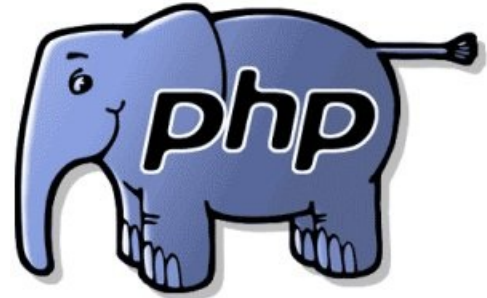




PHP programming



programmazione
lato server in PHP

Giulio Angiani
I.I.S. "Blaise Pascal" - Reggio Emilia



Gestione della sessione con PHP



Il concetto di Sessione nel Web

Il concetto di **sessione** in una applicazione web equivale a poter mantenere viva la comunicazione fra **client** e **server** attraverso la navigazione del sito

In generale, nel web, fra un clic è il successivo, il **server** dimentica l'esistenza del **client**

Per il server, il client potrebbe anche spegnere il computer o chiudere il browser e non ci sarebbe alcun problema.

La gestione della **sessione**, invece, permette al **server** di **ricordare** in qualche modo le operazioni effettuate precedentemente dal **client** e comportarsi di conseguenza.

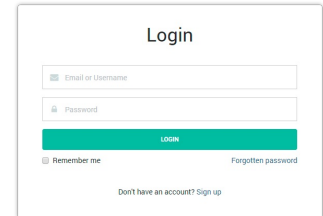


Il concetto di Sessione nel Web

Un esempio classico di interazione che rimane **viva** attraverso la navigazione è l'operazione di **login** di un utente

Tutti noi siamo abituati a fare **un solo login** in un sito. Non mettiamo **username** e **password** ogni volta che facciamo un **click**

Eppure il **server** si **ricorda** dell'operazione di **login** effettuata e si comporta diversamente secondo l'utente connesso



Login

Email or Username

Password

LOGIN

☐ Remember me [Forgotten password](#)

[Don't have an account? Sign up](#)

- la mia webmail mostra le mie comunicazioni e non quelle di un altro utente
- il sito della banca mi fa accedere ai miei conti e alle mie funzionalità, sicuramente diverse da quelle del direttore della banca
- Amazon ricorda i miei ordini pregressi e suggerisce nuovi acquisti in funzione delle mie caratteristiche



Gestire la Sessione in PHP

Per gestire il concetto di **sessione** in PHP ci viene in aiuto una struttura dati apposita: **\$_SESSION**

\$_SESSION è un array associativo **predefinito** del linguaggio che **mantiene vive** le informazioni attraverso la navigazione

NOTA:

- L'array `$_SESSION` viene popolato automaticamente da PHP ogni volta che viene chiamata una pagina. Se le impostazioni del PHP fossero diverse è necessario invocare la funzione **`session_start()`** per creare e popolare i valori di `$_SESSION`

```
<?php
    session_start();    // solitamente è la prima istruzione che viene eseguita

    [qui il codice della pagina]
?>
```

PHP

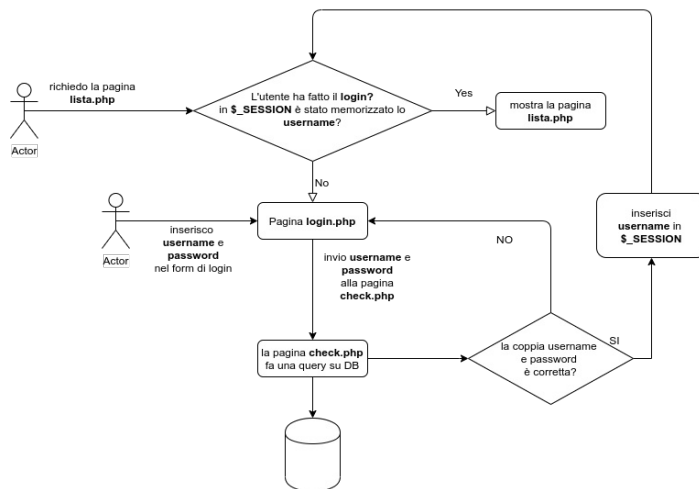


Una form di login

Capiamo l'uso di `$_SESSION` con l'esempio dell'operazione di login
Vogliamo che la nostra applicazione sia accessibile solo agli utenti registrati

Pertanto, se l'utente non ha fatto un login, non deve accedere a nessuna pagina.

Come funziona?: L'idea è di **costringere** l'utente ad inserire le sue credenziali (username e password) prima di poter accedere a qualsiasi pagina (tranne quella di login e di controllo delle credenziali, ovviamente)



Proteggere una pagina da accesso senza login

Per realizzare il nostro progetto è **fondamentale** avere un metodo che **impedisca** l'accesso ad una pagina senza che l'utente sia passato dall'autenticazione

Faremo in modo che **solo** la pagina **check.php** possa inserire la chiave **username** nell'array `$_SESSION` e solamente se l'utente ha inserito le credenziali valide

Tutte le pagine che devono essere protette dovranno:

controllare che `$_SESSION` contenga la chiave **"username"**

- se la contiene vuole dire che l'utente ha passato l'autenticazione
- altrimenti verrà rigirato sulla pagina di login



Proteggere una pagina da accesso senza login

Le istruzioni che permettono il controllo di accesso sono semplici

```
<?php
    session_start();    // solo se la sessione non viene attivata di default
    if (!isset($_SESSION["username"])) {
        header("Location: login.php");
        exit();
    }

    [... qui il codice della pagina protetta da login ...]

?>
```

PHP

Dobbiamo proteggere con queste istruzioni **tutte** le pagine per le quali vogliamo che l'utente acceda solo **dopo** un login



Progettare la pagina di login

Nella pagina di login si inseriscono solamente un campo per lo **username**, uno per la **password** e un bottone per inviare i dati via POST

```
[...]  
<!-- Login Form -->  
<form method=POST action='check.php'>  
  <input type='text' name='usr' placeholder='username'>  
  <input type='password' name='pwd' placeholder='password'>  
  <br>  
  <input type='submit' value='Log In'>  
</form>  
[...]
```

HTML



Progettare la pagina di controllo delle credenziali

Nel nostro caso abbiamo scelto di chiamare **check.php** la pagina che riceverà i dati dalla form di login

PHP

```
<?php
    session_start();
    include("connessione.php");

    // recupero dati da POST
    $usr = $_POST["usr"];
    $pwd = $_POST["pwd"];

    // costruisco la query da esquire sul DB.
    // La tabella studenti è quella che contiene username e password degli studenti registrati
    $sql = "SELECT * FROM studenti WHERE username = '$usr' and password = '$pwd' ";
    $resultset = $conn->query($sql);
    $righe = mysqli_fetch_all($resultset, MYSQLI_ASSOC);

    if (count($righe) > 0) {
        // se ho trovato righe vuol dire che esiste la riga username e password corrispondente
        // metto in $_SESSION il valore dello username
        $_SESSION["username"] = $usr;    // <== questa è l'operazione FONDAMENTALE
        // ridireziono a home.php
        header("Location: home.php");
    }
    else {
        header("Location: login.php"); // rimando alla pagina di login
    }
?>
```



Progettare la pagina di controllo delle credenziali

Nella pagina **check.php** viene effettuato il controllo dell'esistenza della coppia username/password nel database.

Se la query restituisce un risultato (e sarà solo 1 riga):



- l'utente esiste
 - Viene inserito in **\$_SESSION** il parametro che useremo per testare l'avvenuto login (nel caso specifico **username**)
 - Fino all'operazione di **logout** questo parametro sarà in **\$_SESSION** e **durante la navigazione** ogni pagina potrà testare l'avvenuto login
 - Si ridireziona l'utente ad una **landing-page** apposita (nel nostro caso **home.php**)
- altrimenti
 - Nulla viene inserito in **\$_SESSION**
 - l'utente è ridirezionato alla pagina di login dalla quale proviene
 - di solito si aggiunge una scritta che indica che "username o password sono errati"

Miglioramenti

Come abbiamo fatto per la gestione della connessione

Le operazioni di controllo di login possono essere raggruppate in una pagina apposita (es: **tools.php**) dove creiamo una funzione ad-hoc per il controllo

in ogni pagina **protetta** richiameremo questa funzione



```
<?php
// tools.php
function checkUserLogged() {
    if (!isset($_SESSION["username"])) {
        header("Location: login.php");
        exit();
    }
}

?>
```

PHP



Miglioramenti

In ogni pagina protetta scriveremo ora

```
include "tools.php";  
checkUserLogged();
```

PHP

ottenendo lo stesso risultato ma in maniera più compatta

Inoltre, in caso di modifica della gestione del login, dovremo cambiare solo la funzione **checkUserLogged**



Miglioramenti (2)

In caso di errore di login rimandiamo alla pagina login.php specificando il tipo di errore

```
header("Location: login.php?loginerror=1"); // rimando alla pagina di login
```

PHP

nel file **login.php** mostriamo una label in caso l'utente provenga da un errato login

```
// PHP
$message = "";
if (isset($_GET["loginerror"])) {
    // provengo da check con un messaggio di errore
    $message = "username o password errati";
}
```

PHP

```
// HTML
<div class='text-danger'>
    $message
</div>
```

HTML



The screenshot shows a login interface. At the top, there is a large rounded square logo with a blue and white stylized sailboat icon and the text 'ifoa' in blue. Below the logo, there is a text input field labeled 'username'. To the right of the input field, the text '14/15' is visible. The entire form is enclosed in a thin black border.





Giulio Angiani
I.I.S. "Blaise Pascal" - Reggio Emilia