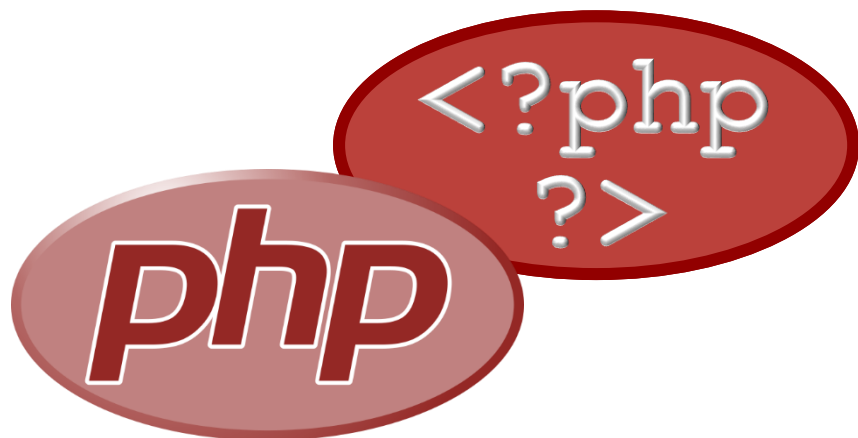


# PHP sintassi





# Argomenti

---

- Sintassi PHP
  - Elementi di base
  - GET e POST
  - Gestione di Cookie e Sessioni
  - Gestione del DB MySQL





# Siti di riferimento

---

- W3Schools:  
<https://www.w3schools.com/php/>
- Sito web di PHP:  
<https://secure.php.net/>
- NB: conviene comunque impraticchirsi con il W3School perché è l'unico disponibile durante l'esame!



# Variabili

- Tutte le variabili iniziano con il carattere \$

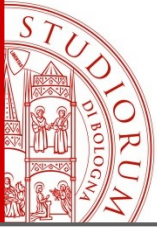
\$nome

- È debolmente tipizzato
  - Possiamo associare alla stessa variabile più tipi di dato
    - Automaticamente PHP converte la variabile nel tipo opportuno

```
$qualcosa = 3;
```

```
$qualcosa = true;
```

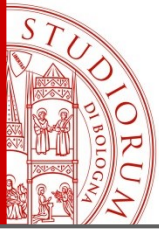
```
$qualcosa = "ciao";
```



# Tipi di dato

---

- Boolean
- Integer
- Float
- String
- Array
- Object
- NULL
- Resource



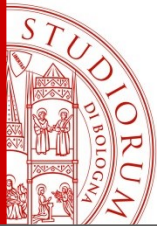
# String

- È possibile definire una stringa in 4 modi:
  1. **Single quoted:** variabili non vengono espanse e gli unici caratteri con escape ammessi sono \ e '.  
Esempio: `$stringa = 'stringa';`
  2. **Double quoted:** variabili vengono espanse, ammesse le più comuni sequenze di escape.  
Esempio: `$stringa = "stringa";`



# String

3. **Heredoc**: si comporta come le double quoted ma senza usarle (quindi il carattere " non deve essere preceduto da \).  
Esempio: `$stringa = <<<ID`  
`stringa`  
`ID;`
4. **Newdoc**: si comporta come le single quoted ma senza usarle, quindi \ e ' sono sempre trattati letteralmente.  
Esempio: `$stringa = <<<'ID'`  
`stringa`  
`ID;`



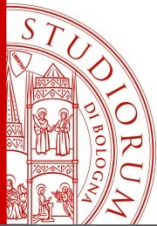
# Array

- Gli array vengono definiti con la funzione **`array()`**.

Esempio: `$esempio = array(1,2,3);`

- In PHP ci sono tre tipi di array:
  1. Indexed.
  2. Associative.
  3. Multidimensional.





# Array - Indexed

- Sono gli array *classici* con indice numerico.
- Esempio:  

```
$gelati = array("cornetto", "ghiacciolo", "ricoperto");
```
- Esempio alternativo equivalente:  

```
$gelati[0] = "cornetto";  
$gelati[1] = "ghiacciolo";  
$gelati[2] = "ricoperto";
```



# Array - Associative

- Array che hanno stringhe come indici.
- Esempio:  

```
$age = array("Peter"=>"35", "Ben"=>"37",  
"Joe"=>"43");
```
- Esempio alternativo equivalente:  

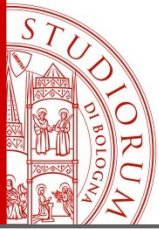
```
$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";
```




# Array - Multidimensional

- Array che contengono uno o più array.
- Esempio:  

```
$age = array(1, array(2,3,5,6), "prova",  
array("ciao",3, True));
```



# Object oriented

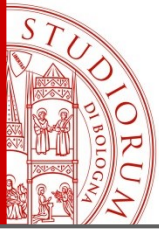
- PHP consente di definire classi e istanziare oggetti.
- Supporta i principali meccanismi dell'OOP:
  - Proprietà e metodi public, private, protected e static.
  - Ereditarietà.
  - Classi astratte.
  - Interfacce.
  - Tratti.  I Tratti (in inglese, Traits) sono una funzionalità di PHP che permette di riutilizzare il codice tra classi, fornendo un modo per includere metodi definiti altrove senza utilizzare l'ereditarietà classica.



# Object - Esempio

---

```
class Persona {  
    private $nome;  
    public $cognome;  
  
    public function __construct($nome, $cognome) {  
        $this->nome = $nome;  
        $this->cognome = $cognome;  
    }  
  
    public function presentati() {  
        echo "Sono ".$this->nome." ".$this->cognome;  
    }  
}
```



# Object – Esempio (2)

```
$gino = new Persona("Gino", "Pino");
```

```
$gino->presentati(); → Chiamata Metodo di un Oggetto  
//Mi chiamo Gino Pino.
```

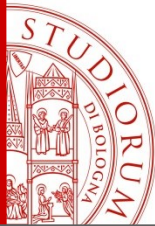
```
echo $gino->nome; → Attributo di un Oggetto  
//Fatal error
```

```
echo $gino->cognome;  
//Pino
```



# Resource

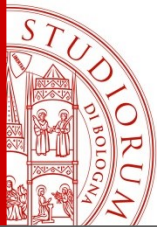
- Una risorsa non è un vero e proprio tipo
- Si tratta di una variabile speciale che contiene il riferimento ad una risorsa esterna.
- Sono create e usate da funzioni speciali.
- Esempi: file aperti o connessioni ad un database.



# Output

- In PHP ci sono due modi per ottenere un output:
  - **Echo**: può stampare 1 o più stringhe e non ha valore di ritorno.
  - **Print**: può stampare 1 sola stringa e restituisce sempre 1.
- Solitamente viene usata **echo** in quanto leggermente più veloce.





# var\_dump

- Funzione per stampare tipo e contenuto di un'espressione, utile in fase di debug

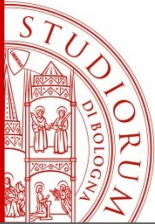
```
<?php
$a = array(1, array("a", "b", "c"));
var_dump($a);
?>
```

```
array(2) {
  [0]=>
  int(1)
  [1]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```



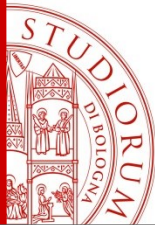
# Variabili Superglobali

- Le **variabili superglobali** sono variabili accessibili ovunque.
- Esempi:
  - **\$GLOBALS**: memorizza tutte le variabili globali
  - **\$\_SERVER**: gestisce informazioni sul server
  - **\$\_GET**: usato per collezionare dati inviati con metodo **GET**
  - **\$\_POST**: usato per collezionare dati inviati con metodo **POST**
  - **\$\_COOKIE**: gestisce i cookie
  - **\$\_REQUEST**: usato per collezionare dati inviati sia con metodo **GET** che con metodo **POST** e i cookie
  - **\$\_SESSION**: gestisce le sessioni



# Esempio Get

- Abbiamo due pagine:
  - `esempio_get.html`
  - `process_get.php`
- **`Esempio_get.html`** contiene un form con un campo di testo e il bottone di submit
- Una volta compilato il form, vogliamo mandare il testo inserito alla pagina **`process_get.php`** che leggerà i dati e restituirà una pagina HTML contenente l'informazione inserita



# Esempio Get

- **esempio\_get.html**

```
...  
<form action="process_get.php" method="get">  
    <label for="idsupereroe">Supereroe</label>  
    <input type="text" id="idsupereroe"  
name="supereroe">  
    <input type="submit">  
</form>
```

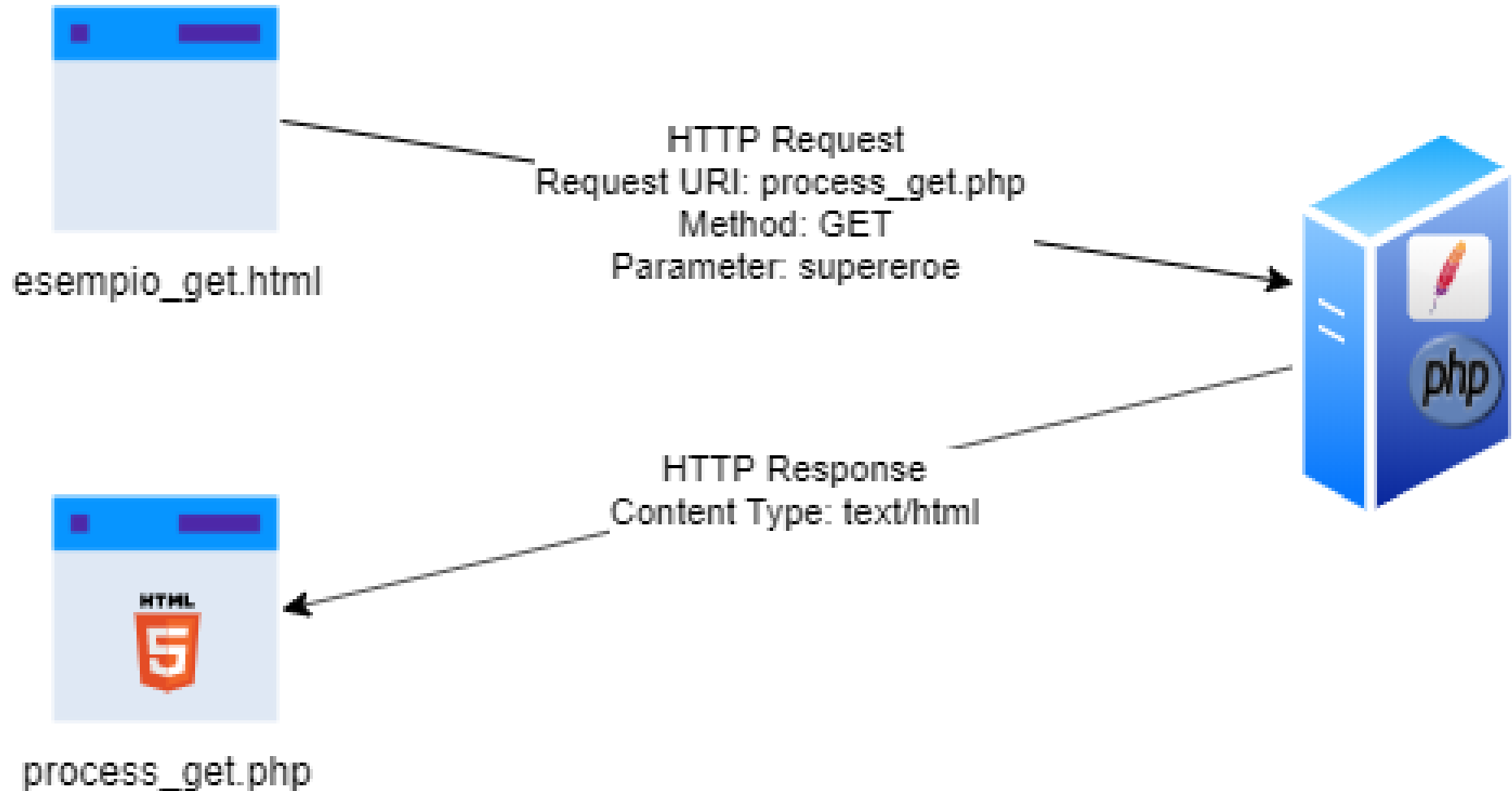
...

- **process\_get.php**

```
...  
<p>GET: <?php echo $_GET['supereroe'] ?></p>  
<p>POST: <?php echo $_POST['supereroe'] ?></p>  
<p>REQ:<?php echo $_REQUEST['supereroe'] ?></p>
```

...

# Esempio Get





# Output Esempio Get (batman)

**GET: batman**

**POST:**

perché é stato utilizzato il metodo GET, non POST

**Notice: Undefined index: supereroe**  
**in *path*\process\_get.php on line 9**

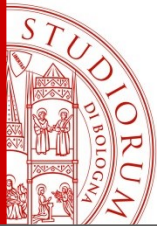
**REQ: batman**



# Esempio Get - Considerazioni

---

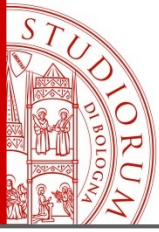
- In pratica, nell'esempio, abbiamo usato il form per generare una richiesta HTTP con metodo GET e con un particolare parametro (supereroe).
- È possibile ottenere lo stesso risultato SENZA usare il form?



# Esempio Get - Considerazioni

- In pratica, nell'esempio, abbiamo usato il form per generare una richiesta HTTP con metodo GET e con un particolare parametro (supereroe).
- È possibile ottenere lo stesso risultato **SENZA** usare il form?
- **Ovviamente sì!**
- **È possibile scrivere l'url direttamente nel browser o usare un qualsiasi software che consenta di effettuare richieste HTTP.**

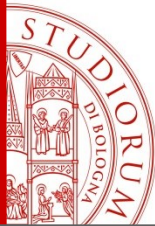




# Esempio Post

---

- Esempio speculare a quello di prima. L'unico cambiamento è il metodo usato (POST anziché GET)



# Esempio Post

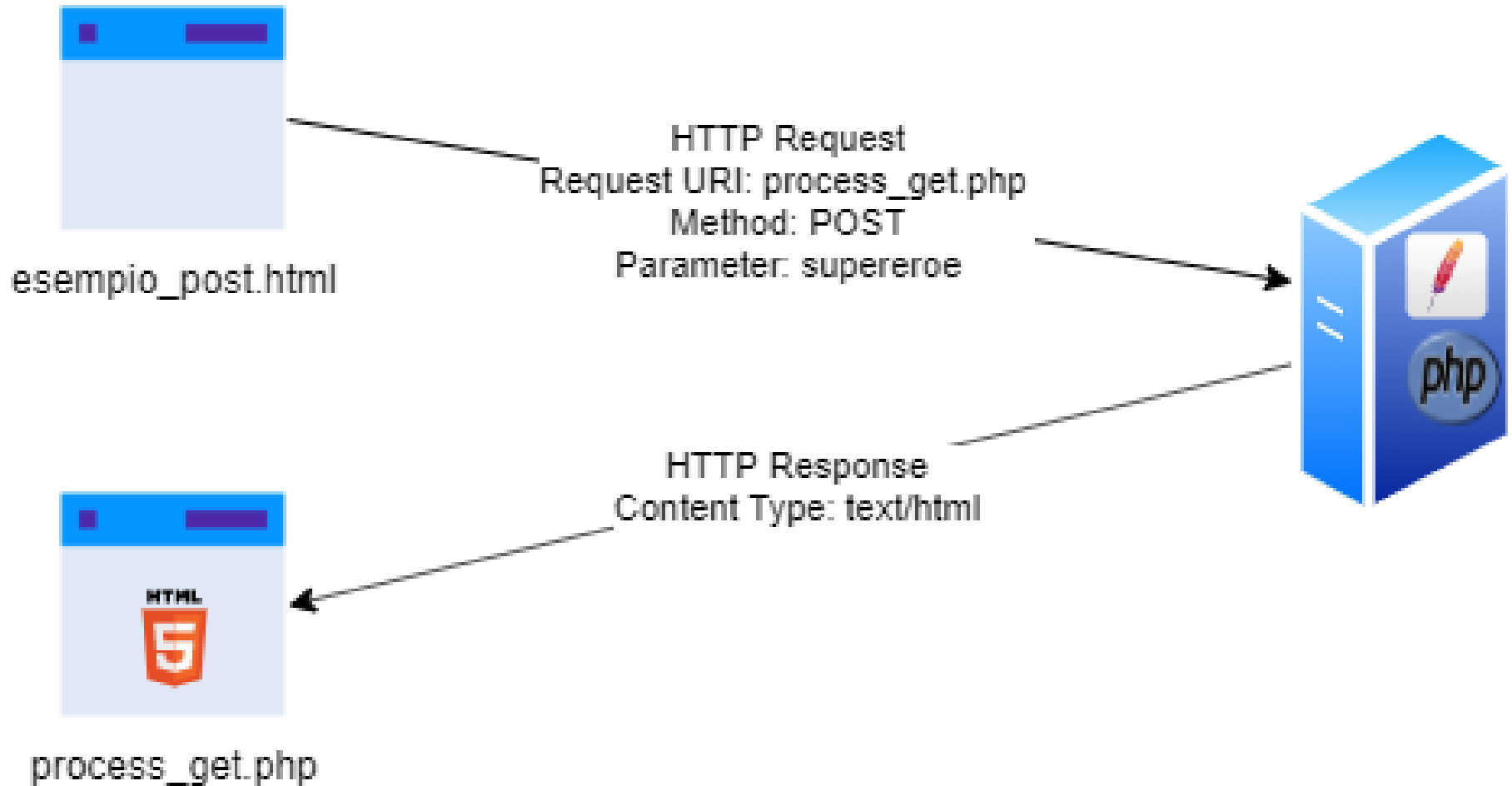
- **esempio\_post.html**

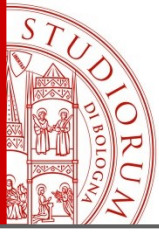
```
...  
<form action="process_get.php" method="post">  
  <label for="idsupereroe">Supereroe</label>  
  <input type="text" id="idsupereroe"  
name="supereroe">  
  <input type="submit">  
</form>
```

- **process\_get.php**

```
...  
<p>GET: <?php echo $ _GET['supereroe'] ?></p>  
<p>POST: <?php echo $ _POST['supereroe'] ?></p>  
<p>REQ:<?php echo $ _REQUEST['supereroe'] ?></p>  
...
```

# Esempio Post





# Output Esempio Post (batman)

---

**GET:**

**Notice: Undefined index: supereroe  
in**

**C:\xampp\htdocs\esempi\esempio\_post\  
process\_get.php on line 8**

**POST: batman**

**REQ: batman**



# Esempio Post - Considerazioni

---

- Valgono le stesse considerazioni dell'esempio precedente.
- I parametri inviati tramite POST sono trasmessi in chiaro?



# Esempio Post - Considerazioni

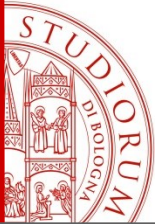
---

- Valgono le stesse considerazioni dell'esempio precedente.
- I parametri inviati tramite Post sono trasmessi in chiaro?
- **Ovviamente sì!**



# Esempio Get v2

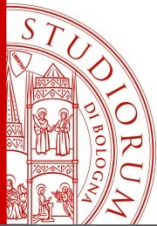
- Gli stessi esempi mostrati in precedenza per GET e POST possono essere realizzati anche utilizzando un'unica pagina che si occupa sia di visualizzare il form, se i dati non sono stati inviati, sia di visualizzare eventuali dati.



# Gestione dello stato

- Come ricordato HTTP è un protocollo **stateless**, che non tiene traccia delle diverse interazioni che sono parte di una specifica attività.
- Per superare questo limite, si utilizzano alcuni meccanismi che consentono di gestire le singole interazioni che avvengono tra client e server come parte di una stessa attività.
- **PHP** fornisce due strumenti per la **gestione dello stato**:
  1. Cookie
  2. Session





# Cookie

- Un cookie consente di salvare un'informazione nel browser dell'utente.
- In PHP è possibile creare un cookie utilizzando la funzione `setcookie()`, specificando nome (unico parametro obbligatorio), valore, validità e percorso.



# Esempio Cookie

Si vuole salvare in un cookie il numero di volte in cui l'utente ha visitato il sito web.

- Esempio:

```
$nome_cookie = "numero_accessi";  
if(!isset($_COOKIE[$nome_cookie])) {  
    echo "Cookie '" . $nome_cookie . "' non settato!  
    Lo setto adesso!";  
    $valore_cookie = 1;  
    setcookie($nome_cookie, $valore_cookie, time() +  
        (60 * 60 * 24 * 30), "/");  
} else {  
    echo "Cookie '" . $nome_cookie . "' settato!<br>";  
    $num_visite = $_COOKIE[$nome_cookie]+1;  
    setcookie($nome_cookie, $num_visite, time() +  
        (60 * 60 * 24 * 30), "/");  
    echo "Il sito è stato visitato: ".$num_visite." volte!";  
}
```



# Cookie (2)

- È possibile cancellare un cookie semplicemente impostando un tempo di validità «passato».

- Esempio:

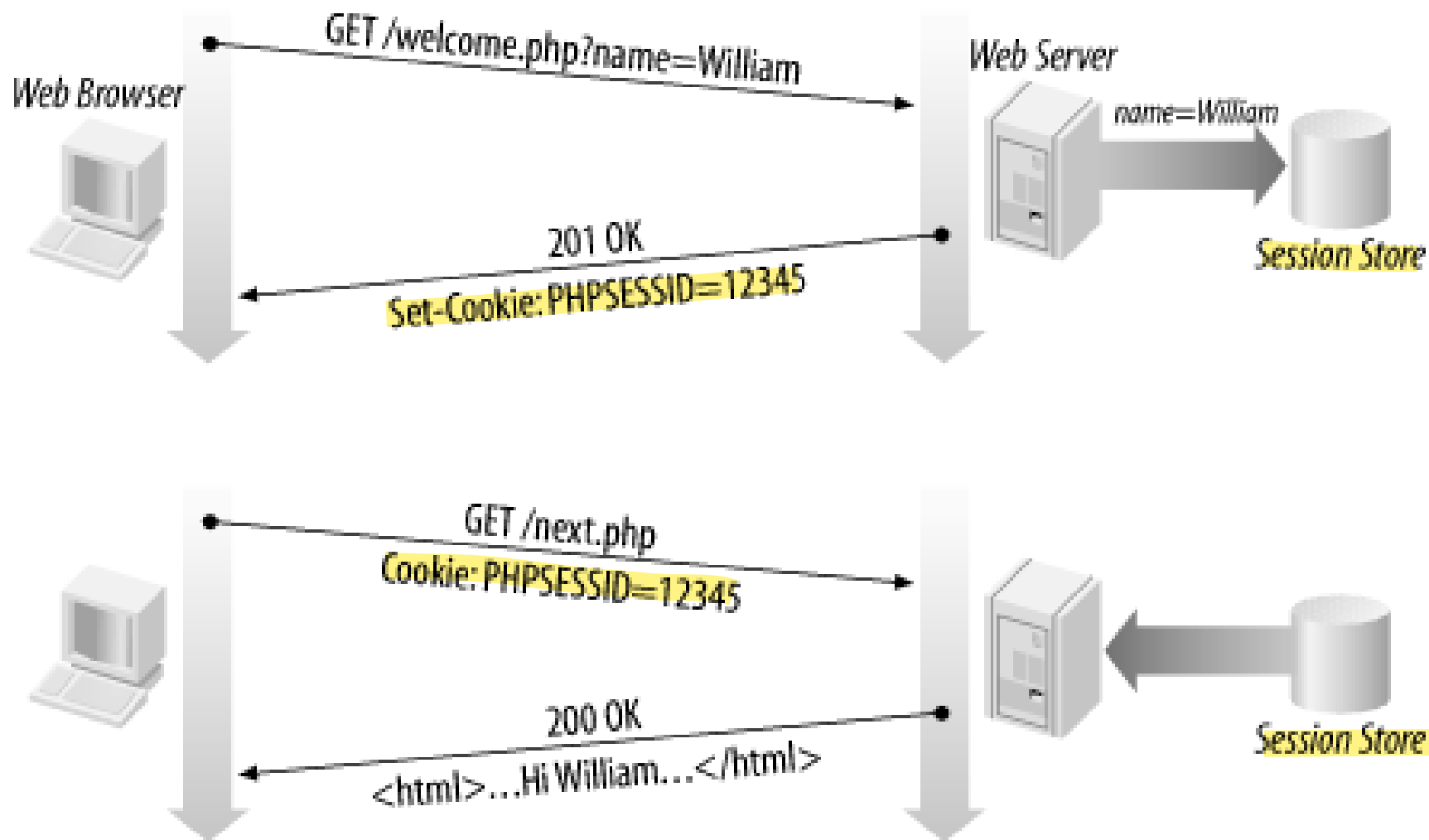
```
$nome_cookie = "numero_accessi";  
setcookie($nome_cookie, "", time() - 100, "/");  
echo "Cookie '" . $nome_cookie . "' cancellato!";
```

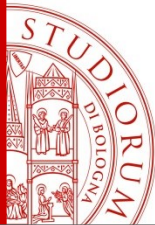


# Session in PHP

- Con una session è possibile salvare un'informazione direttamente sul server.
- Al browser viene assegnato un identificatore di sessione che viene registrato in un cookie (**PHPSESSID**) sul browser dell'utente.
- Alla successiva interazione HTTP, PHP controllerà automaticamente se un id è stato inviato con la richiesta. Se l'id è stato inviato, PHP controlla se esiste una sessione con quell'id e, in caso positivo, rende accessibili le informazioni salvate aggiungendole alla variabile super globale **\$\_SESSION**.
- La gestione del cookie lato client avviene in maniera trasparente.
- Una sessione rimane aperta fino a quando il browser non viene chiuso.

# Esempio Session



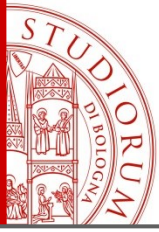


# Session in PHP

`unset()` e `session_unset()` rimuovono solo le variabili all'interno della sessione attiva (cioè, l'array `$_SESSION`), ma non cancellano i dati memorizzati fisicamente nel file di sessione sul server.

- È possibile salvare una variabile nella sessione in due modi:
  - Usando direttamente la variabile super globale `$_SESSION`:  
`$_SESSION['name'] = "William";`
  - Usando la funzione `session_register()`:  
`$name = "William";`  
`session_register("name");`
- Per rimuovere dati è possibile usare:
  - La funzione `unset()` per rimuovere una singola variabile  
`unset($_SESSION['name'])`
  - La funzione `session_unset()` per rimuovere TUTTE le variabili
  - La funzione `session_destroy()` per rimuovere tutte le informazioni della sessione (non solo le variabili) ma NON il cookie `PHPSESSID` (da usare con cautela!).

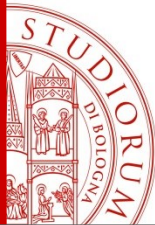
La funzione `session_destroy()` termina la sessione stessa, eliminando tutte le informazioni memorizzate per quell'ID di sessione (compresi i file di sessione sul server) ma non elimina il cookie `PHPSESSID` dal browser dell'utente.



# Session in PHP

---

- *Una sessione rimane aperta fino a quando il browser non viene chiuso.*
- Ma come fa il server a sapere che il browser è stato chiuso?



# Session in PHP

Quando il garbage collector di PHP entra in azione, rimuove i file di sessione che sono considerati scaduti.

- *Una sessione rimane aperta fino a quando il browser non viene chiuso.*
- Ma come fa il server a sapere che il browser è stato chiuso?
- **Semplicemente non lo sa!**
- Ha un suo parametro di configurazione (che si trova nel file `php.ini`) `session.gc_maxlifetime` che indica il tempo di vita di una sessione. Scaduto quello, la sessione viene ritenuta scaduta e ci sarà un garbage collector che si occuperà di liberare la memoria.
- Lato browser invece, il cookie **PHPSESSID** è impostato per scadere alla chiusura del browser.



# Prima domanda

**BONUS**

## **DOMANDA 1 :**

Cosa contiene la variabile super globale `$_REQUEST[]` in PHP?

- ☐ E' un array associativo che contiene il contenuto delle variabili super globali `$_GET`, `$_POST` e `$_COOKIE`
- ☐ E' un array associativo che contiene tutti i log del server
- ☐ E' una variabile speciale che contiene le informazioni riguardanti headers e percorsi assoluti del server
- ☐ E' una variabile speciale che permette di modificare i tipi di richieste accettabili dal server



# PHP e MySQL

- Perché MySQL?
  - È il DBMS più usato in ambito web.
  - È cross-platform
  - Usa lo standard SQL
  - È gratuitamente scaricabile ed usabile
  - È sviluppato, distribuito e supportato da Oracle



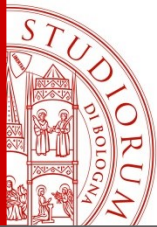
# MySQL

- È un DBMS relazionale.
- Le interrogazioni si esprimono usando SQL.  
Trovate una guida su SQL sul W3Schools  
<http://www.w3schools.com/sql/default.asp>
- Può essere utilizzato per memorizzare **enormi** quantità di dati (con InnoDB una tabella può pesare fino a 64TB).



# PHP connesso a MySQL

- PHP 5 e successivi possono lavorare con MySQL usando
  - ~~MySQL API~~ (deprecate)
  - MySQLi API (i sta per improved)
  - PDO (PHP Data Object)
- Quale dei due scegliere?
  - Ovviamente dipende dal contesto!
  - Entrambi hanno vantaggi/svantaggi: PDO si può usare con 12 differenti DB mentre MySQLi solo con MySQL ma ha prestazioni leggermente migliori
- In laboratorio useremo MySQLi.



# MySQLi

- MySQLi mette a disposizione API sia object-oriented che procedurali. Noi useremo la versione object-oriented.
- Esempio:

```
$servername = "localhost";  
$username = "username";  
$password = "password";  
$sql = "CREATE DATABASE dbname";
```

//object-oriented

```
$conn = new mysqli($servername, $username, $password);  
$conn->query($sql) === TRUE
```

//procedural

```
$conn = mysqli_connect($servername, $username, $password);  
mysqli_query($conn, $sql)
```



# Aprire/chiudere una connessione

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
$dbname = "dbname";
```

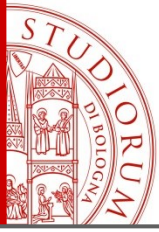
```
// Aprire la connessione
```

```
//$conn = new mysqli($servername, $username,  
$password, $dbname);
```

```
//Chiudere la connessione
```

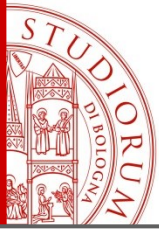
```
$conn->close();
```

```
?>
```



# Eseguire una query

- È possibile eseguire una query SQL di **qualsiasi tipo** utilizzando il metodo *query()*.
- Ovviamente il risultato sarà diverso in base al tipo di query eseguito:
  - Query per la creazione di database o tabelle restituiranno semplicemente TRUE o FALSE.
  - Query per la selezione di dati restituiranno i dati.

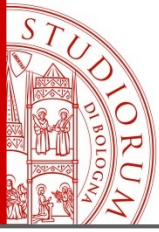


# Prepared Statement

- Per motivi di sicurezza, è preferibile usare i prepared statement al posto del metodo query.
- Con i prepared statement inizialmente la query viene creata con dei placeholder, che vengono valorizzati successivamente.
- Esempio:  

```
$stmt = $conn->prepare("SELECT *  
FROM table WHERE columnname = ?");  
$stmt->bind_param('s', $variabile);  
$stmt->execute();
```





# Prepared Statement

- Quando si esegue il binding è necessario specificare il tipo di parametro.
- Valori ammessi:
  - i: interi
  - d: double
  - s: stringhe
  - b: BLOB

Il BLOB (Binary Large Object) è un tipo di dato utilizzato nei database per memorizzare grandi quantità di dati binari.



# Creazione Tabelle

---

- In uno degli esempi precedenti avete visto come creare il db.
- La creazione del db, così come quella delle tabella, è buona norma NON gestirle con PHP ma con strumenti più adatti, ad esempio MySQL Workbench  
<https://www.mysql.com/it/products/workbench/>



# Domande?

---

