package com.twitter.tweetypie

package repository

import com.twitter.service.talon.thriftscala.\_

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

import com.twitter.tweetypie.backends.Talon

import com.twitter.tweetypie.client\_id.ClientIdHelper

import com.twitter.tweetypie.core.OverCapacity

case class UrlSlug(text: String) extends AnyVal

case class ExpandedUrl(text: String) extends AnyVal

object UrlRepository {

type Type = UrlSlug => Stitch[ExpandedUrl]

/\*\*

\* Builds a UrlRepository from a Talon.Expand arrow.

\*/

def apply(

talonExpand: Talon.Expand,

tweetypieClientId: String,

statsReceiver: StatsReceiver,

clientIdHelper: ClientIdHelper,

): Type = {

val observedTalonExpand: Talon.Expand =

talonExpand

.trackOutcome(statsReceiver, \_ => clientIdHelper.effectiveClientId.getOrElse("unknown"))

val expandGroup = SeqGroup[ExpandRequest, Try[ExpandResponse]] { requests =>

LegacySeqGroup.liftToSeqTry(

Future.collect(requests.map(r => observedTalonExpand(r).liftToTry)))

}

slug =>

val request = toExpandRequest(slug, auditMessage(tweetypieClientId, clientIdHelper))

Stitch

.call(request, expandGroup)

.lowerFromTry

.flatMap(toExpandedUrl(slug, \_))

}

def auditMessage(tweetypieClientId: String, clientIdHelper: ClientIdHelper): String = {

tweetypieClientId + clientIdHelper.effectiveClientId.mkString(":", "", "")

}

def toExpandRequest(slug: UrlSlug, auditMessage: String): ExpandRequest =

ExpandRequest(userId = 0, shortUrl = slug.text, fromUser = false, auditMsg = Some(auditMessage))

def toExpandedUrl(slug: UrlSlug, res: ExpandResponse): Stitch[ExpandedUrl] =

res.responseCode match {

case ResponseCode.Ok =>

// use Option(res.longUrl) because res.longUrl can be null

Option(res.longUrl) match {

case None => Stitch.NotFound

case Some(longUrl) => Stitch.value(ExpandedUrl(longUrl))

}

case ResponseCode.BadInput =>

Stitch.NotFound

// we shouldn't see other ResponseCodes, because Talon.Expand translates them to

// exceptions, but we have this catch-all just in case.

case \_ =>

Stitch.exception(OverCapacity("talon"))

}

}