package com.twitter.tweetypie

package handler

import com.twitter.servo.util.FutureArrow

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.\_

/\*\*

\* Handler for the `getTweetCounts` endpoint.

\*/

object GetTweetCountsHandler {

type Type = FutureArrow[GetTweetCountsRequest, Seq[GetTweetCountsResult]]

def apply(repo: TweetCountsRepository.Type): Type = {

def idToResult(id: TweetId, req: GetTweetCountsRequest): Stitch[GetTweetCountsResult] =

Stitch

.join(

// .liftToOption() converts any failures to None result

if (req.includeRetweetCount) repo(RetweetsKey(id)).liftToOption() else Stitch.None,

if (req.includeReplyCount) repo(RepliesKey(id)).liftToOption() else Stitch.None,

if (req.includeFavoriteCount) repo(FavsKey(id)).liftToOption() else Stitch.None,

if (req.includeQuoteCount) repo(QuotesKey(id)).liftToOption() else Stitch.None,

if (req.includeBookmarkCount) repo(BookmarksKey(id)).liftToOption() else Stitch.None

).map {

case (retweetCount, replyCount, favoriteCount, quoteCount, bookmarkCount) =>

GetTweetCountsResult(

tweetId = id,

retweetCount = retweetCount,

replyCount = replyCount,

favoriteCount = favoriteCount,

quoteCount = quoteCount,

bookmarkCount = bookmarkCount

)

}

FutureArrow[GetTweetCountsRequest, Seq[GetTweetCountsResult]] { request =>

Stitch.run(

Stitch.traverse(request.tweetIds)(idToResult(\_, request))

)

}

}

}