

Capítulo 3: JavaScript, jQuery, JSON y Ajax

Capítulo 4: Creando Aplicaciones ASP.NET MVC

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

4

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.

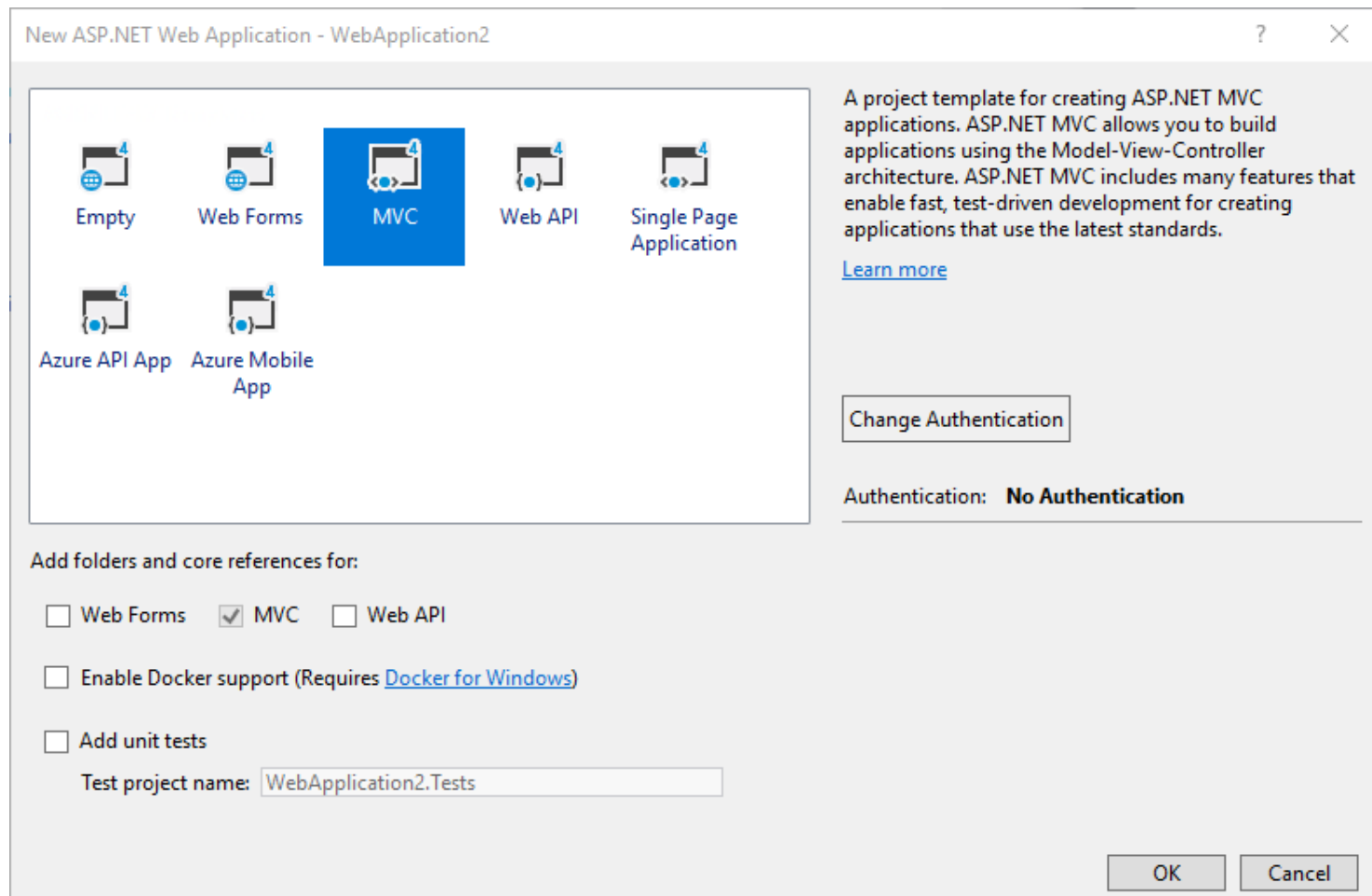


Agenda

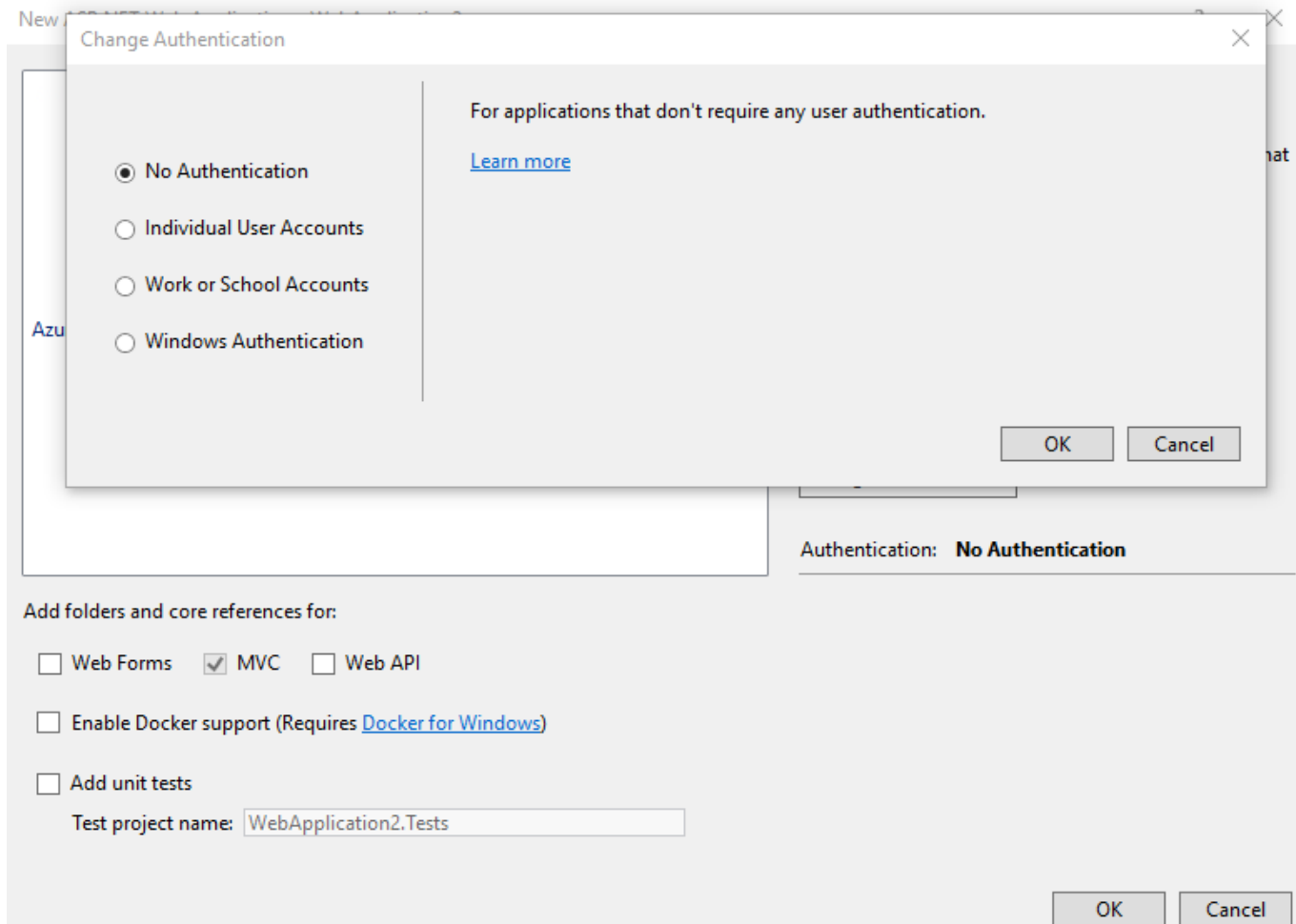
- Plantillas del Visual Studio 2017
- Convenciones de nombres MVC
- Modelos
 - Creación del modelo con Entity Framework (Code First y Database First)
 - Data Annotations
- Controladores
 - Actions y sus tipos
- Vistas
 - Razor Engine
- Vistas parciales



Exploración de las plantillas de Visual Studio 2017

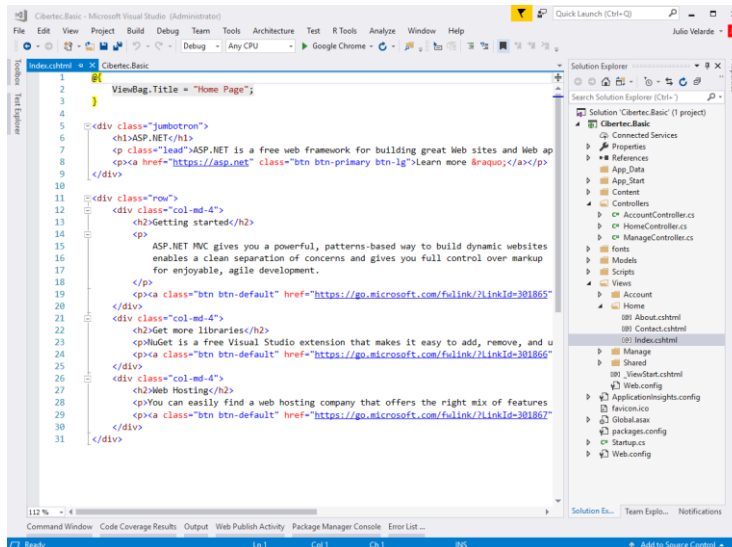
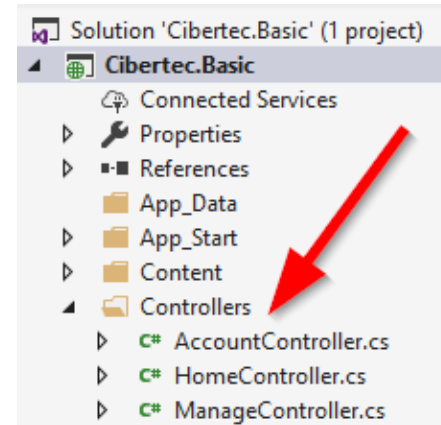


Exploración de las plantillas de Visual Studio 2017



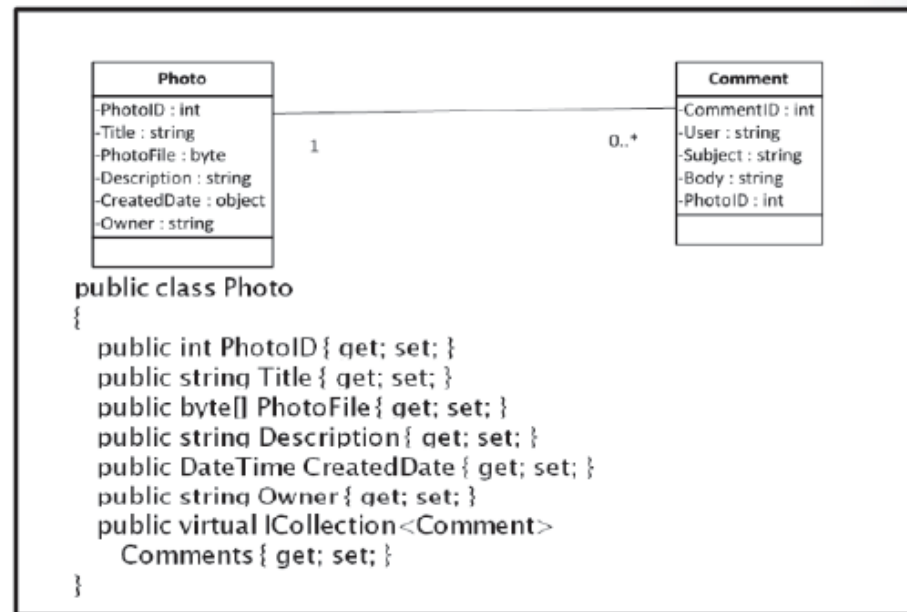
Convenciones de nombres MVC

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



Modelos

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



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



Personalizar los modelos usando atributos y validaciones

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



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();
    }

    0 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 folder 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

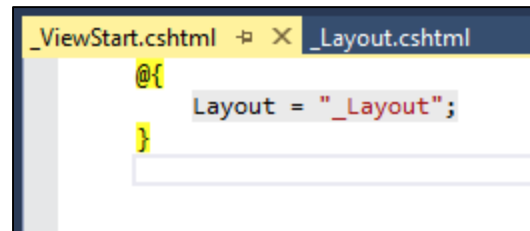
- Este archivo incluye todo el marco y plantilla HTML, es decir, tags como `<head>`.
- Nuevo tag “environment” el cual permite, según el despliegue que se realice, agregar unos u otros archivos.
- `@RenderBody()` : permite renderizar las vistas.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <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")
</head>
<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>
        </button>
        @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
      </div>
      <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li>@Html.ActionLink("Home", "Index", "Home")</li>
          <li>@Html.ActionLink("About", "About", "Home")</li>
          <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        </ul>
        @Html.Partial("_LoginPartial")
      </div>
    </div>
  </div>
  <div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
      <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
    </footer>
  </div>
  @Scripts.Render("~/bundles/jquery")
  @Scripts.Render("~/bundles/bootstrap")
  @RenderSection("scripts", required: false)
</body>
</html>
```

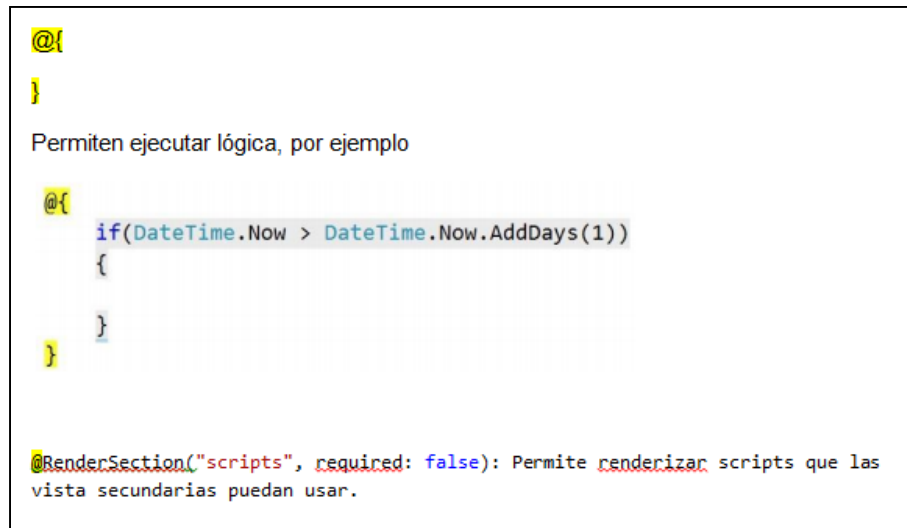


Vista: _ViewStart.cshtml

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



```
_ViewStart.cshtml X _Layout.cshtml
@{
    Layout = "_Layout";
}
```



```
@{
}

Permiten ejecutar lógica, por ejemplo

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

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



Ejercicio N° 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 N° 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 N° 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 N° 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 N° 4.1: Completar los repositorios

Agrega los repositorios faltantes:

- OrderItemRepository
- OrderRepository
- ProductRepository
- SupplierRepository
- UserRepository

