TP2 - Laboratorio 06

1. ABM de Usuarios con Capa de Presentación Web

Objetivos
Agregar a la solución del Trabajo Practico 2 una Capa de Presentación Web para el ABM de Usuarios.

Duración Aproximada
60 minutos

Setup

1. Ir a la carpeta C:\Net\Labs y crear la carpeta TP2L06.

2. Copiar la carpeta TP2 que se encuentra en C:\Net\Labs\TP2L05\ a C:\Net\Labs\TP2L06\
   De esta forma modificaremos esta copia manteniendo la versión anterior intacta.

3. La base de datos a utilizar es la misma que usamos en el laboratorio de la unidad 5.

4. Abrir la solución existente, remover el Sitio Web actual y eliminarlo del File System de Windows.

5. Crear en lugar del Sitio Web que se acaba de eliminar una Aplicación Web con el nombre UI.Web.

   Nota: Visual Studio soporta dos tipos de proyectos web, proyectos Sitio Web (Web Site Project) o Proyectos Aplicación Web (Web Application Project). En este laboratorio vamos a utilizar un proyecto del tipo Aplicación Web porque este tipo de proyectos es el que por diferentes motivos logro la mayor aceptación dentro de la comunidad de desarrolladores .NET. Si están utilizando Visual Studio 2005 es necesario que tengan instalado el Service Pack 1 para poder utilizar los proyectos del tipo Aplicación Web.

Configuración

6. Modificar el web.config para que incluya la información de conexión a la Base de Datos.

```
<connectionStrings>
  <add name="ConnStringLocal" providerName="System.Data.SqlClient"
       connectionString="Server=localhost;Database=tp2_net;Integrated
Security=false; User=net; Password=net;"/>
  <add name="ConnStringExpress" providerName="System.Data.SqlClient"
       connectionString="Server=localhost\SQLEXPRESS;Database=tp2_net; Integrated
Security=false; User=net; Password=net;"/>
  <add name="ConnStringServerISI" providerName="System.Data.SqlClient"
       connectionString="Server=serverisi; Database=tp2_net; Integrated
Security=false; User=net; Password=net;"/>
    </connectionStrings>
```

Master Page, Páginas y Navegación

7. Eliminar el formulario web Default.aspx

8. Agregar una nueva Master Page, nombrarla Site.Master

9. Agregar un nuevo Sitemap al sitio, nombrarlo Web.sitemap

10. Modificar el Site.Master para que quede de la siguiente manera:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs"
Inherits="UI.Web.Site" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>Academia</title>
  </head>
  <body>
    <form id="bodyForm" runat="server">
      <asp:Menu runat="server" ID="menu" DataSourceID="SiteMapDataSource">
      </asp:Menu>
      <asp:SiteMapDataSource ID="SiteMapDataSource" runat="server" />
      <div>
        <asp:ContentPlaceHolder ID="bodyContentPlaceHolder" runat="server">
        </asp:ContentPlaceHolder>
      </div>
    </form>
  </body>
</html>
```

11. Crear dos nuevos Web Content Form que utilicen la Master Page que acabamos de modificar. Nombrarlos Default.aspx y Usuarios.aspx y modificar sus títulos a "Home" y "Usuarios" respectivamente.

12. Modificar el Sitemap para que incluya los dos Web Content Form creado en el paso anterior.

13. Probar que el menú funcione correctamente.

Grilla

14. Modificar Usuarios.aspx incluyendo un Panel y un GridView con la siguientes propiedades en el Content:

```
<asp:Panel ID="gridPanel" runat="server">
  <asp:GridView ID="gridView" runat="server" AutoGenerateColumns="False"
    SelectedRowStyle-BackColor="Black"
    SelectedRowStyle-ForeColor="White"
    DataKeyNames="ID" >
    <Columns>
      <asp:BoundField HeaderText="Nombre" DataField="Nombre" />
      <asp:BoundField HeaderText="Apellido" DataField="Apellido" />
      <asp:BoundField HeaderText="EMail" DataField="EMail" />
      <asp:BoundField HeaderText="Usuario" DataField="NombreUsuario" />
      <asp:BoundField  HeaderText="Habilitado" DataField="Habilitado" />
      <asp:CommandField SelectText="Seleccionar" ShowSelectButton="True" />
    </Columns>
  </asp:GridView>
</asp:Panel>
```

15. Agregar referencias a los Proyectos Business.Logic y Business.Entities y los Using correspondientes en Usuarios.aspx.

16. Incluir la siguiente propiedad en Usuarios.aspx para tener siempre disponible la Business Logic de Usuario mientras usemos el formulario.

```
UsuarioLogic _logic;
private UsuarioLogic Logic
{
  get
  {
    if (_logic==null)
    {
      _logic=new UsuarioLogic();
    }
    return _logic;
  }
}
```

17. Incluir la siguiente función

```
private void LoadGrid()
{
  this.gridView.DataSource = this.Logic.GetAll();
  this.gridView.DataBind();
}
```

18. Modificar el evento Page Load para que incluya un llamado a la función que acabamos de agregar solo en el caso en que la página sea llamada por primera vez.

19. Probar que la Grilla funcione correctamente.

Formulario

20. Agregar los controles necesarios para editar los datos de un Usuario dentro de un nuevo Panel inicializado como invisible debajo del Panel que contiene a la Grilla:

```
<asp:Panel ID="formPanel" Visible="false" runat="server">
  <asp:Label ID="nombreLabel" runat="server" Text="Nombre: "></asp:Label>
  <asp:TextBox ID="nombreTextBox" runat="server"></asp:TextBox>
  <br />
  <asp:Label ID="apellidoLabel" runat="server" Text="Apellido: "></asp:Label>
  <asp:TextBox ID="apellidoTextBox" runat="server"></asp:TextBox>
  <br />
  <asp:Label ID="emailLabel" runat="server" Text="EMail: "></asp:Label>
  <asp:TextBox ID="emailTextBox" runat="server"></asp:TextBox>
  <br />
  <asp:Label ID="habilitadoLabel" runat="server" Text="Habilitado: "></asp:Label>
  <asp:CheckBox ID="habilitadoCheckBox" runat="server" /><br />
  <asp:Label ID="nombreUsuarioLabel" runat="server" Text="Usuario: "></asp:Label>
  <asp:TextBox ID="nombreUsuarioTextBox" runat="server"></asp:TextBox>
  <br />
  <asp:Label ID="claveLabel" runat="server" Text="Clave: "></asp:Label>
  <asp:TextBox ID="claveTextBox" TextMode="Password" runat="server"></asp:TextBox>
  <br />
  <asp:Label ID="repetirClaveLabel" runat="server" Text="Repetir Clave:">
  </asp:Label>
  <asp:TextBox ID="repetirClaveTextBox" TextMode="Password" runat="server">
  </asp:TextBox>
  <br />
</asp:Panel>
```

Acciones

21. Agregar los botones que van a realizar acciones sobre la Grilla en un nuevo Panel debajo de la misma:

```
<asp:Panel ID="gridActionsPanel" runat="server">
  <asp:LinkButton ID="editarLinkButton" runat="server">Editar</asp:LinkButton>
  <asp:LinkButton ID="eliminarLinkButton" runat="server">Eliminar</asp:LinkButton>
  <asp:LinkButton ID="nuevoLinkButton" runat="server">Nuevo</asp:LinkButton>
</asp:Panel>
```

22. Agregar los botones que van a realizar acciones sobre el Formulario en otro Panel debajo del último TextBox:

```
<asp:Panel ID="formActionsPanel" runat="server">
  <asp:LinkButton ID="aceptarLinkButton" runat="server">Aceptar</asp:LinkButton>
  <asp:LinkButton ID="cancelarLinkButton" runat="server">Cancelar</asp:LinkButton>
</asp:Panel>
```

Lógica de Presentación - Propiedades

23. Incluir la siguiente Propiedad junto con una Enumeración de sus posibles valores para definir los estados en los cuales puede estar el Formulario.

```
public enum FormModes
{
  Alta,
  Baja,
  Modificacion
}

public FormModes FormMode
{
  get { return (FormModes)this.ViewState["FormMode"]; }
  set { this.ViewState["FormMode"]= value; }
}
```

Nota: La propiedad se almacena dentro del ViewState de la página porque va a ser necesario mantenerla almacenada entre Postbacks.

24. Agregar una propiedad para almacenar la Entidad que se está editando, otra que contenga el ID seleccionado actualmente y finalmente una que indique si actualmente existe un registro seleccionado.

```
private Usuario Entity
{
  get;
  set;
}

private int SelectedID
{
  get
  {
    if (this.ViewState["SelectedID"] != null)
    {
      return (int)this.ViewState["SelectedID"];
    }
    else
    {
      return 0;
    }
  }
  set
  {
    this.ViewState["SelectedID"] = value;
  }
}

private bool IsEntitySelected
{
  get
  {
    return (this.SelectedID != 0);
  }
}
```

Lógica de Presentación – Mostrar un Usuario Existente para Modificarlo

25. Agregar un Event Handler para el evento SelectedIndexChanged de la Grilla de manera de ir almacenando el ID seleccionado cada vez que el usuario selecciona una fila distinta de la Grilla.

```
protected void gridView_SelectedIndexChanged(object sender, EventArgs e)
{
  this.SelectedID = (int)this.gridView.SelectedValue;
}
```

26. Agregar el siguiente método para completar los controles en base a una Entidad Usuario.

```
private void LoadForm(int id)
{
  this.Entity = this.Logic.GetOne(id);
  this.nombreTextBox.Text = this.Entity.Nombre;
  this.apellidoTextBox.Text = this.Entity.Apellido;
  this.emailTextBox.Text = this.Entity.EMail;
  this.habilitadoCheckBox.Checked = this.Entity.Habilitado;
  this.nombreUsuarioTextBox.Text = this.Entity.NombreUsuario;
}
```

27. Luego agregar un Event Handler al evento Click de editarLinkButton que utilice el método que acabamos de agregar.

```
protected void editarLinkButton_Click(object sender, EventArgs e)
{
  if (this.IsEntitySelected)
  {
    this.formPanel.Visible = true;
    this.FormMode = FormModes.Modificacion;
    this.LoadForm(this.SelectedID);
  }
}
```

28. Testear si los datos de un Usuario se muestran correctamente al seleccionar uno de la Grilla y hacer Click en Editar.

Lógica de Presentación – Modificar un Usuario

29. Agregar los métodos LoadEntity y SaveEntity

```
private void LoadEntity(Usuario usuario)
{
  usuario.Nombre = this.nombreTextBox.Text;
  usuario.Apellido = this.apellidoTextBox.Text;
  usuario.EMail = this.emailTextBox.Text;
  usuario.NombreUsuario = this.nombreUsuarioTextBox.Text;
  usuario.Clave = this.claveTextBox.Text;
  usuario.Habilitado = this.habilitadoCheckBox.Checked;
}

private void SaveEntity(Usuario usuario)
{
  this.Logic.Save(usuario);
}
```

30. Agregar un Event Handler al Evento Click de aceptarLinkButton

```
protected void aceptarLinkButton_Click(object sender, EventArgs e)
{
  this.Entity = new Usuario();
  this.Entity.ID = this.SelectedID;
  this.Entity.State = BusinessEntity.States.Modified;
  this.LoadEntity(this.Entity);
  this.SaveEntity(this.Entity);
  this.LoadGrid();

  this.formPanel.Visible = false;
}
```

31. Testear si el Usuario se modifica correctamente luego de Aceptar.

Lógica de Presentación – Eliminar un Usuario

32. Agregar el método EnableForm para deshabilitar los controles antes de presentar el formulario (el mismo método también servirá para habilitarlos)

```
private void EnableForm(bool enable)
{
  this.nombreTextBox.Enabled = enable;
  this.apellidoTextBox.Enabled = enable;
  this.emailTextBox.Enabled = enable;
  this.nombreUsuarioTextBox.Enabled = enable;
  this.claveTextBox.Visible = enable;
  this.claveLabel.Visible = enable;
  this.repetirClaveTextBox.Visible = enable;
  this.repetirClaveLabel.Visible = enable;
}
```

33. Agregar un Event Handler al Evento Click de eliminarLinkButton

```
protected void eliminarLinkButton_Click(object sender, EventArgs e)
{
  if (this.IsEntitySelected)
  {
    this.formPanel.Visible = true;
    this.FormMode = FormModes.Baja;
    this.EnableForm(false);
    this.LoadForm(this.SelectedID);
  }
}
```

Nota: Debido a que los controles quedan deshabilitados luego de ingresar al formulario para eliminar un Usuario hay que volver a habilitarlos antes de Editar. Para lograr esto alcanza con agregar una llamada al método EnableForm en el Event Handler de editarLinkButton ya existente.

34. Agregar el método DeleteEntity

```
private void DeleteEntity(int id)
{
  this.Logic.Delete(id);
}
```

35. Modificar el Event Handler de aceptarLinkButton para que maneje la eliminación. Esto implica agregar un Switch para diferenciar que operación realizar en función del estado del Formulario.

```
protected void aceptarLinkButton_Click(object sender, EventArgs e)
{
  switch (this.FormMode)
  {
    case FormModes.Baja:
      this.DeleteEntity(this.SelectedID);
      this.LoadGrid();
      break;
    case FormModes.Modificacion:
      this.Entity = new Usuario();
      this.Entity.ID = this.SelectedID;
      this.Entity.State = BusinessEntity.States.Modified;
      this.LoadEntity(this.Entity);
      this.SaveEntity(this.Entity);
      this.LoadGrid();
      break;
    default:
      break;
  }

  this.formPanel.Visible = false;
}
```

36.Testear si el Usuario se elimina correctamente luego de Aceptar

Lógica de Presentación – Agregar un Usuario

37.Agregar un Event Handler al evento Click de nuevoLinkButton

```
protected void nuevoLinkButton_Click(object sender, EventArgs e)
{
  this.formPanel.Visible = true;
  this.FormMode = FormModes.Alta;
  this.ClearForm();
  this.EnableForm(true);
}
```

38.El Event Handler anterior incluye una llamada al método ClearForm para limpiar los controles. Este método todavía no existe así que también hay que agregarlo

```
private void ClearForm()
{
  this.nombreTextBox.Text = string.Empty;
  this.apellidoTextBox.Text = string.Empty;
  this.emailTextBox.Text = string.Empty;
  this.habilitadoCheckBox.Checked = false;
  this.nombreUsuarioTextBox.Text = string.Empty;
}
```

39.Volver a modificar el Event Handler de aceptarLinkButton, esta vez para que también maneje la creación de un nuevo Usuario.

```
case FormModes.Alta:
  this.Entity = new Usuario();
  this.LoadEntity(this.Entity);
  this.SaveEntity(this.Entity);
  this.LoadGrid();
  break;
```

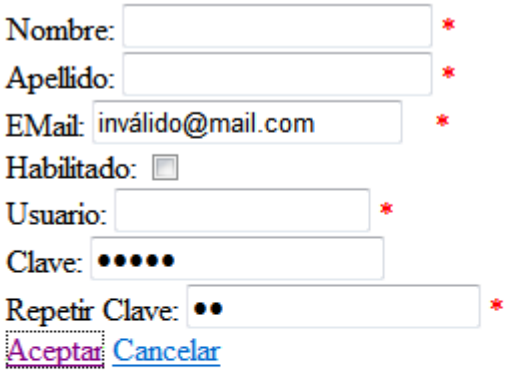
40.Testear si el Usuario se agrega correctamente luego de Aceptar

Lógica de Presentación – Cancelar

41. Agregar un Event Handler al evento Click de cancelarLinkButton para que vuelva a mostrar el Panel de la Grilla.

Validaciones

42. Agregar Validators y un ValidationSummary de manera que el formulario quede de la siguiente manera cuando todas las validaciones son inválidas.



Nota: Evitar que el cancelarLinkButton dispare las validaciones, no es necesario realizarlas en ese caso.

Look and Feel

Modificar la Grilla y el Formulario de edición de Usuarios de manera de lograr un Look & Feel más amigable para el usuario poniendo en práctica los conceptos de Themes y Skins vistos en la teoría.

Reutilización

Evaluar que funcionalidades de Usuarios.aspx podrían llevarse a una clase superior para ser heredadas por todos los ABMs que tengan que implementarse en el TP2.

Fecha	Versión	Autor	Descripción
05/11/2011	1	AJ	Versión Beta