

Estendete il vostro calendario

Nella terza parte di questa serie di tutorial imparerete ad aggiungere le funzionalità per scegliere le corse dinamicamente e vederne i dettagli



Nell'ultima puntata avete imparato le basi dell'uso degli array, dei blocchi di controllo (*if/else*) e delle funzioni, sia personalizzate

che native di PHP. Avete messo tutto insieme per creare un calendario per la stagione di F1. In questa puntata modificherete la funzionalità del calendario permettendo all'utente di selezionare una corsa specifica da un menu a tendina in HTML. Diamo per scontato che abbiate già configurato la vostra piattaforma Linux e che serva già le pagine PHP attraverso il browser Web, come spiegato nella prima puntata. Se così non fosse, fate riferimento all'articolo precedente o alla sezione intitolata **Installare PHP su Linux**.

Form e sicurezza

I form ricorrono spesso nella carriera di ogni sviluppatore, e tuttavia alcune buone pratiche vengono spesso ignorate; specialmente nel muovere i primi passi nel mondo di PHP. Su qualsiasi genere di input da parte dell'utente dovete fare una serie di filtri e **validazioni**, poiché non potete mai essere sicuri della provenienza dei dati e del loro contenuto. Volete verificare che l'utente vi abbia fornito effettivamente solo i dati di cui avete bisogno. La validazione è il processo che vi consente di assicurarvi che l'input ricevuto sia quello che vi aspettate (nel formato corretto, eccetera). Il filtraggio, o bonifica, è la pulizia dell'input da caratteri non desiderati così che sia sicuro da usare. Senza questi processi, aprite i vostri script PHP a possibili attacchi di *code injection*. Sfrutterete le funzioni native di PHP per la validazione e il filtraggio, così da poter essere sicuri nel maneggiare l'input utente. Comincerete creando un file PHP vuoto (ad esempio **racess.php**) e all'inizio scrivete un blocco PHP vuoto con una variabile vuota, **\$race_data** (userete questa variabile più tardi):

```
<?php
```

```
$race_data = false;
```

```
?>
```

Prima di cominciare a scrivere lo script completo avete bisogno di sistemare alcune cose. Per primo create un array di corse all'interno del blocco di codice dopo la riga **\$race_data** appena scritta (rileggete la scorsa puntata se avete dimenticato come si dichiarano gli array). Non dovete usare per forza le corse di F1, potete inserire quelle del GP di moto o qualsiasi altro tipo di dati vogliate. Per quelle di F1, fate riferimento a **www.formula1.com/races/calendar.html** per i dati ufficiali. L'array risulterà circa così (la maggior parte dei dati è stata tagliata per ragioni di spazio):

```
$races = array(
```

```
    'Australia' => array('title' => 'Gran Premio dell'Australia',
```

```
    'location' => 'Melbourne', 'date' => '18/3/2012'),
```

```
    'Malesia' => array('title' => 'Gran Premio di Malesia', 'location'
```

```
    => 'Kuala Lumpur', 'date' => '25/3/2012'),
```

```
    'Cina' => array('title' => 'Gran Premio della Cina', 'location' =>
```

```
    'Shanghai', 'date' => '15/4/2012'),
```

```
    'Bahrain' => array('title' => 'Gran Premio del Bahrain', 'location'
```

```
    => 'Sakhir', 'date' => '22/4/2012'),
```

```
    'Spagna' => array('title' => 'Gran Premio di Spagna', 'location' =>
```

```
    'Catalunya', 'date' => '13/5/2012'),
```

```
...)
```

Potete notare che gli array sono diversi dall'ultima puntata: ora è stata usata un luogo al posto della data come chiave dell'array. Il vostro obiettivo è permettere all'utente di scegliere il luogo della gara, così da poterne vedere i dettagli, e per questo è più comodo avere il dato come chiave.

Aggiungere l'HTML

Il passo successivo è aggiungere il codice HTML e il form per permettere la scelta della gara all'utente. Si suppone abbiate un'infarinatura di HTML, dal momento che una spiegazione dell'HTML sarebbe al di fuori degli scopi di questo articolo. Create una pagina basilare (head, body, ecc.) immediatamente sotto il blocco di codice PHP. Il form è basilare e consiste in un menu a tendina e un pulsante **Submit**. Per popolare dinamicamente i luoghi nel menu a tendina utilizzerete la funzione **foreach()** di PHP. Avete già incontrato in precedenza questa funzione, per riassumere permette di ciclare o iterare in un array e accedere ai dati di ogni elemento. Nell'area body dell'HTML, aggiungete il form seguente:

```
<form method="post">
```

```
<fieldset>
```

```
<label for="location">Scegli una gara:
```

```
</label>
```

```
<select name="location">
```



2012 FIA Formula One World Championship Race Calendar		
01	2012 FORMULA 1 AUSTRALIAN GRAND PRIX (Melbourne)	18 - 19 Mar
02	2012 FORMULA 1 PETRONAS MALAYSIA GRAND PRIX (Kuala Lumpur)	23 - 25 Mar
03	2012 FORMULA 1 UBS CHINESE GRAND PRIX (Shanghai)	13 - 15 Apr
04	2012 FORMULA 1 GULF AIR BAHRAIN GRAND PRIX (Sakhir)	20 - 22 Apr
05	FORMULA 1 GRAN PREMIO DE ESPAÑA SANTANDER 2012 (Catalunya)	11 - 13 May
06	FORMULA 1 GRAND PRIX DE MONACO 2012 (Monte Carlo)	25 - 27 May
07	FORMULA 1 GRAND PRIX DU CANADA 2012 (Montréal)	08 - 10 Jun
08	2012 FORMULA 1 GRAND PRIX OF EUROPE (Valencia)	22 - 24 Jun
09	2012 FORMULA 1 SANTANDER BRITISH GRAND PRIX (Silverstone)	06 - 08 Jul
10	FORMULA 1 GROSSER PREIS SANTANDER VON DEUTSCHLAND 2012 (Hockenheim)	20 - 22 Jul
11	FORMULA 1 ENI MAGYAR NAGYDÍJ 2012 (Budapest)	27 - 29 Jul
12	2012 FORMULA 1 SHELL BELGIAN GRAND PRIX (Spa-Francorchamps)	31 Aug - 02 Sep
13	FORMULA 1 GRAN PREMIO SANTANDER D'ITALIA 2012 (Monza)	07 - 09 Sep
14	2012 FORMULA 1 SINGTEL SINGAPORE GRAND PRIX (Singapore)	21 - 23 Sep
15	2012 FORMULA 1 JAPANESE GRAND PRIX (Suzuka)	05 - 07 Oct
16	2012 FORMULA 1 KOREAN GRAND PRIX (Yongam)	12 - 14 Oct
17	2012 FORMULA 1 AIRTEL GRAND PRIX OF INDIA (New Delhi)	26 - 28 Oct
18	2012 FORMULA 1 ETHAD AIRWAYS ABU DHABI GRAND PRIX (Yas Marina Circuit)	02 - 04 Nov
19	2012 FORMULA 1 UNITED STATES GRAND PRIX (Austin)	16 - 18 Nov
20	FORMULA 1 GRANDE PREMIO DO BRASIL 2012 (São Paulo)	23 - 25 Nov

Il sito ufficiale della Formula 1 (**www.formula1.com**) è la miglior risorsa a vostra disposizione, con tutte le ultime news, recensioni e date delle gare


```
<?php foreach($races as $location =>
$race): ?>
    <option value="<?php echo
$location; ?>"><?php echo $location; ?></option>
<?php endforeach; ?>
</select>
<input type="submit"
name="submit" value="Vedi" />
</fieldset>
</form>
```

Visualizzato lo script sul vostro browser, e dovrete trovarvi un menu a tendina con tutte le gare. Come potete vedere, un minimo di codice vi permette di aggiungere un menu a tendina con tutti gli elementi dell'array. Il menu è stato chiamato 'location', ma potete scegliere qualsiasi nome preferiate. Dopo il menu, è stato aggiunto un pulsante Submit, così da inviare il form dopo la scelta del luogo. Potrete notare che il metodo del form è impostato a **post**. Potete utilizzare **get** se preferite, ma se volete mantenere pulita l'URL è preferibile **post**. Se volete conoscere la differenza tra **POST** e **GET**, fate riferimento al box titolato **GET vs POST in breve**.

Controllare i dati utente

Omettendo l'attributo **action** il form invierà i dati alla pagina stessa. Come sapete che il form è stato inviato? Per default, quando dati in POST o GET vengono inviati a uno script, PHP converte tali dati in un array, utilizzando i nomi dei campi come chiavi; questo è estremamente utile, poiché potete controllare la presenza di uno specifico campo del form con la funzione **isset()**. Aggiungete il codice seguente appena sotto l'array **races**:

```
if(isset($_POST['location']))
    check_for_race($_POST['location']);
```

Con questa riga controllate se sono disponibili dati in **\$ _POST** e se al suo interno esiste la chiave **location**. Se avete modificato il nome della vostra select modificate anche questo codice di conseguenza. Se il campo è presente (ovvero il form è stato inviato), chiamate la funzione **check_for_race()** passando la scelta come argomento. Non avete ancora scritto questa funzione, quindi fatelo ora!

Signori, avviate i motori

Avete il vostro array di gare, la pagina HTML e il form; tutto quello che vi serve ora è di validare l'input utente e controllare i dettagli della gara. Se viene trovata una gara corrispondente nell'array, mostrerete i dettagli all'utente, altrimenti visualizzerete un messaggio d'errore. Copiate il codice PHP seguente appena sotto la riga **if(isset(\$_POST...))**; aggiunto sopra:

```
function check_for_race($location) {
    global $races, $race_data;
    $location = filter_input(INPUT_POST,
'location', FILTER_SANITIZE_STRING,
FILTER_FLAG_STRIP_LOW);
    if(isset($races[$location]))
        $race_data = $races[$location];
    else
        $race_data = 'No matching races
have been found.';
    return;
}
```

Installare PHP su Linux

La maggior parte delle distribuzioni Linux contiene già PHP. Anche se potete lanciare gli script PHP da riga di comando, qui userete un browser Web (e quindi un server Web). Potete seguire questo tutorial caricando i vostri file PHP in un server Web in Rete, se ne avete uno, oppure sfruttare un'installazione standard di Apache 2 sulla vostra macchina locale; quest'ultima soluzione si può rivelare più pratica e veloce del caricare ogni

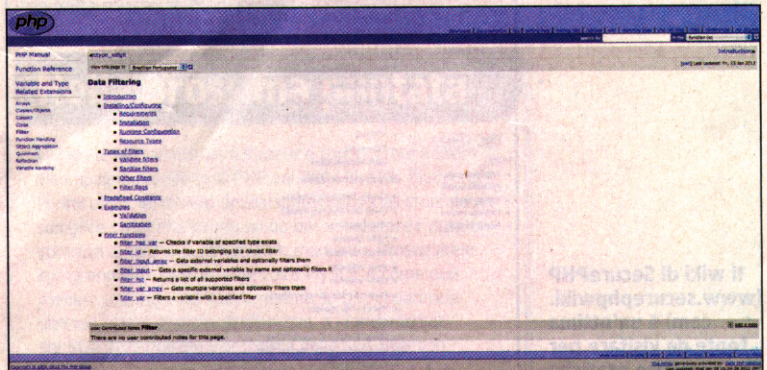
volta i file via FTP. Se avete bisogno di istruzioni per l'installazione e la configurazione, vi raccomandiamo di leggere la sezione **Installation on Unix Systems** nel manuale ufficiale di PHP su <http://php.net/manual/en/install.unix.php>. Alternativamente, esistono centinaia di guide per l'installazione scritte per praticamente qualsiasi distribuzione Linux. Google vi aiuterà nella vostra ricerca se il sito di PHP non basta.

Prima di fare qualsiasi cosa, dichiarate la funzione **check_for_race()**. La funzione conterrà tutto il codice per la validazione e l'assegnamento. La prima riga, che comincia con **global**, è una novità. Dal momento che avete definito l'array **races** all'esterno della funzione, è necessario dire a PHP che volete accedervi dall'interno della funzione stessa. Con la direttiva **global**, preceduta da un array esistente o da nomi di variabile, consentite l'accesso ai dati alla vostra funzione. Lo scope può essere difficile da comprendere: se per voi è una novità, controllate il box informativo per una spiegazione più dettagliata. La riga successiva è dove viene applicata la validazione e il filtraggio. PHP (dalla 5.2) contiene già delle funzioni di filtraggio native per il controllo dell'input. Potete verificare che l'input sia in un determinato formato, in un range, o conforme a un pattern specifico. La funzione **filter_input()** vi permette di validare interi, float, indirizzi e-mail, URL o scrivere le vostre espressioni regolari per controllare determinate stringhe. Potete anche lanciare alcune funzioni di bonifica sui dati, ed è la cosa più interessante! Noterete che sono stati passati quattro argomenti alla funzione **filter_input()**. Il primo indica a PHP che volete gestire i dati che arrivano da **\$ _POST**. Proseguendo, indicate il nome del campo che volete bonificare, **location**. Non dimenticate, se avete modificato il nome del campo in precedenza, di cambiare di conseguenza questa riga. I due argomenti successivi sono quelli realmente interessanti. Il primo, **FILTER_SANITIZE_STRING**, indica a PHP che vi aspettate una stringa, e che volete rimuovere caratteri potenzialmente dannosi; per default, PHP rimuoverà per voi qualsiasi tag dalla stringa. A questo punto passate l'argomento

Tip



Commentate sempre il vostro codice man mano che lo scrivete. Anche se al momento non vi sembra aver molto senso, lo acquisterà quando tornerete su quello stesso codice qualche mese più tardi. I commenti daranno un senso a tutte quelle complicate funzioni che avete scritto.



Il sito ufficiale di PHP è il luogo migliore per apprendere le altre opzioni di filtraggio disponibili nativamente

Tip



Usate un editor di testo che disponga dell'evidenziazione della sintassi per PHP: vi aiuterà ad identificare velocemente nel vostro codice blocchi specifici o funzioni. Esistono molti programmi liberi, quindi date un'occhiata in giro e scegliete ciò che più vi aggrada.

FILTER_FLAG_STRIP_LOW, che indica a PHP di rimuovere qualsiasi carattere potenzialmente dannoso. Esistono sei possibili opzioni che potete utilizzare, dall'encoding delle ampersand (&) al non encoding degli apici all'interno delle stringhe o togliere o codificare valori a fondo scala. I valori a fondo scala per PHP sono in base alla tabella ASCII. La tabella ASCII standard (numeri, lettere e ampersand) inizia da **#32** e finisce a **#127**. Tutto quello che sta sotto il 32 è considerato basso (ad esempio, fine riga, tabulazioni, caratteri nulli, ecc.) e tutto quello sopra il 127 è considerato alto (accentate, simboli di valuta, di numerazione e così via). Specificando se volete tali caratteri rimossi o codificati, potete controllare meglio come PHP gestisce le stringhe. Dal momento che i nomi delle gare, o altri dati, potrebbero includere caratteri stranieri, li lascerete; input come spazi, tabulazioni, eccetera, non sono richiesti, quindi andrete a toglierli. Validare e bonificare i dati

è un'operazione fondamentale quando l'input arriva dal mondo esterno, non solo perché date potere a un attaccante di entrare nel vostro codice, ma anche perché gli utenti non sono sempre prevedibili e possono inserire dati non corretti per sbaglio. Controllando i dati prima di usarli riducete la possibilità di problemi nell'applicazione. Non è assolutamente la sola precauzione che dovrete avere gestendo l'input utente, ma è un buon punto di partenza. Una volta validati e bonificati i dati in input (il valore dal menu a tendina), li riassegnate alla variabile **\$location**, così da assicurarvi di gestire dati sicuri. Le due righe seguenti controlleranno se la gara effettivamente esiste, e cosa fare se così non fosse. Riconoscerete immediatamente la chiamata **isset()** dallo scorso tutorial, ed è anche pressoché identico a come controllate i dati in input da **POST**. Dal momento che le chiavi dell'array sono definite come nomi di luogo, è facile controllare se una data gara esiste. Se la chiave esiste nell'array, avete i dati: lo controllate con la scorciatoia tra parentesi quadre e utilizzando la funzione **isset()** per ritornare **true** o **false** (per massima compatibilità potrete usare

anche la funzione **array_key_exists()**). Se la funzione ritorna true, avete i dati per quella gara, e quindi potete assegnare il valore dell'elemento alla variabile **\$race_data**. Se la funzione ritorna invece false, significa che il valore scelto dall'utente non esiste, quindi ritornerete un messaggio d'errore. Potreste chiedervi come può un utente scegliere un valore inesistente se usate l'array stesso per popolare il menu a tendina nell'HTML. Nove volte su dieci questo non sarà un problema, ma in questo caso state prendendo una misura precauzionale. Talvolta gli URL possono essere malformati o gli utenti possono modificare l'HTML per vedere cosa possono combinare; mostrando solamente l'elemento dell'array o un messaggio d'errore qualora non sia valido, evitate che il vostro script vada in errore

ed esponga messaggi di PHP o altre informazioni potenzialmente private. Potreste pensare che non sia molto importante al momento, ma se utilizzate un database o avete dati

potenzialmente privati nello script o nel server, tanto meno rivelate riguardo lo script o il server, tanto meglio è. Senza queste pratiche, ad esempio, se lo script andasse in errore su una connessione al database, potrebbe stampare lo username e la password, dati utilizzabili per un eventuale attacco. Dovreste essere sempre proattivi nelle vostre misure di sicurezza, per non avere la sfortuna di dover essere reattivi. Dopo aver assegnato un valore a **\$race_data**, che sia un array o un messaggio d'errore, ritornate con **return**. Non è indispensabile, ma è sempre buona norma ritornare da una funzione quando avete terminato la sua routine.

Mostrare i dati all'utente

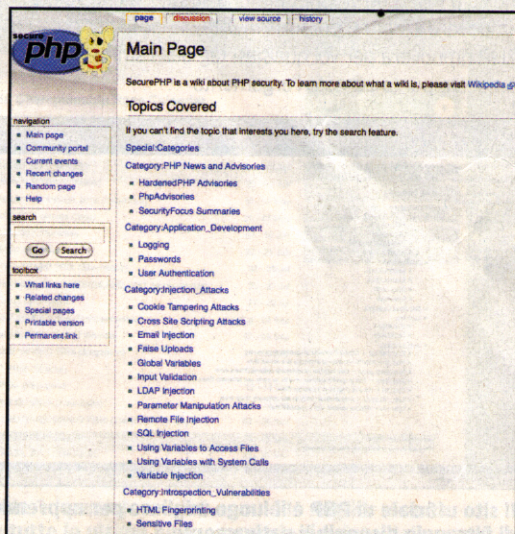
Ora che avete i dati, a prescindere dal contenuto, dovete trovare un modo per mostrarli all'utente. Lo farete integrando un po' di codice PHP aggiuntivo nella pagina HTML, in modo simile a quello che avete usato per il menu a tendina. Sotto al form, nell'area body, aggiungete:

```
<?php if(isset($race_data)): ?> <h3><?php echo $race_data['title']; ?> - <?php echo $race_data['date'];
```

GET vs POST in breve

GET e **POST** sono entrambi parte delle linee guida RFC sul protocollo HTTP che definisce il metodo di trasferimento dati tra il client (il vostro browser Internet, ad esempio) e un server Web. Quando scrivete un URL nel browser, questo invia una richiesta GET (in chiaro) al server per il contenuto che volete visualizzare. Le richieste GET possono diventare facilmente lunghe, e offrono informazioni "copiabili" sulla struttura della vostra applicazione. I dati (tipicamente in formato chiave/valore) vengono inviati come parte dell'URL e interpretati dal server.

Le richieste **POST** sono simili alle **GET**, possono cioè essere usate per recuperare contenuti. Tuttavia, qualsiasi dato inviato con la richiesta (sempre in coppie chiave/valore) è trasmesso in modo trasparente all'utente e l'URL non può essere manipolato facilmente per modificare il dato stesso. **POST** è perfetto per inviare dati a un sito senza offuscarne l'URL.



Il wiki di SecurePHP (www.securephpwiki.com) è un'ottima fonte da visitare per imparare pratiche generali di sicurezza sul codice PHP

