package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.TweetData

import com.twitter.tweetypie.core.ValueState

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.thriftscala.entities.Implicits.\_

import com.twitter.tweetypie.thriftscala.TextRange

import com.twitter.tweetypie.tweettext.Offset

import com.twitter.tweetypie.tweettext.TextModification

import com.twitter.tweetypie.tweettext.TweetText

import com.twitter.tweetypie.util.TweetLenses

object NoteTweetSuffixHydrator {

val ELLIPSIS: String = "\u2026"

private def addTextSuffix(tweet: Tweet): Tweet = {

val originalText = TweetLenses.text(tweet)

val originalTextLength = TweetText.codePointLength(originalText)

val visibleTextRange: TextRange =

TweetLenses

.visibleTextRange(tweet)

.getOrElse(TextRange(0, originalTextLength))

val insertAtCodePoint = Offset.CodePoint(visibleTextRange.toIndex)

val textModification = TextModification.insertAt(

originalText,

insertAtCodePoint,

ELLIPSIS

)

val mediaEntities = TweetLenses.media(tweet)

val urlEntities = TweetLenses.urls(tweet)

val modifiedText = textModification.updated

val modifiedMediaEntities = textModification.reindexEntities(mediaEntities)

val modifiedUrlEntities = textModification.reindexEntities(urlEntities)

val modifiedVisibleTextRange = visibleTextRange.copy(toIndex =

visibleTextRange.toIndex + TweetText.codePointLength(ELLIPSIS))

val updatedTweet =

Lens.setAll(

tweet,

TweetLenses.text -> modifiedText,

TweetLenses.urls -> modifiedUrlEntities.sortBy(\_.fromIndex),

TweetLenses.media -> modifiedMediaEntities.sortBy(\_.fromIndex),

TweetLenses.visibleTextRange -> Some(modifiedVisibleTextRange)

)

updatedTweet

}

def apply(): TweetDataValueHydrator = {

ValueHydrator[TweetData, TweetQuery.Options] { (td, \_) =>

val updatedTweet = addTextSuffix(td.tweet)

Stitch.value(ValueState.delta(td, td.copy(tweet = updatedTweet)))

}.onlyIf { (td, \_) =>

td.tweet.noteTweet.isDefined &&

td.tweet.noteTweet.flatMap(\_.isExpandable).getOrElse(true)

}

}

}