# PHP

Interagire con l'utente, GET && POST

## HTTP

- I metodi HTTP più comuni sono
  - GET, HEAD e POST.
- Il metodo GET è usato per ottenere il contenuto della risorsa indicata come URI (come può essere il contenuto di una pagina HTML).
- HEAD è analogo a GET, ma restituisce solo i campi dell'header, ad esempio per verificare la data di modifica del file. Una richiesta con metodo HEAD non prevede l'uso del body.
- Il metodo POST è usato di norma per inviare informazioni al server (ad esempio i dati di un form). In questo caso l'URI indica che cosa si sta inviando e il body ne indica il contenuto.

## HTTP

- Esempio GET

```
GET /index.html?userid=joe&password=guessme HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/4.0
```

- Esempio POST

```
POST /login.jsp HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

userid=joe&password=guessme
```

## Form

- Non essendo presente un meccanismo di prompt, PHP deve utilizzare i campi di un form HTML per ricevere input
- In PHP è estremamente facile accedere al contenuto dei campi di un form:
  - Ad ogni campo corrisponde una chiave associata ad un array superglobale
- Non si utilizza il (DOM) **D**ocument **O**bject **M**odel

## Form

- Quando l'utente preme un pulsante di tipo submit di un form, sul server viene eseguito lo script indicato dall'attributo ACTION del form stesso
- I dati possono essere inviati attraverso GET o POST

```
<form action="search.php" method="GET">
  <input type="text" name="parole">
  <input type="Submit">
</form>
```

## Leggere i campi dai form

- Dipendentemente dal metodo usato, il valore di un campo viene inserito in un array superglobale avente come chiave lo stesso nome assegnato nel form
- Gli array superglobali sono:
  - $_GET per tutti i valori passati con il metodo get
  - $_POST per tutti i valori passati con il metodo post

## Leggere i campi dai form - Esempio

- **Form.html**

```
<form action="Search.php" method="GET">
  <input type="text" name="Parole">
  <input type="submit">
</form>
```

- **Search.php:**

```
<html><body><h1>
Stai cercando: <?php echo $_GET['Parole']; ?>
</h1></body></html>
```

## Leggere i campi dai form

- Il valore inserito nelle variabili dipende dal tipo di campo a cui si riferiscono:
- Per campi di tipo:
  - TEXT
  - PASSWORD
  - TEXTAREA
  - HIDDEN
- La variabile contiene il testo immesso nel campo.

## Leggere i campi dai form

- Per i campi di tipo RADIO, viene inserito il valore dell'attributo VALUE corrispondente all'elemento selezionato.
- Nel caso dei menu a tendina (SELECT), viene inviato il testo contenuto tra i tag <OPTION> e </OPTION> corrispondente alla voce selezionata.

- **radio.html**

```
<form method="POST" action="radio.php">
  Auto <input type="radio" name="mezzo" value="A"><br>
  Moto <input type="radio" name="mezzo" value="M"><br>
  <input type="submit" value="Seleziona">
</form>
```

- **radio.php**

```
<?php
echo 'Il valore selezionato è: '.$_POST['mezzo'];
?>
```

## Leggere i campi dai form - Esempio

- **select.html**

```
<form method="POST" action="select.php">
    Nazione: <select name="nazione" size="1">
        <option value="swiss">Svizzera</option>
        <option value="french">Francia</option>
    </select>
    <input type="submit" name="Seleziona">
</form>
```

- **select.php**

```
<?php
echo 'La nazione selezionata è: '.$_POST['nazione'];
?>
```

## Leggere i campi dai form

- Per campi di tipo CHECKBOX viene inserito il valore dell'attributo VALUE se l'opzione è stata selezionata (se value è omesso, il valore di default di un checkbox è "on").
- Se la casella non è stata selezionata, la variabile contiene una stringa vuota.

## Leggere i campi dai form - Esempio

- **checkbox.html**

```
<form method="POST" action="checkbox.php">
  Tennis <input type="checkbox" name="sport[]" value="tennis">
  Vela <input type="checkbox" name="sport[]" value="vela">
  Golf <input type="checkbox" name="sport[]" value="golf"><br>
  <input type="submit" value="Seleziona">
</form>
```

- **checkbox.php**

```
<?php
echo 'Gli sport selezionati sono:<br>';
foreach($_POST['sport'] as $sport)
  echo $sport.'<br>';
?>
```

## URL

- È possibile passare i parametri a uno script php attaccando all'indirizzo dello script le copie attributo/valore come in una get
- È possibile passare più coppie di attributo/valore separandole dal carattere &

```
http://localhost/index.php?nome=davide&nazione=svizzera&id=
  12
```

- Le coppie attributo valore sono:
  - nome equivale a davide
  - nazione equivale a svizzera
  - id equivale a 12

## Leggere i valori passati da URL

- Esempio

```
http://localhost/get.php?nome=davide&nazio
  ne=svizzera&id=12
```

**get.php**

```
<?php
echo 'Il nome è '.$_GET['nome'].'<br>';
echo 'Ia nazione è '.$_GET['nazione'].'<br>';
echo 'L\'id è '.$_GET['id'];
?>
```

## Quante pagine?

- Quante pagine HTML/PHP abbiamo bisogno per creare interattività tra l'utente e l'applicazione/server?
  - Almeno 2 pagine (una per il form e l'altra per processarla PHP-GET)
  - Almeno 2 pagine (una per il form e l'altra per processarla PHP-POST)
- Ma c'è un'altra possibilità… fare tutto con una singola pagina PHP/HTML.

## Una sola pagina

- Come fare?
- Semplice, basta **“intercettare” il metodo** con cui viene chiamata la pagina:
  - **GET**, se viene richiamata attraverso un link (sempre) oppure attraverso un form method=”GET”
  - **POST**, sempre e solo attraverso un form con il method=”POST”
- Possiamo quindi intercettare il metodo con l'utilizzo di un array superglobale $_SERVER

## $_SERVER

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
echo 'Pagina POST';
} else {
echo 'Pagina GET';
}
```

- Semplice esempio in cui eseguo un'operazione se la pagina viene chiamata con il metodo POST, altrimenti ne eseguo un'altra.
- Possiamo quindi condizionare l'esecuzione in base allo stato della pagina.
- Provate a stampare **print_r($_SERVER)** per vedere le diverse coppie attributo/valore a disposizione.

## Esempio

- Proviamo a scrivere il codice per stampare un form per l'inserimento di 2 valori.
- Una volta premuto **submit** il form scompare e vengono stampati i valori e il risultato della loro somma
- Il tutto risiede su un'unica pagina HTML/PHP

## Codice

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
echo $_POST['val1'] . ' + ' . $_POST['val2'] . ' = ';
echo $_POST['val1'] + $_POST['val2'];
} else {
?>
<form method="POST" action="#">
Valore 1: <input type="text" name="val1"><br>
Valore 2: <input type="text" name="val2"><br>
<input type="submit">
</form>
<?php } ?>
```

## Sicurezza: test dei dati in arrivo dal client

- "… all input is evil until proven otherwise …"

- I dati in arrivo dal client devono essere validati!
- PHP ha varie funzioni che permettono di testare lo stato delle variabili
- Il test dipende dalle abilità del programmatore!

## Sicurezza: test dei dati in arrivo dal client

| | |
|---|---|
| `isset($_POST["nomevar"])` | **true** (1), se esiste $_POST["nomevar"]<br>**false** (0), altrimenti |
| `empty($_POST["nomevar"])` | **true** (1), se $_POST["nomevar"]è vuota<br>**false** (0), altrimenti |

## Sicurezza: pulizia dei dati in arrivo dal client

- Spesso è necessario ripulire i dati in arrivo dal client …

```
trim($_POST["nomevar"])

toglie spazi, \n, \r, \t, \v, \0
```

```
nl2br($_POST["nomevar"])

rimpiazza \n con <BR>
```

## Sicurezza: casting dei dati in arrivo dal client

```
intval($_GET["nomevar"])
converte $_GET["nomevar"] in intero, se
possibile

Es. 12a -> 12
    a12 -> 0


doubleval($_GET["nomevar"])

strval($_GET["nomevar"])
```

## Sicurezza: regular expressions - sintassi

| **Regular Expression** | **Will match…** |
|---|---|
| foo | The string “foo” |
| ^foo | “foo” at the start of a string |
| foo$ | “foo” at the end of a string |
| ^foo$ | “foo” when it is alone on a string |
| [abc] | a, b, or c |
| [a-z] | Any lowercase letter |
| [^A-Z] | Any character that is not a uppercase letter |
| (gif\|jpg) | Matches either “gif” or “jpg” |
| [a-z]+ | One or more lowercase letters |
| [0-9.-] | Ðny number, dot, or minus sign |
| ^[a-zA-Z0-9_]{1,}$ | Any word of at least one letter, number or _ |
| ([wx])([yz]) | wy, wz, xy, or xz |
| [^A-Za-z0-9] | Any symbol (not a number or a letter) |
| ([A-Z]{3}\|[0-9]{4}) | Matches three letters or four numbers |

## Sicurezza: regular expressions - esempio

```
<?php
$email = "abc123@sdsd.com";
$regex = '/^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$/';
if (preg_match($regex, $email)) {
  echo $email . " is a valid email. We can accept it.";
 } else {
  echo $email . " is an invalid email. Please try again.";
}
?>
```