweetService {

// Can be used to associate replication client with a gate that determines

// if a replication request should be performed.

case class GatedReplicationClient(client: ThriftTweetService, gate: Gate[Unit]) {

def execute(executor: Forked.Executor, action: ThriftTweetService => Unit): Unit = {

if (gate()) executor { () => action(client) }

}

}

}

class ReplicatingTweetService(

protected val underlying: ThriftTweetService,

replicationTargets: Seq[GatedReplicationClient],

executor: Forked.Executor,

) extends TweetServiceProxy {

private[this] def replicateRead(action: ThriftTweetService => Unit): Unit =

replicationTargets.foreach(\_.execute(executor, action))

override def getTweetCounts(request: GetTweetCountsRequest): Future[Seq[GetTweetCountsResult]] = {

replicateRead(\_.replicatedGetTweetCounts(request))

underlying.getTweetCounts(request)

}

override def getTweetFields(request: GetTweetFieldsRequest): Future[Seq[GetTweetFieldsResult]] = {

if (!request.options.doNotCache) {

replicateRead(\_.replicatedGetTweetFields(request))

}

underlying.getTweetFields(request)

}

override def getTweets(request: GetTweetsRequest): Future[Seq[GetTweetResult]] = {

if (!request.options.exists(\_.doNotCache)) {

replicateRead(\_.replicatedGetTweets(request))

}

underlying.getTweets(request)

}

}