




# Desarrollo de Aplicaciones II

Carlos A. Quinto Cáceres  
cqintoc@usmp.pe

Semana 01  
06/08/14



## Agenda

- Introducción MVC
- ASP.NET MVC
- Vista rápida ASP.NET MVC
  - Proyectos
  - Controller
  - View
  - Model

## Logros del día

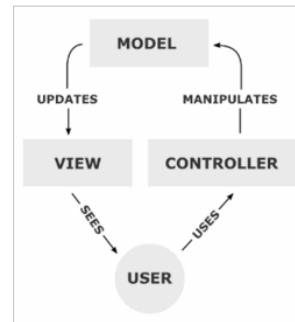
- Al final de la clase, el alumno podrá:
  - Definir los componentes de una aplicación MVC.
  - Crear proyectos ASP.NET MVC y acceder a las páginas creadas, así como crear vistas y modelos.

## Introducción MVC

Principales conceptos

## MVC

- Es un patrón de arquitectura de software.
- Su principal objetivo es separar el código de la aplicación en tres roles, cada uno con una responsabilidad diferente:
  - Modelo
  - Vista
  - Controlador



## Model (Modelo)

- Representa los datos de la aplicación.
- A través del Model se podrá acceder y modificar la información.
- Aquí se podrán tener las funciones para acceder a las tablas y ejecutar las sentencias SQL.

## View (Vista)



- Representa la interfaz de la aplicación que van a visualizar los usuarios.
- Normalmente se encarga de mostrar los datos de un Model en un formato establecido.
- Se trabajan con los datos, pero son se acceden directamente a éstos.

## Controller (Controlador)



- Es el encargado de recibir las peticiones de los usuarios, así como la información que envían.
- Sirve de enlace entre las Views y Models.
- También es responsable de enviar la respuesta al cliente.

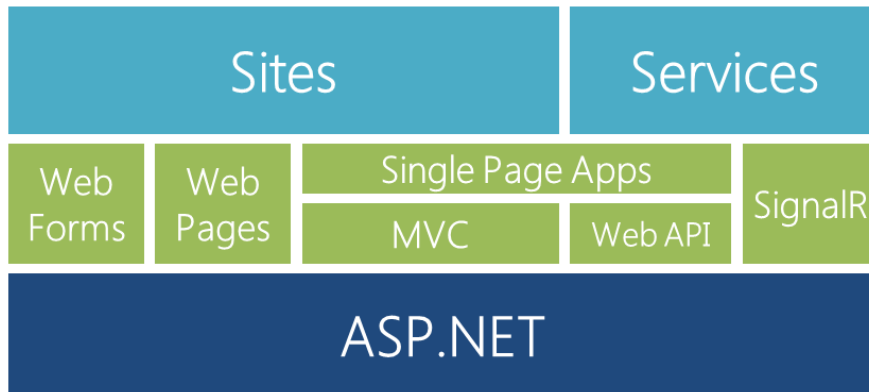
## ¿Por qué usar MVC?

- Facilidad de mantenimiento en las aplicaciones.
- Reutilización de código y componentes.
- Es sencillo crear diferentes presentaciones de los mismos datos.
- Los desarrollos bajo MVC son más fáciles de escalar.
- Mejor control sobre el HTML que se genera.
- Bajo esta estructura, es más fácil realizar pruebas a la aplicación.

## ASP.NET MVC

¿Cómo se trabaja MVC en ASP.NET?

# ASP.NET



## ASP.NET Web Pages

- Para crear aplicaciones Web simples.
- Necesidad de obtener páginas rápidamente.
- Control máximo sobre el código HTML generado.
- Pueden crearse utilizando Visual Studio usando la sintaxis Razor.

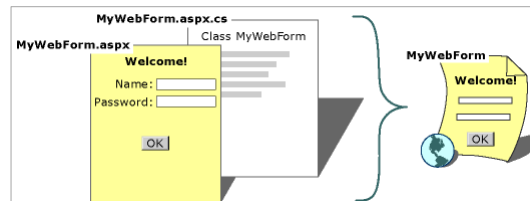
```

<section id="login">
  @if (WebSecurity.IsAuthenticated) {
    <p>
      Hello, <a href="#">
        <a href="#">Logout</a>
      </p>
  } else {
    <ul>
      <li><a href="#">Register</a></li>
      <li><a href="#">Log in</a></li>
    </ul>
  }

```

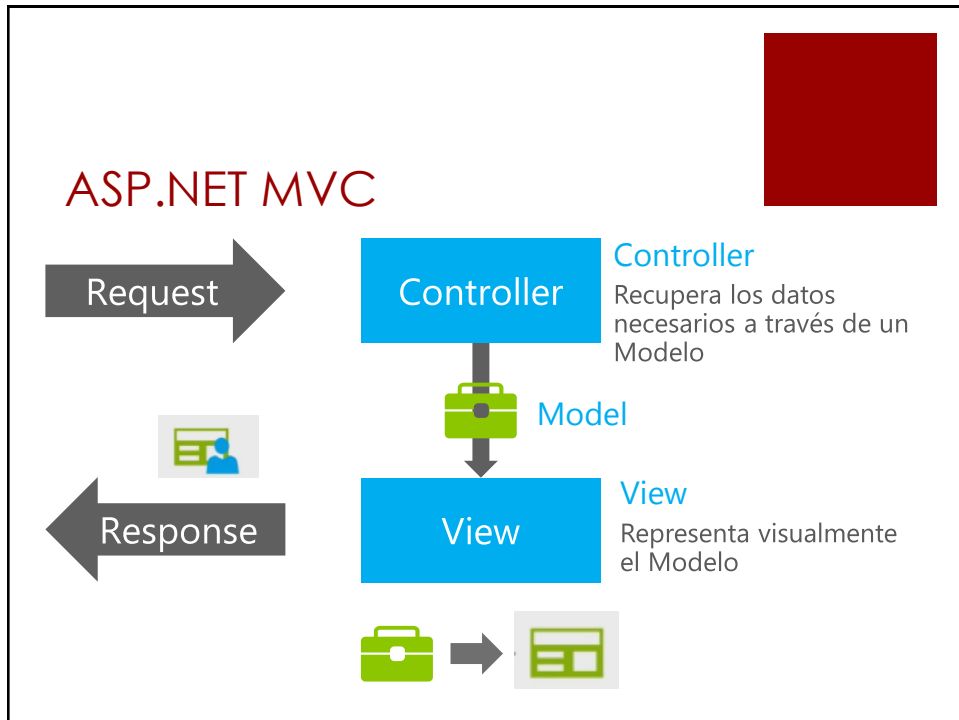
## ASP.NET Web Forms

- Desarrollo de aplicaciones Web basado en controles y eventos.
- Se pueden obtener aplicaciones Web de forma rápida.
- Se tiene poco control sobre el HTML que se genera.



## ASP.NET MVC

- Es el framework que nos permite crear aplicaciones Web haciendo uso del patrón MVC.
- Nos permite aislar la lógica de negocio de la interfaz para permitir un mejor mantenimiento, permitir pruebas y tener una estructura de aplicación más definida.

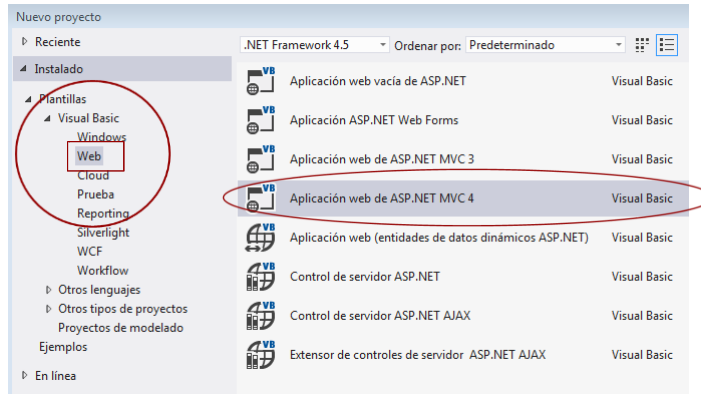


## Vista rápida ASP.NET MVC

Creando los primeros proyectos MVC

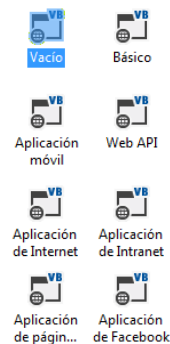


## Proyectos MVC



## Plantillas proyecto

- **Básico:** configuración mínima y con un diseño básico (layout).
- **Aplicación de Internet:** implementa un sitio Web del tipo público con un layout definido e implementado la autenticación con formularios.
- **Aplicación de Intranet:** es similar a la aplicación de Internet, pero utiliza autenticación Windows.
- **Aplicación móvil:** Layout basado en jQuery Mobile.
- **Web API:** orientado para crear servicios HTTP como RESTful.
- **Aplicación de página simple:** para usar Knockout.js y ASP.NET Web API.
- **Aplicación de Facebook:** permite centrarnos en la lógica de la aplicación y no preocuparnos de la integración con Facebook.



## Acceso en Web Forms

- Para las aplicaciones **Web Forms**, el acceso a las páginas era de la siguiente manera:
  - `http://localhost/Default.aspx`
  - Enlaza al archivo físico: `c:/<path>/Default.aspx`
  - `http://localhost/admin/Login.aspx`
  - Enlaza al archivo físico: `c:/<path>/admin/Login.aspx`
  - `http://localhost/Producto/Listado` (sin extensión)
  - Mensaje de error: Error 404

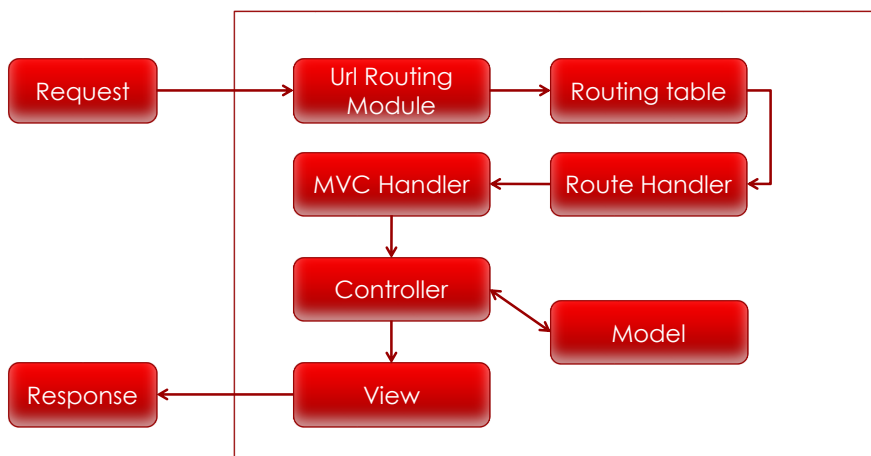
## Acceso en MVC

- Cuando se realiza una petición, un componente conocido como motor de enrutamiento (routing engine) enlaza la petición con una ruta específica.
- La ruta establece el controlador y el método que debe procesar la petición.
- Los métodos que procesan las solicitudes se denominan **action methods**.
- El método produce el resultado a enviarse al cliente, que normalmente es HTML presentado por un View.

## Acceso en MVC

- Para las aplicaciones **ASP.NET MVC**, tenemos un control completo sobre las URL.
- Las URL's podemos enlazarlas a un método específico de un Controller:
  - `http://localhost/productos/index`
  - Controller: Productos y Método: Index
  - `http://localhost/productos`
  - Controller: Productos y Método : Index
  - `http://localhost/servicios/detalle`
  - Controller: Servicios y Método : Detalle

## Ciclo de vida



## Controller

- Los controladores son los encargados de procesar las peticiones HTTP que realizan los usuarios.
- Cada petición es manejada por un controlador específico.
- Cada controlador contiene métodos que son conocidos como **action methods** que normalmente devuelven vistas.

## Convenciones

- Los nombres de los controladores deben de terminar con la palabra Controller.
  - Ejemplo: ProductosController

## Ejercicio 01

1. Crear una aplicación ASP.NET MVC:
  - Nombre: **semana01-01**
  - Analizar la estructura de carpetas
2. Ejecutar la aplicación.
3. Crear el controlador Home y ejecutar la aplicación.
4. Crear la vista para el método correspondiente y ejecutar la aplicación.

## View

- Los métodos de los controladores invocan una vista que será presentada como respuesta al usuario.
- Normalmente las vistas:
  - Incluyen código de servidor para producir un HTML final.
  - Muestran la información de un modelo.

## View

- El código de servidor dentro de las vistas se puede ejecutar por uno de dos motores:
  - ASPX
  - Razor
- **Una forma** mediante la que un controlador puede pasar datos a una vista, es usando la opción **ViewData**.

## Convenciones

- Las vistas deben colocarse dentro del folder:
  - /Views/<nombre controller>
  - Ejemplo: /Views/Productos/
- La vista que se muestra para un método por defecto tiene el mismo nombre del método.
- El orden de búsqueda de una View solicitada es en el siguiente orden de carpetas:
  - /Views/<Controller>/File.aspx
  - /Views/<Controller>/File.ascx
  - /Views/Shared/File.aspx
  - /Views/Shared/File.ascx

## Ejercicio 02

1. Crear una aplicación ASP.NET MVC
  - Nombre: **semana01-02**
2. Crear el controlador: **ContactoController**
3. Enviar desde el controlador a la vista la siguiente información:
  - Nombre de la empresa
  - Teléfono de la empresa
  - Correo de la empresa
  - Dirección de la empresa
4. Mostrar la información en la vista.
5. Probar y ejecutar la aplicación

## Model

- En ASP.NET MVC, los modelos pueden ser:
  - Data model
  - Business model
  - View model
- La información de los modelos se puede pasar desde un controlador a una vista haciendo uso de **ViewData** o haciendo uso de las **strongly typed view** (fuertemente tipada).

## Ejercicio 03



1. Crear una aplicación ASP.NET MVC, crear:
  - Nombre: **semana01-03**
  - Controlador: AlumnoController
  - Método: Detail
  - View: Detail
  - Model: Alumno con las propiedades:
    - Código, Nombre, Edad
2. En el controlador, crear una referencia al modelo, llenarlo con datos de prueba, pasar la información a la vista y mostrar la información del modelo.

## Enlaces de interés



- Visión ASP.NET MVC
  - <https://msdn.microsoft.com/cs-cz/library/dd381412%28v=vs.100%29.aspx>