L’ereditarietà

Una delle proprietà più “succose” della programmazione orientata agli oggetti é certamente l’ereditarietà. Con questo termine intendiamo la possibilità di creare una classe “fotocopia” di un’altra classe, ma con la possibilità di aggiungere o modificare parte di essa. Ma passiamo subito alla pratica che é notoriamente più chiarificatrice della teoria.
Prendiamo ad esempio l’ultima classe che abbiamo visto nel capitolo precedente.

```
<?php

 class HelloWorld
{
	public $saluto;

		public function __construct($lang)
		{
			switch($lang)
			{
				case "en":
				$this->saluto = "Hello World";
				break;

				case "it":
				$this->saluto = "Ciao mondo";
				break;
			}

			$this->Saluta();
		}

		public function Saluta()
		{
			echo $this->saluto;
		}
}

 $obj = new HelloWorld("en");

 ?>
```

Come ricorderete questa classe stampa un saluto personalizzato a dipendenza del parametro passato al costruttore. Diciamo ora che in alcuni casi vorrei utilizzare questa classe ma mi

servirebbe che il saluto fosse stampato con un carattere grande. Badate che é un esempio stupido, la vera utilità di quanto stò dicendo la si vede in grandi applicazioni!
Ho due possibilità: Riscrivo la classe formattando l'output nel metodo Saluta(), oppure scrivo un'estensione della classe, in questo modo:

```
<?php

 class HelloWorldSalutoFormattato extends HelloWorld

{

      public function Saluta()

      {

            echo "
<h1>" . $this->saluto . "</h1>

";

      }

}

 ?>
```

Come vedete la sintassi é:
class -nome classe figlia (chid)- extends -nome classe che va estesa (parent)-
Ora in questa nuova classe abbiamo tutti i metodi e le proprietà della classe parent.
Dichiarando in metodo Saluta (che già esiste nella classe parent), indichiamo che vogliamo riscrivere questo metodo.
Ora quando ci servirà il testo del saluto formattato non ci resterà che istanziare la classe figlia:

```
$obj = new HelloWorldSalutoFormattato("en");
```

Oltre ad avere la possibilità di riscrivere metodi e proprietà, possiamo anche dichiararne di nuovi.

```
<?php

 class HelloWorldSalutoFormattato extends HelloWorld

{
```

```
        public $text = "...";

                public function Saluta()

                {

                        echo "
<h1>" . $this->saluto . "</h1>

";

                }

                public function AddText()

                {

                        echo $this->text;

                }

}

?>
```

In questo caso dichiariamo una nuova proprietà ($text) e un nuovo metodo (AddText()).
Ora istanziando la nostra estensione in questo modo

```
$obj = new HelloWorldSalutoFormattato("en");

 $obj->AddText();
```

Quello che ci comparirà a video sarà

Hello World…

Ripeto, l'esempio non rende la potenzialità di questa procedura. Ma provate ad immaginare quello che dicevo nel capitolo introduttivo; la realizzazione di una grande applicazione come ad

esempio un gioco elettronico. La grande utilità della programmazione ad oggetti non può essere colta immediatamente, ma immagino che cominciate ad intuire che non si tratta proprio solo di un’operazione di facciata o un modo un po’ bizzarro di scrivere il codice.