package com.twitter.tweetypie

package handler

import com.twitter.tweetypie.store.SetRetweetVisibility

import com.twitter.tweetypie.thriftscala.SetRetweetVisibilityRequest

import com.twitter.tweetypie.thriftscala.Share

import com.twitter.tweetypie.thriftscala.Tweet

/\*\*

\* Create a [[SetRetweetVisibility.Event]] from a [[SetRetweetVisibilityRequest]] and then

\* pipe the event to [[store.SetRetweetVisibility]]. The event contains the information

\* to determine if a retweet should be included in its source tweet's retweet count.

\*

\* Showing/hiding a retweet count is done by calling TFlock to modify an edge's state between

\* `Positive` <--> `Archived` in the RetweetsGraph(6) and modifying the count in cache directly.

\*/

object SetRetweetVisibilityHandler {

type Type = SetRetweetVisibilityRequest => Future[Unit]

def apply(

tweetGetter: TweetId => Future[Option[Tweet]],

setRetweetVisibilityStore: SetRetweetVisibility.Event => Future[Unit]

): Type =

req =>

tweetGetter(req.retweetId).map {

case Some(retweet) =>

getShare(retweet).map { share: Share =>

val event = SetRetweetVisibility.Event(

retweetId = req.retweetId,

visible = req.visible,

srcId = share.sourceStatusId,

retweetUserId = getUserId(retweet),

srcTweetUserId = share.sourceUserId,

timestamp = Time.now

)

setRetweetVisibilityStore(event)

}

case None =>

// No-op if either the retweet has been deleted or has no source id.

// If deleted, then we do not want to accidentally undelete a legitimately deleted retweets.

// If no source id, then we do not know the source tweet to modify its count.

Unit

}

}