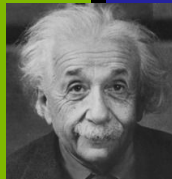


# B - PHP

## ESo2 - Accesso a pagina riservata

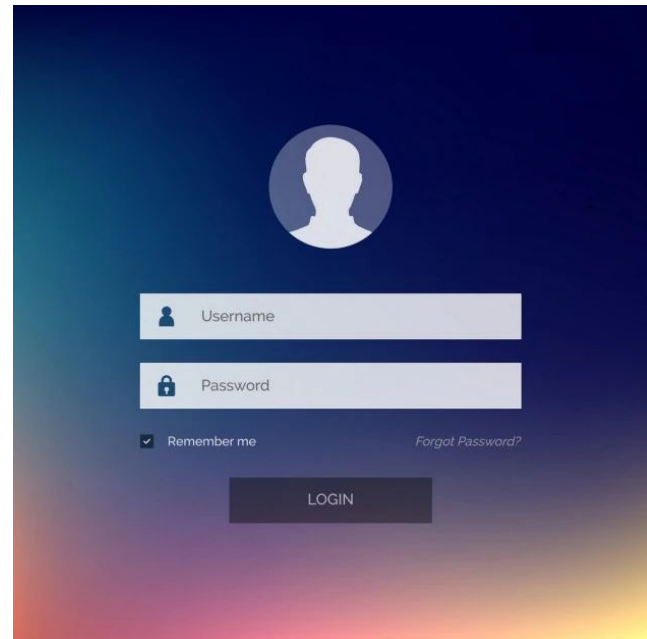
**FB - 10.2024 - v1.3**



“La teoria è quando si sa tutto e niente funziona. La pratica è quando tutto funziona e nessuno sa il perché. Noi abbiamo messo insieme la teoria e la pratica: non c'è niente che funzioni... e nessuno sa il perché!”

# Accesso a pagina riservata

Creare un'applicazione web che permetta tramite un form html l'autenticazione e l'accesso ad una pagina riservata del sito



# Indice

- Obiettivi didattici
- Introduzione teorica
  - Form
  - Invio dati Form al server
  - Single page application (postback)
  - Operatore ternario
- Esercitazione
- Valutazione

# Obiettivi didattici

---



# Obiettivi didattici

- i form html
- i metodi GET e POST del protocollo HTTP
- le variabili globali php “superglobals” `$_GET`, `$_POST` e `$_REQUEST`

# Introduzione teorica

---



# Form



I form HTML sono uno strumento fondamentale per interagire con gli utenti attraverso le pagine web. Essi consentono agli sviluppatori di creare moduli che gli utenti possono compilare e inviare con le informazioni desiderate.

Un form HTML è costituito da una serie di elementi, tra cui campi di input, bottoni e menu a discesa, che consentono agli utenti di inserire dati. Quando un utente compila un form e lo invia, i dati vengono inviati al server web per l'elaborazione o l'archiviazione.

Tag `<form>` racchiude tutto il contenuto del form e ne definisce l'inizio e la fine. L'attributo `"action"` specifica l'URL del server o lo script che elaborerà i dati del form. L'attributo `"method"` specifica il metodo di invio dei dati del form al server, come `"GET"` o `"POST"`.

```
<form action="dat.php" method="get">
  <label for="nome">Nome: </label>
  <input type="text" name="nome"><br>
  <input type="submit" value="Submit">
</form>
```



# Messaggi HTTP



Ecco un esempio di messaggio HTTP con dati form inviati tramite il metodo POST:

## Richiesta HTTP:

```
makefile Copia codice  
  
POST /login HTTP/1.1  
Host: www.esempio.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 32  
  
username=utente&password=segreta
```

## Spiegazione:

- **Metodo POST:** Il metodo usato per inviare i dati.
- **Host:** Il dominio del server a cui si invia la richiesta.
- **Content-Type:** Indica che i dati del form sono codificati come `application/x-www-form-urlencoded`.
- **Content-Length:** La lunghezza del corpo della richiesta.
- **Corpo della richiesta:** Contiene i dati del form inviati (ad esempio, `username` e `password`).

# Messaggi HTTP



Ecco un esempio di richiesta HTTP con il metodo **GET**:

## Richiesta HTTP:

vbnet

Copia codice

```
GET /login?username=utente&password=segreta HTTP/1.1  
Host: www.esempio.com
```

## Spiegazione:

- **Metodo GET**: I dati vengono passati nell'URL, dopo il punto di domanda `?`.
- **Host**: Il server a cui si invia la richiesta.
- **Parametri URL**: Le coppie `username=utente` e `password=segreta` sono i dati inviati nel form.
- **GET** è usato per ottenere dati e non per inviare in modo sicuro informazioni sensibili, poiché i dati sono visibili nell'URL.



# Elementi presenti all'interno dei Form

- `<input>` è l'elemento più utilizzato per creare campi di input, come campi di testo, checkbox, radio button, pulsanti di invio, ecc.
- `<label>` per associare un'etichetta ad un campo di input.
- `<textarea>` rappresenta un controllo di modifica di testo in formato testo normale a più righe, utile quando si vuole consentire agli utenti di inserire una quantità considerevole di testo libero.
- `<option>` viene utilizzato per definire un elemento contenuto in un `<select>`, un `<optgroup>` o un `<datalist>`.
  - `<datalist>` per fornire un elenco di opzioni predefinite per un campo di input.
  - `<select>`: per creare menu a tendina.
    - `<optgroup>` crea un raggruppamento di opzioni all'interno di un elemento `<select>`
- `<button>` è un elemento interattivo attivato dall'utente con un mouse, una tastiera, un dito, un comando vocale o altre tecnologie assistive. Una volta attivato, esegue un'azione, come l'invio di un form o l'apertura di una finestra di dialogo.
- `<fieldset>` per raggruppare elementi del form.
  - `<legend>` per definire il titolo di un gruppo di elementi del form (`<fieldset>`).
- `<output>` per visualizzare il risultato di un calcolo o di un'operazione in un campo di output.
- `<progress>` e `<meter>` per visualizzare una barra di avanzamento.



# Elementi Input presenti all'interno dei Form

All'interno del tag `<form>` troviamo diversi tipi di elementi `<input>` che possono essere utilizzati all'interno di un form, come ad esempio: [[ref1](#), [ref2](#)]

- `<input type="text">` per inserire testo
- `<input type="password">` per inserire password
- `<input type="radio">` per selezionare una sola opzione tra diverse
- `<input type="checkbox">` per selezionare una o più opzioni tra diverse
- `<input type="submit">` per inviare i dati del form
- ...



# Nuovi tipi di input introdotti con HTML5

HTML5 ha introdotto diversi nuovi tipi di input che hanno migliorato significativamente la funzionalità e l'usabilità dei form. Ecco un elenco dei principali nuovi tipi di input introdotti con HTML5 ([esempio completo](#)):

- **Color:** consente di selezionare un colore da una tavolozza di colori predefiniti o direttamente dal suo codice in #esadecimale.
- **Date:** consente di selezionare una data tramite un calendario.
- **Datetime:** consente di selezionare una data e un'ora tramite un calendario.
- **Datetime-local:** consente di selezionare una data e un'ora tramite un calendario, senza il fuso orario.
- **Email:** consente di inserire un indirizzo email.
- **Month:** consente di selezionare un mese e un anno tramite un calendario.
- **Number:** consente di inserire un numero.
- **Range:** consente di selezionare un valore numerico all'interno di un intervallo.
- **Search:** consente di inserire un termine di ricerca.
- **Tel:** consente di inserire un numero di telefono.
- **Time:** consente di selezionare un'ora tramite un orologio.
- **Url:** consente di inserire un indirizzo web.
- **Week:** consente di selezionare una settimana dell'anno tramite un calendario.



# Attributi comuni

Gli elementi `<input>` in HTML5 hanno diverse proprietà che possono essere utilizzate per personalizzare il comportamento e l'aspetto dei campi di input all'interno di un form. Alcune delle proprietà più comuni sono:

- **type**: specifica il tipo di input, come ad esempio "text", "password", "radio", "checkbox", "submit", "email", "tel", "date", "color", "range", "file", "number", "search", "time", "url", "datetime-local", "month", "week", "hidden"
- **name**: specifica il nome del campo di input, che viene utilizzato per identificare il campo quando i dati del form vengono inviati al server
- **value**: specifica il valore predefinito del campo di input
- **placeholder**: fornisce un suggerimento sul tipo di dati da inserire in un campo di input
- **required**: indica che un campo di input è obbligatorio
- **autofocus**: imposta il focus su un campo di input specifico quando la pagina viene caricata
- **readonly**: indica che un campo di input è di sola lettura
- **disabled**: indica che un campo di input è disabilitato
- **maxlength**: specifica il numero massimo di caratteri che possono essere inseriti in un campo di input
- **min** e **max**: specificano il valore minimo e massimo che possono essere inseriti in un campo di input di tipo "number" o "range"
- **step**: specifica l'incremento o il decremento del valore di un campo di input di tipo "number" o "range"
- **pattern**: specifica un'espressione regolare che deve essere soddisfatta dal valore inserito in un campo di input
- **list**: specifica l'id di un elemento `<datalist>` che contiene elementi predefiniti per un campo di input di tipo "text" o "search"
- **autocomplete**: specifica se il browser deve suggerire o meno i valori inseriti in un campo di input.



L'etichetta (label) è un elemento HTML che serve ad associare un'etichetta ad un campo di input di un form. L'utilizzo dell'etichetta è importante per migliorare l'accessibilità del form e per facilitare la comprensione dell'utente. L'attributo "for" dell'etichetta deve corrispondere all'attributo "id" del campo di input.

Esempio di utilizzo:

```
<label for="nome">Nome:</label>  
<input type="text" id="nome" name="nome">
```

L'etichetta può essere personalizzata utilizzando CSS e JavaScript. Ad esempio, è possibile nascondere l'etichetta e utilizzare un'icona al suo posto.

In generale, l'utilizzo dell'etichetta è importante per migliorare l'accessibilità del form e facilitare la comprensione dell'utente. L'etichetta deve essere utilizzata per ogni campo di input del form e deve essere associata correttamente al campo di input tramite gli attributi "for" e "id".



Datalist: l'elemento datalist consente di fornire un elenco di opzioni predefinite per un campo di input. Ad esempio, è possibile fornire un elenco di città per un campo di input "città".

Esempio di utilizzo:

```
<label for="città">Città:</label>
<input type="text" id="città" name="città" list="città-list">
<datalist id="città-list">
  <option value="Roma">
  <option value="Milano">
  <option value="Napoli">
  <option value="Firenze">
</datalist>
```



Città:

- Roma
- Milano
- Napoli
- Firenze





# Elementi progress e meter

Progress: l'elemento progress consente di visualizzare una barra di avanzamento per un'operazione in corso. Ad esempio, è possibile visualizzare una barra di avanzamento per il caricamento di un file. Esempio di utilizzo: [\[ref\]](#)

```
<label for="file">Carica file:</label>
<input type="file" id="file" name="file">
<progress id="progress" value="0" max="100"></progress>
```

Meter: l'elemento meter consente di visualizzare una barra di avanzamento per un valore numerico. Ad esempio, è possibile visualizzare una barra di avanzamento per il punteggio di un quiz. Esempio di utilizzo:

```
<label for="punteggio">Punteggio:</label>
<input type="range" id="punteggio" name="punteggio" min="0" max="100">
<meter id="meter" value="0" min="0" max="100"></meter>
```



# Invio dati Form al server



- L'interazione con l'utente avviene principalmente mediante l'uso di form HTML
  - HTML fornisce vari tag per visualizzare e formattare opportunamente le FORM
  - Quando l'utente "conferma" i dati nella form, le informazioni vengono codificate e inviate al server tramite HTTP
  - Il server elabora i dati e li gestisce (nel nostro caso tramite PHP)

Esempio di Form HTML

Nome:

☒ Rosso ☐ Verde

```
<FORM action="pagina.php"
      method="GET" | "POST">
<INPUT type="text" name="username" />
<INPUT type="radio" name="color"
      value="Rosso">
...
<INPUT type="submit" />
```



## ● GET

- i parametri sono passati accodandoli alla URL del gestore della form

```
http://localhost/manage.php?dato1=pippo&dato2=pluto
```

- la stringa dei parametri (**visibile** nel browser) viene detta **query string**
- può essere costruita artificialmente o inserita in un bookmark
- lunghezza massima querystring 256 caratteri

## ● POST

- i parametri vengono passati direttamente tramite protocollo HTTP
- non sono visibili



- ▶ I dati del form possono essere inviati con una richiesta HTTP di tipo **GET**
  - ▶ I parametri sono accodati all'URL della risorsa che li riceve usando in genere il carattere **?** come separatore
  - ▶ Questo metodo permette di associare la richiesta con i parametri ad un URL (può essere usato come hyperlink, generato in automatico,..)
  - ▶ Esiste un limite al numero dei caratteri e inoltre l'URL deve essere composto da caratteri ASCII
    - ▶ I caratteri non ASCII e i caratteri speciali (?,=,&,...) sono codificati con la sequenza % seguito dalle due cifre esadecimali (es %E8 per è)
    - ▶ Lo spazio è in genere codificato con il carattere + invece che con %20
  - ▶ Il metodo GET non dovrebbe essere usato per passare informazione sensibile (es. password) perché è in chiaro nell'URL e può essere memorizzato nei proxy

# Esempio di form e richiesta GET



```
<form action="/ecom/buy.php" method="get">  
  Prodotto <input type="text" name="prodotto" size="20"><br/>  
  Acquisto <input type="checkbox" name="compra"><br />  
  <input type="submit" value="Invia">  
</form>
```

Prodotto

Acquisto ☒

GET /ecom/buy.php?prodotto=computer+più+2000&compra=on HTTP/1.1  
.....

- ▶ L'attivazione del form con il bottone **submit** genera una richiesta HTTP
  - ▶ L'attributo **action** specifica l'URL della risorsa a cui inviare i dati del form (nell'esempio un URL relativo)
  - ▶ L'attributo **method** indica che la richiesta è di tipo GET e che di conseguenza il client deve accodare i parametri all'URL nella forma URL encoded

# Invio dei dati con HTTP - POST



- ▶ I dati del form possono essere inviati con una richiesta HTTP di tipo POST
  - ▶ I dati sono inseriti nel corpo della richiesta HTTP
  - ▶ Non si ha un riferimento URL alla richiesta con i parametri valorizzati
  - ▶ E' più robusto e sicuro del metodo GET e non ha limitazione sulla dimensione dei dati inviati

```
<form action="/ecom/buy.php" method="post">  
  Prodotto <input type="text" name="prodotto" size="20"><br/>  
  Acquisto <input type="checkbox" name="compra"><br />  
  <input type="submit" value="Invia">  
</form>
```

Prodotto

Acquisto ☒

submit action

```
POST /ecom/buy.php HTTP/1.1  
host: www.mycommerce.it  
Content-type: application/x-www-form-urlencoded  
Content-Length: 38  
  
prodotto=computer+più2000&compra=on
```



- Uno script PHP può accedere ai parametri in tre modi
  - `$_POST["nomepar"]` per il metodo POST
  - `$_GET["nomepar"]` per il metodo GET
- Oppure accedendo all'array globale delle richieste
  - `$_REQUEST["nomepar"]`
  - vale per entrambi i metodi
  - lo script PHP è indipendente dal metodo usato dalla FORM
- Tutti i parametri di passaggio nelle form sono nell'ambiente predefinito di php e sono quindi visualizzabili con la funzione `phpinfo()`





# Variabili predefinite `$_GET`, `$_POST` e `$_REQUEST`

Le variabili globali "superglobals" in PHP sono variabili predefinite che sono sempre disponibili in tutti gli ambiti di uno script PHP. In particolare, `$_GET`, `$_POST` e `$_REQUEST` sono tre variabili superglobali utilizzate per accedere ai dati inviati da un form HTML o da un URL.

`$_GET`: contiene i dati inviati tramite il metodo HTTP GET. Questi dati sono visibili nell'URL della pagina e possono essere memorizzati nei segnalibri del browser. Ad esempio, se si ha un form HTML con un campo di testo chiamato "nome", il valore inserito in questo campo può essere recuperato utilizzando `$_GET['nome']`.

`$_POST`: contiene i dati inviati tramite il metodo HTTP POST. Questi dati non sono visibili nell'URL della pagina e non possono essere memorizzati nei segnalibri del browser. Ad esempio, se si ha un form HTML con un campo di testo chiamato "nome", il valore inserito in questo campo può essere recuperato utilizzando `$_POST["nome"]`.

`$_REQUEST`: contiene i dati inviati tramite il metodo HTTP GET, POST e COOKIE. Questa variabile può essere utilizzata per accedere ai dati inviati da un form HTML o da un URL, indipendentemente dal metodo utilizzato per inviare i dati. Tuttavia, è importante notare che `$_REQUEST` può contenere dati provenienti da cookie, il che potrebbe rappresentare un rischio per la sicurezza.



# Variabili predefinite \$\_SERVER

La variabile `$_SERVER['REQUEST_METHOD']` è una variabile superglobale di PHP che contiene il metodo HTTP utilizzato per la richiesta. In particolare, questa variabile contiene il valore "GET" o "POST" a seconda che la richiesta sia stata effettuata con il metodo GET o POST.

Il metodo GET viene utilizzato per richiedere una risorsa dal server, mentre il metodo POST viene utilizzato per inviare dati al server. Quando un form viene inviato tramite il metodo POST, i dati del form vengono inviati al server in modo nascosto, mentre con il metodo GET i dati del form vengono inviati come parte dell'URL.

Nel codice PHP, la variabile `$_SERVER['REQUEST_METHOD']` viene spesso utilizzata per verificare il metodo HTTP utilizzato per la richiesta. Ad esempio, se si vuole elaborare i dati di un form inviato tramite il metodo POST, si può utilizzare una condizione come questa:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    // elabora i dati del form  
}
```

In questo modo, il codice all'interno della condizione viene eseguito solo se la richiesta è stata effettuata con il metodo POST. Se la richiesta è stata effettuata con il metodo GET, il codice all'interno della condizione non viene eseguito.



- La gestione delle form prevede due passi
  - la visualizzazione della FORM in HTML
  - gestione dei parametri in PHP
- Soluzione tipica: Due pagine distinte
  - una in HTML (estensione .html)
  - l'altra come pagina PHP (estensione .php)



```
<FORM action="FormStudenti.php" method="GET">
  Cognome:   <INPUT type="text" name="cognome"><BR />
  Matricola: <INPUT type="text" name="matricola" maxlength="6">
<BR /><BR />
```

Iniziale del nome:

```
<SELECT name="nome">
  <OPTION>---</OPTION>
  <OPTION>A-H</OPTION>
  <OPTION>I-Z</OPTION>
</SELECT>
<BR /><BR />
```

Provincia:

```
<INPUT type="radio" name="prov" value="PI" /> PI
<INPUT type="radio" name="prov" value="SP" /> SP
<BR /><BR />
```

**FormStudenti.html**



```
<INPUT type="checkbox" name="tutor[]" value="HAS" />  
Ha un tutor  
  
<INPUT type="checkbox" name="tutor[]" value="IS" />  
E' tutor <BR /><BR />  
  
<TEXTAREA name="commento" rows="5">  
Mah ...  
</TEXTAREA> <BR /><BR />  
  
<INPUT type="submit" value="Invia">  
</FORM>
```

FormStudenti.html



- I parametri della form possono essere recuperati dall'array `$_GET`

```
<?PHP
$cognome = $_GET["cognome"];
$matricola = $_GET["matricola"];
$nome = $_GET["nome"];
$provincia = $_GET["prov"];
```

- Nel caso di scelte multiple (es. checkbox e select), il parametro è un array

```
if ($_GET["tutor"]) {
    $hatutor = in_array('HAS', $_GET["tutor"]);
    $etutor = in_array('IS', $_GET["tutor"]);
};
$commento = $_GET["commento"];
```



- In fase di debug è una buona idea stampare i parametri della form
- Qui lo facciamo con HERE document

```
echo<<<END
Ecco i parametri:<BR />
<UL>
  <LI>Cognome:  $cognome</LI>
  <LI>Iniziali del nome: $nome</LI>
  <LI>Matricola: $matricola</LI>
  <LI>Provincia: $prov</LI>
  <LI>Ha tutor?: $hatutor</LI>
  <LI>&Egrave; un tutor?: $etutor</LI>
  <LI>Commento: $commento
</LI>
</UL>
END;
```



# Single page application (postback)



# Postback



Nello sviluppo web, un postback è un POST HTTP alla stessa pagina in cui si trova il modulo. In altre parole, i contenuti del modulo vengono inviati allo stesso URL del modulo.

I postback li troviamo spesso nei form di modifica, in cui l'utente introduce informazioni in un modulo e preme "salva" o "invia", provocando un postback. Il server quindi aggiorna la stessa pagina utilizzando le informazioni che ha appena ricevuto. [\[ref\]](#)

La tecnica del postback consiste nell'utilizzare una condizione per controllare se un form è stato inviato al server e, in tal caso, eseguire determinate azioni. Ad esempio, se un form contiene un campo "nome" e un campo "password", il codice PHP potrebbe controllare se il form è stato inviato e, in tal caso, verificare se il nome e la password inseriti corrispondono a quelli presenti in un database.

# Postback



Ecco un esempio di codice PHP che utilizza la tecnica del postback per gestire il comportamento di un form di login:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    // Il form è stato inviato  
    $nome = $_POST['nome'];  
    $password = $_POST['password'];  
  
    // Verifica se il nome e la password corrispondono a quelli presenti nel database  
    if ($nome=="admin" && $pwd=='123') {  
        // Le credenziali sono corrette, autentica l'utente  
        echo "<h4>Benvenuto $nome <br/>Nell'area riservata del sito!</h4>";  
    } else {  
        // Le credenziali sono errate, mostra un messaggio di errore  
        echo "<h4>Attenzione! Nome utente o password sbagliate</h4>";  
    }  
}
```



# Postback e \$\_SERVER['PHP\_SELF']

Il `$_SERVER['PHP_SELF']` è una variabile superglobale in PHP che rappresenta il nome del file dello script PHP corrente. È spesso utilizzata nell'attributo `action` di un elemento `<form>` HTML quando si desidera inviare il modulo alla stessa pagina PHP da cui è stato caricato il modulo. Questo è comune quando si desidera implementare il concetto di "postback" o gestire l'invio di dati su una singola pagina PHP.

```
<body>
  <h1>Form di Esempio</h1>

  <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
      // Qui gestisci i dati inviati dal form
      $nome = $_POST["nome"];
      $cognome = $_POST["cognome"];
      echo "<p>Ciao, $nome $cognome! Dati inviati con successo.</p>";
    }
  ?>

  <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
    <label for="nome">Nome:</label>
    <input type="text" name="nome" required><br>

    <label for="cognome">Cognome:</label>
    <input type="text" name="cognome" required><br>

    <input type="submit" name="submit" value="Invia">
  </form>
</body>
```



- Soluzione diversa: **form e gestore in un unico script**
- Necessità di distinguere il caso in cui si deve mostrare la form e quello in cui si deve elaborarla
  - **Valore** associato all'input submit
  - Uso di **self**

```
if (isset($_REQUEST['submit']))  
    # trattamento dei parametri  
else {  
    # visualizza la form  
    echo<<<END  
    <FORM method="POST"  
        action="$_SERVER['PHP_SELF']">  
    ...  
    ...  
    <INPUT type="submit" name="submit">  
    END;  
};
```

**form.php**



# Operatore ternario



# Operatore ternario

In PHP, l'operatore ternario permette di scrivere un'espressione "if-else" in una singola riga. La sintassi è la seguente:

```
espressione_condizione ? espressione_vera : espressione_falsa;
```

L'operatore ternario si presenta come una condizione seguita da un "?" e da due espressioni separate da ":". La prima espressione (`espressione_vera`) viene valutata e restituita se la condizione è vera, mentre la seconda espressione (`espressione_falsa`) viene valutata e restituita se la condizione è falsa.



# Operatore ternario

Esempio:

```
$x = 5;  
$y = ($x > 3) ? "x è maggiore di 3" : "x non è maggiore di 3";  
echo $y; //stamperà "x è maggiore di 3"
```

In questo esempio, la condizione è  $x > 3$ . Se questa è vera, la prima espressione "x è maggiore di 3" viene restituita e assegnata alla variabile \$y. Altrimenti, la seconda espressione "x non è maggiore di 3" viene restituita e assegnata alla variabile \$y.

```
username = isset($_POST["user"]) ? $_POST["user"] : "";  
  
if (isset($_POST["user"])) {  
    $username = $_POST["user"];  
} else {  
    $username = "";  
}
```

# Esercitazione

---





# Esercizio a

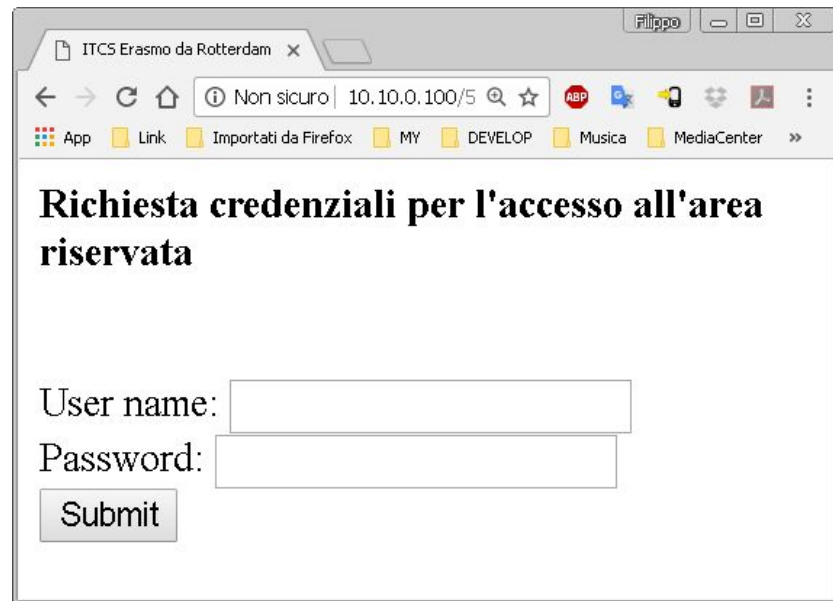
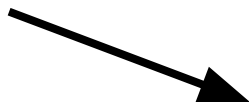
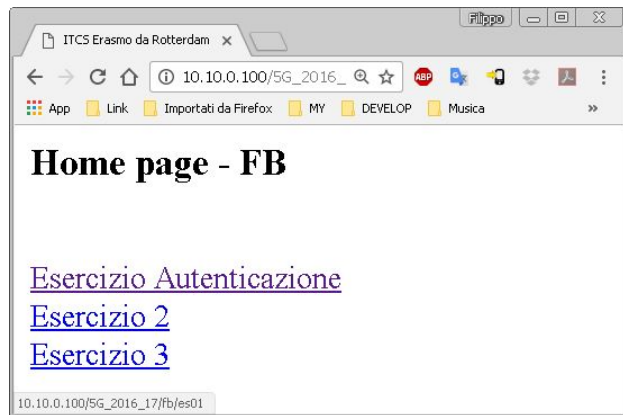


# Esercizio: Accesso a pagina riservata

- Creare la cartella per l'esercizio a "ESo2A" che conterrà i file index.html e login.php
- in htdocs creare index.php (la propria home page) che contenga un link all'applicazione web: "ESo2 A - Accesso a pagina riservata"
- L'applicazione web sarà contenuta all'interno della cartella ESo2A del server web (l'immagine riporta eso1 invece di ESo2A)

```
fb@rpi2:~$ cd /var/www/html/5G_2016_17/fb/  
fb@rpi2:/var/www/html/5G_2016_17/fb$ ls *  
index.html  
  
es01:  
index.php login.php  
fb@rpi2:/var/www/html/5G_2016_17/fb$
```

# La Home page contenente il link agli esercizi svolti





# La pagina index.php per visualizzare il form

```
<body>
```

[HTML input tag](#)

```
<h3>Accesso a pagina riservata</h3>
```

```
<form action="login.php" method="post">
```

```
<label for="username" ><b>Username</b></label>
```

```
<input type="text" name="nomeutente" placeholder="Inserisci il nome utente" /><br />
```

```
<label for="password"><b>Password</b></label>
```

```
<input type="password" name="password" placeholder="Inserisci la password" /><br />
```

```
<input name="submit" type="submit" value="Invia" />
```

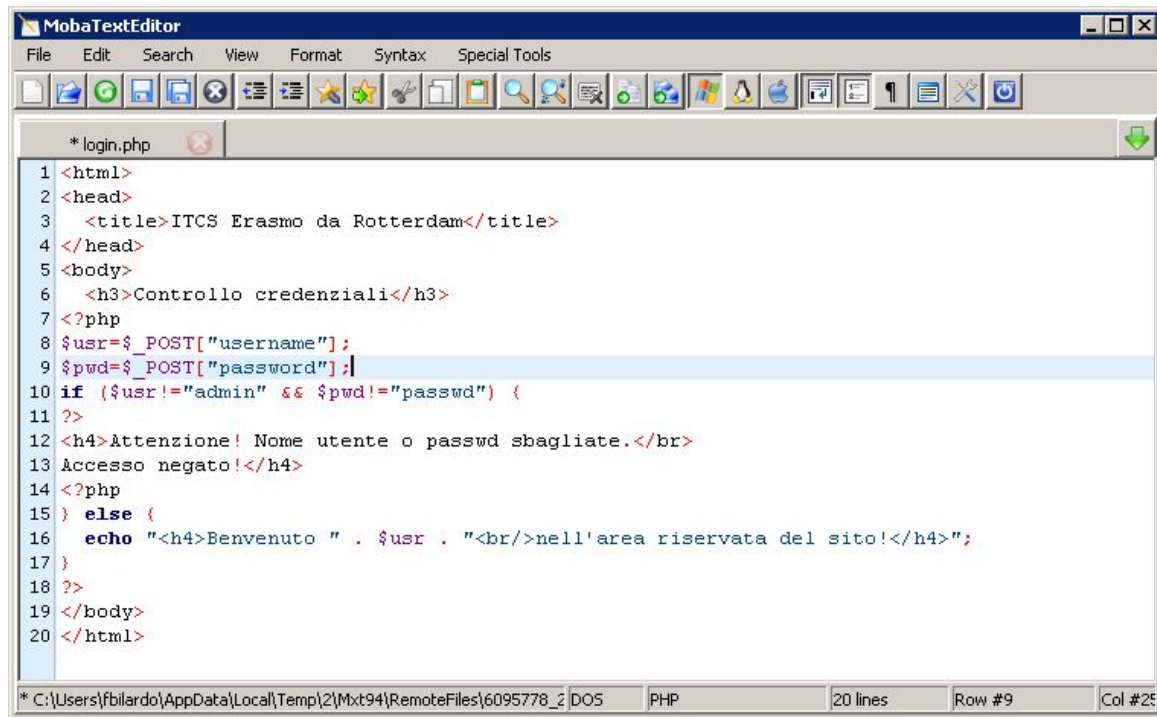
```
</form>
```

```
</body>
```

# La pagina login.php



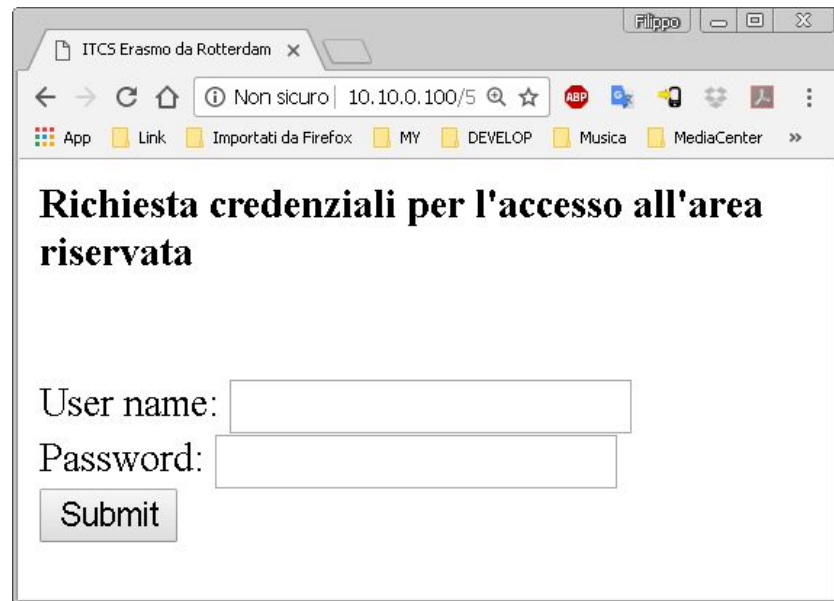
In questa pagina vengono verificate le credenziali precedentemente inserite, permettendo quindi l'accesso alla zona riservata della pagina oppure visualizzando un avviso di accesso negato.



```
* login.php
1 <html>
2 <head>
3   <title>ITCS Erasmo da Rotterdam</title>
4 </head>
5 <body>
6   <h3>Controllo credenziali</h3>
7 <?php
8 $usr=$_POST["username"];
9 $pwd=$_POST["password"];
10 if ($usr!="admin" && $pwd!="passwd") {
11 >
12 <h4>Attenzione! Nome utente o passwd sbagliate.</br>
13 Accesso negato!</h4>
14 <?php
15 } else {
16   echo "<h4>Benvenuto " . $usr . "<br/>nell'area riservata del sito!</h4>";
17 }
18 >
19 </body>
20 </html>
```

\* C:\Users\fbilardo\AppData\Local\Temp\2\Mxt94\RemoteFiles\6095778\_2 DOS PHP 20 lines Row #9 Col #25

# La pagina login.php/2



ITCS Erasmo da Rotterdam

Non sicuro | 10.10.0.100/5

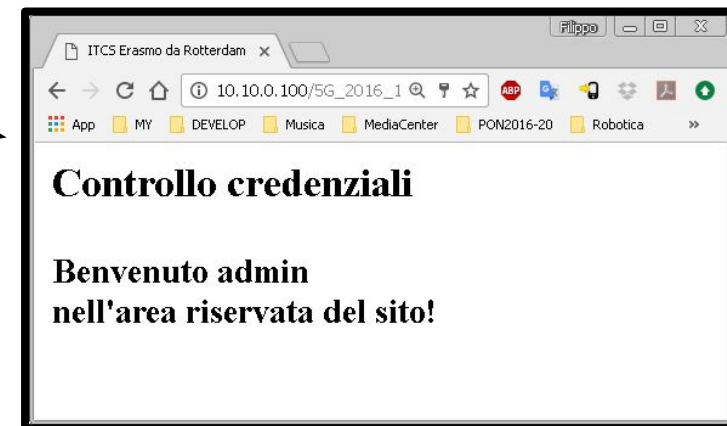
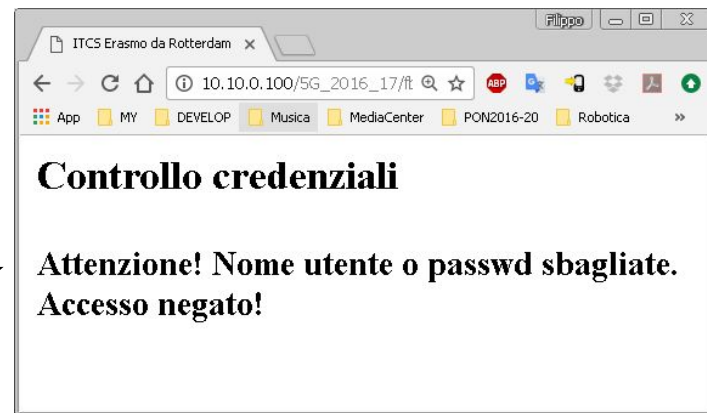
App Link Importati da Firefox MY DEVELOP Musica MediaCenter

## Richiesta credenziali per l'accesso all'area riservata

User name:

Password:

Submit





# Organizzazione dei file

- su C:\htdocs\5H\_Rossi\ESo1\_TAB\_PIT\ - - > cartella css, img
- C:\htdocs\5H\_Rossi\ESo2A\_AccessoPaginaRiservata\
- C:\htdocs\5H\_Rossi\ESo2B\_AccessoPaginaRiservata\
- Consegnare un documento Google contenente il link al codice scritto e gli screenshot del programma in esecuzione



# Esercizio b





## Esercizio b - login con controllo delle credenziali

Applicazione su singola pagina utilizzando la funzione [isset](#) oppure tramite il controllo `if ( $_SERVER['REQUEST_METHOD'] == 'POST' )`

In unica pagina .php scrivere sia codice per visualizzare il form che il codice per visualizzare la risposta, controllando ad esempio che variabile dell'array `$_POST['submit']` sia definita oppure controllando che `$_SERVER['REQUEST_METHOD']` sia uguale a “POST”

Quando la pagina viene richiamata la prima volta o tramite url verrà visualizzato il form di login, premendo il pulsante submit verranno controllate le credenziali inserite, in caso di successo verrà visualizzato un messaggio di benvenuto mentre in caso di fallimento oltre al messaggio di errore verrà riproposto nuovamente il form di login.



# Esercizio c



# Esercizio c - tabella dei quadrati e dei cubi

Obiettivo: sperimentare il postback in PHP

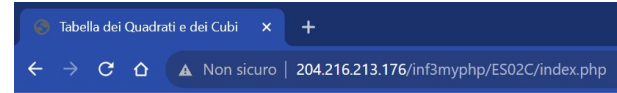
Realizzare una pagina PHP che permetta all'utente di inserire un numero intero N compreso tra 1 e 10. Il numero N va inserito con un menù a tendina. Quando l'utente preme il pulsante "Crea tabella", la pagina deve generare una tabella che contenga i quadrati e i cubi di tutti i numeri da 1 a N e Il form non dovrà essere visualizzato nuovamente.



## Tabella dei Quadrati e dei Cubi

Seleziona un numero intero N (da 1 a 10):

Crea tabella



## Tabella dei Quadrati e dei Cubi

Tabella dei quadrati e dei cubi per i numeri da 1 a 7:

Numero	Quadrato	Cubo
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343

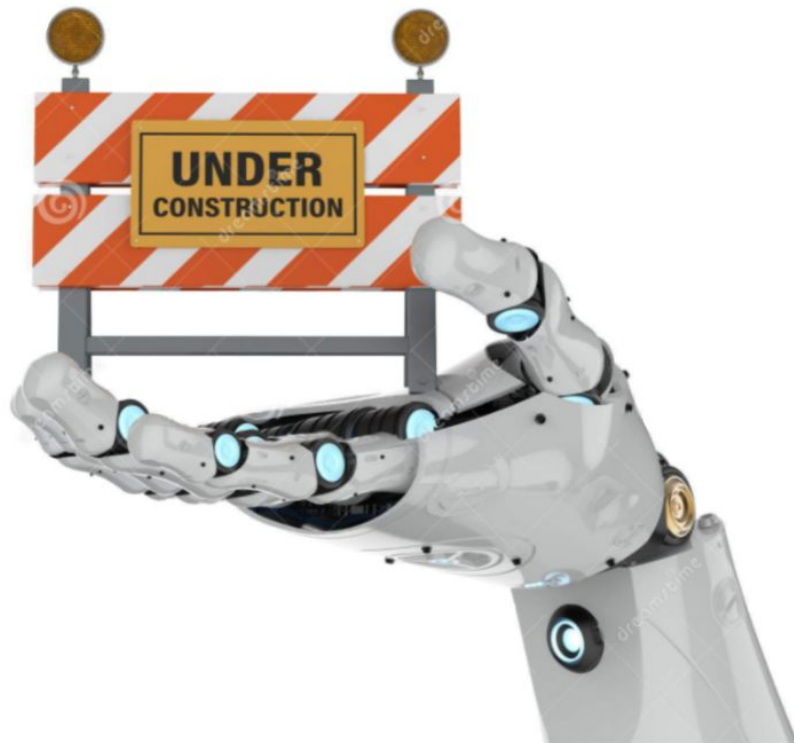
# Valutazione

---

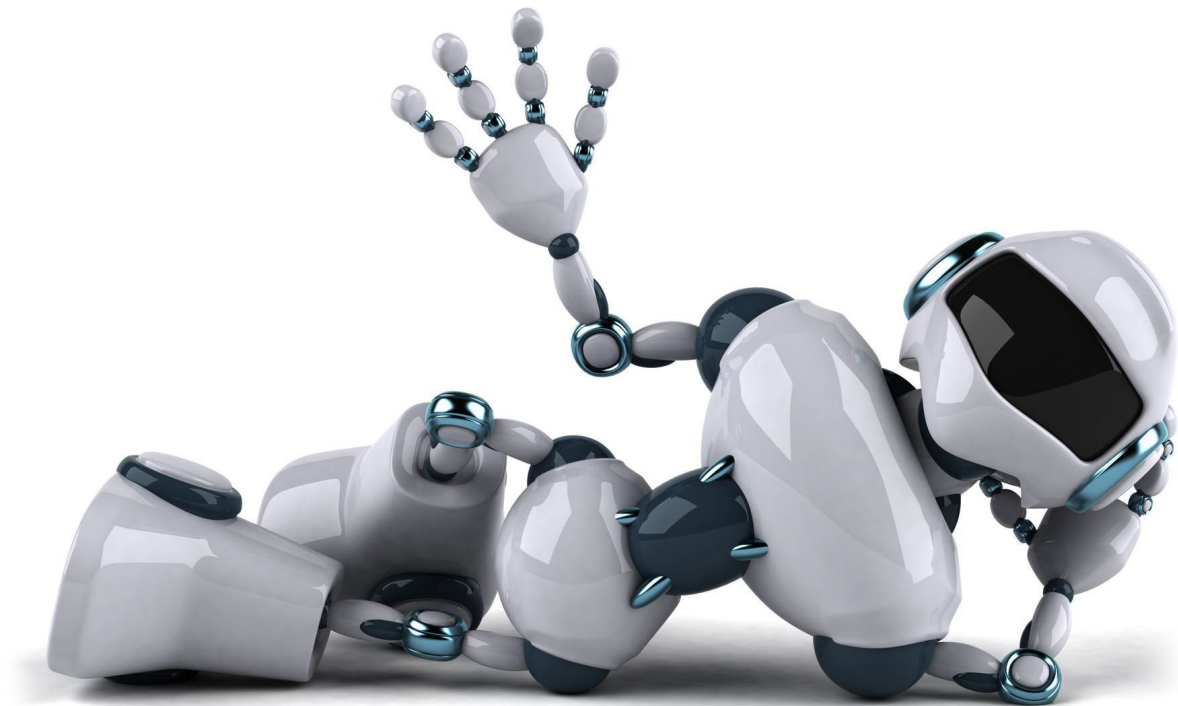
# Consegna del lavoro



Si ricorda l'importanza di svolgere e consegnare i progetti assegnati.



*Grazie per l'attenzione*





- <https://fb-labs.blogspot.com/>
- <https://github.com/filippo-bilardo>
- <https://www.youtube.com/channel/UCoBNbHeKNdgeXjMiWhXuH8A/playlists>
- [https://security.polito.it/~lioy/oimbe/pwr\\_2020\\_e8.pdf](https://security.polito.it/~lioy/oimbe/pwr_2020_e8.pdf)
- Le richieste HTTP (GET e POST)
  - <http://www.html.it/pag/62463/le-richieste-http-get-e-post/>
  - [https://www.tutorialspoint.com/php/php\\_get\\_post.htm](https://www.tutorialspoint.com/php/php_get_post.htm)
  - <http://www.ce.uniroma2.it/~lopresti/Didattica/RetiWeb/RetiWeb0809/HTTP.pdf>
- PHP
  - <http://pages.di.unipi.it/milazzo/teaching/AA1011-LabBD/PHP-ParteI.pdf>
- FORM
  - [https://www.tutorialspoint.com/php/php\\_form\\_introduction.htm](https://www.tutorialspoint.com/php/php_form_introduction.htm)
  - <https://www.geeksforgeeks.org/how-to-get-post-from-multiple-check-boxes/>



- Interazione con l'utente con i form
  - <https://security.polito.it/~lioy/oimbe/form.pdf>
  - <http://pages.di.unipi.it/milazzo/teaching/AA1011-WebProg/slides/9-datiEForm.pdf#page=19>
  - <https://www.php.net/manual/en/tutorial.forms.php>
  - <https://www.phptutorial.net/php-tutorial/php-form/>
  - [https://www.w3schools.com/php/php\\_forms.asp](https://www.w3schools.com/php/php_forms.asp)
  - <http://www.java2s.com/Code/Php/Form/CatalogForm.htm>
  - <https://replit.com/talk/learn/PHP-Form-Handling-Tutorial/56526>
  - <https://www.guru99.com/php-forms-handling.html>
  - <https://csis.pace.edu/~marchese/Cs396N/Lecture/Lec9/lec9.html>
  - [https://security.polito.it/~lioy/oimbe/pwr\\_2020\\_e8.pdf](https://security.polito.it/~lioy/oimbe/pwr_2020_e8.pdf)
  - <https://security.polito.it/~lioy/oimbe/>



# Revisioni



[v1.0 27/03/17](#) - versione iniziale

[v1.1 13/10/22](#) - ampliata l'introduzione teorica

[v1.2 06/10/23](#) - ampliata l'introduzione teorica e modificati gli esercizi

v1.3 25/09/24 - piccole correzioni