

Passaggio di parametri al server da form HTML

Spesso la pagina HTML richiesta dal client consente all'utente di inviare dei dati al server, per l'elaborazione. Un modo molto semplice per comunicare questi dati è l'utilizzo di **form HTML**¹. La comunicazione tra client e server avviene utilizzando il **protocollo HTTP**.

L'invio e l'elaborazione dei dati possono essere organizzati utilizzando due pagine:

- ▶ una **pagina principale** contenente i vari campi dei form, che consentono all'utente di immettere dei dati;
- ▶ una **pagina secondaria**, che viene richiamata dalla principale e che contiene lo script PHP. Questo elaborerà i dati producendo la risposta da inviare al client che ne ha fatto richiesta.

L'invio dei dati dal form avviene mediante un **pulsante submit**.

Per poter inviare e gestire i dati dobbiamo ancora fornire due informazioni: il nome del file che contiene lo script PHP e il metodo HTTP che vogliamo utilizzare per inviare i dati al server. Queste informazioni devono essere specificate rispettivamente negli attributi **action** e **method** del form.

```
<form name="nome_form" method="metodo-HTTP" action="script_php">
```

Metodi HTTP

Il protocollo HTTP (**H**yper **T**ext **T**ransfer **P**rotocol) è usato come principale sistema per la trasmissione di informazioni sul web. Esso si basa sullo scambio di messaggi. I tipi di messaggi sono due:

- ▶ messaggi di richiesta (**HTTP request**);
- ▶ messaggi di risposta (**HTTP response**).

Vediamo, molto in generale, quali sono le caratteristiche di queste due tipologie di messaggi.

■ HTTP request

La richiesta è una linea di testo (ASCII) divisa in 3 parti:

- ▶ **Request line** – Contiene l'indicazione del metodo HTTP, l'URI, che identifica la risorsa che si vuole ottenere e la versione del protocollo.
- ▶ **Header** – Contiene alcune informazioni come il nome del server a cui si riferisce l'URI (**host**) e il tipo di client che ha inviato la richiesta (**user agent**).
- ▶ **Body** – Il contenuto della richiesta.

Le modalità di invio di dati al server sono tipicamente due e dipendono da quale metodo del protocollo http si intende utilizzare: **get** o **post**. Esistono comunque anche ulteriori metodi.²

¹ Si veda <http://www.scribd.com/cinziabb/d/80180572-HTML-Form>

² Metodi HTTP: <http://openskill.info/infobox.php?ID=357>

► Metodo GET

Quando si sceglie il metodo get, le variabili ed il relativo valore vengono fornite allo script destinatario, tramite la barra degli indirizzi del browser. Nell'URL della pagina di risposta potremo vedere tutti i parametri nella barra degli indirizzi (più precisamente nella "**query string**").

La lista dei parametri, ciascuno nella forma **nome_parametro=valore_parametro** appare dopo un punto interrogativo, posto alla fine dell'URL. Ogni parametro è separato da quelli adiacenti mediante il simbolo "&".

Per esempio, se si inserisce come username "aaa" e come password "bbb", la query string avrà la forma seguente:

http://miosito/path/login.html?username=aaa&password=bbb

dove *miosito* è il nome di dominio del sito, cioè il nome che identifica univocamente l'host, e *path* è il percorso interno alla document root del server che conduce al file login.html.

► Metodo POST

Quando si sceglie il metodo post, i dati non compaiono nella query string ma sono contenuti nel messaggio inviato.

■ HTTP response

Il messaggio di risposta è una linea di testo (ASCII) divisa in 3 parti:

- **Status line** – Contiene un codice di stato, nella seguente forma:
 - 1xx: Informational (messaggi informativi).
 - 2xx: Success (la richiesta è stata soddisfatta).
 - 3xx: Redirection (non c'è risposta immediata, ma la richiesta è sensata e viene detto come ottenere la risposta).
 - 4xx: Client error (la richiesta non può essere soddisfatta perché sbagliata).
 - 5xx: Server error (la richiesta non può essere soddisfatta per un problema interno del server).³
- **Header** – Contiene alcune informazioni come il tipo e la versione del server (**server**), il tipo di contenuto restituito (**MIME type**), per esempio text/html, per i documenti HTML, image/jpeg per un'immagine in formato jpeg.
- **Body** – Il contenuto della risposta.

Recupero dei dati inviati al server

■ Recupero dei dati inviati mediante caselle (di testo/di password/nascoste)

I dati inviati mediante il form vengono memorizzati in una variabile superglobale⁴ dal nome **\$_GET**, se il metodo usato per l'invio è get o **\$_POST** se il metodo di invio è post (l'uso delle maiuscole è obbligatorio). Le variabili in questione sono array associativi che:

- contengono tanti elementi quanti sono i dati inviati;
- hanno gli indici che corrispondono ai valori degli attributi name dei controlli del form.

³ Per maggiori dettagli consultare http://it.wikipedia.org/wiki/Elenco_dei_codici_di_stato_HTTP

⁴ Le variabili superglobali sono variabili globali, cioè sempre disponibili all'interno dell'ambiente di esecuzione, e native.

Consideriamo, ad esempio, il seguente form.

```
<html>
<head>
  <title>Esempio Login</title>
</head>
<body>
  <form name="login-form" method="post" action="login.php">
    <label for="username">Username</label>
    <input id="username" name="username" type="text">
    <label for="password">Password</label>
    <input id="password" name="password" type="password">
    <input id="login-button" type="submit" value="Login">
  </form>
</body>
</html>
```

Username	<input type="text"/>	Password	<input type="password"/>	<input type="submit" value="Login"/>
----------	----------------------	----------	--------------------------	--------------------------------------

Se proviamo a stampare l'array `$_POST` dopo aver inviato username e password, utilizzando la funzione `print_r`, otteniamo:

Array ([username] => aaa [password] => bbb)

dove username è il valore dell'attributo username del campo corrispondente del form e password è il valore dell'attributo name del campo password del form, mentre aaa e bbb sono i dati inviati.

Possiamo rappresentare l'array anche con questa notazione più familiare:

<code>\$_POST</code>	aaa	bbb
	username	password

A questo punto è abbastanza evidente che, per recuperare i dati inviati è sufficiente riferirsi agli elementi dell'array `$_POST` utilizzando la notazione tipica degli array:

`$_POST["username"]` e `$_POST["password"]`

Solitamente, per semplificare la scrittura, si assegnano tali valori a delle variabili locali allo script:

`$uid = $_POST["username"]`

`$pwd = $_POST["password"]`

Occorre osservare che i dati memorizzati in `$_POST` o `$_GET` sono tutti quelli che derivano dagli elementi input del form e che possiedono un attributo name valorizzato. Infatti, se si assegna al pulsante submit un attributo name

`<input id="login-button" name="login-button" type="submit" value="Login">`

e si stampa con `print_r` l'array, si ottiene:

Array ([username] => aaa [password] => bbb [login-button] => Login)

Si può notare che il valore inviato, corrispondente al pulsante login-submit, è il contenuto dell'attributo value.

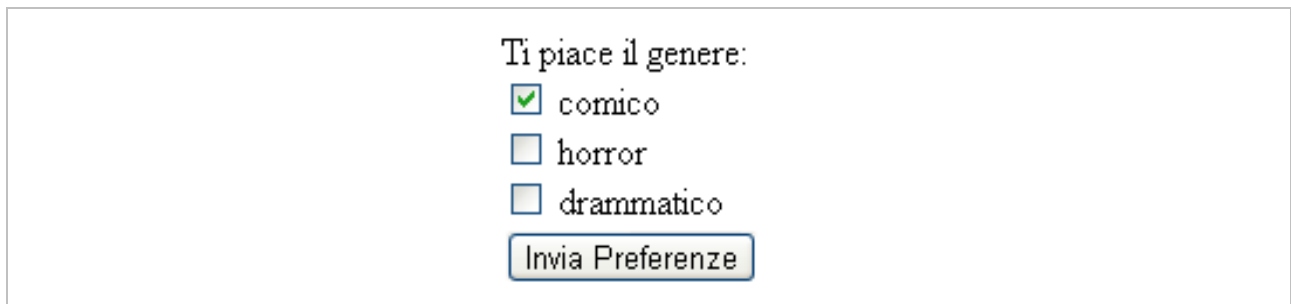
I dati inviati vengono memorizzati anche in un'altra variabile superglobale che si chiama **\$_REQUEST**. Questo array associativo contiene sia gli elementi provenienti dagli array `$_POST` o `$_GET` che quelli provenienti dall'array `$_COOKIE`, che contiene i cookie dell'utente. Se esiste la possibilità che il nome di un cookie coincida con il nome di un dato inviato a mezzo form, conviene non utilizzare `$_REQUEST` poiché i dati potrebbero essere sovrascritti.

■ Recupero dei dati inviati mediante checkbox

Consideriamo il seguente esempio:

```
<html>
  <head>
    <title>
      Form con caselle di controllo
    </title>
  </head>
  <body>
    <form name="checkbox-form" method="post" action="checkbox.php">
      Ti piace il genere: <br/>
      <input type="checkbox" name="opzione1" value="comico" checked/>
      comico <br/>
      <input type="checkbox" name="opzione2" value="horror"/>
      horror <br/>
      <input type="checkbox" name="opzione3" value="drammatico"/>
      drammatico <br/>
      <input type="submit" value="Invia Preferenze" />
    </form>
  </body>
</html>
```

Il codice dell'esempio crea il seguente form:



Ti piace il genere:

☒ comico

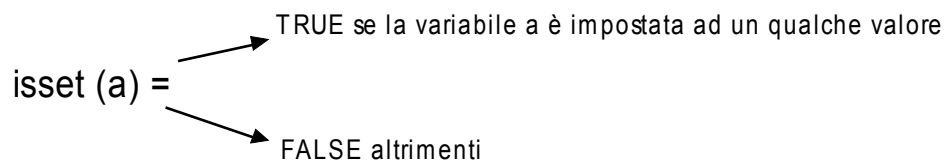
☐ horror

☐ drammatico

Per determinare, mediante uno script PHP, quale casella di controllo l'utente ha selezionato, occorre controllare il contenuto dell'array `$_POST`. Per ogni casella selezionata dall'utente, il contenuto del suo campo value sarà trasferito nell'array all'indice individuato dal valore dell'attributo name. Di conseguenza, supponendo che l'utente scelga la casella di controllo "comico", la situazione sarà la seguente:

- `$_REQUEST["opzione1"] = "comico"`
- `$_REQUEST["opzione2"]` non impostato
- `$_REQUEST["opzione3"]` non impostato

Per controllare se un elemento dell'array è stato impostato, si usa la funzione PHP ***isset***. Questa funzione restituisce true se una variabile è stata impostata e false altrimenti.



Per esempio, il seguente script PHP produce a video le checkbox selezionate.

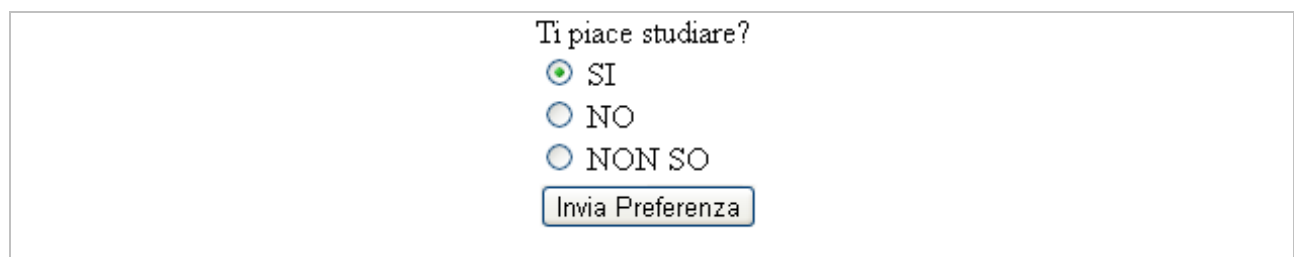
```
<html>
  <head>
    <title>
      Checkbox.php
    </title>
  </head>
  <body>
    <p> Hai scelto le seguenti opzioni: </p> <br/>
    <?php
      if(isset($_POST["opzione1"]))
        echo $_POST ["opzione1"]."<br/>";
      if(isset($_POST ["opzione2"]))
        echo $_POST ["opzione2"]."<br/>";
      if(isset($_POST ["opzione3"]))
        echo $_POST ["opzione3"];
    ?>
  </body>
</html>
```

■ Recupero dei dati inviati mediante radio button

Consideriamo il seguente esempio:

```
<html>
  <head>
    <title>
      Form con gruppo di pulsanti di opzione
    </title>
  </head>
  <body>
    <form name="radiobutton-form" method="post" action=" radiobutton.php">
      Ti piace studiare? <br/>
      <input type="radio" name="domanda1" value="si" checked/>
      SI <br/>
      <input type="radio" name="domanda1" value="no"/>
      NO <br/>
      <input type="radio" name="domanda1" value="non so"/>
      NON SO <br/>
      <input type="submit" value="Invia Preferenza" />
    </form>
  </body>
</html>
```

Il codice dell'esempio produce il seguente form:



Potete determinare nel codice PHP quale pulsante radio l'utente ha selezionato, testando il contenuto di `$_POST["domanda1"]`. Per esempio, il seguente script PHP produce a video il pulsante selezionato.

```
<html>
  <head>
    <title>
      Radiobutton.php
    </title>
  </head>
  <body>
    <p> Hai scelto la seguente opzione: </p> <br/>
    <?php
      echo $_POST ["domanda1"]."<br/>";
    ?>
  </body>
</html>
```

■ Recupero dei dati inviati mediante elenchi a scelta singola

Consideriamo il seguente esempio.

```
<html>
  <head>
    <title> Form con elenco a scelta singola
  </title>
  </head>
  <body>
    <form name="select-single-form" method="post" action=" select-single.php ">
      Fai la tua scelta: <br/>
        <select name="colori">
          <option value="rosso"/> ROSSO </option>
          <option value="blu"/> BLU </option>
          <option value="verde"/> VERDE </option>
          <option value="giallo"/> GIALLO </option>
          <option value="bianco"/> BIANCO </option>
        </select>
        <input type="submit" value="Conferma Scelta" />
      </form>
    </body>
  </html>
```

Il codice dell'esempio crea il seguente form:

Fai la tua scelta:

ROSSO ▼	Conferma Scelta
---------	-----------------

Siccome non è stato impostato l'attributo multiple, l'utente può selezionare solo una voce dall'elenco. Dopo aver premuto il pulsante "Conferma Scelta", la voce selezionata viene inviata al server e la si può recuperare dall'array \$_POST["colori"]. Il seguente script PHP produce in output il colore scelto dall'utente.

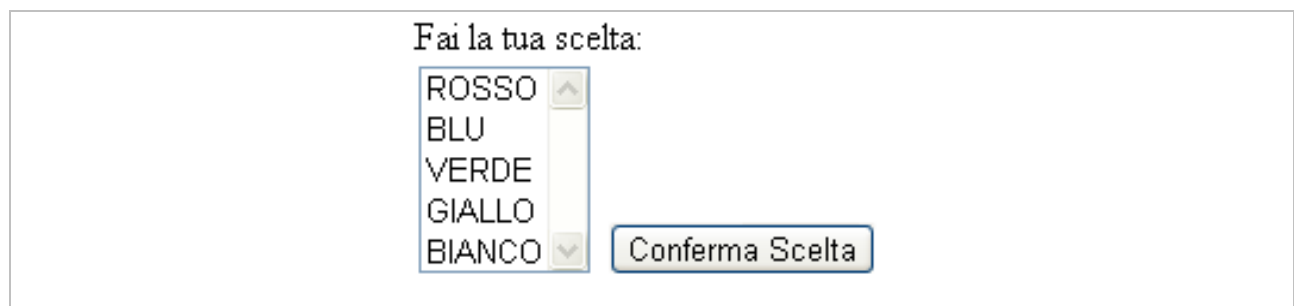
```
<?php
  echo "Hai scelto il colore ".$_POST ["colori"];
?>
```

■ Recupero dei dati inviati mediante elenchi a scelta multipla

Consideriamo il seguente esempio.

```
<html>
  <head>
    <title> Form con elenco a scelta multipla
  </title>
  </head>
  <body>
    <form name="select-multiple-form" method="post" action=" select-multiple.php ">
      Fai la tua scelta: <br/>
        <select name="colori[ ]" multiple>
          <option value="rosso"/> ROSSO </option>
          <option value="blu"/> BLU </option>
          <option value="verde"/> VERDE </option>
          <option value="giallo"/> GIALLO </option>
          <option value="bianco"/> BIANCO </option>
        </select>
        <input type="submit" value="Conferma Scelta" />
      </form>
    </body>
  </html>
```

Il form generato è il seguente:



Osservate che il valore dell'attributo name di **SELECT** è l'array colori[].

In questo caso, l'utente può selezionare più voci dall'elenco. Queste sono inserite in un array all'interno dell'array \$_POST["colori"]. Si tratta quindi di gestire un **array bidimensionale**. Per esempio, se l'utente sceglie i colori blu e bianco, si avrà:

```
$_REQUEST["colori"] [0] = blu
$_REQUEST["colori"] [1] = bianco
```

Per estrarre tutte le scelte dall'array, occorre utilizzare un ciclo **foreach**, come nel seguente esempio.

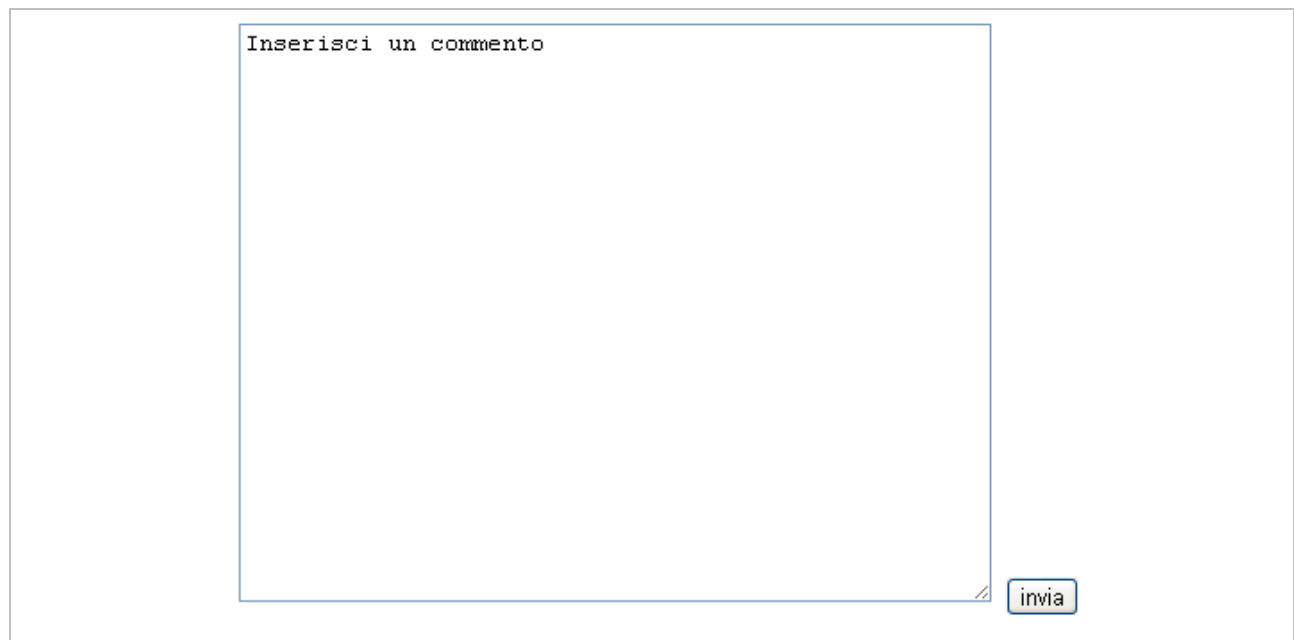
```
<?php
  foreach($_POST ["colori"] as $colore)
  {
    echo "Hai scelto il colore ".$colore."<br/>";
  }
?>
```


■ Recupero dei dati inviati mediante aree di testo

Consideriamo il seguente esempio.

```
<html>
  <head>
    <title> Form con area di testo </title>
  </head>
  <body>
    <form name="textarea-form" method="post" action=" textarea.php ">
      <textarea name="commento" rows="20" cols="50">
        Inserisci un commento
      </textarea>
      <input type="submit" value="invia" />
    </form>
  </body>
</html>
```

Il codice dell'esempio crea la seguente textarea:

A screenshot of a web browser displaying a form. The form consists of a large rectangular text area with a blue border. Inside the text area, the text "Inserisci un commento" is visible at the top. To the right of the text area, there is a small rectangular button with the text "invia" inside it.

Per recuperare il testo è sufficiente accedere a `$_POST["commento"]`, come nel seguente script di esempio.

```
<?php
  echo $_POST ["commento"];
?>
```

Quest'opera è stata rilasciata con licenza Creative Commons Attribution-ShareAlike 3.0 Unported. Per leggere una copia della licenza visita il sito web <http://creativecommons.org/licenses/by-sa/3.0/> o spedisce una lettera a Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.