ページング UI

後はページングの為の UI を描画する処理をテンプレートに書く。公式ドキュメントにも書いてあるが `Paginator` や `Page` オブジェクトにはそれぞれページングの UI を描画するために使える値が含まれている。

```
paginator.count  # 検索対象総件数（例えば 732 件中 101 から 200 件を表示している場合の「732」）
paginator.num_pages  # 総ページ数（例えば 732 件で 100 件ずつの表示ならば「8」）
paginator.page_range  # [1, 2, 3, 4, 5, 6, 7, 8] などといった全ページ番号のリスト．これをテンプレート側で for 文で回して使う

page.has_next()  # 次ページがあるか
page.has_previous()  # 前ページがあるか
page.has_other_pages()  # 他のページがあるか（使うのか?）
page.next_page_number()  # 次ページ番号
page.previous_page_number()  # 前ページ番号
page.start_index()  # 開始件数（例えば 732 件中 101 から 200 件を表示している場合の「101」）
page.end_index()  # 終了件数（例えば 732 件中 101 から 200 件を表示している場合の「200」）

page.number  # ページ番号
page.paginator  # Page インスタンスから Paginator インスタンスを取得する．なので Paginator を Template 側に渡す必要はない
```

これを使って当 Blog では以下のようにページングを実装してみた（`posts` は View 側から渡された `Page` インスタンスである）:

```
<div class="pagination-count">
    {{ posts.paginator.count }} 件中 {{ posts.start_index }} 〜 {{ posts.end_index }} 件を表示しています。
</div>
{% if posts.paginator.num_pages > 1 %}
<div class="pagination">
    {% for i in posts.paginator.page_range %}
        {% if i == posts.number %}
    <em>{{ i }}</em>
        {% else %}
    <a href="?page={{ i }}{% if tag_id %}&tag_id={{ tag_id }}{% endif %}">{{ i }}</a>
        {% endif %}
    {% endfor %}
</div>
{% endif %}
```

ページング程度であればすべて自分で実装することもそこまで大変ではないが、フレームワークで用意されているページングの実装を使うとやはり記述するコードが減って楽だ。