package com.twitter.tweetypie.hydrator

import com.twitter.servo.util.Gate

import com.twitter.spam.rtf.thriftscala.SafetyLevel

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.StatsReceiver

import com.twitter.tweetypie.Tweet

import com.twitter.tweetypie.core.ValueState

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.repository.TweetRepository

import com.twitter.tweetypie.util.EditControlUtil

import com.twitter.tweetypie.serverutil.ExceptionCounter

import com.twitter.tweetypie.thriftscala.EditControl

import com.twitter.tweetypie.thriftscala.EditControlInitial

import com.twitter.tweetypie.thriftscala.FieldByPath

import com.twitter.tweetypie.util.TweetEditFailure.TweetEditGetInitialEditControlException

import com.twitter.tweetypie.util.TweetEditFailure.TweetEditInvalidEditControlException

/\*\*

\* EditControlHydrator is used to hydrate the EditControlEdit arm of the editControl field.

\*

\* For Tweets without edits and for initial Tweets with subsequent edit(s), this hydrator

\* passes through the existing editControl (either None or EditControlInitial).

\*

\* For edit Tweets, it hydrates the initial Tweet's edit control, set as a field on

\* the edit control of the edit Tweet and returns the new edit control.

\*/

object EditControlHydrator {

type Type = ValueHydrator[Option[EditControl], TweetCtx]

val hydratedField: FieldByPath = fieldByPath(Tweet.EditControlField)

def apply(

repo: TweetRepository.Type,

setEditTimeWindowToSixtyMinutes: Gate[Unit],

stats: StatsReceiver

): Type = {

val exceptionCounter = ExceptionCounter(stats)

// Count hydration of edit control for tweets that were written before writing edit control initial.

val noEditControlHydration = stats.counter("noEditControlHydration")

// Count hydration of edit control edit tweets

val editControlEditHydration = stats.counter("editControlEditHydration")

// Count edit control edit hydration which successfully found an edit control initial

val editControlEditHydrationSuccessful = stats.counter("editControlEditHydration", "success")

// Count of initial tweets being hydrated.

val editControlInitialHydration = stats.counter("editControlInitialHydration")

// Count of edits loaded where the ID of edit is not present in the initial tweet

val editTweetIdsMissingAnEdit = stats.counter("editTweetIdsMissingAnEdit")

// Count hydrated tweets where edit control is set, but neither initial nor edit

val unknownUnionVariant = stats.counter("unknownEditControlUnionVariant")

ValueHydrator[Option[EditControl], TweetCtx] { (curr, ctx) =>

curr match {

// Tweet was created before we write edit control - hydrate the value at read.

case None =>

noEditControlHydration.incr()

val editControl = EditControlUtil.makeEditControlInitial(

ctx.tweetId,

ctx.createdAt,

setEditTimeWindowToSixtyMinutes)

Stitch.value(ValueState.delta(curr, Some(editControl)))

// Tweet is an initial tweet

case Some(EditControl.Initial(\_)) =>

editControlInitialHydration.incr()

Stitch.value(ValueState.unmodified(curr))

// Tweet is an edited version

case Some(EditControl.Edit(edit)) =>

editControlEditHydration.incr()

getInitialTweet(repo, edit.initialTweetId, ctx)

.flatMap(getEditControlInitial(ctx))

.map { initial: Option[EditControlInitial] =>

editControlEditHydrationSuccessful.incr()

initial.foreach { initialTweet =>

// We are able to fetch the initial tweet for this edit but this edit tweet is

// not present in the initial's editTweetIds list

if (!initialTweet.editTweetIds.contains(ctx.tweetId)) {

editTweetIdsMissingAnEdit.incr()

}

}

val updated = edit.copy(editControlInitial = initial)

ValueState.delta(curr, Some(EditControl.Edit(updated)))

}

.onFailure(exceptionCounter(\_))

case Some(\_) => // Unknown union variant

unknownUnionVariant.incr()

Stitch.exception(TweetEditInvalidEditControlException)

}

}.onlyIf { (\_, ctx) => ctx.opts.enableEditControlHydration }

}

def getInitialTweet(

repo: TweetRepository.Type,

initialTweetId: Long,

ctx: TweetCtx,

): Stitch[Tweet] = {

val options = TweetQuery.Options(

include = TweetQuery.Include(Set(Tweet.EditControlField.id)),

cacheControl = ctx.opts.cacheControl,

enforceVisibilityFiltering = false,

safetyLevel = SafetyLevel.FilterNone,

fetchStoredTweets = ctx.opts.fetchStoredTweets

)

repo(initialTweetId, options)

}

def getEditControlInitial(ctx: TweetCtx): Tweet => Stitch[Option[EditControlInitial]] = {

initialTweet: Tweet =>

initialTweet.editControl match {

case Some(EditControl.Initial(initial)) =>

Stitch.value(

if (ctx.opts.cause.writing(ctx.tweetId)) {

// On the write path we hydrate edit control initial

// as if the initial tweet is already updated.

Some(EditControlUtil.plusEdit(initial, ctx.tweetId))

} else {

Some(initial)

}

)

case \_ if ctx.opts.fetchStoredTweets =>

// If the fetchStoredTweets parameter is set to true, it means we're fetching

// and hydrating tweets regardless of state. In this case, if the initial tweet

// doesn't exist, we return None here to ensure we still hydrate and return the

// current edit tweet.

Stitch.None

case \_ => Stitch.exception(TweetEditGetInitialEditControlException)

}

}

}