Madrid, 19 de Mayo de 2015
Universidad Complutense de Madrid
Grado de Ingeniería de Computadores
Asignatura: Sistemas Web

VEERY docs

www.veery.es

Desarrollo:
Alberto Miedes Garcés
Javier Bermúdez Blanco
Daniel Pinto Rivero
Rafael Ramirez Urbina
Ignacio María Ferreras Astorqui
Denys Sypko

Descripcion de la Arquitectura

Para generar las vistas al usuario, hemos decidido aplicar nuestros conocimientos adquiridos previamente en TP (Tecnología de la Programación) e IS (Ingeniería de Software) y trabajar con un patrón de arquitectura con el que nos sintiéramos familiarizados, por lo que decidimos utilizar el patrón de Modelo-Vista-Controlador (MVC), que además es bastante predominante en los frameworks dispònibles a nivel de programación web (Rails en Ruby, Laravel y Symfony en PHP, Angular.JS y Express.JS en JavaScript, por mencionar algunas.)

Aunque ya conocemos el patrón MVC, comentaremos brevemente cómo actúa en una aplicación web como la que hemos desarrollado.

Modelo

El modelo se encarga de manejar el estado de la aplicación. Con “estado” nos referimos al concepto de nuestra aplicación. El modelo no conoce nada sobre el HTML, solo provee los métodos para consultar el estado y modificar el estado.

Vista

La vista es la representación de la interfaz de usuario. Una vista puede consultar al modelo, pero no puede cambiar el estado de la aplicación. Una vista se implementa utilizando plantillas que representa el HTML de la página web.

Controlador

Las acciones de los usuarios en la Vista, se envían al Controlador. En un espacio web, esto es posible al permitir que el Controlador maneje las peticiones HTTP entrantes. El Controlador recibe las peticiones del usuario, y las traduce en acciones que el Modelo debe de tomar. Luego selecciona la Vista apropiada para representar la respuesta.

Las ventajas que observamos al elegir este patrón de arquitectura son las siguientes:

Separacion de peticiones y paginas

Gracias a que el Controlador se encuentra a cargo de manejar las pecitiones y seleccionar las Vistas apropiadas, no existe emparejamiento entre una peticion realizada por el usuario y una página de respuesta.

Vistas ilogicas

Debido a que las vistas solo contienen lo necesario para mostrar una página bonita al usuario, toda la lógica se encuentra en objetos ajenos a la Vista. Cambiar la lógica, no requiere tocar la disposición ni el diseño de nuestra aplicación. Y análogamente, cambiar el diseño o la disposición, no involucra cambiar la lógica. De esta forma, al hacer un rediseño de nuestra exitosa aplicación web, no será necesario modificar la lógica, haciéndola más escalable y menos “error/bug-prone”.

La logica es ciega, como el amor

Como todas las acciones que corresponden al estado de nuestra aplicación son manejadas por el Modelo, es posible hacer cambios de implementación sin tocar la interfaz de usuario.

Por otro lado, algunas limitaciones que encontramos con este patrón son los siguientes:

A donde va esto

En ocasiones puede resultar dificil comprender donde una parte especifica de la aplicación debe de ir y su finalidad. El patrón MVC, en comparación con los ejemplos básicos establecidos en clase, resulta extremadamente complejo.

Mucha clase. Mucho objeto.

Siguiendo con la linea de pensamiento de la limitación anterior, desarrollar una aplicación web siguiendo el patrón de MVC resultará en un número mayor de clases y objetos en comparación con una página con un único archivo PHP “monolítico”. Eso quiere decir que al generar la base, se tendrá que realizar mucho mas trabajo. Tardaremos mas en ver los resultados.