

## Capitolo 1

# Orientamento e primi passi

- 1.1 **Il ruolo di PHP nel mondo del Web**
- 1.2 **Cos'ha PHP di tanto speciale?**
- 1.3 **PHP in azione**
- 1.4 **Regole fondamentali dei programmi PHP**
- 1.5 **Sommario**

Ci sono tantissimi motivi per scrivere programmi per computer in PHP. Forse si desidera imparare PHP perché è necessario realizzare un piccolo sito Web contenente alcuni elementi interattivi, oppure PHP viene utilizzato nel proprio luogo di lavoro ed è necessario aggiornarsi. Il presente capitolo fornisce il contesto in cui si inserisce PHP nel puzzle della realizzazione di siti Web: ciò che può fare e perché è tanto valido per ciò che fa. Si darà anche una prima occhiata al linguaggio PHP e lo si vedrà in azione.

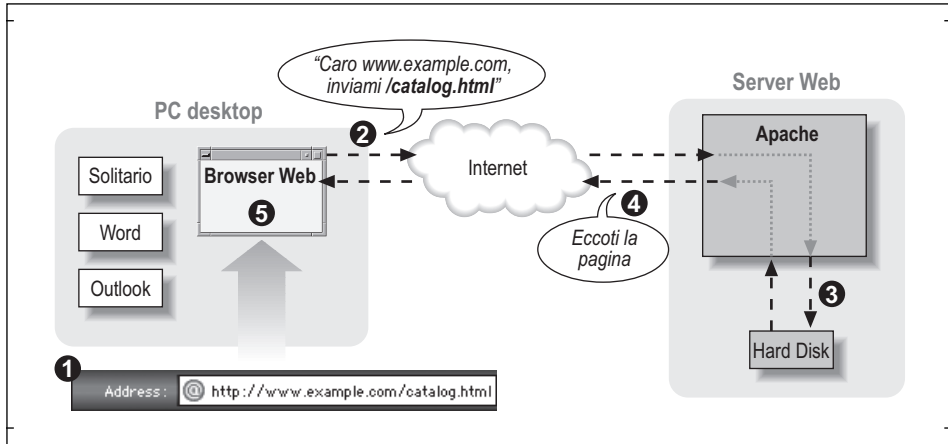
## 1.1 Il ruolo di PHP nel mondo del Web

PHP è un linguaggio di programmazione utilizzato prevalentemente per la realizzazione di siti Web. Invece di essere eseguito su un computer desktop per l'utilizzo da parte di una sola persona, in genere un programma PHP viene eseguito su un server Web e ad esso accedono tantissime persone che utilizzano browser Web sui loro computer. Il presente paragrafo spiega come si inserisce PHP nell'interazione tra un browser Web e un server Web.

Quando ci si siede davanti al computer e si visualizza una pagina Web utilizzando un browser come Internet Explorer o Mozilla, si genera una conversazione su Internet tra il proprio computer e un altro computer. Tale conversazione e il modo in cui viene visualizzata una pagina Web sullo schermo sono illustrati nella Figura 1.1.

Di seguito è spiegato ciò che avviene nei passi numerati del diagramma:

1. Si scrive `www.example.com/catalog.html` nella barra dell'indirizzo di Internet Explorer.
2. Attraverso Internet, Internet Explorer invia un messaggio al computer chiamato `www.example.com`, chiedendo la pagina `/catalog.html`.
3. Apache, un programma in esecuzione sul computer `www.example.com`, riceve il messaggio e legge dall'hard disc il file `catalog.html`.



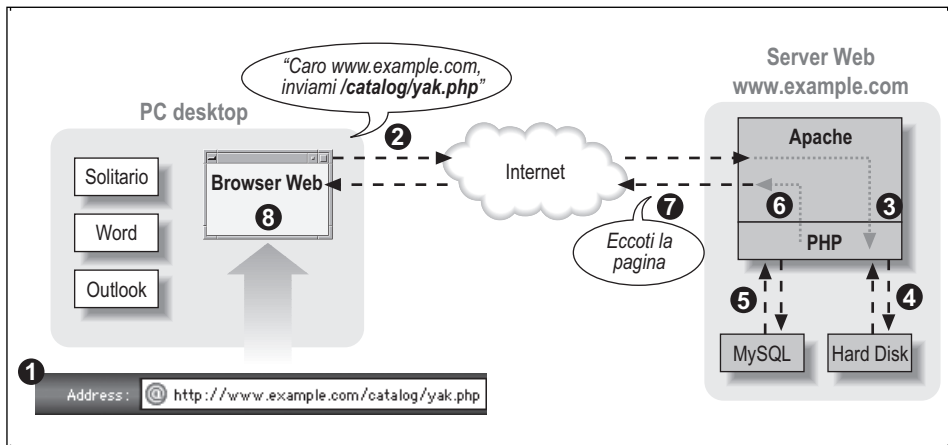
**Figura 1.1** Comunicazione tra client e server senza PHP

4. Come risposta alla richiesta di Internet Explorer, attraverso Internet Apache invia il contenuto del file al computer richiedente.
5. Internet Explorer visualizza la pagina sullo schermo, seguendo le istruzioni dei tag HTML nella pagina.

Ogni qualvolta un browser chiede `http://www.example.com/catalog.html`, il server Web restituisce il contenuto dello stesso file `catalog.html`. La risposta del server Web cambia solo quando viene modificato il file sul server.

Quando è coinvolto PHP, tuttavia, il server effettua più lavoro per la sua parte di conversazione. La Figura 1.2 mostra ciò che accade quando un browser Web chiede una pagina generata da PHP.

Di seguito è spiegato ciò che avviene nei passaggi numerati della conversazione con PHP:



**Figura 1.2** Comunicazione tra client e server con PHP

1. Si scrive `www.example.com/catalog/yak.php` nella barra dell'indirizzo di Internet Explorer.
2. Attraverso Internet, Internet Explorer invia un messaggio al computer chiamato `www.example.com` chiedendo la pagina `/catalog/yak.php`.
3. Apache, un programma in esecuzione sul computer `www.example.com`, riceve il messaggio e chiede all'interprete PHP, un altro programma in esecuzione sul computer `www.example.com`: "Che aspetto ha `/catalog/yak.php`?"
4. L'interprete PHP legge dall'hard disc il file `/usr/local/www/catalog/yak.php`.
5. L'interprete PHP esegue i comandi di `yak.php`, spesso scambiando dati con un programma di database quale MySQL.
6. L'interprete PHP prende l'output del programma `yak.php` e lo restituisce ad Apache come risposta a "Che aspetto ha `/catalog/yak.php`?"
7. Apache invia al computer dell'utente, attraverso Internet, il contenuto della pagina ricevuto dall'interprete PHP, in risposta alla richiesta di Internet Explorer.
8. Internet Explorer visualizza la pagina sullo schermo, seguendo le istruzioni dei tag HTML nella pagina.

"PHP" è un linguaggio di programmazione. Qualcosa nel server Web legge i programmi PHP, che sono istruzioni scritte con questo linguaggio di programmazione, e capisce cosa fare. "L'interprete PHP" segue le istruzioni. Spesso i programmatori dicono "PHP" per intendere il linguaggio di programmazione o l'interprete. Nel presente volume, quando parla di "PHP" l'autore intende il linguaggio di programmazione; quando parla di "interprete PHP" intende ciò esegue i comandi nei programmi PHP e genera le pagine Web.

PHP (il linguaggio di programmazione) sta all'italiano (il linguaggio umano) come l'interprete PHP sta a una persona che parla italiano. La lingua italiana definisce diverse parole e combinazioni che, se lette o ascoltate da una persona che parla italiano, si traducono in diversi significati che fanno in modo che la persona compia azioni sentirsi felice, andare in un negozio a comprare del latte o indossare un maglione. I programmi scritti in PHP (il linguaggio di programmazione) fanno in modo che l'interprete PHP compia azioni come parlare con un database, generare una pagina Web personalizzata o visualizzare un'immagine.

Il presente volume si occupa dei dettagli della scrittura di tali programmi, vale a dire ciò che avviene nel passaggio 5 della Figura 1.2 (l'Appendice A contiene invece dettagli sulla configurazione e l'installazione dell'interprete PHP sul server Web).

PHP è chiamato linguaggio *lato server* perché, come mostra la Figura 1.2, viene eseguito su un server Web. I linguaggi e le tecnologie come JavaScript e Flash, al contrario, sono chiamati *lato client* perché sono eseguiti su un client Web (come un PC desktop). Le istruzioni in un programma PHP fanno in modo che l'interprete PHP su un server Web emetta una pagina Web; le istruzioni in un programma JavaScript fanno in modo che Internet Explorer, mentre è

in esecuzione sul PC desktop, compia azioni come aprire una nuova finestra. Quando il server Web ha inviato al client la pagina Web generata (passaggio 7 nella Figura 1.2), PHP esce di scena. Se il contenuto della pagina contiene codice JavaScript, tale JavaScript viene eseguito sul client ma è completamente scollegato dal programma PHP che ha generato la pagina.

Un normale pagina HTML è come la lettera “siamo spiacenti del fatto che abbia trovato uno scarafaggio nella minestra” che si potrebbe ricevere dopo avere inviato un reclamo a una compagnia aerea infestata dagli scarafaggi. Quando la lettera arriva negli uffici della compagnia aerea, l’indaffarata segretaria del servizio clienti estrae la “lettera di risposta per gli scarafaggi” dall’archivio, ne fa una copia e la invia al cliente che ha reclamato. Tutte le richieste di quel genere ottengono esattamente la stessa risposta.

Al contrario, una pagina dinamica generata da PHP è come una lettera scritta a un amico. Nella pagina si può inserire ciò che si desidera: scarabocchi, disegni, ideogrammi e tenere storie su quanto sia adorabile il proprio bambino quando lancia pappa per tutta la cucina. Il contenuto della lettera è adattato alla persona a cui viene inviata. Quando si inserisce la lettera nella casella della posta, tuttavia, non è più possibile modificarla: viaggia nel mondo e viene letta dall’amico. Non c’è modo di modificare la lettera mentre l’amico la sta leggendo.

Ora, si immagini di inviare una lettera a un amico amante del bricolage. Insieme agli scarabocchi e alle storie si inseriscono istruzioni come “ritaglia la figura della rana nella parte superiore della pagina e incollala sul coniglietto nella parte inferiore della pagina” e “leggi l’ultimo paragrafo della pagina prima di tutti gli altri”. Mentre l’amico legge la lettera, esegue anche le azioni contenute nelle istruzioni della lettera. Tali azioni sono come JavaScript in una pagina Web: vengono impostate quando viene scritta la lettera e non cambiano in un secondo momento. Ma quando il lettore segue le istruzioni, la lettera stessa può cambiare. Analogamente, un browser Web obbedisce ai comandi JavaScript in una pagina e visualizza finestre, cambia le opzioni di menu dei moduli o aggiorna la pagina verso un nuovo URL.

## 1.2 Cos’ha PHP di tanto fantastico?

Si può essere attratti verso PHP perché è gratuito, perché è facile da imparare o perché il proprio capo ha chiesto di cominciare a lavorare su un progetto PHP entro la prossima settimana. Poiché si utilizzerà PHP, è necessario sapere cosa lo rende speciale. Quando qualcuno chiederà “Cos’ha PHP di tanto fantastico?”, utilizzare come risposta questo paragrafo.

### **PHP è gratuito**

Per utilizzare PHP non è necessario pagare nessuno. Che si utilizzi l’interprete PHP su un vecchio PC in cantina o in una stanza piena di server enterprise

da un milione di euro, non esistono costi di licenza, costi di supporto, costi di manutenzione o qualsiasi altro tipo di costo.

La maggior parte delle distribuzioni Linux viene fornita con PHP già installato. Se la propria distribuzione non la possiede, o si utilizza un altro sistema operativo quale Windows, è possibile scaricare PHP da <http://www.php.net/>. L'Appendice A contiene istruzioni dettagliate sull'installazione di PHP.

## **PHP è libero**

Come progetto open source, PHP rende disponibile il suo codice sorgente, che chiunque può esaminare. Se non fa ciò che si desidera, o si è semplicemente curiosi sul funzionamento di caratteristica, è possibile curiosare all'interno dell'interprete PHP (scritto con il linguaggio di programmazione C). Anche se non si ha l'esperienza tecnica per farlo, è possibile assumere qualcuno che effettui le ricerche. La maggior parte delle persone non è in grado di riparare la propria auto, ma è bello poter portare l'auto da un meccanico che può aprire il cofano e ripararla.

## **PHP è indipendente dalla piattaforma**

È possibile utilizzare PHP con un server Web che lavori con Windows, Mac OS X, Linux, Solaris e molte altre versioni di Unix. Inoltre, se si cambia il sistema operativo del server Web, in genere non è necessario modificare i programmi PHP: copiandoli semplicemente dal server Windows al server Unix funzioneranno ancora.

Anche se Apache è il programma per server Web più utilizzato con PHP, è anche possibile utilizzare Microsoft Internet Information Server e qualsiasi altro server Web che supporti lo standard CGI. PHP funziona anche con un grande numero di database, tra cui MySQL, Oracle, Microsoft SQL Server, Sybase e PostgreSQL. Inoltre supporta lo standard ODBC per l'interazione con database.

Se tutti gli acronimi del paragrafo precedente confondono, non ci si deve preoccupare; tutto si riduce a questo: qualsiasi sistema si utilizzi, probabilmente PHP funziona bene con esso e con qualsiasi database si stia già utilizzando.

## **PHP è molto utilizzato**

A marzo del 2004, PHP era installato su oltre 15 milioni di siti Web diversi, da infinite piccole home page personali a giganti come Yahoo!. Esistono molti libri, riviste e siti Web dedicati all'insegnamento di PHP e all'esplorazione di ciò che si può fare con esso. Esistono aziende che forniscono supporto e formazione per PHP. In breve, gli utenti PHP non sono da soli.

## PHP nasconde la sua complessità

Con PHP è possibile realizzare potenti motori di e-commerce che gestiscono milioni di utenti. È anche possibile realizzare un piccolo sito che mantiene automaticamente collegamenti a elenchi variabili di articoli o di comunicati stampa. Quando si utilizza PHP per un progetto più semplice, non si viene ostacolati da questioni importanti solo per un sistema di grandi dimensioni. Quando sono necessarie funzionalità avanzate quali memorizzazione in cache, librerie personalizzate o generazione dinamica di immagini, queste sono disponibili. Se non sono necessarie, non ci si deve preoccupare di esse; è possibile concentrarsi semplicemente sulle basi della gestione di input da parte dell'utente e sulla visualizzazione di output.

## PHP è mirato alla programmazione Web

A differenza di molti altri linguaggi di programmazione, PHP è stato creato fin dall'inizio per la generazione di pagine Web. Questo significa che comuni operazioni di programmazione Web, quali l'accesso all'invio di moduli e la comunicazione con un database, spesso sono più semplici con PHP. PHP ha la capacità di formattare HTML, manipolare date e ore e gestire cookie Web; operazioni che spesso in altri linguaggi di programmazione sono disponibili solo come librerie add-on.

### 1.3 PHP in azione

Il presente paragrafo contiene alcuni listati di programma e spiegazioni sul loro funzionamento. Se non si capisce tutto ciò che avviene in ogni listato, non ci si deve preoccupare: la parte restante del libro serve a questo. Leggere questi listati per avere un'idea dell'aspetto dei programmi PHP e del loro funzionamento, senza preoccuparsi ancora dei dettagli.

Quando riceve un programma da eseguire, l'interprete PHP presta attenzione solo alle parti del programma tra i tag PHP di apertura e di chiusura. Qualsiasi cosa si trovi al di fuori di tali tag viene stampata senza alcuna modifica. Questo facilita l'inserimento di piccole parti di PHP in pagine contenenti prevalentemente HTML. L'interprete PHP esegue i comandi tra `<?php` (il tag PHP di apertura) e `?>` (il tag PHP di chiusura). Generalmente le pagine PHP si trovano in file i cui nomi terminano con `.php`. L'Esempio 1.1 mostra una pagina con un comando PHP.

#### **Esempio 1.1** Hello, World!

```
<html>
<head><title>PHP says hello</title></head>
<body>
```

```
<b>
<?php
print "Hello, World!";
?>
</b>
</body>
</html>
```

L'output dell'Esempio 1.1 è il seguente:

```
<html>
<head><title>PHP says hello</title></head>
<body>
<b>
Hello, World!
</b>
</body>
</html>
```

Nel browser Web, questo ha l'aspetto mostrato nella Figura 1.3.



**Figura 1.3** Salutare con PHP

Tuttavia, la stampa di un messaggio che non cambia mai non è un utilizzo molto interessante di PHP. Si sarebbe potuto inserire il messaggio “Hello, World!” in una normale pagina HTML con lo stesso risultato. Più utile è la stampa di dati dinamici, vale a dire informazioni che cambiano. Una delle fonti più comuni di informazioni per i programmi PHP è l'utente: il browser visualizza un modulo, l'utente inserisce informazioni in esso e preme il pulsante “Invia”, il browser invia tali informazioni al server e infine il server le passa all'interprete PHP, dove diventano disponibili per il programma.

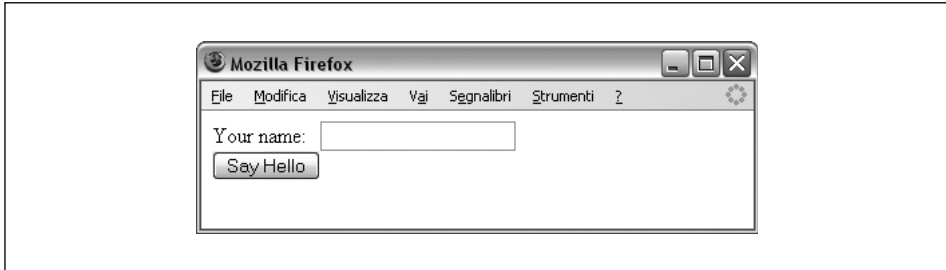
L'Esempio 1.2 è un modulo HTML senza PHP. Il modulo è costituito semplicemente da una casella di testo chiamata user e da un pulsante di invio. Il modulo viene inviato a sayhello.php, specificato con l'attributo action del tag <form>.

**Esempio 1.2** Modulo HTML per l'invio di dati.

```
<form method="POST" action="sayhello.php">
Your Name: <input type="text" name="user">
```

```
<br/>
<input type="submit" value="Say Hello">
</form>
```

Il browser Web visualizza il codice HTML dell'Esempio 1.2 nel modulo mostrato nella Figura 1.4.



**Figura 1.4** Stampa di un modulo.

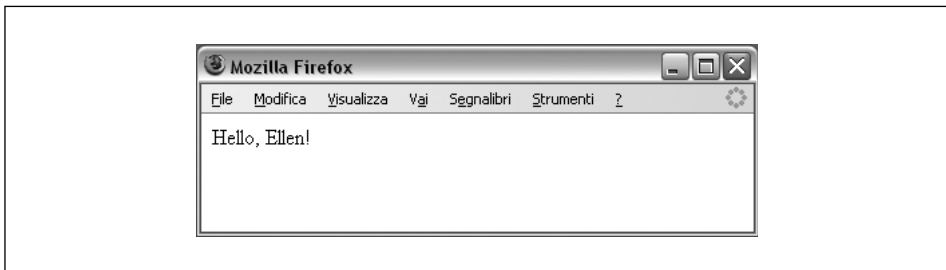
L'Esempio 1.3 mostra il programma `sayhello.php` che stampa un saluto a chiunque sia citato nella casella di testo del modulo.

### **Esempio 1.3** Dati dinamici.

```
<?php
print "Hello, ";
// Stampa quanto inviato nel parametro di modulo chiamato 'user'
print $_POST['user'];
print "!";
?>
```

Se nella casella di testo si scrive Ellen e si invia il modulo, l'Esempio 1.3 stampa Hello, Ellen!. La Figura 1.5 mostra la visualizzazione nel browser Web.

`$_POST` contiene i valori dei parametri di modulo inviati. Nella terminologia di programmazione è una *variabile*, chiamata in questo modo perché è possibile modificare i valori che contiene. In particolare, è una *variabile array*, perché può contenere più valori. Questo particolare array è descritto nel Capitolo 6; gli array sono descritti nel Capitolo 4.



**Figura 1.5** Stampa di un parametro di modulo.

In questo esempio, la riga che comincia con `//` è chiamata *riga di commento*. Le righe di commento esistono per i lettori umani del codice sorgente e vengono ignorate dall'interprete PHP. I commenti sono utili per annotare nei programmi informazioni sul loro funzionamento. Il Paragrafo "Commenti", più avanti in questo capitolo, descrive in dettaglio i commenti.

È anche possibile utilizzare PHP per stampare il modulo HTML che consente a qualcuno di inviare un valore per user, come mostrato nell'Esempio 1.4.

#### **Esempio 1.4** Stampa di un modulo.

```
<?php
print <<< _HTML_
<form method="post" action="$ _SERVER[PHP_SELF]">
Your Name: <input type="text" name="user">
<br/>
<input type="submit" value="Say Hello">
</form>
_HTML_;
?>
```

L'Esempio 1.4 utilizza una sintassi di stringa chiamata *"here document"*. Tutto ciò che si trova tra `<<< _HTML_` e `_HTML_` viene passato al comando `print` per essere visualizzato. Come nell'Esempio 1.3, una variabile all'interno della stringa viene sostituita con il suo valore. Questa volta, la variabile è `$ _SERVER[PHP_SELF]`, una variabile speciale di PHP contenente l'URL (senza il protocollo o il nome di host) della pagina corrente. Se l'URL per la pagina nell'Esempio 1.4 è `http://www.example.com/users/enter.php`, `$ _SERVER[PHP_SELF]` contiene `/users/enter.php`.

Con `$ _SERVER[PHP_SELF]` come azione del modulo, è possibile inserire il codice per la stampa di un modulo e per eseguire azioni con i dati di modulo inseriti nella stessa pagina. L'Esempio 1.5 combina gli Esempi 1.3 e 1.4 in una pagina che visualizza un modulo e stampa un saluto quando viene inviato il modulo.

#### **Esempio 1.5** Stampa di un saluto o di un modulo.

```
<?php
// Stampa un saluto se il modulo e' stato inviato
if ($_POST['user']) {
    print "Hello, ";
    // Stampa quanto inviato nel parametro di modulo chiamato 'user'
    print $_POST['user'];
    print "!";
} else {
    // Altrimenti stampa il modulo
    print <<< _HTML_
<form method="post" action="$ _SERVER[PHP_SELF]">
Your Name: <input type="text" name="user">
<br/>
<input type="submit" value="Say Hello">
```

```
</form>
_HTML_
}
?>
```

L'Esempio 1.5 utilizza il costrutto `if( )` per vedere se il browser ha inviato un valore per il parametro di modulo `user`. Lo utilizza per decidere cosa fare: stampare un saluto o stampare un modulo. Il Capitolo 3 descrive `if( )`. L'utilizzo di `$_SERVER[PHP_SELF]` e l'elaborazione di moduli sono descritti nel Capitolo 6.

PHP ha un'enorme libreria di funzioni interne che possono essere utilizzate nei programmi. Tali funzioni aiutano il programmatore ad effettuare operazioni comuni. Una funzione incorporata è `number_format( )`, che fornisce una versione formattata di un numero. L'Esempio 1.6 utilizza `number_format( )` per stampare un numero.

**Esempio 1.6** Stampa di un numero formattato.

```
<?php print "The population of the US is about:";
print number_format(285266237);
?>
```

L'Esempio 1.6 stampa:

```
The population of the US is about: 285,266,237
```

Il Capitolo 5 tratta le funzioni. Illustra come scrivere le proprie e spiega la sintassi per chiamare funzioni e gestire i risultati di funzioni. Molte funzioni, tra cui `number_format( )`, hanno un *valore restituito*, risultato dell'esecuzione della funzione. Nell'Esempio 1.6 il dato fornito alla seconda dichiarazione `print` per la stampa è il valore restituito da `number_format( )`. In questo caso è il valore della popolazione, formattato con le virgole di separazione per le migliaia.

Un tipo molto comune di programmi scritti in PHP visualizza una pagina Web contenente informazioni recuperate da un database. Quando si consente ai parametri di modulo inviati di controllare ciò che viene estratto dal database, si apre la porta su un universo di interattività sul proprio sito Web. L'Esempio 1.7 mostra un programma PHP che si collega a un server di database, recupera un elenco di portate e i loro prezzi, basati sul valore del parametro di modulo `meal`, e stampa tali portate e prezzi in una tabella HTML.

**Esempio 1.7** Visualizzazione di informazioni da un database.

```
<?php
require 'DB.php';
// Si connette a MySQL eseguito su localhost con nome utente "menu",
// password "good2eaT" e database "dinner"
$db = DB::connect('mysql://menu:good2eaT@localhost/dinner');
// Definisce i pasti ammissibili
$meals = array('breakfast','lunch','dinner');
```

```

// Controlla se il parametro di modulo inviato "meal" e'
// "breakfast", "lunch" o "dinner"
if (in_array($meals, $_POST['meal'])) {
    // In tal caso prende tutte le portate per il pasto specificato
    $q = $db->query("SELECT dish,price FROM meals WHERE meal LIKE '" .
                    $_POST['meal'] ."'");
    // Se non trova portate nel database, dice
    if ($q->numrows == 0) {
        print "No dishes available.";
    } else {
        // Altrimenti stampa ogni portata e il suo prezzo come una riga
        // in una tabella HTML
        print '<table><tr><th>Dish</th><th>Price</th></tr>';
        while ($row = $q->fetchRow( )) {
            print "<tr><td>$row[0]</td><td>$row[1]</td></tr>";
        }
        print "</table>";
    }
} else {
    // Questo messaggio viene stampato se il parametro inviato "meal" non e'
    // "breakfast", "lunch" o "dinner"
    print "Unknown meal.";
}
?>

```

Nell'Esempio 1.7 avvengono molte cose, ma testimone della semplicità e della potenza di PHP è il fatto che per realizzare questa pagina Web dinamica supportata da database occorrono solo 15 righe di codice circa (senza commenti). Di seguito è descritto cosa avviene in tali 15 righe.

La funzione `DB::connect( )` all'inizio dell'esempio imposta la connessione al database MySQL con adeguate informazioni di autenticazione, quali un nome utente e una password. Tali funzioni, come altre funzioni di database utilizzate in questo esempio (`query( )`, `numrows( )` e `fetchRow( )`), sono spiegate in dettaglio nel Capitolo 7.

Gli elementi nel programma che cominciano con un simbolo `$`, come `$db`, `$_POST`, `$q` e `$row` sono variabili. Le variabili contengono valori che possono cambiare mentre viene eseguito il programma o che vengono create in un punto del programma e vengono salvate per un utilizzo successivo. Il Capitolo 2 descrive le variabili.

Dopo avere effettuato la connessione al database, l'operazione successiva è vedere che pasto ha richiesto l'utente. Viene inizializzato l'array `$meals` in modo da contenere i pasti ammessi: `breakfast`, `lunch` e `dinner`. La dichiarazione `in_array($meals, $_POST['meal'])` controlla se il parametro di modulo `meal` inviato (il valore di `$_POST['meal']`) si trova nell'array `$meals`. Se non c'è, l'esecuzione salta alla fine dell'esempio, dopo l'ultimo `else`, e stampa `Unknown meal`.

Se è stato richiesto un pasto accettabile, `query( )` invia una query al database. Per esempio, se il pasto è `breakfast`, la query inviata è come la seguente:

```
SELECT dish,price FROM meals WHERE meal LIKE 'breakfast'
```

Le query a MySQL e a molti altri database sono scritte in un linguaggio chiamato *Structured Query Language* (SQL). Il Capitolo 7 fornisce le basi di SQL. La funzione `query( )` restituisce un identificatore che può essere utilizzato per ottenere ulteriori informazioni sulla query.

La funzione `numrows( )` utilizza tale identificatore per vedere quante portate corrispondenti sono state trovate dalla query nel database. Se non esistono portate applicabili, il programma stampa `No dishes available`, altrimenti stampa informazioni sulle portate corrispondenti.

Il programma stampa l'inizio della tabella HTML, quindi utilizza la funzione `fetchRow( )` per recuperare ogni portata trovata dalla query. La dichiarazione `print` utilizza elementi dell'array restituito da `fetchRow( )` per visualizzare una riga di tabella per portata.

## 1.4 Regole fondamentali dei programmi PHP

Questo paragrafo descrive alcune regole fondamentali sulla struttura dei programmi PHP. Più fondamentali delle basi come “come si stampa qualcosa” o “come si sommano due numeri”, queste proto-basi sono l'equivalente di chi spiega che le pagine di questo libro devono essere lette dall'alto verso il basso e da sinistra verso destra, o che la cosa importante delle pagine sono i segni neri, non le grandi aree bianche.

Se si ha già esperienza con PHP o si è il genere di persona che preferisce giocare con tutti i tasti del nuovo lettore di DVD prima di leggere nel manuale come funzionano, si può passare subito al Capitolo 2 e tornare qui in un secondo momento. Se si corre a scrivere qualche programma PHP per conto proprio, e i programmi si comportano in modo inaspettato o l'interprete PHP si lamenta di “parse errors” quando cerca di eseguire il programma, rivisitare questo paragrafo per ripassare le basi.

### Tag di apertura e di chiusura

Ciascuno degli esempi già visti in questo capitolo utilizza `<?php` come tag PHP di apertura e `?>` come tag PHP di chiusura. L'interprete PHP ignora tutto ciò che si trova al di fuori di tali tag. Il testo prima del tag di apertura o dopo il tag di chiusura viene stampato senza interferenze da parte dell'interprete PHP.

Un programma PHP può avere numerose coppie di tag di apertura e di chiusura, come mostrato nell'Esempio 1.8.

**Esempio 1.8** Tag di apertura e di chiusura multipli.

```
Five plus five is:  
<?php print 5 + 5; ?>
```

```

<p>
Four plus four is:
<?php
    print 4 + 4;
?>
<p>


```

Il codice sorgente PHP all'interno di ogni coppia di tag `<?php ?>` viene elaborato dall'interprete PHP e il resto della pagina viene stampato senza modifiche. L'Esempio 1.8 stampa:

```

Five plus five is:
10<p>
Four plus four is:
8<p>


```

Alcuni programmi PHP più datati utilizzano `<?` come tag di apertura invece di `<?php`. `<?` viene chiamato *tag di apertura breve*, poiché è più breve di `<?php`. In genere è meglio utilizzare il normale tag di apertura `<?php`, poiché il suo funzionamento è certo su qualsiasi server che abbia l'interprete PHP in esecuzione. Il tag breve può essere attivato o disattivato con un'impostazione di configurazione di PHP. L'Appendice A mostra come modificare la configurazione di PHP in modo da controllare quali tag di apertura sono validi nei propri programmi.

Gli esempi restanti nel presente capitolo cominciano tutti con il tag di apertura `<?php` e finiscono con `?>`. In capitoli successivi, non tutti gli esempi hanno tag di apertura e di chiusura, ma si ricordi che, affinché l'interprete PHP riconosca il codice, per i programmi sono necessari.

## Spazi vuoti e sensibilità alle maiuscole

Come tutti i programmi PHP, gli esempi del presente paragrafo sono costituiti da una serie di dichiarazioni, ciascuna delle quali termina con un punto e virgola. È possibile inserire più dichiarazioni PHP sulla stessa riga di un programma, sempre che siano separate da un punto e virgola. È possibile inserire tra dichiarazioni tutte le righe vuote desiderate: l'interprete PHP le ignora. Il punto e virgola indica all'interprete che una dichiarazione è terminata e sta per iniziare un'altra. La mancanza di spazi vuoti o la presenza di molti di essi non influiscono sull'esecuzione del programma (per *spazio vuoto* si intendono caratteri non stampati quali spazi, tabulazioni e interruzioni di riga).

Nella pratica, è buona norma inserire una dichiarazione per riga e righe vuote tra le dichiarazioni solo quando questo migliora la leggibilità del codice sorgente. Gli spazi negli Esempi 1.9 e 1.10 non sono indice di buono stile; è meglio formattare il codice come mostrato nell'Esempio 1.11.

**Esempio 1.9** Questo codice PHP è troppo compresso.

```
<?php print "Hello"; print " World!"; ?>
```

**Esempio 1.10** Questo codice PHP è troppo lungo.

```
<?php  
print "Hello";  
  
print " World!";  
  
?>
```

**Esempio 1.11** Questo codice PHP è ben formattato.

```
<?php  
print "Hello";  
print " World!";  
?>
```

Oltre a ignorare gli spazi vuoti tra righe, l'interprete PHP ignora anche gli spazi vuoti tra parole chiave e valori del linguaggio. Tra `print` e `"Hello, World!"` e tra `"Hello, World!"` e il punto e virgola alla fine della riga è possibile non avere spazi, avere uno spazio o un centinaio di spazi.

Un buono stile di codifica è rappresentato dall'inserimento di uno spazio tra `print` e il valore da stampare, seguito immediatamente da un punto e virgola. L'Esempio 1.12 mostra tre righe, una con troppi spazi, una con pochi spazi e una con la giusta quantità di spazi.

**Esempio 1.12** Spaziature.

```
<?php  
print           "Too many spaces"           ;  
print"Too few spaces";  
print "Just the right amount of spaces";  
?>
```

Le parole chiave del linguaggio (come `print`) e i nomi di funzioni (come `number_format`) non sono sensibili alle maiuscole. L'interprete PHP non si preoccupa del fatto che siano utilizzate lettere maiuscole, minuscole o entrambe quando si inseriscono nei programmi tali parole chiave e nomi di funzioni. Dal punto di vista dell'interprete le dichiarazioni nell'Esempio 1.13 sono identiche.

**Esempio 1.13** Parole chiave e nomi di funzioni non sono sensibili alle maiuscole.

```
// Queste quattro righe fanno la stessa cosa  
print number_format(285266237);  
PRINT Number_Format(285266237);
```

```
Print number_format(285266237);
pRiNt NUMBER_FORMAT(285266237);
```

## Commenti

Come si è visto in alcuni esempi di questo capitolo, i commenti sono un modo per spiegare ad altre persone come funziona il programma. I commenti nel codice sorgente sono una parte essenziale di qualsiasi programma. Quando si scrive il codice, quanto scritto può essere chiarissimo al momento. Tuttavia, qualche mese dopo, quando è necessario modificare il programma, la brillante logica può non risultare tanto evidente. Qui entrano in gioco i commenti: spiegando in linguaggio normale come funziona il programma, i commenti rendono molto più comprensibili i programmi.

I commenti sono ancora più importanti quando la persona che deve modificare il programma non è l'autore originale. Per fare un favore a se stessi e a chiunque possa avere occasione di leggere il codice sorgente, inserire molti commenti nei programmi.

Forse perché sono tanto importanti, PHP fornisce molti modi per inserire commenti nei programmi. Una sintassi già vista è l'inizio della riga con `//`. Questo dice all'interprete PHP di trattare come commento tutto ciò che si trova nella riga. Dopo la fine della riga, il codice viene trattato normalmente. Questo stile di commento viene anche utilizzato in altri linguaggi di programmazione quali C++, JavaScript e Java. È anche possibile inserire `//` su una riga dopo una dichiarazione, per fare in modo che il resto della riga sia trattato come un commento. PHP supporta anche lo stile Perl e di shell dei commenti su riga singola. Queste sono righe che cominciano con `#`. È possibile utilizzare `#` per iniziare un commento nello stesso posto in cui è utilizzabile `//`, ma lo stile moderno preferisce `//` a `#`. Alcuni commenti su riga singola sono mostrati nell'Esempio 1.14.

### Esempio 1.14 Commenti su riga singola con `//` o `#`.

```
// Questa riga e' un commento
print "Smoked Fish Soup ";
print 'costs $3.25.';

# Aggiunge un'altra portata al menu
print 'Duck with Pea Shoots ';
print 'costs $9.50.';
// Si possono inserire // o # in commenti su riga singola
// Anche usando // o # in un altro punto di una riga si avvia un commento
print 'Shark Fin Soup'; // Spero che vada bene!
print 'costs $25.00!'; # Sta diventando costoso!

# Inserendo // o # in una stringa non si crea un commento
print 'http://www.example.com';
print 'http://www.example.com/menu.php#dinner';
```

Per avere un commento su più righe, aprire il commento con `/*` e chiuderlo con `*/`. L'interprete PHP tratta come commento tutto ciò che si trova tra `/*` e `*/`. I commenti su più righe sono utili per disattivare temporaneamente un piccolo blocco di codice. L'Esempio 1.15 mostra alcuni commenti su più righe.

**Esempio 1.15** Commenti su più righe.

```
/* Aggiungeremo alcune cose al menu:
   - Smoked Fish Soup
   - Duck with Pea Shoots
   - Shark Fin Soup
*/
print 'Smoked Fish Soup, Duck with Pea Shoots, Shark Fin Soup ';
print 'Cost: 3.25 + 9.50 + 25.00';

/* Questo e' il vecchio menu:
Le seguenti righe sono all'interno di questo commento quindi non verranno eseguite.
print 'Hamburger, French Fries, Cola ';
print 'Cost: 0.99 + 1.25 + 1.50';
*/
```

In PHP non esiste alcuna regola rigida sullo stile di commento migliore. Spesso i commenti su più righe sono più facili da utilizzare, in particolare quando si desidera commentare un blocco di codice o scrivere alcune righe di descrizione per una funzione. Tuttavia, quando si desidera inserire una breve spiegazione alla fine di una riga, un commento in stile `//` funziona bene. Utilizzare lo stile di commento con cui ci si sente più a proprio agio.

## 1.5 Sommario

Il Capitolo 1 descrive:

- l'utilizzo di PHP da parte di un server Web per creare una risposta o un documento da restituire al browser;
- PHP come linguaggio lato server, vale a dire eseguito sul server Web. Questo al contrario di un linguaggio lato client, come JavaScript;
- cosa si guadagna quando si decide di utilizzare PHP: è gratuito, è libero, è indipendente dalla piattaforma, è popolare ed è progettato per la programmazione Web;
- che aspetto hanno i programmi PHP che stampano informazioni, elaborano moduli e comunicano con un database;
- alcune basi della struttura dei programmi PHP, quali i tag PHP di apertura e di chiusura (`<?php` e `?>`), gli spazi vuoti, la sensibilità alle maiuscole e i commenti.