

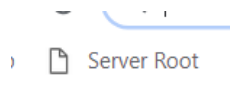
## Laboratorio 7 – Mercoledì 18/11/2020

- A partire da questa lezione sarà necessario utilizzare quanto presente nel pacchetto All-in-one.
- Il programma fondamentale da eseguire ogni volta che dobbiamo lavorare è **XAMPP**.

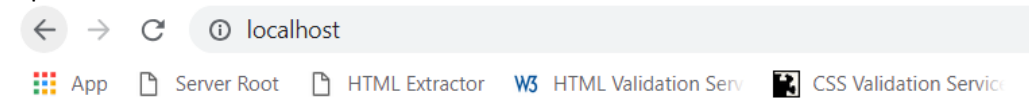


Con l'attivazione del server Apache (sulla porta 80) simuleremo il protocollo HTTP, ovvero la connessione client-server (in questo caso client e server coincidono, il server è *localhost*). L'attivazione di Apache permette anche di eseguire PHP in locale.

- Abbiamo già visto che il pacchetto *All-in-one* offre due browser *portable*: in entrambi troviamo un certo segnalibro



Cliccandolo potremo accedere alla root



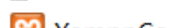
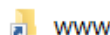
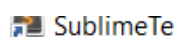
## Index of /

Name	Last modified	Size	Description
<a href="#">PrimaEsercitazione/</a>	2020-10-10 21:08	-	
<a href="#">js.html</a>	2020-11-12 11:51	873	
<a href="#">prova.html</a>	2020-10-11 11:23	53	
<a href="#">test.php</a>	2020-10-08 09:13	19	

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 Server at localhost Port 80

**Osservazione:** noi vediamo la lista dei documenti poiché non è presente un file `index.html` o un file `index.php` nella root. Chiaramente una lista del genere non dovrebbe mai essere resa visibile in un sito pubblico (*gli hacker festeggiano*, cit.)

**Come possiamo collocare documenti nel nostro server locale?** Attraverso il collegamento `www` nella cartella pweb.



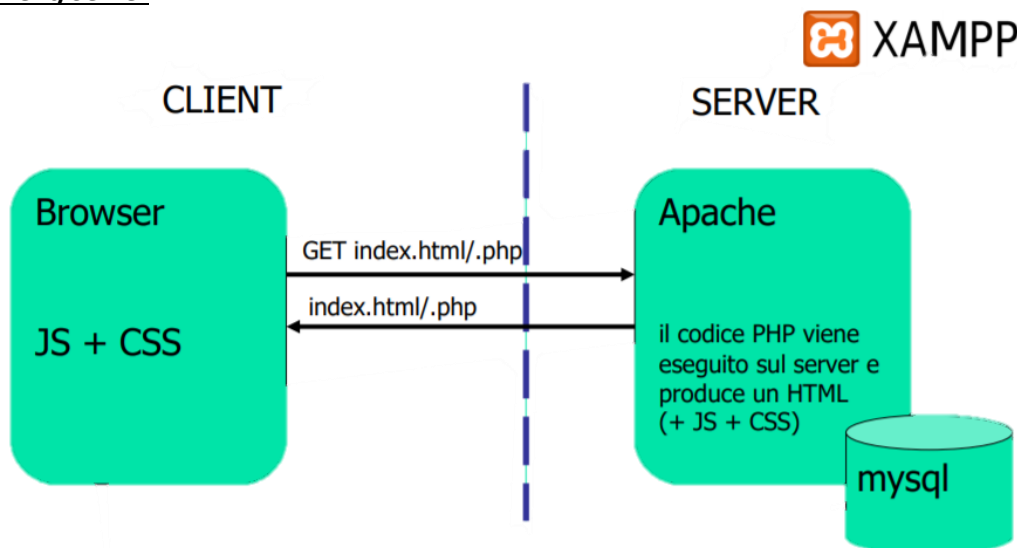
- Proviamo il file `test.php`, che offre una sintesi di quanto installato sul nostro server Apache, in particolare riguardo il PHP. È consigliato visitare questo file ogni volta che dobbiamo lavorare in PHP: possiamo vedere la versione di PHP (in questo corso utilizzeremo PHP5), costanti con il loro valore, DOM, mysqli, codifiche hash disponibili....

### PHP Version 5.5.15



System	Windows NT LAPTOP-2CHPPJP4 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Build Date	Jul 23 2014 14:58:09
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\x86\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\x86\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\x86\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\pweb\tools\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files	(none)

### Architettura Client/Server



- È fondamentale tenere a mente quando si programma come e dove vengono eseguiti i nostri codici.
- La parte server è gestita da XAMPP (ribadisco, noi simuliamo un server), la parte client dal nostro browser.
- Il client fa richieste HTTP verso un server, il server risponde col relativo codice.
- Il codice PHP viene eseguito sul lato server, che produce (al di là dell'estensione) un codice HTML!!! Il client otterrà in risposta un codice HTML (la pagina da caricare) con relativo CSS e JS.
- Il server si limita a restituire il codice JS e CSS: ci pensa il browser a gestire questi due linguaggi (linguaggi standardizzati con differenze minime tra i principali browser).

### var\_dump

- Introduciamo la funzione `var_dump` con cui possiamo stampare in modo esplicito il contenuto di una variabile. Possiamo vedere il tipo di variabile, la lunghezza della variabile e il suo contenuto.  
`string(5) "prova"`

- Adesso la cosa può risultare insignificante, ma vedremo come `var_dump` sia VITALE quando lavoreremo su array di dimensione elevata. Se ponete il seguente codice  

```
echo '<pre>';
var_dump($variabile);
echo '</pre>';
```

 Stamperemo l'array in modo ordinato visualizzando i vari elementi dell'array, le proprietà, ed eventuali subarrays.

### Esercizio della tavola pitagorica

- Scrivere un'applicazione PHP che disegna una tavola pitagorica<sup>1</sup> grande  $N \times N$ .
- Versione con valore N fisso

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tavola Pitagorica</h1>
  <table border = "1">
    <?php
      for ($i = 1; $i <= 10; $i++) {
        print "<tr>";
        for ($j = 1; $j <= 10; $j++) {
          $r = $i*$j;
          print "<td>".$r."</td>";
        }
        print "</tr>";
      }
    <?>
  </table>
</body>
</html>
```

**Esempio**

```
$r=6*8=48
print "<td>48</td>";
```

## Tavola Pitagorica

\$i	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100
\$j	1	2	3	4	5	6	7	8	9	10

- Versione con valore N indicato da noi (metodo GET). Servono due file: un file PHP che restituisce la tavola pitagorica e un file html con cui noi indichiamo N.
  - o index.html

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tavola Pitagorica</h1>
  <form action="tavola.php">
    N: <input type="text" name="N">
    <input type="submit">
  </form>
</body>
</html>
```

## Tavola Pitagorica

N:

- o tavola.php (con la sottomissione della form sarà aperta la pagina `tavola.php?N=numero`)

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tavola Pitagorica</h1>
  <table border = "1">
    <?php
      $N = $_GET['N'];
      for ($i = 1; $i <= $N; $i++) {
        print "<tr>";
```

**Unica differenza:** nelle condizioni dei due `for` abbiamo sostituito la costante 10 con la variabile `$N`

Il valore di `$N` è ottenuto dall'array associativo `$_GET`

<sup>1</sup> Una **tavola pitagorica** è una matrice di numeri naturali dove ciascun elemento di posizione  $i, j$  consiste nel prodotto  $i \cdot j$

```

        for ($j = 1; $j <= $N; $j++) {
            $r = $i*$j;
            print "<td>".$r."</td>";
        }
        print "</tr>";
    }
    ?>
</table>
</body>
</html>

```

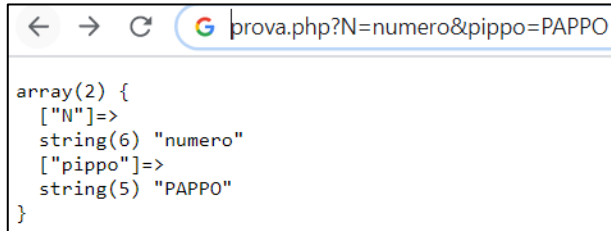
Per avere un'idea più chiara eseguite il seguente codice:

```

echo '<pre>';
var_dump($_GET);
echo '</pre>';

Vedrete tutta la struttura di $_GET

```



```

array(2) {
    ["N"]=>
    string(6) "numero"
    ["pippo"]=>
    string(5) "PAPPO"
}

```

- Versione con valore N indicato da noi (metodo POST). Servono due file: un file PHP che restituisce la tavola pitagorica e un file html con cui noi indichiamo N.

o index.html

```

<!DOCTYPE html>
<html>
<body>
    <h1>Tavola Pitagorica</h1>
    <form method="post" action="tavola.php">
        N: <input type="text" name="N">
        <input type="submit">
    </form>
</body>
</html>

```

#### Differenze:

- method modificato
- array associativo da cui prendiamo il valore \$N diverso

o tavola.php (con la sottomissione della form sarà aperta la pagina tavola.php)

```

<!DOCTYPE html>
<html>
<body>
    <h1>Tavola Pitagorica</h1>
    <table border = "1">
        <?php
            $N = $_POST['N'];
            for ($i = 1; $i <= $N; $i++) {
                print "<tr>";
                for ($j = 1; $j <= $N; $j++) {
                    $r = $i*$j;
                    print "<td>".$r."</td>";
                }
                print "</tr>";
            }
        ?>
    </table>
</body>
</html>

```

Per avere un'idea più chiara eseguite il seguente codice:

```

echo '<pre>';
var_dump($_POST);
echo '</pre>';

Vedrete tutta la struttura di $_POST

```

- **Quali sono le differenze?** In caso di metodo GET tutte le variabili sottomesse stanno nel link, mentre nel metodo POST queste sono nascoste. Inoltre la dimensione dei valori col metodo GET è limitata (comunque si parla di dimensioni elevate).

## Login e password

- Vogliamo realizzare il nucleo di un sistema login in cui, dato un username e una password, si dice all'utente se i dati inseriti sono corretti.
- Nei file di Tesconi è presente la versione scritta all'inizio del laboratorio, qua troverete la versione già corretta e il perché questa versione sia quella migliore.

### - index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>LOGIN</title>
  <style>
    body { font-family: "Lucida Console"; }
  </style>
</head>
<body>
  <form method="post" action="login.php">
    Username: <input type="text" name="username"></input><br>
    Password: <input type="password" name="password"></input><br>
    <input type="submit" value="Login"></input>
  </form>
</body>
</html>
```

Username:

Password:

Quando si gestiscono dati riservati è importante utilizzare il metodo `$_POST` (col metodo `$_GET` sarebbe visibile la password nell'URL, non il massimo della sicurezza).

Utilizziamo due funzioni:

- `password_hash($passw_da_codificare, tipo_codifica)`
- `password_verify($passw_da_verificare, $hash_passw_giusta)`

### - login.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>LOGIN</title>
</head>
<body>
<?php
$pwd_hash = "$2y$10$4oGZge6quDgEXdXpBRAQO.p9iR5NR.rN4sJq18ZWohndF//8rTYOS";
// Abbiamo ottenuto il valore di $pwd_hash con la seguente funzione:
password_hash("pippo", PASSWORD_BCRYPT);

if (isset($_POST['username']) && isset($_POST['password'])) {

if ($_POST['username'] == "maurizio" && password_verify($_POST['password'], $pwd_hash)) {
  print "LOGIN AVVENUTO CON SUCCESSO";
} else {
  print "LOGIN ERRATO";
}

  }
  ?>
</body>
</html>
```

Se l'username è maurizio e la password è pippo  
(password\_verify restituisce un booleano) allora  
stampa "LOGIN AVVENUTO CON SUCCESSO"

### Attacco forza bruta per individuare password

- Quando codifichiamo le password conviene sempre utilizzare le funzioni più aggiornate.
- Una funzione molto popolare e standard dieci anni fa è la md5(). Con i tempi dei calcolatori di allora decodificare una password criptata in md5 richiedeva un tempo inumano (anche dieci anni). Adesso, con l'emergere dei computer quantistici e in generale di calcolatori più veloci, ci vuole molto meno tempo (l'uso di questa funzione è caldamente sconsigliato, noi non la useremo).
- Al di là di queste questioni dimostriamo attraverso il seguente codice PHP come sia facile trovare una password nabba (precisamente una password con nome e anno di nascita: maurizio1974)

```
<?php
$pwd_hash = "b0ce88403a33765ce720cf0d30a7456b"; //md5("maurizio1974");

$file = fopen("nomi_italiani.txt","r");
$found = false;

while(!feof($file)) {
    $nome = fgets($file);

    foreach (range(1970, 1990) as $key => $value) {
        $pwd_test = trim($nome).$value;
        $pwd_md5 = md5($pwd_test);
        if ($pwd_md5 == $pwd_hash) {
            print "TROVATA! ".$pwd_test;
            $found = true;
        }
    }
}

if (!$found) print "PWD NON TROVATA!";
fclose($file);
?>
```

File di testo contenente migliaia di nomi comuni.  
[Lo potete scaricare liberamente da Github](#)

Chiaramente troveremo la password  
maurizio1974

- Utilizziamo diverse funzioni PHP per scorrere il nostro file di testo.
- Per ogni riga considero un array di elementi che va dal numero 1970 al numero 1990 (genero l'array con la funzione range).

nomepersonaANNO

**Osservazione:** il docente ha eseguito questo file PHP su server Apache in una macchina virtuale di Windows a basse prestazioni. Capite da tutto questo che password di questo tipo sono facili da trovare!!

### Indovina numero

- Recuperiamo un esercizio che abbiamo già fatto con Javascript.
- Ogni volta che ricarichiamo una pagina PHP le variabili saranno re-inizializzate. Come possiamo tenere a mente un numero che una persona deve indovinare? Ricorriamo alle variabili \$\_SESSION.

- Contrariamente ai form precedenti gestiamo il tutto in un'unica pagina (index.php):

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"> <title>INDOVINA IL NUMERO - PHP</title>
</head>
<body>
    <?php
        session_start();

        if (!isset($_SESSION['num']) || isset($_GET['avvia'])) {
            $_SESSION['num'] = rand(1,100);
            //print "MI SONO INVENTATO ".$_SESSION['num'];
        }
    ?>
```

Rigenero il numero in due casi: quando ci connettiamo alla pagina per la prima volta e la variabile \$\_SESSION['num'] non è impostata, quando indichiamo un valore in \$\_GET['avvia']

Tenere a mente le funzioni isset() e rand(): con la prima verifichiamo se esiste la variabile, con la seconda generiamo un numero randomico appartenente all'intervallo indicato (non dobbiamo fare le rotture viste in Javascript)

```
<h3>Prova a indovinare il numero che ho pensato!</h3>
<form action="index.php">
  <input name="num" type="text" autofocus></input>
  <input type="submit"></input>
  <button name="avvia" value="true">Riavvia</button>
</form>

<?php
if (isset($_GET['num']) && is_numeric($_GET['num'])) {
  print "Hai provato con ".$_GET['num'];

  if ($_SESSION['num'] > $_GET['num']) {
    print "<h4>Il numero che ho pensato è più grande!</h4>";
  }
  else if ($_SESSION['num'] < $_GET['num']) {
    print "<h4>Il numero che ho pensato è più piccolo!</h4>";
  }
  else {
    print "HAI VINTO!!!";
  }
}
else {
  print "devi inserire un numero per giocare";
}
?>
</body>
</html>
```

Ricordarsi che il button con name="avvia" sarà validato solo se premuto.

Con la funzione `is_numeric()` controllo se il valore di `$_GET['num']` consiste in un numero.

**Prova a indovinare il numero che ho pensato!**

devi inserire un numero per giocare

**Osservazione:** quanto fatto non è il massimo dell'ottimizzazione. Scelta migliore è fare una pagina del genere ricorrendo ad AJAX.