Pensare ad oggetti

PREMESSA: In questa sezione del corso tratterò la programmazione ad oggetti così come supportata nelle versioni 5.x di PHP.

Iniziamo oggi l'affascinante capitolo della programmazione orientata agli oggetti. Chi di voi ne ha sentito parlare ma non ne è mai entrato in contatto avrà diverse aspettative, per cui chiarisco subito alcuni punti.

La programmazione ad oggetti non ci permette di fare di più. Quello che facciamo con il paradigma ad oggetti lo possiamo fare con il paradigma procedurale. Si tratta unicamente di un modo diverso di scrivere il codice, direi che concettualmente è un modo molto diverso. Questo modo diverso di scrivere ed organizzare il codice è più complesso ma ci garantisce una grandissima chiarezza, un'elevatissima estendibilità ed una lunga serie di vantaggi che scoprirete.

Inoltre saremo costretti a spendere più tempo in fase di progettazione, cosa che si rivelerà positiva. Se volessi descrivere la mia giornata odierna con un paradigma procedurale (quello che abbiamo utilizzato fino ad ora) farei qualcosa del genere: Mi sono alzato, ho fatto colazione, ho vestito mio figlio, l'ho portato all'asilo, sono andato a fare la spesa, ho pranzato, ho cenato, ho fatto la doccia, ho guardato la televisione, sono andato a dormire.

Con il paradigma procedurale descriviamo una serie di procedure appunto che si svolgono i sequenza. Se volessi descrivere la mia giornata con un paradigma ad oggetti dovrei fare un ragionamento un po' più ampio. Credo che inizierei con pensare ad un elemento che sia il più generale possibile e che contenga di conseguenza quelle azioni che svolgo certamente ogni giorno. Questo elemento lo chiamerò classe mentre le azioni le chiamerò metodi. Definirò quindi la classe Giornata(), che sarà la classe generale contenente i metodi che rientrano nella casistica generale di tutte o quasi tutte le mie giornate. In questa classe definirò quindi diversi metodi: Mangiare(), Lavarsi(), Alzarsi(), AndareADormire(), Bere(), ecc… Queste sono azioni che svolgo ogni giorno. Potrei domandarmi anche che varianti possono avere queste azioni. Ad esempio Mangiare() cosa? A che ora? Queste varianti le chiamerò proprietà.

Ora a partire da questa classe generale potrò creare delle estensioni (child) più specifiche. Le estensioni della mia classe principale (parent) ereditano tutti i metodi e le proprietà. Potremo inoltre definire dei metodi più specializzati o ridefinire dei metodi esistenti.

Dunque scriverò ad esempio la classe Lunedì() come estensione della classe Giornata() che conterrà il metodo Spesa() e il metodo PortareFiglioAllAsilo() (Cose che faccio solo il lunedì). Questa classe, senza bisogno di riscriverli, conterrà anche i metodi e le proprietà della classe parent (Giornata()). Potrò dunque utilizzare l'estensione Lunedì() per descrivere la mia giornata odierna non dovendomi più preoccupare di dover ridefinire ad esempio il metodo Mangiare().

Pensate ora all'applicazione che abbiamo realizzato nel capitolo precedente. Ogni modulo potrebbe essere un metodo e questi metodi fare parte di una classe. In questa classe avremmo quindi descritte tutte le procedure che servono a far funzionare la nostra applicazione. La possiamo includere in ogni pagina con il vantaggio di tenere separate queste procedure dallo svolgimento logico del codice. Il modulo di cancellazione potrebbe avere ad esempio questo aspetto:

```
Delete($_GET['id']);
header("Location: showall.php");
```

Ovviamente il metodo Delete() conterrà le procedure che si occupano della connessione al database come pure le procedure per la cancellazione del record. Ma questo non ci interessa, a noi interessa unicamente lo svolgimento logico del codice che in questo caso è: Cancella il record con questo id e mandami alla pagina showall.php. Questo è un esempio semplicissimo, ma immaginate un codice molto complesso.
Utilizzando il paradigma ad oggetti questo codice risulterà molto più semplice da modificare sia nella logica che nelle procedure, avrà una struttura chiara e coerente e sarà estendibile. Per questi motivi le grandi applicazioni sono scritte ad oggetti.

Immaginate la realizzazione di un videogioco ad esempio. Vi sarà una classe che gestisce i personaggi in senso generale. Tutti i personaggi muoiono se la loro energia scende sotto il 10, possono saltare, se uccidono un nemico guadagnano 10 punti, se distruggono un edificio guadagnano 50 punti, eccetera. Ognuna di queste caratteristiche sarà gestita da un metodo della classe personaggi(). Ogni programmatore del team avrà poi il compito di sviluppare un personaggio che sarà un'estensione della classe personaggi(). Tutti i personaggi del videogioco avranno quindi una coerenza strutturale derivata dall'aver ereditato le caratteristiche generali dalla classe parent, poi ovviamente ogni personaggio potrà avere le proprie particolarità, il proprio vestito, una mossa segreta particolare, un arma speciale.

Penso che a questo punto cominciate ad intravvedere come la programmazione ad oggetti richieda un maggiore sforzo in fase di progettazione ma infine ci ripaga con un codice più elegante, più semplice da gestire, da modificare e da estendere e, se ben scritto, altamente riutilizzabile.