テンプレートを拡張しよう

いよいよ最後のレッスンです。このレッスンではテンプレート拡張というものを学びます。

Djangoで開発する上では、テンプレートファイル（HTML）をたくさん作ることになるでしょう。そして、その全てのページで共通して表示させたい部分が出てくるはずです。サービス名（サイト名）やヘッダー、フッターは基本的にどのページにも表示されることが多いので、その例となります。

`<header>`タグや`<footer>`タグをコピーして全てのファイルにペーストしていく、というやり方でもよいのですが、何度も同じ作業を繰り返すのはとても面倒ですし、プログラミングの考え方的にも同じ働きを持つコードを繰り返し記述するのはナンセンスです。

そこで、Djangoでは共通部分は１つのファイルにまとめることができる機能があります。ヘッダーやフッター部分は、そのベースとなるファイルにまとめてしまうのです。そして、各ページでそのベースファイルを拡張させて編集していきます。

具体的なコードを見た方が理解がはやいと思いますので、まずはtemplates/app内にbase.htmlというファイルを作成してください。

~/memo/app/templates/app/base.html

```
<!DOCTYPE html>
<html>

<head>
  <title>DjangoBrosMemo</title>
</head>

<body>
  <h1>DjangoBrosMemo</h1>

  <div class="container">
    {% block content %}{% endblock %}
  </div>
</body>
</html>
```

base.htmlファイルは、あらゆるテンプレートファイルのベースとなるものですので、`<html>`、`<head>`、`<body>`タグなどの全ページに共通して必要とされるものを記述します。今回は、「DjangoBrosMemo」というサイト名も全ページに表示させたいので、`<h1>DjangoBrosMemo</h1>`と書いています。

`{% block content %}{% endblock %}`の部分で、他のHTMLを取り込みます。例えば、index.htmlは以下のようになります。

~/memo/app/templates/app/index.html

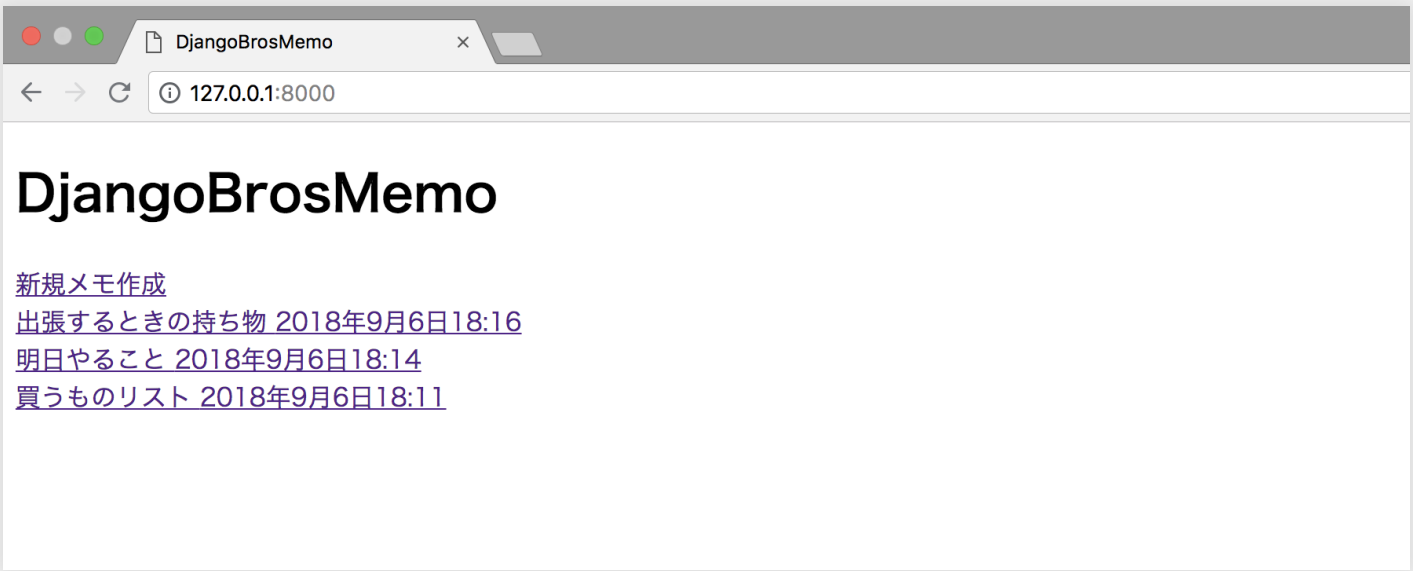
```
{% extends 'app/base.html' %}

{% block content %}

  <a class="btn" href="{% url 'app:new_memo' %}">新規メモ作成</a>
```

```
  {% for memo in memos %}
    <a href="{% url 'app:detail' memo.id %}" class="memo-title">
      <div>
        {{ memo }}
        <span class="updated_datetime">{{ memo.updated_datetime }}</span>
      </div>
    </a>
  {% endfor %}

{% endblock %}
```

１行目の、`{% extends 'app/base.html' %}`で、このファイルがbase.htmlを拡張したものであることを表しています。これにより、base.htmlで記述した`<h1>`タグなどがこのファイルでも表示されるようになります。

次に、index.htmlにこれまであったコードを、`{% block content %}`と`{% endblock %}`で囲っています。この囲った部分が、base.htmlの`{% block content %}{% endblock %}`の部分に取り込まれることになります。

このようにサイト名が表示されていれば、うまく拡張ができています。



ここまでできたら、他のテンプレートファイルもbase.htmlを拡張したものにしていきましょう。index.htmlと同様に、`{% extends 'app/base.html' %}`と`{% block content %}{% endblock %}`を足していくだけです。終わったら、全てのページにサイト名が表示されているか確認してください。

今回は、headタグの中身は非常に少ないコードですが、本来であればここにたくさんのコードが書かれることになります。それをいちいち全てのファイルに書いていく必要がなくなるため、開発工数の削減につながりますし、余計な記述ミスが減ったり可読性が高くなるというメリットもあります。また、仮にサイト名が「BrosMemo」に変わった場合は、base.htmlの`<title>DjangoBrosMemo</title>`の部分だけを変更すれば、全てのページのサイト名が変更されます。

CSSファイルを適用する

最後にCSSでデザインして終わりにしましょう。１つ目のチュートリアルでは、HTMLのタグに直接CSSを記述しましたが、今回はCSSファイルを作成して、それをテンプレートファイルで読み込むようにします。

CSS、Javascript、画像などのサーバー処理を伴わないファイルは静的ファイル（staticファイル）と呼びます。Djangoでは、静的ファイルはstaticというディレクトリに格納するのが一般的です。

では、実際にstaticディレクトリを作成してみましょう。staticディレクトリはアプリディレクトリの中に作成します。このようなディレクトリ構成を作成してください。

`memo/app/static/app/css/style.css`

templatesディレクトリを作ったときと似たような構成ですね。アプリ内にstaticというディレクトリを作成し、さらにその中にアプリと同名のディレクトリを作っています。今回作るのはCSSファイルなので、さらにその中にcssというディレクトリを作成しています。

画像を保存する場合は、`memo/app/static/app/image/sample.jpg`のような構成にするとよいでしょう。

いま作ったstyle.cssをHTMLで読み込むときは、以下のように記述します。１行目に`{% load static %}`、headタグ内に`<link rel="stylesheet" type="text/css" href="{% static 'app/css/style.css' %}">`を追加しています。

~/memo/app/templates/app/base.html

```
{% load static %}

<!DOCTYPE html>
<html>

<head>
  <title>DjangoBrosMemo</title>
  <link rel="stylesheet" type="text/css" href="{% static 'app/css/style.css' %}">
</head>

<body>
  <h1>DjangoBrosMemo</h1>

  <div class="container">
    {% block content %}{% endblock %}
  </div>

</body>
</html>
```

`{% static 'app/css/style.css' %}`は、staticテンプレートタグといいます。パスを指定することによって、自動的にstaticディレクトリ配下の静的ファイルを読み込んでくれます。

staticテンプレートタグが使えるのは、１行目の`{% load static %}`があるからです。

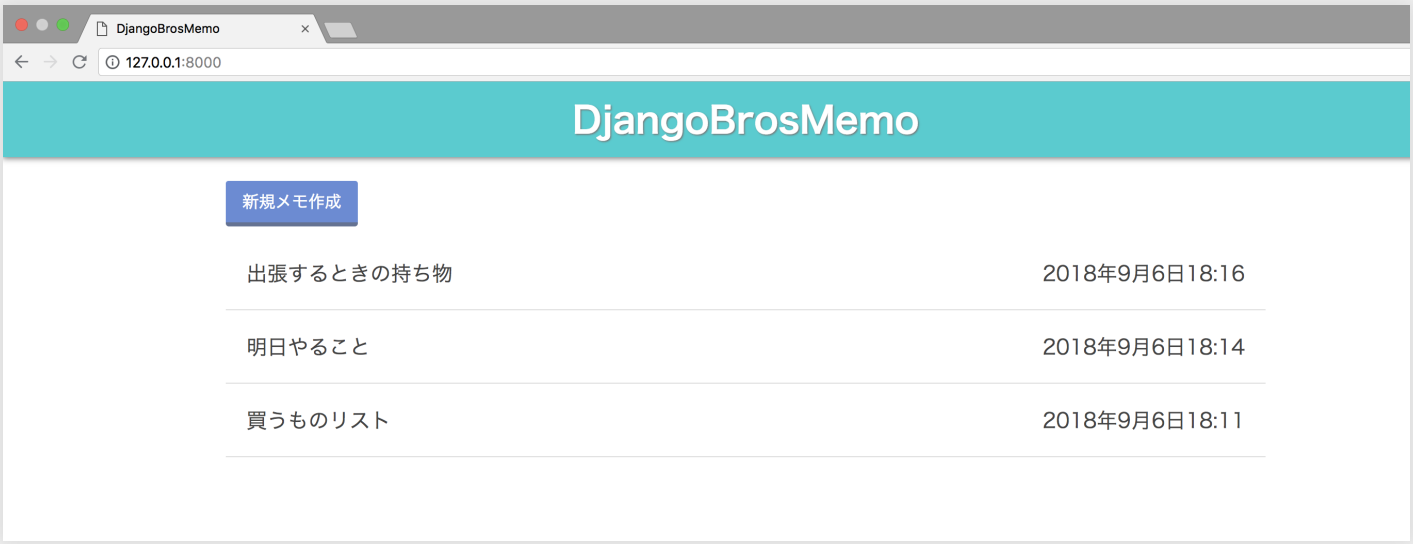
style.cssには、以下のコードをコピーして保存してください。保存できたら、ページをリロードしてCSSが適用されているか確認してください。稀にリロードしてもCSSが適用されない場合があります。その場合は、スーパーリロードをしてみてください。スーパーリロード

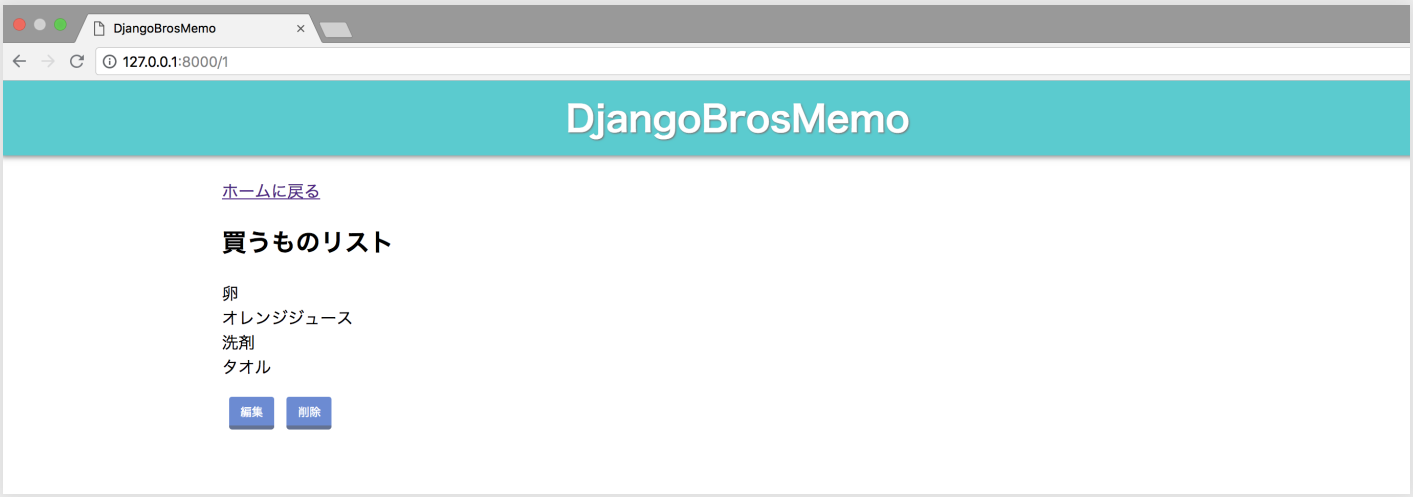
とは、キャッシュをクリアした上で再度読み込みをすることです。方法はブラウザなどの環境によって異なりますので、検索して調べてみてください。

~/memo/app/static/app/css/style.css

```
body {
  margin: 0;
  padding: 0;
}

button{
  background:none;
  border:0;
  cursor: pointer;
}

form{
  display: inline-block;
}

h1 {
  margin-top: 0;
  padding: 10px;
  font-size: 35px;
  background-color: #00ced1;
  color: #fff;
  text-align: center;
  box-shadow: 0 1px 8px #777;
  text-shadow: 1px 1px 1px #777;
}

.btn{
  display: inline-block;
  padding: 0.5em 1em;
  text-decoration: none;
  background: #668ad8;
  color: #FFF;
  border-bottom: solid 4px #627295;
  border-radius: 3px;
  margin-bottom: 10px;
}

.container{
  width: 70%;
  margin: auto;
}

.memo-title {
  display: block;
  border-bottom: 1px solid #ddd;
  padding: 20px;
  font-size: 20px;
  color: #444;
  text-decoration: none;
}

.memo-title:hover{
  background-color: #eee;
}

.updated_datetime{
  float: right;
}
```

うまくできたでしょうか。CSSが適用されると、これらの画像のようになります。





Congratulations!

これで、このチュートリアルはおわりです！１つ目のチュートリアルでは、ページを表示させるRead機能だけでしたが、今回は新規作成、アップデート、削除などのサーバーサイド側の処理をたくさん学習しました。Read機能だけでは、いわばホームページのようなものしか作れませんが、CRUD機能によってデータを操作できるので一気に便利になります。ほとんどのWebサービスはCRUDをベースに作られていますのでしっかりポイントをおさえておきましょう。

次のチュートリアルでは、ログイン機能なども学び、より発展的なサービスを作りたいと思いますので楽しみにしててくださいね！！