time-stamp: 18.10.2023 06:43:03
updated: 23.10.2023 05:21:47

Фрагментация HTML

1. Фабула

(1) Представьте себе, что Вы – сотрудник Компании.

(2) Ранее команда получила обратную связь от конечного пользователя, из которой стало известно об ошибке в боте для корпоративного мессенджера: некоторые сообщения, который бот принимает со стороны backend, он не может разместить в канале мессенджера – возвращается ошибка, сообщение не отправляется, о чём бот сигнализирует в Sentry.

(3) Анализ логов позволил установить причину: оказалось, что API мессенжера (API содержит функцию отправки сообщения, принимающую текст сообщения в формате HTML) вводит ограничение на длину сообщения – 4096 символов (далее `max_len`), а отправляемое сообщение может оказаться существенно большего размера.

(4) Соответственно, возникает потребность разделить HTML-текст исходного сообщения на фрагменты, не превышающие `max_len` и, для каждого полученного фрагмента сформировать отдельное сообщение в канал мессенджера, сохраняя разметку исходного сообщения.

(5) Если бы речь шла об обычном тексте, это не вызвало сложностей, но как было сказано в (3), исходное сообщение представляет собой текст в формате HTML, который если порезать на части произвольно, некоторые теги окажутся разорванными, и сформированные фрагменты не будут корректными HTML, что приведёт к нарушению заданного форматирования в сообщениях конечному пользователю.

(6) Вам поручается обеспечить корректное разделение исходного HTML-сообщения на фрагменты так, чтобы каждый из этих фрагментов содержал корректную структуру тегов и, соответственно, содержал корректный HTML.

2. Правило разделения на фрагменты

(1) Текст сообщения с backend, которое предстоит обрабатывать, может содержать (но не обязательно должно) блоки – фрагменты HTML-текста, заключённые в следующие теги: `<p>`, `<b>`, `strong`, `<i>`, `<ul>`, `<ol>`, `<div>`, `<span>` (далее "блочные теги").

Для того, чтобы уложиться в заданный размер (`max_len`) блоки можно разделять, но важно сохранить их структуру: в фрагменте до места разделения блочные теги должны быть закрыты, а в фрагменте после места разделения – открыты заново.

(2) **ВАЖНО!** Остальные теги разрывать нельзя.

(3) Например, если место место разделения расположено, как показано в примере:

```
<p>
  ... ... ...
  <b>
    ... ... ...
    <a href="https://www.google.com/">Google search</a>
    <ul>
      <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
```

```
      <li>Ut enim ad minim veniam, quis nostrud exercitation ullamco.</li>
- - - Место разделения - - -
      <li>Duis aute irure dolor in reprehenderit in voluptate.</li>
    </ul>
  </b>
</p>
```

после разделения мы должны получить два фрагмента HTML с закрытыми тэгами блоков.

Первый фрагмент (обратите внимание, что тэги `<p>`, `<b>` и `<ul>` корректно закрыты)

```
<p>
  ... ... ...
  <b>
    ... ... ...
    <a href="https://www.google.com/">Google search</a>
    <ul>
      <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
      <li>Ut enim ad minim veniam, quis nostrud exercitation ullamco.</li>
    </ul>
  </b>
</p>
```

Второй фрагмент (обратите внимание, что теги `<p>`, `<b>` и `<ul>` открыты заново вначале фрагмента):

```
<p>
  <b>
    <ul>
      <li>Duis aute irure dolor in reprehenderit in voluptate.</li>
    </ul>
  </b>
</p>
```

2. Дополнительные требования и замечания по реализации

(1) Решение должно быть оформлено в в виде функции-генератора в модуле `msg_split.py` со следующим интерфейсом:

```
MAX_LEN = 4096


def split_message(source: str, max_len=MAX_LEN) -> Generator[str]:
    """Splits the original message (`source`) into fragments of the specified length
(`max_len`)."""
```

(2) Если какой-то фрагмент не удаётся сделать меньше `max_len` функция `split_message` должна выбрасывать исключение, в котором должна содержаться детализация возникшей проблемы.

(3) Напишите unit-тесты для решения.

(4) Для проверки результатов решения подготовьте скрипт, который будет выполнять чтение исходного сообщения из текстового файла, и выводить на стандартный вывод (`stdout`) результат обработки, разделяя фрагменты тэгом `<hr>`.

Скрипт должен принимать в качестве параметра значение `max_len`. Пример вызова скрипта из командной строки:

```
$ python split_msg.py --max-len=3072 ./test-1.html
```

(5) Результат работы должен быть представлен в виде репозитория на GitHub.
Файл `README.md` должен содержать указание на основной скрипт, и описание содержимого репозитория.

3. Рекомендации

Перечисленное в этом разделе носит исключительно рекомендательный характер.

(1) Выбор места разделения.
Это очевидно, но на всякий случай: при закрытии тегов блоков нельзя забывать, что закрывающие тэги также являются частью фрагмента, т.е. должны укладываться в ограничение `max_len` вместе с остальным содержимым этого фрагмента.

(2) Экспериментируйте с значением `max_len` – это позволит выявить проблемные места в алгоритме.

(3) Для обработки HTML можно использовать любое решение, как `HTMLParser` из "батареек" Python, так и более продвинутые подходы (например).
Возможно, Beautiful Soup окажется хорошим помощником, но окончательный выбор на Ваше усмотрение.

(4) При написании программы для проверки (раздел 2 тезис (4)) Вам понадобится обработка командной строки. Пакет Click является замечательным решением для этой цели, но выбор за Вами.

(5) Если Ваше решение будет использовать внешние зависимости (Beautiful Soup, Click), эти зависимости должны быть перечислены или в `requirements.txt` для *pip*, или (будет плюсом) в `pyproject.toml` для Poetry.