

PHP

Introduzione

Caratteristiche

Il codice PHP

Variabili e Tipi di Dato

Variabili

Tipi di Dato

Conversione di Tipo

Array

Scope di una variabile

Funzioni

Passaggio di parametri

Funzioni Predefinite

Strutture di Controllo Condizionale ed Iterativo

If - If...else

Switch

While - Do...while

For - Foreach

Break, continue

Gestione contenuti delle FORM

Passaggio di parametri

Interfacciamento a MySQL

PHP e DB

MySQL

Esempi d'uso

PHP e MySQL

Gestione Integrata

Visualizza

Inserisci

Modifica

DBAccess

Accesso autorizzato

Session

Introduzione

I documenti possono essere contenuti statici o dinamici.

Il contenuto di un documento statico non cambia in funzione delle richieste

fatte al server dai client, è tempo invariante, generalmente è un documento HTML presente nel server che viene restituito tramite protocollo HTTP(S) al client a seguito di una richiesta.

Un documento il cui contenuto varia a seconda di diversi parametri elaborati (ad esempio quando effettuiamo una ricerca su Google) necessita di un programma che elabori questi parametri per ottenere il contenuto della pagina. Oltre al JS analizziamo ora il linguaggio PHP.

Caratteristiche

- PHP Hypertext Preprocessor (abbreviato in PHP, acronimo ricorsivo) è un linguaggio di programmazione *full-function* (a differenza del JS che non consente di operare accedendo a tutte le risorse della macchina in cui è in esecuzione) che consente di codificare algoritmi all'interno di pagine HTML (o sorgenti scritte solo in PHP). Il contenuto della pagina cambierà in base ai parametri passati come argomenti.

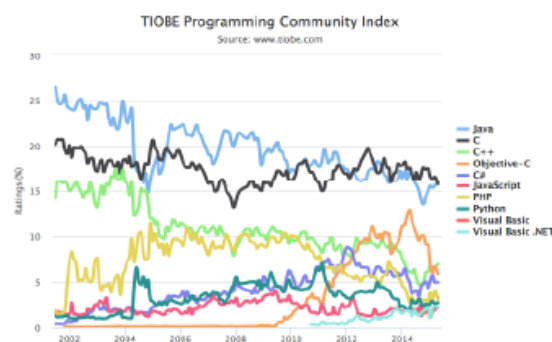
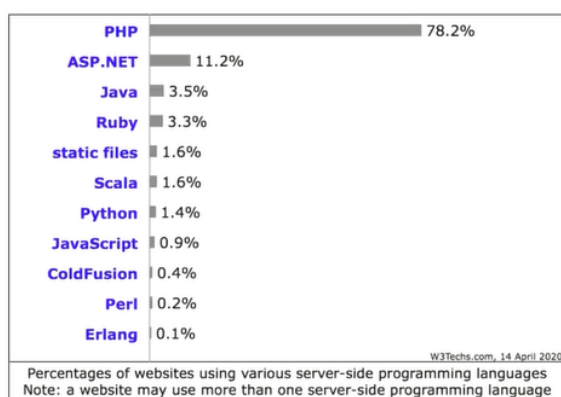
Due possibili modalità:

- *embedded* in HTML (come visto con JS), inserendo quindi porzioni di codici PHP all'interno del codice HTML, il risultato sarà quindi inserito all'interno del documento sostituendo la porzione di codice PHP e determinando in maniera dinamica il contenuto del documento.
 - Codice scritto interamente in PHP che genera attraverso l'istruzione `echo()` (o analoghe) il risultato del codice PHP
- È stato sviluppato a partire dal 1995. L'ultima versione, PHP7, è del 2016 (attualmente 7.4.X)
 - **Il codice PHP viene sempre eseguito dal lato Server**, è un codice che non arriva mai al Client, al quale arriva solamente il risultato della sua elaborazione. Questo determina che i codici PHP sono invisibili lato Client e non sono quindi visibili, copiabili o modificabili.
L'ambiente server deve essere integrato con il traduttore per eseguire il codice in maniera autonoma. PHP è infatti un modulo di Apache che si attiva recuperando il codice ed eseguendolo.
 - È *free*, ma rimane aggiornato da un gruppo di persone.

- È utilizzabile in molti SO (Unix/Linux, Windows, MacOS, ...) e da diversi tipi di server (Apache, IIS, nginx, ...) → È molto utilizzato
- È *multiparadigma*: può essere usato come linguaggio imperativo (come C) o "ad oggetti" (come Java), nasce come imperativo e si sviluppa verso "a oggetti" (utilizzato in Laravel).
- È un *linguaggio di programmazione completo*, che consente di accedere a tutte le risorse dal lato server. Possiamo aprire un file, creare un file e accedere alla linea di comunicazione accedendo ad altri server connessi al server.

Popolarità di PHP

Molto utilizzato.



Il codice PHP

In maniera analoga a quanto abbiamo visto in JS, possiamo inserire un codice PHP all'interno del codice HTML, che una volta eseguito genererà una parte di contenuti del documento stesso in maniera dinamica, dipendenti da parametri che possono (volendo) essere passati dal Client.

Per includere un codice PHP in un documento HTML, si può:

- Utilizzare la forma `<?php <istruzioni_php> ?>` per inserire istruzioni del linguaggio. Queste tag non sono tag HTML, ma queste righe in un altro linguaggio diverso dall' HTML non creano problemi, in quanto queste righe vengono interpretate dal modulo PHP lato server, che le riconosce ed esegue, restituendo il risultato; quindi questi codici PHP non verranno mai

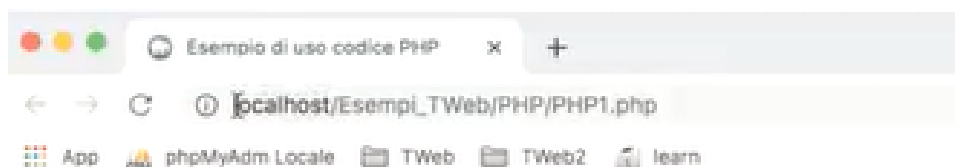
visti dal lato client in quanto il client riceverà soltanto il risultato dell'esecuzione del codice PHP.

- Utilizzare la forma `<?= <testo> ?>` per produrre in output la stringa `<testo>` (equivale ad utilizzare la forma abbreviata dell'istruzione `echo <testo>`). Questa variante utilizza il PHP per la generazione di testo, non di istruzioni. Affinchè questa abbreviazione sia ben processata da Apache, all'interno del file di configurazione dell'ambiente PHP (`php.ini`) è necessario settare il parametro `short_open_tag` : se questo parametro è settato a `true` questa forma viene ben processata (generalmente `true` di default).
- Se la nostra pagina è formata solo da codice PHP, il quale genererà totalmente il documento HTML, il codice sarà contenuto tra le tag `<?php>` e `<?>`.
- Tutte le istruzioni terminano con `;`

Il codice PHP può essere collocato in qualsiasi sezione del documento HTML (`<head>` , `<body>`)

▼ Esempio PHP#1

I file che contengono una qualunque riga di PHP vengono salvati con l'estensione `.php` ; è infatti un riferimento per il server che dovrà andare a cercare ed eseguire il codice PHP, prima di restituire il codice al Client.



Stampa da codice PHP inserito nella sezione HEAD

Stampa da codice PHP inserito nella sezione BODY

Stampa da codice HTML inserito nella sezione BODY

Da notare l'URL `localhost/Esempi_TWeb/PHP/PHP1.php` , il server ha interpretato il codice PHP e restituito al Client il codice HTML.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Esempio di uso codice PHP</title>
    <?php
      /* Codice nella sezione BODY */
      echo "<p>Stampa da codice PHP inserito nella s
ezione HEAD</p>";
    ?>
  </head>
  <body>
    <?= '<p>Stampa da codice PHP inserito nella sezi
one BODY</p>'; ?>
      /* con l'apice singolo la stringa viene
poi interpretata come HTML
      con apice doppio viene considerata come
semplice stringa */
    <p>Stampa da codice HTML inserito nella sezione
BODY</p>
  </body>
</html>

```

Variabili e Tipi di Dato

Variabili

- Identificatore di variabile → `<varid> ::= $ <letter> {<letter>|<digit>|_}` (deve iniziare con lettera)
- L'identificatore è 'case sensitive'
- Le variabili non vengono dichiarate, ma vengono create contestualmente dell'assegnazione del loro valore → `$indice=10`
- Il binding variabile-tipo è dinamico

Tipi di Dato

Tipo	Operatori
Integer	+, -, *, /, %, ++, --, =, +=, -=, *=, /=, %=, ==, !=, !==, >, <, >=, <=, ?:, &, , ^, ~, >>, <<, (int)
Real	Come integer, tranne gli operatori su bit (&, , ^, ~, >>, <<) che mancano e l'operatore di casting (int) che viene sostituito da (double)
Booleano	and, or, xor, &&, , !
Stringhe	., .=, (string)
Array	Dipendenti dal tipo degli elementi, oltre a (array)
Object	new, (object)

Da notare la possibilità di scegliere tra gli operatori di uguaglianza stretta e di uguaglianza lasca.

Tra gli operatori booleani `and` e `&&` hanno priorità diversa, `and` è più lasco, analogamente gli altri.

Conversione di Tipo

- Conversione stringa → numero

- Implicita:

```
$Y="3"; $Y*="7"; // $Y=21
```

- Esplicita

```
doubleval() , intval()
```

- Conversione numero → stringa

```
$Y=3; $Y.="Tavoli"; // $Y="3Tavoli"
```

Array

- Vengono definiti tramite il costrutto → `array({<elemval>}|{<elemnam>=><elemval>})`

```
$num=array(1, 17, "uno", 2=>16, "tre"); // 2=>16 assegna  
16 all'elemento [2]
```

```
$num[0] --> 1  
$num[1] --> 17  
$num[2] --> 16  
$num[3] --> "tre"
```

- Possono essere multidimensionali

```
$per = array(array("Mario", "Rossi", "Studente", 21),  
             array("Giovanni", "Bianchi", "Impie  
gato", 45));
```

- Possono essere associativi, cioè con identificatore dei singoli elementi svincolato dalla loro posizione nell'array, molto utile per creare coppie chiave-valore

```
$temp=array("Dom"=>15.5, "Lun"=>13.7, "Mar"=>14.9);  
$temp["Dom"] --> 15.5  
$temp["Lun"] --> 13.7  
$temp["Mar"] --> 14.9
```

- Hanno numerose funzioni ad essi associate predefinite nel linguaggio:
 - `sort()` : funzione di ordinamento di un array, opera ordinando per valore (non tiene conto della chiave)
 - `asort()` : funzione di ordinamento di un array, ordina mantenendo il legame chiave-valore, se si sposta il valore la chiave rimane legata al valore dopo l'ordinamento
 - `ksort()` : ordinamento per valore di chiave, mantiene il legame chiave-valore
 - `each()` : estrae il successivo, utile nei cicli di `for`
 - `foreach()` : crea ciclo iterativo che estrae tutti gli elementi
 - `list()` : operatore al quale posso passare uno o più identificatori di parametri.

```
list($a, $b) = array("Marco", "Giovanni") //$a = "Marco", $b = "Giovanni"
```

▼ Esempio PHP#2

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Esempio di uso di array in PHP</title>
    </head>
    <body>
        <?php
            echo '<h1 style="text-align: center;">Esempio
di uso di array</h1>';
            //attenzione all'uso corretto degli
apici doppi e singoli
            echo "<h3>Assegnazione e stampa dell'array (1
0,30,20,40,50)</h3>";
            $s1=array(10,30,20,40,50);
            for ($i=0;$i<5;$i++) {
                echo "$s1[$i] "; // l'interprete PHP interp
reta la stringa tra apici doppi
            }

            echo "<h3>Ordinamento e stampa dell'array prec
edente</h3>";
            sort($s1);
            for ($i=0;$i<5;$i++) {
                echo "$s1[$i] ";
            }

            echo '<h3>Assegnazione e stampa di un array as
sociativo</h3>';
            $s1=array("Lun"=>17.5, "Mer"=>16.8, "Mar"=>14.
9);
```

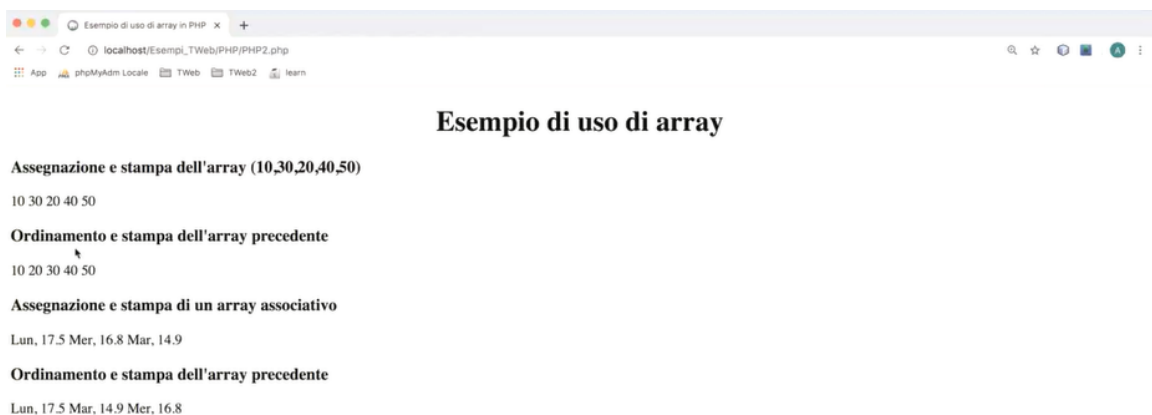


```

        while(list($nome,$valore) = each($s1)) {
            echo"$nome, $valore "; // stampiamo le coppi
e nome-valore
        }

        echo "<h3>Ordinamento e stampa dell'array prec
edente</h3>";
        ksort($s1);
        while(list($nome,$valore) = each($s1)) {
            echo"$nome, $valore ";
        }
    ?>
</body>
</html>

```



Scope di una variabile

In PHP le variabili possono essere:

- **Locali**, definite all'interno di un blocco di programma utilizzabili solo in esso
- **Globali**, definite all'esterno di un blocco di programma e utilizzabili tramite l'uso dell'attributo `global`

Lo scope, in PHP, è quindi **STATICO**.

Funzioni

- **Dichiarazione**

```
function <nome_funzione> ([<parametri>]) {<istruzioni>}
```

```
function somma ($a,$b) {  
    $risultato = $a + $b;  
    return $risultato;  
}
```

- **Attivazione**

```
$c = somma(3,7)
```

Possono essere:

- Predefinite nel linguaggio
- Definite in un file esterno (incluse con `require` (se non trovo la funzione si blocca) e `include` (se non trovo la funzione vado comunque avanti)) → File esterno che contiene le istruzioni, posso creare un DB di funzioni
- Definite nel documento

Passaggio di parametri

In PHP sono possibili due tecniche di passaggio dei parametri:

- Per valore: `function a ($x, $y) {...}`
- Per riferimento `function a (&$x, &$y) {...}`

L'operatore `&`, che definisce un alias a livello di *symbol-table* di una variabile, può essere usato in qualsiasi istruzione PHP.

▼ Esempio PHP#3

```
<!DOCTYPE html>  
<html>  
    <head>  
        <meta charset="UTF-8">  
        <title>Esempio di uso di funzioni in PHP</title>
```

```

<?php

    // Passaggio di Parametri per valore
    function vswap ($x, $y) {
        $tmp=$x;
        $x=$y;
        $y=$tmp;
    }

    // Passaggio di Parametri per riferimento
    function rswap (&$x, &$y) {
        $tmp=$x;
        $x=$y;
        $y=$tmp;
    }

    // Uso di variabili globali
    function x_alla_y ($y) {
        global $x;
        $tmp=1;
        for ($i=1;$i<=$y;$i++) {
            $tmp*=$x;
        }
        return $tmp;
    }
?>
</head>
<body>
    <?php
        echo '<h1 style="text-align: center;">Esempio
di uso di funzioni</h1>';

        echo '<h3>Funzione swap con parametri passati
per valore</h3>';
        $x=3;$y=5;
        echo "<p>Prima dell'attivazione x=$x, y=$y</p
>";

        vswap($x,$y);

```

```

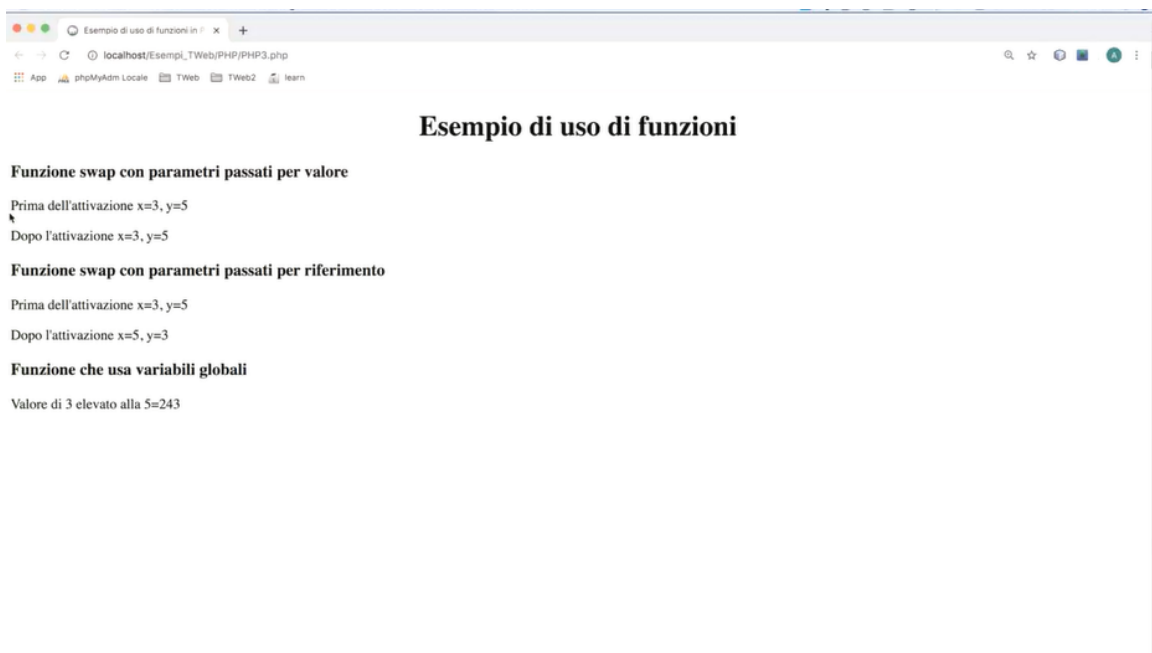
        echo "<p>Dopo l'attivazione x=$x, y=$y</p>";

        echo '<h3>Funzione swap con parametri passati
per riferimento</h3>';
        $x=3;$y=5;
        echo "<p>Prima dell'attivazione x=$x, y=$y</p
>";

        rswap($x,$y);
        echo "<p>Dopo l'attivazione x=$x, y=$y</p>";

        echo '<h3>Funzione che usa variabili globali</
h3>';
        $x=3;$y=5;           // la variabile globale $x
avrà valore 3
        $ris=x_alla_y($y); // la variabile globale $x
avrà valore 3
        echo "<p>Valore di $x elevato alla $y=$ris</p
>";
        ?>
    </body>
</html>

```



Funzioni Predefinite

Funzioni predefinite per gli array

<code>array()</code>	<code>arsort()</code>	<code>krsort()</code>
<code>array_keys()</code>	<code>asort()</code>	<code>list()</code>
<code>array_merge()</code>	<code>count()</code>	<code>rsort()</code>
<code>array_reverse()</code>	<code>current()</code>	<code>sizeof()</code>
<code>array_shift()</code>	<code>each()</code>	<code>sort()</code>
<code>array_slice()</code>	<code>end()</code>	...
<code>array_values()</code>	<code>ksort()</code>	

- `array_merge()` : prende due array e li unisce
- `array_reverse()` : inverte l'ordine di un array
- `array_shift()` : possibilità di estrarre sotto-componenti di un array
- `current()` : mi dice l'elemento corrente durante una scansione

Funzioni matematiche predefinite

<code>abs()</code>	<code>cos()</code>	<code>pow()</code>
<code>acos()</code>	<code>decbin()</code>	<code>rand()</code>
<code>asin()</code>	<code>dechex()</code>	<code>sin()</code>
<code>atan()</code>	<code>exp()</code>	<code>sqrt()</code>
<code>base_convert()</code>	<code>log()</code>	<code>tan()</code>
<code>bindec()</code>	<code>max()</code>	...
<code>ceil()</code>	<code>min()</code>	

- `bindec()` : conversione dal binario al decimale
- `ceil()` : arrotondamento all'intero superiore
- `base_convert()` : conversione da una base ad un'altra (entrambe passate come argomento)

Funzioni predefinite per le stringhe		
<code>chr()</code>	<code>ord()</code>	<code>strcmp()</code>
<code>count_chars()</code>	<code>printf()</code>	<code>strpos()</code>
<code>crypt()</code>	<code>rtrim()</code>	<code>strstr()</code>
<code>echo()</code>	<code>sscanf()</code>	<code>substr()</code>
<code>explode()</code>	<code>strchr()</code>	<code>wordwrap()</code>
<code>implode()</code>	<code>strcmp()</code>	...
<code>ltrim()</code>	<code>strlen()</code>	

- `explode()` : prende una stringa e la suddivide in parti che vengono assegnate a elementi di un array (ad esempio specificando il carattere dello spazio " " posso dividere il testo parola per parola)
- `crypt()` : ottengo codice criptato a partire da una stringa
- `ltrim()` e `rtrim()` : elimina da una stringa tutti gli spazi rispettivamente a sinistra del primo carattere significativo e a destra dell'ultimo carattere significativo.

Strutture di Controllo Condizionale ed Iterativo

Il PHP non modifica nulla a livello sintattico rispetto i linguaggi da cui eredita (linguaggio C).

If - If...else

```
if (<espressione_condizionale>) {
    <istruzioni>;
}
```

```
if (<espressione_condizionale>) {
    <istruzioni>;
}
else {
    <istruzioni>;
}
```

Switch

```
switch(<espressione>) {  
    case <etichetta>: <istruzioni>; break;  
    case <etichetta>: <istruzioni>; break;  
    ...  
    default: <istruzioni>;  
}
```

While - Do...while

```
while (<espressione_condizionale>) {  
    <istruzioni>;  
}
```

```
do {  
    <istruzioni>;  
} while (<espressione_condizionale>);
```

Le `<istruzioni>` vengono ripetute fintanto che l' `<espressione_condizionale>` è vera

For - Foreach

```
for (<inizializzazione>; <condizione>;<step>) {  
    <istruzioni>;  
}
```

```
foreach <espressione_array> { <istruzioni>; }
```

`foreach` consente di scandire gli elementi di un array e di usarne i valori all'interno del ciclo, presente nei linguaggi che utilizzano le collezioni, cioè le strutture dati legati agli array associativi, nei casi in cui cioè abbiamo delle coppie chiave-valore in cui non è presente un valore numerico da ciclare.

▼ Esempio PHP#4

Generiamo tutto il codice HTML grazie all'esecuzione del codice PHP. Per controllare un codice PHP non basta aprire il file tramite browser web come

per i file HTML e visualizzare così il documento. Dobbiamo installare un server (Apache) che farà partire il codice PHP generando quello HTML, digitando nel server l'URL `localhost:` con il path del file.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Esempio di uso della funzione FOREACH in
PHP</title>
    <?php

        // Stampa i valori degli elementi di un array
        function stampa_val ($a) {
            foreach($a as $val) { // per ogni elemento
$a associa $val
                echo"$val, ";
// stampo ogni valore, apice doppio perchè permette l'in
interpretazione
            }
            echo'<br>';
        }

        // Stampa gli indici ed i valori degli element
i di un array
        function stampa_ind_val ($a) {
            foreach($a as $ind=>$val) {
                echo "$ind=>$val, ";
            }
            echo'<br>';
        }
    ?>
  </head>
  <body>
    <?php
        echo '<h1 style="text-align: center;">Esempio
di uso di foreach()</h1>';
```



```

    $a1=array(10,20,-45,78,3);
    echo '<h3>Array (10,20,-45,78,3)</h3>';

    // Stampa i valori degli elementi
    stampa_val($a1);

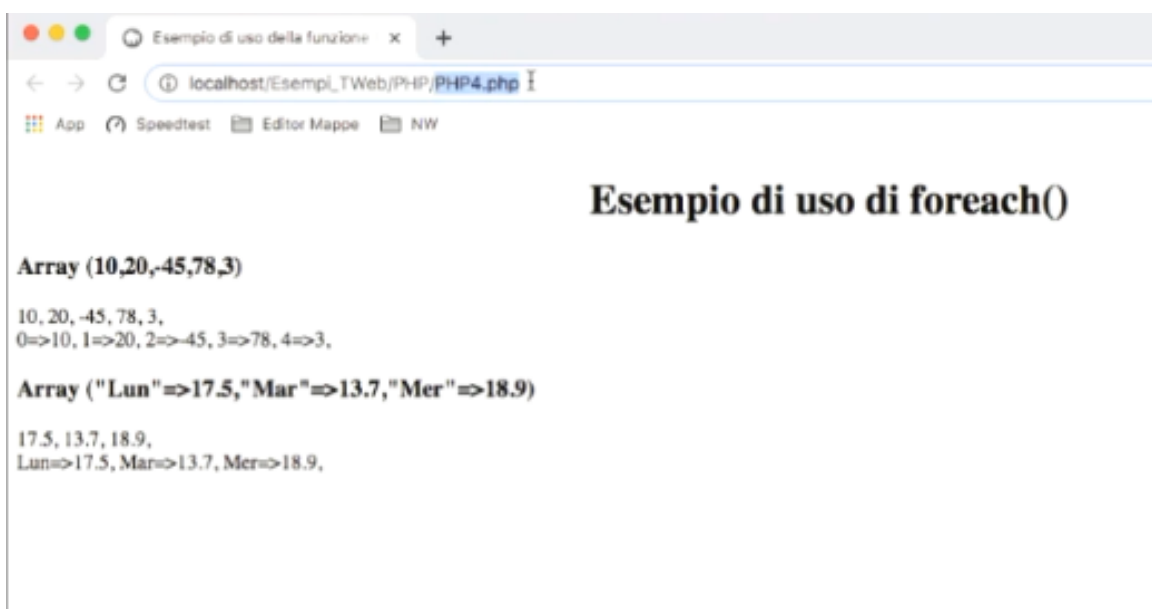
    // Stampa gli indici ed i valori degli element
i
    stampa_ind_val($a1);

    $a2=array("Lun"=>17.5,"Mar"=>13.7,"Mer"=>18.
9);
    echo '<h3>Array ("Lun"=>17.5,"Mar"=>13.7,"Me
r"=>18.9)</h3>';

    //Stampa i valori degli elementi
    stampa_val($a2);

    //Stampa gli indici ed i valori degli elementi
    stampa_ind_val($a2);
    ?>
</body>
</html>

```



Tutti gli array sono associativi, utilizzerà il valore della posizione (nell'array) come chiave (0⇒10, 1⇒20, ...)

Break, continue

- `break` consente di interrompere un ciclo iterativo prima che la condizione di fine ciclo sia verificata
- `continue` permette di passare direttamente alla successiva iterazione di un ciclo

Gestione contenuti delle FORM

Studiamo ora le peculiarità del PHP per quanto riguarda l'interazione Client-Server per la gestione dei contenuti web.

Un primo uso del PHP è quello che ci consente di acquisire (lato Server) il contenuto di una form HTML che l'utente ha riempito.

In HTML, quando una pagina contiene una

`form` l'utente riempie i contenuti della `form`, quando clicca sul bottone "Submit" i contenuti vengono trasformati in una stringa (tante coppie nome-valore legate dal simbolo `&`), la stringa viene inviata ad un server che specifichiamo nell'attributo `action` della form, cioè l'URL della risorsa a cui viene inviata la form. Ad esempio, al momento di una registrazione compilo una *FORM* con i miei dati, i quali verranno controllati per vedere se la mail è scritta correttamente, se la password è abbastanza complessa, se il nome utente è libero... Alla fine il server invierà al client una pagina di approvazione della registrazione.

Al programma specificato nell'URL dell'attributo `action` arriveranno i dati della form sotto forma di stringa, il programma dovrà quindi prendere la stringa e dividere i componenti legati dall' `&`, per ogni componente dovrà separare il campo del nome dal suo valore, se il valore contiene dei `+` andranno sostituiti con degli spazi, i codici speciali andranno sostituiti con i rispettivi caratteri, al fine di ottenere una stringa "pulita".

Questo processo è implementato di base dall'interprete PHP il quale, nel momento in cui un programma PHP viene attivato come richiesta di una *submit* di una form, non fa altro che processare la stringa che gli arriva e mettere a disposizione il suo contenuto all'interno di array associativi.

Questi specifici array associativi non vanno definiti dal programmatore, ma sono definite come

variabile superglobale predefinita, messe a disposizione a tutti i programmi in esecuzione nell'ambiente PHP.

La gestione dell'input dei dati provenienti da una FORM è estremamente semplice in PHP. Ad ogni elemento di `<form>` con `name=<nome>` corrisponde, in PHP, una variabile superglobale predefinita:

```
$_GET["<nome>"] -> Per il metodo GET
$_POST["<nome>"] -> Per il metodo POST
$_REQUEST["<nome>"] -> Per qualsiasi metodo
```

Ognuno di questi tre corrisponde al tipo di messaggio (definito nell'attributo `method` di una FORM).

- **GET** : la stringa della FORM viene inviata come un parametro dell'intestazione (*header*) del messaggio, il corpo è vuoto. (Generalmente pochi parametri).
Con questo metodo il dato viene visualizzato sulla barra di navigazione del browser (parametro della ricerca di Google), la modalità GET aggancia la stringa di ricerca all'URL della risorsa. Creo un solo indirizzo con l'URL della risorsa, un separatore `?` e la stringa del dato.
- **POST**: il dato viene inoltrato nel corpo (*body*) del messaggio (Generalmente molti parametri).
Con questo metodo il dato non è visibile perchè viaggia nel corpo, con implicazione della sicurezza del dato (non approfondiamo l'argomento).

In relazione al metodo di invio del messaggio al Server, viene popolato dall'interprete PHP l'array `$_GET["<nome>"]` o l'array `$_POST["<nome>"]`. Esiste l'array `$_REQUEST["<nome>"]` in cui il dato finisce sia se inviato in modalità GET sia se mandato in modalità POST, il dato lo trovo sempre, ma senza sapere il metodo. Se il metodo di inoltro non mi interessa, posso visualizzare il dato nel terzo array.

Gli array sono array associativi e contengono le coppie nome-valore della form utilizzando il valore dell'attributo `name` della form (univoco ed identificativo) come indice dell'array associativo. Se nella FORM ho un attributo il cui valore `name` è *nome*, avrò a disposizione nell'array `$_REQUEST[]` un elemento con chiave *nome*, se nella FORM ho un attributo il cui valore `name` è *cognome*, avrò a

disposizione nell'array `$_REQUEST[]` un elemento con chiave *cognome*, i valori corrispondenti saranno i valori inseriti dall'utente.

Potrei avere più elementi della FORM che condividono lo stesso nome (`<select name=<nome> multiple>`), sarebbe un problema, in quanto nell'indice avrei più elementi con lo stesso identificatore, come nel caso di più elementi nella check-box che condividono lo stesso identificatore.

In alcuni casi quindi gli array `$_GET["<nome>"]` , `$_POST["<nome>"]` e `$_REQUEST["<nome>"]` avranno, come valori, degli array di valori. Essendo il binding variabile-tipo dinamico io posso avere un array con elementi componenti di tipi di dati diversi (ad esempio stringhe, interi e array).

Per specificare la presenza di più elementi con lo stesso identificatore, è sufficiente assegnare ad essi un attributo `name="<nome>[]"` : la presenza delle parentesi quadre consente, quando la stringa arriva al PHP, di riconoscere quali elementi associare ad un array e quali non associare ad un array, è una convenzione, perché i valori immessi non sono più memorizzati come un singolo valore, come un array di valori nell'array PHP `$_<metodo>["<nome>"][]` .

▼ Esempio PHP#5

Variante della FORM delle pubblicazione di recensioni di libri.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Modulo di recensione</title>
  </head>
  <body style="background-color:#ffffcc">
    <h1 style="text-align: center;">Recensione di pu
bblicazione</h1>
    <p>
      Questa pagina può; essere utilizzata dai rec
ensori autorizzati per inviare all'editore le recensioni
dei testi a loro
      sottoposti.
    <br>
```

Si prega di compilare in ogni sua parte il modulo seguente

</p>

```
<form action="PHP5.php" method="post" target="_blank">
```

//la form va inviata al programma PHP5.php contenuto nella stessa cartella

```
<fieldset title="Inserisci i tuoi dati identificativi">
```

```
<legend><b>Sezione 1: Dati del Recensore</b></legend>
```

```
<label>Nome e Cognome <input name="nome" type="text" size="25" value="Mario Rossi" autofocus></label>
```

```
<label>Password di identificazione <input name="pass" type="password" maxlength="10" placeholder="massimo 10 caratteri"></label>
```

```
</fieldset>
```

```
<br>
```

```
<fieldset>
```

```
<legend><b>Sezione 2: Dati della Pubblicazione Recensita</b></legend>
```

```
<label>Titolo <input name="titolo" type="text" size="25"></label>
```

```
<label>Autore <input name="autore" type="text" size="25"></label>
```

```
<br>
```

```
<br>
```

```
<!-- Per usare correttamente il prossimo elemento bisogna -->
```

```
<!-- impostare ENCTYPE="multipart/form-data" nella definizione di FORM -->
```

```
<label>File contenente il testo <input name="testo" type="file" size="40"></label>
```

Ricevuta attraverso:

```
<label>Posta Ordinaria <input type="checkbox" name="mezzo[]" value="posta" checked></label>
```

```

        <label>e-mail <input type="checkbox" nam
e= "mezzo[]" value="email"></label>
        // ho due possibili scel
te, cioè piu check-box nello stesso gruppo di elementi
        // quindi specifico name
= "mezzo[]"
    </fieldset>
    <br>
    <fieldset>
        <legend><b>Sezione 3: Giudizio del Recen
sore</b></legend>
        <label>Illeggibile <input type="radio" n
ame="giudizio" value="ilg"></label>
        <label>Leggibile a Fatica <input id="lf"
type="radio" name="giudizio" value="lgf"></label>
        <label>Leggibile <input id="lg" type="ra
dio" name="giudizio" value="lgg" checked></label>
        <label>Piacevole <input type="radio" nam
e="giudizio" value="pcv"></label>
        <label>Capolavoro <input type="radio" na
me="giudizio" value="cpv"></label>
    </fieldset>
    <br>
    <fieldset>
        <legend><b>Sezione 4: Commento del Recen
sore</b></legend>
        <textarea name="Commento" rows="5" cols
="40">Inserisci un commento esteso alla pubblicazione</t
extarea>
    </fieldset>
    <br>
    <fieldset>
        <legend><b>Sezione 5: Indicazioni per
l'Editore</b></legend>
        <label for="pb">Il testo merita la pubbl
icazione?</label>
        <select id="pb" name="pub" size="1">
            <option value="no">No</option>

```

```

        <option value="fno">Forse no</option>
    >

    <option value="fsi" selected> Forse
si</option>

    <option value="si">Si</option>
</select>
<label for="dv">Se si, dove?</label>
<select id="dv" name="pub_dove[]" size
="2" multiple>

    <option value="ri">Rivista</option>
    <option value="li">Libro</option>
    <option value="mo">Collana Monografi
ca</option>

    <option value="ec">Edizioni Economic
he</option>

</select>
</fieldset>
<br>
<input type="submit" value="Invia">
<input type="reset" value="Azzera">
</form>
</body>
</html>

```

Recensione di pubblicazione

Questa pagina può essere utilizzata dai recensori autorizzati per inviare all'editore le recensioni dei testi a loro sottoposti.
Si prega di compilare in ogni sua parte il modulo seguente

Sezione 1: Dati del Recensore

Nome e Cognome Password di identificazione

Sezione 2: Dati della Pubblicazione Recensita

Titolo Autore

File contenente il testo Nessun file selezionato Ricevuta attraverso: Posta Ordinaria ☒ e-mail ☐

Sezione 3: Giudizio del Recensore

Illeggibile ☐ Leggibile a Fatica ☐ Leggibile ☐ ☒ Piacevole ☐ Capolavoro ☐

Sezione 4: Commento del Recensore

Inserisci un commento esteso alla pubblicazione

Sezione 5: Indicazioni per l'Editore

Il testo merita la pubblicazione? Se sì, dove?

Il programma `PHP5.php` è quello che riceve la FORM compilata dall'utente per effetto dell' `action`.

È un programma che creerà una pagina in cui saranno stampati gli elementi inseriti nella form per far verificare i dati all'utente.

Per recuperare il dato utilizzo l'array

`POST` (quello specificato nell'attributo `method` della form), ma avrei potuto utilizzare anche l'array `REQUEST` .

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Form e PHP</title>
  </head>

  <body>
    <h1 style="text-align: center;">Stampa dati form
con elementi a valore multiplo con programma PHP</h1>
    Nome= <?= $_POST["nome"] ?><br>
        <!-- stampa il valore di <?= $_POST["nom
e"] ?>, cioè il nome*/-->
    Password= <?= $_POST["pass"] ?><br>
    Titolo Pubblicazione= <?= $_POST["titolo"] ?><br
>
    Autore Pubblicazione= <?= $_POST["autore"] ?><br
>

    Nome del file= <?= $_POST["testo"] ?><br>
    Mezzo di inoltro del testo= <?php
        foreach($_POST["mezzo"] as $val)
            {echo "$val ";};
    ?>
    <br>
    <!--stampa tutti i valori di <?= $_POST["mezzo"] ?>, cio
è tutte le possibili scelte-->
    Giudizio= <?= $_POST["giudizio"] ?><br>
    Commento= <?= $_POST["Commento"] ?><br>
    Da pubblicare?= <?= $_POST["pub"] ?><br>
    Dove= <?php
        for($i=0;$i<sizeof($_POST["pub_dove"]);$i++)
            {echo $_POST["pub_dove"][$i];
```



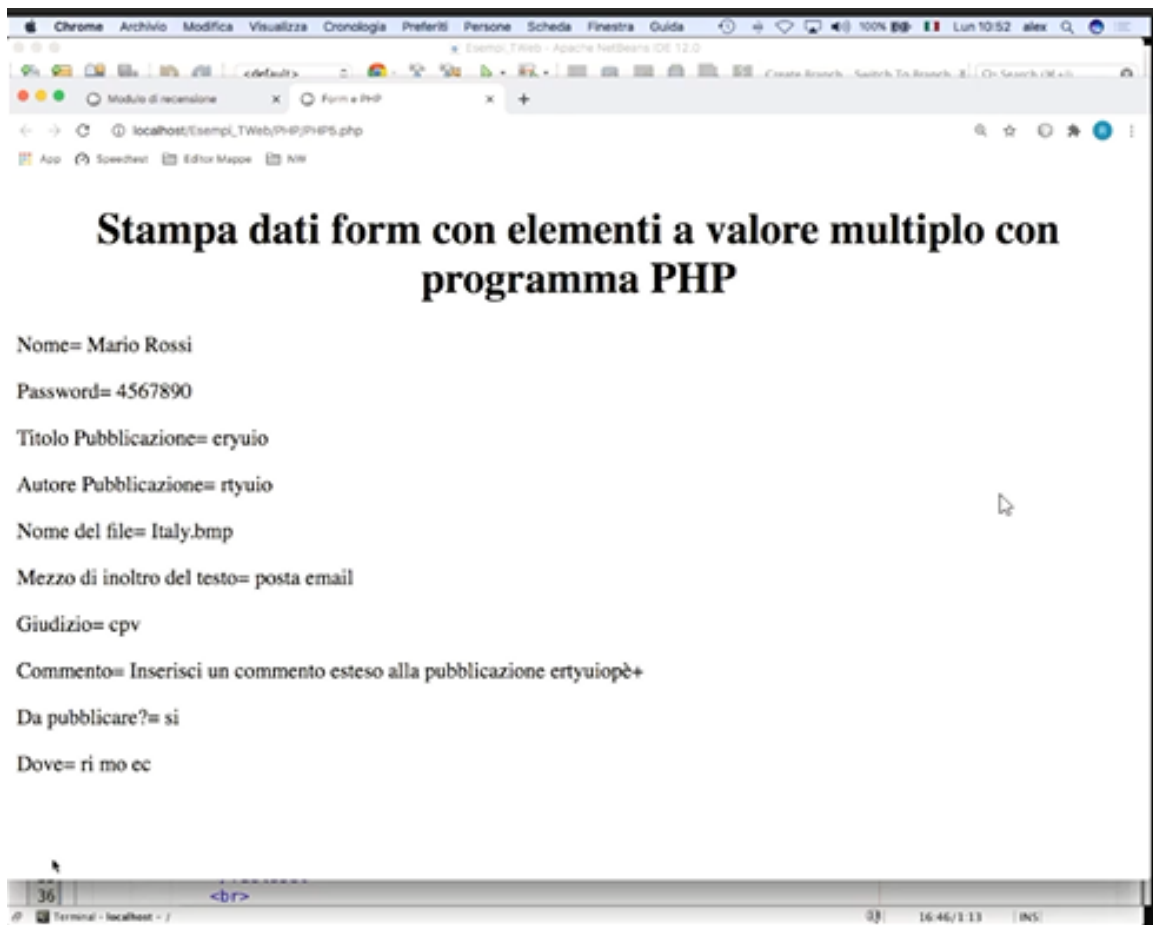
```

        echo ' ';
    }

    //ciclo equivalente alla foreach

?>
</body>
</html>

```



Passaggio di parametri

Con meccanismo analogo a quello usato per l'acquisizione di dati da una **form**, è possibile passare parametri ad un programma PHP (parametrizzazione delle chiamate a PHP).

In PHP possiamo implementare un algoritmo che in funzione delle condizioni in cui si trova ad operare produce un output specifico, in funzione di input che gli passo posso scatenare diversi algoritmi.

Il problema:

Nel caso in cui abbiamo più form concatenate (ad esempio nella stipulazione di una polizza auto) il server otterrà degli input a blocchi (prima la prima form, poi la seconda e così via) e deve sapere di quale form fanno parte i dati ricevuti in modo da inserirli nel giusto posto. Il programma che gestisce l'input, articolato in più form, è un programma unico, quindi per potergli dare contezza di quale dati delle form stia ricevendo potremmo parametrizzare la sua chiamata, cioè chiamare il programma con il parametro della form corrispondente.

Per una ricerca su Google chiamiamo sempre lo stesso programma, ma con parametri differenti.

Il parametro NON è il contenuto della FORM, ma è qualcosa che va aggiunto all'URL del programma che deve processare i dati della FORM attraverso la sua specifica nell' `action` della FORM stessa.

Per parametrizzare la chiamata ad un programma PHP che elabora il contenuto di una form, posso utilizzare un'aggiunta all'URL che identifica il programma da attivare (`HTTP://localhost/PHP6.php`) aggiungendo una stringa che inizia con `?` e aggiungendo i parametri attraverso coppie nome-valore (`?a=1&b=due`), scritti in maniera analoga alle stinghe di risposta della form (coppie separate da `&`, valori associati ai valori tramite `=`, spazi sostituiti da `+` e codici UNICODE per i caratteri speciali).

```
HTTP://localhost/PHP6.php ?a=1&b=due
```

Il programma PHP che riceve questa stringa non sa se sia il risultato di una form HTML o se qualcuno ha scritto questa stringa nell'indirizzo di ricerca del browser, il programma invocato *ipotizza* che sia l'inoltro di un dato di una FORM di tipo GET (la stringa della FORM viene inviata come un parametro dell'intestazione) e mappa tutte le coppie nome-valore presenti dopo il `?`. Posso utilizzare questa sintassi per invocare in maniera parametrica un programma PHP.

La soluzione:

Possiamo mettere (nella pagina che contiene la prima form delle mie form concatenate) nell' `action` della form l'indirizzo del programma PHP che deve analizzare la mia form, seguito da dei parametri che indichino in quale delle form concatenate io mi trovi (ad esempio: `HTTP://localhost/PHP6.php?a=1` per indicare che ci troviamo alla prima delle form concatenate, `a` è un nome scelto casualmente).

▼ Esempio PHP#6

Genera una pagina con un'intestazione e poi un corpo generato da codice PHP. Acquisisco attraverso una `foreach` tutte le coppie nome-valore e stampo queste coppie che altro non sono che i parametri passati nella sintassi dell'URL, dimostrando che i parametri passati sono realmente utilizzabili nel codice.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Passaggio parametri in PHP</title>
  </head>

  <body>
    <?php
      echo '<h1 style="text-align: center;">Parametri
i Passati al programma PHP</h1>';
      foreach($_GET as $ind=>$val) {
        echo "$ind => $val";
        echo '<br>';
      }
    ?>
  </body>
</html>
```

Nell'esempio l'URL è: `localhost/Esempi_Tweb/PHP/PHP6.php ?uno=23&due=pippo`



Interfacciamento a MySQL

Capire come interfacciarsi lato Server ad un DBMS (DataBase Management System), quest'ultimo costituirà un altro server della nostra applicazione al quale il Server web si collegherà come Client per accedere ai dati. Utilizzeremo il PHP per implementare l'accesso al nostro DBMS Server, nel nostro caso abbiamo scelto di utilizzare MySQL come DBMS.

PHP e DB

- È possibile configurare il linguaggio PHP per inglobare una serie di funzioni che consentono l'accesso alle primitive di gestione di un DataBase
- Nella fase di installazione del PHP, può essere specificato il DB al quale il linguaggio si interfaccia, scelto tra:
 - MySQL
 - Oracle
 - sqlite
 - ...

MySQL

Stabile, potente, facile da utilizzare e open source. È un database di classe enterprise, la conoscenza di MySQL non è una conoscenza a livello puramente accademico.

È un DBMS free, nato per piattaforme Linux, che supporta il linguaggio SQL, il quale è:

- Robusto
- Affidabile
- In grado di gestire DB di notevoli dimensioni
- Con una buona gestione della sicurezza dell'accesso

Fork del progetto: MariaDB (licenza GNU GPL)

Comandi MySQL		
<code>create database</code>	<code>insert</code>	<code>show tables</code>
<code>create index</code>	<code>join</code>	<code>show variables</code>
<code>create table</code>	<code>load data infile</code>	<code>update</code>
<code>delete</code>	<code>replace</code>	<code>use</code>
<code>drop database</code>	<code>select</code>	<code>...</code>
<code>drop index</code>	<code>show databases</code>	
<code>drop table</code>	<code>show index</code>	

In maniera molto semplificata potremmo dire che il database relazionale è un modo di organizzare la struttura in modo matriciale, creando una sorta di tabella con gli opportuni comandi per scrivere, modificare, cancellare ed accedere alle mie tabelle. Si possono mettere in relazioni diverse tabelle in caso di necessità (tabella dei numeri di telefono collegata a quella dei dati dell'interstatario del numero).

Le operazioni si dividono in comandi per la strutturazione del database (Data Definition Language, DDL) e quelli per modificare e manipolare la tabella (Data Manipulation Language, DML).

Esempi d'uso

```
create database biblio;
use biblio;
create table libri (titolo char(25), autore char(20), cod int,
prezzo double(5,2), data_vendita date);
describe libri;
insert into libri values('Il Nome della Rosa','Umberto Eco',1,24.32,'2013-04-18');
insert into libri (titolo,prezzo) values ('HTML',32.00);
select * from libri;
select titolo, cod from libri;
select * from libri where titolo='HTML' and prezzo > 30.00;
select * from libri where titolo like 'HTML%';
select * from libri order by titolo desc;
update libri set prezzo=34.00 where titolo='HTML';
delete from libri where titolo='HTML';
alter table libri change cod codice int not null;
alter table libri add unique(codice);
```

PHP e MySQL

Abbiamo bisogno di uno strumento che ci consenta nel linguaggio PHP di interfacciarsi con il server MySQL, questo strumento è una libreria, noi utilizziamo la libreria MySQLi (già integrata in XAMPP) che ha una serie di funzione per gestire la connessione tra PHP e il DBMS MySQL.

Queste librerie non fanno altro che simulare l'attività di un client, al momento della chiamata si configurano come un client MySQL il quale interagisce con il server host (nella nostra macchina). Le funzioni in rosso sono quelle che utilizzeremo

Funzioni PHP per l'accesso a MySQL

<code>mysqli_affected_rows()</code>	<code>mysqli_free_result()</code>
<code>mysqli_close()</code>	<code>mysqli_num_rows()</code>
<code>mysqli_connect()</code>	<code>mysqli_query()</code>
<code>mysqli_fetch_assoc()</code>	<code>mysqli_select_db()</code>
<code>mysqli_fetch_field_direct()</code>	...
<code>mysqli_field_count()</code>	

- `mysqli_connect()` : funzione principe, stabilisce la connessione tra il client MySQL che il nostro programma PHP emula con il server.
- `mysqli_close()` : funzione duale alla precedente, chiude la connessione. Ogni connessione aperta deve essere chiusa per evitare attacchi malevoli e occupazione di risorse. La chiusura della connessione mantiene comunque nell'ambiente PHP una serie di informazioni associate alla connessione stessa, vengono eliminate tramite un meccanismo di *garbage collection* che ogni tanto PHP fa partire.
- `mysqli_free_result()` : le informazioni mantenute temporaneamente in memoria costituiscono un carico importante di memoria occupata, è quindi importante esplicitare l'eliminazione dei dati temporanei con questa funzione, va sempre attivata per evitare la saturazione dello spazio.

MySQLi non implementa funzioni specifiche per la manipolazioni dei DB, ma ci mette a disposizione un'unica funzione

- `mysqli_query()` : ci consente di inviare una query SQL (istruzione) al server, andrà usata per qualunque azione che vogliamo compiere nel server.

Ci sono poi una serie di funzioni di servizio:

- `mysqli_num_rows()` : se chiamata dopo una query ci consente di contare le righe (cioè tuple) di una tabella ottenuta dall'ultima operazione. Utile per meccanismi di recupero dell'informazione.
- `mysqli_fetch_assoc()` : ogni volta che si attiva avremo che il record successivo a quello in cui ho effettuato l'accesso precedentemente viene recuperato e viene associato ad un array associativo in PHP, avrà come chiavi i nomi

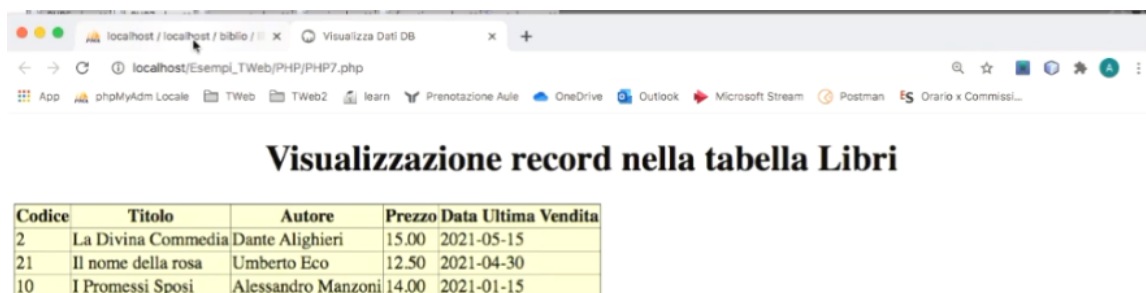
delle colonne (gli attributi) e come valori i valori della tabella. Costituisce un buon convertitore di formato.

▼ Esempio PHP#7

Esempio di come creare un contenuto a partire dalla mia tabella del DB. La prima cosa da fare è costruire, nel nostro ambiente server, una tabella MySQL all'interno della quale andiamo a memorizzare dei record di tipo *libri*. Abbiamo a disposizione anche *phpMyAdmin* (utilizzato in SIBDD). Nell'esempio accediamo al DB *biblio* che contiene due tabelle: *libri* e *utenti*. Analizziamo la tabella *libri*. Le funzioni base si conoscono dal corso di SIBDD. La tabella *libri* ha cinque attributi e tre tuple

			titolo	autore	codice	prezzo	data_vendita	
<input type="checkbox"/>	 Modifica	 Copia	 Elimina	La Divina Commedia	Dante Alighieri	2	15.00	2021-05-15
<input type="checkbox"/>	 Modifica	 Copia	 Elimina	Il nome della rosa	Umberto Eco	21	12.50	2021-04-30
<input type="checkbox"/>	 Modifica	 Copia	 Elimina	I Promessi Sposi	Alessandro Manzoni	10	14.00	2021-01-15
 <input type="checkbox"/> Seleziona tutto			Se selezionati:  Modifica  Copia  Elimina  Esporta					

Doppiamo costruire un programma che visualizza i record in una tabella di un DB.



Visualizzazione record nella tabella Libri

Codice	Titolo	Autore	Prezzo	Data Ultima Vendita
2	La Divina Commedia	Dante Alighieri	15.00	2021-05-15
21	Il nome della rosa	Umberto Eco	12.50	2021-04-30
10	I Promessi Sposi	Alessandro Manzoni	14.00	2021-01-15

Se modifichiamo i valori della nostra tabella del DB viene modificato anche nel nostro programma in maniera dinamica.

Viene definita una tabella in HTML, il codice PHP crea un ciclo iterativo che riempie i valori dei vari attributi per ogni tupla della tabella del DB.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Visualizza Dati DB</title>
  </head>
  <body>
    <h1 style="text-align: center;">Visualizzazi
one record nella tabella Libri</h1>
    <table style="background-color:#ffffcc; bord
er-collapse: collapse;" border>
      <tr><th>Codice</th><th>Titolo</th><th>Au
tore</th><th>Prezzo</th><th>Data Ultima Vendita</th></tr>
    >

    <?php
      include("include/connect.php"); // includo
file con dati necessari per l'accesso
      $conn=mysqli_connect($HOST, $USER, $PASSWOR
D,$DB); //creiamo connessione
      //salvo la connessione come una
variabile $conn che utilizzerò alla chiusura

      $ris=mysqli_query($conn, "select
* from libri");

      //lancio query per ottenere le in
formazioni sui libri

      //$ris oggetto del risultato del
la query

      while ($row=mysqli_fetch_assoc
($ris)) {
        //passo risposta della query
come parametro della funzione
```

```

//estrae il risultato della
riga successiva ad ogni invocazione
//ritorna l'array associativ
o -> $row

//il ciclo continua fino a q
uando ci sono righe nella tabella del DB

$co=$row["codice"]; //valore
della chiave "codice" -> attributo della tabella
$ti=$row["titolo"];
$au=$row["autore"];
$pr=$row["prezzo"];
$dv=$row["data_vendita"];
echo "<tr><td>$co</td><td>$ti</t
d><td>$au</td><td>$pr</td><td>$dv</td></tr>";
    }

mysqli_free_result($ris); //li
bero lo spazio occupato dai miei dati
mysqli_close($conn); //chiudo la connessio
ne

?>
</table>
</body>
</html>

```

```
$conn=mysqli_connect($HOST, $USER, $PASSWORD, $DB);
```

- Creiamo una connessione tra client e server
- Definiamo l'host, nel nostro caso è la nostra stessa macchina, ma non è detto che sia sempre così
- Inseriamo l'username e la password definiti da chi gestisce la base di dati
- Specifichiamo il nome del DB

Questi quattro dati sono generalmente contenuti in un unico file che definisce le stringhe usate nei vari programmi

```
<!-- connect.php -->

<?php
    $USER="tweb";
    $DB="biblio";
    $HOST="localhost";
    $PASSWORD="tweb";
```

Questo file ha la tag di apertura, ma non ha quella di chiusura, questa è una regola di sicurezza. Se io ho un codice scritto interamente in PHP è buona regola non mettere la tag di chiusura perchè si presta ad attacchi malevoli.

Gestione Integrata

Variante della struttura a blocchi vista precedentemente, la pagina mette a disposizione le funzioni *Visualizza*, *Inserisci* e *Modifica*.

Visualizza

Visualizza la tabella *libri* del mio DB *biblio*.

mauris. Maecenas vitae nunc mauris. Phasellus risus augue, cursus eget vestibulum sed, ultricies non justo. Duis aliquet dui tincidunt est sagittis fermentum. Praesent nec feugiat tortor. Sed elit erat, tempus ullamcorper pulvinar sed, placerat at augue.

Funzioni

- Visualizza
- Inserisci
- Modifica

Visualizzazione record nella tabella Libri

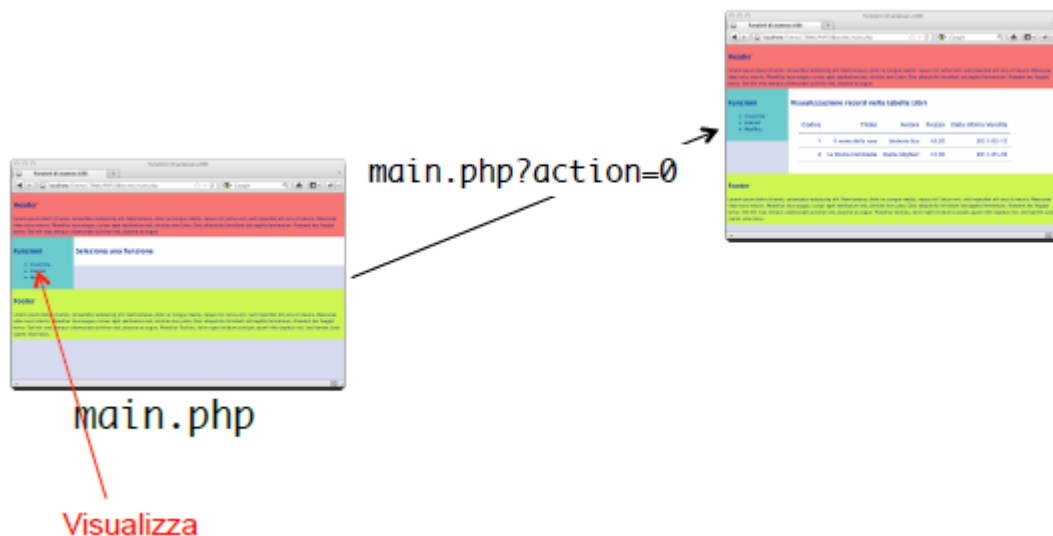
Codice	Titolo	Autore	Prezzo	Data Ultima Vendita
2	La Divina Commedia	Dante Alighieri	16.00	2021-05-15
21	Il nome della rosa	Umberto Eco	12.50	2021-04-30
10	I Promessi Sposi	Alessandro Manzoni	14.00	2021-01-15

Footer

Nella prima rotta specifichiamo quale sarà il parametro della chiamata di tipo GET.

Sarà

`main.php?action=0`, chiamerà cioè lo stesso programma che genera la homepage passandogli come parametro `action=0`. Progettiamo quindi un parametro `action` che se avrà valore uguale a `0` determinerà l'apertura del programma che si dedica alla visualizzazione della tabella.



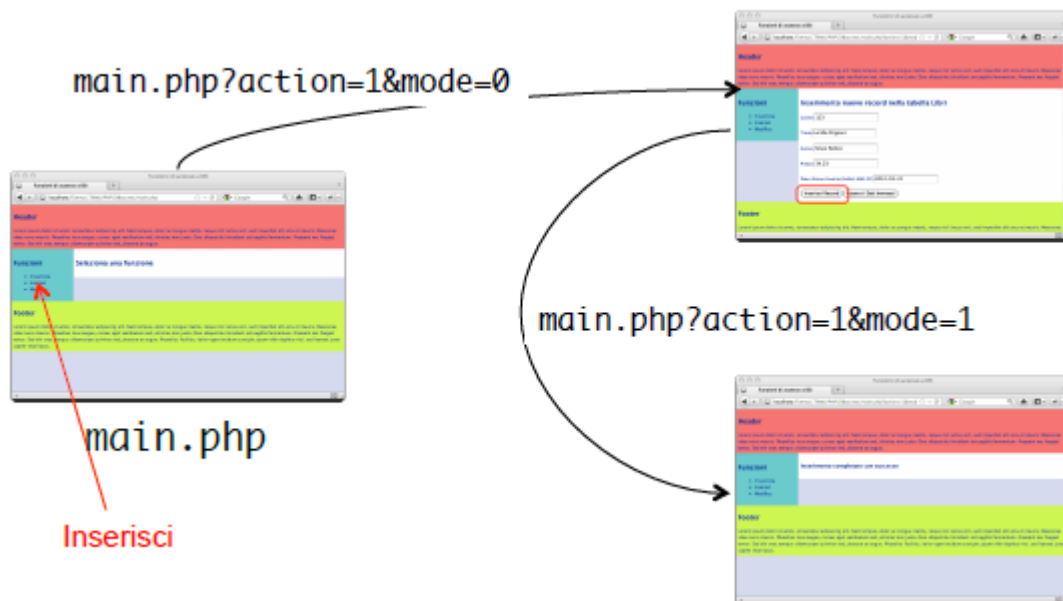
Inserisci

FORM che ci consente di inserire un nuovo elemento nella tabella *libri*.

Nel caso dell'inserimento abbiamo il secondo parametro da istanziare che assume i valori 0 e 1 relativamente al primo e al secondo step impostato.

The image shows two screenshots of a web form. The left screenshot shows the 'Inserimento nuovo record nella tabella Libri' form. It has a sidebar with 'Funzioni' (Visualizza, Inserisci, Modifica) and a main area with input fields for 'Codice', 'Titolo', 'Autore', 'Prezzo', and 'Data Ultima Vendita (AAAA-MM-GG)'. There are buttons for 'Inserisci Record' and 'Azzerare i Dati Immessi'. The right screenshot shows the 'Inserimento completato con successo' message. It has a sidebar with 'Funzioni' (Visualizza, Inserisci, Modifica) and a main area with a success message. Both screenshots have a 'Header' and 'Footer' section with placeholder text.

Nella seconda funzione dobbiamo implementare entrambe le schermate, implementando un secondo parametro. Il primo indica l'azione e la seconda implementa la fase dell'azione.



Modifica

Seleziono quale elemento modificare

Header

Funzioni

- Visualizza
- Inserisci
- Modifica

Sceita del record nella tabella Libri da modificare

Seleziona	Codice	Titolo	Autore	Prezzo	Data Ultima Vendita
<input type="radio"/>	2	La Divina Commedia	Dante Alighieri	16.00	2021-05-15
<input type="radio"/>	21	Il nome della rosa	Umberto Eco	12.50	2021-04-30
<input type="radio"/>	10	I Promessi Sposi	Alessandro Manzoni	14.00	2021-01-15
<input type="radio"/>	37	ertyuio	uio	123.00	2021-04-20

Modifica

Una volta selezionato l'elemento mi appare la FORM per modificarlo. Il codice è non modificabile

Header

Funzioni

- Visualizza
- Inserisci
- Modifica

Modifica record nella tabella Libri

Titolo: ertyuio

Autore: uio

Prezzo: 123.00

Data Ultima Vendita (AAAA-MM-GG): 2021-04-20

Modifica Dati Azzera i Dati Immessi

Footer

Header

Funzioni

- Visualizza
- Inserisci
- Modifica

Modifica completata con successo

Footer

L'azione in questo caso si divide in tre fasi :



Primo passaggio: ci consente di selezionare l'elemento della tabella mysql da modificare in un elenco che viene proposto a video con tutti quanti i valori

Secondo passaggio: l'elemento selezionato ci viene riproposto in termini di contenuti all'interno di una form nella quale possiamo modificare le componenti del nostro record

Terzo passaggio: prende i dati inseriti nella form della seconda fase, li aggiorna sul database e infine ritorna il messaggio di ok.

DBAccess

Il codice è diviso in tre parti.

▼ main.php

Similare a quello analizzato negli esempi precedenti

```
<?php include("functions.php"); ?>
<!DOCTYPE HTML>
<html>
  <head>
    <title>Funzioni di accesso a DB</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="sty
```

```

le.css">
    </head>
    <body>
        <div id="header">
            <h2>Header</h2>
            Lorem ipsum dolor sit amet, consectetur adip
            iscing elit. Nam tempus, dolor ac congue mattis, neque n
            isl luctus orci, sed imperdiet
            elit arcu et mauris. Maecenas vitae nunc mau
            ris. Phasellus risus augue, cursus eget vestibulum sed,
            ultricies non justo. Duis aliquet
            dui tincidunt est sagittis fermentum. Praese
            nt nec feugiat tortor. Sed elit erat, tempus ullamcorper
            pulvinar sed, placerat at augue.
        </div>
        <div id="main">
            <div id="navbar">
                <h2>Funzioni</h2>
                <ul>
                    <!--creiamo le ancore per le rotte d
ei programmi-->
                    <li><a href="main.php?action=0">Visu
alizza</a></li>
                    <li><a href="main.php?action=1&mode=
0">Inserisci</a></li>
                    <li><a href="main.php?action=2&mode=
0">Modifica</a></li>
                </ul>
            </div>
            <div id="content">
                <?php
                    if (isset($_GET["action"])) { //controll
o se ci sono parametri action o no
                        require("functions.php"); //si inter
rompe il programma se require() non lo trova
                        //include() invece si usa quando pos
so fare a meno di un programma
                        switch ($_GET["action"]) { //control

```

```

lo quale parametro

        case 0: //visualizza
            show();
            break;
        case 1: //inserisci
            insert();
            break;
        case 2: //modifica
            update();
            break;
        default:
            show(); // lo chiamo se ad e
sempio metto action=7
    }
    } else echo '<h2>Seleziona una funzione
</h2>';

    ?>
</div>
</div>
<div id="footer">
    <h2>Footer</h2>
    Lorem ipsum dolor sit amet, consectetur adip
iscing elit. Nam tempus, dolor ac congue mattis, neque n
isl luctus orci, sed
        imperdiet elit arcu et mauris. Maecenas vita
e nunc mauris. Phasellus risus augue, cursus eget vestib
ulum sed, ultricies non
        justo. Duis aliquet dui tincidunt est sagitt
is fermentum. Praesent nec feugiat tortor. Sed elit era
t, tempus ullamcorper
        pulvinar sed, placerat at augue. Phasellus f
acilisis, dolor eget tincidunt suscipit, quam nibh dapib
us nisi, sed laoreet
        justo sapien vitae lacus.
    </div>
</body>
</html>

```


▼ functions.php

```
<?php
//funzione di visualizzazione
function show() {
    include("../include/connect.php"); //percorso diverso rispetto a Esempio PHP#7
    echo '<h2>Visualizzazione record nella tabella Libri</h2>';
    echo '<table>';
    echo '<tr><th>Codice</th><th>Titolo</th><th>Autore</th><th>Prezzo</th><th>Data Ultima Vendita</th></tr>';

    // Estrazione dati da DB e inserimento in righe della tabella
    $conn=mysqli_connect($HOST, $USER, $PASSWORD,$DB);
    $ris=mysqli_query($conn, "select * from libri");
    while ($row=mysqli_fetch_assoc($ris)) {
        $co=$row["codice"];
        $ti=$row["titolo"];
        $au=$row["autore"];
        $pr=$row["prezzo"];
        $dv=$row["data_vendita"];
        echo "<tr><td>$co</td><td>$ti</td><td>$au</td><td>$pr</td><td>$dv</td></tr>";
    }
    mysqli_free_result($ris);
    mysqli_close($conn);
    echo '</table>';
}

//funzione di inserimento
function insert() {
    //suddivisa in due sezioni a seconda del parametro mode
    if ($_GET["mode"]==0) {

        // modo 0: mostra la form di inserimento dati
```

```

        echo '<h2>Inserimento nuovo record nella tabella
Libri</h2>';
        echo '<form action="main.php?action=1&mode=1" me
thod="post">';
        echo '<label>Codice <input type="text" name="c
o"></label><br><br>';
        echo '<label>Titolo <input type="text" name="t
i"></label><br><br>';
        echo '<label>Autore <input type="text" name="a
u"></label><br><br>';
        echo '<label>Prezzo <input type="text" name="p
r"></label><br><br>';
        echo '<label>Data Ultima Vendita (AAAA-MM-GG) <i
nput type="text" name="dv"></label><br><br>';
        echo '<input type="submit" value="Inserisci Reco
rd">';
        echo '<input type="reset" value="Azzera i Dati I
mmessi">';
        echo '</form>';
    }
    elseif ($_GET["mode"]==1) {

        // modo 1: inserisci i dati provenienti dalla fo
rm nel DB
        include("../include/connect.php");
        $conn=mysqli_connect($HOST, $USER, $PASSWOR
D,$DB);

        //      $ris=mysqli_query($conn, "SET CHARACTER SET 'utf
8'");

        //comando viene definito come l'unione di tre st
ringhe
        $comando="insert into libri (titolo, autore, cod
ice, prezzo, data_vendita) ";
        $comando.="values ('" .$_POST["ti"]."', '" . $_PO
ST["au"]."', '" .$_POST["co"]."', '" .
        //i valori li prendiamo dai dati inseriti nella

```

```

FORM tramite il metodo POST
    //ATTENZIONE all'uso degli apici doppi e semplic
i per generare correttamente la stringa
    //ognuno delle 7 stringhe colorate è una stringa
diversa concatenate tra loro

    $comando=$_POST["pr"].'', '$_POST["dv"].''');
    $ris=mysqli_query($conn, $comando);
    mysqli_close($conn);
    echo '<h3>Inserimento completato con successo</h
3>';
    }
}

function update() {
    include("../include/connect.php");
    if ($_GET["mode"]==0) {

        // Visualizza i record per la selezione di quell
o da modificare
        echo '<h2>Scelta del record nella tabella Libri
da modificare</h2>';
        echo '<form action="main.php?action=2&mode=1" me
thod="post">';
        echo '<table>';
        echo '<tr><th>Seleziona</th><th>Codice</th><th>T
itolo</th><th>Autore</th><th>Prezzo</th><th>Data Ultima
Vendita</th></tr>';

        // Estrazione dati da DB e inserimento in righe
della tabella
        $conn=mysqli_connect($HOST, $USER, $PASSWOR
D,$DB);
        $ris=mysqli_query($conn, "select * from libr
i");
        while ($row=mysqli_fetch_assoc($ris)) {
            $co=$row["codice"];
            $ti=$row["titolo"];

```

```

        $au=$row["autore"];
        $pr=$row["prezzo"];
        $dv=$row["data_vendita"];
        echo '<tr><td><input type ="radio" name="rmod" value="'. $co. '"></td>';
        echo "<td>$co</td><td>$ti</td><td>$au</td><td>$pr</td><td>$dv</td></tr>";
        //ad ogni elemento associo un radio button con un corrispondente valore identificativo
        //il codice identificativo viene passato alla mode=1 per modificare il libro
    }
    mysqli_free_result($ris);
    mysqli_close($conn);
    echo '</table>';
    echo '<input type="submit" value="Modifica">';
    echo '</form>';
}

if ($_GET["mode"]==1) {

    // Estrai i dati dal DB
    $conn=mysqli_connect($HOST, $USER, $PASSWORD, $DB);
    $comando="select * from libri where codice=".$POST["rmod"];
    $ris=mysqli_query($conn,$comando);
    $row=mysqli_fetch_assoc($ris);
    $oti=$row["titolo"];
    $oau=$row["autore"];
    $opr=$row["prezzo"];
    $odv=$row["data_vendita"];
    mysqli_free_result($ris);
    mysqli_close($conn);

    // Visualizza i dati, permetti la modifica e attiva il programma di aggiornamento
    echo '<h2>Modifica record nella tabella Libri</h2>';
}

```

```

2>';
        echo '<form action="main.php?action=2&mode=2" method="post">';

        //per portare il valore del codice alla fase tre
        senza renderlo visibile lo nascondo
        echo '<input type="hidden" value="'.$_POST["rmod"].'" name="co">';

        //ad ogni elemento della FORM associo il valore
        attuale dell'elemento
        echo '<label>Titolo <input type="text" value="'.
        $_POST["ti"].'" name="ti"></label><br><br>';
        echo '<label>Autore <input type="text" value="'.
        $_POST["au"].'" name="au"></label><br><br>';
        echo '<label>Prezzo <input type="text" value="'.
        $_POST["pr"].'" name="pr"></label><br><br>';
        echo '<label>Data Ultima Vendita (AAAA-MM-GG) <input type="text" value="'.
        $_POST["dv"].'" name="dv"></label><br><br>';
        echo '<input type="submit" value="Modifica Dati">';
        echo '<input type="reset" value="Azzera i Dati Immessi">';
        echo '</form>';
    }

    elseif ($_GET["mode"]==2) {

        // Modifica i dati nel DB
        $conn=mysqli_connect($HOST, $USER, $PASSWORD, $DB);

        //comando sql di update
        //il campo del del codice non viene modificato, ma si eredita quello già messo
        $comando="update libri set codice=".$_POST["co"].", titolo='".$_POST["ti"];
        $comando.="', autore='".$_POST["au"]."', prezzo

```

```

o=".$_POST["pr"];
        $comando.="data_vendita='".$_POST["dv"]."'
where codice=".$_POST["co"];
        mysqli_query($conn, $comando);
        mysqli_close($conn);
        echo "<h3>Modifica completata con successo</h3
>";
    }
}

```

ATTENZIONE: tutte le funzioni che si interfacciano a MySQLi devono essere inserite in un contesto che gestisca eventualmente un errore che può essere dovuto al DB offline, ad una query scritta male etc.

In questi esempi non si bada alla sicurezza, è facile inserire un codice PHP malevolo all'interno di una FORM che alteri il funzionamento del programma, in questi casi si parla di PHP injection.

▼ style.css

```

body {
    margin: 0;
    background-color: #d6daed;
    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
    color: #003399;
    font-size: 10px;
}

#header {
    background-color: #FF7373;
    padding: 0.5em;
}

#main {
    margin-left: 15em;
}

#navbar {

```

```

    background-color: #5CCCCC;
    width: 14em;
    padding: 1em 0.5em;
    float: left;
    margin-left: -15em;
}

#content {
    background-color: #FFFFFF;
    border-left: 1px dotted #5CCCCC;
    padding: 1em 0.5em;
}

#footer {
    background-color: #C9F76F;
    padding: 0.5em;
    clear: left;
}

h1, h2, h3 {
    font-family: "Lucida Sans Unicode", "Lucida Grande", S
ans-Serif;
}

a {
    color: #003399;
}

a:link {
    text-decoration: none
}

table {
    border-collapse: collapse;
    font-family: "Lucida Sans Unicode", "Lucida Grande", S
ans-Serif;
    font-size: 12px;
    margin: 20px;
}

```

```

        text-align: right;
    }

    th {
        border-bottom: 2px solid #6678B1;
        color: #003399;
        font-size: 14px;
        font-weight: normal;
        padding: 10px 8px;
    }

    td {
        border-bottom: 1px solid #CCCCCC;
        padding: 6px 8px;
    }
}

```

Accesso autorizzato

Problema:

Implementare un meccanismo per l'accesso alle pagine ai soli utenti autorizzati.

Soluzione:

- Definire il profilo degli utenti del sito
- Identificare in modo permanente gli utenti che accedono al sito (Autenticazione)
- Generare le pagine ad accesso controllato solo dopo aver verificato che l'utente collegato sia autorizzato (Autorizzazione)

Autenticazione : meccanismo che consente al sito di identificare un utente a partire da un elenco (di utenti che il sito reperisce da qualche parte) come un utente riconosciuto dal sito stesso e che ha quindi dei privilegi rispetto ad altri utenti del sito. Servono due cose:

1. Un'informazione persistente che permane anche se il server si spegne che è una tabella MySQL nel quale salviamo i profili degli utenti che nel sito sono riconosciuti (la tabella contiene le credenziali che consentono all'utente di autenticarsi).

2. Costruire un meccanismo lato server che consenta di mantenere memoria di un certo client. Una volta fatto il login il sito deve avere memoria di noi e interpretare ogni nostra richiesta come una richiesta di un utente autenticato. Questo fintanto che non facciamo il logout o non scada la sessione.

Autorizzazione : una volta che l'utente si è autenticato dobbiamo filtrare il contenuto del nostro sito per fare in modo che certi contenuti siano visibili solo a certi livelli di utenza. Se nel nostro sito l'utente cliente può comprare biglietti il bottone "compra" appare solo se chi ha aperto la pagina è un utente che si è autenticato come cliente, altrimenti non appare. Creare due pagine diverse, una per chi è registrato e una per chi non è registrato, non va bene perché l'utente non registrato potrebbe cogliere la chiamata copiando la parametrizzazione che attiva la seconda pagina.

Inoltre possiamo categorizzare i vari utenti, la classificazione si gestisce aggiungendo una proprietà chiamata "*ruolo*" che viene assegnato quando l'utente si registra o gestiti in maniera diretta da noi.

Session

Il meccanismo della sessione serve a mantenere la persistenza dell'informazione fra diverse sessioni HTTP (sessione nel protocollo HTTP: richiesta risposta e fine, il concetto di sessione nel PHP: meccanismo legato alla sessione PHP che introduce una memoria). L'istruzione `session_start()` va attivata in tutte le pagine del nostro sito che devono usare il meccanismo delle sessioni e va attivata come prima istruzione assoluta sia della sezione PHP che HTML. Per mantenere la memoria il PHP usa due meccanismi, uno lato server e uno lato client. Per quello lato server abbiamo che per ogni client che si collega alla mia applicazione l'ambiente server associa un file univoco nella memoria del server nel quale il programma che usa questo file può memorizzare delle informazioni che, essendo associate al client che ha fatto richiesta, ed essendo memorizzate su file, sono persistenti. Tutte le volte che lo stesso client si collega alla nostra applicazione, la nostra applicazione può accedere allo stesso file e recuperare delle informazioni che sono memorizzate nel file. Nel momento in cui l'utente compila la form username e password noi potremo lato server attivare un programma che verifica che i dati inseriti siano corrispondenti a un utente esistente e una volta verificata questa condizione il nostro programma estrarrà dalla tabella utente il ruolo dell'utente e memorizzare il file di quel client, i dati e il ruolo dell'utente.

▼ main.php

```
<?php
    session_start();
    include("functions.php");
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Funzioni di accesso a DB con sessioni</title>
        <meta charset="UTF-8">
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <div id="header">
            <h2>Header</h2>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam tempus, dolor ac congue mattis, neque nisl luctus orci, sed imperdiet elit arcu et mauris. Maecenas vitae nunc mauris. Phasellus risus augue, cursus eget vestibulum sed, ultricies non justo. Duis aliquet tincidunt est sagittis fermentum. Praesent nec feugiat tortor. Sed elit erat, tempus ullamcorper pulvinar sed, placerat at augu.
        </div>
        <div id="main">
            <div id="navbar">
                <?php include("navbar.php")?>
            </div>
            <div id="content">
                <?php
                    if (isset($_GET["action"])) {
                        switch ($_GET["action"]) {
                            case 0:
                                show();
                        }
                    }
                </?php>
            </div>
        </div>
    </body>
</html>
```

```

                break;
            case 1:
                insert();
            break;
            case 2:
                update();
            break;
            case 99:
                echo '<h2>' . $_GET["msg"] . '</h2>';

                break;
            default:
                show();
        }
    } else echo '<h2>Seleziona una funzione
</h2>';

    ?>
</div>
</div>
<div id="footer">
    <h2>Footer</h2>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam tempus, dolor ac congue mattis, neque nisl luctus orci, sed imperdiet elit arcu et mauris. Maecenas vitae nunc mauris. Phasellus risus augue, cursus eget vestibulum sed, ultricies non justo. Duis aliquet dui tincidunt est sagittis fermentum. Praesent nec feugiat tortor. Sed elit erat, tempus ullamcorper pulvinar sed, placerat at augue. Phasellus facilisis, dolor eget tincidunt suscipit, quam nibh dapibus nisi, sed laoreet justo sapien vitae lacus.
</div>
</body>
</html>

```

▼ functions.php

```
<?php
function show() {
    include("../include/connect.php");
    echo '<h2>Visualizzazione record nella tabella Libri
</h2>';
    echo '<table>';
    echo '<tr><th>Codice</th><th>Titolo</th><th>Autore</
th><th>Prezzo</th><th>Data Ultima Vendita</th></tr>';

    // Estrazione dati da DB e inserimento in righe dell
a tabella
    $conn=mysqli_connect($HOST, $USER, $PASSWORD,$DB);
    $ris=mysqli_query($conn, "select * from libri");
    while ($row=mysqli_fetch_assoc($ris)) {
        $co=$row["codice"];
        $ti=$row["titolo"];
        $au=$row["autore"];
        $pr=$row["prezzo"];
        $dv=$row["data_vendita"];
        echo "<tr><td>$co</td><td>$ti</td><td>$au</t
d><td>$pr</td><td>$dv</td></tr>";
    }
    mysqli_free_result($ris);
    mysqli_close($conn);
    echo '</table>';
}

function insert() {
    if (!isset($_SESSION["auth"]) || $_SESSION["auth"] !
= 'admin') {
        echo '<h2>Funzione non disponibile</h2>';
        return;
    }
    if ($_GET["mode"]==0) {

        // modo 0: mostra la form di inserimento dati
```

```

        echo '<h2>Inserimento nuovo record nella tabella
Libri</h2>';
        echo '<form action="main.php?action=1&mode=1" me
thod="post">';
        echo '<label>Codice <input type="text" name="c
o"></label><br><br>';
        echo '<label>Titolo <input type="text" name="t
i"></label><br><br>';
        echo '<label>Autore <input type="text" name="a
u"></label><br><br>';
        echo '<label>Prezzo <input type="text" name="p
r"></label><br><br>';
        echo '<label>Data Ultima Vendita (AAAA-MM-GG) <i
nput type="text" name="dv"></label><br><br>';
        echo '<input type="submit" value="Inserisci Reco
rd">';
        echo '<input type="reset" value="Azzera i Dati I
mmessi">';
        echo '</form>';
    }
    elseif ($_GET["mode"]==1) {

        // modo 1: inserisci i dati provenienti dalla fo
rm nel DB
        include("../include/connect.php");
        $conn=mysqli_connect($HOST, $USER, $PASSWOR
D,$DB);
        $comando="insert into libri (titolo, autore, cod
ice, prezzo, data_vendita) ";
        $comando.="values ('".$_POST["ti"]."', '". $_POS
T["au"]."', '". $_POST["co"]."', '";
        $comando.="$_POST["pr"]."', '$_POST["dv"]."')";
        $ris=mysqli_query($conn, $comando);
        mysqli_close($conn);
        echo '<h3>Inserimento completato con successo</h
3>';
    }
}

```

```

function update() {
    if (!isset($_SESSION["auth"]) || $_SESSION["auth"] != 'admin') {
        echo '<h2>Funzione non disponibile</h2>';
        return;
    }
    include("../include/connect.php");
    if ($_GET["mode"]==0) {

        // Visualizza i record per la selezione di quello da modificare
        echo '<h2>Scelta del record nella tabella Libri da modificare</h2>';
        echo '<form action="main.php?action=2&mode=1" method="post">';
        echo '<table>';
        echo '<tr><th>Seleziona</th><th>Codice</th><th>Titolo</th><th>Autore</th><th>Prezzo</th><th>Data Ultima Vendita</th></tr>';

        // Estrazione dati da DB e inserimento in righe della tabella
        $conn=mysqli_connect($HOST, $USER, $PASSWORD,$DB);
        $ris=mysqli_query($conn, "select * from libri");
        while ($row=mysqli_fetch_assoc($ris)) {
            $co=$row["codice"];
            $ti=$row["titolo"];
            $au=$row["autore"];
            $pr=$row["prezzo"];
            $dv=$row["data_vendita"];
            echo '<tr><td><input type="radio" name="record" value="'. $co. '"></td>';
            echo "<td>$co</td><td>$ti</td><td>$au</td><td>$pr</td><td>$dv</td></tr>";
        }
        mysqli_free_result($ris);
        mysqli_close($conn);
    }
}

```

```

echo '</table>';
echo '<input type="submit" value="Modifica">';
echo '</form>';
}

if ($_GET["mode"]==1) {

    // Estrai i dati dal DB
    $conn=mysqli_connect($HOST, $USER, $PASSWORD, $DB);

    $ris=mysqli_query($conn, "select * from libri where codice=$_REQUEST[rmod]");
    $row=mysqli_fetch_assoc($ris);
    $oti=$row["titolo"];
    $oau=$row["autore"];
    $opr=$row["prezzo"];
    $odv=$row["data_vendita"];
    mysqli_free_result($ris);
    mysqli_close($conn);

    // Visualizza i dati, permetti la modifica e attiva il programma di aggiornamento
    echo '<h2>Modifica record nella tabella Libri</h2>';

    echo '<form action="main.php?action=2&mode=2" method="post">';
    echo '<input type="hidden" value="'. $_REQUEST["rmod"].'" name="co">';
    echo '<label>Titolo <input type="text" value="'. $oti.'" name="ti"></label><br><br>';
    echo '<label>Autore <input type="text" value="'. $oau.'" name="au"></label><br><br>';
    echo '<label>Prezzo <input type="text" value="'. $opr.'" name="pr"></label><br><br>';
    echo '<label>Data Ultima Vendita (AAAA-MM-GG) <input type="text" value="'. $odv.'" name="dv"></label><br><br>';

    echo '<input type="submit" value="Modifica Dati">';
}

```

```

i">';
        echo '<input type="reset" value="Azzera i Dati Immessi">';
        echo '</form>';
    }

    elseif ($_GET["mode"]==2) {

        // Modifica i dati nel DB
        $conn=mysqli_connect($HOST, $USER, $PASSWORD,$DB);

        $comando="update libri set codice=".$_POST["co"].",titolo='".$_POST["ti"];
        $comando.="',autore='".$_POST["au"]."',prezzo=".$_POST["pr"];
        $comando.="data_vendita='".$_POST["dv"]."'
        where codice=".$_POST["co"];
        mysqli_query($conn, $comando);
        mysqli_close($conn);
        echo "<h3>Modifica completata con successo</h3>";
    }
}

```

▼ login.php

```

<?php
session_start();
include("../include/connect.php");
$conn=mysqli_connect($HOST, $USER, $PASSWORD,$DB);
$ris=mysqli_query($conn, "select * from utenti where user='".$_POST["user"]."");
if (!mysqli_num_rows($ris)) {
    mysqli_close($conn);
    header("Location: main.php?action=99&msg=Errore:utente+sconosciuto");
    exit;
}

```



```

$row=mysqli_fetch_assoc($ris);
if ($row["pass"]!=$_POST["pass"]) {
    mysqli_free_result($ris);
    mysqli_close($conn);
    header("Location: main.php?action=99&msg=Errore:+pas
sword+errata");
    exit;
}
$_SESSION['user']=$row["user"];
$_SESSION['auth']=$row["cat"];
mysqli_free_result($ris);
mysqli_close($conn);
header("Location: main.php");

```

▼ logout.php

```

<?php
session_start();
$_SESSION = array();
session_destroy();
header("Location: main.php");

```

▼ navbar.php

```

<h2>Funzioni</h2>
<ul>
<li><a href="main.php?action=0">Visualizza</a></li>
<?php
    if (isset($_SESSION["user"]) && $_SESSION["auth"] ==
'admin') {
        echo '<li><a href="main.php?action=1&mode=0">Ins
erisci</a></li>';
        echo '<li><a href="main.php?action=2&mode=0">Mod
ifica</a></li>';
    }
?>
</ul>
<br>

```

```

<br>
<?php
    if (!isset($_SESSION["user"])) {
        echo '<h3 >Login</h3>';
        echo '<form action="login.php" method="post">';
        echo '<label>User: <input name="user" type="text" size="10"></label><br>';
        echo '<label>Pass: <input name="pass" type="password" size="10"></label><br><br>';
        echo '<input type="submit" value="Accedi"></p>';
        echo '</form>';
    }
    else {
        echo '<form action="logout.php" method="post">';
        echo '<input type="submit" value="Logout">';
        echo '</form>';
    }
?>

```

▼ style.css

```

body {
    margin: 0;
    background-color: #d6daed;
    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
    color: #003399;
    font-size: 10px;
}

#header {
    background-color: #FF7373;
    padding: 0.5em;
}

#main {
    margin-left: 15em;
}

```

```

}

#navbar {
    background-color: #5CCCCC;
    width: 14em;
    padding: 1em 0.5em;
    float: left;
    margin-left: -15em;
}

#content {
    background-color: #FFFFFF;
    border-left: 1px dotted #5CCCCC;
    padding: 1em 0.5em;
}

#footer {
    background-color: #C9F76F;
    padding: 0.5em;
    clear: left;
}

h1, h2, h3 {
    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
}

a {
    color: #003399;
}

a:link {
    text-decoration: none
}

table {
    border-collapse: collapse;
    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
}

```

```

ans-Serif;
    font-size: 12px;
    margin: 20px;
    text-align: right;
}

th {
    border-bottom: 2px solid #6678B1;
    color: #003399;
    font-size: 14px;
    font-weight: normal;
    padding: 10px 8px;
}

td {
    border-bottom: 1px solid #CCCCCC;
    padding: 6px 8px;
}

```

Per accedere attiviamo la funzione `session_start()` dove la prima volta che viene attivata crea il file e lo mette a disposizione, tutte le altre volte lo recupera e mette a disposizione in ambiente PHP i dati al suo interno. Questo attraverso una variabile super globale (ossia variabili speciali del linguaggio che consentono di gestire i dati inviati tramite un form e di recuperare informazioni sul browser (client) o il sito (server)).

1. Ogni volta che il client si connette al server si deve riconoscere attraverso un meccanismo di identificazione del client sul server, nell'ambiente PHP viene utilizzato il **cookie**.

La prima volta che un client si connette ad un server (un server che ha il meccanismo delle sessioni), il server oltre a mandargli il messaggio richiesto gli manda nell'intestazione della richiesta un messaggio in cui è codificata una stringa e viene richiesto al client di memorizzare quella stringa in un **cookie**. Questa stringa, insieme all'url del sito che gliel'ha invitata, corrisponde al riconoscimento del client sul server.

2. La navbar crea la lista con le azioni che l'utente può fare, ma lo fa tenendo conto se l'utente è autenticato o no. Se non lo è genera il *login*. Il primo `` viene generato in modo incondizionato, una seconda sezione che genera le voci solo dopo il login. L'ultima sezione in PHP genera la form di login e logout.

Nella prima parte del PHP abbiamo un *isset* : abbiamo due componenti `$session` dove la prima ha `user` mentre la seconda ha `auth` che è valorizzato con la stringa `'admin'`. Con l'*isset* verifichiamo se le due variabili sono valorizzate e nel caso lo siano possiamo visualizzare il resto della pagina. Nel momento in cui il client si collega per la prima volta al server e richiede la pagina iniziale, non essendo ancora stato implementato il meccanismo delle sessioni, la variabile `$session` sarà vuota e quindi entrambe le condizioni sono false e avremo che al primo collegamento le due ancora non verranno visualizzate.

La prima condizione ci dice che nella sessione c'è un utente associato a questo client, la seconda condizione verifica che la classe di utenza dell'utente che si è collegato sia la classe identificata con la stringa `'admin'`.

In fondo alla pagina abbiamo sempre un *isset* : nel caso in cui non ci sia nella variabile `$session` una componente `user` , il nostro programma genera la form di login. Se invece l'*isset* è falso viene mostrato il bottone di logout.

3. `login.php` : abbiamo una submit in una form, quindi il server ha inviato al client la pagina con la navbar con la form, l'utente ha inserito username e password , i dati inseriti e l'url parte come richiesta dal client al server e quindi sul server riparte tutto l'ambiente PHP che si era spento al termine della fase precedente. A ogni interazione l'ambiente PHP si ricrea e poi si spegne perdendo tutto. La prima cosa che deve fare è attivare `session_start()` in modo tale da recuperare i contenuti del file. Con il `require` il nostro file PHP fa la `connect.php` dove una volta presi utente e password vede se c'è un utente con quel username e password e se c'è farà determinate cose altrimenti ne farà altre.

Poi abbiamo tutto il collegamento al database. Nel database attiviamo la query che estrae tutti i dati dalla tabella utenti dove i valori della colonna `user` è uguale al valore dello username che l'utente ha indicato nella form che recupero attraverso il `$_POST["user"]` . Con questa istruzione cerchiamo nella tabella *utenti* se c'è un utente che ha come valore dell'attributo `user` lo username digitato nella form. Per vedere se esiste o no usiamo l'`if` con cui contiamo le righe estratte e sarà una sola riga dato che lo username è

unico. Se il numero delle righe contate è zero (abbiamo un `not` davanti perché il PHP considera il valore zero come `false` ma con il `not` davanti diventa `true`) chiudiamo la connessione e diamo il messaggio di errore. Per il messaggio di errore funziona che quando l'applicazione lato server deve inviare un messaggio all'utente, invia come risposta al client una direttiva che determina un'ulteriore richiesta che riparte con un parametro (URL della risorsa) che viene specificato nella prima risposta. In sintesi il client fa una richiesta al server, il server va in una condizione di errore, rimanda indietro al client una direttiva che dice "a desso ricarica la pagina" e la pagina ricaricata contiene il messaggio di errore. Questo passaggio in cui il client richiede al server il caricamento di una pagina di errore è quello che noi abbiamo codificato come azione 99 nel `main.php` . Quando il client fa una richiesta di tipo 99 il server produce in output una stringa nel quale include il contenuto della `$_GET["msg"]` che è un parametro che la richiesta contiene. Il server setta questo parametro quando arrivato a una condizione di errore manda indietro al client un messaggio dice "ricarica la pagina". Questo messaggio sarà `action=99` e `msg=la_stringa_che_vogliamo_stampare` .

Come fa un programma lato server a dire al client un comando che fa sì che il client rinvii una richiesta al server secondo una parametrizzazione definita ?

Attraverso la funzione header che mi consente di specificare come parametro uno qualunque dei parametri che io posso inserire nell'intestazione di un messaggio HTTP attraverso l'indicazione della coppia nome-valore del parametro con le due componenti separate da `:` .

Quando il client riceve da un server un messaggio che ha l'attributo *location* definito fa partire la richiesta rispecificata col parametro *location*.

Se invece lo username è giusto recuperiamo il dato estratto nella base di dati e lo mettiamo in un'array associativo per comodità. Adesso verifichiamo se la password inserita è uguale a quella che abbiamo nel database. Se è sbagliata accadrà lo stesso meccanismo di prima ma con un messaggio di errore differente. Se invece l'`if` sulla password non è verificato significa che l'utente ha inserito utente e password corretti e memorizziamo il profilo del nostro utente in `$_session` così da poterlo recuperare nelle successive richieste. Quindi definiamo un nuovo elemento nella `$_session` e memorizziamo una componente che si chiama user che

contiene lo username e un'altra che chiamiamo auth che contiene la sua categoria. Infine liberiamo il database e chiudiamo la connessione.

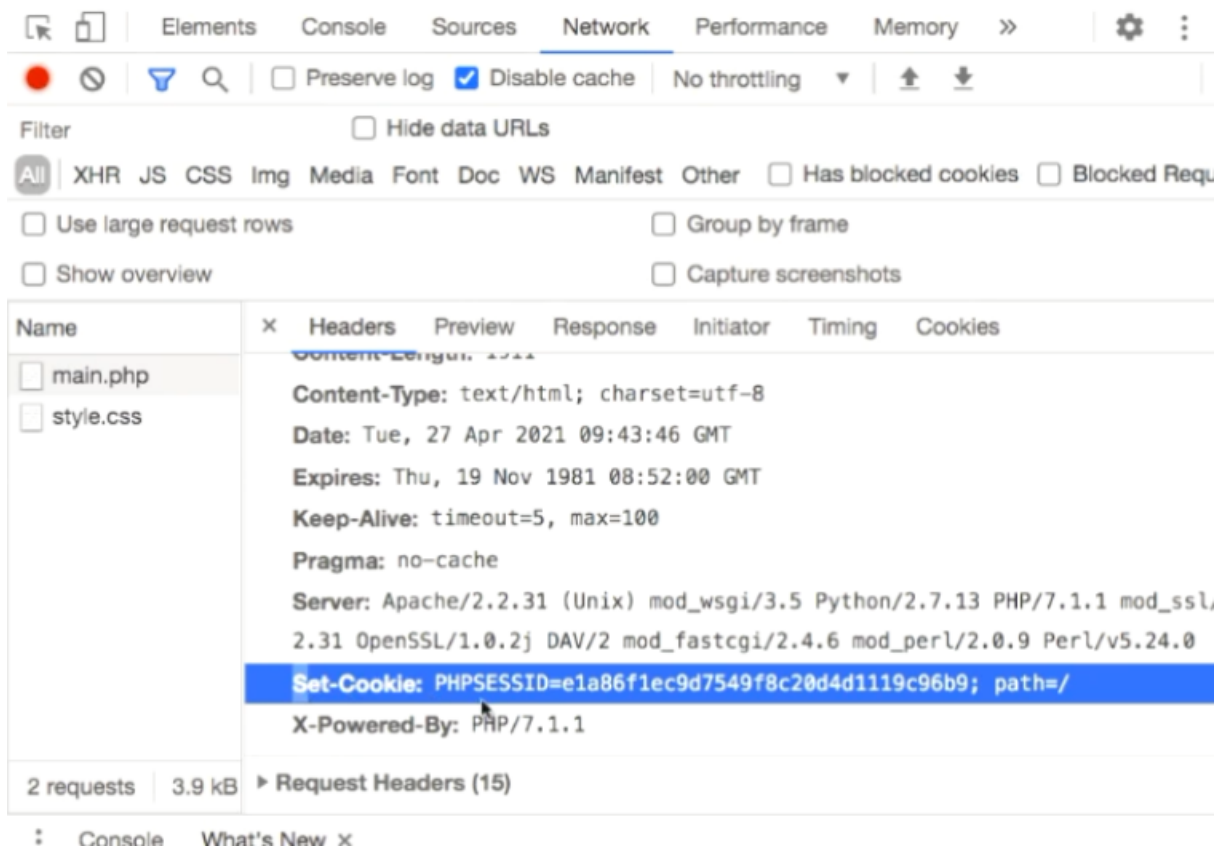
Anche in questo caso riattiviamo il nostro programma dal `main.php`, cioè diciamo al client di ricaricare `main.php` senza parametri ma adesso che verrà ricaricata la pagina la prima condizione isset `$session_user` non sarà più falsa e la componente auth sarà associata ad admin.

4. Quando faccio il logout parte la richiesta per `logout.php` che va a distruggere la sessione. Non cancella il file ma azzerà tutto quello che abbiamo registrato nella sessione, ovvero la componente user e auth. Parte sempre con la `session_start()`, azzerà nella memoria PHP l'array `$_session` e poi esegue la `session_destroy()` che cancella fisicamente il contenuto del file. Dopodiché con il concetto di header chiediamo al client di ricaricare il `main.php` che sarà quello iniziale.
5. L'ultimo passaggio è quello dell'autorizzazione. Su `function.php` sulla `insert` e sull'`update` metto in testa un `if` dove vedo se la richiesta è di un utente autorizzato e ha un'autorizzazione di tipo admin e se queste due cose non vengono rispettate facciamo comparire il messaggio di errore.

Attiviamo una richiesta da parte del Client della homepage della nostra applicazione delle sessioni, nel frattempo tracciamo *Network* sugli strumenti di sviluppatore. Notiamo che sono due i messaggi che si sono scambiati il Client e il Server.

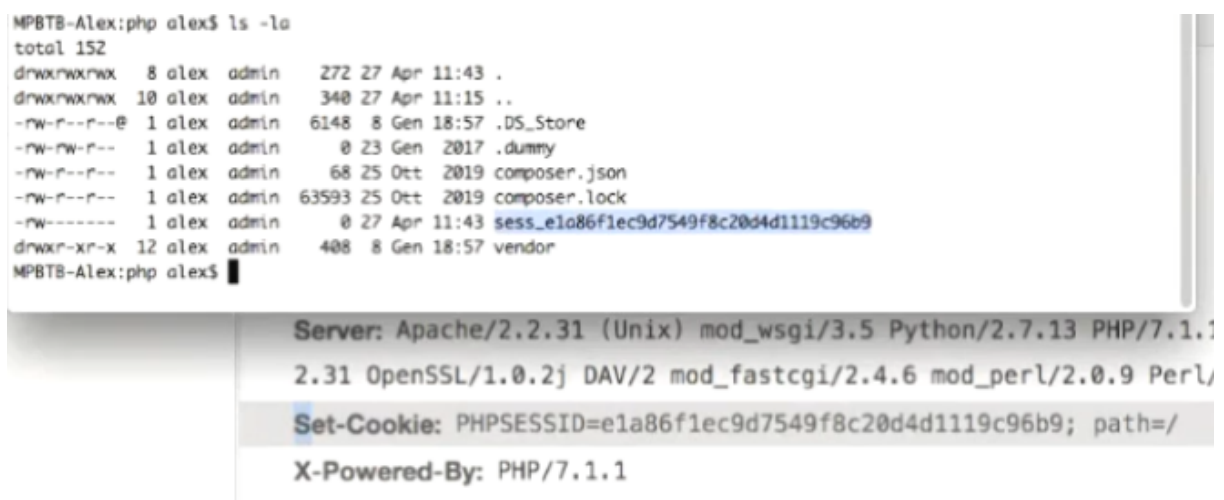
Il primo messaggio, `main.php`, è una richiesta di tipo *GET* che il Client ha fatto al Server, una richiesta associata al nostro URL.

Analizziamo nel dettaglio i *Response Headers*, cioè gli Headers della risposta, tra i vari parametri troviamo la direttiva `Set-Cookie` con il parametro con nome `PHPSESSID` ed un valore corrispondente.



Questa è la direttiva che indica al Client di creare nel suo ambiente locale un *cookie*, associato al Server, che contiene la stringa, la quale identifica la sessione che è stata creata nel Server.

All'interno della cartella che contiene i file di sessioni (lato Server) è stato creato un file avente codice uguale a quello del valore `Set-Cookie`.



Questo file verrà utilizzato nel momento in cui il nostro utente si autentica, andando a scrivere nel file l'username e la classe dell'utente. Lato Client notiamo la creazione di un nuovo *Cookie* nell'elenco dei *Cookie*.

Il Server ha creato un identificatore del Client e ha detto al Client di marcare attraverso i

Cookie questo identificatore di sessione.

Da adesso in poi, ogni volta che il Client interagisce con il Server invia anche il *Cookie*.

Facendo partire la funzione *Visualizza* tra i Request Headers troviamo il *Cookie*, questo permetterà alla macchina di recuperare delle informazioni della sessione.

```
▼ Request Headers    View source
Accept: text/html,application/xhtml+xml,application/xml;q=
0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application
signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: no-cache
Connection: keep-alive
Cookie: PHPSESSID=e1a86f1ec9d7549f8c20d4d1119c96b9
```

Facendo il Login notiamo che il file di sessione nel Server è stato scritto. Nel momento del Logout invece il file di sessione non viene cancellato, ma viene azzerato, neanche il *Cookie* del Client viene cancellato, perchè le due macchine continuano a dialogare attraverso la stessa sessione. Rimane la struttura dati pronta per un altro Login. Generalmente dopo 30 minuti il file di sessione viene svuotato anche senza Logout, in questo caso l'utente viene disconnesso,

Il file di sessione viene mantenuto anche se il documento è vuoto, per una politica che fa l'ambiente che gestisce le *Session* di PHP, perchè così la struttura rimane pronta nel momento del bisogno, in modo da non dover ricreare da zero tutto il meccanismo di creazione del file se effettuo più Login e Logout in poco tempo. Il file delle sessione può essere utilizzato per memorizzare informazioni associate ad un certo tipo di Client. Data la sequenzialità di alcune azioni potrei evitare di specificare alcuni parametri

ridondanti andando a memorizzare nel file di sessione il valore dell'action.
Snellisco andando a memorizzare nel file di sessione alcuni parametri di elaborazione, al di là del Login e Logout, per questo momento viene mantenuto.
Il file di sessione è un contenitore per le informazioni condivise tra un certo Client e un certo Server.