1 Curso POO Patróns de arquitectura e de deseño

1.1 Patróns de arquitectura e de deseño

No ámbito da programación, os patróns son esquemas ou estruturas que ofrecen solucións a problemas comúns que xorden no desenvolvemento de aplicacións.

Podemos diferenciar polo menos dous tipos de patróns:

- **Patróns de arquitectura**: ofrecen xeitos estándar de organizar os distintos compoñentes dunha aplicación.
- **Patróns de deseño**: ofrecen solucións a problemas concretos en certas estructuras.

1.1.1 Patróns de arquitectura

No ámbito da Programación Orientada a Obxectos, un dos patróns de arquitectura máis nomeados hoxe en día é o **Modelo-Vista-Controlador** (**MVC**). Como o seu nome indica, componse de:

- **Modelo**. Representa a información do sistema e implementa a lóxica de negocio. Recibe peticións de acción do Controlador e envía á Vista, directamente ou a través do Controlador, a información que é preciso amosar.
- **Vista**. Presenta a información necesaria ao usuario, co que interactúa e do cal recibe instruccións.
- **Controlador**. Recibe da Vista as accións do usuario, e envía peticións ao Modelo para modificar a información do sistema e actualizar a Vista.

Pódense distinguir dúas variantes MVC: activo e pasivo. Nun **MVC activo** a Vista, que representa ao interface do usuario, recibe eventos que emprega para actualizar a súa información. Nun **MVC pasivo**, a Vista non é notificada dos cambios nos datos. Cando falamos de MVC en aplicacións web, sempre se trata de MVC pasivo, debido ás limitacións das páxinas web para recibir eventos do servidor.

Outro patrón, variante de MVC, empregado no desenrolo de aplicacións web é o **Modelo-Vista-Presentador** (**MVP**). Neste patrón, a separación de capas é máis clara:

- **Modelo**. Representa a información do sistema. Realiza sobre a información as operacións indicadas polo Presentador, e devólvelle os datos que solicita.
- **Vista**. Representa á interface do usuario. Envía ao Presentador os eventos resultantes da interacción do usuario, e recibe deste a información necesaria para actualizar o interface.
- **Presentador**. Implementa a lóxica de negocio. Comunícase coa Vista para recibir os eventos do usuario, indica ao Modelo a información a modificar ou recuperar, e envía os cambios resultantes á Vista para que se actualice.

1.1.2 Patróns de deseño

Existen moitos patróns de deseño. Normalmente agrúpanse en tres categorías, dependendo do seu obxectivo:

- **Patróns estructurais**. Definen composicións de clases e obxectos para compoñer estructuras maiores.
- **Patróns creacionais**. Ofrecen mecanismos para crear, compoñer e representar obxectos.
- **Patróns de comportamento**. Definen formas de implementar algoritmos e de establecer comunicacións entre obxectos.

Por exemplo, dentro dos patróns creacionais atópase o patrón **Singleton**, que ofrece un xeito de limitar a instanciación dunha clase a un único obxecto. A idea é facer o constructor privado e controlar a instanciacións de obxectos mediante un método da propia clase.

```
<?php
class Singleton
{
    protected static $instancia = null;
    protected function __construct() {}
    protected function __clone() {}

    public static function instancia()
    {
        if (!isset(self::$instancia)) self::$instancia = new Singleton();
        return self::$instancia;
    }
}?>
```

--Víctor Lourido 03:33 16 ago 2013 (CEST)