# Introducción a ASP.NET

## Marcos de Desarrollo

Diseño e implementación de aplicaciones Web con .NET

# Contenido

- **Introducción a ASP.NET**
- **Visual Studio**
  - **Websites y Web Projects**
  - **Crear una página Web (*Web Form*)**
  - **Tipos de archivo en una aplicación ASP.NET (*Web Project*)**
  - **Toolbox**
  - **Modelos de código (inline code y code-behind)**
- *Web Forms*
  - El modelo de eventos de ASP.NET
  - *View State*
  - Flujo de página
  - Transferencia de control entre páginas
  - Trazas
  - Páginas de Error
- Controles de servidor
- Master Pages
- Internacionalización
- Autenticación

# Objetivos

- Conocer los fundamentos de ASP.NET
- Saber crear una aplicación web ASP.NET de tipo Web Project
- Comprender el modelo Code-Behind de ASP.NET

# ¿Qué es ASP.NET?

- Es la evolución de Active Server Pages (ASP)
- ASP.NET es el framework de programación Web dentro de .NET
  - Al codificar aplicaciones ASP.NET se tiene acceso a las clases del .NET Framework
  - Permite desarrollar aplicaciones Web con un modelo “similar” al utilizado para aplicaciones Windows
  - El componente fundamental de ASP.NET es el **WebForm**
    - Una aplicación Web ASP.NET consta de uno o más WebForms
- En ASP.NET se permite utilizar cualquier lenguaje .NET

# Visual Studio

- La gran mayoría de los sitios web con ASP.NET se desarrolla con Visual Studio
- Características:
  - Servidor Web integrado. Sólo acepta conexiones locales
  - Desarrollo multilenguaje
  - Genera parte del código necesario
  - *Intellisense*, comentado y descomentado automático, …
  - Depuración. Ejecución paso a paso, *breakpoints*, ver información en memoria, …

# *Websites y Web Projects*

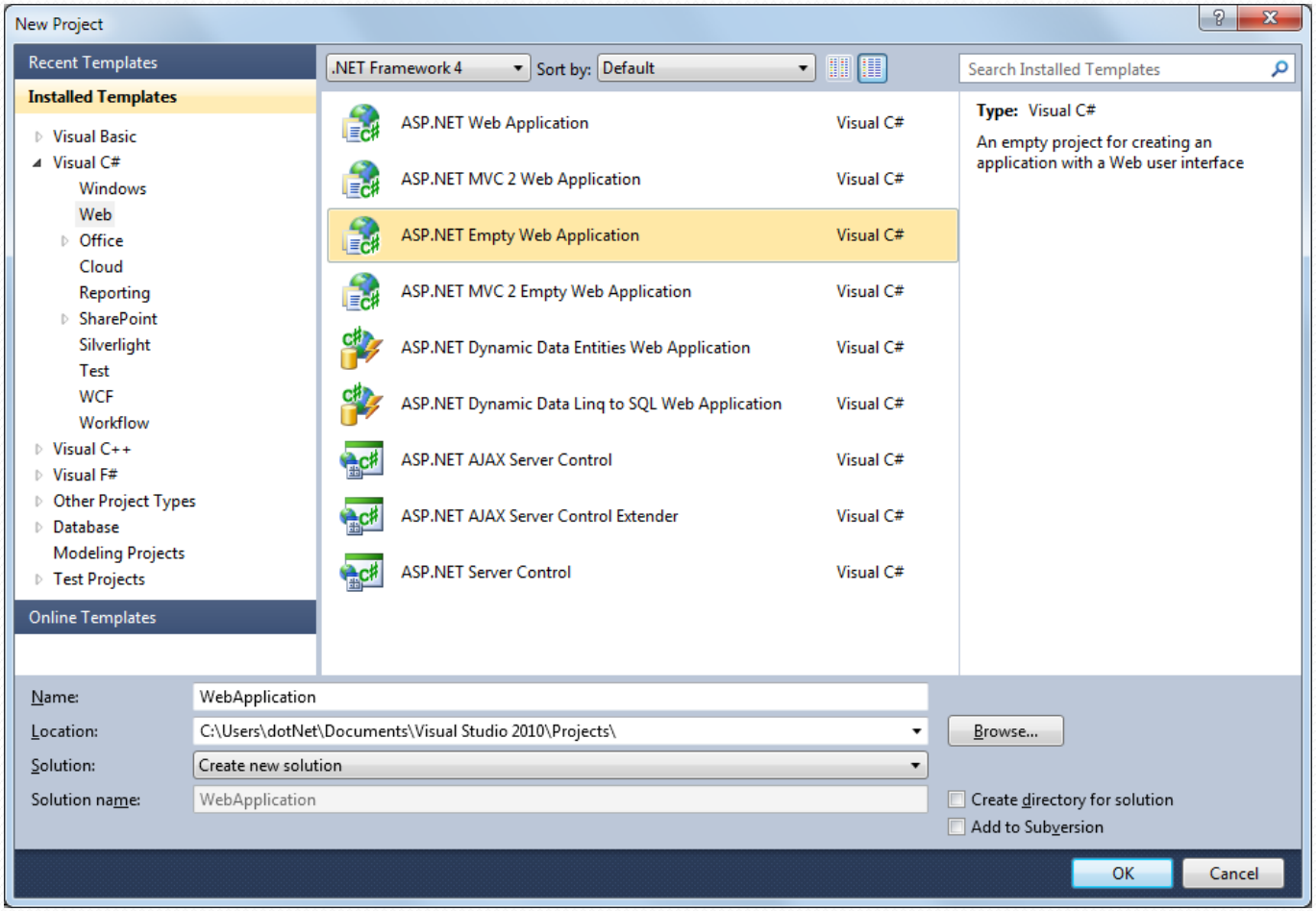
- VS ofrece dos modos de crear una aplicación Web con ASP.NET:
  - **Web Project**. Desarrollo basado en un proyecto
    - File > New > New Project > Asp.NET (Empty) Web Application
    - VS crea archivo .csproj (asumiendo C#) que almacena los archivos que pertenecen al proyecto y características de depuración
      - Estructura similar a un proyecto de consola o ventanas
    - Cuando se ejecuta, VS compila todo el proyecto a un solo ensamblado, antes de lanzar el navegador web
    - Deploy: ensamblado + archivos .aspx
  - **Website**. Desarrollo sin proyecto
    - File > New WebSite
    - En este caso VS asume que cada archivo en el directorio del sitio web forma parte de la aplicación
    - En este caso VS no necesita compilar el código. En lugar de esto, ASP.NET compila el código la primera vez que se solicita un página (aunque se puede utilizar pre-compilación)
    - Deploy : archivos .aspx + código asociado (.aspx.cs, si se utiliza C-B)

# ¿*Website* o *Web Project*?

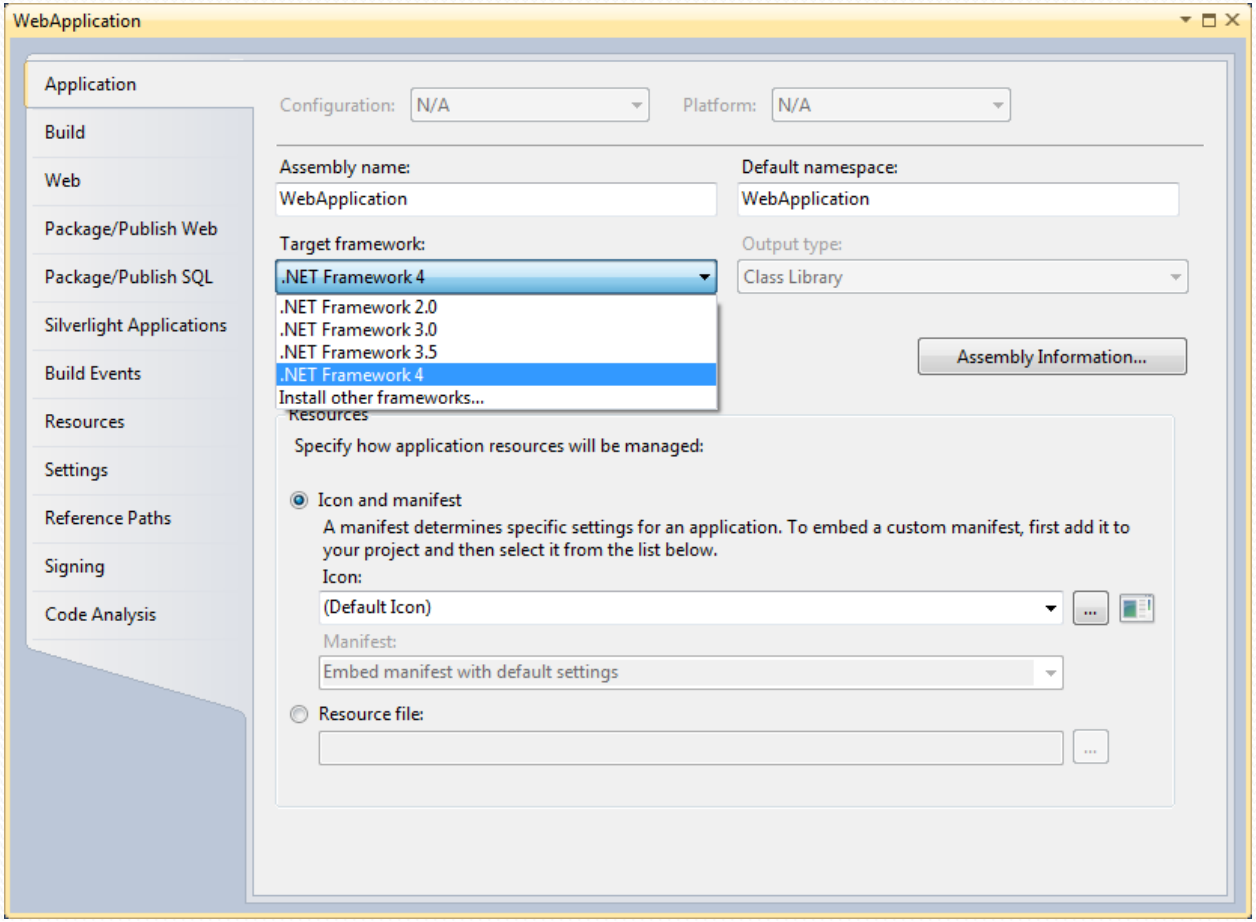
- En proyectos muy pequeños, puede resultar más sencillo utilizar *Website*

- Para proyectos medianos o grandes es preferible utilizar *Web Project*
  - Es el tipo de desarrollo que utilizaremos

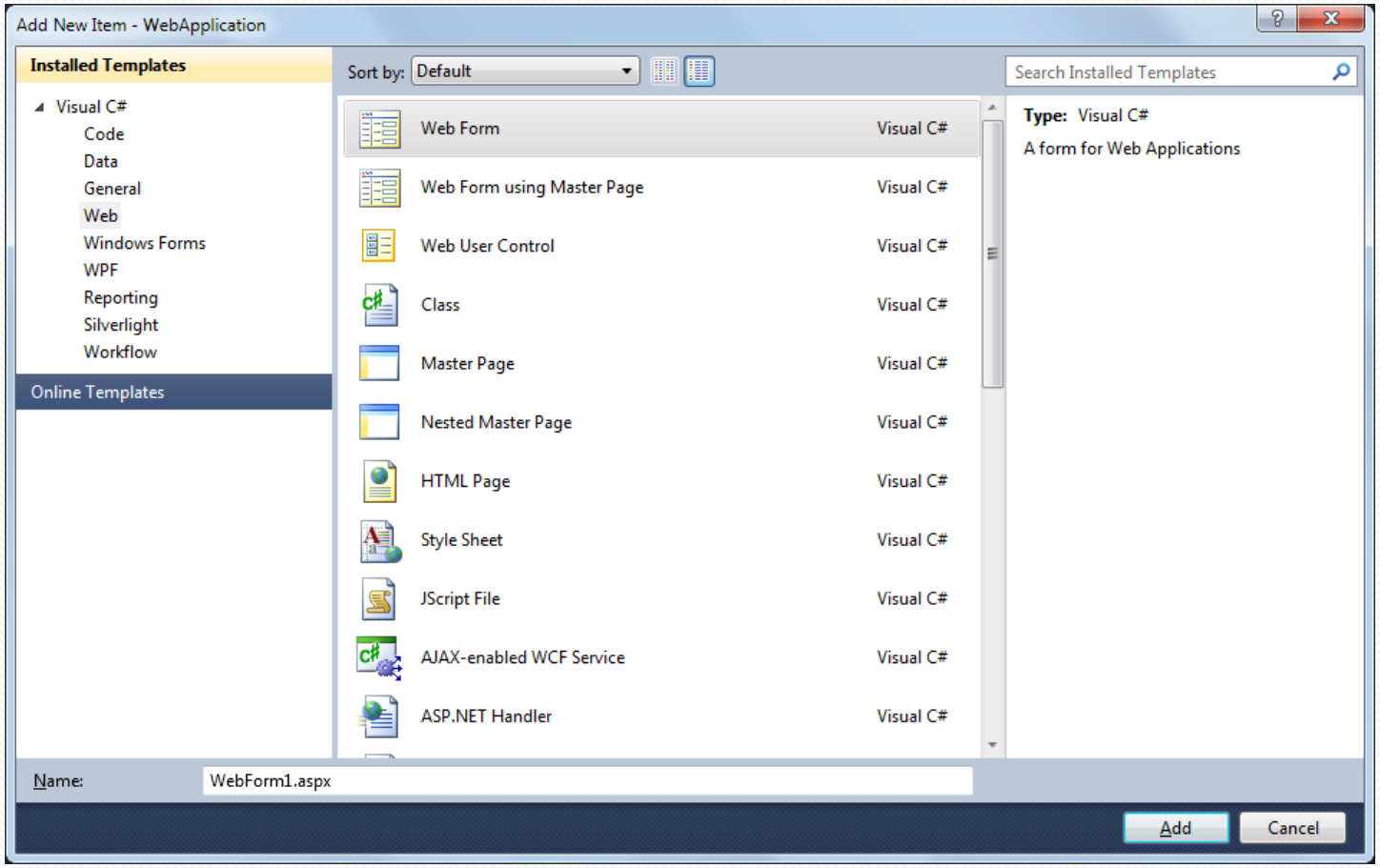
# Crear un *Web Project*



# Versión del *.NET Framework*

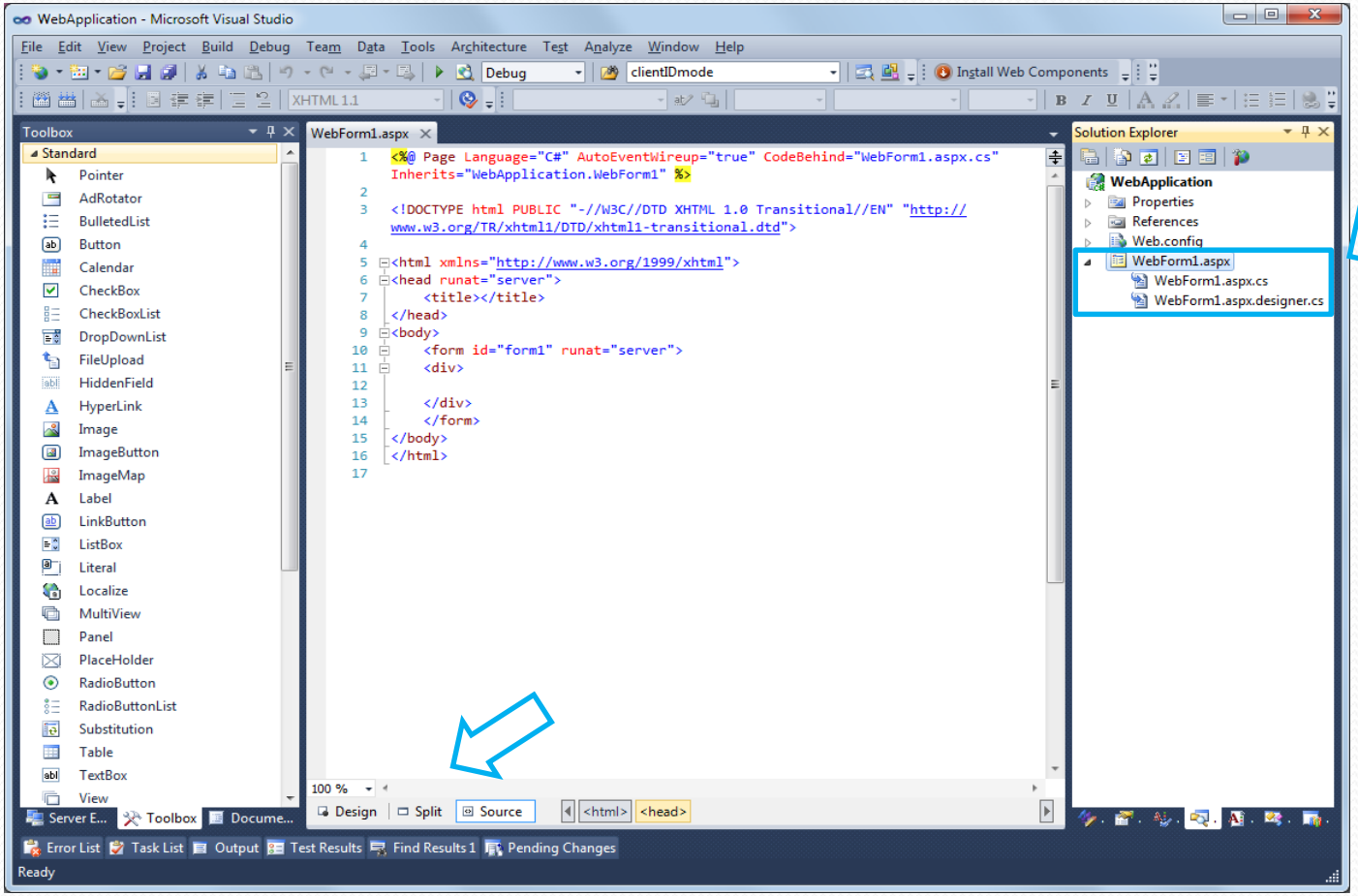


# Crear una página Web



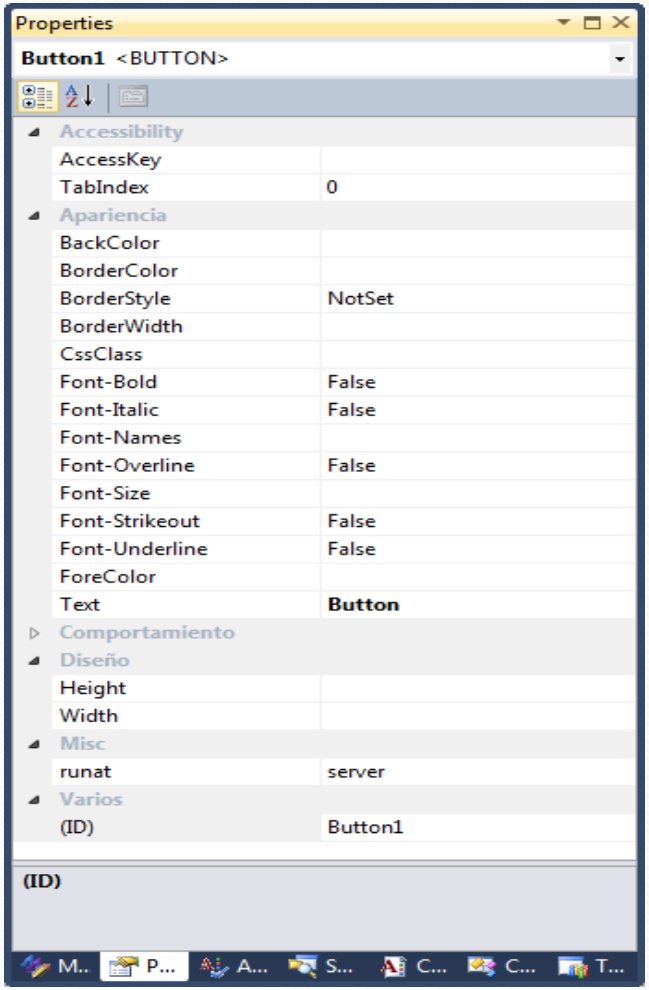

Solution Explorer (sobre el nombre del proyecto) > Add > New Item

# Crear una página Web

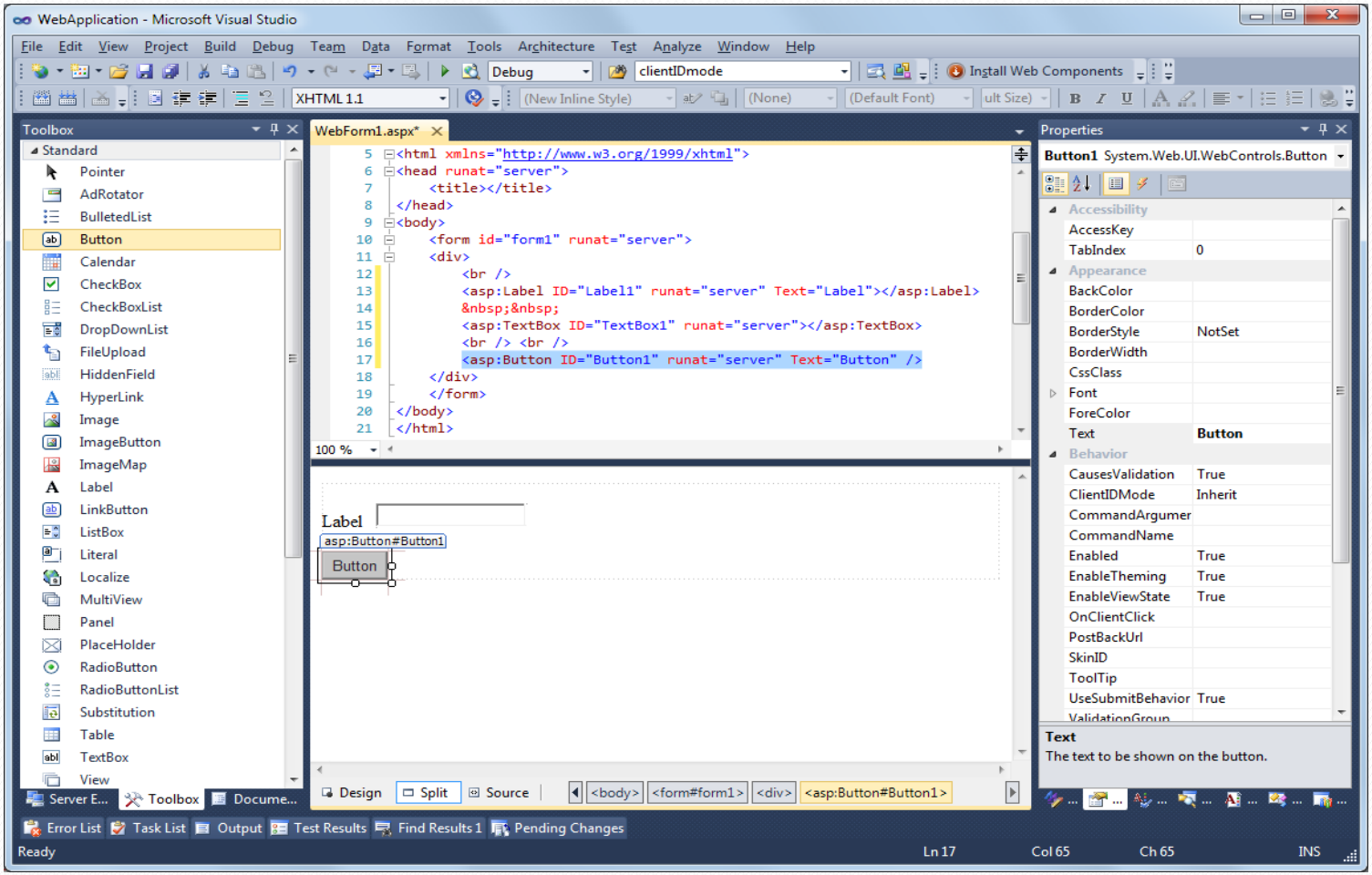


# Crear una página Web

- Los controles (Label, Button, Text Box, ...) se pueden:
  - Escribir manualmente en vista de código
  - Arrastrar de "ToolBox" a la vista de diseño o a la de código
- La ventana "Properties" permite configurar los controles



# Crear una página Web

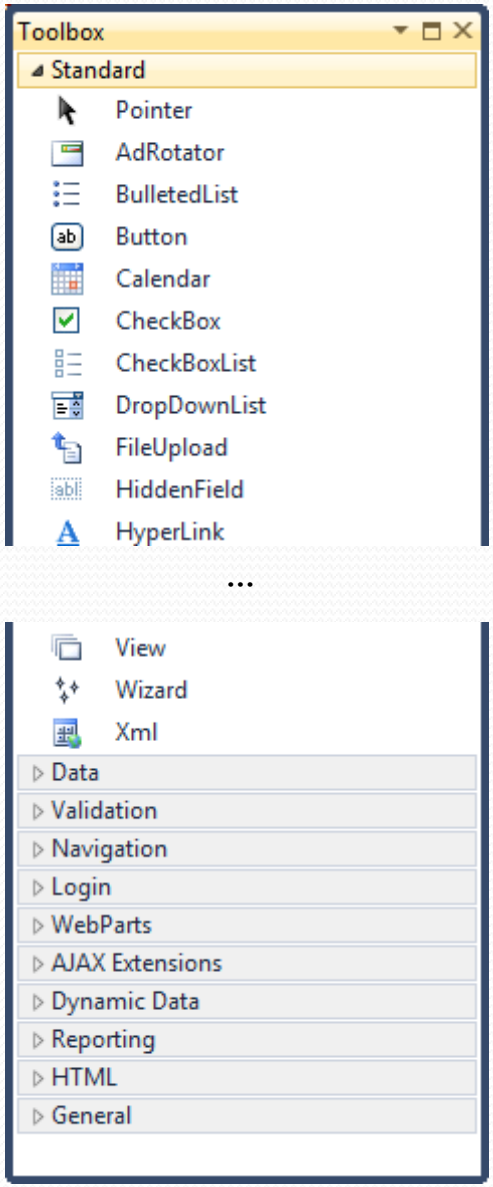


# Tipos de archivo en una aplicación ASP.NET

- En la ventana "Solution Explorer" podemos encontrar:
  - **.aspx:** páginas Web ASP.NET. Contienen la interfaz de usuario y, opcionalmente, código.
  - .ascx: controles de usuario.
  - .asmx o .svc: servicios Web ASP.NET
  - **web.config:** archivo de configuración de la aplicación Web
  - **global.asax:** permite definir variables globales y reaccionar a eventos de la aplicación (arranque de la aplicación, inicio de una sesión, ...)
    - No se crea por defecto. Si se necesita, hay que añadirlo.
  - **.cs:** código C# asociado a una página .aspx (*code-behind*)
  - .master: páginas maestras (*Master Pages*)
  - Otros componentes: imágenes, archivos XML, hojas de estilos, etc.

# ToolBox

- La caja de herramientas agrupa distintos tipos de controles que se pueden utilizar en una página ASP.NET
  - Standard:
    - controles ASP.NET
  - Validation
    - controles de validación
  - HTML
    - controles HTML estáticos (tradicionales)
  - …

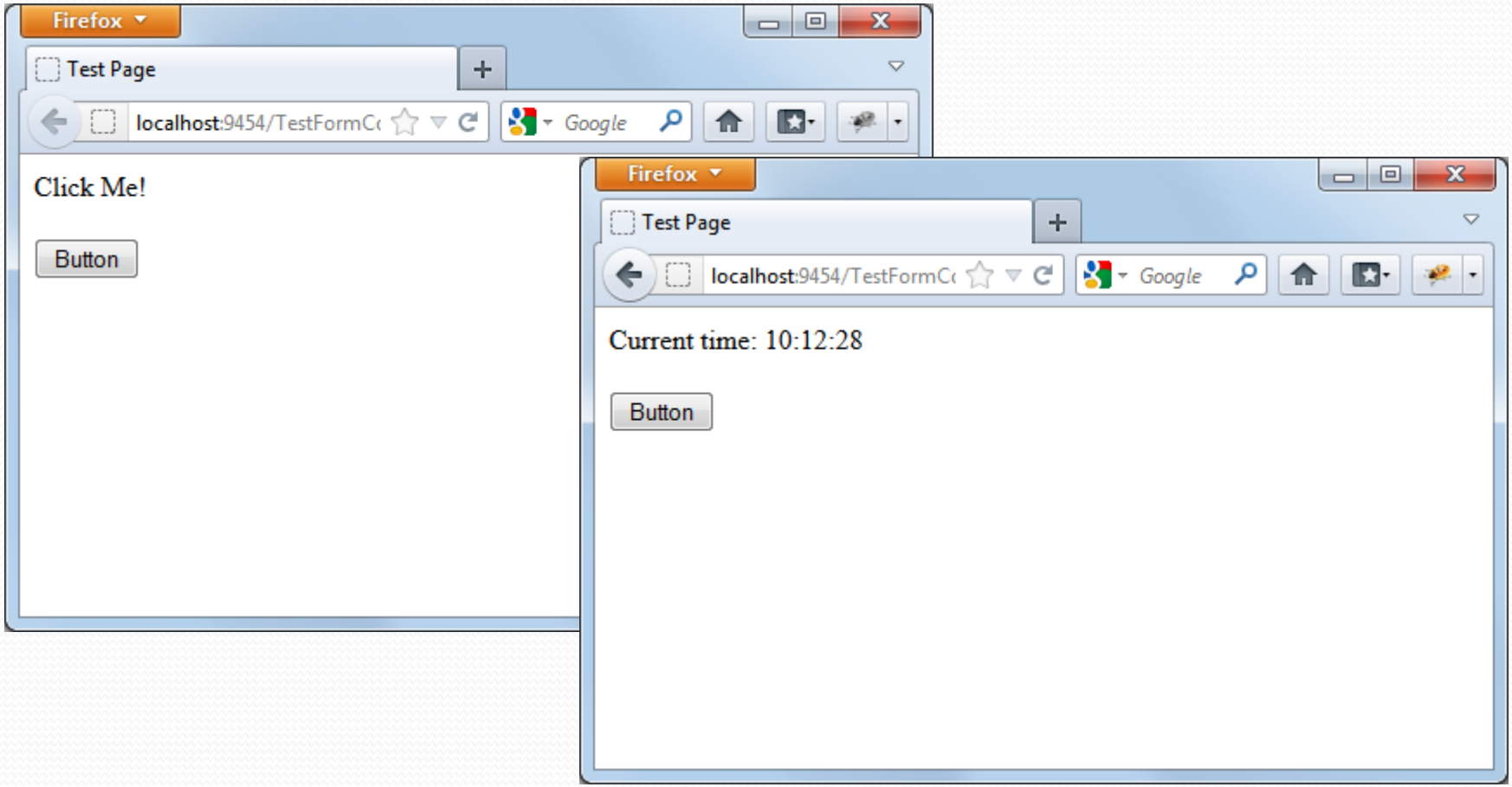


# Modelo de código

- VS soporta dos modelos para codificar páginas Web:
  - Inline code:
    - Código y etiquetas HTML se almacenan en único archivo .aspx
    - Código se encierra en uno o más bloques <script>
    - Se puede hacer *debug*, utilizar *IntelliSense*, …
    - Sólo en *Website*
  - Code-Behind:
    - Separa cada página en:
      - .aspx, que contiene etiquetas HTML y controles ASP.NET
      - .aspx.cs, que contiene el código fuente de la página
      - .aspx.designer.cs, que contiene código generado automáticamente (básicamente, definiciones de controles)
    - Separación clara de la interfaz de usuario
    - En *Website* y en *Web Project*, con pequeñas diferencias

# Ejemplo

- Veamos un sencillo ejemplo, codificado con los dos modelos



# Ejemplo de modelo *Inline code*

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Current time: " + DateTime.Now.ToLongTimeString();
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Test Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="Click Me!" />
        <br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />
    </div>
    </form>
</body>
</html>
```

TesFormInline.aspx

# Ejemplo de modelo *Code-behind* (.aspx)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind ="TestFormCodeBehind.aspx.cs"
    Inherits="WebApplication.TestFormCodeBehind" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Test Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="Click Me!"></asp:Label><br />
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />
    </div>
    </form>
</body>
</html>
```

TesFormCodeBehind.aspx

# Ejemplo de modelo *Code-behind* (.aspx.cs)

```
using System;



namespace WebApplication
{
    public partial class TestFormCodeBehind : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            this.Label1.Text = "Current time: " + DateTime.Now.ToLongTimeString();
        }

    }
}
```

TesFormCodeBehind.aspx.cs

# Ejemplo de modelo *Code-behind* (.aspx.designer.cs)

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace WebApplication {

    public partial class TestFormCodeBehind {

        protected global::System.Web.UI.HtmlControls.HtmlHead Head1;

        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        protected global::System.Web.UI.WebControls.Label Label1;

        protected global::System.Web.UI.WebControls.Button Button1;
    }
}
```

TesFormCodeBehind.aspx.designer.cs

# Bibliografía

- Recomendada:
  - M. MacDonald, A. Freeman, M. Szpuszta. ***Pro ASP.Net 4 in C# 2010***. 4th Ed. Apress. 2010.