Incapsulamento dei metodi e delle proprietà

Trattiamo brevemente l'argomento relativo a quella che viene definita la "visibilità".
Fino ad ora abbiamo dichiarato come "public" tutti i metodi e le proprietà delle nostre classi. In effetti esistono altri due livelli di visibilità che sono "protected" e "private".
Public: Con questo attributo dichiariamo i metodi e le proprietà che vogliamo siano visibili anche all'esterno della classe.
Private: Con questo attributo dichiariamo i metodi e le proprietà che vogliamo siano visibili solo all'interno della classe nella quale sono dichiarati.
Protected: Con questo attributo dichiariamo i metodi e le proprietà che vogliamo siano visibili solo nella classe all'interno della quale sono dichiarati e nelle classi figlie.
Facciamo un esempio

```
class Genitore
{
        public $var1;
        protected $var2;
        private $var3;

                public function Metodo1()
                {

                }

                protected function Metodo2()
                {

                }

                private function Metodo3()
                {

                }
}

 class Figlia extends Genitore
{
        //.....
}
```

$var1 é accessibile nella classe Genitore e nella classe Figlia come pure all'esterno della classe dopo che essa é stata istanziata, ad esempio

```php
$obj = new Genitore();
$obj->var1 = 12;
```

$var2 é accessibile unicamente nella classe Genitore e nella classe Figlia. Questo codice:

```php
$obj = new Genitore();
$obj->var2 = 12;
```

Produrrà un Fatal Error.
$var3 é accessibile unicamente nella classe Genitore.
Lo stesso discorso vale per i metodi:

```php
$obj = new Genitore()
$obj->Metodo1();
```

Questo codice funzionerà, ma:

```php
$obj = new Genitore()
$obj->Metodo2();
//oppure
$obj = new Genitore()
$obj->Metodo3();
```

Questi codici produrranno un Fatal Error, Come pure il tentativo di utilizzare o riscrivere Metodo3() nella classe Figlia.

Queste procedure hanno un'utilità per la chiarezza del codice e per evitare pasticci. Se non abbiamo bisogno di accedere ad un metodo o a una proprietà dall'esterno della classe, dichiariamoli come minimo "protected". Questo eviterà spiacevoli intoppi, in particolare quando l'applicazione inizia a crescere o quando ci sono molte persone che maneggiano la classe.

Un'ultima cosa. E' possibile che sia necessario dover utilizzare un metodo o una proprietà all'esterno della classe, ma che contemporaneamente non vogliamo che questa venga riscritta in un'eventuale estensione. Non possiamo dichiararla come private altrimenti non sarebbe più visibile all'esterno. In questo caso ci viene in aiuto l'attributo final. Utlizzandolo impediamo che un metodo venga riscritto anche se dichiarato come pubblico.

```php
public final function Metodo()
{

 }
```

Questo metodo sarà utilizzabile all'esterno della classe ma non riscrivibile in una sua estensione.
Ovviamente una cosa del genere non ha nessun senso

```
private final function Metodo()
{

 }
```