

PHP

VALERIO MANDARINO

# TIERS

- ▶ Le applicazioni web sono costruite su più livelli (Tiers)
  - ▶ Client: Web browser
  - ▶ Web: HTTP server
  - ▶ Business: Application server
  - ▶ Data: Database server

# Il client tier

- ▶ Si occupa del rendering della pagina HTML
  - ▶ impagina il testo, le immagini e tutti gli elementi utilizzando eventuali fogli di stile (CSS)
  - ▶ Esegue eventuale codice JavaScript (che gira sul client) che aggiunge della ‘logica’ alla pagina web
- ▶ Desktop web browser: Firefox, Brave, Opera, Safari, Chrome, Edge etc...
- ▶ Mobile browser su smartphone e tablet
- ▶ (Esistono altri client che usano HTTP che non sono servono per navigare in internet bensì per usare le funzionalità del protocollo HTTP)

# Il server stack

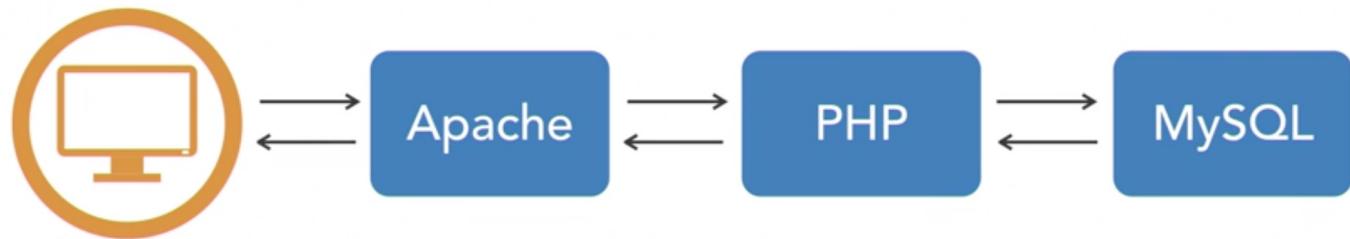
- ▶ Software che collabora:
  - ▶ HTTP Server (Apache, IIS, Cherokee, lighttpd, Caddy, NGINX, etc...)
  - ▶ Application Server (PHP, ASP.NET, Node.js, etc...)
  - ▶ Database Server (Oracle, SQL Server, MySQL, MariaDB, etc...)

# Stack AMP

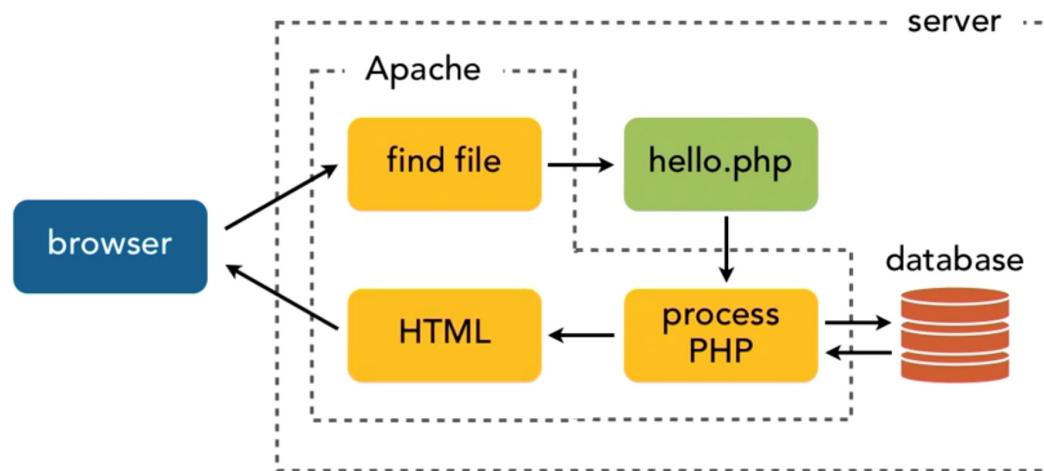
- ▶ APACHE – Server Web: processo in ascolto sulle porte TCP 80 (o 8080) per HTTP e 443 per HTTPS che gestisce le richieste, da parte di un client, di trasferimento di pagine web
- ▶ MySQL / MariaDB – RDBMS: relational database management system composto da un client a riga di comando e un server in ascolto sulla porta 3306
- ▶ Perl / PHP / Python – linguaggi ad alto livello usati per eseguire sul server script che generano pagine dinamiche
- ▶ Può esserci un solo processo in ascolto su una porta. Questo comporta che non si possono usare più web server o database sulle stesse porte

GNU+Linux / MacOS / Windows -> Lamp server / Mamp Server / Wamp server

# Stack AMP



# Architettura three-tier



# Installazione di Apache

- ▶ I sistemi operativi macOS meno recenti hanno già integrati sia Apache che PHP
- ▶ Sugli altri sistemi operativi si devono installare

# Installazione Apache su macOS

- ▶ brew install httpd
- ▶ sudo brew services start httpd (o apache2)
- ▶ sudo brew services stop httpd (o apache2)
- ▶ sudo brew services restart httpd (o apache2)
  
- ▶ La root directory è: /var/www/html

# Apache su MacOS

## ► MacOS

- ▶ apachectl start
- ▶ apachectl stop
- ▶ apachectl restart
- ▶ httpd -v
- ▶ chmod 777 /Libray/Webserver/Documents/ (root directory)
- ▶ code /etc/apache2/httpd.conf

# Configurazione httpd.conf

```
❶ httpd.conf x
etc > apache2 > ❷ httpd.conf
154 #LoadModule lbmethod_bytraffic_module libexec/apache2/mod_lbmethod_bytraffic.so
155 #LoadModule lbmethod_bybusyness_module libexec/apache2/mod_lbmethod_bybusyness.so
156 ##LoadModule heartbeat_module libexec/apache2/mod_lbmethod_heartbeat.so
157 LoadModule unixd_module libexec/apache2/mod_unixd.so
158 #LoadModule heartbeat_module libexec/apache2/mod_heartbeat.so
159 #LoadModule heartmonitor_module libexec/apache2/mod_heartmonitor.so
160 #LoadModule dav_module libexec/apache2/mod_dav.so
161 LoadModule status_module libexec/apache2/mod_status.so
162 LoadModule autoindex_module libexec/apache2/mod_autoindex.so
163 #LoadModule asis_module libexec/apache2/mod_asis.so
164 #LoadModule info_module libexec/apache2/mod_info.so
165 #LoadModule cgi_module libexec/apache2/mod_cgi.so
166 #LoadModule dav_fs_module libexec/apache2/mod_dav_fs.so
167 #LoadModule dav_lock_module libexec/apache2/mod_dav_lock.so
168 #LoadModule vhost_alias_module libexec/apache2/mod_vhost_alias.so
169 LoadModule negotiation_module libexec/apache2/mod_negotiation.so
170 LoadModule dir_module libexec/apache2/mod_dir.so
171 #LoadModule imagemap_module libexec/apache2/mod_imagemap.so
172 #LoadModule actions_module libexec/apache2/mod_actions.so
173 #LoadModule spelling_module libexec/apache2/mod_spelling.so
174 #LoadModule userdir_module libexec/apache2/mod_userdir.so
175 LoadModule alias_module libexec/apache2/mod_alias.so
176 #LoadModule rewrite_module libexec/apache2/mod_rewrite.so
177 #LoadModule php7_module libexec/apache2/libphp7.so
178 #LoadModule perl_module libexec/apache2/mod_perl.so
179 LoadModule hfs_apple_module libexec/apache2/mod_hfs_apple.so
180
181 <IfModule unixd_module>
182 #
183 # If you wish httpd to run as a different user or group, you must run
184 # httpd as root initially and it will switch.
185 #
186 # User/Group: The name (or #number) of the user/group to run httpd as.
187 # It is usually good practice to create a dedicated user and group for
188 # running httpd, as with most system services.
189 #
```

# Mamp

The screenshot shows the homepage of the MAMP & MAMP PRO website. The header features a large blue and green abstract graphic. Below it, the word "Mamp" is displayed in a white sans-serif font. The main navigation bar includes links for "MAMP PRO", "MAMP", "NAMO", "More products ▾", "Downloads", "Support", "DE", and an Apple logo. The central focus is the "MAMP & MAMP PRO" logo, which includes a white elephant icon, the text "MAMP & MAMP PRO", "Your local web development solution", and a "PHP 8 Support" button. Below the logo is a call-to-action section with buttons for "Free Download", "Try Now", and "Buy Now". The background of the main content area is a scenic sunset over a savanna landscape with silhouettes of elephants and a acacia tree.



## MAMP for Windows

MAMP is a free, local server environment that can be installed under [macOS](#) and Windows with just a few clicks. MAMP provides them with all the tools they need to run [WordPress](#) on their desktop PC for testing or development purposes, for example. You can even easily test your projects on mobile devices. It does not matter whether you prefer the web server [Apache](#) or [Nginx](#) in addition to [MySQL](#) as database server, or whether you want to work with [PHP](#), [Python](#), [Perl](#) or [Ruby](#).

[Free MAMP download >](#)

# Installazione su Windows

The screenshot shows the Apache Lounge forum index page. At the top, there's a navigation bar with links for About, Forum Index, Downloads, Search, Register, Log in, and RSS. Below that is a sidebar with social media links for Twitter (@Apachelounge) and a "Keep Server Online" section. The main content area has a header "Apache Lounge Webmasters". A banner at the top of the content area says "Please help fellow Apacherians and share your knowledge. Thanks!". The main content is organized into several sections:

- Forum Index**: A table showing recent posts across various forums. Columns include Post ID, Date, and Author.
- News & Hangout**: News and announcements.
- Apache**: Questions about the Apache Server.
- Third-party Modules**: Discussion about third-party modules like mod\_security, mod\_jk etc.
- Other Software**: Installation and issues with other software like Perl, PHP, MySQL, Tomcat etc.
- Coding & Scripting Corner**: Coding discussions in Perl, PHP, HTML, JavaScript, Ajax etc.
- How-to's & Documentation & Tips**: How-to guides and documentation tips.
- Building & Member Downloads**: Building Apache and Modules, member downloads.
- Webmaster Tools & Utilities**: Webmaster tools like Statistics, Monitoring, Logfile Analyzers, Admin Gui etc.
- Hardware & Networking**: Discussion about hardware and network related topics.
- Recent Discussions**: A table showing the latest forum posts.

At the bottom of the page, there's a footer with links for Reverse proxy configuration, LOG4J, Critical vulnerability in Apache Log4j library, Season Greetings, OpenSUSE 13.1 and 1.1.1m release 14-dec-2021, Proxy configuration for OBIEE, running PHP 8.1 on Apache 2.4, Okay Java Quickstart, and Installing Apache in Ubuntu.

# Configurazione httpd.conf

```
httpd.conf X
C:\apache>conf > httpd.conf
21 # NOTE: Where filenames are specified, you must use forward slashes
22 # instead of backslashes (e.g., "c:/apache" instead of "c:\apache").
23 # If a drive letter is omitted, the drive on which httpd.exe is located
24 # will be used by default. It is recommended that you always supply
25 # an explicit drive letter in absolute paths to avoid confusion.
26 #
27 #
28 # ServerRoot: The top of the directory tree under which the server's
29 # configuration, error, and log files are kept.
30 #
31 # Do not add a slash at the end of the directory path. If you point
32 # ServerRoot at a non-local disk, be sure to specify a local disk on the
33 # Mutex directive, if file-based mutexes are used. If you wish to share the
34 # same ServerRoot for multiple httpd daemons, you will need to change at
35 # least PidFile.
36 #
37 Define SRVROOT "c:/Apache24"
38
39 ServerRoot "${SRVROOT}"
40
41 #
42 # Mutex: Allows you to set the mutex mechanism and mutex file directory
43 # for individual mutexes, or change the global defaults I
44 #
45 # Uncomment and change the directory if mutexes are file-based and the default
46 # mutex file directory is not on a local disk or is not appropriate for some
47 # other reason.
48 #
49 # Mutex default:logs
50
51 #
```

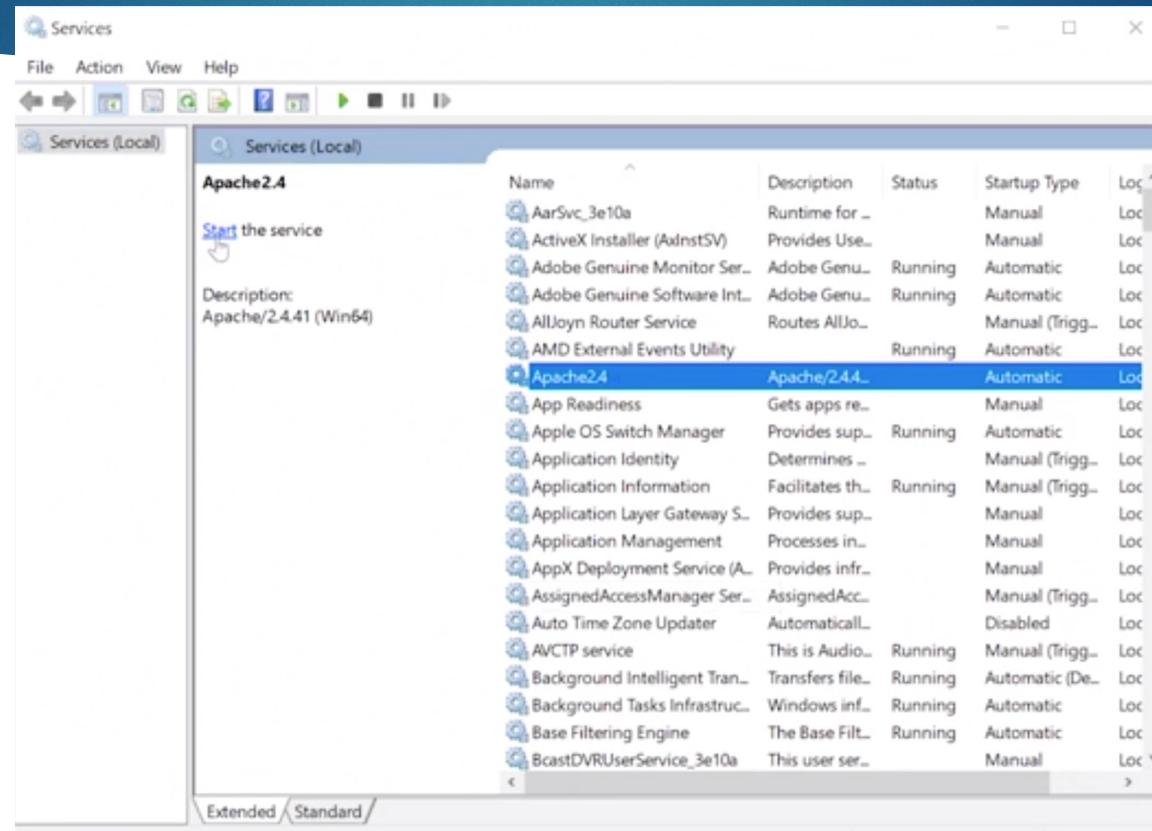
# Configurazione httpd.conf

```
httpd.conf •  
C:\> apache > conf > httpd.conf  
205 # <VirtualHost> definition. These values also provide defaults for  
206 # any <VirtualHost> containers you may define later in the file.  
207 #  
208 # All of these directives may appear inside <VirtualHost> containers,  
209 # in which case these default settings will be overridden for the  
210 # virtual host being defined.  
211 #  
212 #  
213 #  
214 # ServerAdmin: Your address, where problems with the server should be  
215 # e-mailed. This address appears on some server-generated pages, such  
216 # as error documents. e.g. admin@your-domain.com  
217 #  
218 ServerAdmin admin@example.com  
219 #  
220 #  
221 # ServerName gives the name and port that the server uses to identify itself.  
222 # This can often be determined automatically, but we recommend you specify  
223 # it explicitly to prevent problems during startup.  
224 #  
225 # If your host doesn't have a registered DNS name, enter its IP address here.  
226 #  
227 ServerName localhost|  
228 #  
229 #  
230 # Deny access to the entirety of your server's filesystem. You must  
231 # explicitly permit access to web content directories in other  
232 # <Directory> blocks below.  
233 #  
234 <Directory />  
235 AllowOverride none
```

# Apache su Windows

- ▶ La root directory è c:\Apache24\httpdocs
- ▶ Per fare partire Apache: cd c:\Apache24\bin
- ▶ .\httpd.exe
- ▶ Anziché farlo partire manualmente è più conveniente creare un servizio:
- ▶ cd:\Apache24\bin\
- ▶ .\httpd.exe -k install
- ▶ net start apache2.4
- ▶ net stop apache2.4

# Apache come service



# Configurazione httpd.conf

```
❶ httpd.conf •  
C: > apache > conf > ❷ httpd.conf  
521 # Configure mod_proxy_html to understand HTML4/XHTML1  
522 <IfModule proxy_html_module>  
523   Include conf/extra/proxy-html.conf  
524 </IfModule>  
525  
526 # Secure (SSL/TLS) connections  
527 #Include conf/extra/httpd-ssl.conf  
528 #  
529 # Note: The following must must be present to support  
530 #       starting without SSL on platforms with no /dev/random equivalent  
531 #       but a statically compiled-in mod_ssl.  
532 #  
533 <IfModule ssl_module>  
534   SSLRandomSeed startup builtin  
535   SSLRandomSeed connect builtin  
536 </IfModule>  
537  
538 LoadModule php7_module "C:/php/php7apache2_4.dll"  
539 AddHandler application/x-httpd-php .php  
540 PHPIniDir "C:/php"  
541
```

# Wamp

The screenshot shows the official WampServer website. At the top, there's a dark header bar with the WampServer logo, the text "WampServer Apache, PHP, MySQL on Windows", and navigation links for "START", "DOWNLOAD", and "FORUM". There are also "FRANÇAIS" and "РУССКИЙ" language links. Below the header, the main content area features a large cartoon illustration of a scientist with glasses, holding test tubes with blue liquid, and a flask with purple liquid, with bubbles floating around. A red stamp-like graphic says "CONTRIBUTION ALTER WAY". To the left of the scientist, the text reads "WAMP SERVER, a Windows web development environment.". Below this, a paragraph describes WampServer as a Windows web development environment that includes Apache2, PHP, and a MySQL database, with the ability to manage databases via PhpMyAdmin. A "START USING WAMP SERVER" button is present. At the bottom, there's a yellow banner with two small beakers containing green and blue liquid, and the text "START WITH WAMP SERVER". A small note below the banner states: "WampServer installs automatically all you need to start developing web applications and is very intuitive to use. You will be able to tune your server without even touching the setting files." Social sharing icons for Facebook, Twitter, and LinkedIn are at the bottom left.

# Installazione Apache su GNU+Linux (Ubuntu)

- ▶ sudo apt-get install apache2
- ▶ sudo service apache2 start
- ▶ La root directory è: /var/www/html

# Perché PHP?

- ▶ PHP gira su varie piattaforme (Windows, Linux, Unix, Mac OS X, etc.)
- ▶ PHP è compatibile con quasi tutti i server usati oggi (Apache, IIS, etc.)
- ▶ PHP supporta un ampio numero di database
- ▶ PHP è efficiente
- ▶ La versione 7 include strumenti molto potenti come:
  - ▶ prestazioni notevolmente migliorate rispetto a PHP 5.6.
  - ▶ nuovi operatori
  - ▶ nuovi iteratori
  - ▶ error handling potenziato
  - ▶ codifica delle password
  - ▶ strict types (es. operatore “spaceship” <=>)

# Framework

- ▶ Sono stati implementati numerosi framework basati su PHP.

Alcuni sono:

Laravel - CodeIgniter - Symfony - Laminas Project - Phalcon - CakePHP - Yii –  
FuelPHP - Slim - PHPixie - Fat-Free Framework - WordPress - Drupal - Joomla



BASI

CORSO PHP - 2025

# Linguaggio di scripting

- ▶ Differenza tra uno script e un programma:
  - ▶ Script:
    - ▶ viene eseguito in risposta ad un evento
    - ▶ di solito esegue le istruzioni in modo sequenziale
    - ▶ poca interazione con l'utente
  - ▶ Programma:
    - ▶ viene eseguito anche non in corrispondenza di un evento
    - ▶ esegue il codice in maniera non necessariamente sequenziale
    - ▶ forte interazione con l'utente

# Sintassi

- ▶ Il codice PHP inizia con `<?php` e termina con `?>`.
- ▶ L'estensione predefinita dei file è `.php`.
- ▶ I file PHP possono contenere sia HTML che codice PHP.
- ▶ Ogni istruzione termina con `;`.
- ▶ Le parole chiave (if, else, echo, ecc.) non sono case-sensitive.
- ▶ Le variabili sono case-sensitive: `$color` ≠ `$COLOR`.

```
<!DOCTYPE html>
<html>
<body>
<h1>La mia prima pagina</h1>
<?php
    echo "Hello World!";
?>
</body>
</html>
```

# Commenti

- ▶ Tipi di commenti:

```
// Commento su una riga  
# Commento su una riga  
/* Commento  
   su più righe */
```

# Variabili

- ▶ Le variabili si dichiarano con il simbolo \$ seguito dal nome e vengono create al primo utilizzo.
- ▶ I nomi:
  - devono iniziare con una lettera o underscore.
  - non possono iniziare con un numero.
  - possono contenere solo lettere, numeri e underscore.
  - sono **case-sensitive**.
- ▶ PHP è a **tipizzazione dinamica**: il tipo è assegnato in base al valore, ma da PHP 7 è possibile specificare i **tipi dei parametri e del valore di ritorno** delle funzioni, abilitando la **modalità strict**. Un tipo errato genera un errore.
- ▶ Tipi di dato principali:  
`string, integer, float, boolean, array, object, NULL, resource.`
- ▶ `object` è un'istanza di una classe definita dall'utente.
- ▶ `resource` è un puntatore a una risorsa esterna, (file, connessione a un database, socket).

vedere `esempio01.php`

# Scope delle variabili

- ▶ Global Scope: dichiarata fuori da ogni funzione, accessibile solo all'esterno.
- ▶ Local Scope: dichiarata dentro una funzione, accessibile solo al suo interno.

keyword `global`: permette di usare variabili globali dentro le funzioni.

`$GLOBALS` array: tutte le variabili globali sono accessibili tramite `$GLOBALS['nome']`

keyword `static`: mantiene il valore di una variabile locale tra più chiamate della stessa funzione.

vedere `esempio02.php`

# I comandi echo e print

- ▶ Entrambi servono per **stampare dati a schermo**.
- ▶ Differenze:
  - echo **non** restituisce valore, print restituisce **1**.
  - echo accetta **più argomenti**, print solo **uno**.
  - echo è **più veloce**.

vedere `esempio03.php`

# Tipi di dati: numeri

- ▶ **Integer:** numero intero senza parte decimale.
- ▶ Formati ammessi: decimale (10), esadecimale (0x), ottale (0), binario (0b).
- ▶ Verifica con `is_int($x)`.
- ▶ Casting `(int)$x;`
- ▶ Costanti: `PHP_INT_MAX`, `PHP_INT_MIN`, `PHP_INT_SIZE`.
- ▶ Range:

32 bit: da -2,147,483,648 a 2,147,483,647;

64 bit: da -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807

Se il valore supera questi limiti → viene convertito automaticamente in float.

(Anche un'operazione con un float restituisce un float.)

# Tipi di dati: numeri

- ▶ **Float:** numero con parte decimale o in forma esponenziale.
- ▶ Verifica con `is_float($x)`.
- ▶ Casting `(float)$x;`
- ▶ Costanti: `PHP_FLOAT_MAX`, `PHP_FLOAT_MIN`, `PHP_FLOAT_DIG`, `PHP_FLOAT_EPSILON`.

# Tipi di dati: numeri

- ▶ **Infinity**: valore numerico oltre il limite massimo.
- ▶ Verifica con `is_finite($x)`; e `is_infinite($x)`;
  
- ▶ **NaN** (Not a Number): risultato di operazioni matematiche non valide.
- ▶ Verifica con `is_nan($x)`;
  
- ▶ **Number strings**: stringhe che contengono un valore numerico.
- ▶ Verifica con `is_numeric($x)`;
- ▶ Casting (`string`) `$x`;

vedere `esempio04.php`

# Tipi di dati: stringhe

- ▶ **string:** le stringhe si definiscono tra doppi (" ") o singoli (' ') apici.
  - Doppi apici → interpretano variabili e caratteri speciali
  - Singoli apici → stampano il testo letterale
- ▶ Verifica: `is_string($x)`
- ▶ Casting: `(string)$x` O `strval($x)` Array e oggetti non possono essere convertiti direttamente in stringa, se non implementano il metodo `__toString()`.
- ▶ Funzioni principali:

<code>strlen()</code> → restituisce la lunghezza della stringa	<code>strtolower()</code> → converte in minuscolo
<code>str_word_count()</code> → conta le parole	<code>str_replace()</code> → sostituisce parte
<code>strpos()</code> → trova la posizione di un testo	<code>strrev()</code> → inverte la stringa
<code>strtoupper()</code> → converte in maiuscolo	<code>trim()</code> → rimuove spazi iniziali e finali
<code>explode()</code> → divide una stringa in un array usando un separatore	<code>.</code> → concatena due o più stringhe
<code>substr()</code> → estrae una porzione di stringa.	

# Stringhe: caratteri di escape

- ▶ \ serve a inserire simboli speciali all'interno delle stringhe

Codice	Risultato	Descrizione
\'	'	Apice singolo
\\"	"	Virgolette doppie
\\$	\$	Variabili PHP letterali
\n	Nuova riga	
\r	Carriage return	
\t	Tabulazione	
\f	Form feed	
\ooo	Valore ottale	
\xhh	Valore esadecimale	

vedere `esempio05.php`

# Tipi di dati: Boolean

- ▶ Valori considerati false: false, 0 ,0.0, "", "0", NULL, [], Oggetto senza proprietà
- ▶ Tutti gli altri valori sono interpretati come true.
  
- ▶ Verifica con: `is_bool($x)`
- ▶ Casting (`bool`) `$x`.

# Casting

- ▶ Tipi di casting disponibili:
- ▶ (string) → converte qualsiasi valore in testo.
- ▶ (int) → tronca decimali e converte stringhe numeriche o booleani in interi.
- ▶ (float) → converte numeri interi e stringhe numeriche in decimali.
- ▶ (bool) → restituisce false per 0, stringhe vuote, NULL o false; true in tutti gli altri casi.
- ▶ (array) → trasforma il valore in un array con un solo elemento; NULL diventa un array vuoto; gli oggetti diventano array associativi.
- ▶ (object) → trasforma il valore in un oggetto anonimo della classe stdClass inserendolo in una proprietà chiamata scalar; array indicizzati diventano oggetti con indici come proprietà; array associativi mantengono le chiavi come nomi di proprietà.
- ▶ (unset) → converte in NULL

# Type Juggling

- ▶ string vs null: converte null in ""
  - ▶ boolean vs altro: converte altro in boolean
  - ▶ numero vs altro: converte altro in numero
  
  - ▶ 

```
echo 0 == FALSE ? 'uguale' : 'diverso';
```

Restituisce uguale
  - ▶ 

```
echo 0 === FALSE ? 'uguale' : 'diverso';
```

Restituisce diverso
  
  - ▶ Usando === e !== il Type Juggling non avviene
- ▶ Fare attenzione alla funzione empty(). Restituirà true su:
- ""
  - 0
  - "0"
  - null
  - false
  - array() //vuoto

vedere `esempio06.php`

# Funzioni matematiche

- ▶ PHP fornisce un insieme di funzioni per eseguire operazioni numeriche. Alcune di queste sono:
- ▶ `pi()` → restituisce il valore di  $\pi$ .
- ▶ `min()` / `max()` → determinano il valore minimo o massimo di un insieme di numeri.
- ▶ `abs()` → calcola il valore assoluto.
- ▶ `sqrt()` → – restituisce la radice quadrata.
- ▶ `round()` → arrotonda un numero decimale all'intero più vicino.
- ▶ `rand()` → genera un numero casuale, con possibilità di definire intervallo minimo e massimo.

vedere `esempio07.php`

# Costanti

- ▶ Per dichiarare una costante si usa la funzione `define (nome, valore)`.
- ▶ È possibile creare costanti anche con `const`.
- ▶ Il nome deve iniziare con una lettera o un underscore. Non usa il simbolo `$`.
- ▶ Le costanti sono automaticamente globali in tutto lo script.
- ▶ Differenze tra `const` e `define ()`:
  - `const` non può essere usata all'interno di blocchi (funzioni, `if`, ecc.).
  - `define ()` può essere usata ovunque, anche dentro un blocco.
- ▶ Array costanti: da PHP 7 è possibile definire array come costanti con `define ()`.
- ▶ Le costanti sono sempre **globali** e accessibili da qualsiasi punto del programma.

vedere `esempio08.php`

# Magic Constants

- ▶ Le magic constants sono valori predefiniti che PHP sostituisce automaticamente durante l'esecuzione.
- ▶ Il loro valore dipende dal contesto in cui vengono utilizzate.
- ▶ `__CLASS__` → restituisce il nome della classe in cui è usata.
- ▶ `__DIR__` → indica la directory del file corrente.
- ▶ `__FILE__` → mostra il percorso completo del file.
- ▶ `__FUNCTION__` → restituisce il nome della funzione corrente.
- ▶ `__LINE__` → indica il numero di riga nel file.
- ▶ `__METHOD__` → mostra nome della classe e del metodo in cui è usata.
- ▶ `__NAMESPACE__` → restituisce il nome del namespace corrente.
- ▶ `__TRAIT__` → mostra il nome del trait corrente. (alternativa all'ereditarietà)
- ▶ `ClassName::class` → restituisce il nome completo (namespace + classe) di una classe.

vedere `esempio09.php`

# Operatori

- ▶ PHP li suddivide nei seguenti gruppi:
- ▶ **aritmetici** → eseguono operazioni matematiche (somma, sottrazione, moltiplicazione, divisione, modulo, potenza).
- ▶ **assegnazione** → assegnano o modificano valori nelle variabili (=, +=, -=, \*=, /=, %=).
- ▶ **confronto** → confrontano due valori (==, ===, !=, !==, <, >, <=, >=, <=>).
- ▶ **incremento/decremento** → aumentano o diminuiscono di 1 il valore di una variabile (++\$x, \$x++, --\$x, \$x--).
- ▶ **logici** → combinano condizioni logiche (and, or, xor, &&, ||, !).
- ▶ **per stringhe** → concatenano o estendono stringhe (. e .=).
- ▶ **per array** → confrontano o uniscono array (+, ==, ===, !=, <>).
- ▶ **assegnazione condizionale** → restituiscono valori in base a condizioni (?: e ??).

vedere `esempio10.php`

# Condizioni

- ▶ `if` → esegue un blocco di codice se la condizione è vera.
- ▶ `if...else` → esegue un blocco se la condizione è vera e un altro se è falsa.
- ▶ `if...elseif...else` → gestisce più condizioni alternative.
- ▶ `switch` → sceglie tra diversi blocchi di codice in base al valore di una variabile.

vedere `esempio11.php`

# Loop

- ▶ Tipi di cicli in PHP:
- ▶ while → esegue il blocco finché la condizione rimane vera.
- ▶ do...while → esegue il blocco almeno una volta, poi continua finché la condizione è vera.
- ▶ for → esegue il blocco per un numero specificato di iterazioni.
- ▶ foreach → scorre ogni elemento di un array ed esegue il blocco per ciascuno di essi.

vedere `esempio12.php`

# ESERCIZI

LOOP, CONDIZIONI, OPERATORI, STAMPA A VIDEO, RAND(), COSTANTI

# Indovina il numero

- ▶ Scrivere uno script PHP che simuli un gioco in cui **il computer tenta di indovinare un numero segreto scelto casualmente da sé stesso.**
1. All'inizio del programma:
    1. Genera un numero segreto casuale compreso tra due costanti.
    2. Imposta il numero massimo di tentativi con una costante.
  2. Durante ogni tentativo:
    1. Finché non indovina e ci sono tentativi a disposizione, genera una ipotesi casuale.
    2. Se l'ipotesi è corretta, stampa un messaggio di vittoria e termina il ciclo.
    3. Altrimenti da un indizio sulla grandezza del numero segreto rispetto all'ipotesi.
  3. Al termine del gioco:
    1. Mostra un messaggio e un link per ricaricare la pagina e giocare di nuovo.

vedere `esempio12_1.php`

# Simulazione di lancio di una moneta con istogramma

- ▶ Scrivi un programma PHP che simuli il lancio ripetuto di una moneta.
- 1. Definire le costanti `NUMERO_LANCI` per stabilire quanti lanci eseguire e `STAMPA_PASSI` per decidere se mostrare ogni singolo lancio (1 = sì, 0 = no).
- 2. Usare un ciclo `for` per eseguire i lanci e generare casualmente un risultato (0 o 1).
- 3. Contare quante volte esce testa e quante croce e stampare ogni lancio se `STAMPA_PASSI == 1`
- 4. Al termine, stampare un riepilogo visivo con una riga di asterischi per ciascun tipo di risultato.

vedere `esempio12_2.php`

# FUNZIONI

CORSO PHP - 2025

# Funzioni

- ▶ PHP include oltre 1000 funzioni predefinite per svolgere compiti specifici.
- ▶ Oltre a queste, è possibile creare funzioni personalizzate.
- ▶ I nomi delle funzioni sono case insensitive.
- ▶ Si possono impostare valori predefiniti per gli argomenti.
- ▶ Usando &, le modifiche ai parametri si riflettono sulla variabile originale.
- ▶ L'operatore ... consente di accettare un numero indefinito di parametri. (Deve essere l'ultimo nella lista.)

Tipizzazione (PHP 7+):

- ▶ È possibile specificare il tipo degli argomenti e del valore di ritorno.
- ▶ Con `declare(strict_types=1)` si attiva la modalità rigorosa, che genera errore in caso di tipi non corrispondenti.

vedere `esempio13.php`

# ARRAY

CORSO PHP - 2025

# Array

- ▶ Tipi di Array:
  - Array **indicizzati** → con indici numerici.
  - Array **associativi** → con chiavi testuali.
  - Array **multidimensionali** → che contengono uno o più array al loro interno.
- ▶ Gli elementi di un Array:
  - Possono essere di qualsiasi tipo: stringhe, numeri, oggetti, funzioni o altri array.
  - È possibile mescolare tipi di dati diversi nello stesso array.
- ▶ È possibile mescolare chiavi numeriche e testuali nello stesso array.
- ▶ PHP offre molte funzioni integrate per la manipolazione degli array.

# Creazione e iterazione

- ▶ PHP offre un'ampia gamma di funzioni integrate per creare, modificare, ordinare e analizzare gli array.
- ▶ Creazione e struttura
  - ▶ `array(elementi)` → crea un array
  - ▶ `[elementi]` → sintassi corta
  - ▶ `compact()` → crea un array contenente variabili e i loro valori.
  - ▶ `range(intervallo)` → genera un array con una sequenza di elementi.
- ▶ `foreach` permette di scorrere gli elementi di un array.

Per modificare un array all'interno di un `foreach`, si usa `&` per passare l'elemento per riferimento.

Dopo il ciclo è necessario usare `unset()` per evitare effetti collaterali.
- ▶ Negli array associativi si possono aggiungere più elementi con l'operatore `+=`.

vedere `esempio14.php`

# ESERCIZI

ARRAY E DIZIONARI, INTEGRAZIONE DI HTML, FUNZIONI

# Simulazione di lancio di dadi con istogramma

- ▶ Scrivere uno script PHP che simuli il lancio di un certo numero di dadi a  $n$  facce, per un numero prefissato di volte.
- ▶ Dopo ogni lancio, o al termine della simulazione, il programma deve mostrare un istogramma testuale che rappresenti la frequenza di ciascun risultato ottenuto.

vedere `esercizio14_1.php`

# Generatore di poesia casuale

- ▶ Scrivere uno script PHP che generi automaticamente brevi poesie composte da versi creati casualmente.
- ▶ Definire una costante NUMERO\_VERSI che indica quanti versi generare.
- ▶ Definire quattro array \$soggetti, \$verbi, \$oggetti, \$chiusure contenenti parole o frasi appartenenti alle diverse categorie.
- ▶ Estrarre casualmente un elemento da ciascuno dei primi tre array usando una funzione.
- ▶ Costruire una frase unendo gli elementi.
- ▶ Con una probabilità del 50%, aggiungere una chiusura tratta dall'array \$chiusure.
- ▶ Restituire la stringa finale con un punto.
- ▶ Stampare un link che permetta di generare un nuovo poema.

vedere esercizio14\_2.php

# Generatore di personaggi di D&D

- ▶ Scrivere uno script PHP che generi casualmente un personaggio di Dungeons & Dragons salvi le seguenti caratteristiche in un dizionario:
- ▶ Nome generato casualmente combinando sillabe predefinite salvate in array.
- ▶ Razza, Classe e Origine scelte in modo casuale da appositi array.
- ▶ Caratteristiche (sotto-dizionario che include): Forza, Destrezza, Costituzione, Intelligenza, Saggezza, Carisma, ottenute tirando 3 dadi a 6 facce per ciascuna.
- ▶ Visualizzare tutte le informazioni del personaggio in una tabella HTML, mostrando per ogni caratteristica il relativo punteggio.
- ▶ Alla fine, aggiungere un link che permetta di ricaricare la pagina per generare un nuovo personaggio.

vedere esercizio14\_3.php

# OPERAZIONI SU ARRAY

CORSO PHP - 2025

# Ordinamento di array

- ▶ PHP fornisce diverse funzioni per ordinare gli array, sia indicizzati che associativi.

`sort()` → ordina un array in ordine crescente.

`rsort()` → ordina un array in ordine decrescente.

`asort()` → ordina un array associativo in ordine crescente in base ai valori.

`ksort()` → ordina un array associativo in ordine crescente in base alle chiavi.

`arsort()` → ordina un array associativo in ordine decrescente in base ai valori.

`krsort()` → ordina un array associativo in ordine decrescente in base alle chiavi.

`natsort()` / `natcasesort()` → ordinano con “ordine naturale” (alfanumerico).

- ▶ Note:

`sort()` e `rsort()` riordinano gli elementi modificando l’array originale.

`asort()`, `ksort()`, `arsort()`, `krsort()` mantengono la relazione tra chiave e valore.

vedere `esempio15_1.php`

# Altre funzioni per array

## ► Modifica di chiavi e valori

`array_change_key_case()` → converte tutte le chiavi in maiuscolo o minuscolo.

`array_combine()` → unisce un array di chiavi e uno di valori.

`array_fill()` / `array_fill_keys()` → riempie un array con valori specificati.

`array_replace()` / `array_replace_recursive()` → sostituisce i valori di un array con quelli di altri.

## ► Suddivisione e fusione

`array_chunk()` → divide un array in blocchi.

`array_merge()` / `array_merge_recursive()` → unisce più array.

`array_splice()` → rimuove o sostituisce elementi.

`array_slice()` → restituisce una porzione di array.

vedere `esempio15_2.php`

# Altre funzioni per array

## ▶ Filtraggio e trasformazione

`array_filter()` → filtra gli elementi in base a una funzione.

`array_map()` → applica una funzione a ogni elemento.

`array_walk()` / `array_walk_recursive()` → esegue una funzione su ogni elemento.

`array_reduce()` → riduce l'array a un singolo valore.

## ▶ Ricerca e verifica

`in_array()` → verifica se un valore esiste.

`array_key_exists()` → controlla se una chiave esiste.

`array_search()` → restituisce la chiave di un valore.

`array_keys()` / `array_values()` → restituiscono rispettivamente tutte le chiavi o i valori.

vedere `esempio15_3.php`

# Altre funzioni per array

## ► Confronto e differenze

`array_diff()` / `array_diff_assoc()` / `array_diff_key()` → confrontano array restituendo le differenze.

`array_intersect()` / `array_intersect_assoc()` / `array_intersect_key()` → restituiscono gli elementi comuni.

Versioni *u* come `array_udiff()` o `array_uintersect()` permettono confronti con funzioni definite dall'utente.

## ► Manipolazione di elementi

`array_push()` / `array_pop()` → aggiungono o rimuovono elementi alla fine.

`array_unshift()` / `array_shift()` → aggiungono o rimuovono elementi all'inizio.

`array_unique()` → rimuove i duplicati.

`shuffle()` → mescola casualmente gli elementi.

vedere esempio15\_4.php

# Altre funzioni per array

## ► Conteggio e calcoli

`count()` / `sizeof()` → restituiscono il numero di elementi.

`array_sum()` → somma i valori.

`array_product()` → calcola il prodotto dei valori.

`array_count_values()` → conta quante volte ogni valore compare.

## ► Altre utilità

`array_reverse()` → inverte l'ordine degli elementi.

`array_flip()` → scambia chiavi e valori.

`array_rand()` → restituisce una o più chiavi casuali.

`list()` → assegna variabili come se fossero elementi di un array.

`extract()` → importa variabili da un array nel contesto corrente.

vedere `esempio15_5.php`

# SUPERGLOBALS

CORSO PHP - 2025

# Superglobals

- ▶ In PHP esistono alcune variabili predefinite dette superglobals, sempre accessibili da qualsiasi punto del codice, indipendentemente dallo scope.

`$GLOBALS` → contiene tutte le variabili globali in un array associativo.

`$_SERVER` → contiene informazioni sul server e sull'ambiente di esecuzione.

`$_REQUEST` → raccoglie dati da `$_GET`, `$_POST` e `$_COOKIE`.

`$_POST` → dati inviati tramite metodo POST.

`$_GET` → dati inviati tramite metodo GET.

`$_FILES` → informazioni sui file caricati tramite form.

`$_ENV` → variabili d'ambiente.

`$_COOKIE` → dati dei cookie.

`$_SESSION` → variabili di sessione.

# \$\_SERVER

- ▶ `$_SERVER` è una variabile superglobale che contiene informazioni sull'ambiente del server, sulle intestazioni HTTP, sui percorsi e sullo script in esecuzione.

`$_SERVER['PHP_SELF']` → nome del file dello script corrente

`$_SERVER['GATEWAY_INTERFACE']` → versione del protocollo CGI usato dal server

`$_SERVER['SERVER_ADDR']` → indirizzo IP del server

`$_SERVER['SERVER_NAME']` → nome del server

`$_SERVER['SERVER_SOFTWARE']` → software e versione del server

`$_SERVER['SERVER_PROTOCOL']` → protocollo e versione

`$_SERVER['HTTP_USER_AGENT']` → browser e sistema operativo

`$_SERVER['HTTP_ACCEPT_LANGUAGE']` → le lingue preferite impostate nel browser dell'utente, in ordine di priorità

`$_SERVER['REQUEST_METHOD']` → metodo HTTP utilizzato

`$_SERVER['REQUEST_TIME']` → timestamp di inizio della richiesta

`$_SERVER['QUERY_STRING']` → stringa di query dell'URL

vedere [esempio16.php](#)

# \$\_SERVER

`$_SERVER['HTTP_ACCEPT']` → header Accept della richiesta

`$_SERVER['HTTP_ACCEPT_CHARSET']` → charset

`$_SERVER['HTTP_HOST']` → host della richiesta HTTP

`$_SERVER['HTTP_REFERER']` → URL di provenienza

`$_SERVER['HTTPS']` → indica se usa HTTPS

`$_SERVER['REMOTE_ADDR']` → IP del client

`$_SERVER['REMOTE_HOST']` → nome host del client

`$_SERVER['REMOTE_PORT']` → porta usata dal client

`$_SERVER['SCRIPT_FILENAME']` → percorso assoluto dello script in esecuzione

`$_SERVER['SERVER_ADMIN']` → email dell'amministratore

`$_SERVER['SERVER_PORT']` → porta del server

`$_SERVER['SERVER_SIGNATURE']` → firma del server

`$_SERVER['PATH_TRANSLATED']` → percorso fisico dello script sul server

`$_SERVER['SCRIPT_NAME']` → percorso relativo dello script corrente

`$_SERVER['SCRIPT_URI']` → URI completo della pagina attuale

# HTTP\_USER\_AGENT

- ▶ HTTP\_USER\_AGENT restituisce la stringa identificativa del browser e del sistema operativo utilizzata dal client nella richiesta HTTP.
- ▶ Esempi:
  1. **Firefox:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:144.0) Gecko/20100101 Firefox/144.0
  2. **Safari:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.3 Safari/605.1.15
  3. **Brave:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
  4. **Opera:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36 OPR/122.0.0.0

# HTTP\_USER\_AGENT

- ▶ Analizziamo lo user agent di Safari:
  - ▶ Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.3 Safari/605.1.15
- ▶ Mozilla/5.0 → ereditato dagli anni '90, quando quasi tutti i siti controllavano se il browser fosse "Mozilla" (cioè Netscape).
  - ▶ Se si omettesse, molti siti vecchi servirebbero versioni errate o non compatibili.
- ▶ AppleWebKit/605.1.15 → indica il motore di rendering reale.
- ▶ KHTML è il motore originale da cui è nato WebKit
- ▶ like Gecko serve a dire *mi comporto come Gecko*, così i siti scritti per Firefox funzionano anche su Safari.
- ▶ Safari/605.1.15 → identifica esplicitamente Safari.

# ESERCIZI

SUPERGLOBALS, FUNZIONI, CONDIZIONI

# Rilevare browser, SO e lingua dell'utente

- ▶ Scrivere uno script che analizzi le informazioni inviate dal client e visualizzi a schermo:
  1. Il nome del browser utilizzato.
  2. Il sistema operativo rilevato.
  3. La lingua principale impostata nel browser.

vedere esercizio16\_1.php

# Personalizzazione dinamica della pagina

- ▶ Scrivere uno script PHP che personalizzi l'aspetto e il contenuto della pagina in base alle informazioni rilevate dal client.
  1. impostare un colore di sfondo diverso per ciascun browser o engine.
  2. mostrare un'immagine diversa a seconda del sistema operativo rilevato.
  3. visualizzare un messaggio di benvenuto tradotto nella lingua del client (o in inglese se questa non supportata).
- ▶ Suggerimenti:
  1. Utilizzare le funzioni `getBrowser()`, `getOS()`, `getLanguage()`
  2. Per il messaggio utilizzare il seguente array: `$messaggi = ['it' => 'Benvenuto nel nostro sito!', 'en' => 'Welcome to our website!', 'es' => ';Bienvenido a nuestro sitio web!', 'fr' => 'Bienvenue sur notre site web !', 'de' => 'Willkommen auf unserer Website!',];`

vedere esercizio16 2.php

# Personalizzazione dinamica della pagina

- ▶ Versione proposta da uno studente:
  1. Fare l'esercizio precedente utilizzando un css da modificare o selezionare nello script php.
- ▶ Nota: l'ultima versione è un esempio che mostra che **non è possibile inserire codice PHP all'interno di un file CSS statico**, poiché il PHP non viene interpretato nei file con estensione .css.

vedere esercizio16\_3.php - esercizio16\_5.php