Capítulo 4: Creando Aplicaciones ASP.NET MVC

Capítulo 5: Aplicando Técnicas en una Aplicación ASP.NET MVC



Creando Aplicaciones ASP.NET MVC 6

Visual Studio 2017 Web Developer



Objetivos

Al finalizar el capítulo, el alumno:

- Aplicar correctamente las plantillas que proporciona el Visual Studio 2017 para la creación de aplicaciones ASP.NET MVC.
- Implementar modelos.
- Implementar controladores.
- Implementar vistas.

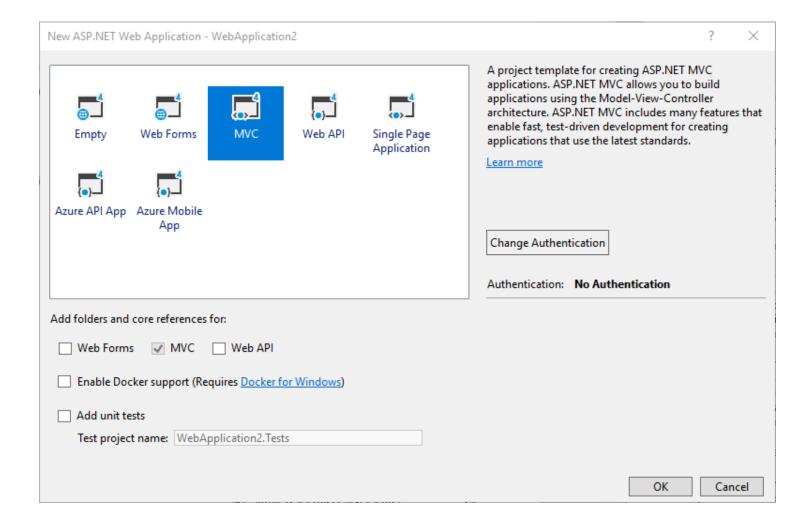


Agenda

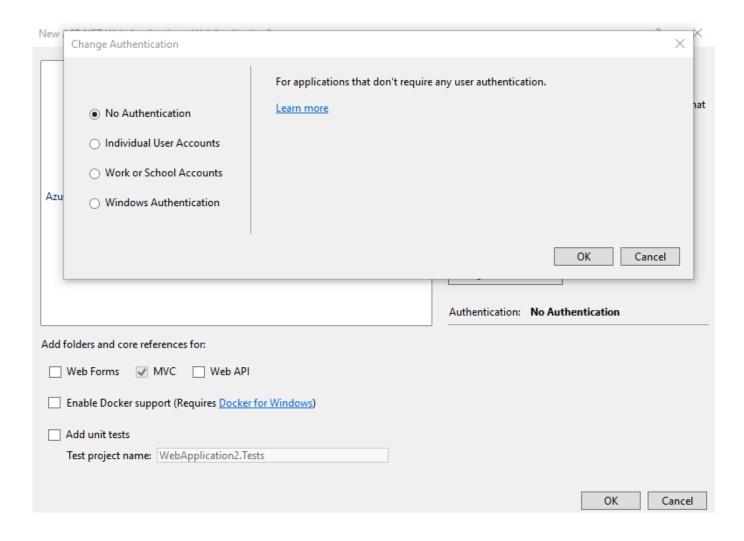
- Plantillas del Visual Studio 2017
- Convenciones de nombres MVC
- Modelos
 - Creación del modelo con Entity Framework (Code First y Database First)
 - Data Annotations
- Controladores
 - Actions y sus tipos
- Vistas
 - Razor Engine
- Vistas parciales



Exploración de las plantillas de Visual Studio 2017



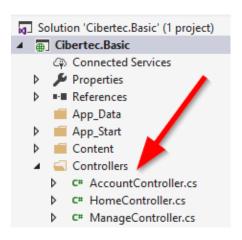
Exploración de las plantillas de Visual Studio 2017



Convenciones de nombres MVC

- Controlador (controller)
 - Prefijo Controller
 - En la carpeta Controllers
- Vistas (view)
 - Carpeta Views

```
▼ P Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test R Tools Analyze Window Help
 💿 - 🔘 👸 - 监 🔛 🍄 🤟 - 🧠 - Debug - Any CPU
                                                       - ▶ Google Chrome - 💍 - 🎜 🛒 🔚 😘 🍱 📜 게 게 게 다
                                                                                                                  0048-0-500
                    ViewBag.Title = "Home Page";
                                                                                                                   <h1>ASP.NET</h1>
                    ASP.NET is a free web framework for building great Web sites and Web ap
                                                                                                                     ▶ ■■ References
                    <a href="https://asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a>
                                                                                                                       App_Data
                                                                                                                    ▶ ■ App_Start
              cdiv class="row">
cdiv class="col-md-4">
                                                                                                                       ▶ C* AccountController.c
                        <h2>Getting started</h2>
                                                                                                                       ▶ C* HomeController.cs
                                                                                                                       ▶ Cª ManageController.cs
                           ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites
                           enables a clean separation of concerns and gives you full control over markup
                            for enjoyable, agile development.
                                                                                                                     ▶ ■ Scripts
                                                                                                                       (9) About.cshtml
(9) Contact.cshtm
                       <h2>Get more libraries</h2>
                                                                                                                         (@) Indexcshtml
                        NuGet is a free Visual Studio extension that makes it easy to add, remove, and u
                                                                                                                       ▶ ■ Manage
▶ ■ Shared
                        <a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkId=301866</p>
                   <div class="col-md-4">
                                                                                                                          (9) ViewStart.cshtml
                                                                                                                          ₩ Web.config
                                                                                                                    ▶ ♠ ApplicationInsights.config
♠ favicon.ico
                        (p)You can easily find a web hosting company that offers the right mix of features
                        <a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkId=301867</p>
                                                                                                                       packages.config
                                                                                                                    b C<sup>a</sup> Startup.cs
b ♥ Web.config
         d Window Code Coverage Results Output Web Publish Activity Package Manager Console Error List .
```





Modelos

- Conjunto de clases (entidades) que representan el dominio de la aplicación.
- Debe representar lógica de negocio, validaciones y acceso a datos.

```
Photo
                                                               Comment
  -PhotoID : int
                                                             CommentID : int
  -Title : string
                                                             User : string
  -PhotoFile : byte
                                                             Subject : string
  -Description : string
                                                             Body: string
  -CreatedDate : object
                                                             PhotoID: int
  -Owner : string
public class Photo
  public int PhotoID { get; set; }
  public string Title { get; set; }
  public byte[] PhotoFile { get; set; }
  public string Description { get: set: }
  public DateTime CreatedDate { get; set; }
  public string Owner { get; set; }
  public virtual ICollection < Comment>
    Comments { get; set; }
```

Personalizar los modelos usando atributos y validaciones

Hacer uso del NameSpaces:
 System.ComponentModel.DataAnnotations.

```
public class Movie {
   public int ID { get; set; }

   [Required]
   public string Title { get; set; }

   [DataType(DataType.Date)]
   public DateTime ReleaseDate { get; set; }

   [Required]
   public string Genre { get; set; }

   [Range(1, 100)]
   [DataType(DataType.Currency)]
   public decimal Price { get; set; }

   [StringLength(5)]
   public string Rating { get; set; }
}
```

Personalizar los modelos usando atributos y validaciones

Atributo	Descripción
Display	Es el texto o etiqueta que aparece en la a lado de control. Ejemplo: [Displax(Name = "Email")]
DataType	El tipo de dato se utiliza para renderizar el control adecuado en la vista. Ejemplo control de tipo número, teléfono, password, etc.
	[DataType(DataType.Password)] [DataType(DataType.DateTime)]
	[sassiges(sassigesteecetene/]
Required	Es la etiqueda que indica que el valor de campo es obligatorio. [Required]
EmailAddress	Valida un dirección de email. [EmailAddress]
StringLength	Indica el tamaño máximo y mínimo de caracteres en un campo.
	<pre>[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]</pre>

Personalizar los modelos usando atributos y validaciones

Compare	Permite comparar dos cadenas de datos.
	<pre>[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]</pre>
	El primer parámetro de la anotación Password representa la propiedad de la clase con la cual se desea comparar.
Range	Permite indica el mínimo y máximo valor para un tipo de dato numérico.
	[Range(0, 99999)]
RegularExpression	Permite configurar una expresión regular para una propiedad de la clase.
	[RegularExpression(@"^[a-zA-Z''- '\s]{1,40}\$", ErrorMessage = "Numbers and special characters are not allowed in the name.")]



Introducción e implementación de Controladores

- El controlador entonces responde a las acciones del usuario, carga la información de un modelo y lo pasa a la vista que finalmente es renderizada como una página web.
- Se hereda de una clase base Controller se encuentra en el namespace Microsoft.AspNet.Mvc

```
5 references
public class HomeController : Controller
{
    4 references
    public IActionResult Index()
    {
        return View();
    }

    O references
    public IActionResult About()
    {
        ViewData["Message"] = "Your application description page.";
        return View();
    }
}
```

Introducción e implementación de Controladores

- Los métodos que exponen los controladores se les conoce como Acciones o Actions.
- El controlador y las acciones ambos forman un endpoint web.

http://localhost:<puerto>/<CONTROLADOR>/<ACCION>

http://localhost:<puerto>/Home/Index http://localhost:<puerto>/Home/About



Controladores – Tipos de Acciones

- Los métodos que exponen los controladores se les conoce como Acciones o Actions.
- El controlador y las acciones ambos forman un endpoint web.

Tipo de acción	Descripción
PartialViewResult	Son usadas para mostrar partes de la aplicación web, como por ejemplo mostrar el nombre del usuario que ingreso al sistema.
RedirectResult	Se usa esta acción para redireccionar a una url específica.
RedirectToAction	Se usa para redireccionar a una acción específica.
JsonResult	Se usa para devolver información al browser en formato JSON.



Introducción e implementación de Vistas

- La vista es la respuesta que da el controlador en forma de HTML y que contiene los datos del modelo.
- Están ubicadas en el fólder Views.
- ¿Qué extensiones tienen? .cshtml o .vbhtml dependiendo del lenguaje de programación, si es C# la extensión es cshtml, si es Visual Basic la extensión es vbhtml.
- ¿Qué lenguaje se utiliza? HTML a través del pre compilador de HTML llamado Razor.



Vista: layout.cshtml

- Este archivo incluye todo el marco y plantilla HTML, es decir, tags como <head>.
- Nuevo tag
 "environment" el cual
 permite, según el
 despliegue que se
 realice, agregar unos
 u otros archivos.
- @RenderBody(): permite renderizar las vistas.

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
   @Styles.Render("~/Content/css")
   @Scripts.Render("~/bundles/modernizr")
<body>
   <div class="navbar navbar-inverse navbar-fixed-top">
       <div class="container">
           <div class="navbar-header">
              <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
               @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
           <div class="navbar-collapse collapse">
               @Html.ActionLink("Home", "Index", "Home")
                   @Html.ActionLink("About", "About", "Home")
                   aHtml.ActionLink("Contact", "Contact", "Home")
              @Html.Partial("_LoginPartial")
          </div>
       </div>
    <div class="container body-content">
       @RenderBody()
           © @DateTime.Now.Year - My ASP.NET Application
       </footer>
   </div>
   @Scripts.Render("~/bundles/jquery")
   @Scripts.Render("~/bundles/bootstrap")
   @RenderSection("scripts", required: false)
</html>
```



Vista: _ViewStart.cshtml

 Mediante esta vista se puede indicar cuál es el Layout por defecto en la aplicación.

```
Permiten ejecutar lógica, por ejemplo

if(DateTime.Now > DateTime.Now.AddDays(1))
{

iRenderSection("scripts", required: false): Permite renderizar scripts que las vista secundarias puedan usar.
```

Ejercicio Nº 4.1: Crear un proyecto en MVC

Crear de un proyecto MVC

Al finalizar el laboratorio, el alumno logrará:

 Familiarizarse con las plantillas existentes para la creación de proyectos MVC.

Ejercicio Nº 4.2: Implementa Patrón Repositorio y Unit Of Work con Dapper.

Implementar el patrón repositorio y unit of work, haciendo el uso del Micro ORM Dapper.

Al finalizar el laboratorio, el alumno logrará:

Diseñar e implementar el patrón repositorio y unit of work.



Ejercicio Nº 4.3: Crear el Controller para Customer

Crea un nuevo MVC Controller para administrar la tabla Customer.

Al finalizar el laboratorio, el alumno logrará:

Conocer el como crear controladores con MVC.



Ejercicio Nº 4.4: Crear View para Customer

Basado en con Customer Controller, procede a crear la Vista que muestre los resultados devueltos por nuestro repositorio.

Al finalizar el laboratorio, el alumno logrará:

Hacer uso del scaffolding para crear Vistas.

Tarea Nº 4.1: Completar los repositorios

Agrega los repositorios faltantes:

- OrderItemRepository
- OrderRepository
- ProductRepository
- SupplierRepository
- UserRepository

