



Calendari online

Continuando il tutorial introduttivo di due mesi fa, vi facciamo esplorare gli array e le funzioni così da costruire un calendario di eventi personalizzato per il vostro sito Web



Due mesi fa avete appreso le basi del PHP, incluso come fu creato il linguaggio e come successivamente crebbe. Allo stesso tempo, avete avuto una prima infarinatura su alcune parti del linguaggio, come le variabili, le stringhe, gli interi e la funzione interna di PHP **date()**. In questa puntata espanderete queste parti, e vi introdurremo il concetto di array e di funzione per costruire un calendario funzionante. Prima di cominciare assicuratevi di avere la vostra piattaforma Linux configurata per servire pagine PHP attraverso il browser, come spiegato nello scorso numero. Quindi, cos'è esattamente un array? Per definirlo, torniamo alla puntata di due numeri fa dove avete utilizzato una variabile (**\$display_text**) per conservare un messaggio in stringa semplice. Il problema delle variabili è che conservano un solo pezzo di informazione alla volta: non sarebbe magnifico poter registrare valori multipli in una singola variabile? Ecco dove entrano in gioco gli array.

Array

Il miglior modo per pensare un array è come tipo speciale di variabile che contiene altre variabili. Un array vi permette di conservare quanti elementi volete al suo interno (l'unico limite è la memoria a disposizione di PHP). Potete iterare tutti gli elementi al suo interno (traversing) e PHP offre più di 70 funzioni per manipolare i vostri array, come la ricerca, il conteggio degli elementi, la rimozione dei duplicati e addirittura l'inversione dell'ordine degli elementi.

“Un array vi permette di conservare al suo interno quanti elementi volete”

Per crearne uno, il codice è banale:

```
$data = array();
```

Ora avete un array vuoto chiamato **\$data**. Gli array sono strutturati per utilizzare un indice chiave-valore; per default, quando aggiungete elementi a un array vuoto, la posizione del primo elemento sarà **0**, quella del secondo **1** e così via. Potete anche creare array con dati prepopolati (se sapete già cosa dovrà contenere). Per farlo, estendete il codice di prima inserendo il contenuto separato da virgole:

```
$data = array('Rosso','Arancio','Giallo','Verde','Blu');
```

Ed ecco dove entra in gioco il sistema di chiavi.

Il modo in cui PHP interpreta l'array appena definito è:

```
0 = 'Rosso', 1 = 'Arancio', 2 = 'Giallo', 3 = 'Verde', 4 = 'Blu'
```

Come potete vedere, ogni chiave è associata al valore corrispondente; la cosa importante da ricordare è che gli array iniziano sempre dalla chiave **0** e non

dalla **1** come molti pensano: all'inizio è facile da dimenticare.

Array associativi

Gli array hanno anche

la flessibilità di permettervi di specificare le vostre chiavi (in questo caso di definiscono array associativi). Questo aiuta molto se volete registrare un valore su una specifica chiave senza affidarvi a indici automatici. Diciamo che volete registrare i dati di una persona in un array, potete fare una cosa del genere:

```
$persona = array('name' => 'Ciro Mattia Gonano', 'comune' => 'Rovigo', 'età' => 30);
```

Utilizzando l'istruzione di associazione (**=>**) dite a PHP che volete creare una chiave denominata **name** e memorizzarvi **Ciro Mattia Gonano**. Potete memorizzare un qualsiasi tipo di dato in un array, anche altri array. Il modo in cui PHP interpreta il vostro array **persona** è come vi aspettate: **'name' = 'Ciro Mattia Gonano', 'comune' = 'Rovigo', 'età' = 30**

Quando volete usare l'elemento di un array, quello che dovete fare è chiamare l'array e la chiave desiderata:

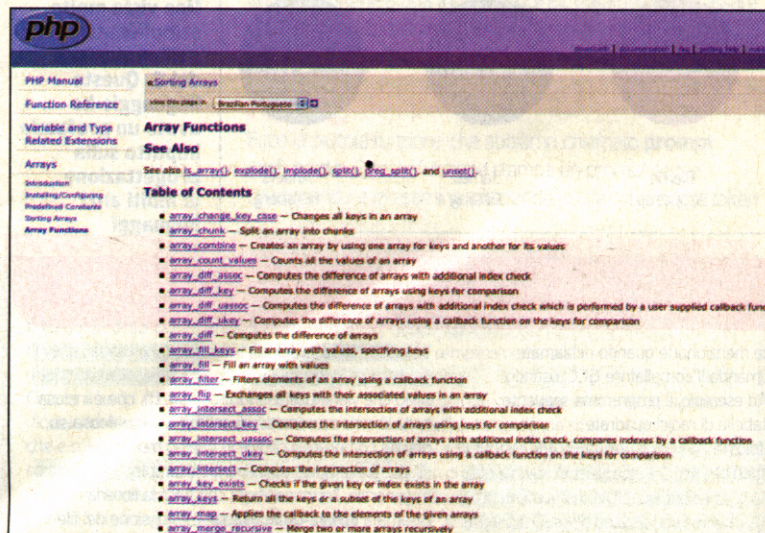
```
echo $data[0];
```

Questo stamperà la stringa **Rosso** a video. Per stampare **Arancio** dovrete semplicemente cambiare la chiave da **0** a **1**. L'array **persona** è altrettanto semplice: per stampare a video il nome sarà sufficiente

```
echo $persona['name'];
```

Aggiungere dati al vostro array

Ora avete definito gli array esistenti, ma volete aggiungerci dati: come fare? Esistono diversi modi per farlo, e spesso dipende dal fatto che gli array



Malgrado tutti i libri scritti sul linguaggio, il sito di PHP è tutt'ora il manuale di riferimento più completo e aggiornato a vostra disposizione

siano associativi o meno. Per aggiungere un elemento a **\$data**, ad esempio, potete usare

```
$data['Indaco'];
```

Utilizzando le parentesi quadre vicino al nome dell'array, PHP riconosce l'azione come un push del valore alla fine dell'array; esiste anche una funzione per farlo:

```
array_push($data, 'Indaco');
```

Questa funzione prende un minimo di due argomenti. Il primo è l'array nel quale volete inserire i dati, e tutti gli elementi successivi verranno inseriti alla fine dell'array. In questo modo potete inserire velocemente elementi multipli in una sola volta:

```
array_push($data, 'Indaco', 'Viola');
```

Se vi serve inserire solamente un elemento, il primo metodo (parentesi quadre) è raccomandato, perché non ha l'overhead di sistema che ha una chiamata a funzione. Se volete aggiungere un altro elemento al vostro array **\$persone**, vi basterà specificare la chiave che volete usare, assieme al valore richiesto: **\$persone['professione'] = 'Sviluppatore';**

Array di array

Come detto prima, un array può contenere qualsiasi tipo di elemento, inclusi altri array. La pratica di array multipli è comune e vi tornerà estremamente utile. Ancora, esistono diversi modi per farlo, e quello che userete sarà in base alla vostra struttura di array. Per questo esempio, immaginate di avere un array di piloti F1 della squadra McLaren. Aprite un editor di testo, inserite il codice PHP seguente e salvatelo

“La pratica di usare array multipli è comune e vi tornerà molto utile”

come **drivers.php**:

```
<?php
$drivers[] = array(
    'name' => 'Jenson Button',
    'nationality' => 'British',
    'championships' => 1,
);
$drivers[] = array(
    'name' => 'Lewis Hamilton',
    'nationality' => 'British',
    'championships' => 1,
);
?>
```

Le parentesi quadre hanno istruito PHP a inserire i dati del pilota alla fine dell'array. Ogni elemento nell'array dichiarato deve essere separato da virgola. Ora il vostro array **\$drivers** contiene due elementi, e ognuno di tali elementi è un array di dati contenenti informazioni sul pilota che volete visualizzare.

Agli occhi di PHP, i dati di Jenson Button sono memorizzati in **\$drivers[0]** mentre quelli di Lewis Hamilton in **\$drivers[1]**. Potete creare anche chiavi personalizzate, in luogo di **0** e **1**. Ora potete semplicemente visualizzare i dati con **echo** e quindi specificare la chiave dell'array (come **\$drivers[0]**) ma come potete visualizzare ogni elemento se non

Installare PHP su Linux

La maggior parte delle distribuzioni Linux includono PHP. Anche se potete lanciare script PHP da riga di comando, userete il browser Web (e quindi il server Web) per questo tutorial. Potete seguire la guida caricando i vostri file PHP in un server Web su internet, se ne avete uno; oppure potete utilizzare un'installazione standard di Apache 2 nella vostra macchina Linux locale, cosa che probabilmente è più facile

e veloce di dover caricare ogni volta i file via FTP. Se vi servono istruzioni per installazione o la configurazione nella vostra macchina, puntate il browser all'indirizzo <http://php.net/manual/en/install.unix.php>; in alternativa, esistono centinaia di guide per l'installazione per praticamente qualunque tipologia di Linux: vi basterà una rapida ricerca con Google.

sapete quant'è grande un array? Fortunatamente esiste una semplice funzione di controllo chiamata **foreach()** che vi permette di farlo. Potreste domandarvi perché non conoscete la lunghezza di un array: bene, conoscete esattamente la composizione dei dati per ogni pilota, ma una query su database potrebbe ritornare un solo pilota, o cinque. Potreste calcolare il numero totale degli elementi dell'array con una funzione PHP, ma **foreach()** è più semplice e vi permette di scrivere codice più compatto. Il controllo **foreach()** vi dà modo di iterare su un array. Utilizzando **drivers.php** appena creato, copiate il codice sottostante sotto al blocco che contiene l'array:

```
foreach($drivers as $driver) {
    echo 'Nome: ' . $driver['name'] . '<br />';
    echo 'Nazionalità: ' . $driver['nationality'] . '<br />';
    echo 'Campionati vinti: ' . $driver['championships'] . '<br /><br />';
}
```

Visualizzando il file sul vostro browser, vedrete una lista di piloti sullo schermo:

- Nome: Jenson Button
- Nazionalità: British

- Campionati vinti: 1
- Nome: Lewis Hamilton
- Nazionalità: British
- Campionati vinti: 1

A ogni ciclo di **\$drivers**, il valore dell'elemento corrente viene assegnato a **\$driver** (il primo sarà Jenson Button), e il puntatore interno dell'array si sposterà di uno; quindi nel ciclo successivo otterrete l'elemento seguente (Lewis Hamilton). Questo fino

Tip



Utilizzate un editor di testo con l'evidenziazione della sintassi per PHP: vi aiuterà a identificare velocemente blocchi di codice o funzioni al suo interno. Esistono diversi programmi gratuiti, quindi provatene qualcuno e trovate quello che si avvicina di più ai vostri gusti.

```
1 <?php
2
3 is_race_day();
4
5 function is_race_day() {
6     $race_dates = array(
7         '18/3/2012' => array('title' => 'Australian Grand Prix', 'location' => 'Melbourne'),
8         '25/3/2012' => array('title' => 'Malaysia Grand Prix', 'location' => 'Kuala Lumpur'),
9         '15/4/2012' => array('title' => 'Chinese Grand Prix', 'location' => 'Shanghai'),
10        '22/4/2012' => array('title' => 'Bahrain Grand Prix', 'location' => 'Sakhir'),
11        '13/5/2012' => array('title' => 'Spanish Grand Prix', 'location' => 'Catalunya'),
12    );
13
14    $date = date('m/d/Y');
15
16    if(array_key_exists($date, $race_dates)) {
17        echo "Today's race is the " . $race_dates[$date]['title'] . " in " . $race_dates[$date]['location'];
18    }
19    else {
20        echo "There is no race today.";
21    }
22    }
23
24    Running syntax check with PHP 5.3.6 (cli)
25    No syntax errors detected
```

Utilizzando un editor di codice con il controllo di sintassi integrato (come Eclipse o NetBeans) vi risparmierà tempo e frustrazioni!

siano associativi o meno. Per aggiungere un elemento a **\$data**, ad esempio, potete usare

```
$data['Indaco'];
```

Utilizzando le parentesi quadre vicino al nome dell'array, PHP riconosce l'azione come un push del valore alla fine dell'array; esiste anche una funzione per farlo:

```
array_push($data, 'Indaco');
```

Questa funzione prende un minimo di due argomenti. Il primo è l'array nel quale volete inserire i dati, e tutti gli elementi successivi verranno inseriti alla fine dell'array. In questo modo potete inserire velocemente elementi multipli in una sola volta:

```
array_push($data, 'Indaco', 'Viola');
```

Se vi serve inserire solamente un elemento, il primo metodo (parentesi quadre) è raccomandato, perché non ha l'overhead di sistema che ha una chiamata a funzione. Se volete aggiungere un altro elemento al vostro array **\$persona**, vi basterà specificare la chiave che volete usare, assieme al valore richiesto: **\$persona['professione'] = 'Sviluppatore';**

Array di array

Come detto prima, un array può contenere qualsiasi tipo di elemento, inclusi altri array. La pratica di array multipli è comune e vi tornerà estremamente utile. Ancora, esistono diversi modi per farlo, e quello che userete sarà in base alla vostra struttura di array. Per questo esempio, immaginate di avere un array di piloti F1 della squadra McLaren. Aprite un editor di testo, inserite il codice PHP seguente e salvatelo

“La pratica di usare array multipli è comune e vi tornerà molto utile”

come **drivers.php**:

```
<?php
$drivers[] = array(
    'name' => 'Jenson Button',
    'nationality' => 'British',
    'championships' => 1,
);
$drivers[] = array(
    'name' => 'Lewis Hamilton',
    'nationality' => 'British',
    'championships' => 1,
);
?>
```

Le parentesi quadre hanno istruito PHP a inserire i dati del pilota alla fine dell'array. Ogni elemento nell'array dichiarato deve essere separato da virgola. Ora il vostro array **\$drivers** contiene due elementi, e ognuno di tali elementi è un array di dati contenenti informazioni sul pilota che volete visualizzare.

Agli occhi di PHP, i dati di Jenson Button sono memorizzati in **\$drivers[0]** mentre quelli di Lewis Hamilton in **\$drivers[1]**. Potete creare anche chiavi personalizzate, in luogo di **0** e **1**. Ora potete semplicemente visualizzare i dati con **echo** e quindi specificare la chiave dell'array (come **\$drivers[0]**) ma come potete visualizzare ogni elemento se non

Installare PHP su Linux

La maggior parte delle distribuzioni Linux includono PHP. Anche se potete lanciare script PHP da riga di comando, userete il browser Web (e quindi il server Web) per questo tutorial. Potete seguire la guida caricando i vostri file PHP in un server Web su internet, se ne avete uno; oppure potete utilizzare un'installazione standard di Apache 2 nella vostra macchina Linux locale, cosa che probabilmente è più facile

e veloce di dover caricare ogni volta i file via FTP. Se vi servono istruzioni per installazione o la configurazione nella vostra macchina, puntate il browser all'indirizzo <http://php.net/manual/en/install.unix.php>; in alternativa, esistono centinaia di guide per l'installazione per praticamente qualunque tipologia di Linux: vi basterà una rapida ricerca con Google.

sapete quant'è grande un array? Fortunatamente esiste una semplice funzione di controllo chiamata **foreach()** che vi permette di farlo. Potreste domandarvi perché non conoscete la lunghezza di un array: bene, conoscete esattamente la composizione dei dati per ogni pilota, ma una query su database potrebbe ritornare un solo pilota, o cinque. Potreste calcolare il numero totale degli elementi dell'array con una funzione PHP, ma **foreach()** è più semplice e vi permette di scrivere codice più compatto. Il controllo **foreach()** vi dà modo di iterare su un array. Utilizzando **drivers.php** appena creato, copiate il codice sottostante sotto al blocco che contiene l'array:

```
foreach($drivers as $driver) {
    echo 'Nome: ' . $driver['name'] . '<br />';
    echo 'Nazionalità: ' . $driver['nationality'] .
    '<br />';
    echo 'Campionati vinti: ' . $driver['championships'] . '<br /><br />';
}
```

Visualizzando il file sul vostro browser, vedrete una lista di piloti sullo schermo:

- Nome: Jenson Button
- Nazionalità: British

- Campionati vinti: 1
- Nome: Lewis Hamilton
- Nazionalità: British
- Campionati vinti: 1

A ogni ciclo di **\$drivers**, il valore dell'elemento corrente viene assegnato a **\$driver** (il primo sarà Jenson Button), e il puntatore interno dell'array si sposterà di uno; quindi nel ciclo successivo otterrete l'elemento seguente (Lewis Hamilton). Questo fino

Tip



Utilizzate un editor di testo con l'evidenziazione della sintassi per PHP: vi aiuterà a identificare velocemente blocchi di codice o funzioni al suo interno. Esistono diversi programmi gratuiti, quindi provatene qualcuno e trovate quello che si avvicina di più ai vostri gusti.

```
1 <?php
2
3 is_race_day();
4
5 function is_race_day() {
6     $race_dates = array(
7         '18/3/2012' => array('title' => 'Australian Grand Prix', 'location' => 'Melbourne'),
8         '25/3/2012' => array('title' => 'Malaysia Grand Prix', 'location' => 'Kuala Lumpur'),
9         '15/4/2012' => array('title' => 'Chinese Grand Prix', 'location' => 'Shanghai'),
10        '22/4/2012' => array('title' => 'Bahrain Grand Prix', 'location' => 'Sakhir'),
11        '13/5/2012' => array('title' => 'Spanish Grand Prix', 'location' => 'Catalunya'),
12    );
13
14    $date = date('m/d/Y');
15
16    if(array_key_exists($date, $race_dates)) {
17        echo "Today's race is the " . $race_dates[$date]['title'] . " in " . $race_dates[$date]['location'];
18    }
19    else {
20        echo "There is no race today.";
21    }
22 }
23
24 Running syntax check with PHP 5.3.6 (cli)...
25 No syntax errors detected
```

Utilizzando un editor di codice con il controllo di sintassi integrato (come Eclipse o NetBeans) vi risparmierà tempo e frustrazioni!



Calendari online

Continuando il tutorial introduttivo di due mesi fa, vi facciamo esplorare gli array e le funzioni così da costruire un calendario di eventi personalizzato per il vostro sito Web



Due mesi fa avete appreso le basi del PHP, incluso come fu creato il linguaggio e come successivamente crebbe. Allo stesso tempo, avete avuto una prima infarinatura su alcune parti del linguaggio, come le variabili, le stringhe, gli interi e la funzione interna di PHP **date()**. In questa puntata espanderete queste parti, e vi introdurremo il concetto di array e di funzione per costruire un calendario funzionante. Prima di cominciare assicuratevi di avere la vostra piattaforma Linux configurata per servire pagine PHP attraverso il browser, come spiegato nello scorso numero. Quindi, cos'è esattamente un array? Per definirlo, torniamo alla puntata di due numeri fa dove avete utilizzato una variabile (**\$display_text**) per conservare un messaggio in stringa semplice. Il problema delle variabili è che conservano un solo pezzo di informazione alla volta: non sarebbe magnifico poter registrare valori multipli in una singola variabile? Ecco dove entrano in gioco gli array.

Array

Il miglior modo per pensare un array è come tipo speciale di variabile che contiene altre variabili. Un array vi permette di conservare quanti elementi volete al suo interno (l'unico limite è la memoria a disposizione di PHP). Potete iterare tutti gli elementi al suo interno (traversing) e PHP offre più di 70 funzioni per manipolare i vostri array, come la ricerca, il conteggio degli elementi, la rimozione dei duplicati e addirittura l'inversione dell'ordine degli elementi.

“Un array vi permette di conservare al suo interno quanti elementi volete”

Per crearne uno, il codice è banale:

```
$data = array();
```

Ora avete un array vuoto chiamato **\$data**. Gli array sono strutturati per utilizzare un indice chiave-valore; per default, quando aggiungete elementi a un array vuoto, la posizione del primo elemento sarà **0**, quella del secondo **1** e così via. Potete anche creare array con dati prepopolati (se sapete già cosa dovrà contenere). Per farlo, estendete il codice di prima inserendo il contenuto separato da virgole:

```
$data = array('Rosso','Arancio','Giallo','Verde','Blu');
```

Ed ecco dove entra in gioco il sistema di chiavi.

Il modo in cui PHP interpreta l'array appena definito è:

```
0 = 'Rosso', 1 = 'Arancio', 2 = 'Giallo', 3 = 'Verde', 4 = 'Blu'
```

Come potete vedere, ogni chiave è associata al valore corrispondente; la cosa importante da ricordare è che gli array iniziano sempre dalla chiave **0** e non

dalla **1** come molti pensano: all'inizio è facile da dimenticare.

Array associativi

Gli array hanno anche

la flessibilità di permettervi di specificare le vostre chiavi (in questo caso di definiscono array associativi). Questo aiuta molto se volete registrare un valore su una specifica chiave senza affidarvi a indici automatici. Diciamo che volete registrare i dati di una persona in un array, potete fare una cosa del genere:

```
$persona = array('name' => 'Ciro Mattia Gonano', 'comune' => 'Rovigo', 'età' => 30);
```

Utilizzando l'istruzione di associazione (**=>**) dite a PHP che volete creare una chiave denominata **name** e memorizzarvi **Ciro Mattia Gonano**. Potete memorizzare un qualsiasi tipo di dato in un array, anche altri array. Il modo in cui PHP interpreta il vostro array **persona** è come vi aspettate: **'name' = 'Ciro Mattia Gonano', 'comune' = 'Rovigo', 'età' = 30**

Quando volete usare l'elemento di un array, quello che dovete fare è chiamare l'array e la chiave desiderata:

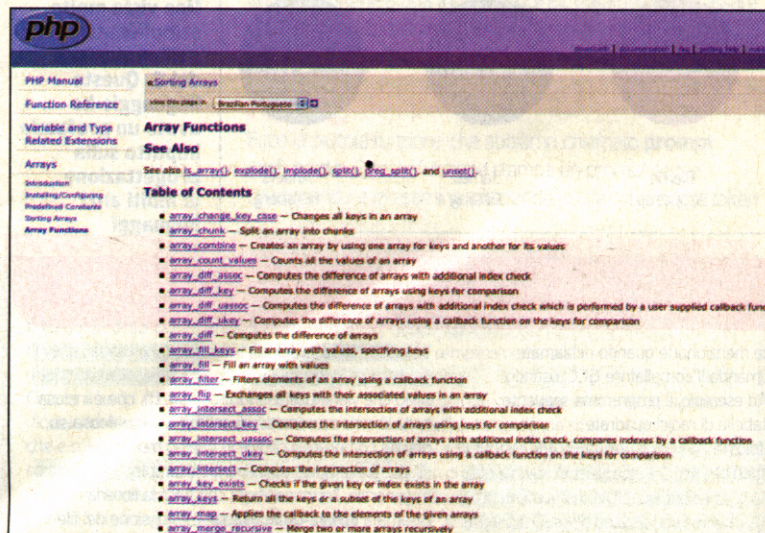
```
echo $data[0];
```

Questo stamperà la stringa **Rosso** a video. Per stampare **Arancio** dovrete semplicemente cambiare la chiave da **0** a **1**. L'array **persona** è altrettanto semplice: per stampare a video il nome sarà sufficiente

```
echo $persona['name'];
```

Aggiungere dati al vostro array

Ora avete definito gli array esistenti, ma volete aggiungerci dati: come fare? Esistono diversi modi per farlo, e spesso dipende dal fatto che gli array



Malgrado tutti i libri scritti sul linguaggio, il sito di PHP è tutt'ora il manuale di riferimento più completo e aggiornato a vostra disposizione

siano associativi o meno. Per aggiungere un elemento a **\$data**, ad esempio, potete usare

```
$data['Indaco'];
```

Utilizzando le parentesi quadre vicino al nome dell'array, PHP riconosce l'azione come un push del valore alla fine dell'array; esiste anche una funzione per farlo:

```
array_push($data, 'Indaco');
```

Questa funzione prende un minimo di due argomenti. Il primo è l'array nel quale volete inserire i dati, e tutti gli elementi successivi verranno inseriti alla fine dell'array. In questo modo potete inserire velocemente elementi multipli in una sola volta:

```
array_push($data, 'Indaco', 'Viola');
```

Se vi serve inserire solamente un elemento, il primo metodo (parentesi quadre) è raccomandato, perché non ha l'overhead di sistema che ha una chiamata a funzione. Se volete aggiungere un altro elemento al vostro array **\$persona**, vi basterà specificare la chiave che volete usare, assieme al valore richiesto: **\$persona['professione'] = 'Sviluppatore';**

Array di array

Come detto prima, un array può contenere qualsiasi tipo di elemento, inclusi altri array. La pratica di array multipli è comune e vi tornerà estremamente utile. Ancora, esistono diversi modi per farlo, e quello che userete sarà in base alla vostra struttura di array. Per questo esempio, immaginate di avere un array di piloti F1 della squadra McLaren. Aprite un editor di testo, inserite il codice PHP seguente e salvatelo

“La pratica di usare array multipli è comune e vi tornerà molto utile”

come **drivers.php**:

```
<?php
$drivers[] = array(
    'name' => 'Jenson Button',
    'nationality' => 'British',
    'championships' => 1,
);
$drivers[] = array(
    'name' => 'Lewis Hamilton',
    'nationality' => 'British',
    'championships' => 1,
);
?>
```

Le parentesi quadre hanno istruito PHP a inserire i dati del pilota alla fine dell'array. Ogni elemento nell'array dichiarato deve essere separato da virgola. Ora il vostro array **\$drivers** contiene due elementi, e ognuno di tali elementi è un array di dati contenenti informazioni sul pilota che volete visualizzare.

Agli occhi di PHP, i dati di Jenson Button sono memorizzati in **\$drivers[0]** mentre quelli di Lewis Hamilton in **\$drivers[1]**. Potete creare anche chiavi personalizzate, in luogo di **0** e **1**. Ora potete semplicemente visualizzare i dati con **echo** e quindi specificare la chiave dell'array (come **\$drivers[0]**) ma come potete visualizzare ogni elemento se non

Installare PHP su Linux

La maggior parte delle distribuzioni Linux includono PHP. Anche se potete lanciare script PHP da riga di comando, userete il browser Web (e quindi il server Web) per questo tutorial. Potete seguire la guida caricando i vostri file PHP in un server Web su internet, se ne avete uno; oppure potete utilizzare un'installazione standard di Apache 2 nella vostra macchina Linux locale, cosa che probabilmente è più facile

e veloce di dover caricare ogni volta i file via FTP. Se vi servono istruzioni per installazione o la configurazione nella vostra macchina, puntate il browser all'indirizzo <http://php.net/manual/en/install.unix.php>; in alternativa, esistono centinaia di guide per l'installazione per praticamente qualunque tipologia di Linux: vi basterà una rapida ricerca con Google.

sapete quant'è grande un array? Fortunatamente esiste una semplice funzione di controllo chiamata **foreach()** che vi permette di farlo. Potreste domandarvi perché non conoscete la lunghezza di un array: bene, conoscete esattamente la composizione dei dati per ogni pilota, ma una query su database potrebbe ritornare un solo pilota, o cinque. Potreste calcolare il numero totale degli elementi dell'array con una funzione PHP, ma **foreach()** è più semplice e vi permette di scrivere codice più compatto. Il controllo **foreach()** vi dà modo di iterare su un array. Utilizzando **drivers.php** appena creato, copiate il codice sottostante sotto al blocco che contiene l'array:

```
foreach($drivers as $driver) {
    echo 'Nome: ' . $driver['name'] . '<br />';
    echo 'Nazionalità: ' . $driver['nationality'] .
    '<br />';
    echo 'Campionati vinti: ' . $driver['championships'] . '<br /><br />';
}
```

Visualizzando il file sul vostro browser, vedrete una lista di piloti sullo schermo:

- Nome: Jenson Button
- Nazionalità: British

- Campionati vinti: 1
- Nome: Lewis Hamilton
- Nazionalità: British
- Campionati vinti: 1

A ogni ciclo di **\$drivers**, il valore dell'elemento corrente viene assegnato a **\$driver** (il primo sarà Jenson Button), e il puntatore interno dell'array si sposterà di uno; quindi nel ciclo successivo otterrete l'elemento seguente (Lewis Hamilton). Questo fino

Tip



Utilizzate un editor di testo con l'evidenziazione della sintassi per PHP: vi aiuterà a identificare velocemente blocchi di codice o funzioni al suo interno. Esistono diversi programmi gratuiti, quindi provatene qualcuno e trovate quello che si avvicina di più ai vostri gusti.

```
1 <?php
2
3 is_race_day();
4
5 function is_race_day() {
6     $race_dates = array(
7         '18/3/2012' => array('title' => 'Australian Grand Prix', 'location' => 'Melbourne'),
8         '25/3/2012' => array('title' => 'Malaysia Grand Prix', 'location' => 'Kuala Lumpur'),
9         '15/4/2012' => array('title' => 'Chinese Grand Prix', 'location' => 'Shanghai'),
10        '22/4/2012' => array('title' => 'Bahrain Grand Prix', 'location' => 'Sakhir'),
11        '13/5/2012' => array('title' => 'Spanish Grand Prix', 'location' => 'Catalunya'),
12    );
13
14    $date = date('m/d/Y');
15
16    if(array_key_exists($date, $race_dates)) {
17        echo "Today's race is the " . $race_dates[$date]['title'] . " in " . $race_dates[$date]['location'];
18    }
19    else {
20        echo "There is no race today.";
21    }
22 }
23
24 Running syntax check with PHP 5.3.6 (cli).
25 No syntax errors detected
```

Utilizzando un editor di codice con il controllo di sintassi integrato (come Eclipse o NetBeans) vi risparmierà tempo e frustrazioni!