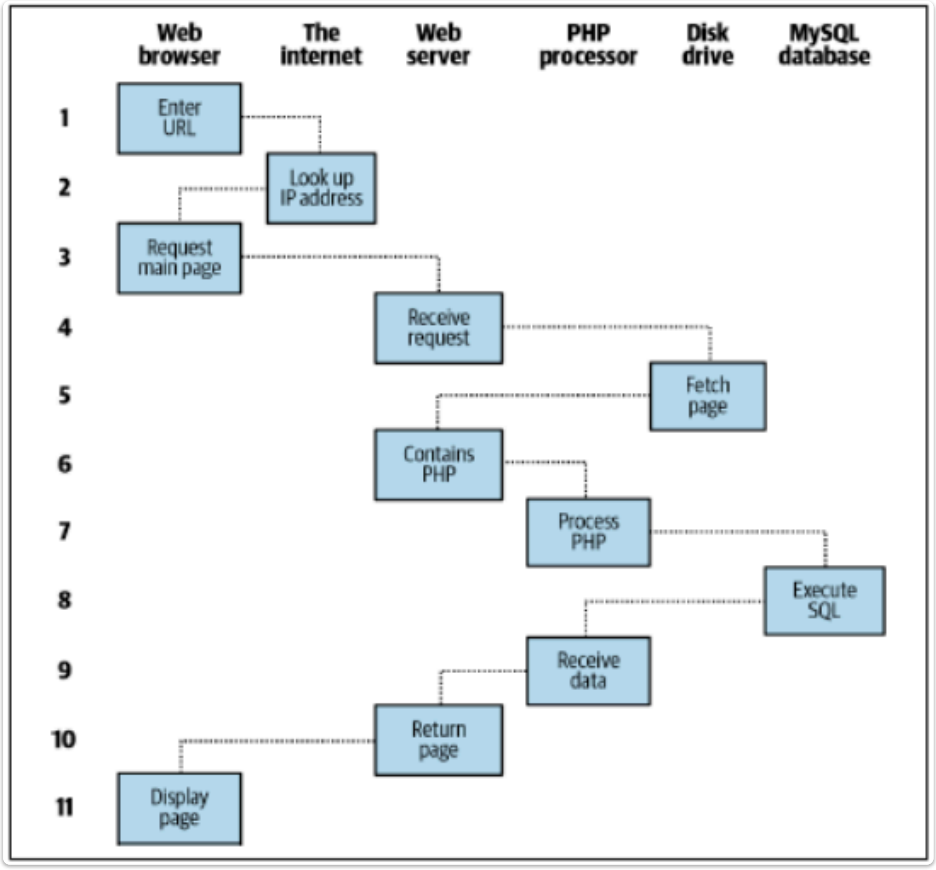
PHP HyperText Preprocessor, linguaggio di programmazione per estendere HTML. Il linguaggio convive all'interno dei documenti HTML e viene eseguito dal server prima che la pagina venga inviata all'utente. Si integra in un documento mediante i tag:

```
<?php
?>
```

# 1 Architettura


1. Browser richiede al server un documento `.php`;
2. Server invia il documento al modulo PHP che interpreta lo script, produce un output e lo restituisce (il modulo interpreta solo il codice PHP, tutto quello che non è PHP rimane invariato);
3. Il server risponde alla richiesta del browser inviando l'output del modulo PHP




# 2 Dettagli del linguaggio

Il linguaggio è case insensitive rispetto ai costrutti nativi ma è buona norma adottare sempre lo stesso stile. Generalmente si applicano le convenzioni Java. Qui altre best practice: https://www.php-fig.org/psr/.

## 2.1 Variabili

Sono rappresentate dal simbolo `$` e sono case-sensitive. Il nome può iniziare esclusivamente con una lettera oppure un underscore `_`. Non è necessario dichiararle. Il linguaggio non è tipizzato e le variabili sono inizializzate automaticamente in base all'ambito in cui vengono usate:

- 0, ambito numerico;
- "", stringhe;
- false, booleano

È buona norma inizializzare sempre le variabili per tenere sotto controllo il codice. Si può usare il metodo `isset()` per verificare se una variabile è assegnata ed ha un valore diverso da `null`.
Il metodo `unset()` consente di rimuovere completamente una variabile dall'ambiente PHP.

Le variabili sono visibili in tutto lo script PHP ma non nelle funzioni se non passate come parametro. Le variabili locali (ossia dichiarate in una funzione) sono invece visibili solamente nella funzione e nelle definizioni annidate. Tali variabili vengono distrutte, secondo il meccanismo dei record di attivazione, nel momento in cui la funzione termina.
Il modificatore `static` consente di lasciare in vita le variabili locali anche al termine della relativa funzione. Chiamate successive alla stessa funzione manterranno il valore precedente che aveva la variabile al termine della precedente chiamata.

Le variabili globali, ossia visibili in tutto il codice, si dichiarano mediante la keyword `global`.

### 2.1.1 Variabili variabili

Sono variabili il cui nome è determinato dinamicamente in base al valore di una variabile. L'accesso si effettua mediante il doppio `$$`:

```
$foo = 'bar';
$$foo = 'ciao';

/* Dopo la 2 istruzione, la variabile $bar contiene il valore "ciao" */
```

### 2.1.2 Variabili riferimento

Consentono di creare un riferimento ad una variabile esistente, in modo tale che qualsiasi modifica alla variabile originale, venga riflessa anche sul riferimento. Si usa il simbolo `&`:

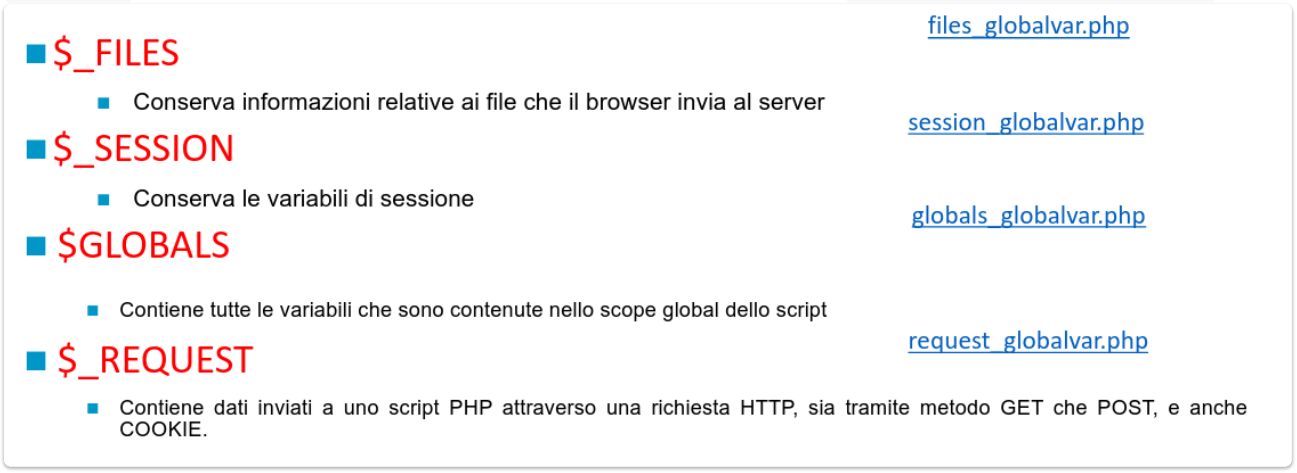
```
$nero = &$bianco;
/* Le due variabili fanno riferimento alla stessa area di memoria.
   Si puó modificare il valore di $nero attraverso la variabile $bianco.
*/
```

### 2.1.3 Variabili superglobali

Gli script `PHP` possono accedere ad un insieme di informazioni eterogenee, dette EGPCS (Enviroment, GET, POST, Cookie and Server). L'accesso a tali informazioni avviene tramite delle particolari variabili dette variabili superglobali, che vengono create automaticamente dall'ambiente PHP e sono accessibili ovunque:

1. `$_ENV`: informazioni sull'ambiente di esecuzione del server web (hostname, porta, protocollo, ecc...);
2. `$_GET`: informazioni trasmesse tramite metodo HTTP GET;
3. `$_POST`: informazioni trasmesse tramite metodo HTTP POST;

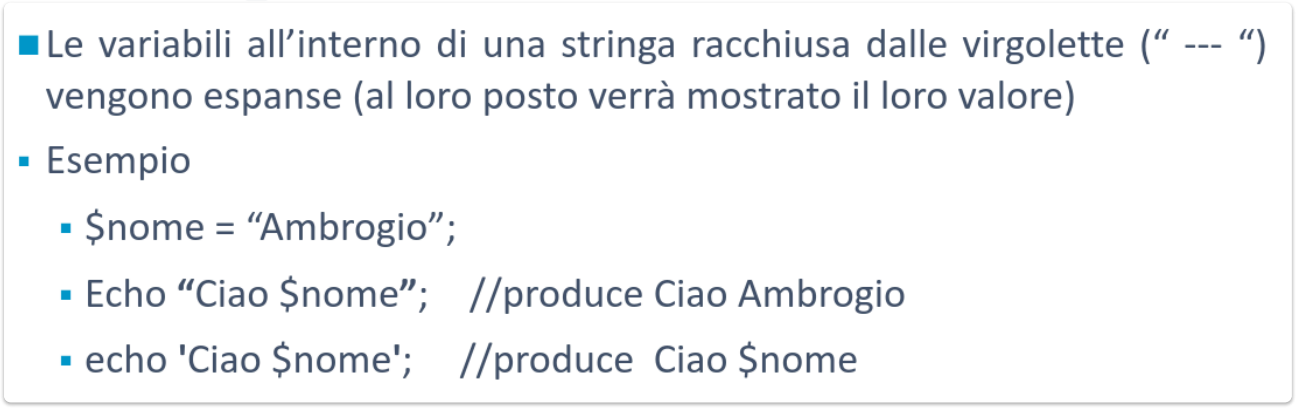
4. `$_COOKIE` : conserva eventuali cookie passati tramite una richiesta HTTP;
5. `$_SERVER` : conserva informazioni sulla configurazione del server. Eg. `$_SERVER['DOCUMENT_ROOT']`




## 2.2 Tipi di dato

Esistono 8 tipi di dato: 4 scalari, 2 composti e 2 speciali.

- Scalari: interi, float, booleani, stringhe;
- Composti: array ed oggetti;
- Speciali: null e risorse (una risorsa è ad esempio il riferimento ottenuto dopo l'apertura di una connessione ad un database)
  Per quanto riguarda le stringhe, PHP supporta il meccanismo del globbing, allo stesso modo di `bash` .
  Variabili inserite tra `"` vengono quindi espanse e sostituite con il loro valore.




Si possono inserire stringhe su più linee usando l'operatore `<<<` , chiamato anche herodoc.

```
<<<etichetta
testo
etichetta
```

- Etichetta: identificatore di testo arbitrario. Solitamente si usa `\_END` oppure `HTML` ;

- Testo: stringa effettiva;

**Invece di avere una situazione del tipo:**

```
$name = 'Simone';
$html = '<ul>';
$html .= '<li>' . $name . '</li>';
$html .= '<li>elemento 1</li>';
$html .= '<li>elemento 2</li>';
$html .= '<li>elemento 3</li>';
$html .= '</ul>';
echo $html;
```

**Possiamo usare Heredoc in questo modo:**

```
$name = 'Simone';
$html  = <<< HTML
<ul> <li>$name</li>
      <li>elemento 1</li>
      <li>elemento 2</li>
      <li>elemento 3</li> </ul>
HTML;
echo $html;
```

heredoc.php

## 2.3 Commenti

```
// Questo é un commento
# Questo é un altro commento
/* Questo
   é un commento
   su piú linee
*/
```

## 2.4 Costanti

Le costanti possono contenere solamente valori scalari. Si usa la keyword `define`:

```
define('NomeCostante', Valore);
define('Massimo', 49);
echo Massimo
```

L'accesso alla costante avviene direttamente mediante il suo nome, non occorre usare `$`. Hanno scope globale.

## 2.5 Sequenze di Escape

## Sequenze di escape più comuni

- \n = newline (nuova riga)
- \r = carriage return (ritorno a capo)
- \t = carattere di tabulazione (TAB)
- \$ = carattere dollaro ($)
- \" = carattere doppi apici (")
- \\ = carattere backslash (\)

- Se l'output che lo script deve generare contiene delle virgolette, allora bisogna far precedere le virgolette dal backslash (\) ed includere l'output nelle virgolette doppie
- Ad esempio

```
print ("<h1 style=\"color:blue;\">");
```

produce in output: <h1 style="color:blue;">

Il linguaggio mette a disposizione anche alcune particolari funzioni che convertono una serie di caratteri nelle relative sequenze di escape. Sono molto utili in quelle situazioni in cui una stringa contiene dei caratteri che possono essere interpretati come elementi HTML come `<` `>` `&`.

## Caratteri speciali HTML

- A volte in una stringa ci possono essere dei caratteri che possono essere interpretati come parte di elementi HTML (e.g., >, <, &, …)
- PHP fornisce la funzione htmlspecialchars()

che converte tutti questi caratteri nelle relative sequenze di escape (entità);

Il mio obiettivo è che a video venga mostrato (es: sto scrivendo un tutorial su html)

```
This is some <b>bold</b> text.
```

```
<?php
$str = "This is some <b>bold</b> text.";
echo htmlspecialchars($str);
?>
```

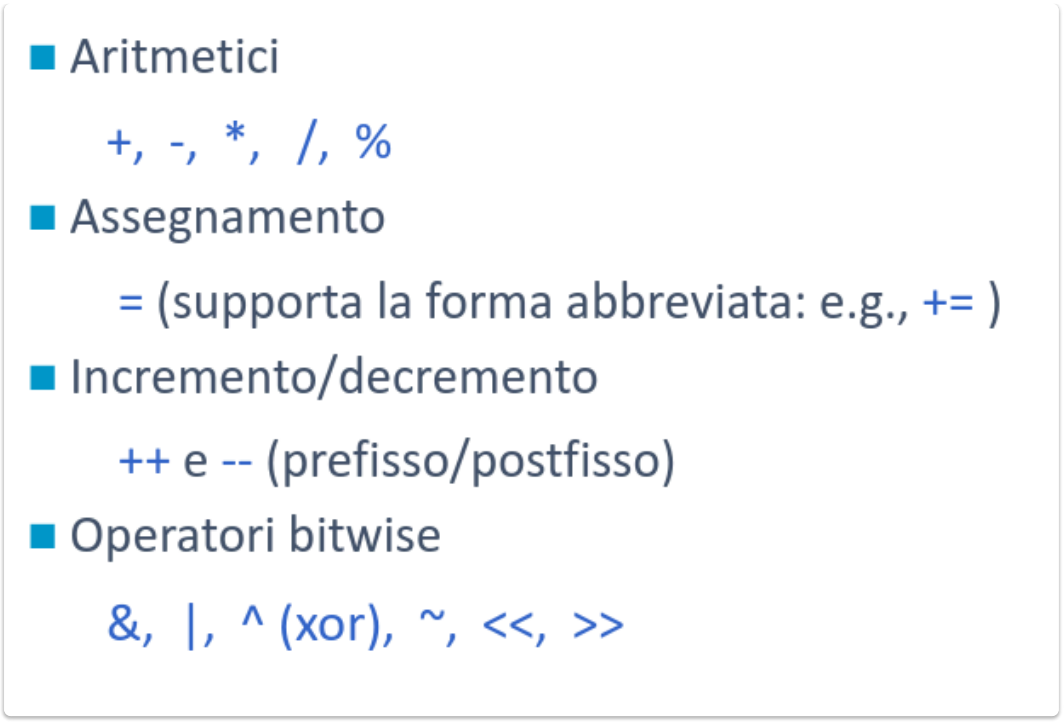
Se non utilizzo htmlspecialchars() a video vedrò

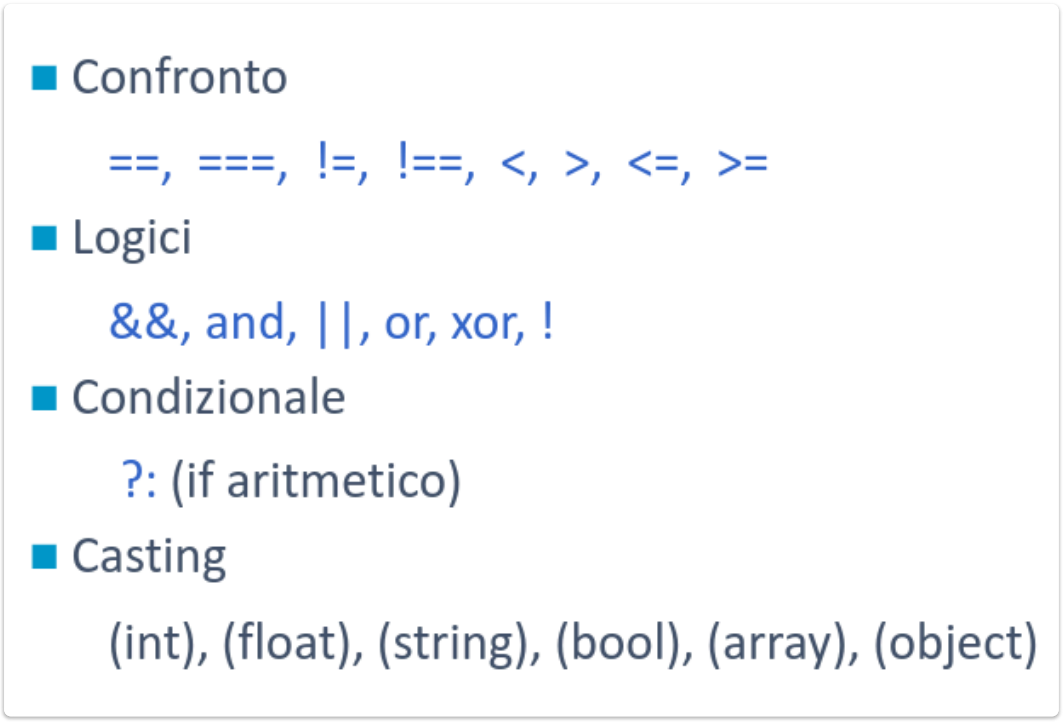
This is some **bold** text..

In HTML diventa:

```
<body>
This is some &lt;b&gt;bold&lt;/b&gt; text.
</body>
```

## 2.6 Operatori




Gli operatori sono pressoché identici a quelli di altri linguaggi come `C` o `Java`.

### 2.6.1 Confronto

- `==` testa se due espressioni sono uguali; `"1" == "1" -> true ; "1" == 1 -> true`
- `===` testa se due espressioni sono identiche; `"1" = "1" -> true; "1" = 1 -> false

## 2.7 Includere codice

PHP consente di includere un documento (o una libreria) in uno script PHP mediante 2 direttive:

1. `include`: in caso di fallimento genera un semplice warning. Generalmente si usa per includere codice HTML.
2. `require`: in caso di fallimento genera un errore fatale che interrompe la visualizzazione a schermo. Generalmente si usa per includere librerie;

Le varianti `require_once` ed `include_once` garantiscono che l'inclusione venga effettuata una sola volta.

## 2.8 Funzioni

Si possono definire funzioni mediante la seguente sintassi:

```
function NomeFunzione([lista parametri]){
    implementazione;
}
```

Il nome della funzione è case-insensitive. Ad ogni parametro è possibile assegnare anche un valore di default effettuando un'assegnazione direttamente nella lista dei parametri passati.

```
<?php  // il terzo parametro ha un valore di default
 function descrivi_film($titolo, $regista, $bn_colori = 'colori')
 {  echo "Titolo: $titolo <br>";
    echo "Regista: $regista <br>";
    echo "B/N - colori: $bn_colori <br>";
 }

 // chiamo la funzione passando 2 soli parametri
// il terzo è sottointeso
 descrivi_film('Il malato immaginario','Tonino Cervi');
echo '<br>';

 // chiamo la funzione passando tutti e 3 i parametri
 descrivi_film('I soliti ignoti','Mario Monicelli','bianco e nero');
 ?>
```

L'istruzione `return` consente alla funzione di restituire un valore. Per restituire più valori si costruisce un array e si restituisce quello.

## 2.9 Array

PHP offre la possibilità di utilizzare come indici per gli array sia numeri che stringhe... in questo secondo caso parliamo di array associativi. La creazione e l'inizializzazione possono essere effettuati in diversi modi:

```
$disney[] = "pippo";
$disney[2] = "pluto";
$disney = array("pippo", "pluto", "paperino");
$disney = array("n1" => "pippo", "n2" => "pluto", "n3" => "paperino"); // usando => si
possono specificare gli indici per ogni elemento
```

L'operatore `=>` consente di specificare gli indici di ogni elemento dell'array:

```php
$frutta = array(1=>"mele", 2=>"pere", 3=>"banane");
```

### 2.9.1 Accesso ad array

Gli indici partono da 0, per conoscere la lunghezza dell'array si può usare la funzione `count`.L'accesso ad elementi non inizializzati non genera alcun errore. L'accesso ad array associativi avviene mediante una stringa:

```php
$corso['rundo'];
$corso["parziale"];
$corso[$nome];
```

Per l'accesso può tornare utile l'impiego di un ciclo `foreach` che consente di iterare automaticamente su tutti gli elementi dell'array.

```php
foreach($nomeArray as $Elemento){
    istruzioni;
}
```

La variabile $Elemento assumerà tutti i valori contenuti nell'array, uno per ogni iterazione.
Volendo è possibile scorrere contemporaneamente chiavi e valori:

```php
foreach($nomeArray as $indice => $elemento){
    istruzioni;
}
```

- `$indice` assume i valori delle chiavi;
- `$elemento` assume i valori degli elementi contenuti nell'array

Altre funzioni su iteratori:

1. `current()` : restituisce il valore dell'elemento puntato dall'iteratore corrente;
2. `key()` : restituisce la chiave dell'elemento corrente;
3. `reset()` : muove l'iteratore al 1 elemento dell'array;

```php
<h2>utilizzo di iteratori su array</h2>
<?php
$linguaggi = $arrayName = array('calcolatori' =>'assembler' ,'tsw' => 'php', 'fondamenti' => 'c' );
reset($linguaggi);
echo "<table style=\"border:solid 1px;\">\n";
echo "<tr><th>Chiave </th><th>Valore </th></tr>";
do{
  echo "<tr><td>".key($linguaggi)."</td><td>".current($linguaggi)."</td></tr>\n";
}while(next($linguaggi));

echo"</table>\n";
 ?>
```

### 2.9.2 Ordinare array

- sort()
  - Ordina i valori dell'array (crescente)
  - distrugge il valore delle chiavi
- asort()
  - Ordina i valori dell'array conservando l'associazione tra il valore e la sua chiave
- ksort()
  - Ordina le chiavi conservando l'associazione tra il valore e la sua chiave
- rsort() – arsort() – krsort()
  - Ordinamento in ordine inverso

### 2.9.3 Conversione tra variabili ed array

PHP fornisce le funzioni `extract()` e `compact()` che permettono di creare variabili da array e viceversa.

```php
$frutta = array("uno"=>"limone", "due"=>"arancia", "tre"=>"banana");
extract($frutta);
```

Crea le variabili `$uno, $due, $tre` con i valori: limone, arancia e banana.
`compact()` effettua l'operazione opposta: a partire da una lista di variabili, crea un array associativo, le cui chiavi sono i nomi delle variabili:

```php
$nome = "ambrogio";
$cognome = "unz";
$corso = "tsw";
$record = compact('nome', 'cognome', 'corso');
// oppure
$chiavi=array('nome', 'cognome', 'corso');
$record=compact($chiavi)
```

### 2.9.4 Conversioni tra array e stringhe

- `explode()`: converte una stringa in un array. Si può usare su una stringa per ottenere un array di sotto-stringhe individuate dal delimitatore: `explode(delimitatore, stringa [, limite])`;

```php
$pizza = "mozzarella pomodoro basilico olio";
$ingredienti = explode(" ", $pizza);```

- `implode()`: converte un array in una stringa. Si puó applicare ad un array per ottenere una stringa rappresentante gli elementi dell'array divisi da un delimitatore: `implode(separatore, array)`;
```php
$array = array('ambrogio', 'ambunz@unisa.it');
$stringa = implode(",", $array);
```

### 2.9.5 Ricerca in array

- `array_key_exists(key, array)`: true se esiste la chiave nell'array;
- `in_array(valore, array)`: true se esiste il valore nell'array;
```

### 2.9.6 Array multidimensionali

Un array multidimensionale è un array che contiene altri array:

```
$riga0 = array(1, 2, 3);
$riga1 = array(4, 5, 6);
$riga2 = array(7, 8, 9);
$multi = array($riga0, $riga1, $riga2);
print_r($multi);
```

# 3 Interazione con HTML

Per passare dati a script PHP si possono utilizzare diverse tecniche, é possibile per esempio utilizzare i `form` di HTML oppure usare `js/ajax`.

```
<form method="GET" action="form.php">
<input type=”text” name=”pippo”/>
</form>
```

Ogni variabile relativa ad un modulo è memorizzata in un array associativo accessibile attraverso il nome del controllo: `$_GET['pippo´]`. Se si usa `POST` invece:

```
<form method="POST" action="form.php">
<input type=”text” name=”pippo”/>
</form>****
```

Il valore viene salvato nella variabile `$_POST['pippo']`. Gli indici degli array associativi `$_GET, $_POST, $_FILES` sono i `name` usati all'interno degli input del form.

## 3.1 Checkbox multiple

- Checkbox selezionata: viene passato il valore dell'attributo `value` se presente altrimenti `"on"`.
- Checkbox non selezionata: non viene inviato nessun valore.
  Le checkbox multiple tuttavia devono avere lo stesso attributo `name` per cui, per evitare che venga passato solamente l'ultimo valore, è necessario aggiungere le parentesi `[]` al name del checkbox per indicare che si vuole passare un array.

```
<input type="checkbox" name="gusto[]" value="Cioccolato"/>
```

Lo stesso discorso si applica al tag `select` di HTML:

```
<select name=“linguaggi[ ]” multiple>
<option value=“c”> C </option>
<option value=“cpp”> C++ </option>
<option value=“java”>Java </option>
</select>
```

## 3.2 Passaggio di parametri con GET senza modulo

Il passaggio dei parametri tramite GET può essere effettuato direttamente utilizzando la URL: `?param1=value1&param2=value2…`.

# 4 Moduli autochiamanti

Si può usare un solo script PHP per generare un modulo e per elaborarlo: modulo autochiamante. La variabile `$_SERVER['PHP_SELF']` ci consente di conoscere il nome dello script corrente.

# 5 Sticky Form

Sono script che, una volta terminata l'elaborazione, continuano a mostrare i valori inseriti dall'utente e il modulo stesso.

```
<html>
<head><title> Conversione di Temperatura </title></head>
<body>
<?php $fahr =$_GET['fahrenheit']; ?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Temperatura Fahrenheit :
<input type="text" name="fahrenheit"
value="<?php echo $fahr ?>">
<input type="submit" name="Converti in Celsius!"> </form>
<?php
if (!is_null($fahr)) {
$celsius = ($fahr -32)*5/9;
printf("%.2fF &egrave; %.2fC", $fahr, $celsius);
}
?> </body> </html>
```

Una volta effettuato il `submit`, il form continua a mostrare le informazioni inserite dall'utente. Nel caso in cui siano presenti delle checkbox, é necessario salvare manualmente i valori `value` delle stesse in quanto vengono inviate solamente le checkbox selezionate dall'utente.

```
function make_checkboxes ($name, $query, $options ) {
    foreach ($options as $value => $label) {
        printf('<input type="checkbox" name="%s[ ]" value="%s" ', $name, $value);
        if (in_array($value, $query)) {
            echo "checked ";
        }
        echo "> $label <br>\n";
    }
}
```

# 6 Upload di file

L'upload di file si effettua mediante il tag `<input type="file" name="NomeFile">` e settando l'attributo `enctype="multipart/form-data"` a livello del form.

Le informazioni sul file inviato vengono salvate nell'array `$_FILES["NomeFile"]`. Si tratta di un array multidimensionale in quanto, ogni suo elemento è un array di 5 elementi con:

- `name`: nome del file originale sulla macchina dell'utente;
- `type`: mime-type del file inviato;
- `tmp_name`: nome temporaneo sul server
- `error`: codice di errore associato all'upload
- `size`: grandezza del file

Per verificare se un file é stato caricato correttamente si usa la funzione `is_uploaded_file(string $nomeFile)`. I file caricati possono anche essere spostati in altre directory: `move_uploaded_file (string $filename , string $destination)`.

# 7 Conservazione dello Stato

## 7.1 Cookie

Un cookie è una stringa (un token) scambiato tra client e server con l'obiettivo di mantenere alcune informazioni relative alle connessione:

- Il client mantiene lo stato delle connessioni precedenti con un cookie e lo invia al server ogni volta che richiede un documento;

HTTP infatti è un protocollo stateless, ovvero non tiene traccia dello stato delle sessioni precedenti per usarle in quelle successive... i cookie (e le sessioni), possono essere usati per rendere HTTP un protocollo stateful... ciò consente per esempio di autenticare gli utenti e tener traccia degli stessi.

- Il meccanismo dei Cookie sfrutta gli header dei messaggi HTTP che client e server si scambiano. La risposta del server ed le richieste successive del server usano intestazioni diverse:
  - **Set-Cookie**: header della risposta, il client può memorizzarlo e rispedirlo alla prossima richiesta.
  - **Cookie**: header della richiesta. Il client decide se spedirlo sulla base del nome del documento, dell'indirizzo IP del server, e dell'età del cookie

```
HTTP

HTTP/2.0 200 OK
Content-Type: text/html
Set-Cookie: yummy_cookie=chocolate
Set-Cookie: tasty_cookie=strawberry

[page content]
```

```
HTTP

GET /sample_page.html HTTP/2.0
Host: www.example.org
Cookie: yummy_cookie=chocolate; tasty_cookie=strawberry
```

Un cookie, essendo una stringa testuale, si compone di una serie di coppie `nome=valore`, separati da un punto e virgola.

### Formato dei cookie

- Expires
- Specifica la durata del cookie (la data di scadenza).
  - Se indicata ed è successiva alla data settata sul sistema dell'utente, il cookie verrà memorizzato sul disco dell'utente
  - Se non è indicata, il cookie associato al documento dura solo fino alla chiusura della sessione del browser (risiede in RAM)
- path
- Specifica la directory sul server a cui è associato il cookie (e.g., path=/corsi/)
  - Il cookie sarà accessibile da tutte le URL residenti nel sottoalbero radicato in /corsi/

## Formato dei cookie

- domain
- Indica il dominio in cui il cookie è visibile
  - Se "domain=unisa.it" e "path=/" il cookie è visibile anche in www.diem.unisa.it, mail.unisa.it, ...
- secure
- indica che il cookie deve essere inviato solo quando browser e server sono connessi tramite HTTPS o un altro protocollo sicuro

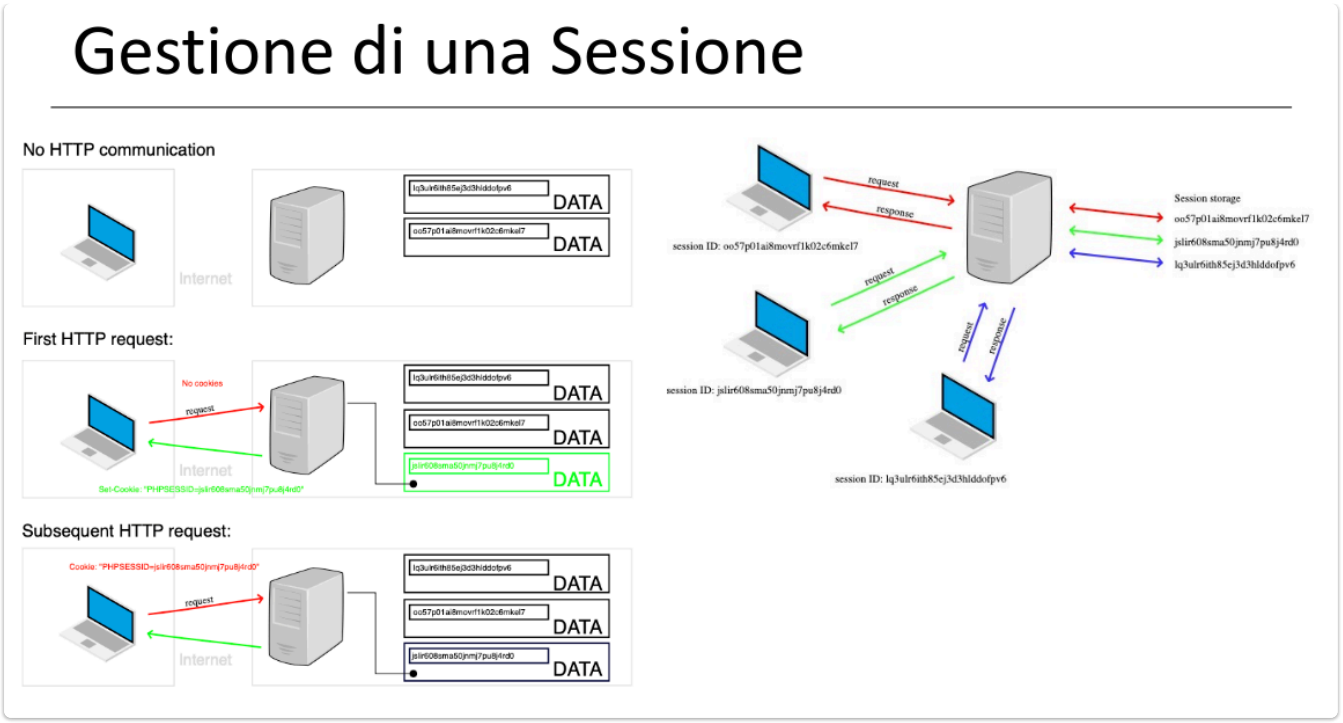
Dato che i cookie vengono settati nella sezione header dei messaggi HTTP, è necessario settarli in PHP prima che il BODY del pacchetto venga inviato, usando la funzione `setcookie(nome, valore, expire, path, domain, secure)`.
Le informazioni sui cookie vengono salvate nell'array `$_COOKIE`:

```
$nomeCliente = $_COOKIE["custname"];
```

## 7.2 Sessioni

Con le sessioni è possibile salvare (via server) un certo numero di variabili e propagarle da una pagina all'altra.
La gestione delle sessioni è server side:

1. Il browser richiede una risorsa al server;
2. Server crea una sessione identificata dal SID, session id. A tale sessione sono associate le variabili di sessione memorizzate sul server HTTP;
3. Nelle richieste successive, il browser fornisce il session id per identificare la sessione e quindi il server può recuperare le variabili di sessione relative




L'ID della sessione si puó propagare in 2 modi diversi:

- Cookie: soluzione da preferire
- Parametro URL: da usare quando i cookie non sono disponibili

PHP è in grado di trasformare i link in modo trasparente. Se l'opzione `session.use_trans_sid` è abilitata, gli URI relativi saranno modificati per contenere automaticamente l'id di sessione.

### 7.2.1 Creazione della sessione

Ogni script che ha bisogno di usare le variabili della sessione (o crearne una nuova), devono usare la funzione `session_start()` all'inizio dello script (dovrebbe essere la prima funzione chiamata all'interno del documento, prima anche del codice HTML). Se la sessione non esiste ancora (quindi la richiesta HTTP non ha associato alcun ID), il server ne crea uno nuovo e lo restituisce al browser.
È possibile creare nuove variabili di sessione e memorizzarle in `$_SESSION`.

## Recuperare variabili di sessione

- Per accedere alle variabili di sessione o creare nuove variabili si utilizza l'array associativo $_SESSION.

```php
<?php
session_start(); // Inizializza la sessione

// Imposta dati di sessione
$_SESSION['username'] = 'mario';
$_SESSION['email'] = 'mario@example.com';
?>
```

```php
<?php
session_start(); // Inizializza la sessione

// Accedi ai dati di sessione
$username = $_SESSION['username'];
$email = $_SESSION['email'];

echo "Username: " . $username . "<br>";
echo "Email: " . $email . "<br>";
?>
```

### 7.2.2 Distruzione della sessione

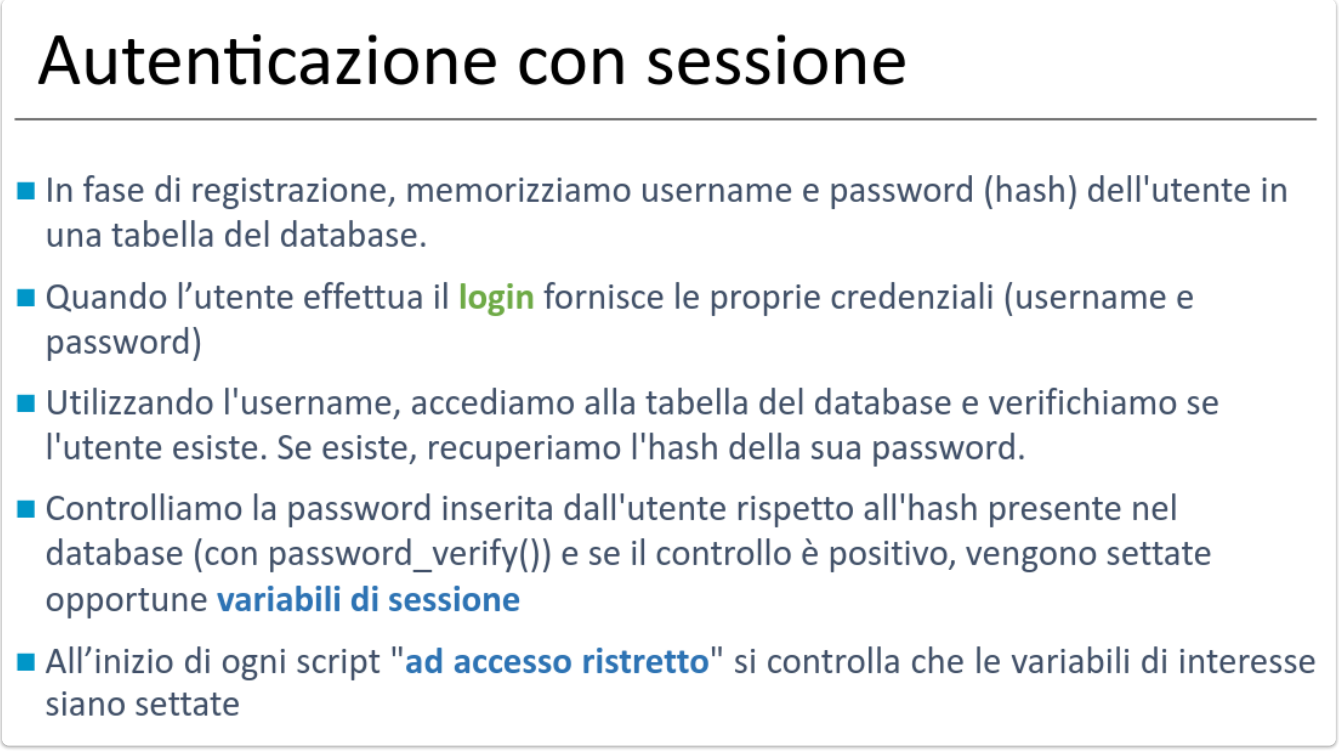
La distruzione della sessione in corso e tutte le informazioni ad essa associate, come le variabili di sessione, avviene mediante il comando `session_destroy()`. Tale funzione tuttavia non cancella il cookie associato dal momento che non ha accesso alla macchina client... la cancellazione deve essere effettuata manualmente impostando un cookie vuoto:

```
<?php
function destroy_session_and_data(){
session_start();
$_SESSION = session_unset();//UNSET DI TUTTE LE VARIABILI DI SESSIONE
if (session_id() != "" || isset($_COOKIE[session_name()]))
    //CANCELLA IL COOKIE CHE MEMORIZZA l'ID della sessione sul client
    setcookie(session_name(), "'', time() - 2592000, '/');
    session_destroy(); //DISTRUGGE LA SESSIONE
}
?>
```

## 7.3 Autenticazione

Puó avvenire in diversi modi, tuttavia si consiglia di usare le sessioni:

- Autenticazione HTTP gestita dal server;
- Cookie
- Sessioni




L'autenticazione prevede il salvataggio di alcune informazioni sensibili dell'utente all'interno di un database.

### 7.3.1 Memorizzazione sicura di password

La memorizzazione delle password avviene sotto forma di hash tramite l'applicazione di un particolare algoritmo. PHP mette a disposizione il metodo `password_hash()` per il calcolo dell'hash della password dell'utente.

```
password_hash ( string $password , int $algo [, array $op4ons ] ) : string
```

- La funzione riceve come parametro la password e il tipo di algoritmo di hashing che si vuole utilizzare.
- È consigliabile specificare la costante PASSWORD_DEFAULT per fare in modo che venga utilizzato sempre l'algoritmo di hashing più sicuro attualmente implementato in PHP

```
echo password_hash("miaPassword", PASSWORD_DEFAULT);
```

- Il risultato di password_hash può essere memorizzato nel database come password dell'utente. Si consiglia l'utilizzo di un campo di testo lungo 255 caratteri per memorizzare la password hash.

La funzione `password_verify (string $password , string $hash ) : bool` consente di verificare se la password ( `$password` ) corrisponde alla relativa versione `$hash` .

```
Dataview (inline field '='): Error:
-- PARSING FAILED -----------------------------------------------

> 1 | =
    | ^

Expected one of the following:

'(', 'null', boolean, date, duration, file link, list ('[1, 2, 3]'), negated
field, number, object ('{ a: 1, b: 2 }'), string, variable
```

```
Dataview (inline field '=='): Error:
-- PARSING FAILED -----------------------------------------------

> 1 | ==
    | ^

Expected one of the following:

'(', 'null', boolean, date, duration, file link, list ('[1, 2, 3]'), negated
field, number, object ('{ a: 1, b: 2 }'), string, variable
```

```
Dataview (inline field '>'): Error:
-- PARSING FAILED -----------------------------------------------

> 1 | >
    | ^

Expected one of the following:

'(', 'null', boolean, date, duration, file link, list ('[1, 2, 3]'), negated
field, number, object ('{ a: 1, b: 2 }'), string, variable
```