1 Lezione del 18-11-24

1.1 Sviluppo lato server

Il prossimo linguaggio che vedremo, il **PHP** è un linguaggio di scripting **lato server**. Attraverso lo sviluppo lato server, non siamo più limitati a fornire pagine statiche atraverso il nostro web server, ma possiamo *generare* contenuti **dinamici**.

1.1.1 Alcune tecnologie lato server

Vediamo alcune tecnologie per lo sviluppo lato server:

- **JSP** (Java Server Pages): un ambiente che usa il linguaggio di programmazione Java. Veniva spesso usato nei sistemi informatici di grandi aziende (banche, ecc,,.). Ad oggi è stato soppiantato, al meno al di fuori degli ambiti aziendali, da altri framework;
- **Node.js**: un ambiente lato server che permette di usare il JavaScript anche sul lato server. Questo rende Node.js estremamente vantaggioso per gli sviluppatori che vogliono usare lo stesso linguaggio sia lato client che lato server. Inoltre, Node.js è il suo stesso web server, ergo elimina la necessità di altri web server (come ad esempio Apache);
- **PHP**: sta per *Personal Home Page*, o più recentemente *PHP: Hypertext Preprocessor*, è una tecnologia lato server che può essere usata direttamente all'interno dell'HTML. Viene compilato in una rappresentazione intermedia (*opcode*) simile al *bytecode* del Java. Viene usato largamente nei cosiddetti **CMS**, ossia Content Management System, fra cui ricordiamo WordPress.

1.2 Responsabilità del web server

Riassumendo, possiamo dire che il web server deve occuparsi di:

- Gestire connessioni attraverso il protocollo a livello di *application layer* HTTP;
- Rispondere alle richieste di risorse statiche (viste finora, in HTML, JavaScript e CSS) e dinamiche (generate col PHP o altre tecnologie lato server);
- Gestire i permessi all'accesso di determinate risorse;
- Cifrare e comprimere dati;
- Gestire più domini e URL;
- Gestire connessioni a tecnologie databse quali MySQL e MariaDB;
- Gestire cookie e stati;
- Caricare e gestire file generati dagli utenti o altri web server.

1.2.1 Lo stack LAMP

Abbiamo visto che tecnologie lato server vengono create attraverso determinati *software stack*, fra cui ricordiamo il LAMP (XAMPP su Windows), che comprende:

- **L**: il sistema operativo Linux (più probabilmente un sistema operativo GNU/Linux, costituito da software di GNU sul kernel Linux);
- **A**: il web server Apache;
- **M**: il DBMS (*Database Management System*) MariaDB (storicamente MySQL);
- **P**: il linguaggio di scripting PHP (e storicamente il *Perl*, che forma la seconda "P" in XAMPP).

Possiamo vedere Apache come un **intermediario** fra le richieste HTTP e il nostro codice in PHP. Quando il browser invia una richiesta, il web server carica la risorsa corrispondente, esegue l'eventuale codice PHP per generare risorse, ecc... e restituisce la pagina completa. Sui sistemi Linux, Apache è un **daemon** (*servizio* su Windows), cioè un processo che viene eseguito in background sulla macchina server.

Apache implementa le sue funzionalità attraverso diversi **moduli**:

- `mod_auth`: si occupa di autorizzare gli utenti che accedono alla pagina attraverso il protocollo HTTP;
- `mod_rewrite`: si occupa di accorciare gli URL e effettuare redirezioni;
- `mod_ssl`: si occupa della validazione di certificati SSL;
- `mod_php5`: PHP, a sua volta diviso in alcuni moduli:
  - **PHP core**: la libreria base del PH, che include funzionalità di base come gestione di stringhe, array, matematica, ecc...
  - **Extension layer**: agganci esterni per database, protocolli come l'FTP, e formati come l'XML;
  - **Zend Engine**: si occupa della lettura effettiva di un file PHP, la compilazione e l'esecuzione.

1.2.2 Apache e esecuzione parallela

Il server Apache può funzionare in due modalità fondamentali:

- **Multi-process**, anche detta *worker*: un nuovo processo viene creato per ogni richiesta che viene ricevuta;
- **Multi-threaded**, anche detta *preforked*: più processi (solitamente uno per thread, o *flusso di esecuzione*) gestiscono più richieste contemporaneamente.

1.3 PHP

Il PHP, come il JavaScript, ha sintassi *C-like* ("simile al C"). Gli script in PHP hanno estensione `.php`, e hanno l'aspetto di file HTML inframezzati da opportuni **tag PHP**:

```
1 <?php echo "Ciao vecchio!" ?>
```

1.3.1 Commenti

I commenti in PHP possono essere:

- Su linea singola, indicati con `//` o `#`;
- Su più linee, indicati con la classica sintassi `/* ... */`.

1.3.2 Variabili

Le variabili in PHP sono a **tipizzazione dinamica**, e il valore di una variabile può cambiare nel suo tempo di vita (*loose typing*). Per dichiarare (e in seguito accedere a) una variabile bisogna prefiggere al suo identificatore il simbolo `$`. I tipi supportati dal PHP sono:

- **Boolean**: tipi booleani (vero / falso);
- **Integer**: interi con segno;
- **Float**: numeri a virgola mobile;
- **String**: stringhe di caratteri;
- **Array**: collezioni di oggetti;
- **Object**: istanze di classe.

Le variabili **costanti** vengono definite attraverso la funzione `define()`, e di li in poi riferite senza usare il simbolo `$`.

1.3.3 Concatenazione di stringhe

Le stringhe in PHP vengono concatenate attraverso l'operatore punto (`.`). È permesso usare direttamente il nome di variabile di una stringa dentro a un altra stringase si antepone il prefisso `$`:

```
$firstName = "Tizio";
$lastName = "Caio";
// queste istruzioni sono equivalenti
echo "<em>" . $firstName . $secondName . "</em>";
echo "<em> $firstName $secondName </em>";
```

1.3.4 Sintassi alternativa per le strutture di controllo

Il PHP prevede una sintassi alternativa, oltre a quella tipica del C, per le varie strutture di controllo (if, for, ecc...). Si possono infatti usare i due punti al posto della graffa aperta (e quindi omettere la graffa chiusa):

```
<?php if($condizione = "vera") : ?>
  <a href="primo.php">Primo</a>
<?php else : ?>
  <a href="secondo.php">Secondo</a>
<?php endif ;?>
```

1.3.5 Output

L'output in PHP si può fare con la funzione (vista finora) `echo`, o attraverso la `printf`, derivata direttamente dal C. Ricordiamo che in C la `printf` si aspetta un **segnaposto**, in forma:

- `b`: binario;
- `d`: decimale (intero con segno);
- `f`: virgola mobile;
- `o`: ottale;
- `x`: esadecimale;
- `s`: stringa.

con eventuale specifica di precisione (ad esempio, `%.2f` restituisce un numero a virgola mobile con 2 cifre significative).

1.3.6 Inclusione di file

Il PHP permette di includere all'interno di script altri file (quindi altri script) in PHP. Questo si può fare attraverso le funzioni:

- `include`: include un file, e nel caso si verifichi un errore si limita a mostrare un avviso;
- `include_once`: come sopra, ma si assicura di non includere lo stesso file più volte;
- `require` :include un file, e nel caso si verifichi un errore arresta l'esecuzione;
- `require_once`: come sopra, come sopra;

L'inclusione avviene come una copiatura diretta del file dove si trova la direttiva di inclusione, e quindi allo stesso livello di visibilità (di blocco, di funzione, ecc...) in cui si trova la direttiva di inclusione stessa.

1.3.7 Funzioni

Le funzioni in PHP vengono dichiarate sempre attraverso la parola chiave `function`. Gli argomenti di funzione vengono detti in PHP **parametri**. È supportato il passaggio con valori di riferimento, e ancora per valore e per **riferimento**, usando la sintassi (`&`) del C. Si noti che il passaggio di default è comunque per **valore**.

Inoltre, notiamo che le variabili definite all'esterno di funzioni **non** sono visibili all'interno delle funzioni, a meno di non usare la parola chiave `global`:

```
$count = 56;

function func() {
  echo $count; // errore o output vuoto
  echo global $count: // tutto ok
}

echo $count; // tutto ok
```

1.3.8 Array in PHP

Le array in PHP sono effettivamente **mappe ordinate** (*array associativi*), cioè coppie ordinate di **indici** (sostanzialmente *chiavi*) e **valori**).

Possiamo infatti usare la sintassi tradizionale:

```
$days = array("Lun", "Mar", "Mer", "Gio", "Ven", "Sab", "Dom");
// oppure:
$days = ["Lun", "Mar", "Mer", "Gio", "Ven", "Sab", "Dom"];
```

o la sintassi associativa:

```
$days = array(0 => "Lun", 1 => "Mar", 2 => "Mer", 3 => "Gio", 4 => "Ven",
    5 => "Sab", 6 => "Dom");
```

Sono supportati, come in C, *array di array*, cioè **array multidimensionali**.

Per l'iterazione su array si può usare la funzione `count`, che restituisce la dimensione di un array, oppure il costrutto `foreach`, simle al `for(... of ...)` del JavaScript.