

Django

03 Templates

Professor: Ritomar Torquato

03 Templates

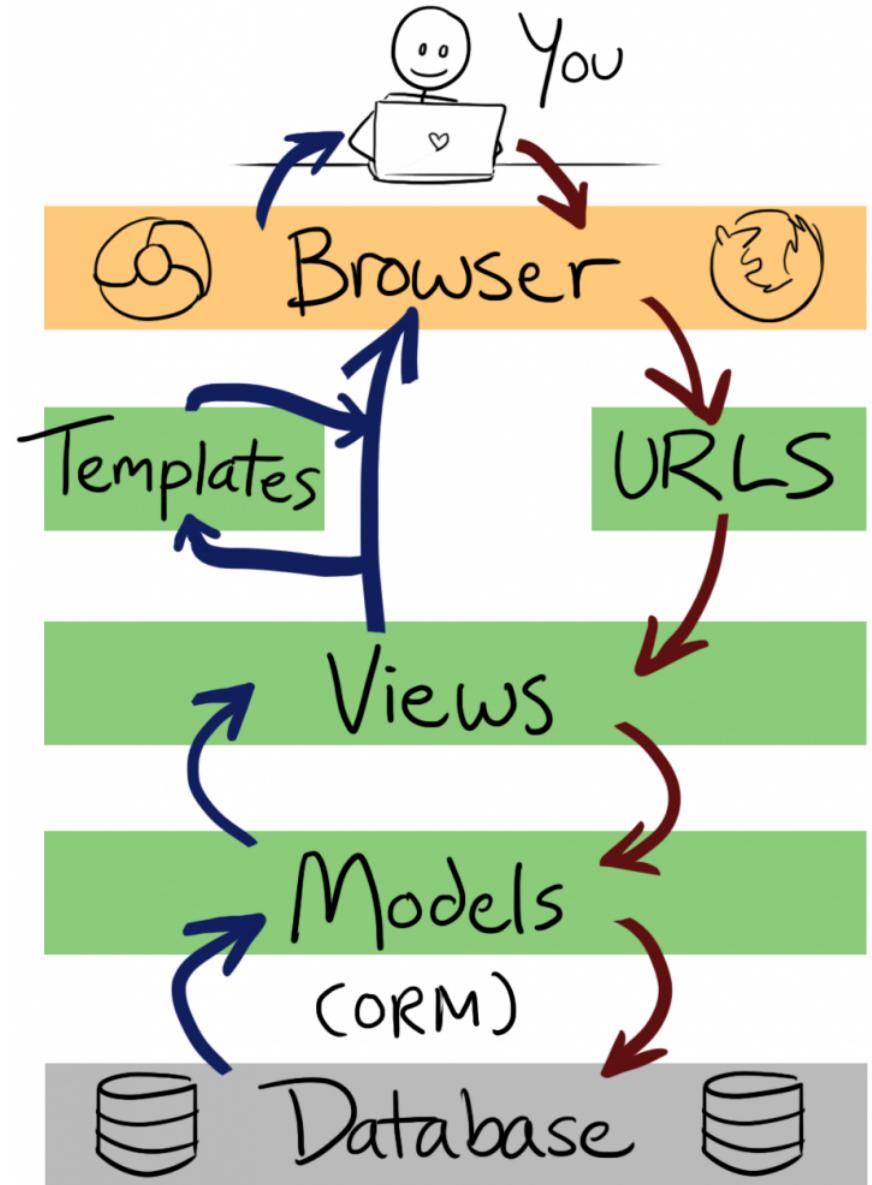
Objetivos: mostrar como a camada de visão do django pode ser auxiliada pelo uso de templates.

O que são templates?

Django funciona no padrão M T V



Django funciona na
arquitetura M T V



O que são templates?



Forma para ser preenchida ...

O que são templates?



... em um contexto ...

O que são templates?



... para gerar o resultado.

O que são templates?

Em Django é texto, com marcações especiais pré-definidas ...

```
Template("Olá {{nome}}! Tenha {{ período }}.")
```

... para ser contextualizado ...

```
Context("Maria", "um bom dia")
```

... e gerar um resultado.

Olá Maria! Tenha um bom dia.

O que são templates?

Templates e Contextos no shell do Python

```
(myenv) $> python manage.py shell
Python 3.5.1
>>> from django.template import Template, Context
>>> template = Template ("Olá, {{ nome }}! {{ msg }}.")
>>> context1 = Context({"nome": "Maria", "msg": "Bom dia"})
>>> context2 = Context({"nome": "João", "msg": "Boa tarde"})
>>> context3 = Context({"nome": "Ana", "msg": "Boa noite"})
>>> template.render(context1)
'Olá, Maria! Bom dia.'
>>> template.render(context2)
'Olá, João! Boa tarde.'
>>> template.render(context3)
'Olá, Ana! Boa noite.'
```

O que são templates?

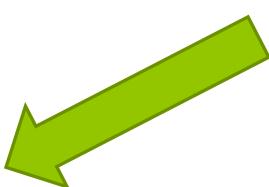
Normalmente templates são carregados de arquivos HTML.



O que são templates?

O local apropriado para colocar os arquivos de template é em uma pasta chamada "templates" dentro da pasta da aplicação.

```
mysite/  
  manage.py  
  mysite/  
    core/  
      templates/  
        index.html  
      static/  
        style.css  
        profile.png
```



O que são templates?

Django possui uma linguagem própria para templates e normalmente gera documento HTML

A seguir, mostramos um exemplo básico de template...

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

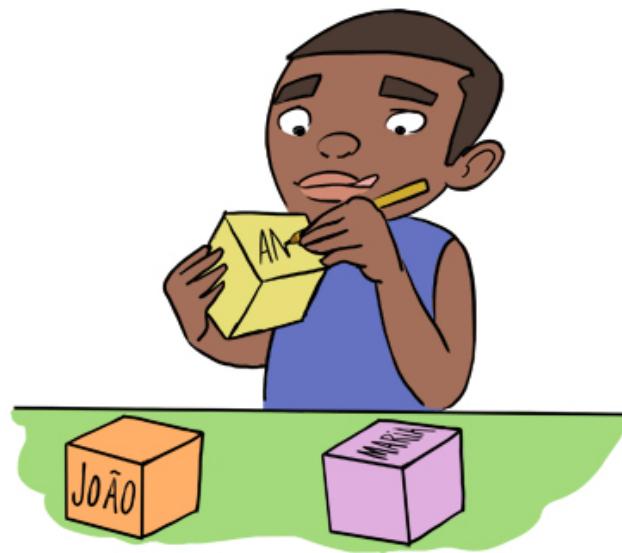
{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Template para gerar HTML. Vejamos elementos chave...

`{ { variável } }`

`{ { variável } }` é substituída por um valor



{ { variável } }

O nome deve conter apenas alfanuméricos e sublinhado (_)

É possível acessar atributos e métodos de objetos; itens de listas, tuplas e dicionários usando ponto (.)

```
{ { dicionario.chave } }      # Acessa a chave do dicionário  
{ { objeto.atributo } }       # Acessa o atributo do objeto  
{ { objeto.metodo } }         # Acessa o método do objeto  
{ { lista.0 } }               # Acessa o índice da lista
```

Ordem: Primeiro o Django procura por um dicionário; depois procurar por um atributo ou método de objeto; e depois ele irá procurar um índice de lista.

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Variáveis

| filtro

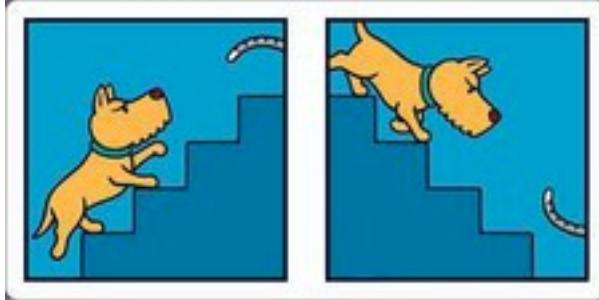
| filtro formata o valor para exibição

```
 {{ nome|lower }}      # mostra a variável {{ nome }} aplicando  
                      # o filtro | lower que converte para  
                      # minúsculas
```

A E I O U

a e i o u

| filtro

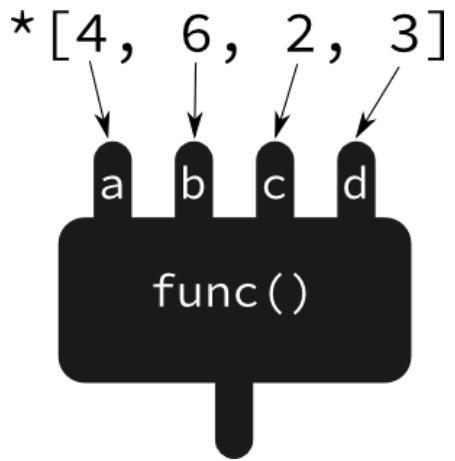


Filtros podem ser combinados. A saída de um filtro pode ser usada como entrada de outro.

```
 {{ text|escape|linebreaks } }
```

```
# altera a variável {{ text }} com caracteres de escape HTML e,  
# então, converte quebras de linhas em tags de parágrafo <p>.
```

| filtro

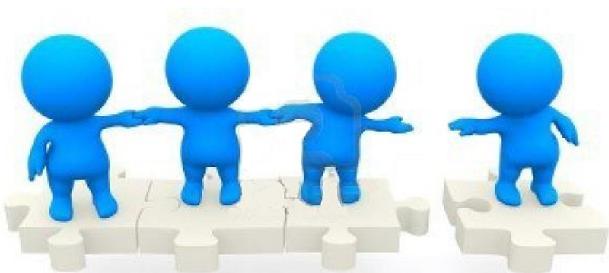


Alguns filtros podem receber argumentos, usando dois pontos (:)

```
{{ bio|truncatewords:10 }}
```

Mostra apenas os primeiro 10 caracteres da variável {{ bio }}.

| filtro



Argumentos que contém caracteres de espaço devem ser colocados entre aspas.

```
# Junta os elementos de uma lista com uma vírgula seguida de  
# um caractere de espaço.  
{{ list|join:", " }}
```

| filtro

Django vem com cerca de sessenta filtros. Você pode ler tudo sobre eles na documentação de referência.

<https://docs.djangoproject.com/pt-br/1.10/ref/templates/builtins/#ref-templates-builtins-filters>

A seguir, comentaremos sobre alguns desses filtros.

| filtro

`{{ value|default:"vazio" }}` Se uma variável é Falsa ou vazia, usa o valor fornecido, caso contrário, usa o valor da variável.

`{{ value|length }}` Retorna o comprimento da variável. Funciona para strings e listas. Se value é ['a', 'b', 'c'], a saída será 3.

`{{ value|filesizeformat }}` Formata um valor para a forma "amigável". Se value é 123456789, a saída do filtro será 117.7 MB.

`{{ value|join:" // " }}` Junda uma lista com uma string. Se value é a lista ['a', 'b', 'c'], a saída será a string "a // b // c".

`{{ value|upper }}` Converte uma stringa para maiúsculas. Se value é "Maria é bonita.", a saída será "MARIA É BONITA.".

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

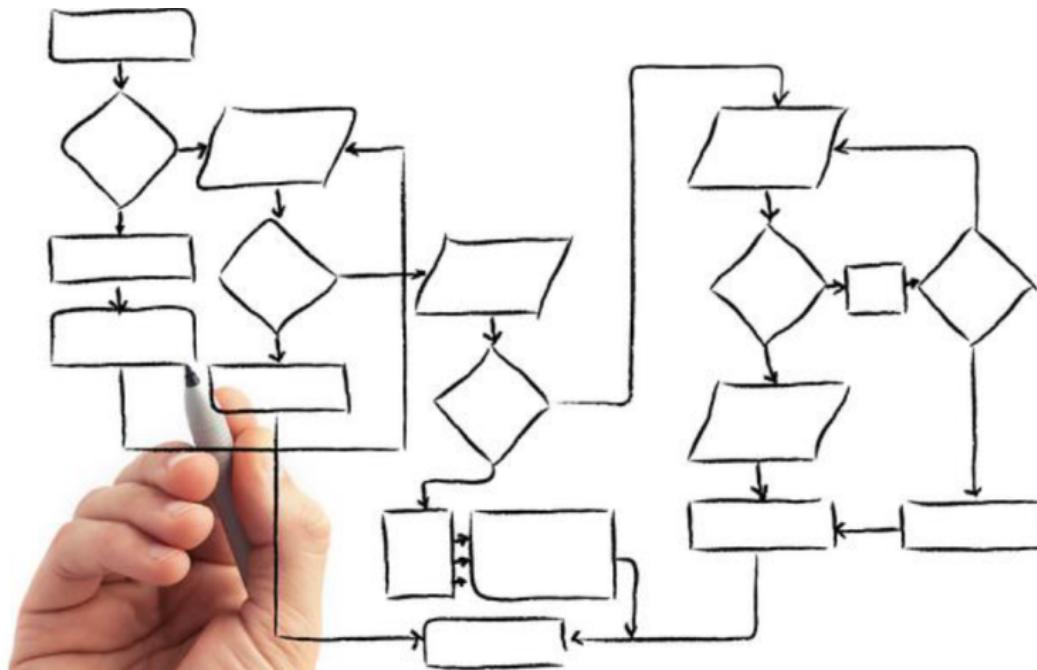
{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Exemplos de filtros

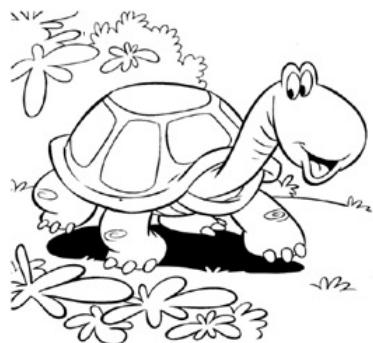
{% tag %}

{% tag %} pode controlar o fluxo



{ % tag % }

{ % tag % } pode criar texto



{% tag %}

{% tag %} pode carregar texto externo

IMPORT

{ % tag % }

Algumas tags requerem marcadores de
início e fim.

{ % tag % }

...

tag contents

...

{ % endtag % }) .

{% for %}

A tag for repete para cada item em uma lista,
Por exemplo, para exibir uma lista de atletas
provenientes de `athlete_list`:

```
<ul>
  {% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
  {% endfor %}
</ul>
```

if, elif, e else

Avalia uma variável e, se verdadeira, um bloco é executado.

```
{% if athlete_list %}  
    Number of athletes: {{ athlete_list|length }}  
{% elif athlete_in_locker_room_list %}  
    Athletes should be out of the locker room soon!  
{% else %}  
    No athletes.  
{% endif %}
```

if, elif, e else

É possível usar filtros e operadores em instruções.

```
{% if athlete_list|length > 1 %}  
    Team: {% for athlete in athlete_list %} ... {% endfor %}  
{% else %}  
    Athlete: {{ athlete_list.0.name }}  
{% endif %}
```

É importante ficar atendo que vários filtros retornam string, e usá-los com operadores matemáticos pode não funcionar como esperado.
length é uma exceção que funciona bem.

{# comentários #}

É possível criar comentários com {# ... #}. Tudo entre esses caracteres será ignorado.

{# *greeting* #}hello

{# {*% if foo %*}bar{*% else %*} #}

Arquivos estáticos

Sites geralmente precisam servir arquivos adicionais como imagens, JavaScript, ou CSS. No Django, nós nos referímos a estes arquivos como “arquivos estáticos”.



Arquivos estáticos

O local apropriado para colocar os arquivos estáticos é em uma pasta chamada "static" dentro da pasta da aplicação.

```
mysite/
└── manage.py
└── mysite/
    └── core/
        └── templates/
            └── index.html
        └── static/ ←
            ├── style.css
            └── profile.png
```

Arquivos estáticos

```
{% load static %}
```

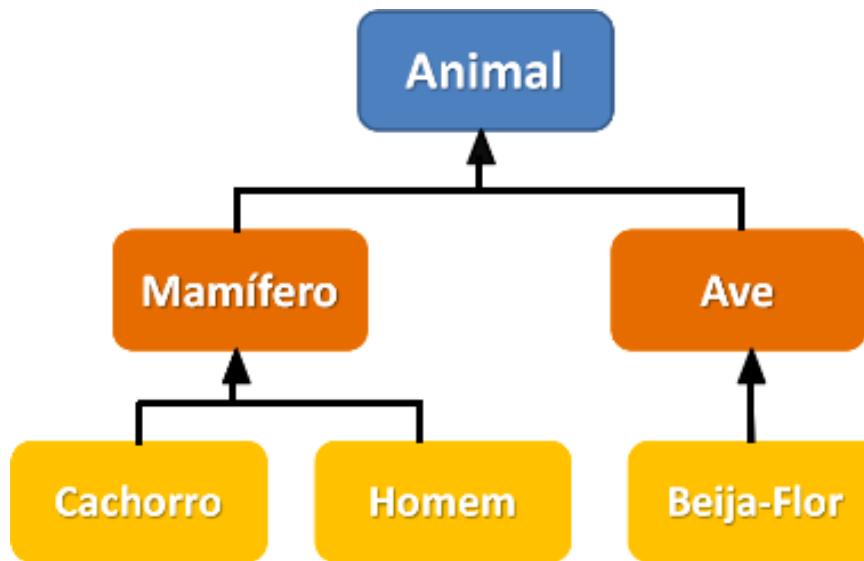
```
{% load static %}  

```

```
{% static "caminho/arquivos.ext" %}:
```

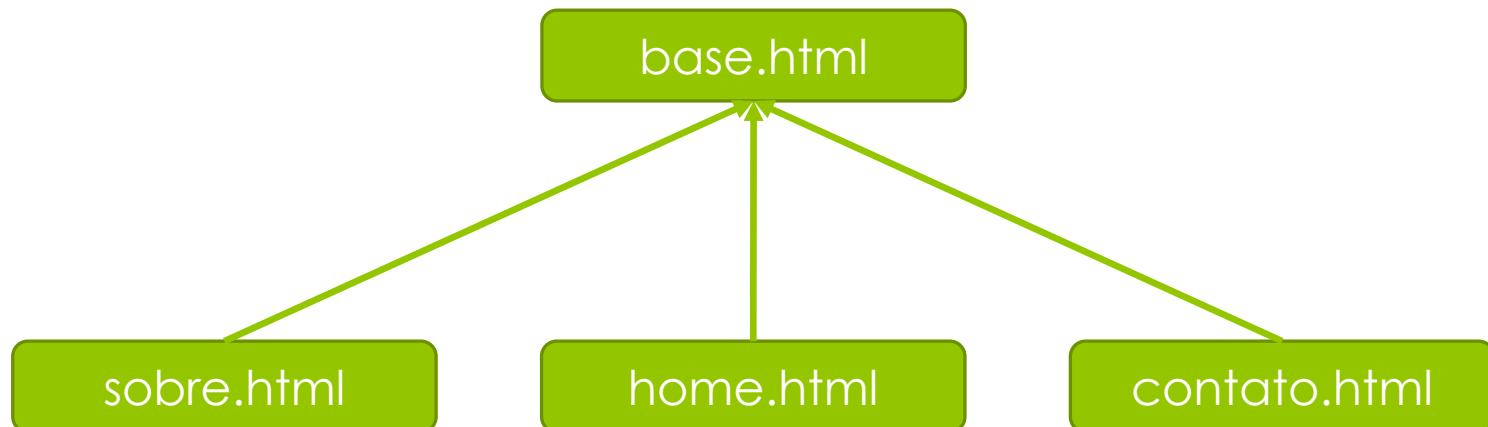
block e extends

Servem para o mecanismo de herança de templates.



block e extends

A herança de templates permite a construção de um esqueleto base que contém os elementos do site e define blocos que serão sobrescritos nos filhos.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.css" />
    <title>{% block title %}My amazing site{% endblock %}</title>
</head>

<body>
    <div id="sidebar">
        {% block sidebar %}
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/blog/">Blog</a></li>
        </ul>
        {% endblock %}
    </div>

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

É comum que o nome do bloco tenha o mesmo nome de um id de elemento, mas não é obrigatório.

base.html

```
{% extends "base.html" %}
```

```
{% block title %}My amazing blog{% endblock %}
```

```
{% block content %}  
{% for entry in blog_entries %}  
    <h2>{{ entry.title }}</h2>  
    <p>{{ entry.body }}</p>  
{% endfor %}  
{% endblock %}
```

Perceba que o bloco sidebar de base.html não foi referenciado.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="stylesheet" href="style.css" />
  <title>My amazing blog</title>
</head>

<body>
  <div id="sidebar">
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/blog/">Blog</a></li>
    </ul>
  </div>

  <div id="content">
    <h2>Entry one</h2>
    <p>This is my first entry.</p>

    <h2>Entry two</h2>
    <p>This is my second entry.</p>
  </div>
</body>
</html>
```

Sidebar permanece
inalterado.

Página
Renderizada

Os demais blocos são
modificados com o novo
conteúdo.

block e extends

Herança em vários níveis:

- base.html que encapsula a visão geral do site.
- sectionname.html para cada sessão do site. Por exemplo, base_news.html, base_sports.html. Esses herdam de base.html e incluem os estilos específicos.
- templates para cada tipo de página, como um novo artigo ou entrada no blog. Esses herdam do template da sessão apropriada.

Se utilizada, a tag `{% extends %}` deve ser a primeira que aparece no template.

block e extends

Herança em vários níveis:

Se você não é um web desing você é normal. Para fazer algo estilizado e elegante podemos usar templates disponíveis gratuitamente na Internet. Por exemplo em:

www.quackit.com/html/templates/simple_website_templates.cfm