



<https://docs.cherrypy.dev/en/latest/>

SERVIDORES WEB COM CHERRYPY

Servidor WEB - Introdução

- Um servidor Web é uma aplicação de *software* que comunica um recurso (página HTML, imagem, vídeo, audio) a um cliente (Web Browser) através do protocolo HTTP.
- Um servidor Web é dinâmico se permitir aos utilizadores requisitar serviços como, por exemplo, aceder a dados que não foram previamente carregadas no servidor quando este foi desenvolvido. Ou seja, durante o funcionamento do servidor ele adquire dados novos (ficheiros de imagem, áudio, etc ...).
- Um servidor Web tem persistência (também designado por memória) se conseguir manter a informação sobre os dados adquiridos durante o seu funcionamento, de maneira a ser possível apresentá-las posteriormente. O que implica armazenar a informação relevante sobre os mesmos dados numa base de dados.

Servidor WEB sem Persistência

`public_html`

`html`

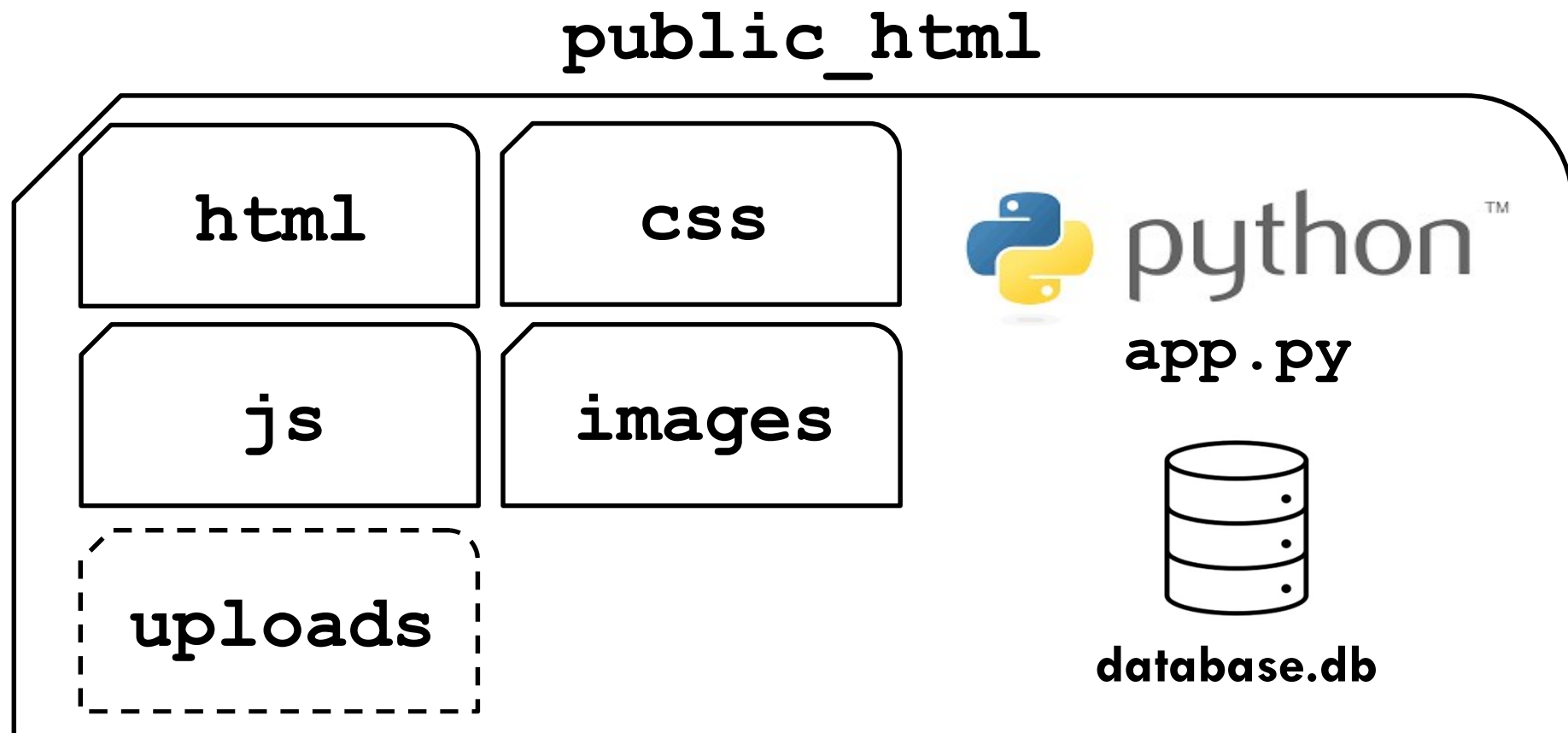
`css`

`js`

`images`

 `python`TM
`app.py`

Servidor WEB com Persistência



Servidor WEB - Organização - 1

- Uma aplicação em Python (**app.py**) para criar, armazenar, consultar os conteúdos pretendidos pelo utilizador. No caso de ter persistência também existe uma base de dados (**database.db**) para armazenar, atualizar e pesquisar informação sobre os dados que foram adquiridos pelo servidor.
- O diretório **html** para as páginas (HTML) que definem a estrutura da aplicação, entre elas a página de arranque que habitualmente se designa por **index.html**.
- O diretório **css** para os ficheiros de estilos que configuram o aspeto das páginas.
- O diretório **js** para os ficheiros de JavaScript que dão dinâmica às páginas.
- O diretório **images** para as imagens estáticas que ilustram as páginas.
- O diretório (por exemplo designado por **uploads**) para armazenar os ficheiros de audio e/ou de imagem adquiridos pelo servidor, caso ele tenha persistência.

Servidor WEB - Organização - 2

- Cada página HTML (**filename.html**) tem normalmente associada um ficheiro de JavaScript (**filename.js**) que gere toda a dinâmica da página.
- A utilização do evento da biblioteca jQuery **\$(document).ready()** evita que qualquer código seja executado antes do documento carregar completamente.
- Serve nomeadamente para:
 - ▣ Obter os argumentos de entrada da página como, por exemplo: passar a identidade do utilizador às páginas de gestão dos dados de modo a limitar o acesso apenas aos utilizadores autorizados; passar o identificador dos dados a que se pretende aceder.
 - ▣ Contruir páginas HTML com dados previamente adquiridos pelo servidor em função da identidade do utilizador.
 - ▣ Criar elementos HTML dinâmicos como, por exemplo gerar um seletor dinâmico a partir de dados armazenados em bases de dados.

Servidor WEB - Diretórios Estáticos

```
# The absolute path to this file's base directory
baseDir = os.path.dirname(os.path.abspath(__file__))

# Dictionary with this application's static directories configuration
config = {
    "/": { "tools.staticdir.root": baseDir },
    "/html": { "tools.staticdir.on": True, "tools.staticdir.dir": "html" },
    "/js": { "tools.staticdir.on": True, "tools.staticdir.dir": "js" },
    "/css": { "tools.staticdir.on": True, "tools.staticdir.dir": "css" },
    "/images": { "tools.staticdir.on": True, "tools.staticdir.dir": "images" },
    "/uploads": { "tools.staticdir.on": True, "tools.staticdir.dir": "uploads" },
}

class Root():
    @cherrypy.expose
    def index(self):
        return open("html/index.html")
```

Biblioteca jQuery

- jQuery é uma biblioteca JavaScript multi-plataforma projetada para simplificar a programação (*scripting*) do lado do cliente de HTML.
- A sintaxe do jQuery foi projetada para tornar mais fácil a navegação nos elementos de um documento.
- Permite nomeadamente:
 - ▣ selecionar elementos do DOM (baseados no seu: nome; id; classe; tipo; atributos; valores de atributos; etc ...)
 - ▣ criar animações
 - ▣ processar eventos
- Ao invés de usar atributos HTML para identificar as funções para manipulação de eventos, o jQuery lida com eventos puramente em JavaScript.
- A sintaxe jQuery foi feita a pensar especialmente na seleção de elementos HTML e na execução de alguma ação sobre eles.

Instruções jQuery - 1

- Seleção de um elemento

- `document.getElementById("idname");`
- `$("#idname")[0];`

- Valor de um elemento

- `document.getElementById("idname").value;`
- `$("#idname").val();`

- Inicializar o *innerHTML* de um elemento

- `$("#idname").html(string_html);`

- Acrescentar ao *innerHTML* de um elemento

- `$("#idname").append(string_html);`

Instruções jQuery - 2

□ Invocar o método HTTP **GET**

- `$.get("cherrypy method name",`
- `{parameters list}, if necessary`
- `function(response) {`
- `processing response;`
- `});`

□ Invocar o método HTTP **POST**

- `$.post("cherrypy method name", . . .);`

□ Processar um evento

- `$("#idname").on("event", function for event);`

□ Criar código HTML com avaliação de variáveis

- `var text = ` html tags mixed with ${variable} `;`