

Exercici 1: Creació i Consum d'una API REST amb PHP

Context:

En aquest exercici, treballareu en la creació d'una API REST que gestionarà una llista de productes d'una botiga online. Aquesta API permetrà afegir, consultar i esborrar productes utilitzant els mètodes HTTP POST, GET i DELETE. A més, desenvolupareu un petit script PHP que consumirà aquesta API per mostrar la llista de productes i permetre l'eliminació o afegit de nous productes.

Objectius:

- Crear una API REST bàsica en PHP que gestioni productes.
- Implementar els mètodes GET, POST i DELETE.
- Consumir aquesta API mitjançant cURL des d'un altre script PHP.

Requisits del Cas Pràctic:

1. Creació de l'API (api.php):

- **GET:** Retorna la llista de productes en format JSON.
- **POST:** Afegeix un nou producte (amb "name" i "price") a la "base de dades" (un fitxer JSON).
- **DELETE:** Elimina un producte pel seu "id".

2. Base de dades simulada: Utilitzeu un fitxer JSON (**products.json**) per emmagatzemar els productes. Exemple inicial:

```
[  
  
  {"id": 1, "name": "Producte 1", "price": 10.99},  
  {"id": 2, "name": "Producte 2", "price": 19.99},  
  {"id": 3, "name": "Producte 3", "price": 5.99}  
]
```

3. Consumir l'API (client.php):

- Llistar els productes utilitzant una petició GET.
- Afegir un nou producte mitjançant una petició POST.
- Permetre eliminar un producte pel seu id utilitzant DELETE.

Resultat esperat:

- Heu de mostrar com es fa la petició i com es processa la resposta.
- La pàgina hauria de mostrar la llista de productes i proporcionar opcions per afegir o eliminar productes.

Exercici 2: Generació Dinàmica de Contingut amb Twig

Context:

En aquest exercici, creareu una petita aplicació web per gestionar una llista de llibres utilitzant Twig com a motor de plantilles. Aquesta aplicació permetrà visualitzar la llista de llibres i afegir nous llibres mitjançant formularis.

Objectius:

- Comprendre i aplicar el motor de plantilles Twig.
- Separar la lògica del servidor de la presentació amb plantilles netes i organitzades.
- Generar contingut dinàmic basat en dades introduïdes per l'usuari.

Requisits del Cas Pràctic:

1. Configuració de Twig:

- Instal·leu Twig utilitzant Composer:
composer require "twig/twig:^3.0"

2. Estructura del projecte:

```
/project
├── /templates
│   ├── base.html.twig
│   ├── book_list.html.twig
│   └── add_book.html.twig
├── index.php
├── add_book.php
└── books.json
```

3. Funcionalitats:

- **index.php**: Llegeix els llibres de **books.json** i els mostra a **book_list.html.twig**.
- **add_book.php**: Formulari per afegir nous llibres (títol, autor, any) utilitzant la plantilla **add_book.html.twig**. Quan l'usuari envia el formulari, el llibre es desa a **books.json** i es redirigeix a la pàgina principal.

Exemple de plantilla (book_list.html.twig):

```
{% extends 'base.html.twig' %}

{% block title %}Llista de Llibres{% endblock %}

{% block content %}
<h1>Llibres Disponibles</h1>
<ul>
  {% for book in books %}
    <li><strong>{{ book.title }}</strong> de {{ book.author }} ({{ book.year }})</li>
  {% else %}

```

```
<li>No hi ha llibres disponibles.</li>
{% endfor %}
</ul>
<a href="add_book.php">Afegir un nou llibre</a>
{% endblock %}
```

Resultat esperat:

- Heu d'entendre com funciona la separació de la lògica de la presentació.
- L'aplicació ha de ser clara, amb plantilles ben estructurades i contingut dinàmic adaptat a les dades introduïdes.