Gestione dell'input

Corso di Laboratorio di Informatica

Prof. Dei Rossi, Leonardo Essam

Introduzione

Settimana scorsa abbiamo iniziato a vedere che uno dei vantaggi di PHP è la possibilità di "passare dei dati" da front-end (HTML) e back-end (codice PHP).

Questo avviene tramite l'uso delle <form>:

```
<form action="welcome.php" method="POST">
    <label for="idName">Name:</label>
    <input type="text" id="idName" name="name">

    <label for="idEmail">E-mail:</label>
    <input type="email" id="idEmail" name="email">

    <input type="submit">
</form>
```

Gli elementi della form in HTML

Il linguaggio HTML mette a disposizione diversi tipi di campo di input, tra cui:

- Campo di testo (`type="text"`);
- Numerico (`type="number"`);
- Indirizzo e-mail (`type="email"`);
- Nascosto (`type="hidden"`);
- E tanti altri...

Per riferimento: (link)

Gli attributi di un input nella form (1)

Osserviamo l'esempio precedente:

```
<form action="welcome.php" method="POST">
   [...]
</form>
```

Notiamo che il tag `<form>` ha due attributi:

- La destinazione (`action`);
- Il metodo della richiesta (`method`).

Gli attributi di un input nella form (2)

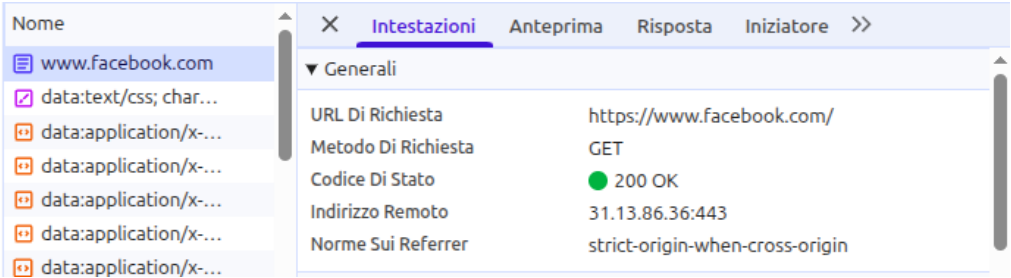
La destinazione (`action`) indica la pagina PHP alla quale inviare i dati della form, osserviamo che la pagina di destinazione non deve per forza coincidere con la pagina attuale!

Il metodo della richiesta (`method`) rappresenta concettualmente il tipo di richiesta che si vuole fare al server.

Gli attributi di un input nella form (3)

I tipi di richiesta più comuni sono GET e POST.

La richiesta di tipo GET viene usata come predefinita quando si vuole accedere ad una pagina Web:



Gli attributi di un input nella form (4)

La richiesta di tipo POST invece viene usata quando è l'utente a voler inviare dei dati al server.

Ciò che si vuole inviare possono essere dati di qualsiasi tipo:

- Campi di testo (ad esempio una form di log-in);
- Caricamento di un file (ad esempio un post su Instagram);
- ...

Gli attributi di un input nella form (5)

Il fatto di differenziare i due tipi di richiesta, oltre che a una correttezza formale porta anche dei vantaggi pratici.

Primo tra tutti è la possibilità, soprattutto per le SPA (single-page application), di differenziare il tipo di azione da compiere basandosi sul tipo di richiesta in ingresso.

Passaggio da front-end a back-end (1)

Tornando all'esempio precedente:

```
<form action="welcome.php" method="POST">
   <label for="idName">Name:</label>
   <input type="text" id="idName" name="name">

   [...]
</form>
```

Osserviamo che il campo di input per il nome ha due elementi identificativi:

- Un ID (`id="idName"`);
- Un nome (`name="name"`).

Buonarroti

Cosa cambia?

Passaggio da front-end a back-end (2)

Il campo `ID` è l'attributo che viene usato lato front-end per accedere all'entità dell'input HTML tramite il DOM. Un esempio è:

```
1  let name = document.getElementById("idName");
2  [...]
```

Il campo `ID` identifica quindi il singolo elemento della pagina HTML **unicamente** a livello di browser Web e non di back-end!

Passaggio da front-end a back-end (3)

Il campo `name` invece rappresenta il singolo input all'interno della form in cui si trova, questo significa che è possibile avere una pagina Web con due o più form dove dentro ognuna di esse ha al suo interno un input con lo stesso nome.

Riassumendo:

- **Campo ID:** univoco a livello globale della pagina;
- **Campo name:** univoco a livello locale della form.

Passaggio da front-end a back-end (4)

Lato server quindi l'attributo interessato è il `name`.

All'atto della *submit* di una form, PHP riceve dal browser dell'utente un insieme sotto forma di oggetto chiave-valore dove la chiave corrisponde al nome del campo di input e il valore è il contenuto dell'input stesso.

Gestione dell'input (1)

Tornando all'esempio iniziale:

```
<form action="welcome.php" method="POST">
    <label for="idName">Name:</label>
    <input type="text" id="idName" name="name">

    <label for="idEmail">E-mail:</label>
    <input type="email" id="idEmail" name="email">

    <input type="submit">
</form>
```

Gestione dell’input (2)

Ciò che il browser invierà al server sarà:

```
HTTP/1.1 POST /welcome.php

name: "Leonardo Essam"
email: "leonardo.deirossi@buonarroti.tn.it"
```

Gestione dell’input (3)

Lato server è possibile accedere a tali attributi utilizzando l’array associativo in base al metodo di richiesta utilizzato.

Per le richieste di tipo POST:

```
$name = $_POST["name"];
$email = $_POST["email"];
```

Allo stesso modo, per le richieste di tipo GET:

```
$name = $_GET["name"];
$email = $_GET["email"];
```

Gestione dell'input (4)

Segue quindi che `$_POST` e `$_GET` sono due array predefiniti di PHP che contengono al loro interno i dati ricevuti dalle rispettive richieste POST e GET dal client.

Per le richieste di tipo GET, le coppie chiave-valore sono quelle che vengono aggiunte come parametro alla URL della pagina:

```
/welcome.php?name=Leonardo%20Essam&email=leona...
```

Buonarroti