

## 25/11/08 - Paradigmas y Patrones

Al desarrollar un programa, normalmente, el desarrollador elige un paradigma de programación y algún patrón de diseño, ya sea desarrollado por él, por su forma de trabajo a través de los años de experiencia, o tomado de alguna teoría o grupo de trabajo que lo haya conseguido transmitir.

Los grandes paradigmas de programación han llevado a que existan lenguajes de propósito general orientados únicamente a uno de estos paradigmas, o lenguajes que permiten, de una forma muy flexible, usar uno u otro de estos paradigmas.

Así mismo, los patrones de diseño, en sí, no están ligados, de forma estrecha, a un paradigma concreto, con lo que se pueden llevar a cabo, normalmente, en la mayoría de lenguajes de propósito general.

Los paradigmas de programación más importantes son:

- **Spaguetti**: este tipo de programación está basada en la consecución de mandatos y órdenes en líneas consecutivas, y una serie de etiquetas o numeración de las mismas líneas, que permitan el salto, en cualquier momento, de una parte del código a otra. Lenguajes de este tipo son: Ensambladores, Basic, Logo y muchos tipos de shell script.
- **Funcional o Procedimental**: este tipo de programación se basa en la separación del código en unidades funcionales, normalmente identificadas como verbos o acciones (abrir, cerrar, coge, eliminar, imprime, ...). Los lenguajes que hacen uso de este tipo de paradigma son: PHP, Perl, C, Fortran y Pascal, entre otros.
- **Modular**: es parecido al tipo anterior, solo que las unidades funcionales se agrupan en módulos, bajo un nombre (o jerarquía de nombres), que le permite tener una organización y uso más lógico, así como seguridad y tipos de optimización al poder incluir solo partes o módulos específicos al código. Los lenguajes que hacen uso de la modularización son: PHP (versión 5.3 y superiores), Perl, Modula-2 y C (con los "namespaces"), entre otros.
- **Orientada a Objetos**: este tipo de programación muestra una ideología que se centra en los datos en sí, con una nomenclatura diferente a la funcional, que incluye terminos como: clase, objeto, atributo, método, herencia, etc. Los lenguajes orientados a objetos son: Smalltalk, C++, Java y Python, entre otros.

- **Declarativo:** es otra ideología diferente a la funcional, que se usa, sobre todo, en Inteligencia Artificial. Se basa en decir qué es lo que hay que hacer, y no el cómo. En este tipo de lenguajes encajarían todos los de 4ª generación, como puede ser incluso SQL, pero los lenguajes más comunes son: Prolog y Lisp; otros lenguajes que derivan de estos: Scheme y Clips.
- **Orientado a la Concurrencia:** en un mundo cada vez más paralelo, donde cada proceso de servidor se ejecuta de forma paralela, pero con necesidad de combinación con otros procesos, nace la necesidad de un paradigma de programación que se base en la concurrencia, más que en los demás aspectos. Con esta visión hay lenguajes como: Haskell, Erlang y Scala, entre otros; que permiten hacer concurrencia de forma más fácil.
- **Orientado a Eventos:** este tipo de programación esté íntimamente ligada al lenguaje y entorno en que se programe, ya que los eventos serán dados por el sistema en sí, y su forma de tratarlos y las facilidades que se puedan conseguir vendrán determinadas, en parte, por el lenguaje que se use. Es el caso de la programación de interfaces de usuario con Swing en Java, por ejemplo.

Habiendo elegido el paradigma que más facilite la forma en la que queramos realizar el diseño y codificación, así como el lenguaje, seleccionar un patrón de diseño puede ser el siguiente paso. Patrones hay muchos y sería complicado listarlos todos, pero voy a poner los que más suelo usar y los que he ido aprendiendo... si sabéis alguno más que se me escape, podéis agregarlo como comentario. De todas formas, en la wikipedia se puede tener más información sobre los [Patrones de Diseño](#).

Los tipos de los patrones de diseño son, según la wikipedia:

- **Creacionales:** que son los que se usan para la creación o instanciación de datos, objetos o recursos (Factory, Singleton, ...).
- **Estructurales:** son los que hacen de mediación entre dos partes específicas, como Proxy, que se encarga de agregar seguridad a un código ya programado.
- **Comportamiento:** establece la forma de funcionamiento de una parte de código en sí, dando una capa de abstracción al código para su mejor comprensión.
- **Sistema:** son patrones de diseño más completos que envuelven a

todo el sistema a desarrollar en sí, no solo una parte o un cierto algoritmo. El más característico es el MVC, Modelo-Vista-Controlador, que separa en tres unidades lógicas el desarrollo de la aplicación.

Los patrones de diseño se pueden mezclar entre sí, aunque los de Sistema es más complicado mezclarlos entre sí, sí que pueden ser usados conjuntamente con el resto de patrones, como los creacionales y de estructura, sin problemas.