



CES 22



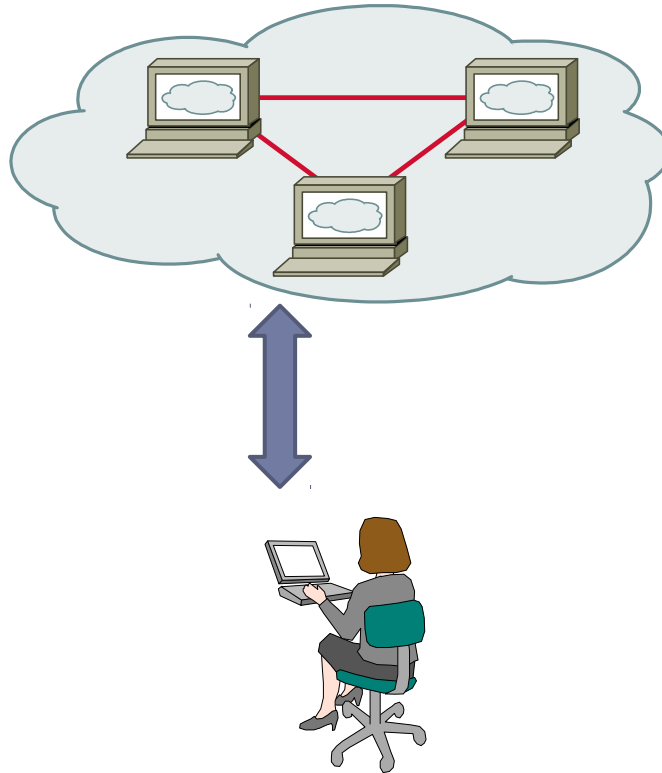
Aula 11 – Desenvolvimento Web com Flask

Objetivos

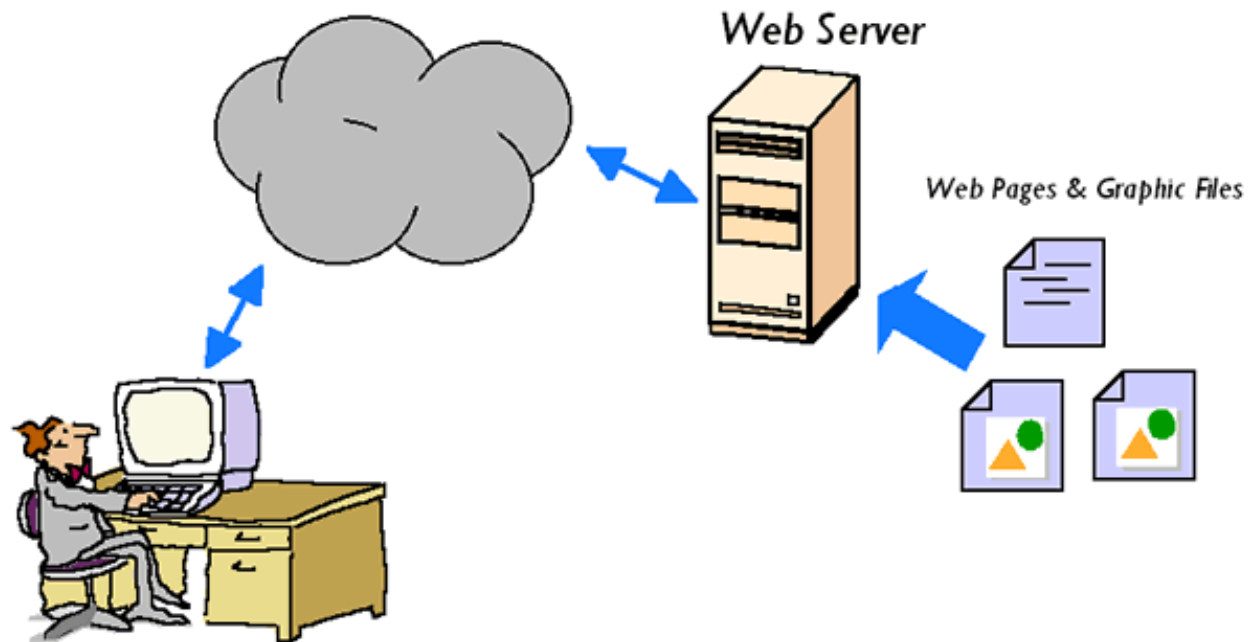
- ▶ Breve introdução para a Web
- ▶ Protocolo HTTP
- ▶ CGI
- ▶ Web Framework
- ▶ Flask



WWW nasceu em 1989 no CERN



Web Server



Elementos Básicos

- ▶ HTML é a linguagem para apresentação de conteúdo entendido pelos navegadores.
- ▶ HTTP é o protocolo de comunicação entre clientes e servidores.
- ▶ URL usado para localizar um servidor e um recurso.

`http://localhost:8080/FirstServletProject/jsps/hello.jsp`

- ▶ **Solicitação HTTP:**

`GET /FirstServletProject/jsps/hello.jsp HTTP/1.1`

`Host: localhost:8080`

`Cache-Control: no-cache`



Resposta do Servidor

200 OK

Date: Wed, 07 Aug 2013 19:55:50 GMT

Server: Apache-Coyote/1.1

Content-Length: 309

Content-Type: text/html; charset=US-ASCII

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Hello</title>
</head>
<body>
<h2>Hi There!</h2>
<br>
<h3>Date=Wed Aug 07 12:57:55 PDT 2013
</h3>
</body>
</html>
```



CGI (Common Gateway Interface)

- ▶ Desde os primeiros servidores, foi percebido a necessidade de criação de páginas dinâmicas.
- ▶ Para tal foi definido uma interface padronizada para a conexão com aplicações que geram páginas dinâmicas.
- ▶ Exemplo de um endereço CGI:
 - ▶ <http://www.rebol.com/cgi-bin/test-cgi.cgi>



Solicitações HTTP

- ▶ GET, usado para acessar dados em um recurso.
- ▶ POST, usado para enviar dados de um formulário HTML.
- ▶ PUT, altera informações de um recurso.
- ▶ DEL, exclui recurso.
- ▶ HEAD, usado para obter informações do cabeçalho da resposta de uma solicitação GET.
- ▶ OPTIONS, recupera os métodos que o servidor aceita.



Frameworks Web

- ▶ Para o desenvolvimento de aplicações dinâmicas no servidor existem vários frameworks de desenvolvimento.
 - ▶ JSP/Servlets Java
 - ▶ NodeJs
 - ▶ Ruby on Rails
 - ▶ Django
 - ▶ Flask



O que faz um framework web?

- ▶ Processa solicitações HTTP.
- ▶ Auxilia na interpretação de uma URL.
- ▶ Auxilia na criação de uma resposta HTML.
- ▶ Auxilia no acesso a serviços extras como banco de dados.



Flask

- ▶ Desenvolvido por Armin Ronacher (programador austríaco nascido em 1989).
- ▶ Baseado em Werkzeug, uma implementação do padrão WSGI (Web server gateway interface), e Jinja2 uma engine de templates para Python.



Hello World

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route('/')  
def hello_world():  
    return 'Hello World'
```

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

Python Hello.py

Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)



-
- ▶ `app.route(rule, options)`
 - ▶ The **rule** parameter represents URL binding with the function.
 - ▶ The **options** is a list of parameters to be forwarded to the underlying Rule object.
 - ▶ In the above example, `‘/’` URL is bound with **hello_world()** function. Hence, when the home page of web server is opened in browser, the output of this function will be rendered.
 - ▶ Finally the **run()** method of Flask class runs the application on the local development server.
-



- ▶ `app.run(host, port, debug, options)`
- ▶ All parameters are optional

Sr.No	Parameters & Description
1	host Hostname to listen on. Defaults to 127.0.0.1 (localhost). Set to '0.0.0.0' to have server available externally
2	port Defaults to 5000
3	debug Defaults to false. If set to true, provides a debug information
4	options To be forwarded to underlying Werkzeug server



```
@app.route('/hello')  
def hello_world():  
    return 'hello world'
```

► **http://localhost:5000/hello**

```
def hello_world():  
    return 'hello world'  
app.add_url_rule('/hello', 'hello', hello_world)
```



Regras variáveis

```
from flask import Flask
app = Flask(__name__)
@app.route('/hello/<name>')
def hello_name(name):
    return 'Hello %s!' % name
if __name__ == '__main__':
    app.run(debug = True)
```

http://localhost:5000/hello/TutorialsPoint

Hello TutorialsPoint!




```
from flask import Flask
app = Flask(__name__)

@app.route('/blog/<int:postID>')
def show_blog(postID):
    return 'Blog Number %d' % postID

@app.route('/rev/<float:revNo>')
def revision(revNo):
    return 'Revision Number %f' % revNo

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

http://localhost:5000/blog/11

Blog Number 11

http://localhost:5000/rev/1.1

Revision Number 1.100000

Construção de URL

```
from flask import Flask, redirect, url_for  
app = Flask(__name__)
```

```
@app.route('/admin')  
def hello_admin():  
    return 'Hello Admin'
```

```
@app.route('/guest/<guest>')  
def hello_guest(guest):  
    return 'Hello %s as Guest' % guest
```

```
@app.route('/user/<name>')  
def hello_user(name):  
    if name == 'admin':  
        return redirect(url_for('hello_admin'))  
    else:  
        return redirect(url_for('hello_guest', guest = name))
```

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



<http://localhost:5000/user/admin>

Hello Admin

<http://localhost:5000/user/mvl>

Hello mvl as Guest

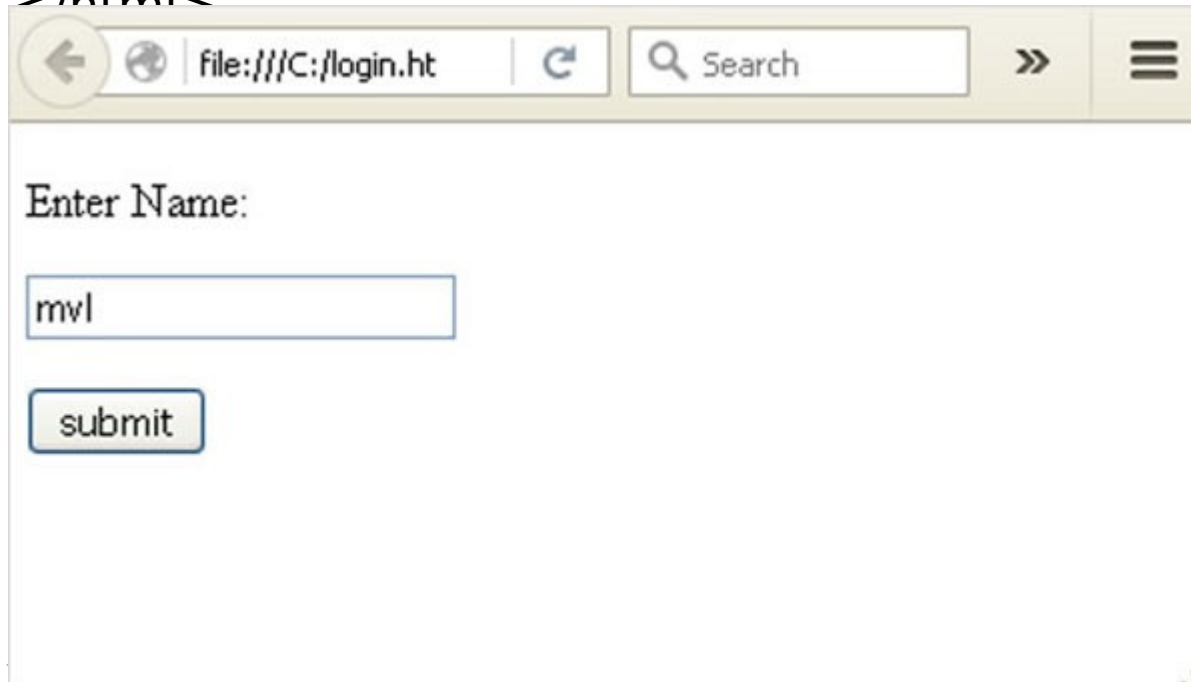


```
<html>  
  <body>
```

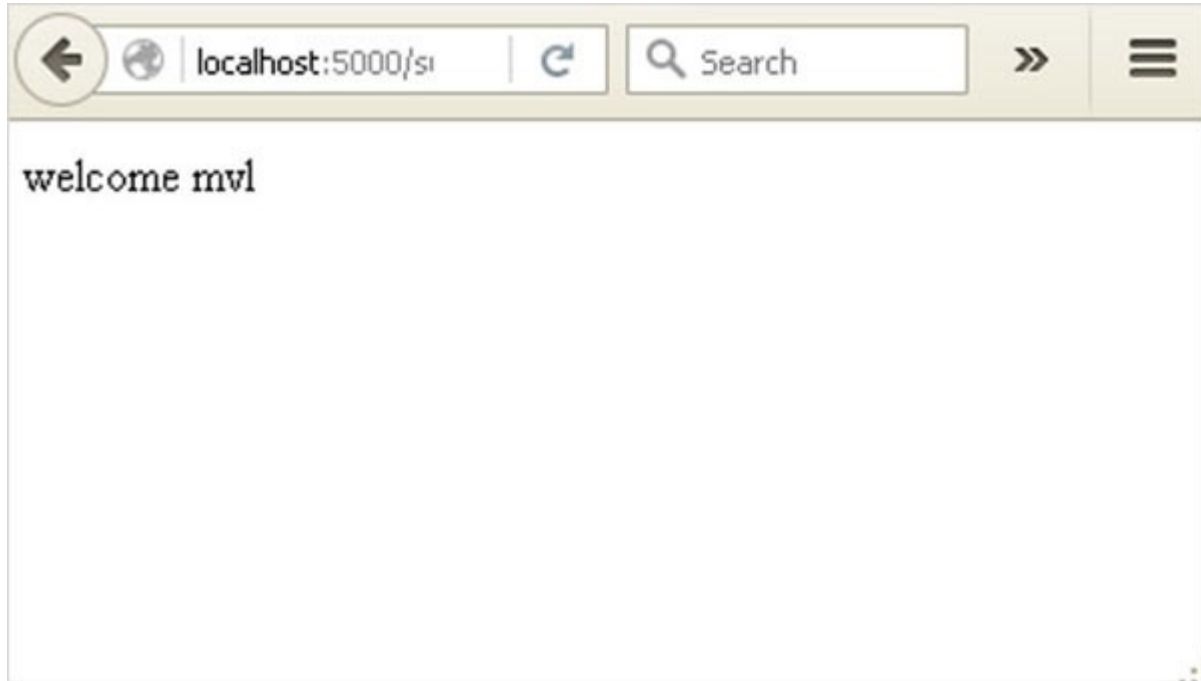
```
    <form action = "http://localhost:5000/login" method =  
    "post">  
      <p>Enter Name:</p>  
      <p><input type = "text" name = "nm" /></p>  
      <p><input type = "submit" value = "submit" /></p>  
    </form>
```

```
</body>
```

```
</html>
```



A screenshot of a web browser window. The address bar shows the file path: file:///C:/login.ht. The page content displays a form with the label "Enter Name:" followed by a text input field containing the text "mvl". Below the input field is a button labeled "submit".



Retornar HTML como resposta

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return '<html><body><h1>'Hello
World'</h1></body></html>'

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



Usar template para montar a resposta

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return
    render_template('hello.html')

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



Pasta para localizar templates

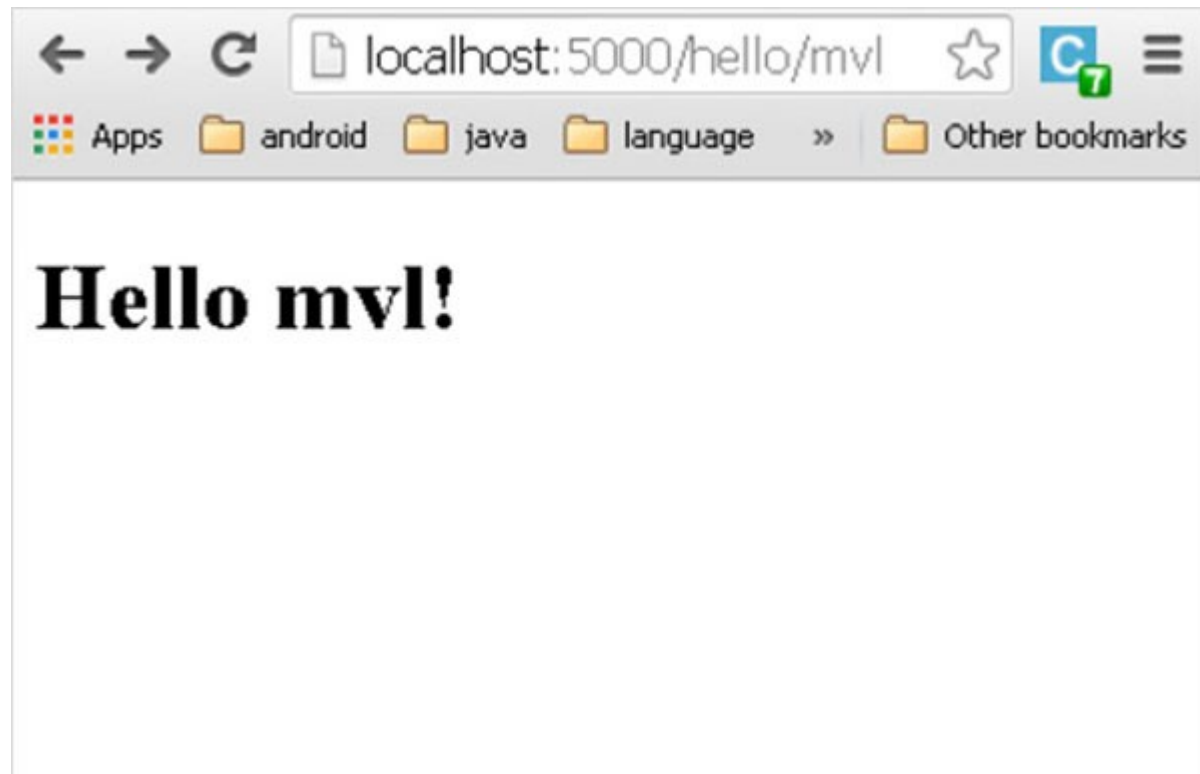
- Application folder
 - Hello.py
 - templates
 - hello.html



Jinja2 template engine

```
<!doctype html>  
<html>  
  <body>  
  
    <h1>Hello {{ name }}!</h1>  
  
  </body>  
</html>
```

```
from flask import Flask, render_template  
app = Flask(__name__)  
  
@app.route('/hello/<user>')  
def hello_name(user):  
    return render_template('hello.html', name =  
user)  
  
if __name__ == '__main__':  
    app.run(debug = True)
```



Projeto 2.o Bimestre

- ▶ Desenvolver uma aplicação usando Flask.
 - ▶ Obrigatoriamente utilizar um banco de dados.

