Índice de Contenidos

HTML Forms

- Un HTML Form es la porción de un documento HTML que aparece entre las etiquetas **`<form></form>`**
- Un botón **submit (`<input type="submit">`)** juega un rol especial
  - Cuando es pulsado, el navegador envía el HTML Form junto con cualquier entrada de datos del usuario al servidor Web
- La forma en la que se envía el HTML Form, dependerá del atributo **`Method`**:
  - Si el atributo **`Method`** del form **no está presente** o tiene el valor **GET**, el navegador enviará al servidor un comando **HTTP GET**
  - Si el atributo **`Method`** del form tiene el valor **POST**, el navegador enviará al servidor un comando **HTTP POST**

HTML Forms

Method = GET

```
<form method="get">
 . . .
</form>

GET /suma.html?op1=2&op2=2
   HTTP/1.1
  .
  .
  .
Connection: Keep-Alive
[blank line]
```

El navegador envía los datos introducidos como una cadena de consulta

Method = POST

```
<form method="post">
 . . .
</form>

POST /suma.html HTTP/1.1
  .
  .
Content-Type: ...
Content-Length: 11
[blank line]
op1=2&op2=2
```

El navegador envía los datos introducidos en el cuerpo de la solicitud HTTP

Procesamiento en el Servidor

- Existen varias tecnologías de procesamiento
  - **CGI (Common Gateway Interface)**
    - Define una API de bajo nivel
    - Popular en entornos UNIX, no tanto en Windows
  - **ISAPI (Internet Server Application Programming Interface)**
    - Son DLL Windows que “corren” bajo Internet Information Server (IIS) Escritas en C++
    - Mejor rendimiento que CGI
  - **ASP (Active Server Pages)**
    - Simple solución: **HTML** + **Script del lado del servidor**
    - Programadas en **JScript** o **VBScript**
    - Objetos intrínsecos que abstraen detalles de bajo nivel de HTTP. Objetos **Request** y **Response**
    - Permite usar ADO (ActiveX Data Objects) para acceso a datos

¿Qué es ASP.NET?

- ASP.NET es el framework de programación Web dentro de .NET
- 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
- Independencia del cliente (navegador, S.O., dispositivo físico, etc.)
- Permite utilizar cualquier lenguaje .NET

Componentes de una aplicación Web ASP.NET

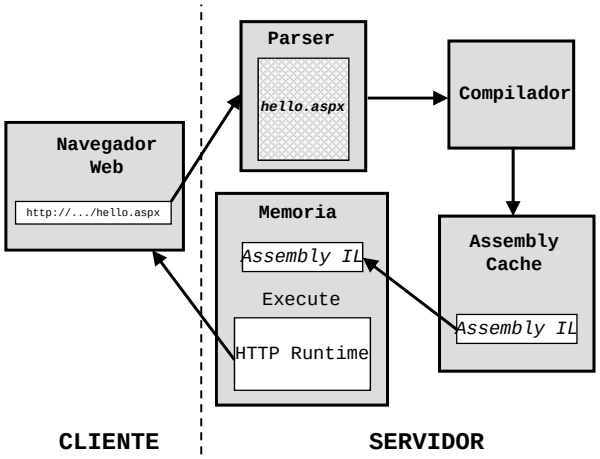
- WebForms (Formularios Web)
  - Uno o más archivos con extensión **.aspx**
- Archivos Code-Behind
  - Archivos asociados a WebForms que contienen código que se ejecutará en el lado del servidor (e.g. VB.NET, C#, etc.)
- Archivos de configuración con formato XML
  - Un único archivo **`Machine.config`** por servidor
  - Un archivo **`Web.config`** por cada aplicación

Componentes de una aplicación Web ASP.NET

- **`Global.asax`**
  - Contiene los métodos desde los que el usuario puede gestionar los diferentes eventos globales de una aplicación
    - Inicio/fin de aplicación, inicio/fin de sesión, etc.
- Directorio BIN
  - Contiene el *assembly* de la aplicación (e.g. **`MyWebApp.dll`**)
  - Contiene referencias proyecto: cero o más *assemblies* (externos)
- Enlaces a Servicios Web XML
  - Permiten a la aplicación ASP.NET enviar y recibir datos desde Servicios Web

Modelo de ejecución de ASP.NET

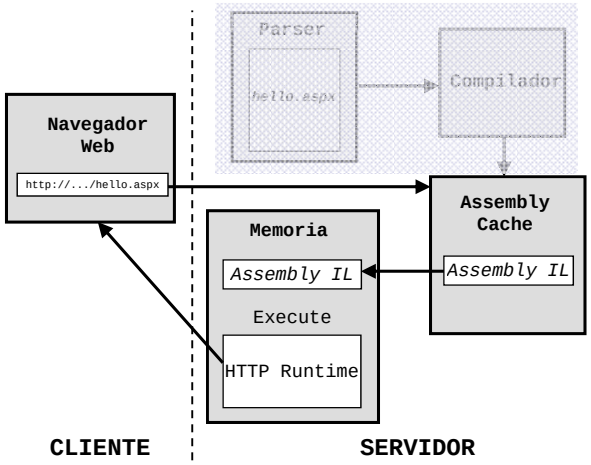
Primera petición



1. El navegador cliente envía una petición GET HTTP al servidor
2. El parser de ASP.NET interpreta el código fuente
3. ASP.NET invoca al compilador
4. El Runtime carga y ejecuta el código intermedio (IL)

Modelo de ejecución de ASP.NET

Segunda petición



1. El navegador cliente envía una petición GET HTTP al servidor
2. El Runtime carga y ejecuta inmediatamente el código intermedio ya compilado durante el primer acceso

Diferentes tipos de proyectos

- ASP.NET Web Site
  - File – New WebSite
  - No tiene estructura de proyecto
    - Pertenencia de un fichero al proyecto basada en su ubicación en la carpeta del Web Site
- ASP.NET Web Application
  - File – New – New Project – Asp.NET Web Application
  - Proyecto para la realización de aplicaciones Web (es el que usaremos)
- ASP.NET Web Service Application
  - File – New – New Project – Asp.NET Service Application
  - Proyecto para la realización de servicios web

Formularios Web (Web Forms)

- ¿Qué es un Formulario Web?
  - Componentes de un Web Form
- Controles de servidor
  - HTML
  - WebControls
- Eventos en un Web Form
- Ciclo de vida de un Web Form

Formularios Web (Web Forms)

- Web Form
  - Es una página expresada en lenguaje de marcas que es compilada y ejecutada dinámicamente en el servidor para generar la salida solicitada por el cliente (explorador ó dispositivo)
  - Se compone de:
    - Parte vista: Contiene código HTML y declaraciones de controles del lado del servidor
      - Tiene extensión **`.aspx`**
    - Código asociado, denominado **CodeBehind**
- Separación del aspecto visual (vista) del código (controlador)
  - Permite trabajo independiente de diseñadores gráficos y programadores

Formularios Web (Web Forms)

Componentes de un Web Form

- La parte vista de un Web Form (e.g. **`PageName.aspx`**) puede incluir:
  - Directivas de página
    - Se indican dentro de **`<%@ Page ... %>`**
    - Permiten especificar atributos específicos de una página .aspx
      - **`CodeBehind`**: fichero de código asociado
      - **`ContentType`**: tipo MIME de la response
      - **`ErrorPage`**: URL ante aparición de errores
      - **`Inherits`**: clase base del objeto Page
      - **`Language`**: lenguaje de programación empleado
      - **`Trace`**: habilitación de la traza para la página actual
      - **`EnableViewState`**: habilitación de la propiedad VIEWSTATE
      - etc.
    - e.g.

```
<%@ Page Language="C#" CodeBehind="PageName.aspx.cs"
ErrorPage="/DefaultError.html" Trace="true"%>
```

Formularios Web (Web Forms)

Componentes de un Web Form

- La parte vista de un Web Form puede incluir (cont.):
  - Controles:
    - `<input type="text">`
    - `<asp:Button runat="server">`
  - Comentarios
    - `<!-- html comment -->`
    - `<%-- asp.net comment --%>`
  - Data bind expressions
    - `<%# expression %>`
  - Bloques de código
    - `<script runat="server"> ... </script>`
  - Render code
    - `<%= %>` o `<% %>`
    - No recomendado; preferible etiquetas `<script runat="server">` asociadas a los eventos de los controles

Formularios Web (Web Forms)

Componentes de un Web Form. Ejemplo `Default.aspx`

```
<%@ Page Language="C#" CodeBehind="Default.aspx.cs"
    Inherits="ASPDotNetTutorial.Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>ASP.NET Tutorial</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

Formularios Web (Web Forms)

Componentes de un Web Form. Ejemplo **`Default.aspx.cs`**

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


        }
    }
}
```

Formularios Web (Web Forms)

Componentes de un Web Form. Ejemplo **`Default.aspx.designer.cs`**

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.42
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace ASPDotNetTutorialMasterPages
{
    public partial class Default
    {
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;
    }
}
```

Formularios Web (Web Forms)

Componentes de un Web Form

- El código asociado a un Web Form se conoce como **CodeBehind**
  - Implementa el patrón **Page Controller** (lo veremos más adelante)
  - Es una *partial class* (una sola clase, implementada en dos archivos) que contiene el código asociado a la página (controlador)
    - e.g. **`PageName.aspx.cs`**
      - **`public partial class PageName : System.Web.UI.Page`**
      - Incluye código generado por el programador
    - e.g. **`PageName.aspx.designer.cs`**
      - **`public partial class PageName`**
      - Incluye código autogenerado por el IDE

Formularios Web (Web Forms)

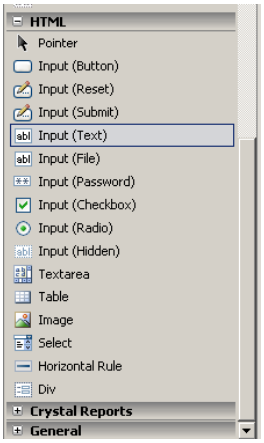
Controles de Servidor

- Componentes que se ejecutan en el lado del servidor
- Encapsulan partes de la interfaz de usuario
- Poseen el atributo **`runat="server"`**
- Pueden mantener su “estado” entre “PostBacks” al servidor
  - Para esto hacen uso del atributo **`ViewState`**
- Poseen un modelo de objetos común
  - Ej.: todos tienen las propiedades **`ID`** y **`Text`**
- Generan HTML específico según navegador del cliente

Formularios Web (Web Forms)

Tipos de Controles de Servidor

- **HTML**
  - Por defecto, los elementos HTML no son accesibles desde código del lado del servidor
    - Desde CodeBehind no es posible acceder a sus propiedades



Formularios Web (Web Forms)

Tipos de Controles de Servidor

- **HTML runat="server"**
  - Agregando **`runat="server"`** y el atributo **`id`**, se convierten en Controles de Servidor HTML
  - Se encuentran definidos como objetos dentro del Namespace
    - **`System.Web.UI.HtmlControls`**
  - Ejemplo:
    - .aspx

```
<input type="text" id="txtName" runat="server"/>
<span id="spnStart" runat="server">Start</span>
```

    - .aspx.cs

```
HtmlButton txtName;

txtName.BackColor = System.Drawing.Color.Grey;
String userName = txtName.Value;
```

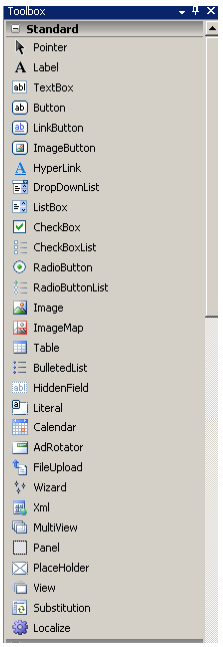
Formularios Web (Web Forms)

Tipos de Controles de Servidor

- **WebControls**
  - Sólo accesibles del lado del servidor
  - Poseen mayor funcionalidad
    - Definidos como objetos dentro del Namespace
      - `System.Web.UI.WebControls`
  - Ejemplo

```
<asp:TextBox ID="txtUserName"
  runat="server" Text="Your name"/>
```

  - Tipos de WebControls
    - Intrínsecos, de Validación, “Ricos”, listas vinculables datos
  - No tienen relación 1:1 con elementos HTML



Formularios Web (Web Forms)

Equivalencias de Controles

- Botón HTML

```
<input type="button" value="Search"/>
```

- Controles de Servidor HTML

```
<input type="button" value="Search" id="btnSearch"
runat="server" name="btnSearch"/>
```

- Controles de Servidor Web (WebControls)

```
<asp:Button ID="btnSearchASP" runat="server" Text="Search"/>
```

Formularios Web (Web Forms)

Equivalencias de Controles

WebControl	HTML equivalente
`<asp:button>`	`<input type="submit">`
`<asp:checkbox>`	`<input type="checkbox">`
`<asp:hyperlink>`	`<a href="…"> </a>`
`<asp:image>`	`<img src="…">`
`<asp:imagebutton>`	`<input type="image">`
`<asp:label>`	`<span> </span>`
`<asp:panel>`	`<div> </div>`
`<asp:radiobutton>`	`<input type="radio">`
`<asp:table>`	`<table> </table>`
`<asp:textbox>`	`<input type="text">`
`<asp:listbox>`	`<select size="…"> </select>`

Formularios Web (Web Forms)

WebControls – Intrínsecos

- Proveen nombres estándar, con propiedades comunes a los controles

```
<asp:RadioButton ID="rbtRadioButton" runat="server" BackColor="red"
    Text= "..." />
<asp:CheckBox ID="chkCheckBox" runat="server" BackColor="red"
    Text= "..." />
```

- Incluyen propiedades específicas

```
<asp:CheckBox ... Checked="true"/>
```

- Generan HTML acorde al navegador que llama a la página

```
<span style="background-color:Red;">
    <input id="chkCheckBox" type="checkbox" name="chkCheckBox"
    checked="checked" />
</span>
```

Formularios Web (Web Forms)

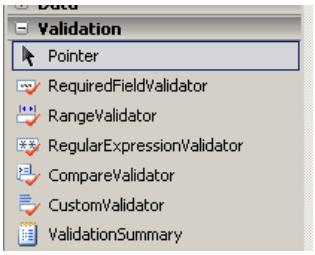
WebControls – Validación

- Son elementos ocultos que validan las entradas de datos contra algún patrón
- El proceso de validación puede ser llevado a cabo en:
  - Cliente
    - EnableClientScript="True" (opción por defecto)
    - El navegador del cliente debe soportar lenguaje script
    - Le da al usuario una respuesta inmediata
    - Reduce el número de "PostBacks"
  - Servidor
    - Repite la validación (siempre)
    - Permite validaciones más complejas (e.g. contra datos almacenados en base de datos)

Formularios Web (Web Forms)

WebControls – Validación

- ASP.NET proporciona 6 controles
  - **RequiredFieldValidator**. Campo obligatorio
  - **RangeValidator**. Valor dentro de un rango de tipos
  - **RegularExpressionValidator**. Valida contra un patrón o expresión regular
  - **CompareValidator**. Valida contra un valor constante o contra otro control
  - **CustomValidator**. Se dispara un evento (Cliente o Servidor) donde se controla la validación
  - **ValidationSummary**. No es un validador en sí mismo. Muestra los mensajes de error generados por otros controles de forma agrupada



Formularios Web (Web Forms)

WebControls – Validación

- Propiedades importantes
  - **`ControlToValidate`**
  - **`Text`**
  - **`ErrorMessage`**
  - **`Display`**
- En el lado del servidor se puede determinar si **todas** las validaciones fueron exitosas mediante la **`Page.IsValid`**
  - La propiedad **`IsValid`** **no** está disponible en los eventos **`Init`** ni **`Load`** del WebForm

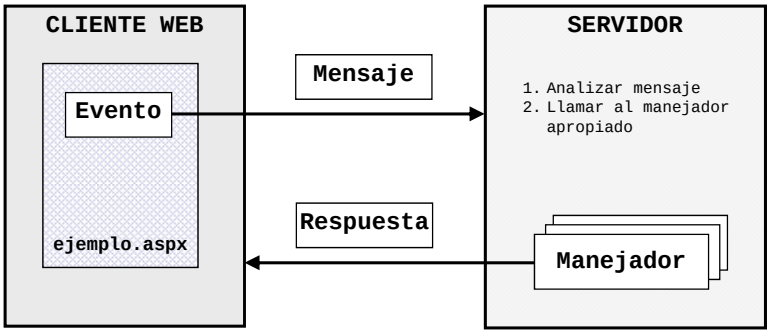
Formularios Web (Web Forms)

WebControls – Controles "Ricos"

- Controles con lógica de IU compleja encapsulados de forma sencilla
- Ejemplos:
  - **`<asp:AdRotator>`.** Permite mostrar banners de una secuencia predeterminada o aleatoria
  - **`<asp:Calendar>`**. Permite disponer de un calendario personalizable
  - **`<asp:Xml>`.** Permite formatear y mostrar el contenido de un fichero XML de acuerdo a un conjunto de estilos definidos en un fichero XSL

Formularios Web (Web Forms)

Eventos en un WebForm – Modelo de Eventos



Formularios Web (Web Forms)

Eventos en un WebForm

- Declaración de eventos en un control del lado del cliente
  - e.g. **`Default.aspx`**

```
<asp:Button ID="btnExample" runat="server" Text="Aceptar"
OnClick="btnExampleClick" />
```

- Atención del evento en el servidor (Code Behind)
  - e.g. **`Default.aspx.cs`**

```
protected void btnExampleClick(object sender, EventArgs e)
{
    this.btnExample.Text = "Pulsado";
}
```

PostBack

- Ocurre cuando una página genera un formulario HTML cuyos valores son reenviados a la misma página
  - Ejemplo: validación de controles
- ASP y otras tecnologías servidor pierden estado de la página entre llamadas
  - … a no ser que a nivel de código el estado se mantenga explícitamente
- ASP.NET mantiene el estado de los controles de servidor entre PostBack's
  - Válido para llamadas por GET y POST
  - Controles de servidor se rellenan automáticamente tras el PostBack
  - No se almacena estado en el servidor

PostBack

- El evento **`Page_Load`** se lanza en cada petición a un Web Form
  - Comportamiento diferente dependiendo de si es o no la primera vez que se muestra la página
    - Si un control tiene habilitado el VIEWSTATE , sólo necesita inicializarse en la primera llamada (**`Page.IsPostBack==false`**)

```
private void Page_Load(object server, EventArgs e)
{
    if  (Page.IsPostBack==false)
    {
        // Initialization code
    }
    else
    {
        // Request is a postback
    }
}
```

PostBack

- Existe la posibilidad de que el PostBack no devuelva control a la página que originó la llamada, sino que lo haga a una página diferente
- Propiedades relacionadas:
  - ***control*.PostBackUrl**
    - Dirección de la página empleada como PostBack
  - **Page.PreviousPage**
    - Dirección de la página que originó el PostBack
  - **PreviousPage.IsCrossPagePostBack**
    - Equivalente a **IsPostBack**
    - Indica si la página actual se abrió como consecuencia de un PostBack originado en una página diferente a la actual

PostBack

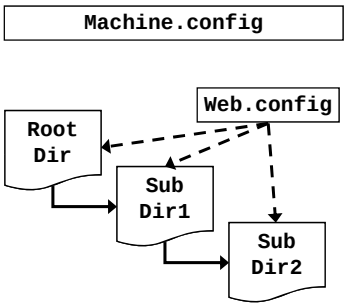
```
<!– InitialPage.aspx -->
<html>

    <body>
        <form id="form1" runat="server">
            <asp:TextBox ID="Input" runat="server" />
            <asp:Button ID="Button" Text="Test" runat="server"
                        PostBackUrl="TargetPostBack.aspx" />
        </form>
    </body>

</html>
```

Configuración

- Niveles configurables
  - servidor, root, subdirectorios web
- Ficheros
  - *Machine.config*
  - *Web.config*



Configuración
Machine.config

- Configuración del servidor
  - Conjunto por defecto de secciones de configuración
  - Heredado por todas las aplicaciones Web
- Ubicación
  - `C:\WINDOWS\Microsoft.NET\Framework\vn.n.nnnnn\CONFIG\machine.config`

Configuración
Web.config

- Equivalente Web al fichero **`App.config`**
  - Opciones de usuario en la sección **`<applicattionSettings>`**
    - Conjunto pares (clave/valor)
    - Accesibles mediante **`Settings.Default`**
  - Opciones aplicación Web en la sección **`<system.web>`**
    - **`<authentication>`**
    - **`<authorization>`**
    - **`<customErrors>`**
    - **`<globalization>`**
    - **`<httpRuntime>`**
    - **`<sessionState>`**
    - **`<trace>`**
    - …

Configuración
Web.config

- Existe la posibilidad de definir secciones personalizadas
  - **`ConfigurationSettings.GetSection(SECTION NAME)`**
- Puede haber múltiples **`Web.config`** por aplicación Web, pero sólo uno por directorio
  - Configuración se aplica al propio directorio y a sus subdirectorios
  - Configuración subdirectorios sobrescribe configuración heredada
- IIS gestiona **`Web.config`**
  - Restringe acceso: clientes no pueden acceder a su contenido
  - Monitorización
    - Caché, por motivos de eficiencia
    - Actualización periódica
      - Cambios aplicados en cuanto se detectan, sin reiniciar IIS

Configuración
Web.config

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">

 <Es.UDC.DotNet.MiniPortal.Properties.Settings>

      <setting name="UserProfileDAOFactory_providerInvariantName"
            <value>System.Data.SqlClient</value>
      </setting>

<Es.UDC.DotNet.MiniPortal.Properties.Settings>

<system.web>
    <trace enabled="true" localOnly="true" pageOutput="true" />
    <compilation debug="true" />
    <httpRuntime appRequestQueueLimit="100" executionTimeout="600" />
    <sessionState mode="InProc" cookieless="false" timeout="20" />
    <customErrors mode="RemoteOnly" defaultRedirect="/errors/error.html">
        <error statusCode="404" redirect="/errors/pagenotfound.html" />
    </customErrors>
</system.web>

</configuration>
```

Configuración
Global.asax

- Situado en la carpeta raíz de la aplicación Web
  - No accesible a través del Servidor Web
- Declaraciones de eventos y objetos a nivel de aplicación
- Eventos originados a nivel de...
  - Aplicación
  - Sesión
  - Petición
- A diferencia de **`Machine.config`** y **`Web.config`**, **`Global.asax`** no es XML, sino código fuente
  - Compilado antes de la primera petición
  - Cambios detectados sin reiniciar servidor Web

Configuración
Global.asax: orden eventos

- Primera Petición
  - Application_Start
- Primera petición por cada usuario
  - Session_Start
- Cada Petición
  - Application_BeginRequest
  - Application_Authenticate
  - Application_EndRequest
- Error en tiempo de ejecución
  - Application_Error
- Logout de usuario / Timeout de sesión
  - Session_End
- Parada o Reinicio del Servidor Web
  - Application_End

Configuración
Global.asax: uso típico de eventos

- Application_Start
  - Útil para cargar la información de configuración específica de un sitio Web
- Session_Start
  - Inicialización variables de sesión
- Application_BeginRequest
  - Acciones de personalización
  - Texto para ser incluido al principio de cada página
- Applicattion_Authenticate
  - Código adicional a la autenticación: log, comprobación de roles, etc.
- Application_EndRequest
  - Texto para ser añadido al final de cada página

Configuración
Global.asax: uso típico de eventos

- Application_Error
  - Útil para enviar e-mail o escribir al event log cuando un ocurra un error que no pueda ser gestionado
- Session_End
  - Escribir a un log o base de datos los logout de cada usuario
- Application_End
  - Útil para escribir a un log cuando se ha parado una aplicación Web
- Parámetros comunes
  - Object sender
  - EventArgs e

Configuración
Global.asax: ejemplo

```
public class Global : System.Web.HttpApplication
{
    protected void Application_Start(object sender, EventArgs e)
    {
        Application.Lock();
        << ... >>
        Application.UnLock();
    }

    protected void Application_End(object sender, EventArgs e)
    {

    }

    protected void Session_Start(object sender, EventArgs e)
    {

    }

    << ... >>

}
```

Gestión del Estado

- Protocolo HTML es *stateless*
- Entonces …
  - ¿Cómo se almacenan datos entre peticiones?
  - ¿Cómo se envían datos de una página a otra?

Gestión del Estado
Lado Cliente

- Cliente solicita (request) una página inicial
- Servidor genera respuesta HTTP (response) que se envía al cliente
  - Respuesta incluye datos (estado)
- Usuario consulta la respuesta y realiza nueva petición
  - Esta segunda petición debe incluir los datos recibidos en la respuesta previamente recibida
- Servidor recibe y procesa datos
  - Servidor puede ser el mismo o diferente en ambas peticiones

Gestión del Estado
Lado Cliente

- Parámetros incluidos en la URL (**`tag <a href="…">`**)
  - Query string
  - Visibilidad elevada
    - Puede no ser conveniente
- Elementos de formularios ocultos **`(<input type="hidden"`**)
  - **`__VIEWSTATE`**
- Cookies
  - Limitaciones
    - Tamaño máximo (4Kb)
    - Número total (300)
    - Cookies por sitio web (20)
  - Usuario puede bloquearlas

Gestión del Estado
Lado Cliente - ViewState

- ViewState puede utilizarse como mecanismo genérico de estado
  - Almacena estado de los controles entre una petición y la siguiente
- Características
  - Puede deshabilitarse a nivel de control o a nivel de página (habilitado por defecto)
    - **`EnableViewState="false"`**
  - Pueden añadirse datos serializables
    - ViewState["variableName"] = variableValue;
  - Problema: Uso de ancho de banda

Gestión del Estado
Lado Cliente - Cookies

- Almacenan datos en el navegador del cliente
- Creación

```
HttpCookie cookie =
    new HttpCookie("loginName", UserProfile.LoginName);
cookie.Expires = DateTime.Now.AddDays(30);
Response.Cookies.Add(cookie);
```

- Lectura

```
HttpCookie cookie = Request.Cookies["loginName"];
```

- Borrado

```
HttpCookie cookie = Request.Cookies["loginName"];
cookie.Expires = DateTime.Now.AddDays(-1);
Response.Cookies.Add(cookie);
```

Cookies
Ejemplo

- Registro de la última visita

```
protected void Page_Load(object sender, EventArgs e)
{
    String lastVisit;

    if (Request.Cookies["lastVisit"] == null)
    {
        lblWelcome.Text = "Welcome. This is your first visit today";
    }
    else
    {
        lastVisit = Request.Cookies["lastVisit"];
        lblWelcome.Text = "Your last visit was on " + lastVisit + ".";
    }

    DateTime time = DateTime.Now;

    Response.Cookies["lastVisit"] = time.ToString();
    Response.Cookies["lastVisit"].Expires = time.AddDays(1);
}
```

Gestión del Estado
Lado Cliente - Cookies

- Propiedades
  - **`Domain`**
    - Servidor del que se descargó la cookie
  - **`Expires`**
    - Fecha (objeto **`DateTime`**) en la que el navegador borrará la cookie
  - **`Name`**
    - Nombre de la cookie
  - **`Value`**
    - Contenido de la cookie

Gestión del Estado
Lado Servidor

- Variables de Aplicación
  - Compartidas entre todas las sesiones
- Variables de Sesión
  - Accesibles sólo al propietario de la sesión
  - Requieren envío de SessionID
- Almacenamiento persistente
  - ASP.NET State Service
  - Base de Datos

Gestión del Estado
Variables de Aplicación

- Estado aplicación se almacena en una instancia de la clase **`HttpApplicationState`**
- Accesible a través de la propiedad **`Page.Application`**
  - Colección pares (clave, valor)
  - Ejemplo acceso: **`Application["Languages"];`**
- Modificación en entornos concurrentes
  - **`Application.Lock`** antes de actualizar
  - **`Application.Unlock`** después de actualizar
- Inicialización a través del fichero **`Global.asax`**

Gestión del Estado
Variables de Sesión

- ¿Qué es una sesión?
  - Contexto en el que un usuario se comunica con un servidor a través de múltiples peticiones HTTP
- Problemas
  - HTTP es no orientado a estado (stateless)
  - HTTP es no orientado a sesiones (sessionless)
- Concepto de sesión manejado a nivel de programación
- Estado aplicación se almacena en una instancia de la clase **`HttpSessionState`**
- Accesible a través de la propiedad **`Page.Session`**
  - Colección pares (clave, valor)
  - Ejemplo acceso: **`Session["loginName"];`**

Gestión del Estado
Variables de Sesión

- Conceptos involucrados con el manejo de la sesión
  - Session identifier: cadena ascii de 120 bits
  - Session events: `Session_OnStart`, `Session_OnEnd`
  - Session variables: almacenamiento datos entre peticiones
- Por defecto, se almacena en una cookie generada automáticamente
  - ASP.NET_SessionID
- Opcionalmente puede gestionarse a través de la propia URL
  - No requiere cambios en el código aplicación.
    - Links relativos siguen funcionando (**`<a href>`**)
    - Redirecciones mediante **`HttpResponse.ApplyAppPathModifier`**
  - Ejemplo: **`http://server/site/(uqwfp455t2qav155)/default.aspx`**
  - Web.Config:

```
<configuration>
    <sessionState cookieless="UseUri" timeout="20"/>
</configuration>
```

Gestión del Estado
Variables de Sesión

- Propiedades objeto Session
  - **`Count`**
    - Número de pares (clave, valor) almacenados
  - **`Keys`**
    - Conjunto de las claves almacenadas en la sesión
  - **`IsNewSession`**
    - Indica si la sesión se ha creado durante la carga de la página actual
  - **`SessionID`**
    - Identificador de sesión
  - **`Timeout`**
    - Máximo número de minutos durante los que la sesión puede permanecer inactiva antes de ser eliminada
    - Tiempo por defecto: 20 minutos

Gestión del Estado
Transferencia de Control entre Páginas

- Hipervínculo
- Postback
- **`Response.Redirect`**
  - Origina redirección HTTP
  - Indica al navegador una nueva dirección de destino
- **`Server.Transfer`**
  - Similar a una redirección, pero internamente en el servidor
- **`Server.Execute`**
  - Ejecuta una página y devuelve el control a la página origen
  - Ambas páginas deben pertenecer al mismo servidor

Gestión del Estado
Transferencia de Control entre Páginas

- **`Response.Redirect`**
  - Indica al servidor Web que cambie a otra página
    - **`Response.Redirect("/MainPage.aspx");`**
  - Dirección destino puede ser externa
    - **`Response.Redirect("http://www.google.com/");`**
  - Crea una nueva request

Gestión del Estado
Transferencia de Control entre Páginas

- **`Server.Transfer`**
  - Redirecciona al usuario a una nueva página
    - `Server.Transfer("/MainPage.aspx");`
  - Mantiene recursos
    - Request se mantiene
    - Únicamente se transfiere el control a una nueva página del servidor
  - Dirección destino ha de ser interna al servidor
  - Mantiene la URL original en el navegador

Gestión del Estado
Transferencia de Control entre Páginas

- **`Server.Execute`**
  - Flujo de información
    - Detiene la carga de la página original
    - Inicia la carga de la página indicada como parámetro
    - Continúa la carga de la página original
  - Propiedades
    - Ambas páginas (original y la empleada como parámetro) han de ser internas al servidor
    - Se mantiene estado de los objetos entre las páginas

Autenticación

- Proceso mediante el que se validan las credenciales de usuario
- Objetivo: controlar acceso a recursos
- Niveles:
  - IIS
  - ASP.NET
    - Configurable desde web.config

Autenticación
Autenticación Internet Information Server

- Configurar IIS en el modo de autenticación elegido
  - Anónimo
  - Básica
  - Digest
  - Certificados Digitales
  - Integrada
- Configurar ASP.NET mediante web.config

```
<system.web>
      <authentication mode="Windows" />
</system.web>
```

Autenticación
Autenticación ASP.NET

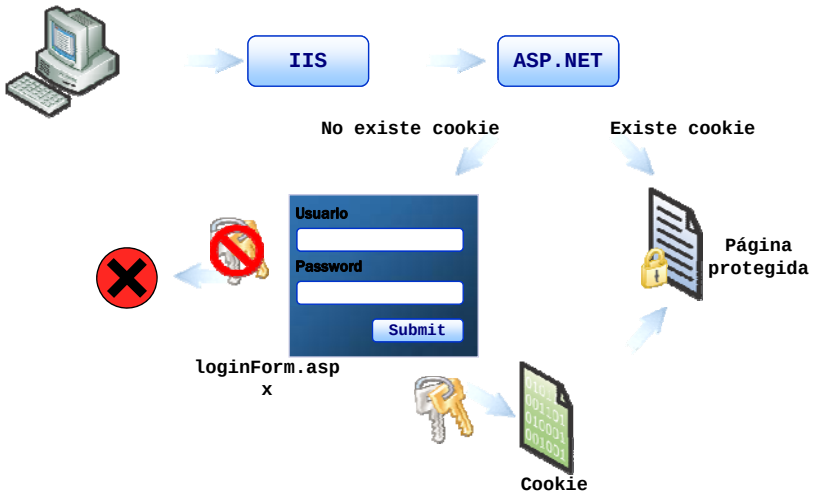
- Modos de autenticación
  - None
    - No se realiza autenticación
    - Acceso anónimo permitido a toda la aplicación Web
  - Windows
    - Delega autenticación en IIS
  - Forms
    - Autenticación basada en formularios
  - Passport
    - Autenticación a través del servicio Web MS Passport

Autenticación
Autenticación ASP.NET

- Basada en formularios web
  - Página de autenticación
  - Cookie de autorización (authentication ticket)
- Configuración
  - Establecer autenticación IIS como anónima
  - Web.config

```
<system.web>
    <authentication mode="Forms">
        <Forms loginURL="/loginForm.aspx" name="loginForm"/>
    </authentication>
</system.web>
```

Autenticación
Autenticación ASP.NET



Autenticación
Autenticación ASP.NET

- Validación Usuario: personalizada o integrada en web.config
- Redirección automática
  - **`FormsAuthentication.RedirectFromLoginPage(string userName, bool createPersistentCookie)`**
  - Dirección destino una vez realizada la autenticación
    - Solicitud de página protegida : se redirecciona a esta página
    - Solicitud de página de login : se redirecciona a defaultURL (configurado en Web.config)
- Redirección personalizada
  - **`Response.Redirect, Server.Transfer, Server.Execute`**
  - **`FormsAuthentication.SetAuthCookie`**
  - **`FormsAuthentication.GetAuthCookie`**
- Revocación del authentication ticket
  - **`FormsAuthentication.SignOut()`**

Autenticación
Autenticación ASP.NET

- ¿Qué ocurre si cliente deshabilita las cookies?
  - URL rewriting: authentication ticket en la propia URL
- Ejemplo:

```
<system.web>
    <authentication mode="Forms">
    <forms name=".ASPXAUTH"
        loginUrl="/Authentication.aspx"
        timeout="30"
        path="/"
        defaultUrl="/MainPage.aspx"
        cookieless="UseUri" />
    </authentication>
</system.web>
```

Autorización

- Es posible controlar de forma declarativa a qué recursos del sitio Web tienen acceso los usuarios
  - Se configura en el web.config
  - e.g. sólo se permite acceso al sitio Web a los usuarios autenticados

```
<authorization>
    <deny users= "?" />
</authorization>
```

- Comodines
  - * todos los usuarios
  - ? Usuarios anónimos

Autorización

- Es posible definir control de acceso para recursos específicos, mediante una sección **`<location>`**
  - e.g. se concede permiso de acceso a la página **`Register.aspx`** a todos los usuarios

```
<location path="Register.aspx">
    <system.web>
        <authorization>
            <allow users ="*" />
        </authorization>
    </system.web>
</location>
```

Páginas de Error

- Ejecución aplicación web puede originar excepciones
  - Controladas: **`IncorrectPasswordException`**, **`DuplicateInstanceException`**,...
  - No Controladas
    - Originadas por algún tipo de error interno
      - Acceso a base de datos
      - etc.
    - Encapsuladas como excepciones **`InternalErrorException`**
  - Errores HTML
    - Página no encontrada
    - Error Servidor
    - Etc.
- Página de Error
  - Página a la que se redirecciona la aplicación en caso de ocurrir una excepción no controlada
- Se pueden definir en dos niveles:
  - A nivel de página: atributo **`PageError`**
  - A nivel de aplicación: sección **`customErrors`** del Web.config

Páginas de Errores
Nivel de Página

- Atributo PageError
  - Especifica que página se muestra cuando la página actual origina una excepción no capturada
- Ejemplo

```
<%@ Page Language="C#" CodeBehind="Register.aspx.cs"
        Inherits="Es.UDC.DotNet.MiniPortal.HTTP.HTML.Register"
        PageError="InternalError.aspx" %>
```

Páginas de Errores
Nivel de Aplicación

- Ejemplo web.config

```
<customErrors mode="RemoteOnly"
              defaultRedirect="InternalError.aspx">

    <error statusCode="403" redirect="NoAccess.htm" />
    <error statusCode="404" redirect="FileNotFound.htm" />

</customErrors>
```

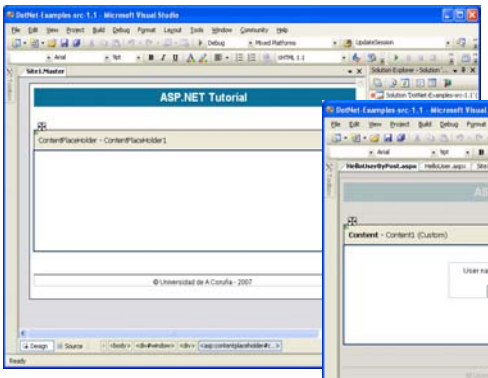
- Opciones atributo **mode**
  - On
    - Habiltita los errores personalizados
    - Si no se especifica el atributo **defaultRedirect**, los usuarios verán un error genérico.
  - Off
    - Deshabilita los errores personalizados
    - Esto permite mostrar los errores detallados estándar
  - RemoteOnly
    - Especifica que los errores personalizados sólo deben mostrarse en los clientes remotos y que los errores de ASP.NET se muestren en el host local.
    - Éste es el valor predeterminado

Master Pages

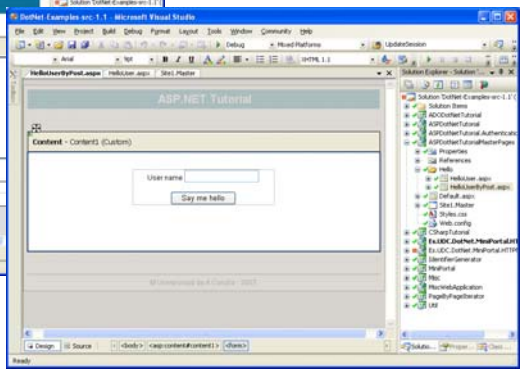
- ASP.NET 1.x carecía de sistema de plantillas
  - Solución: controles de usuario
- ASP.NET 2.0: "Master Pages"
  - Las Master Pages (páginas maestras) permiten crear un diseño común, que será compartido por varias Content Pages (páginas de contenido)
  - Solución más elegante al problema de definir un "look and feel" común

Master Pages

Página Maestra (Master Page)



Página de Contenido (Content Page)



Master pages
Página maestra

```
<%@ Master Language="C#" AutoEventWireup="true" Codebehind="Site1.master.cs"
    Inherits="ASPDotNetTutorialMasterPages.Site1" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>ASP.NET Tutorial</title>
    <link href="Styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div id="window">
        <!-- Page title. -->
        <div id="pageTitle">ASP.NET Tutorial</div>
        <!-- Body content. -->
        <div>
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
            </asp:ContentPlaceHolder>
        </div>
        <!-- Footer. -->
        <div id="footer">&copy; Universidad de A Coruña - 2007</div>
    </div>
</body>
</html>
```

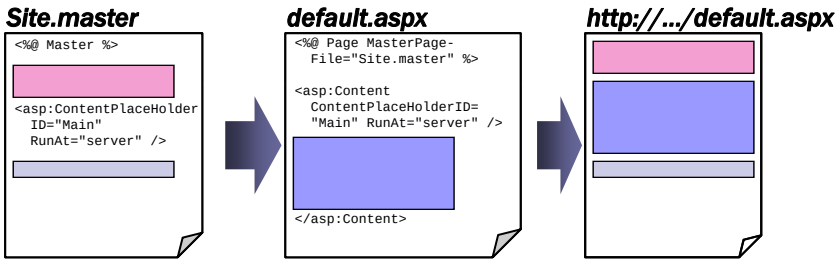
Master pages
Página de contenido

```
<%@ Page Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
         Codebehind="HelloUserByPost.aspx.cs"
         Inherits="ASPDotNetTutorialMasterPages.Hello.HelloUserByPost"
         Title="HelloUserByPost" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
     runat="server">
    <form method="POST" action="HelloUser.aspx">
        <div id="form">
            User name
            <input type="text" name="userName" size="16" maxlength="16" />
            <br />
            <br />
            <input type="submit" value="Say me hello" />
        </div>
    </form>
</asp:Content>
```

Master Pages

- Las páginas maestras definen el contenido común y las zonas de contenido variable (**<asp:ContentPlaceHolder>**)
  - Es posible definir varias zonas de contenido variable en una misma página maestra
- Las páginas de contenido hacen referencia a las páginas maestras y rellenan las zonas de contenido variable (**<asp:Content>**)



Master Pages
Contenido por defecto

- Los controles **ContentPlaceHolder** pueden definir contenido por defecto
- El contenido por defecto se muestra únicamente si la página de contenido no lo sobrescribe

```
<%@ Master %>
...
<asp:ContentPlaceHolder ID="Main" runat="server">
This is default content that will appear in the absence of a
matching Content control in a content page
<asp:ContentPlaceHolder>
```

Master Pages

Aplicar una página maestra a un sitio Web

- Es posible designar la página maestra de todas las páginas pertenecientes a un mismo sitio Web
  - Archivo **`Web.config`**

```
<configuration>
   <system.web>
      <pages masterPageFile="~/MasterPage.master" />
   </system.web>
</configuration>
```

- Todas las páginas **que tengan controles Content** se combinarán con la página principal especificada
- Se asegura que todas las páginas del sitio Web seguirán el diseño de la página maestra, incluso aquéllas que carezcan de las directivas **`@ Page MasterPageFile`**
- Las páginas maestras establecidas utilizando las directivas **`@ Page MasterPageFile`** en los archivos **`.aspx`** sobrescriben las páginas maestras designadas en el archivo Web.config

Master Pages

Página maestra según el navegador

- Es posible seleccionar automáticamente una página maestra dependiendo del navegador
- Ejemplo:

```
<%@ Page Language="C#" MasterPageFile="~/General.Master"
Mozilla:MasterPageFile="~/GeneralMozilla.Master"
Opera:MasterPageFile="~/GeneralOpera.Master" %>
```

- Navegadores
  - AvantGo, Default, Docomo, Ericsson, IE, Jphone, MME, Mozilla, Netscape, Nokia, Openwave, Opera, Panasonic, Pie, Webtv

Trazas

- ASP.NET soporta traceado de páginas asp
  - Sencillo incluir sentencias de debug
    - No necesario debug mediante Response.WriteLine()
  - Sentencias debug independientes de la habilitación de la traza
- Visualización de la traza en el propio navegador
- Habilitación
  - Nivel de página
  - Nivel de aplicación
  - Traza a nivel de página prevalece sobre nivel de aplicación

Trazas

- Objeto Trace
  - Tipo: System.Web.TraceContext
  - Accesible a través de objeto Page
- Métodos
  - `Trace.Write`:
    - Mensajes Informativos
  - `Trace.Warn`:
    - Mensajes de Error
- Propiedades
  - `Trace.IsEnabled`:
    - True si la traza está habilitada para la aplicación o la página actual
  - `Trace.Mode`
    - `SortByTime`
    - `SortByCategory`

Trazas
Habilitación a nivel de página

- Directiva Trace

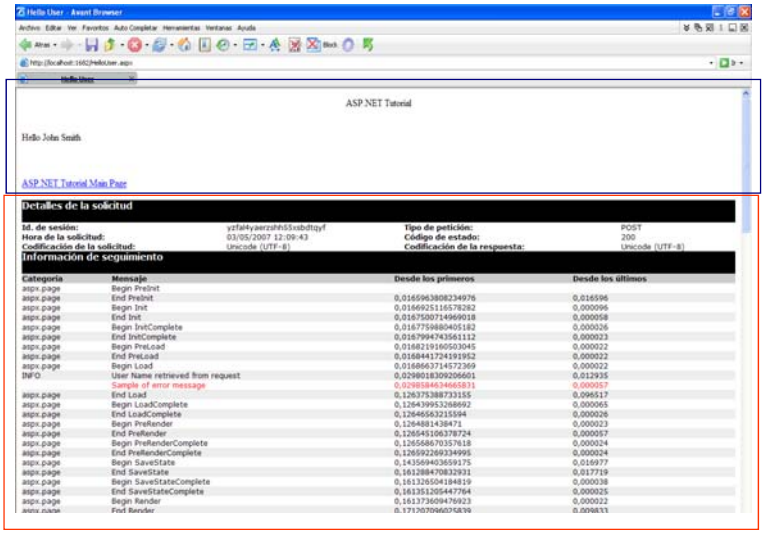
```
<%@ Page trace="true" localOnly="true"%>
```

- Incluir sentencias de traza

```
Trace.Write("INFO", "UserName retrieved");
Trace.Warn("Sample of Error Message");
```

- Acceder página .aspx desde el navegador
- Visualizar elementos de traza

Trazas
Habilitación a nivel de página



Trazas

- Información contenida en una traza
  - Detalles Solicitud
    - SessionID, Get/Post, Hora, …
  - Información Seguimiento
    - Listado llamadas a métodos
  - Árbol de Control
    - Controles incluidos en la página
  - Estado Sesión
    - Variables de sesión
  - Estado Aplicación
    - Variables de Aplicación
  - Cookies Solicitud
  - Cookies Respuesta
  - Cabeceras
    - Headers HTTP
  - Formularios
    - Colección formularios
  - Cadenas de Consulta
    - Query String
  - Variables del Servidor
    - SERVER_NAME , LOGON_USER, …

Trazas
Habilitación a nivel de aplicación

- Configuración a través de web.config

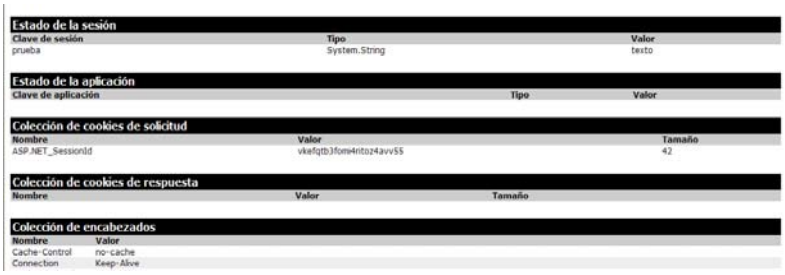
```
<configuration>
    <system.web>
        <trace enabled="true"/>
    </sytem.web>
</configuration>
```

- Llamada a páginas .aspx
- Consulta de la traza
  - **http://sitename/AppName/trace.axd**

Trazas
Habilitación a nivel de aplicación

- **`http://localhost/trace.axd`**

- Detalle



Trazas
Habilitación a nivel de aplicación

- Opciones configuración traza
  - enabled
  - requestLimit
    - Limita el almacenamiento de las trazas a un número especificado
  - pageOutput
    - Muestra información de traza en la propia página
    - Equivalente a traza a nivel de página
  - traceMode
    - Visualización por tiempo (SortByTime) o por categoría (SortByCategory)
  - localOnly
    - Traza únicamente visible en la máquina local