Approfondimento su PHP

UniversiBO
www.universibo.unibo.it

9 maggio 2011

Outline

OOP in PHP - Introduzione

PHP in versione 5.3 supporta la programmazione orientata agli oggetti (OOP), fornendo le seguenti caratteristiche:

- classi e interfacce
- gestione delle eccezioni
- namespace
- reflection

Vediamo ora un esempio di classe che in cui utilizziamo:

- metodi
- costruttore
- distruttore
- namespace
- tipe hinting (controllo di tipo)

Esempio di classe 1/2

```
namespace Disney; // dichiarazione di namespace

// Le classi sono tutte pubbliche
// Nome completo della classe: \Disney\Goofy
class Goofy {
    /**
     * @var DonaldDuck
     */
    private $_donaldDuck;

    // costruttore con type hinting
    public function __construct(DonaldDuck $donaldDuck) {
        $this->_donaldDuck = $donaldDuck;
    } ...
```

Esempio di classe 2/2

```
...
    /* distruttore
     * invocato quando l'oggetto viene deallocato */
    public function __destruct() {
        $this->_donaldDuck->sayGoodbyeToDaisy();
    }
    /**
     * Getter per "Paperino"
     * @return DonaldDuck
     */
    public function getDonaldDuck() {
        return $this->_donaldDuck;
    }
}
```

Metodi magici

PHP prevede che un oggetto possa implementare dei metodi detti **magici** che non sono invocati direttamente dallo sviluppatore ma dal motore di PHP. Andremo ad analizzare i più importanti:

- **__toString()**: per la conversione da oggetto a stringa, non è presente un'implementazone di default
- **__get($name)**: gestisce la lettura di un attributo inesistente o inaccessibile
- **__set($name, $value)**: gestisce la scrittura di un attributo inesistente o inaccessibile
- **__call($name, $arguments)**: gestisce la chiamata ad un metodo inesistente o inaccessibile
- **__callStatic($name, $arguments)**: gestisce la chiamata ad un metodo statico inesistente o inaccessibile

Class loading

In PHP il caricamento delle classi è lasciato allo sviluppatore, può essere effettuata in tre modalità:

1. includendo esplicitamente i file di definizione
2. implementando la funzione magica __autoload
3. registrando una propria funzione come autoloader

Nell'esercitazione vedremo le prime due modalità.

Esercitazione: Separazione delle componenti

Come separare le componenti di un'applicazione web? Scaricate il file "Esempio di codice del 2 maggio 2011" da UniversiBO e importatelo nel workspace. Durante l'esercitazione vedremo:

- come separare la parte di presentazione
- come separare la parte relativa ai dati
- come realizzare la funzione __autoload

La View

Per separare la parte di presentazione:

1. sposteremo la nostra parte di presentazione in un file separato
2. definiremo una nostra classe **View**, la quale avrà il compito di contenere le informazioni di presentazione e caricare il file con il contenuto di presentazione
3. vedremo come è possibile accedere ai dati della **View** dal file di presentazione
4. sfrutteremo le caratteristiche di PHP (metodi magici, funzione extract) per utilizzare una sintassi più compatta

Il pattern DAO

Per l'accesso ai dati definiremo una classe **Iscrizione** ed il relativo **DAO**, il quale avrà il compito di svolgere quattro tipi di operazioni:

- **create**: creazione di nuovi record
- **retrieve**: caricamento dei record da database
- **update**: aggiornamento dei record
- **delete**: eliminazione di record

Questo insieme di operazioni è detto in gergo **CRUD**.

L'autoloading

La funzione di __autoload può essere molto semplice, o coinvolgere complessi meccanismi di ottimizzazioni, quali caching,ecc.
Vedremo un esempio semplice di autoloader, nel quale utilizzeremo:

- la direttiva include_once
- le funzioni is_readable() e is_file()
- la funzione str_replace()
- le funzioni class_exists() e interface_exists()

Siete pronti?

Cominciamo!