\*\*\*

<form id="form1" runat="server">

<div>

<%--

Выражения <%# %> позволяют привязать свойства элементов управления к защищенным или открытым

полям и свойствам текущей страницы. Для того что бы фактически произвести привязку и извлечь

значения из свойств, нужно вызвать метод DataBind();

--%>

<asp:Literal Text="<%# FirstName %>" runat="server" />

<asp:Label runat="server"

Text="<%# LastName %>"

Font-Size="<%# size %>"

Font-Bold="true" />

</div>

</form>

public partial class \_01\_SingleValueBinding : System.Web.UI.Page

{

protected string FirstName

{

get { return "Jhon"; }

}

protected string LastName

{

get { return "Doe"; }

}

protected int size = 24;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Если не вызвать данный метод, выражения привязки не прочтут значения из свойств страницы.

this.DataBind();

}

}

}

\*\*\*

Привязка нескольких значений

<body>

<form id="form1" runat="server">

<div>

<asp:DropDownList ID="DropDownList1" runat="server"></asp:DropDownList>

</div>

</form>

</body>

protected void Page\_Load(object sender, EventArgs e)

{

string[] values = { "Item 1", "Item 2", "Item 3", "Item 4" };

// Элементы управления, поддерживающие множественную привязку, содержат свойство

// DataSource в которое можно поместить тип IEnumerable

DropDownList1.DataSource = values;

DropDownList1.DataBind();

}

\*\*\*

<body>

<form id="form1" runat="server">

<div>

<asp:DropDownList ID="DropDownList1" runat="server">

</asp:DropDownList>

<asp:Button Text="Отобразить" runat="server" OnClick="Button\_Click" />

<br />

<asp:Label ID="OutputLabel" runat="server" />

</div>

</form>

protected void Page\_Load(object sender, EventArgs e)

{

if (IsPostBack)

{

// При PostBack запросе выполнять привязку данных не нужно.

return;

}

List<BrowserItem> items = new List<BrowserItem>();

items.Add(new BrowserItem() { BrowserId = 1, BrowserName = "Mozilla Firefox" });

items.Add(new BrowserItem() { BrowserId = 2, BrowserName = "Google Chrome" });

items.Add(new BrowserItem() { BrowserId = 3, BrowserName = "Internet Explorer" });

items.Add(new BrowserItem() { BrowserId = 4, BrowserName = "Opera" });

items.Add(new BrowserItem() { BrowserId = 5, BrowserName = "Safari" });

// Установка источника данных.

DropDownList1.DataSource = items;

// Имя свойства одного элемента из источника данных, которое будет использоваться для

// отображения названия пункта в элементе управления.

DropDownList1.DataTextField = "BrowserName";

// Имя свойства для Value элемента в списке.

DropDownList1.DataValueField = "BrowserId";

// Привязка значений.

DropDownList1.DataBind();

}

protected void Button\_Click(object sender, EventArgs e)

{

string text = "Text: " + DropDownList1.SelectedItem.Text + "<br />";

text += "Value: " + DropDownList1.SelectedItem.Value + "<br />";

OutputLabel.Text = text;

}

public class BrowserItem

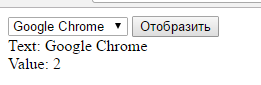
{

public int BrowserId { get; set; }

public string BrowserName { get; set; }

}

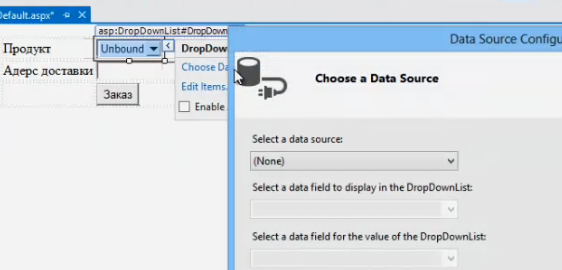
}



\*\*\*

Создать базу – Add – new items – sql database

Базу в App Data



<form id="form1" runat="server">

<div>

<table>

<tr>

<td>Продукт</td>

<td>

<%--В атрибутах DataTextField и DataValueField указаны имена колонок базы данных.--%>

<asp:DropDownList ID="DropDownList1" runat="server"

DataSourceID="ProductsSqlDataSource"

DataTextField="Name"

DataValueField="Id">

</asp:DropDownList>

<%--Источник данных--%>

<asp:SqlDataSource ID="ProductsSqlDataSource"

runat="server"

ConnectionString="<%$ ConnectionStrings:ProductsConnectionString %>"

SelectCommand="SELECT [Id], [Name] FROM [Products]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td>Адрес доставки</td>

<td>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

<asp:Button ID="Button1" runat="server" Text="Заказ" />

</td>

</tr>

</table>

</div>

</form>

<?xml version="1.0"?>

webconfig

<!—

For more information on how to configure your ASP.NET application, please visit

<http://go.microsoft.com/fwlink/?LinkId=169433>

🡪

<configuration>

<connectionStrings>

<add name=”ProductsConnectionString” connectionString=”Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Database1.mdf;Integrated Security=True”

providerName=”System.Data.SqlClient” />

</connectionStrings>

<system.web>

<compilation debug=”true” targetFramework=”4.5” />

<httpRuntime targetFramework=”4.5” />

</system.web>

</configuration>

\*\*\*

<form id="form1" runat="server">

<div>

<table>

<tr>

<td>Категория</td>

<td>

<asp:DropDownList

ID="CategoriesDropDownList"

runat="server"

AutoPostBack="True"

DataSourceID="CategoriesSqlDataSource"

DataTextField="CategoryName"

DataValueField="Id"></asp:DropDownList>

<asp:SqlDataSource ID="CategoriesSqlDataSource"

runat="server"

ConnectionString="<%$ ConnectionStrings:ConnectionString %>"

SelectCommand="SELECT [Id], [CategoryName] FROM [Categories]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td>Товар</td>

<td>

<asp:DropDownList ID="ProductsDropDownList"

runat="server" DataSourceID="ProductsSqlDataSource"

DataTextField="Name"

DataValueField="Id"></asp:DropDownList>

<asp:SqlDataSource ID="ProductsSqlDataSource"

runat="server"

ConnectionString="<%$ ConnectionStrings:ConnectionString %>"

SelectCommand="SELECT [Id], [Name] FROM [Products] WHERE ([CategoryId] = @CategoryId)">

<SelectParameters>

<asp:ControlParameter

ControlID="CategoriesDropDownList"

Name="CategoryId"

PropertyName="SelectedValue"

Type="Int32" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td>Адрес доставки</td>

<td>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

<asp:Button ID="Button1" runat="server" Text="Сделать заказ" />

</td>

</tr>

</table>

</div>

</form>

\*\*\*

DetailsView

<body>

<form id="form1" runat="server">

<div>

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px" AutoGenerateRows="False"

DataKeyNames="ID" DataSourceID="SqlDataSource1" AllowPaging="True">

<%--Каждый элемент BoundField привязывает элемент разметки к данным из источника данных--%>

<Fields>

<asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False" ReadOnly="True" />

<asp:BoundField DataField="Login" HeaderText="Login" />

<asp:BoundField DataField="Password" HeaderText="Password" />

<asp:BoundField DataField="Email" HeaderText="Email" />

</Fields>

</asp:DetailsView>

<%--строка подключения извлекается из файла web.config--%>

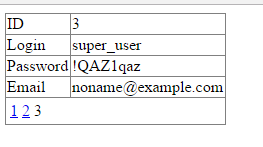
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:DatabaseConnection %>"

SelectCommand="SELECT \* FROM [Users]"></asp:SqlDataSource>

</div>

</form>

</body>



\*\*\*

Insert. Update. Delete

<body>

<form id="form1" runat="server">

<div>

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px"

AutoGenerateRows="False" DataKeyNames="ID" DataSourceID="SqlDataSource1"

AllowPaging="True">

<Fields>

<asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False" ReadOnly="True" />

<asp:BoundField DataField="Login" HeaderText="Login" />

<asp:BoundField DataField="Password" HeaderText="Password" />

<asp:BoundField DataField="Email" HeaderText="Email" />

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" ShowInsertButton="True" />

</Fields>

</asp:DetailsView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"

ConnectionString="<%$ ConnectionStrings:DatabaseConnection %>"

DeleteCommand="DELETE FROM [Users] WHERE [ID] = @ID"

InsertCommand="INSERT INTO [Users] ([Login], [Password], [Email]) VALUES (@Login, @Password, @Email)"

SelectCommand="SELECT \* FROM [Users]"

UpdateCommand="UPDATE [Users] SET [Login] = @Login, [Password] = @Password, [Email] = @Email WHERE [ID] = @ID">

<DeleteParameters>

<asp:Parameter Name="ID" Type="Int32" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="Login" Type="String" />

<asp:Parameter Name="Password" Type="String" />

<asp:Parameter Name="Email" Type="String" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="Login" Type="String" />

<asp:Parameter Name="Password" Type="String" />

<asp:Parameter Name="Email" Type="String" />

<asp:Parameter Name="ID" Type="Int32" />

</UpdateParameters>

</asp:SqlDataSource>

</div>

</form>

\*\*\*

DetailsView Style

<body>

<form id="form1" runat="server">

<div>

<%--AutoGenerateRows="True" позволяет контролу сформировать структуру на основе всех значений, которые возвращает источник данных--%>

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="300" AutoGenerateRows="True"

CellPadding="4" DataKeyNames="ID" AllowPaging="true" DataSourceID="SqlDataSource1"

ForeColor="#333333" GridLines="None">

<%--Элементы определяющие стиль DetailsView--%>

<AlternatingRowStyle BackColor="White" ForeColor="#284775" />

<CommandRowStyle BackColor="#E2DED6" Font-Bold="True" />

<EditRowStyle BackColor="#999999" />

<FieldHeaderStyle BackColor="#E9ECF1" Font-Bold="True" />

<FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />

<RowStyle BackColor="#F7F6F3" ForeColor="#333333" />

</asp:DetailsView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:DatabaseConnection %>"

SelectCommand="SELECT \* FROM [Users]"></asp:SqlDataSource>

</div>

</form>

\*\*\*

ObjectDataSource

<body>

<form id="form1" runat="server">

<div>

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px" DataKeyNames="ID"

AllowPaging="True" AutoGenerateRows="False" DataSourceID="ObjectDataSource1">

<Fields>

<asp:BoundField DataField="Id" HeaderText="Id" />

<asp:BoundField DataField="Name" HeaderText="Name" />

<asp:BoundField DataField="Price" HeaderText="Price" />

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" ShowInsertButton="True" />

</Fields>

</asp:DetailsView>

<!--Представляет бизнес объект, представляющий данные для элемента управления.-->

<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"

DataObjectTypeName="Product"

TypeName="ProductsDatabase"

DeleteMethod="RemoveProduct"

InsertMethod="AddProduct"

SelectMethod="SelectAllProducts"

UpdateMethod="UpdateProduct"></asp:ObjectDataSource>

</div>

</form>

class

public class ProductsDatabase

{

private static List<Product> \_products;

public static List<Product> Products

{

get

{

if (\_products == null)

{

\_products = new List<Product>();

\_products.Add(new Product() { Id = 1, Name = "Mobile phone", Price = 220 });

\_products.Add(new Product() { Id = 2, Name = "Laptop", Price = 999 });

}

return \_products;

}

set { \_products = value; }

}

// Получение всех элементов из источника данных

public List<Product> SelectAllProducts()

{

return Products;

}

// Добавление нового элемента в источник данных.

public void AddProduct(Product p)

{

Products.Add(p);

}

// Удаление элемента из источника данных.

public void RemoveProduct(Product p)

{

Products.Remove(p);

}

// Обновление элемента в источнике данных.

public void UpdateProduct(Product p)

{

int index = Products.IndexOf(p);

Products.Remove(p);

Products.Insert(index, p);

}

}

/// <summary>

/// Класс представляющий одну запись в источнике данных.

/// </summary>

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public double Price { get; set; }

public Product()

{

}

public override bool Equals(object obj)

{

Product temp = obj as Product;

if (temp == null)

{

return false;

}

return temp.Id == this.Id;

}

// При переопределении метода Equals метод GetHashCode желательно переопределить.

public override int GetHashCode()

{

return this.Id;

}

}

\*\*\*

ADO

<body>

<form id="form1" runat="server">

<div>

<!--Чтение одного значения из базы-->

<asp:Button ID="ReadOneValueButton" Text="Чтение одной записи" runat="server" OnClick="ReadOneValueButton\_Click" />

Результат:

<asp:Label ID="ReadOneValueOutput" EnableViewState="false" runat="server" />

<hr />

<!--Построчное считыванеи информации-->

<asp:Button ID="ReadAllButton" Text="Чтение всех записи" runat="server" OnClick="ReadAllButton\_Click" />

Результат:

<br />

<asp:Label ID="ReadAllOutput" EnableViewState="false" runat="server" />

<hr />

<!--Построчное считывание информации-->

Login

<asp:TextBox ID="LoginTextBox" runat="server"></asp:TextBox>

<br />

Passowrd

<asp:TextBox ID="PasswordTextBox" runat="server"></asp:TextBox>

<br />

Email

<asp:TextBox ID="EmailTextBox" runat="server"></asp:TextBox><br />

<asp:Button ID="AddNewEntryButton" Text="Добавить новую запись" runat="server" OnClick="AddNewEntryButton\_Click" />

<asp:Label ID="ErrorOutput" runat="server" EnableViewState="false" />

<hr />

Id

<asp:TextBox ID="IdToRemoveTextBox" runat="server" />

<!--OnClientClick JavaScript функция, которая выполняется на стороне клиента до того как на сервер будет отправлен запрос.

Если функция возвращает значение false - отправка запроса отменяется-->

<asp:Button ID="RemoveByIdButton" Text="Удалить запись по ID" runat="server" OnClientClick="return confirm('Вы уверены, что хотите удалить запись?')"

OnClick="RemoveByIdButton\_Click" />

<asp:Label ID="ErrorOutput2" runat="server" EnableViewState="false" />

</div>

</form>

public partial class \_09\_WorkingWithDatabase : System.Web.UI.Page

{

SqlConnection \_connection;

protected void Page\_Load(object sender, EventArgs e)

{

// Чтение значения строки подключения из web.config из секции <connectionStrings>

string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnection"].ConnectionString;

// Настройка объекта подключения к базе и открытие подключения.

\_connection = new SqlConnection(connectionString);

\_connection.Open();

}

protected void Page\_Unload(object sender, EventArgs e)

{

// При выгрузке страницы из памяти сервера закрываем подключение к базе данных.

if (\_connection != null && \_connection.State != ConnectionState.Closed)

{

\_connection.Close();

}

}

// Чтение одного значения из базы данных.

protected void ReadOneValueButton\_Click(object sender, EventArgs e)

{

try

{

// Создание объекта запроса.

SqlCommand command = new SqlCommand("SELECT Login FROM Users WHERE ID=1", \_connection);

// Чтение первой строки и первой колонки (одного значения) из результата который возвращает запрос.

object result = command.ExecuteScalar();

string login = Convert.ToString(result);

ReadOneValueOutput.Text = login;

}

catch (Exception ex)

{

ReadOneValueOutput.Text = "Error:<br />" + ex.Message;

ReadOneValueOutput.ForeColor = Color.Red;

}

}

// Чтение нескольких строк из базы данных.

protected void ReadAllButton\_Click(object sender, EventArgs e)

{

SqlDataReader reader = null;

try

{

// Создание объекта запроса.

SqlCommand command = new SqlCommand("SELECT \* FROM Users", \_connection);

string result = string.Empty;

// Создание объекта для построчного считывания данных из базы.

reader = command.ExecuteReader();

while (reader.Read())

{

// Получение данных из колонок.

result += "ID = " + Convert.ToString(reader["ID"]) + "; ";

result += "Login = " + Convert.ToString(reader["Login"]) + "; ";

result += "Password = " + Convert.ToString(reader["Password"]);

result += "<br />";

}

ReadAllOutput.Text = result;

}

catch (Exception ex)

{

ReadAllOutput.Text = "Error:<br />" + ex.Message;

ReadAllOutput.ForeColor = Color.Red;

}

finally

{

// Если reader был открыт освобождаем память, закрывая объект.

if (reader != null)

{

reader.Close();

}

}

}

// Добавление новой записи в базу.

protected void AddNewEntryButton\_Click(object sender, EventArgs e)

{

try

{

SqlCommand command = new SqlCommand("INSERT INTO Users (Login, Password, Email)VALUES(@Login, @Password, @Email)", \_connection);

// Инициализация переменных в запросе.

command.Parameters.AddWithValue("Login", LoginTextBox.Text);

command.Parameters.AddWithValue("Password", PasswordTextBox.Text);

command.Parameters.AddWithValue("Email", EmailTextBox.Text);

// Выполнение запроса.

command.ExecuteNonQuery();

}

catch (Exception ex)

{

ErrorOutput.Text = "Error:<br />" + ex.Message;

ErrorOutput.ForeColor = Color.Red;

}

}

// Удаление записи из базы данных.

protected void RemoveByIdButton\_Click(object sender, EventArgs e)

{

try

{

SqlCommand command = new SqlCommand("DELETE FROM Users WHERE ID=@UserId", \_connection);

// Инициализация переменных в запросе.

command.Parameters.AddWithValue("UserID", IdToRemoveTextBox.Text);

// Выполнение запроса. Метод ExecuteNonQuery возвращает количество удаленных в базе строк.

int affectedRows = command.ExecuteNonQuery();

// Добавление в ответ пользователю тега скрипт, в котором с помощью javascript функции alert выводиться сообщение.

string script = string.Format("alert('Удалено {0} строк');", affectedRows);

ClientScript.RegisterClientScriptBlock(this.GetType(), "MessageBox", script, true);

}

catch (Exception ex)

{

ErrorOutput2.Text = "Error:<br />" + ex.Message;

ErrorOutput2.ForeColor = Color.Red;

}

}