



PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PHP

Prof. Ada Bagozi
ada.bagozi@unibs.it



Cos'è il PHP?



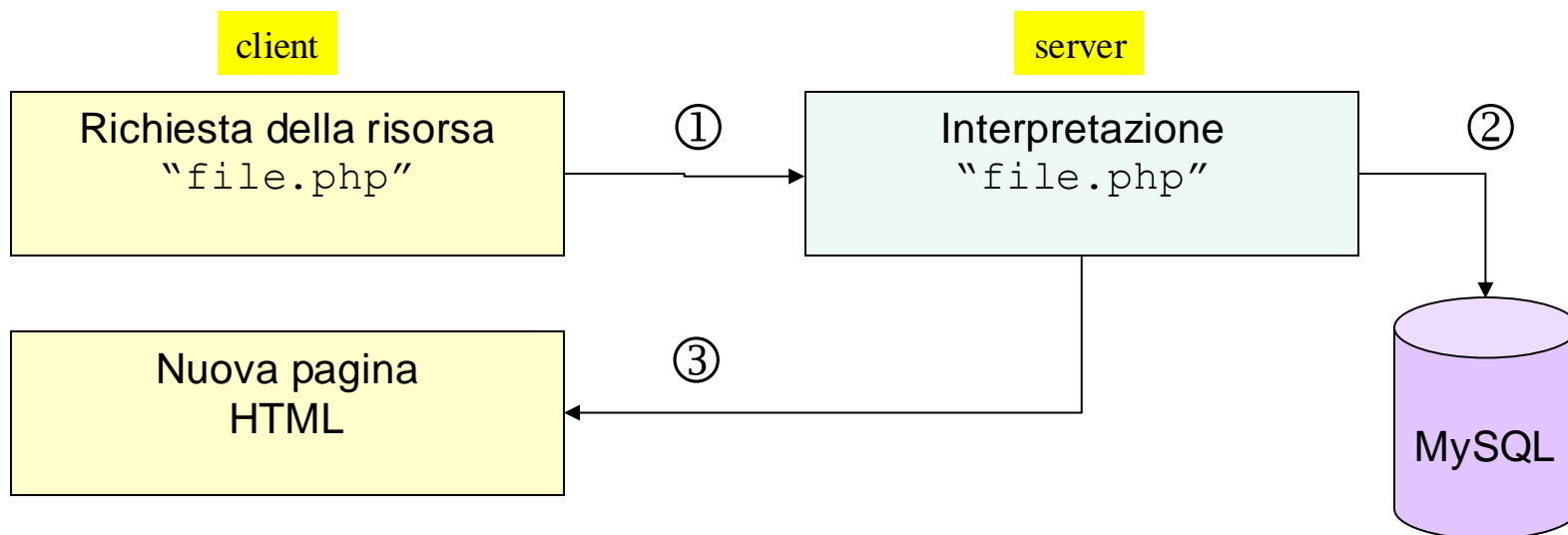
- ✓ *PHP: Hypertext Preprocessor* è un linguaggio di programmazione utilizzato prevalentemente per la programmazione di pagine Web
- ✓ Alcune caratteristiche “tecniche”
 - ✓ è un linguaggio di *scripting*
 - ✓ è *interpretato* (non *compilato*)
 - ✓ è *server-side*
 - ✓ è *debolmente tipizzato*
 - ✓ consente la *programmazione Object Oriented*
- ✓ Ed inoltre.. è *Open Source*



Esecuzione di codice PHP



- ✓ Il codice PHP viene inserito in una pagina HTML (sul server)
- ✓ Il codice PHP viene interpretato (sul server)
- ✓ Viene generata una nuova pagina HTML



Struttura del codice PHP (I)



- ✓ Il codice PHP è racchiuso tra due tag `<?php` e `?>`

```
<?php
    ... codice PHP ...
?>
```

- ✓ Le istruzioni PHP terminano con il punto e virgola

```
echo 'Hello ';
echo 'world!';
```

- ✓ I commenti hanno tre diversi stili

```
/* Autore: Devis Bianchini
   Ultima modifica: 15 aprile 2011
*/
echo '<p>Ordine gestito.</p>'; // Inizio stampa ordine
echo '<p>Ordine gestito.</p>'; # Inizio stampa ordine
```



Variabili



- ✓ Nome di una variabile (identificatore):
 - ✓ Inizia con il simbolo **\$**
 - ✓ Formato da lettere, cifre e underscore '_'
 - ✓ Lunghezza illimitata
 - ✓ Non può iniziare con una cifra
 - ✓ Case sensitive: **\$quantity** \neq **\$Quantity**
- ✓ Una variabile viene creata nel momento in cui viene assegnata la prima volta

```
$quantity = 0;  
$costo = 0.00;  
...  
$quantity = $q;
```

Tipi di variabili



Tipo	Natura del dato
<code>integer</code>	Numeri interi
<code>double</code>	Numeri reali
<code>string</code>	Stringhe di caratteri
<code>boolean</code>	Valori logici (<code>true</code> o <code>false</code>)
<code>array</code>	Vettori di dati

Esistono altri due tipi “speciali”:

- ✓ **NULL**: variabili cui non è assegnato un valore o sono state assegnate con NULL
- ✓ **Resource**: rappresentano risorse esterne (esempio: connessione al database)

Tipizzazione delle variabili in PHP



- ✓ Le variabili sono molto “elastiche” nell’assegnamento del tipo di dati

```
$quantity = 0;  
$quantity = 'Hello';
```

- ✓ Casting

```
$quantity = 0;  
$costo = (double)$quantity;
```

Tipizzazione debole e array (I)



```
<?php
    $v1 = array();

    $v2 = array("lunedì", "martedì", "mercoledì");
    echo $v2[2] // stampa mercoledì, ossia l'elemento in posizione 2

    $v3 = array(
        0=>"lunedì",
        1=>"martedì",
        2=>"mercoledì");
    // $v3 ha lo stesso contenuto di $v2

    $v4 = array(
        "mela"=>"frutta",
        "pera"=>"frutta",
        "carciofo"=>"verdura");
    // $v4 è un array associativo
    echo $v4["pera"] // stampa frutta
?>
```


Tipizzazione debole e array (II)



```
<?php
    $v5 = array(
        3=>"giovedì",
        4=>"venerdì",
        5=>"sabato");
    // array in cui il primo indice è 3
    $v5[4] = "venerdì"; // modifica un valore
    $v5[] = "domenica"; // aggiunge un elemento in coda
    $v5[1] = "martedì"; // aggiunge un elemento in posizione 1

    // Un vettore può avere altri vettori come elementi
    // Inoltre il tipo degli elementi può essere eterogeneo
    $automobili = array(
        "Paperone">array(1,213),
        "Paperino">313,
        "Qui">"nessuna");

    echo count($automobili); // stampa la lunghezza dell'array, cioè 3
?>
```

Variabili predefinite



PHP mette a disposizione un gran numero di variabili predefinite

Sono principalmente dedicate a descrivere il server su cui è in funzione, le richieste HTTP, i dati inseriti nei campi di una form

Alcune variabili predefinite possono essere dipendenti dalla piattaforma

```
<?php
echo "<a href=\"http://\",
    $_SERVER["HTTP_HOST"],      // nome del sito
    $_SERVER["PHP_SELF"],      // nome dello script
    "\">Link a me stesso</a>\n";
?>
```



Lavorare con le variabili



È possibile inserire direttamente una variabile semplice in una stringa

```
echo "$qbanane banane <br />";
```

 \Rightarrow 2 banane

Tuttavia, le stringhe tra apici semplici non applicano alcuna sostituzione:

```
echo '$qbanane banane <br />';
```

 \Rightarrow \$qbanane banane

Costanti



Mantengono un valore (come una variabile), che però non può essere cambiato

```
define (<nome-costante>, <valore>)
```

```
define (PERCENTUALE_SCONTO, 25);
```

Operatori (I)



Operatori **aritmetici**

Simbolo	Nome	Esempio
+	Addizione	<u>\$a</u> + <u>\$b</u>
-	Sottrazione	<u>\$a</u> - <u>\$b</u>
*	Moltiplicazione	<u>\$a</u> * <u>\$b</u>
/	Divisione	<u>\$a</u> / <u>\$b</u>
%	Modulo	<u>\$a</u> % <u>\$b</u>

Operatore di **concatenazione delle stringhe**

```
$a = 'Strumenti Musicali ' ;  
$b = 'in Franciacorta';  
$titolo = $a . $b;
```

Operatori (II)



Operatore di **assegnamento**

<variabile> = <espressione>

`$b = 6 + ($a = 5);`

Operatori di **assegnamento combinato**

Simbolo	Esempio	Equivalente a
<code>+=</code>	<code><u>\$a</u> += <u>\$b</u></code>	<code><u>\$a</u> = <u>\$a</u> + <u>\$b</u></code>
<code>-=</code>	<code><u>\$a</u> -= <u>\$b</u></code>	<code><u>\$a</u> = <u>\$a</u> - <u>\$b</u></code>
<code>*=</code>	<code><u>\$a</u> *= <u>\$b</u></code>	<code><u>\$a</u> = <u>\$a</u> * <u>\$b</u></code>
<code>/=</code>	<code><u>\$a</u> /= <u>\$b</u></code>	<code><u>\$a</u> = <u>\$a</u> / <u>\$b</u></code>
<code>%=</code>	<code><u>\$a</u> %= <u>\$b</u></code>	<code><u>\$a</u> = <u>\$a</u> % <u>\$b</u></code>
<code>.=</code>	<code><u>\$a</u> .= <u>\$b</u></code>	<code><u>\$a</u> = <u>\$a</u> . <u>\$b</u></code>

Operatori (III)



Operatore di **incremento** (++) e **decremento** (--)

Sono entrambi disponibili in due forme

✓ Prima della variabile `$a = 5;`
`echo ++$a;`

prima **\$a** viene
incrementato (6) e poi
visualizzato (6)

✓ Dopo la variabile `$a = 5;`
`echo $a++;`

prima **\$a** viene
visualizzato (5) e poi
Invincrementato (6)

```
$a = 3;  
$b = 6;  
echo ($a--) * (++$b);
```



```
$a: 2  
$b: 7  
echo: 21
```

Operatori (IV)



Operatore di **confronto**

Simbolo	Nome	Esempio
==	Uguaglianza	<u>\$a</u> == <u>\$b</u>
===	Identità	<u>\$a</u> === <u>\$b</u>
!=	Disuguaglianza	<u>\$a</u> != <u>\$b</u>
!==	Non identità	<u>\$a</u> !== <u>\$b</u>
<	Minore di	<u>\$a</u> < <u>\$b</u>
>	Maggiore di	<u>\$a</u> < <u>\$b</u>
<=	Minore o uguale	<u>\$a</u> <= <u>\$b</u>
>=	Maggiore o uguale	<u>\$a</u> >= <u>\$b</u>

Operatore identità: operandi **uguali e dello stesso tipo**

```
$a = 0;  
$b = '0' ;
```



```
$a == $b: true  
$a === $b: false
```


Operatori (V)



Operatori logici

Simbolo	Nome	Esempio	Risultato															
!	NOT	!\$a	<table><tr><td>\$a</td><td>!\$a</td></tr><tr><td>true</td><td>false</td></tr><tr><td>false</td><td>true</td></tr></table>	\$a	!\$a	true	false	false	true									
\$a	!\$a																	
true	false																	
false	true																	
&&	AND	\$a && \$b	<table><tr><td>\$a</td><td>\$b</td><td>\$a && \$b</td></tr><tr><td>true</td><td>true</td><td>true</td></tr><tr><td>true</td><td>false</td><td>false</td></tr><tr><td>false</td><td>true</td><td>false</td></tr><tr><td>false</td><td>false</td><td>false</td></tr></table>	\$a	\$b	\$a && \$b	true	true	true	true	false	false	false	true	false	false	false	false
\$a	\$b	\$a && \$b																
true	true	true																
true	false	false																
false	true	false																
false	false	false																
	OR	\$a \$b	<table><tr><td>\$a</td><td>\$b</td><td>\$a \$b</td></tr><tr><td>true</td><td>true</td><td>true</td></tr><tr><td>true</td><td>false</td><td>true</td></tr><tr><td>false</td><td>true</td><td>true</td></tr><tr><td>false</td><td>false</td><td>false</td></tr></table>	\$a	\$b	\$a \$b	true	true	true	true	false	true	false	true	true	false	false	false
\$a	\$b	\$a \$b																
true	true	true																
true	false	true																
false	true	true																
false	false	false																

Operatori (VI)



Operatore **condizionale** (ternario)

```
( <condizione> ? <valore se vera> : <valore se falsa> )
```

```
echo ($voto >= 18 ? 'promosso' : 'bocciato')
```

```
$max = ($a >= $b ? $a : $b)
```

```
$max3 = ($a >= $b && $a >= $c ?  
          $a : ($b >= $c ? $b : $c))
```

```
echo 'Hai ordinato `.$qbanane.` banan`  
      .($qbanane == 1 ? `a` : `e`)
```



Type juggling



Alcune conversioni di tipo avvengono automaticamente in base agli operatori utilizzati nelle espressioni

La conversione non modifica il tipo degli operandi, che rimangono inalterati

```
<?php  
$a = "0";  
  
$a .= 2;  
  
$a += 2;  
  
$b = $a + 1.3;  
  
?>
```

Funzioni di variabili



Per testare o modificare lo stato di una variabile si possono usare le seguenti funzioni:

- ✓ **isset(\$var): true** se **\$var** esiste, altrimenti **false**
- ✓ **isset(\$a, \$b, \$c, ...): true** se tutte esistono, altrimenti **false**
- ✓ **unset(\$var):** elimina **\$var**
- ✓ **empty(\$var): true** se **\$var** non esiste o ha valore zero, altrimenti **false**

```
$qbanane = 3;  
echo 'echo1: ' . isset($qbanane) . '<br />'; // TRUE  
echo 'echo2: ' . isset($qmele) . '<br />'; // FALSE  
echo 'echo3: ' . empty($qbanane) . '<br />'; // FALSE  
echo 'echo4: ' . empty($qmele) . '<br />'; // TRUE
```

Funzioni di accesso al tipo



- ✓ **is_boolean(\$a)**: verifica se la variabile contiene un valore booleano
- ✓ **is_integer(\$a)**: verifica se la variabile contiene un numero intero
- ✓ **is_float(\$a), is_double(\$a), is_array(\$a), is_resource(\$a)**
- ✓ **is_null(\$a)**: verifica se la variabile contiene il valore **null**
- ✓ **is_numeric(\$a)**: verifica se il contenuto della variabile è compatibile con un valore numerico (ossia se è un numero o una stringa convertibile in numero)
- ✓ **gettype(\$a)**: restituisce il nome del tipo della variabile sotto forma di stringa

L'istruzione if



```
if (<condizione>
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se <condizione> è vera

```
if ($quantita_totale == 0)
    echo 'Non hai ordinato alcun articolo! <br />';
```

```
if ($costo_totale >= 30000)
{
    echo 'Hai diritto ad uno sconto del 10% <br />';
}
```

L'istruzione else



```
else  
<istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se la condizione del precedente **if** è falsa

```
if($quantita_totale == 0)  
    echo "Non hai ordinato alcun articolo! <br />";  
else  
{  
    echo 'Ecco la lista degli articoli: <br />';  
    echo '<ul>';  
    echo "<li>$qpomodori pomodori</li>";  
    echo "<li>$qbanane banane</li>";  
    echo "<li>$qmele mele</li>";  
    echo "</ul>";  
}
```

L'istruzione elseif



È usato quando si hanno più di due rami decisionali:

```
elseif <condizione>
    <istruzioni>
```

```
if($costo_totale <= 10000)
    $sconto = 0.0;
elseif($costo_totale > 10000 && $costo_totale <= 20000)
    $sconto = 0.10;
elseif($costo_totale > 20000 && $costo_totale <= 40000)
    $sconto = 0.20;
elseif($costo_totale > 40000 && $costo_totale <= 80000)
    $sconto = 0.25;
else
    $sconto = 0.30;
```

Viene eseguito solo il blocco di istruzioni corrispondente alla prima condizione vera

Se nessuna condizione vera, viene eseguito il blocco della **else** (se specificata)



L'istruzione switch



```
switch (<condizione>) {  
  case <valore1>: <codice> break;  
  case <valore2>: <codice> break;  
  ....  
  default: <codice>; break;  
}
```

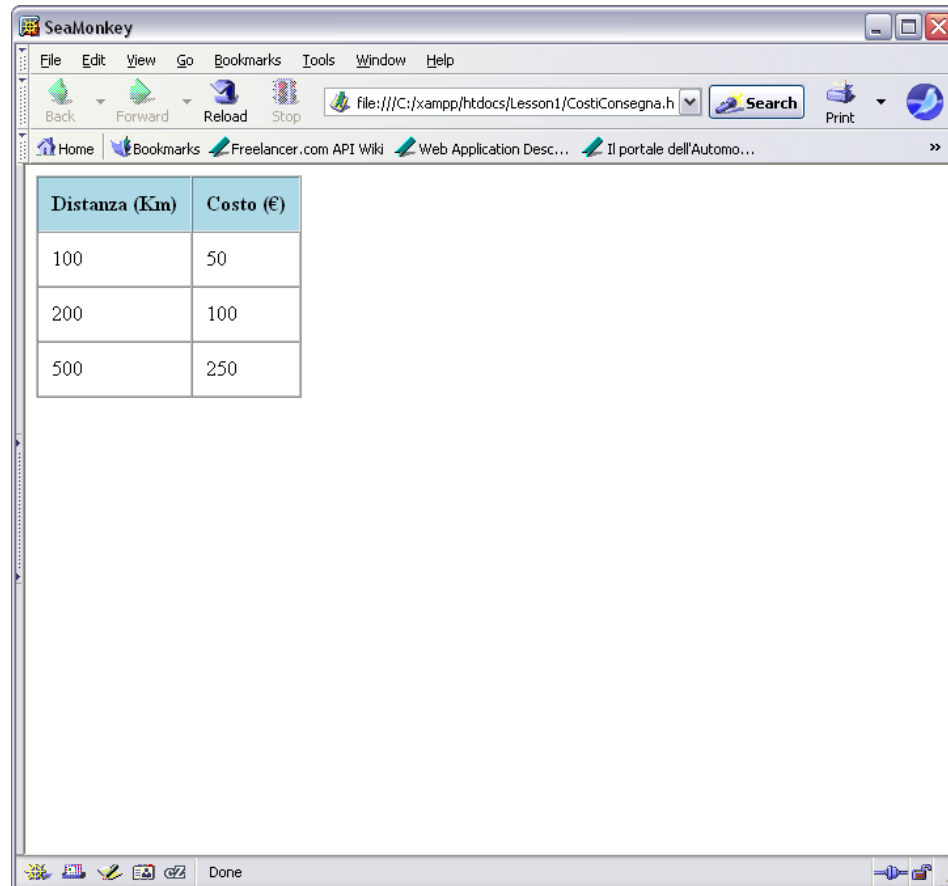
Il costrutto **switch** permette di diversificare l'operato del programma al valore dell'espressione testata

```
switch ($nome)  
{  
    case 'Luca':  
    case 'Giorgio':  
    case 'Franco':  
        echo "Ciao, vecchio amico!"; break;  
    case 'Mario': echo "Ciao, Mario!"; break;  
    default: print "Benvenuto, chiunque tu sia";  
}
```

Iterazioni



Sono utilizzate quando è necessario eseguire più volte lo stesso blocco di istruzioni



The screenshot shows a SeaMonkey web browser window. The address bar displays a local file path: `file:///C:/xampp/htdocs/Lesson1/CostiConsegna.h`. The browser's content area displays a table with two columns: "Distanza (Km)" and "Costo (€)". The table contains three rows of data: (100, 50), (200, 100), and (500, 250).

Distanza (Km)	Costo (€)
100	50
200	100
500	250

Iterazioni



Sono utilizzate quando è necessario eseguire più volte lo stesso blocco di istruzioni

The screenshot shows the SeaMonkey web browser window. The address bar displays a local file path: `file:///C:/xampp/htdocs/Lesson1/CostiConsegna.h`. The browser's content area displays an HTML table with two columns: "Distanza (Km)" and "Costo (€)". The table contains three data rows: (100, 50), (200, 100), and (500, 250). To the right of the table, the corresponding HTML code is shown, illustrating the use of iteration to generate the table structure.

Distanza (Km)	Costo (€)
100	50
200	100
500	250

```
<table border="1" cellspacing="0" cellpadding="10">
  <thead style="background-color:lightblue">
    <tr>
      <th>Distanza (Km)</th>
      <th>Costo (<euro;></th>
    </tr>
  </thead>
  <tr>
    <td>100</td>
    <td>50</td>
  </tr>
  <tr>
    <td>200</td>
    <td>100</td>
  </tr>
  ...
  <tr>
    <td>500</td>
    <td>250</td>
  </tr>
</table>
```

Ciclo while



```
while (<condizione>
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito fino a quando <condizione> è vera

```
<table border="1" cellspacing="0" cellpadding="10">
  <thead style="background-color:lightblue">
    <tr>
      <th>Distanza (Km)</th>
      <th>Costo (&euro;)</th>
    </tr>
  </thead>
  <?php
    $distanza = 100;
    while($distanza <= 500)
    {
      echo '<tr><td>' . $distanza . '</td>';
      echo '<td>' . ($distanza/2) . '</td></tr>';
      $distanza += 100;
    }
  ?>
</table>
```

Ciclo for



```
for (<istruzione1>; <condizione>; <istruzione2>)  
    <istruzioni>
```

Il significato è esprimibile in termini di ciclo **while** come segue

```
<istruzione1>;  
while (<condizione>)  
{  
    <istruzioni>  
    <istruzione2>;  
}
```

Ciclo for



```
for (<istruzione1>; <condizione>; <istruzione2>)  
<istruzioni>
```

Il significato è esprimibile in termini di ciclo **while** come segue

```
<istruzione1>;  
while (<condizione>)  
{  
    <istruzioni>  
    <istruzione2>;  
}
```

```
<table border="1" cellspacing="0" cellpadding="10">  
  <thead style="background-color:lightblue">  
    <tr>  
      <th>Distanza (Km)</th>  
      <th>Costo (&euro;)</th>  
    </tr>  
  </thead>  
<?php  
  for($distanza = 100; $distanza <= 500; $distanza += 100)  
  {  
    echo '<tr><td>' . $distanza . '</td>';  
    echo '<td>' . ($distanza/2) . '</td></tr>';  
  }  
<?>  
</table>
```

Ciclo do..while



```
do
    <istruzioni>
while (<condizione>)
```

È simile al ciclo **while**, ma <condizione> viene testata dopo l'esecuzione di <istruzioni>

```
<table border="1" cellspacing="0" cellpadding="10">
  <thead style="background-color:lightblue">
    <tr>
      <th>Distanza (Km)</th>
      <th>Costo (&euro;)</th>
    </tr>
  </thead>
  <?php
    $distanza = 100;
    do
    {
      echo '<tr><td>' . $distanza . '</td>';
      echo '<td>' . ($distanza/2) . '</td></tr>';
      $distanza += 100;
    }while($distanza <= 500)
  <?>
</table>
```

Ciclo foreach



```
foreach (array as $valore)  
    istr
```

Il ciclo si ripete tante volte quanti sono gli elementi dell'array e all'interno del ciclo ogni volta è disponibile, nella variabile **\$valore**, il valore dell'elemento corrispondente all'iterazione

```
foreach (array as $chiave=>$valore)  
    istr
```

Questa versione è particolarmente utile per gli array associativi; oltre al valore è presente anche la chiave dell'elemento, presente nella variabile **\$chiave**

Funzioni



```
function nome_funzione ($arg1, $arg2)
{
    // codice della funzione (corpo)

    // l'istruzione return serve per restituire un valore
    // come risultato ed è opzionale
    return <risultato>;
}
```

Funzioni con argomenti di default



```
function nome_funzione ($arg1, $arg2="default")
{
    // codice della funzione (corpo)
return <risultato>;
}

. . .

$ris1 = nome_funzione("primo","secondo");    // senza l'uso del
                                              // default
$ris2 = nome_funzione("primo");              // con l'uso del default
```



Esempio: gestione di un ordine



```
<body>
  <h1>Gestione Ordine</h1>
  <?php
    echo '<p>Ordine gestito alle ore ';
    echo date("H:i, jS F");
    echo '. Grazie per aver scelto i nostri prodotti.</p>';
  ?>
</body>
```

Argomento della funzione **date()** = stringa che specifica il formato:

H = ore (in formato 24-ore)

i = minuti

j = giorno del mese

S = suffisso ordinale (tipicamente, *th*)

F = nome del mese

09:52, 15th Apri

Visibilità delle variabili



La **visibilità** (o **scope**) di una variabile **\$v** è la porzione del codice in cui è visibile **\$v**:

- ✓ Le variabili **superglobali** sono visibili ovunque nello script (es., **\$_SERVER**)
- ✓ Le costanti sono visibili ovunque nello script
- ✓ Le variabili **globali** sono visibili ovunque nello script, tranne che nelle funzioni
- ✓ Le variabili dichiarate globali nelle funzioni si riferiscono alle omonime variabili dichiarate globali fuori delle funzioni
- ✓ Le variabili dichiarate **statiche** in una funzione sono visibili solo nella funzione ma mantengono il loro valore tra una chiamata e l'altra della funzione
- ✓ Le variabili **locali** create in una funzione sono visibili solo nelle funzione e scompaiono al termine della esecuzione della funzione

Visibilità delle variabili: esempio



```
$a = 1;
function nome_funzione ()
{
    global $a;      // senza questa riga la funzione non stampa
                    // niente

    echo $a;
}

. . .

nome_funzione();
```

Classi e oggetti (I)



- ✓ Nel linguaggio PHP il concetto di classe è quello tradizionale dei linguaggi di programmazione ad oggetti

```
class nome_classe
{
    var variabili_comuni_alla_classe;

    function nome_classe(parametro1="valore_di_default1",...) {...}

    function nome_metodo1(parametro1, parametro2, ...) {...}

    function nome_metodo2(parametro1, parametro2, ...) {...}

    ...
}
```

Classi e oggetti (II)



- ✓ La gestione delle classi e della programmazione ad oggetti è stata enormemente potenziata nell'ultima versione di PHP (PHP 5)
 - ✓ sono stati introdotti i modificatori di visibilità **private**, **public** e **protected**, su cui valgono le comuni regole della programmazione a oggetti
 - ✓ sono stati introdotti i metodi **__construct** (nome univoco per il costruttore) e **__destruct** (nome univoco metodo distruttore)
 - ✓ è possibile passare gli oggetti per **riferimento** e non per **valore**

Costruttori e distruttori



- ✓ PHP 5 ha introdotto un nome standardizzato per i *costruttori*, **`__construct`**
 - ✓ se la classe viene rinominata, non è necessario modificare anche il nome del suo costruttore
- ✓ Un'altra novità in PHP 5 è l'introduzione del metodo *distruttore*, **`__destruct()`**
 - ✓ utile per operazioni di pulizia, come l'eliminazione di file temporanei o la chiusura di connessioni a database (*garbage collection*)
 - ✓ la garbage collection avviene nel momento in cui viene eliminato l'ultimo riferimento all'oggetto

Classi e oggetti: esempio (I)



```
<?php
2 references | 0 implementations
class Book
{
    3 references
    private $id;
    3 references
    private $title;
    3 references
    private $author_lastName;
    3 references
    private $authorID;

    2 references | 0 overrides
    public function __construct($identifier, $book_title, $author_name, $author_id)
    {
        $this->id = $identifier;
        $this->title = $book_title;
        $this->author_lastName = $author_name;
        $this->authorID = $author_id;
    }

    2 references | 0 overrides
    function getId() {
        return $this->id;
    }
}
```

Classi e oggetti: esempio (II)



```
<?php
2 references | 0 implementations
class Author
{
    3 references
    private $id;
    3 references
    private $firstName;
    3 references
    private $lastName;

    2 references | 0 overrides
    public function __construct($identifier, $first_name, $last_name)
    {
        $this->id = $identifier;
        $this->firstName = $first_name;
        $this->lastName = $last_name;
    }

    0 references | 0 overrides
    public function getId()
    { ...
    }
```

Istanziazione



- ✓ Una volta definita la struttura della classe, è possibile istanziare uno o più oggetti di quel tipo
 - ✓ lasciando i valori di default
 - ✓ specificando nuovi parametri al momento dell'istanziazione

```
$author = new Author(2, "Umberto", "Eco");  
$book = new Book(5, "Il nome della rosa", $author->getLastName(), $author->getId());
```

Invocazione dei metodi di una classe



- ✓ Dopo aver istanziato la classe, i metodi della classe possono essere invocati per andare a modificare le variabili di stato della classe stessa
 - ✓ le invocazioni dei metodi su una istanza sono indipendenti dalle invocazioni di metodi su istanze diverse

```
echo $author->getFirstName() . " " . $author->getLastName();  
  
$book->setTitle("My second life");
```

PHP 5 e oggetti



- ✓ In PHP 5 è stato anche potenziato il meccanismo dell'ereditarietà (parola chiave **extends**)
 - ✓ sono state introdotte le **interfacce**
 - ✓ è stato introdotto l'operatore **instanceof**
 - ✓ è possibile definire metodi e classi come **final**
 - ✓ è stato introdotto l'uso di **static** per definire proprietà e metodi statici

Interfacce



Grazie all'uso delle interfacce, viene superato il vincolo dell'ereditarietà da una singola classe; infatti una classe può implementare più interfacce

```
interface interfaccia_1 {  
    /* metodi astratti */  
    function funzione1();  
}  
  
interface interfaccia_2 {  
    /* metodi astratti */  
    function funzione2();  
}  
  
/* classe che implementa 2 interfacce */  
class classe_test implements interfaccia_1, interfaccia_2 {  
    function funzione1() {  
        /* implementazione */  
    }  
  
    function funzione2() {  
        /* implementazione */  
    }  
}
```

L'operatore "instance of"



- ✓ Viene utilizzato per verificare la classe di un oggetto

```
if($book instanceof Book)
{
    echo "Ho trovato un'istanza di Book";
} else {
    echo "Falso allarme!";
}
```

Metodi e classi final



- ✓ Se un metodo è dichiarato **final**, non può essere effettuato per esso *l'overriding* dalle classi che lo ereditano

```
class mia_classe
{
    final function mia_funzione() {
        /* questo metodo non potrà essere sovrascritto
           dalle classi che lo ereditano*/
    }
}
```

- ✓ Se una classe è dichiarata come **final**, non può essere sottoposta al meccanismo dell'ereditarietà

```
final class FinalClass
{
    /* ... */
}
```


Metodi e classi static



```
class settings
{
    static $max_users = 5;
}

echo settings::$max_users;

class lib
{
    static function get_time() {
        return time();
    }
}

echo lib::get_time();
```

Riuso del codice PHP



- ✓ L'inclusione di codice PHP in altri file avviene tramite la funzione **require(nome_file)**

inclusione.php

```
<?php
echo '<p><em>Questo messaggio si trova nel file di inclusione ...</em></p>';
?>
```

main.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <title>Inclusione di File</title>
</head>
<body>
  <h1>Inclusione di File</h1>
  <?php
    echo '<p>Ecco il contenuto del file di inclusione:</p>';
    require('inclusione.php');
    echo '<p>Il messaggio precedente non era scritto in questo file!</p>';
  ?>
</body>
</html>
```



- ✓ Per evitare di includere un file più volte, si usa **require_once**

Require vs include



- ✓ Nelle ultime versioni del PHP non viene fatta nessuna distinzione tra l'uso di **require** e di **include**, entrambe le funzioni servono per includere file esterni e hanno la medesima sintassi
- ✓ Le due funzioni in realtà si comportano in modo diverso solo nel caso in cui ci siano degli errori di inclusione del file
 - ✓ **include(nome_file)** restituisce un warning, mentre lo script prosegue
 - ✓ **require(nome_file)** segnala l'errore e blocca lo script
- ✓ Entrambe le funzioni hanno l'estensione **_once** per evitare il caricamento multiplo dello stesso file

Altre novità in PHP 5



Gestione delle eccezioni con **try/throw/catch**

```
<?php
try {
    $a = -1; // valore inaspettato
    if ( $a < 0 )
        throw new Exception('$a è negativo');

    // il codice che segue una eccezione non viene eseguito
    echo 'Questo testo non verrà mai visualizzato';
}
catch (Exception $e) {
    echo "Eccezione intercettata: ", $e, "n";
}
?>
```

I namespace in PHP



```
<php
namespace Acme\Database\Connection;

use Acme\Database\Connection\Connection;
use Acme\Database\Connection2\Connection as ConnectionTwo;

class MySQL
{
    // Corpo della classe

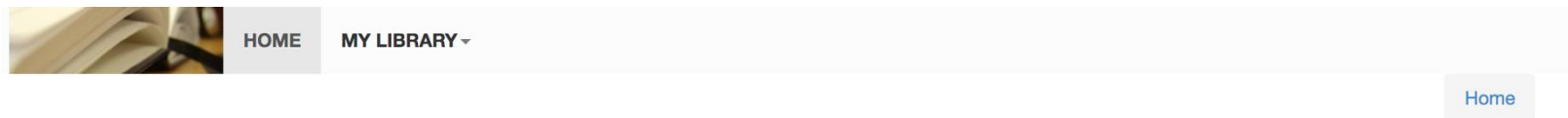
    public function getConnection()
    {
        return new Connection(); // Restituisce un'istanza della classe Acme\Dat
    }

    public function getConnection2()
    {
        return new ConnectionTwo(); // Restituisce un'istanza della classe Acme\
    }
}
```

Running example



Riprogettiamo l'esempio di riferimento utilizzando PHP e MySQL



My Books List

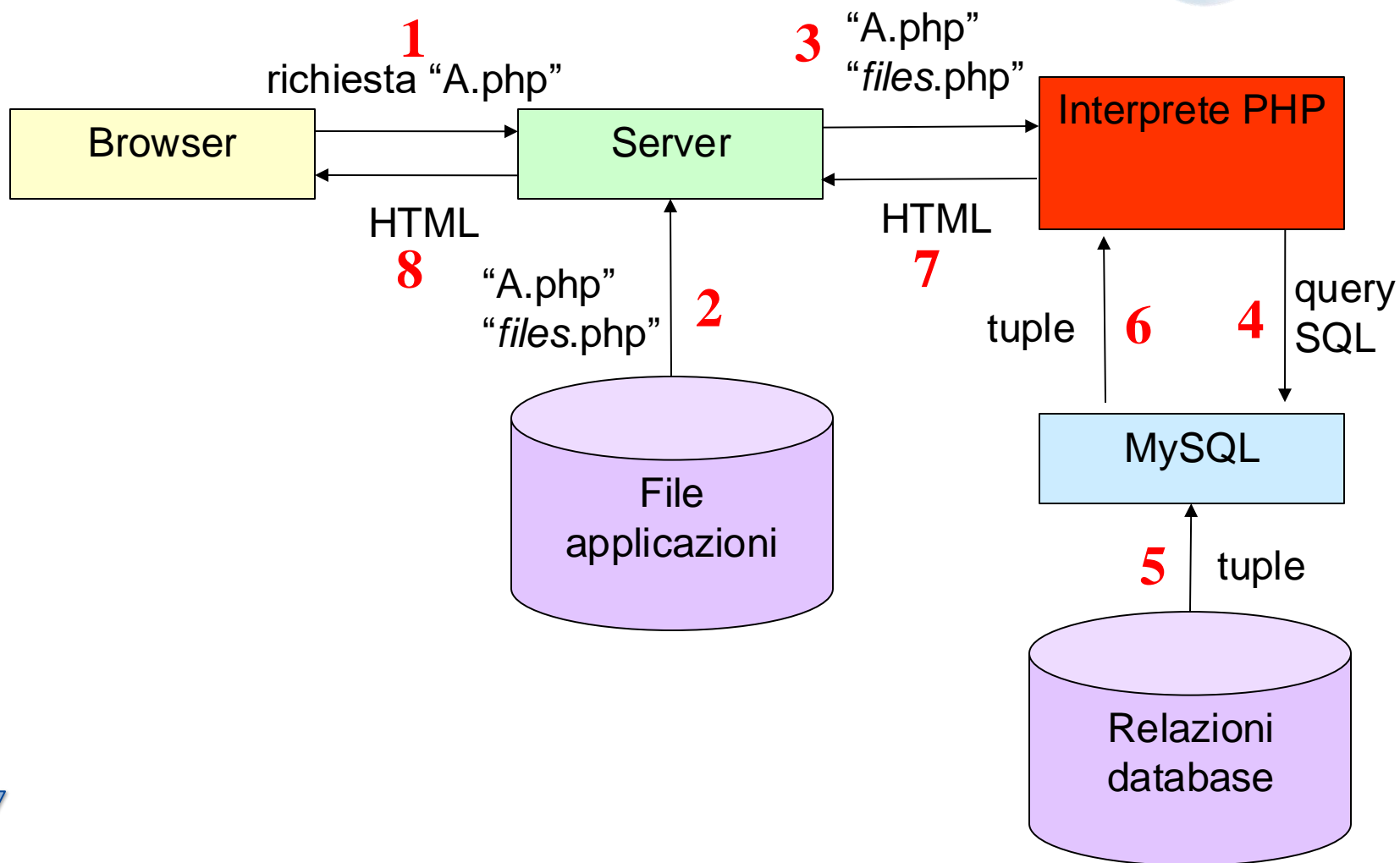
Un semplicissimo esempio di sito web realizzato durante il corso di Programmazione Web e Servizi Digitali. Il sito riporta l'elenco dei libri che sto leggendo o che ho letto, e la lista degli autori che hanno popolato le mie letture e la mia fantasia. Il sito web continuerà a crescere durante questo semestre, completandosi di volta in volta grazie all'applicazione delle tecnologie web che verranno presentate nel corso. Buon divertimento!

Semina un atto, e raccogli un'abitudine; semina un'abitudine, e raccogli un carattere; semina un carattere, e raccogli un destino.

— [Il pensiero del Buddha]



Interazione tra PHP e MySQL



Alcune funzioni di utilità...



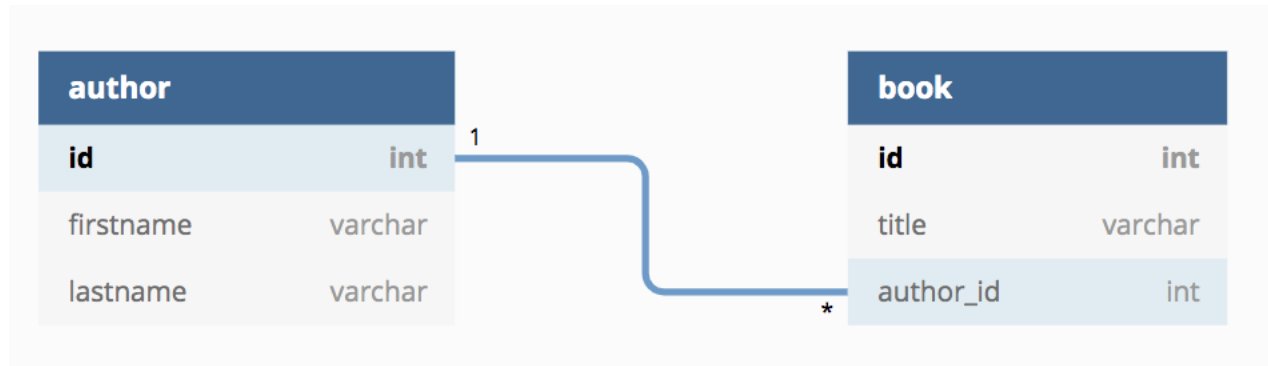
Per costruire porzioni ripetitive della pagina XHTML

```
function html_head($titolo) {  
    $result = ' <head>';  
    $result .= ' <title>'.$titolo.'</title>';  
    $result .= ' <meta charset="UTF-8">';  
    $result .= ' <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">';  
    $result .= ' <!-- Fogli di stile -->';  
    $result .= ' <link rel="stylesheet" href="http://' . $_SERVER['HTTP_HOST'] . '/PW_runningExample_v3_PHP/css/bootstrap.css">';  
    $result .= ' <link rel="stylesheet" href="http://' . $_SERVER['HTTP_HOST'] . '/PW_runningExample_v3_PHP/css/style.css">';  
    $result .= ' <!-- jQuery e plugin JavaScript -->';  
    $result .= ' <script src="http://code.jquery.com/jquery.js"></script>';  
    $result .= ' <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>';  
    $result .= ' <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>';  
    $result .= ' <script src="http://' . $_SERVER['HTTP_HOST'] . '/PW_runningExample_v3_PHP/js/bootstrap.min.js"></script>';  
    $result .= ' <!-- Bootstrap Icons -->';  
    $result .= ' <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css">';  
    $result .= ' </head>';  
  
    return $result;  
}
```



utils/XHTML_functions.php

II database



Alcune funzioni di utilità...



Per costruire porzioni ripetitive della pagina XHTML

Per modellare classi PHP corrispondenti alle tabelle del database (seguendo la tecnica *ORM – Object-Relational Mapping*, che vedremo nella programmazione agile)

- `Author.php`, `Book.php`

Per modellare il *DAO (Data Access Object)*, astraendo dalla sottostante “*storage layer*”

- `DataLayer.php`



Alcune funzioni di utilità...



Per costruire porzioni ripetitive della pagina XHTML

Per modellare classi PHP corrispondenti alle tabelle del database (seguendo la tecnica *ORM – Object-Relational Mapping*, che vedremo nella programmazione agile)

- `Author.php`, `Book.php`

Per modellare il *DAO (Data Access Object)*, astraendo dalla sottostante “*storage layer*”

- `DataLayer.php`



Connessione a MySQL



```
new PDO("mysql:host={$HOST};dbname={$DB_NAME}", $USERNAME, $PASSWORD);
```

PHP Data Object - apre una connessione al server ***\$HOST***, database ***\$DB_NAME***, con utente ***\$USERNAME*** e password ***\$PASSWORD***



Esempio (I)



```
<?php
$USERNAME = "devis";
$PASSWORD = "bianchini";
$HOST = "localhost";
$DB_NAME = "MyLibraryDB";
```

utils/config.php



Esempio (II)



```
include_once('../utils/config.php');  
include_once('Author.php');  
include_once('Book.php');
```

2 references | 0 implementations

class DataLayer



4 references

```
private $pdo;
```

```
/**
```

```
 * Constructor of the DataLayer class
```

```
*/
```

2 references | 0 overrides

```
public function __construct()
```

```
{
```

```
    global $HOST, $USERNAME, $PASSWORD, $DB_NAME;
```

```
    try {
```

```
        $this->pdo = new PDO("mysql:host={$HOST};dbname={$DB_NAME}", $USERNAME, $PASSWORD);
```

```
        // Set the PDO error mode to exception
```

```
        $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
    } catch(PDOException $e){
```

```
        die("ERROR: Could not connect. " . $e->getMessage());
```

```
    }
```

```
}
```



Esecuzione di query SQL



(new PDO) ->query(\$sql):

invia una query SQL al database a cui si è attualmente connessi

a)

```
$sql = "SELECT * FROM book ORDER BY title";  
try {  
    $statement = $this->pdo->query($sql);
```

b)

```
$sql = "SELECT * FROM author where id = :id";  
try {  
    $statement = $this->pdo->prepare($sql);  
    $statement->execute(array(':id' => $id));
```

c)

```
$sql = "SELECT * FROM author where id = :id";  
try {  
    $statement = $this->pdo->prepare($sql);  
    $statement->bindParam(':id', $id);  
    $statement->execute();
```

Elaborazione del risultato (I)



`$queryStatement->fetch(PDO::FETCH_ASSOC)` :
estrae un record dal risultato di una query (ogni volta che viene invocata) sotto forma di array associativo (i nomi delle colonne del record diventano i nomi degli elementi dell' array)

```
public function findAuthorById($id)
{
    $sql = "SELECT * FROM author where id = :id";
    try {
        $statement = $this->pdo->prepare($sql);
        $statement->execute(array(':id' => $id));

        $author = $statement->fetch(PDO::FETCH_ASSOC);

        // Check if author found
        if ($author) {
            return new Author($author['id'], $author['firstname'], $author['lastname']);
        } else {
            return null; // Author not found
        }
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```


Elaborazione del risultato (I)



`$queryStatement->fetch(PDO::FETCH_ASSOC)` :
estrae un record dal risultato di una query (ogni volta che viene invocata) sotto forma di array associativo (i nomi delle colonne del record diventano i nomi degli elementi dell' array)

```
public function findAuthorById($id)
{
    $sql = "SELECT * FROM author where id = :id";
    try {
        $statement = $this->pdo->prepare($sql);
        $statement->execute(array(':id' => $id));

        $author = $statement->fetch(PDO::FETCH_ASSOC);

        // Check if author found
        if ($author) {
            return new Author($author['id'], $author['firstname'], $author['lastname']);
        } else {
            return null; // Author not found
        }
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```

`$author`

id	2
firstname	Alessandro
lastname	Manzoni

Elaborazione del risultato (II)



`$queryStatement->fetchAll(PDO::FETCH_ASSOC):`

simile al precedente, ma in questo caso recupera tutti i record nel risultato di una query, mettendoli in un array di array associativi

```
public function listBooks()
{
    $sql = "SELECT * FROM book ORDER BY title";
    try {
        $statement = $this->pdo->query($sql);

        $books = $statement->fetchAll(PDO::FETCH_ASSOC);
        $booksList = array();
        foreach ($books as $book) {
            $author = $this->findAuthorById($book['author_id']);
            $booksList[] = new Book($book['id'], $book['title'], $author->getFirstName()." ".$author->getLastName(), $book['author_id']);
        }
        return $booksList;
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```

Modifica del database



`$queryStatement->rowCount()` :

restituisce il numero di tuple coinvolte nella query appena eseguita

```
public function findBooksByAuthorID($author_id)
{
    $sql = "SELECT * FROM book where author_id = :author_id";
    try {
        $statement = $this->pdo->prepare($sql);
        $statement->bindParam(':author_id', $author_id);
        $statement->execute();

        // Get the number of affected rows
        $affectedRows = $statement->rowCount();

        if($affectedRows != 0)
        {
            return true;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```

Running example (I)



La lista degli autori...



HOME

MY LIBRARY ▾

[Home](#) / [My Library](#) / [Authors](#)

My Authors

[Create new author](#)

Author's name

Dante Alighieri

[Edit](#)

[Delete](#)

Vito Mancuso

[Edit](#)

[Delete](#)

Alessandro Manzoni

[Edit](#)

[Delete](#)

Edgar Allan Poe

[Edit](#)

[Delete](#)

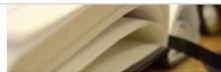


Running example (I)



La lista degli autori...

La lista è caricata automaticamente dal database...



HOME

MY LIBRARY ▾

[Home](#) / [My Library](#) / [Authors](#)

My Authors

[Create new author](#)

Author's name

Dante Alighieri

[Edit](#)

[Delete](#)

Vito Mancuso

[Edit](#)

[Delete](#)

Alessandro Manzoni

[Edit](#)

[Delete](#)

Edgar Allan Poe

[Edit](#)

[Delete](#)

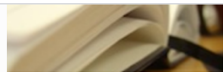


Running example (I)



La lista degli autori...

La lista è caricata automaticamente dal database...



HOME

MY LIBRARY ▾

[Home](#) / [My Library](#) / [Authors](#)

My Authors

[Create new author](#)

Author's name

Dante Alighieri

[Edit](#)

[Delete](#)

Vito Mancuso

[Edit](#)

[Delete](#)

Alessandro Manzoni

[Edit](#)

[Delete](#)

Edgar Allan Poe

[Edit](#)

[Delete](#)

Non è possibile rimuovere un autore con dei libri associati



Running example (II)



Creare/modificare un autore...



HOME

MY LIBRARY ▾

[Home](#) / [My Library](#) / [Authors](#) / [Edit author](#)

Edit Author

First Name

Last Name

 Save

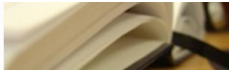
 Cancel



Running example (II)



Creare/modificare un autore...



HOME

MY LIBRARY ▾

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

[Home](#) / [My Library](#) / [Authors](#) / [Edit author](#)

Edit Author

First Name

Last Name

 Save

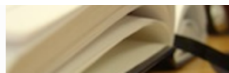
 Cancel



Running example (II)



Creare/modificare un autore...



HOME

MY LIBRARY ▾

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

[Home](#) / [My Library](#) / [Authors](#) / [Edit author](#)

Edit Author

First Name

Alessandro

Last Name

Manzoni

Alcuni aspetti della pagina sono differenziati tra le due pagine (e.g., “Edit Author”/“Create new Author”)

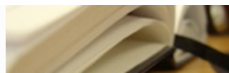
Save

Cancel

Running example (II)



Creare/modificare un autore...



HOME

MY LIBRARY ▾

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

[Home](#) / [My Library](#) / [Authors](#) / [Edit author](#)

Edit Author

First Name

Alessandro

Last Name

Manzoni

Alcuni aspetti della pagina sono differenziati tra le due pagine (e.g., “Edit Author”/“Create new Author”)

Save

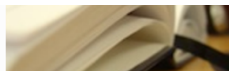
Cancel

Valori precaricati in caso di modifica di un autore (partendo dal parametro `$_GET['id']`)

Running example (II)



Creare/modificare un autore...



HOME

MY LIBRARY ▾

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

[Home](#) / [My Library](#) / [Authors](#) / [Edit author](#)

Edit Author

First Name

Alessandro

Last Name

Manzoni

Alcuni aspetti della pagina sono differenziati tra le due pagine (e.g., “Edit Author”/“Create new Author”)

Save

Cancel

Valori precaricati in caso di modifica di un autore (partendo dal parametro `$_GET['id']`)

La creazione/modifica nel database è effettuata invocando lo script stesso

Running example (III)



Eliminare un autore...



HOME

MY LIBRARY ▾

[Home](#) / [My Library](#) / [Authors](#) / Delete author

Delete author "Dante Alighieri" from the list

Deleting author. Confirm?

Revert

The author **will not be removed** from the data base

[↶ Back to authors' list](#)

Confirm

The author **will be permanently removed** from the data base

[🗑 Delete](#)

Running example (III)



Eliminare un autore...



HOME MY LIBRARY ▾

Nome dell'autore precaricato
partendo dal parametro
`$_GET['id']`

[Home](#) / [My Library](#) / [Authors](#) / Delete author

Delete author "Dante Alighieri" from the list

Deleting author. Confirm?

Revert

The author **will not be removed** from the data base

[↶ Back to authors' list](#)

Confirm

The author **will be permanently removed** from the data base

[🗑 Delete](#)

Running example (III)



Eliminare un autore...



HOME MY LIBRARY ▾

Nome dell'autore precaricato
partendo dal parametro
`$_GET['id']`

[Home](#) / [My Library](#) / [Authors](#) / Delete author

Delete author "Dante Alighieri" from the list

Deleting author. Confirm?

Revert

The author **will not be removed** from the data base

[↩ Back to authors' list](#)

Confirm

The author **will be permanently removed** from the data base

[Delete](#)

La cancellazione dal database è
effettuata invocando lo script
stesso



MANIPOLAZIONE DELL'INTESTAZIONE HTTP



Gli header HTTP



- ✓ Normalmente, PHP è usato per costruire il corpo della risposta HTTP del server, che per default presenta un'intestazione HTTP standard contenente, tra gli altri, il tipo MIME del file HTML (**text/html**)
- ✓ Talvolta è necessario gestire le intestazioni del protocollo HTTP (**headers**)
- ✓ Due tipologie di **headers**
 - ✓ **request headers**: inviati dal browser verso il server quando viene generata una richiesta HTTP (per esempio, tramite il click su un link)
 - ✓ **response headers**: inviati dal server al browser in risposta ad una determinata richiesta (per esempio, sotto forma di comune pagina HTML)

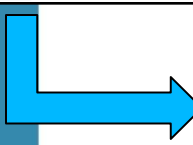
Header HTTP nella richiesta



- ✓ Per visualizzare gli headers inviati nella richiesta si può usare la funzione **getallheaders()** che produce un array contenente tutte le informazioni relative agli headers inviati in input (alias di **apache_request_headers()**)

```
<?php
# chiamata alla funzione per la raccolta dei request headers
$headers = getallheaders();
# visualizzazione dei valori dell'array tramite ciclo
foreach ($headers as $name => $content)
{
    echo "[$name] = $content<br />\n";
}
?>
```

Elenca contenuto directory



```
[Host] = localhost
[Connection] = keep-alive
[Accept] = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
[User-Agent] = Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.63 Safari/537.31
[Referer] = http://localhost/prova/headers/
[Accept-Encoding] = gzip,deflate,sdch
[Accept-Language] = it-IT,it;q=0.8,en-US;q=0.6,en;q=0.4
[Accept-Charset] = ISO-8859-1,utf-8;q=0.7,*;q=0.3
[Cookie] = PHPSESSID=r7vuie4l3mi4t75f45001c5m23; mail=osor@osor.it
```

Header HTTP nella risposta



- ✓ Esiste un'analoga funzione **apache_response_headers()** per visualizzare gli headers nella risposta HTTP
- ✓ Talvolta è necessario modificare/aggiungere altre intestazioni
 - ✓ per re-dirigere il browser su un'altra pagina (eventualmente dopo un certo tempo)
 - ✓ per evitare il caching della pagina
 - ✓ per modificare il **content type** della pagina
 - ✓ ...
- ✓ A tale scopo esiste la funzione **header()**
- ✓ La funzione **header()** deve essere invocata prima di qualsiasi altro output, di qualunque elemento HTML o di qualunque spazio

```

# modifica dell'header relativo allo status della richiesta,
# utile nel caso in cui si debbano risolvere problematiche
# relative alla riscrittura delle URL
header($_SERVER["SERVER_PROTOCOL"]." 200 OK");

# header per le richieste non soddisfatte dal server
header($_SERVER["SERVER_PROTOCOL"]." 404 Not Found");

# header prodotto dall'impossibilità di accedere alla
# risorsa richiesta
header($_SERVER["SERVER_PROTOCOL"]." 403 Forbidden");

# header prodotto per segnalare una risorsa
# spostata in modo permanente su un percorso diverso
# da quello utilizzato per definire la richiesta
header($_SERVER["SERVER_PROTOCOL"]." 301 Moved Permanently");

# header prodotto nel caso di malfunzionamenti del server
# in seguito al tentativo di soddisfare una richiesta
header($_SERVER["SERVER_PROTOCOL"]." 500 Internal Server Error");

# rindirizzamento della richiesta su un percorso diverso
# da quello specificato
header('Location: http://www.miosito.it/');

# rindirizzamento della richiesta entro un periodo
# di tempo definito (delay)
header('Refresh: 5; url=http://www.miosito.it/');
echo 'Il browser verrà redirezionato entro 5 secondi';

# override del valore relativo all'header X-Powered-By
header('X-Powered-By: PHP/5.2.8');

# modifica dell'header relativo al linguaggio utilizzato
header('Content-language: en');

# modifica dell'header relativo alla data dell'ultima modifica
$time = time() - 60;
header('Last-Modified: '.gmdate('D, d M Y H:i:s', $time).' GMT');

```



```

# modifica dell'header relativo allo status della richiesta,
# utile nel caso in cui si debbano risolvere problematiche
# relative alla riscrittura o # header per segnalare l'assenza di modifiche
header($_SERVER["SERVER_PROTOCOL"] header($_SERVER["SERVER_PROTOCOL"]." 304 Not Modified");

# header per le richieste non # impostazione del Content-Length per operazioni di caching
header($_SERVER["SERVER_PROTOCOL"] header('Content-Length: 168');

# header prodotto dall'imposs # header per il download
# risorsa richiesta header('Content-Type: application/octet-stream');
header($_SERVER["SERVER_PROTOCOL"] header('Content-Disposition: attachment; filename="file.zip"');
header('Content-Transfer-Encoding: binary');

# header prodotto per segnala # scaricamento di un file
# spostata in modo permanente readfile('file.zip');
# da quello utilizzato per de header($_SERVER["SERVER_PROTOCOL"]);

# header prodotto nel caso di # disabilitazione della cache per il documento corrente
# in seguito al tentativo di header('Cache-Control: no-cache, no-store, max-age=0, must-revalidate');
header($_SERVER["SERVER_PROTOCOL"] header('Expires: Mon, 26 Jul 1997 05:00:00 GMT'); # Date in the past
header('Pragma: no-cache');

# rindirizzamento della richi # impostazione del content type per
# da quello specificato # le varie tipologie di estensioni:
header('Location: http://www. header('Content-Type: text/html; charset=iso-8859-1');
header('Content-Type: text/html; charset=utf-8');

# rindirizzamento della richi header('Content-Type: text/plain');
# di tempo definito (delay) header('Content-Type: image/jpeg');
header('Refresh: 5; url=http: header('Content-Type: application/zip');
echo 'Il browser verrà redire header('Content-Type: application/pdf');
header('Content-Type: audio/mpeg');

# override del valore relativ header('Content-Type: application/x-shockwave-flash');
header('X-Powered-By: PHP/5.2

# richiesta di autenticazione
# modifica dell'header relati header($_SERVER["SERVER_PROTOCOL"]." 401 Unauthorized");
header('Content-language: en' header('WWW-Authenticate: Basic realm="Accesso Riservato"');
echo 'Inserire i dati per il login';

# modifica dell'header relativo alla data dell'ultima modifica
$time = time() - 60;
header('Last-Modified: '.gmdate('D, d M Y H:i:s', $time).' GMT');

```

Buffering



- ✓ Le intestazioni devono precedere qualunque contenuto, perché una volta che la pagina inizia ad essere inviata non è più possibile modificare l'intestazione della risposta HTTP
- ✓ Questo vincolo può essere aggirato con il meccanismo del *buffering*
 - ✓ la funzione `ob_start()` viene posta immediatamente prima del contenuto della pagina, compresi eventuali spazi bianchi
 - ✓ in tal modo, lo script non inizia ad inviare la pagina fino a quando non ha terminato la sua esecuzione
 - ✓ si può ripristinare il consueto meccanismo di emissione immediata dell'output invocando la funzione `ob_end_flush()`



PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PHP

Prof. Ada Bagozi
ada.bagozi@unibs.it

