Definição de *Template*

Basicamente, um *template* é um arquivo de texto que pode ser transformado em outro arquivo (um arquivo HTML, um CSS, um CSV, etc…).

Um *template* contém:

- **Variáveis** que podem ser substituídas por valores, a partir do processamento por uma *Engine* de *Templates* (núcleo, *core*, "motor" de *templates*).
- ***Tags*** que controlam a lógica do *template* .
- **Filtros** que adicionam funcionalidades ao *template*.

Por exemplo, abaixo está representado um *template* mínimo que demonstra alguns conceitos básicos:

```
1  {  # base.html contém o template que usaremos como esqueleto #}
2  {% extends "base.html" % }
3
4  {% block conteudo % }
5      < h1 > {{section.title}} < /h1 >
6
7    {% for funcionario in funcionarios % }
8      < h2 >
9      < a href = "{% url 'website:funcionario_detalhe' pk=funcionario.id %}" >
10     {{funcionario.nome | upper}}
11     < /a >
12     < / h2 >
13   {% endfor % }
14 {% endblock % }
15
```

Alguns pontos importantes:

- **Linha 1:** Utilizamos comentário com a *tag* {# comentário #}.
- **Linha 2:** Utilizamos {% extends "base.html" %} para estender de outro *template*, ou seja, utilizá-lo como base, passando o caminho para ele.
- **Linha 4:** Podemos facilitar a organização do *template*, criando blocos com {% block <nome_do_bloco> %}{% endblock %}.
- **Linha 5:** Podemos interpolar código vindo do servidor com o conteúdo do nosso *template* com {{ secao.titulo }} - dessa forma, estamos acessando o atributo titulo do objeto secao (que deve estar no **Contexto** da requisição).
- **Linha 7:** É possível iterar sobre objetos de uma lista através da *tag* {% for objeto in lista %}{% endfor %}.

- **Linha 10:** Podemos utilizar **filtros** para aplicar alguma função à algum conteúdo. Nesse exemplo, estamos aplicando o filtro upper, que transforma todos os caracteres da *string* em maísculos, no conteúdo de funcionario.nome. Também é possível encadear filtros, por exemplo: {{ funcionario.nome|upper|cut:" " }}

Django criou uma linguagem que contém todos esses elementos.

Chamaram-na de **DTL** - *Django Template Language*! Veremos mais dela ali embaixo!

Para utilizar os *templates* do Django, é necessário primeiro **configurar sua utilização.**

E é isso que veremos agora!

Configuração

Se você já deu uma espiada no nosso arquivo de configurações, o settings.py, você já deve ter visto a seguinte configuração:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {},
    },
]
```

Mas você já se perguntou **o que essa configuração quer dizer**?

Nela:

- BACKEND é o caminho para uma classe que implementa a API de *templates* do Django.
- DIRS define uma lista de diretórios onde o Django deve procurar por *templates*. A ordem da lista define a ordem de busca.
- APP_DIRS define se o Django deve procurar por *templates* dentro dos diretórios dos *apps* instalados em INSTALLED_APPS.
- OPTIONS contém configurações específicas do BACKEND escolhido, ou seja, dependendo do *backend* de *templates* que você escolher, você poderá configurá-lo utilizando o OPTIONS.

Fonte:

https://pythonacademy.com.br/blog/desenvolvimento-web-com-python-e-django-template

Documentação Django:

https://docs.djangoproject.com/pt-br/3.1/intro/tutorial03/