Milián Rojas Edgar Enrique MR180696

MCV

Modelo Vista-Controlador

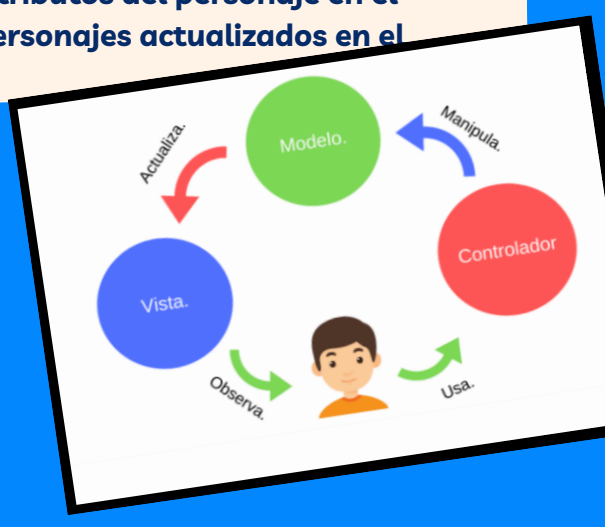
El "Modelo vista-controlador" (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes separados.

Es un modelo muy maduro que ha demostrado su vigencia a lo largo de los años en todo tipo de aplicaciones, y en multitud de lenguajes y plataformas de desarrollo.

ESTRUCTURA MVC

El modelo MVC se rige por la siguiente estructura

- **Modelo: solo contiene los mejores datos de la aplicación, no contiene técnicas que describan cómo se presentan los datos al usuario.**
- **Vista: Presenta los datos de modelos personalizados. Los espectadores saben cómo acceder a los datos del modelo, pero se desconoce el significado de esta información o qué puede hacer el usuario para manipularla.**
- **Controlador: Ubicado entre la pantalla y el modelo. Escuche el evento que se está ejecutando en la pantalla y siga los pasos apropiados para el evento. Por ejemplo, el conductor puede actualizar el modelo cambiando los atributos del personaje en el videojuego o cambiando la pantalla que muestra los personajes actualizados en el juego.**



CONFIGURAR RUTAS AMIGABLES

1. En primer lugar, tienes que modificar la tabla en la que se guarda el artículo.
2. Implementamos las URL amigables, podemos hacerlo a mano desde el código o podemos implementar una función para que lo haga automáticamente basándonos en el titulo del artículo.
3. Cambiamos el archivo .htaccess ya que será el encargado de redirigir la nueva URL al contenido que se guarda en la base de datos.
4. Cambiamos el index

```
function seo_url($vp_string){
    $vp_string = trim($vp_string);
    $vp_string = html_entity_decode($vp_string);
    $vp_string = strip_tags($vp_string);
    $vp_string = strtolower($vp_string);
    $vp_string = preg_replace('~[^ a-z0-9_.]~', ' ', $vp_string);
    $vp_string = preg_replace('~ ~', '-', $vp_string);
    $vp_string = preg_replace('~-+~', '-', $vp_string);
    $vp_string .= "/";
    return $vp_string;
}
}
```

Función para implementar las URL amigables

ENVIO DE DATOS A CONTROLADORES

Hay dos formas principales de lograr esto: definir un Modelo específico fuertemente tipado y luego pasárselo a la Vista, o usar los contenedores ViewData/ViewBag para volver disponibles los datos a la Vista.

Puedes pasar datos de un Controller a la Vista usando diferentes enfoques, pero el recomendado es casi siempre la vista con tipado riguroso, donde defines el tipo de Model que tu vista espera. El uso de ViewData/ViewBag son alternativas para el tipado riguroso

ENVIO DE DATOS A LOS MODELOS

lo que haremos será serializar el formulario completo dentro de la propiedad «data» de la acción «Ajax» con lo que estaremos enviando todos los campos del formulario y, una vez en la acción del controlador, gracias al ModelBinding, el parámetro del modelo, en este caso «Car car» se rellenará de forma automática con todos los valores que procedan, los validamos con el ModelState y hemos acabado. Esto es totalmente flexible a cambios y, por lo tanto, no nos dará dolores de cabeza cuando tengamos que hacer modificaciones en el formulario.

ENVIO DE CONTROLADORES Y VISTAS

Separar nuestra lógica de controlador de nuestra representación de vista trae varios grandes beneficios. En particular, ayuda a aplicar una clara "separación de preocupaciones" entre el código de la aplicación y el formato de la interfaz de usuario/código de representación. Esto hace que sea mucho más fácil probar la lógica de la aplicación de prueba unitaria de forma aislada de la lógica de representación de la interfaz de usuario. Facilita la modificación posterior de las plantillas de representación de la interfaz de usuario sin tener que realizar cambios en el código de la aplicación. Y puede facilitar que los desarrolladores y diseñadores colaboren juntos en proyectos. Podemos actualizar nuestra clase DinnersController para indicar que queremos usar una plantilla de vista para devolver una respuesta de interfaz de usuario HTML cambiando las firmas de método de nuestros dos métodos de acción de tener un tipo de valor devuelto de "void" para tener en su lugar un tipo de valor devuelto de "ActionResult". A continuación, podemos llamar al método auxiliar View() en la clase base Controller para devolver un objeto "ViewResult"

Escenario:
Supongamos que tenemos un formulario en una vista de una aplicación web realizada con ASP.MVC y necesitamos enviar los datos, que corresponden a un modelo de la aplicación, a nuestra acción del controlador.

```
[HttpPost]
public JsonResult Create(Car car) {
    try {
        if (!ModelState.IsValid) return Json(false);
        car.Status = «Solicitados»;
        return Json(true);
    } catch {
        return Json(false);
    }
} |
}
```

CONTROLADOR

```
[HttpPost]
public JsonResult Create(Car car) {
    try {
        if (!ModelState.IsValid) return Json(false);
        car.Status = «Solicitados»;
        return Json(true);
    } catch {
        return Json(false);
    }
} |
}
```

```
[HttpPost]
public JsonResult Create(Car car) {
    try {
        if (!ModelState.IsValid) return Json(false);
        car.Status = «Solicitados»;
        return Json(true);
    } catch {
        return Json(false);
    }
} |
}
```

CÓDIGO JAVASCRIPT

ENVIO DE DATOS A LAS VISTAS

Método ViewData: El primero de los métodos es el denominado ViewData. ViewData es un diccionario de clave (una cadena) - valor (un objeto) que el controlador pasa a la vista.

Para mostrar los datos en la vista, simplemente usamos la propiedad ViewData que tienen las vistas y que funciona exactamente igual.

El uso de ViewData tiene dos puntos débiles que deben tenerse presentes:

1. *Las claves son cadenas, por lo que si nos equivocamos el compilador no puede ayudarnos. Tampoco herramientas de refactoring pueden darnos soporte.*
2. *ViewData["clave"] siempre devuelve un object por lo que debemos ir haciendo casting si queremos obtener el tipo real de lo que hay almacenado.*

Método ViewBag:
ViewBag funciona de forma muy parecida a ViewData. Al igual que ViewData, ViewBag es un diccionario de clave - valor. Pero se aprovecha de las capacidades de programación dinámica que ofrece C#, para en lugar de usar cadenas para especificar la clave, usar propiedades.

Método Model:
El tercer método para pasar información de una acción de un controlador a una vista, es usando la propiedad Model. La propiedad Model no funciona como las anteriores, sino que lo que se pasa es un objeto, que se manda de la acción hacia la vista.
A diferencia de ViewData y ViewBag que existen tanto en el controlador como en la vista, el controlador no tiene una propiedad Model. En su lugar se usa una sobrecarga del método View() y se manda el objeto como parámetro.
El controlador crea un objeto de la clase Usuario y manda ese objeto a la vista, pasándolo como parámetro a la función View.
¿Y desde la vista? Pues usamos la propiedad Model para acceder a dicho objeto

Usar este mecanismo es la manera preferida de pasar datos desde una acción a una vista (ya que en lugar de tener datos desperdigados en n claves los podemos tener organizados en una clase). A las clases que se pasan desde las acciones a las vistas (como nuestra clase Usuario) se les conoce también con el nombre de ViewModels (para distinguirlas de las clases que conforman el Modelo del patrón MVC, ya que los ViewModels lo único que hacen es contener datos que mostrará una vista).

Bibliografía:
-Alvarez, M. A. (2020). *¿Qué es MVC?* (p. https://desarrolloweb.com/articulos/que-es-mvc.html). Miguel Angel Alvarez. Miguel Angel Alvarez.

-García, V. (2019). Patrón de diseño MVC (p. https://blog.nearsoftjobs.com/patrón-de-diseño-mvc-2366948b5fc7). Valeria García Cobian. Valeria García Cobian.

-Rubén Andrés. (2018). Qué es .htaccess, para qué sirve y códigos fundamentales (p. https://computerhoy.com/noticias/internet/que-es-htaccess-que-sirve-codigos-fundamentales-76211).

-Porras Rodriguez, S. (2015). ASP.NET MVC | Cómo enviar modelo con JavaScript desde un formulario a una acción (p. https://geeks.ms/santypr/2015/08/19/asp-net-mvc-cmo-enviar-modelo-con-javascript-desde-un-formulario-a-una-accin/). Santiago Porras Rodriguez. Santiago Porras Rodriguez.

-Tomás, E. (2011). Pasar datos de los controladores a las vistas (p. https://desarrolloweb.com/articulos/pasar-datos-controladores-vistas-dotnet.html). Eduard Tomás. Eduard Tomás.