Server-Side Script con PHP

I Server-Side Script rappresentano un'evoluzione rispetto agli applicativi CGI che venivano realizzati in linguaggi come C o C++. La difficoltà nel debugging di questi ultimi ha portato alla necessità di sviluppare tecnologie più facili da gestire e testare. PHP è un vero linguaggio di scripting, interpretato al momento dell'esecuzione, mentre JSP (JavaServer Pages) è basato su Java ma non è propriamente un linguaggio di scripting, poiché la pagina JSP viene tradotta in una servlet Java, compilata e quindi eseguita.

PHP, acronimo di Personal Home Page, è un linguaggio interpretato con type checking dinamico, la cui sintassi deriva dal C. Una delle sue caratteristiche distintive è che le variabili non necessitano di essere dichiarate, e sebbene supporti la programmazione a oggetti, questa funzionalità è utilizzata da relativamente pochi sviluppatori.

7.1 - Caratteristiche fondamentali di PHP

Il codice PHP è integrato all'interno di uno scheletro HTML, arricchendo la pagina con parti procedurali necessarie per generare i contenuti dinamici. Queste parti sono delimitate dai marcatori `<?php ... ?>`. Un esempio comune dell'utilizzo di PHP è la lettura dei campi provenienti da form HTML e la loro visualizzazione, come nel caso di una richiesta GET con URL del tipo `http://.../Form_GET.php?nome=Pippo%20Pippone&age=25`.

Nel codice PHP, possiamo verificare se sono stati inviati parametri tramite GET utilizzando `count($_GET)` e accedere ai loro valori attraverso la variabile predefinita `$_GET`. Per visualizzare le informazioni nel browser, possiamo utilizzare frammenti di codice PHP all'interno della struttura HTML. Ad esempio, possiamo elencare tutti i campi inviati tramite la form utilizzando un ciclo `foreach` su `array_keys($_GET)` e poi accedere ai valori con `$_GET[$name]`.

7.2 - Variabili e strutture dati in PHP

PHP utilizza una tipizzazione dinamica, dove le variabili vengono definite automaticamente al primo utilizzo e devono iniziare con il simbolo `$`. Esistono diverse variabili predefinite che consentono di ottenere informazioni sulla richiesta HTTP e sull'ambiente di esecuzione, come `$_GET`, `$_POST` e `$_SERVER`.

Le variabili `$_GET` e `$_POST` sono mappe "chiave/valore" che vengono popolate in base al metodo HTTP utilizzato. Se il metodo è GET, viene predisposta la mappa `$_GET` ; se è POST, viene predisposta la mappa `$_POST` . La variabile non utilizzata esiste comunque, ma rimane vuota. Per accedere ai valori, si utilizza la notazione con parentesi quadre indicando il nome della chiave, ad esempio `$_GET["nome"]` .

Per gestire array e mappe in PHP, possiamo utilizzare funzioni come `array_keys()` per ottenere l'elenco delle chiavi e `count()` per determinare il numero di elementi. Gli elementi di un array sono accessibili tramite la notazione con parentesi quadre, con le posizioni che iniziano da 0. La scansione di un array può essere effettuata sia con un ciclo a contatore (`for`) sia con il ciclo `foreach` .

Per scoprire quale metodo HTTP è stato utilizzato per la chiamata, possiamo accedere alla chiave 'REQUEST_METHOD' della mappa `$_SERVER` e gestire di conseguenza i parametri:

```
$method = $_SERVER['REQUEST_METHOD'];
if($method == 'GET') {
    $params = $_GET;
}
if($method == 'POST') {
    $params = $_POST;
}
```

7.3 - Funzioni e classi in PHP

Le funzioni in PHP vengono definite con la parola chiave `function` . Non è possibile specificare i tipi dei parametri, ma si può specificare il tipo del valore restituito. È anche possibile effettuare il casting del tipo di un'espressione e richiedere il passaggio dei parametri per reference.

Un esempio di definizione di funzione è:

```
function somma_div(&$res, $v1, $v2=1): float {
    $r = $v1 / $v2;
    $res = $r;
    return (float)($v1 + $v2);
}
```

In questa funzione, `&$res` è un parametro passato per reference, `$v2=1` indica un valore di default, `: float` specifica il tipo del valore restituito, e `(float)` effettua un casting del risultato.

Le variabili nelle funzioni sono esclusivamente locali e non hanno accesso alle variabili esterne. Le funzioni possono essere ricorsive, ricevere un numero variabile di parametri, e possono essere passate come parametri.

PHP supporta diversi tipi di variabili: String, Integer, Float, Boolean, Array e Object. Il linguaggio implementa anche la programmazione orientata agli oggetti attraverso il concetto di classe, con un modello ricco che include campi e metodi statici e dinamici, visibilità pubblica e privata, ereditarietà e polimorfismo.

7.4 - Programmazione a oggetti in PHP

La definizione di una classe in PHP include campi privati e pubblici, un costruttore, e metodi. Ad esempio:

```
class Figura {
    private $tipo;
    public $id;

    public function __construct($id, $tipo) {
        $this->id = $id;
        $this->tipo = $tipo;
    }

    public function che_tipo() {
        return $this->tipo;
    }

    public function get_area() {
        return 0;
    }
}
```

L'ereditarietà si implementa con la parola chiave `extends`. Una sottoclasse può richiamare il costruttore della superclasse con `parent::__construct()` e può effettuare l'overriding dei metodi. Ad esempio:

```
class Rettangolo extends Figura {
    private $area;

    function __construct($id, $base, $altezza) {
        parent::__construct($id, "Rettangolo");
        $this->area = (float)$base * (float)$altezza;
    }

    public function get_area() {
        return $this->area;
    }
}
```

PHP supporta anche oggetti dinamici attraverso la classe `stdClass`. Un oggetto di questa classe è inizialmente vuoto ma può essere esteso con nuovi campi "al volo", simile al terzo modo di creare oggetti in JavaScript. Questo permette una grande flessibilità nella manipolazione degli oggetti.

Infine, PHP supporta la gestione delle eccezioni. Per catturarle, si utilizza il blocco `try-catch`. Sono ammessi catch multipli, esistono molte sottoclassi di Exception, è possibile definire proprie eccezioni, e l'istruzione `throw` permette di generare un'eccezione.

7.5 - Connessione a database relazionali con PHP

La connessione a database relazionali è essenziale per le applicazioni web dinamiche, permettendo di implementare un'architettura a tre livelli. I passi fondamentali sono: stabilire la connessione con il database, preparare l'istruzione SQL da eseguire, eseguirla, e, se l'istruzione è una query, scandire il result set.

Per la connessione ai database, PHP utilizza i servizi PDO (PHP Data Objects), una libreria che consente di interfacciarsi con vari tipi di database relazionali in modo generico. Prima dell'introduzione di PDO, era necessario utilizzare librerie specifiche per ogni DBMS.

Per stabilire una connessione, si crea un oggetto PDO specificando i parametri di connessione:

```php
try {
    $conn = new PDO("mysql:host=".$servername.";dbname=".$dbname, $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    echo "DB Error: " . $e->getMessage();
    $error = true;
}
```

L'esecuzione di una query avviene tramite il metodo `query()` dell'oggetto PDO:

```php
$result = $conn->query("SELECT * FROM ARTICOLO");
```

Il result set è paragonabile a un array e può essere scandito con un ciclo `foreach`. Ogni elemento del result set descrive una riga della tabella e si comporta come una mappa, permettendo di accedere agli attributi tramite il loro nome:

```php
foreach($result as $riga) {
    echo $riga["CODICE"];
    echo $riga["DESCRIZIONE"];
    echo $riga["PREZZO"];
}
```

Per chiudere la connessione, è sufficiente distruggere l'oggetto PDO assegnandogli `null`:

```php
$conn = null;
```

7.6 - Gestione di JSON in PHP

PHP offre funzionalità integrate per la gestione di JSON. Un oggetto PHP può essere facilmente serializzato in JSON tramite la funzione `json_encode()`, che restituisce una stringa con la rappresentazione JSON dell'oggetto. Vengono serializzati solo i campi pubblici, mentre i campi privati e i metodi non vengono inclusi. Per cambiare il MIME Type della risposta, si può utilizzare l'header:

```php
header('Content-type: application/json');
```

La deserializzazione di una stringa JSON si effettua con la funzione `json_decode()`, che restituisce un oggetto di tipo `stdClass`. Di conseguenza, non si ritorna alla classe originale, ma si ottiene comunque

un oggetto con cui lavorare.

Per ricevere un documento JSON tramite il metodo POST, è possibile verificare il tipo di contenuto della richiesta e estrarre il corpo:

```
if($_SERVER["CONTENT_TYPE"] == "application/json") {
    $body = file_get_contents('php://input');
    $obj = json_decode($body);
    // ...
}
```

Quando si lavora con chiamate AJAX, può essere utile creare un file di log sul server per tracciare eventuali errori non visualizzabili direttamente. La funzione `error_log()` permette di scrivere su un file di log:

```
error_log(testo, 3, nomefile);
```

PHP è un linguaggio molto vasto che nel tempo ha subito numerose evoluzioni. Questo documento fornisce gli strumenti di base per scrivere server-side script completi, adatti per le moderne applicazioni web, ma rappresenta solo una parte delle funzionalità offerte dal linguaggio.