

Flask

Desenvolvendo aplicações Web com o Framework Flask - Aula 4

Um pouco mais sobre Templates com Jinja2 – Parte 1

Flask – Templates com Jinja2

O que veremos nesta aula

Nesta aula serão abordados conceitos básicos de templates Jinja2 à partir da perspectiva do Flask. Também veremos como criar aplicativos com templates modulares e extensíveis.

Veremos:

- Templates Jinja2
- Usar o Bootstrap
- Composição de bloco e herança de layout
- Criando um processador de contexto customizado
- Criando um filtro personalizado Jinja2
- Criando uma macro personalizada para formulários
- Formatação de data e hora com Moment.js

Flask – Templates com Jinja2

Introdução

Nós já vimos nas aulas anteriores que podemos criar uma aplicação web completa em Flask sem a necessidade de usarmos templates, porém, para grandes aplicações que envolvem milhares de linhas de código HTML, JavaScript e CSS, obviamente que precisaremos utilizar templates.

O Flask fornece suporte padrão para o Jinja2, embora possamos utilizar qualquer motor de templates adequado.

O Jinja2 oferece muitos recursos adicionais que fazem nossos modelos se tornarem poderosos e modulares.

Flask – Templates com Jinja2

Templates Jinja2

A maioria das aplicações Flask segue um padrão específico para os modelos de layout.

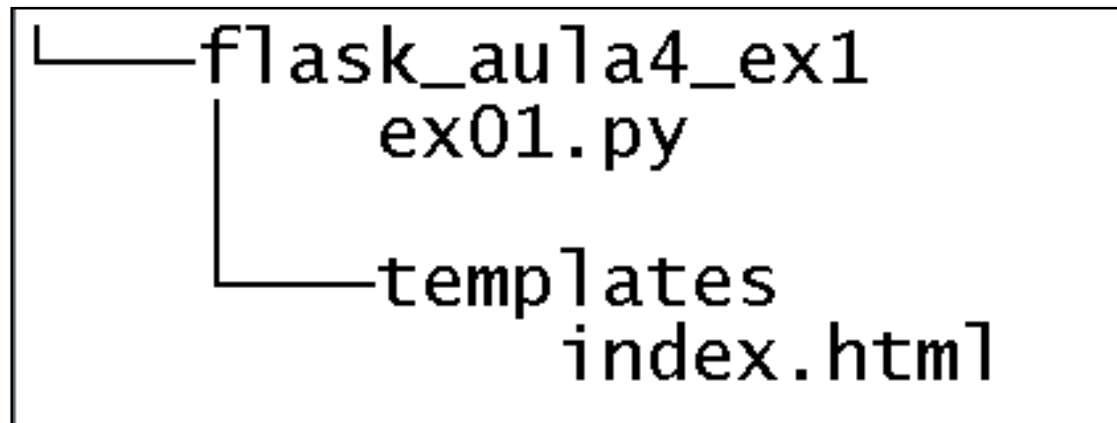
Como vimos nas aulas anteriores, por padrão o Flask espera que os templates (ou modelos) estejam dentro de uma pasta chamada “templates” na raiz da aplicação.

Se esta pasta estiver presente, o Flask irá ler automaticamente o conteúdo, disponibilizando o mesmo para uso através do método `render_template()`.

Flask – Templates com Jinja2

Templates Jinja2

Nossa aplicação terá um arquivo chamado `ex01.py` e o template chamado `index.html` na pasta `templates`.



Flask – Templates com Jinja2

Templates Jinja2

Código do arquivo ex01.py, conteúdo do index.html e resultado.

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
@app.route('/ola')
def ola_mundo():
    usuario = request.args.get('usuario', 'evaldo')
    return render_template('index.html', usuario=usuario)

if __name__ == '__main__':
    app.run(debug = True)
```

```
<html>
  <head>
    <title>Curso Python para Todos</title>
  </head>
  <body>
    <h1>Olá {{usuario}}!!!</h1>
    <p>Bem-vindo à aula de Flask.</p>
  </body>
</html>
```

Olá maria!!!

Bem-vindo à aula de Flask.

Olá evaldo!!!

Bem-vindo à aula de Flask.

Flask – Templates com Jinja2

Templates Jinja2

No arquivo ex01.py, o argumento transmitido na URL é obtido à partir do objeto request usando `request.args.get('usuario')` (o segundo argumento é o valor padrão, caso seja omitido) e passado para o contexto do template que está sendo renderizado usando `render_template`. O argumento é então analisado usando o marcador e posição Jinja2 “`{{usuario}}`”

```
...  
usuario = request.args.get('usuario', 'evaldo')  
return render_template('index.html', usuario=usuario)  
...
```

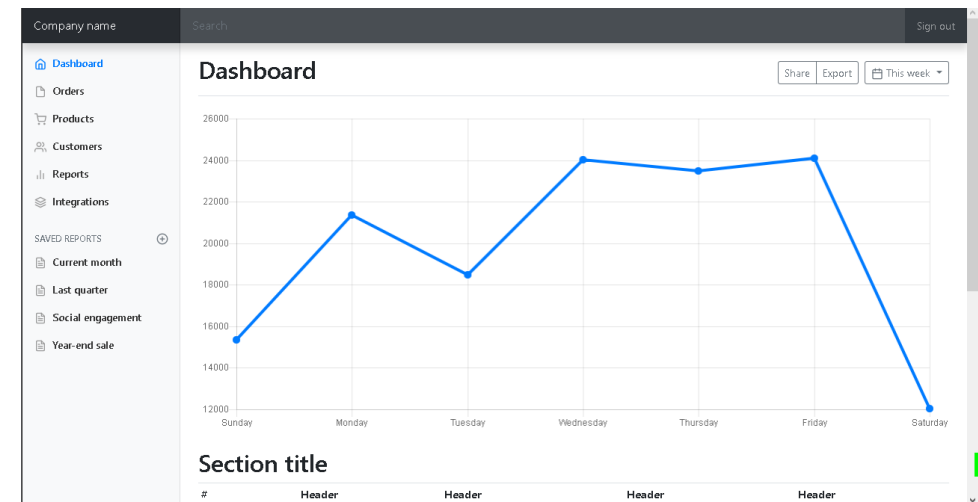
```
<h1>Olá {{usuario}}!!!</h1>  
<p>Bem-vindo à aula de Flask.</p>
```

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Normalmente, uma aplicação Web possui várias páginas diferentes umas das outras. Blocos de código, como cabeçalhos e rodapés serão os mesmos em quase todas as páginas da aplicação. Da mesma forma, o menu também permanece o mesmo. Geralmente, apenas o bloco do contêiner central muda e o resto permanece o mesmo. Para isso, o Jinja2 fornece uma ótima maneira de herança entre os modelos.

É uma boa prática termos um modelo base onde possamos estruturar o layout básico do site junto com o cabeçalho e rodapé.



Flask – Templates com Jinja2

Composição de bloco e herança de layout

Em nosso próximo exemplo vamos criar um pequeno aplicativo onde teremos uma home page e uma página de produtos (similar à que vemos nas lojas de comércio eletrônico).

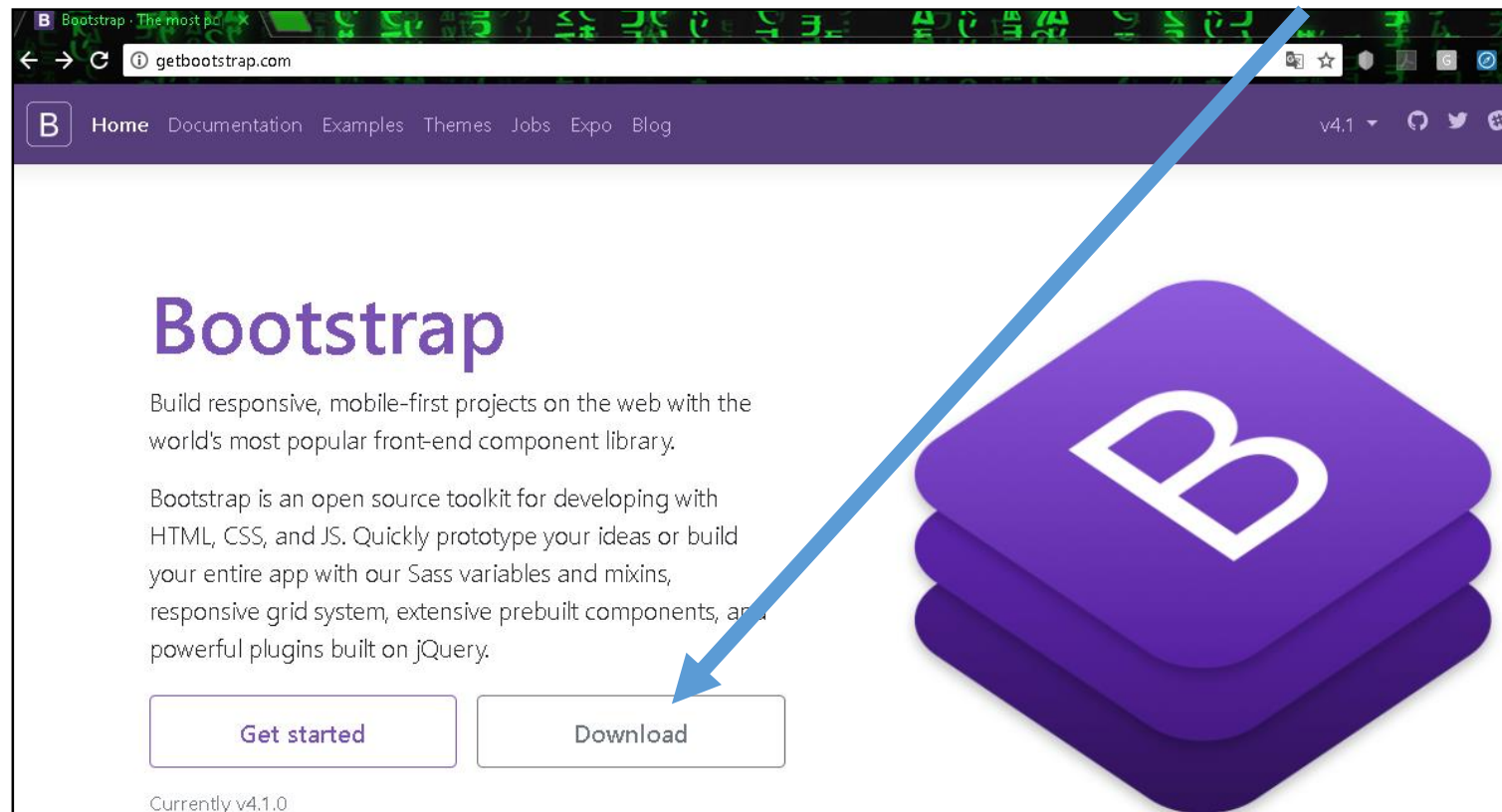
Usaremos o framework Bootstrap para dar um design minimalista aos nossos modelos.

O Bootstrap pode ser baixado em <http://getbootstrap.com>.

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Acesse <http://getbootstrap.com> e clique em Download.



Flask – Templates com Jinja2

Composição de bloco e herança de layout

Na seção “Compiled CSS and JS”, clique em Download.

Será feito download

do arquivo

bootstrap-4.1.0-dist.zip

que é a versão

disponível quando

esta aula foi gravada.

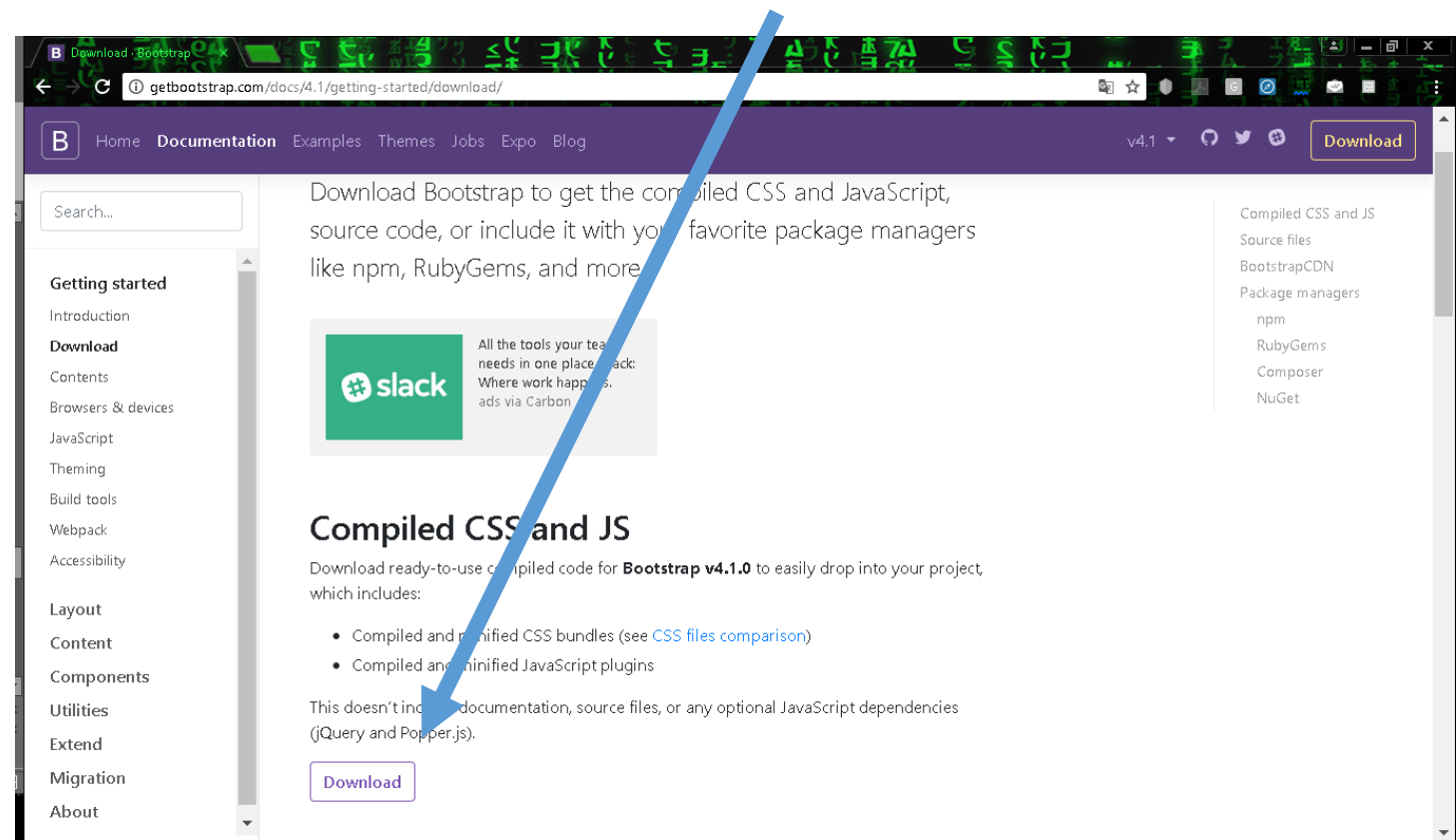
Descompacte e veja que

dentro da pasta

Bootstrap-4.1.0-dist estão

as pastas js e css com arquivos

que usaremos no projeto.



Flask – Templates com Jinja2

Composição de bloco e herança de layout

Em nossa aplicação vamos colocar alguns produtos fixos no arquivo `models.py`, que serão lidos em `views.py` e enviados através do método `render_template()`.

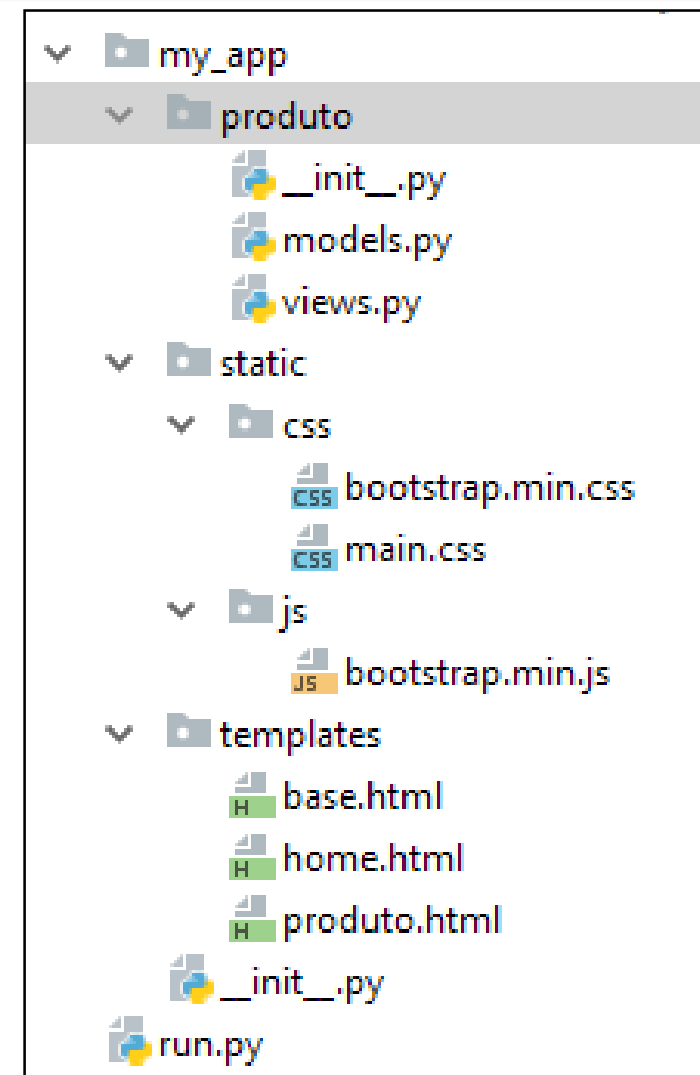
O restante da análise e exibição é manipulado pelo Jinja2.

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Veja a estrutura da aplicação.

Os arquivos static/js/bootstrap.min.js e static/css/bootstrap.min.css são arquivos padrões do Bootstrap localizados no arquivo bootstrap-4.1.0-dist.zip que baixamos e descompactamos.



Flask – Templates com Jinja2

Composição de bloco e herança de layout

Vamos agora trabalhar no armazenamento simples dos produtos. Será um armazenamento de valor-chave sem persistência, ou seja, nesta aula não vamos trabalhar com banco de dados ainda, nossos produtos estarão em uma estrutura fixa.

Os produtos serão definidos no `models.py`

```
PRODUTOS = {
    'celular': {
        'descricao': 'Galaxy S9',
        'categoria': 'Telefones Celulares',
        'valor': '4.299,00'
    },
    'notebook': {
        'descricao': 'Samsung Essentials',
        'categoria': 'Computadores e Notebooks',
        'valor': '1.614,99'
    },
    'tablet': {
        'descricao': 'Lenovo TAB3',
        'categoria': 'Tablets e Ipads',
        'valor': '549,90'
    },
    'ipad': {
        'descricao': 'Ipad',
        'categoria': 'Tablets e Ipads',
        'valor': '2.149,99'
    }
}
```

Flask – Templates com Jinja2


Composição de bloco e herança de layout

Vamos agora criar o arquivo views.py seguindo o estilo Blueprint aprendido nas aulas anteriores.

O nome do blueprint “produto” que é passado no construtor Blueprint será anexado aos endpoints definidos nesse blueprint.

Vamos olhar a seguir o base.html para ficar mais claro.

O método abort() é útil quando desejamos anular uma solicitação com uma mensagem de erro específica. O Flask fornece páginas básicas de mensagens de erro que podem ser personalizadas conforme necessário.



```
from werkzeug import abort
from flask import render_template
from flask import Blueprint
from my_app.produto.models import PRODUTOS

product_blueprint = Blueprint('produto', __name__)

@product_blueprint.route('/')
@product_blueprint.route('/home')
def home():
    return render_template('home.html', produtos=PRODUTOS)

@product_blueprint.route('/produto/<key>')
def produto(key):
    produto = PRODUTOS.get(key)
    if not produto:
        return render_template('erro.html')
    return render_template('produto.html', produto=produto)
```

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Veja agora o conteúdo do arquivo `my_app/__init__.py`:

```
from flask import Flask
from my_app.produto.views import product_blueprint

app = Flask(__name__)
app.register_blueprint(product_blueprint)
```

Além do CSS padrão do Bootstrap teremos nosso CSS personalizado no arquivo `my_app/static/css/main.css`:

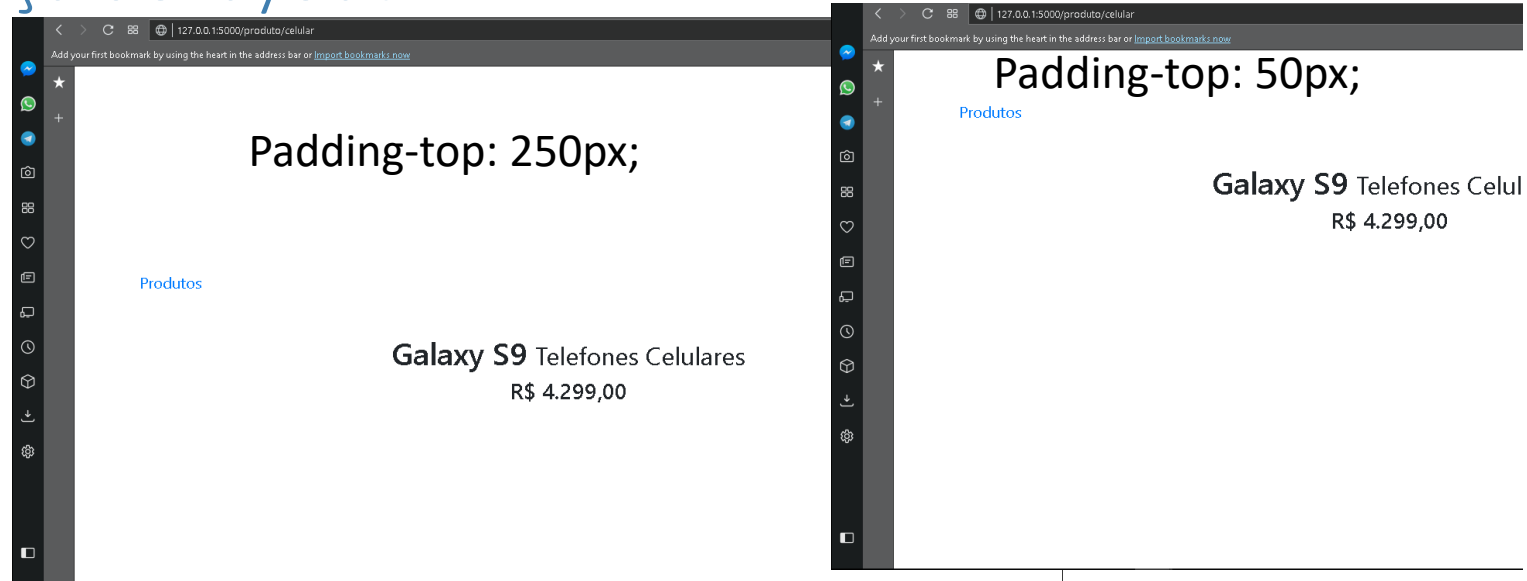
```
body {
    padding-top: 50px;
}
.top-pad {
    padding: 40px 15px;
    text-align: center;
}
```


Flask – Templates com Jinja2

Composição de bloco e herança de layout

Detalhando o CSS

```
body {  
    padding-top: 50px;  
}  
.top-pad {  
    padding: 40px 15px;  
    text-align: center;  
}
```



Veja nas imagens a alteração do padding-top no corpo (body) do documento. Os valores padding e text-align (top-pad) foram utilizados para centralizar os produtos e especificar a distância entre o menu “Produtos” e o texto com a descrição do produto.

O padding é o “preenchimento”, o espaço definido em pixels.

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Agora vamos criar o arquivo `my_app/templates/base.html`, que é um modelo que atua como base para todos os outros modelos.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Catálogo de produtos</title>
    <link href="{{ url_for('static',filename='css/bootstrap.min.css') }}" rel="stylesheet">
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
  </head>
  ...CONTINUA
```

Foi definido o título da página e adicionados os arquivos de estilo CSS do Bootstrap e o nosso `main.css`.

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Continuação do código do arquivo base.html.

```
CONTINUA...
    <body>
        <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="{{ url_for('produto.home') }}">Produtos</a>
                </div>
            </div>
        </div>
    ...CONTINUA
```

Foram criadas as “div” usando classes do Bootstrap para navegação para exibir o menu “Produtos” na página que redireciona para a página home.

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Continuação do código do arquivo base.html.

```
CONTINUA...
    <div class="container">
        {% block container %}{% endblock %}
    </div>

    <!-- jQuery (Plugins JavaScript necessários para Bootstrap) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min.js"></script>
    <script src="{{ url_for('static', filename='js/bootstrap.min.js') }}"></script>
</body>
</html>
```

Foi criada uma div da classe container com um bloco de código container que vai ser substituído pelo arquivo de produtos. Nessa página foram adicionados os arquivos de JavaScript do Bootstrap e JQuery.

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Vamos agora escrever nossa página principal que ficará em `my_app/templates/home.html`. Nesta página vamos iterar por todos os produtos e exibir os mesmos.

```
{% extends 'base.html' %}
{% block container %}
    <div class="top-pad">
        {% for id, produto in produtos.items() %}
            <div class="well">
                <h2>
                    <a href="{{ url_for('produto.produto', key=id) }}">{{ produto['descricao'] }}</a>
                    <small>R$ {{ produto['valor'] }}</small>
                </h2>
            </div>
        {% endfor %}
    </div>
{% endblock %}
```

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Agora vamos criar a página individual do produto que ficará em `my_app/templates/produto.html`.

```
{% extends 'home.html' %}
{% block container %}
    <div class="top-pad">
        <h1>{{ produto['descricao'] }}
            <small>{{ produto['categoria'] }}</small>
        </h1>
        <h3>R$ {{ produto['valor'] }}</h3>
    </div>
{% endblock %}
```

Flask – Templates com Jinja2

Composição de bloco e herança de layout

A maior parte do código anterior é composta de HTML e Jinja2 que já foi explicado em aulas anteriores. Um ponto importante a ser observado é como o método `url_for()` é usado para URLs blueprint. O nome do blueprint é anexado a todos os endpoints. Isso se torna muito útil quando temos vários blueprints dentro de um aplicativo e alguns deles podem ter URLs semelhantes.

```
<a class="navbar-brand" href="{{ url_for('produto.home') }}">Produtos</a>
```

```
<a href="{{ url_for('produto.produto', key=id) }}">{{ produto['descricao'] }}</a>
```

Flask – Templates com Jinja2

Composição de bloco e herança de layout

Vamos agora implementar o run.py.

```
from my_app import app  
  
app.run(debug=True)
```


Flask – Templates com Jinja2

Composição de bloco e herança de layout

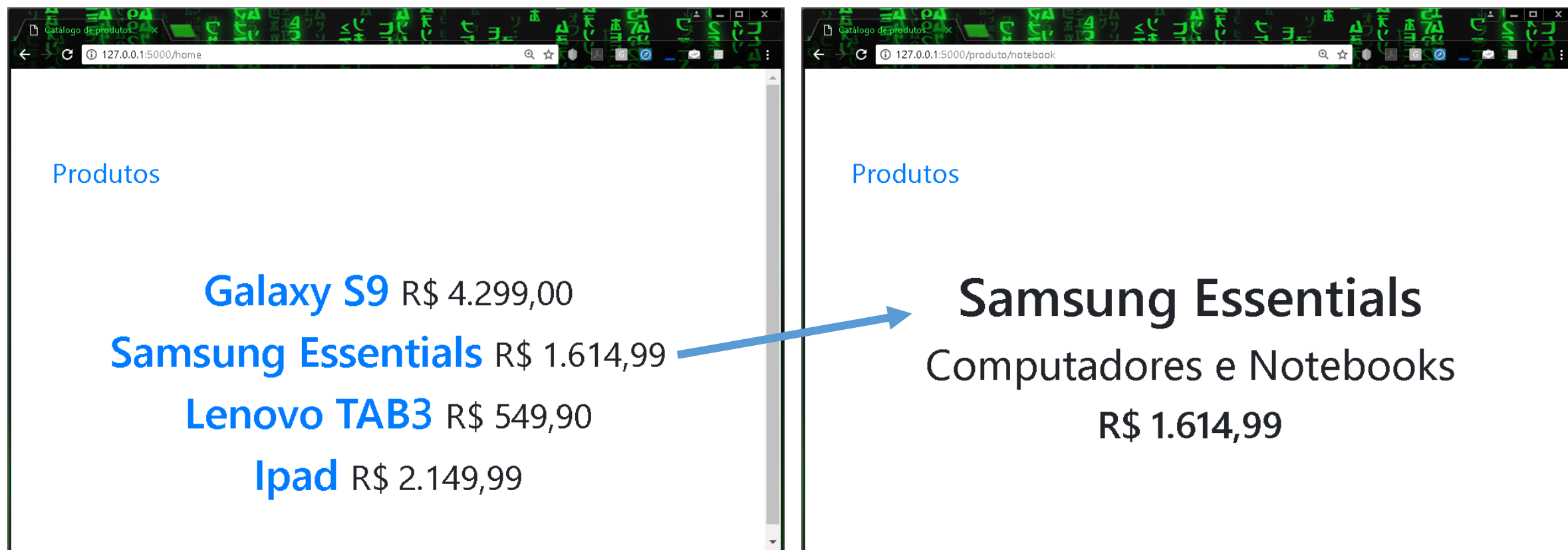
Em nosso projeto seguimos um padrão de herança. O arquivo `base.html` atuou como o modelo base para todos os outros modelos. O arquivo `home.html` foi herdado de `base.html` e `produto.html` herdado de `home.html`.

Em `produto.html`, também sobrescrevemos o bloco do contêiner, que foi preenchido pela primeira vez em `home.html`.

Ao executarmos a aplicação veremos o resultado conforme imagens a seguir.

Flask – Templates com Jinja2

Composição de bloco e herança de layout



CONTINUA...