package com.twitter.product\_mixer.component\_library.decorator.urt.builder.richtext.twitter\_text

import com.twitter.product\_mixer.core.model.marshalling.response.urt.richtext.Plain

import com.twitter.product\_mixer.core.model.marshalling.response.urt.richtext.RichText

import com.twitter.product\_mixer.core.model.marshalling.response.urt.richtext.RichTextFormat

import com.twitter.product\_mixer.core.model.marshalling.response.urt.richtext.Strong

import scala.collection.mutable

object TwitterTextFormatProcessor {

lazy val defaultFormatProcessor = TwitterTextFormatProcessor()

}

/\*\*

\* Add the corresponding [[RichTextFormat]] extraction logic into [[TwitterTextRenderer]].

\* The [[TwitterTextRenderer]] after being processed will extract the defined entities.

\*/

case class TwitterTextFormatProcessor(

formats: Set[RichTextFormat] = Set(Plain, Strong),

) extends TwitterTextRendererProcessor {

private val formatMap = formats.map { format => format.name.toLowerCase -> format }.toMap

private[this] val formatMatcher = {

val formatNames = formatMap.keys.toSet

s"<(/?)(${formatNames.mkString("|")})>".r

}

def renderText(text: String): RichText = {

process(TwitterTextRenderer(text)).build

}

def process(richTextBuilder: TwitterTextRenderer): TwitterTextRenderer = {

val text = richTextBuilder.text

val nodeStack = mutable.ArrayStack[(RichTextFormat, Int)]()

var offset = 0

formatMatcher.findAllMatchIn(text).foreach { m =>

formatMap.get(m.group(2)) match {

case Some(format) => {

if (m.group(1).nonEmpty) {

if (!nodeStack.headOption.exists {

case (formatFromStack, \_) => formatFromStack == format

}) {

throw UnmatchedFormatTag(format)

}

val (\_, startIndex) = nodeStack.pop

richTextBuilder.mergeFormat(startIndex, m.start + offset, format)

} else {

nodeStack.push((format, m.start + offset))

}

richTextBuilder.remove(m.start + offset, m.end + offset)

offset -= m.end - m.start

}

case \_ => // if format is not found, skip this format

}

}

if (nodeStack.nonEmpty) {

throw UnmatchedFormatTag(nodeStack.head.\_1)

}

richTextBuilder

}

}

case class UnmatchedFormatTag(format: RichTextFormat)

extends Exception(s"Unmatched format start and end tags for $format")