BOLETÍN DE EJERCICIOS TEMA 6

PROFESOR: MARTÍN GARCÍA FIGUEIRA
martin@ciclosmontecastelo.com

AVISO: Para estos ejercicios debes importar el script Script_Ejercicios_Tema6

1. **Ejercicio 1: Sistema de routing**
    - Crea una aplicación web que tenga dos controladores diferentes: "Home" y "Products", cada uno con dos acciones diferentes.
    - Define rutas distintas para cada acción y comprueba que puedas acceder a ellas mediante la barra de direcciones del navegador.
    - Las acciones que debes definir son:
        - "Index", para el controlador "Home", con la ruta "/Home/Index"
        - "List", para el controlador "Products" con la ruta "/Products/List".

2. **Ejercicio 2: Attribute routing**
    - Modifica el ejercicio 1 para utilizar attribute routing.
    - Define rutas para cada acción utilizando atributos en lugar de la convención de nomenclatura por defecto.
    - Por ejemplo, debes agregar el atributo "[Route("home/index")]" a la acción "Index" del controlador "Home" para definir la ruta "/home/index" para esa acción.

3. **Ejercicio 3: Atributos de selección de acción**
    - Crea una acción que utilice el atributo [**HttpPost**] y otra que utilice el atributo [**HttpGet**].
    - Crea un formulario HTML que envíe datos a la acción con el atributo [HttpPost] y comprueba que los datos se envíen correctamente.

- Debes crear un formulario de inicio de sesión que envíe un nombre de usuario y una contraseña a una acción con el atributo [HttpPost] que verifique las credenciales y redirija al usuario a la página de inicio si son válidas.

4. Ejercicio 4: Acciones con parámetros simples

- Crea una acción que reciba un parámetro de cadena y lo muestre en una vista.
- Define una ruta para esta acción que incluya un parámetro de cadena en la URL.
- Debes crear una acción "Details" en el controlador "Products" que reciba el ID de un producto como parámetro de cadena y muestre los detalles de ese producto en una vista. La ruta para esta acción podría ser "/Products/Details/{id}".

5. Ejercicio 5: Acciones con modelos complejos

- Crea una clase de modelo con varias propiedades y una acción que reciba ese modelo y lo muestre en una vista.
- Crea un formulario HTML que envíe datos al modelo y comprueba que los datos se muestren correctamente en la vista.
- Debes crear una clase de modelo "Order" con propiedades para el nombre y la dirección del cliente, los detalles del pedido y el método de pago.
- Debes crear una acción "OrderDetails" en el controlador "Orders" que reciba un objeto "Order" como parámetro y muestre los detalles del pedido en una vista.
- Finalmente, crea un formulario de pedido que envíe los datos de un objeto "Order" a la acción "OrderDetails".

6. Ejercicio 6: Binding

- Crea un formulario HTML que envíe datos a una acción mediante el método POST.
- Define un modelo de datos que represente los datos enviados por el formulario y utiliza el binding de ASP.NET Core MVC para enlazar los datos a ese modelo.
- Guarda los datos en una base de datos y comprueba que se hayan guardado correctamente.
- Debes crear un formulario de contacto que envíe el nombre, la dirección de correo electrónico y el mensaje del usuario a una acción "Contact" en el controlador "Home".
- También debes crear un modelo de datos "ContactForm" con propiedades para esos campos y utilizar el binding para enlazar los datos del formulario a un objeto "ContactForm".
- Finalmente, en la acción "Contact" debes guardar los datos del objeto "ContactForm" en una base de datos y mostrar un mensaje de confirmación al usuario en una vista.

7. Ejercicio 7: ModelState

- Modifica el ejercicio 6 para validar los datos enviados por el formulario utilizando el objeto "ModelState".
- Agrega restricciones a los campos del formulario y comprueba que se muestren mensajes de error si se ingresan datos inválidos.
- Debes agregar una restricción para que el campo de correo electrónico sea obligatorio y tenga un formato válido utilizando el atributo [Required] y el atributo [EmailAddress] respectivamente.

8. Ejercicio 8: Variables de sesión

- Crea una aplicación web que tenga una página de inicio de sesión y una página de perfil.
- Cuando el usuario inicia sesión, guarda su nombre de usuario en una variable de sesión y redirige al usuario a la página de perfil.
- En la página de perfil, muestra el nombre de usuario almacenado en la variable de sesión.
- Comprueba que el nombre de usuario se mantenga incluso después de cerrar y volver a abrir el navegador.
- Debes utilizar la clase "HttpContext.Session" para almacenar el nombre de usuario en una variable de sesión y recuperarlo en la página de perfil.

9. Ejercicio 10: Variables de aplicación y caché

- Agrega un sistema de caché a una acción que tenga un tiempo de ejecución prolongado.
- Crea una acción que calcule un valor complejo (por ejemplo, un número Fibonacci grande) y guárdalo en la cache utilizando la clase "IMemoryCache".
- Luego, agrega un atributo [ResponseCache] a la acción para indicar que se debe utilizar la cache para esta acción. Comprueba que la cache se esté utilizando correctamente y que los tiempos de respuesta sean más rápidos después de la primera vez que se ejecute la acción.
- Debes crear una acción "Fibonacci" en el controlador "Home" que calcule el número Fibonacci de un número dado y lo almacene en la cache.
- Luego, agrega el atributo [ResponseCache(Duration = 60)] a la acción para indicar que la respuesta se debe almacenar en caché durante 60 segundos.