
A.D.S.

Análise e
Desenvolvimento
de Sistemas

Disciplina: Script

Sumário

Planejamento	3
Primeiro passo:.....	3
Segundo passo:.....	3
Criando o exemplo acima:	4
Planejamento – Master Page (Layout).....	4
Verificando a versão do .NET Framework.....	5
Definir o padrão de pastas e organizar os arquivos.....	5
Definição.....	5
dll MySqlData na pasta Bin	6
String de Conexão	7
Script - String de Conexão.....	7
Script Completo: Web.Config com a String de Conexão	7
Classe Mapped	8
Criando a classe Mapped	8
Importar as seguintes Bibliotecas	8
Classe Mapped sem comentário	8
Comentando os métodos da classe Mapped	9
Método de Connection- Comentado.....	9
Método Command - Comentado:.....	9
Método Adapter - Comentado	9
Método Parameter – Comentado	9
Criando as classes Básicas	10
Observe que esta classe tem uma chave estrangeira em sua tabela no Banco de Dados	10
Classe de Persistência	11
Persistência da classe Tipo Empresa.....	11
Método Insert.....	11
Método Insert Comentado	11
Testando - Insert.....	12
Método Insert da Tabela Empresa.....	13
Persistência – SELECT * From...	14
Persistência – SELECT * From... Comentado.....	14
Carregar DropDownList	15
Carregar GridView	15
Delete	16
Carregando vários elementos em uma div de maneira dinâmica.....	16
Código ASPX.....	16
Code be Hidden	16
Update.....	17

Disciplina de Script

Update – Comentado	17
SelectId	18
Redirecionamento - Exemplo:	20
Aplicando mascaras nos controles.....	21
Links que perder a referência	22
Código.....	22
Visão do Browser	22
Mensagem de erro	22
Upload Simples	23
Upload - Criando um diretório com o ID da empresa e salvando o arquivo com o id da empresa.....	25
Upload Simples de Imagens.....	27
Modal	30
Chamar Modal Através do Code be Hidden.....	31
Validando formulário	Erro! Indicador não definido.
Gerenciamento de Sessão	41
Utilizando Sessão	41
Exemplo 01:	41
LOGIN	43
Exemplo 02:	43

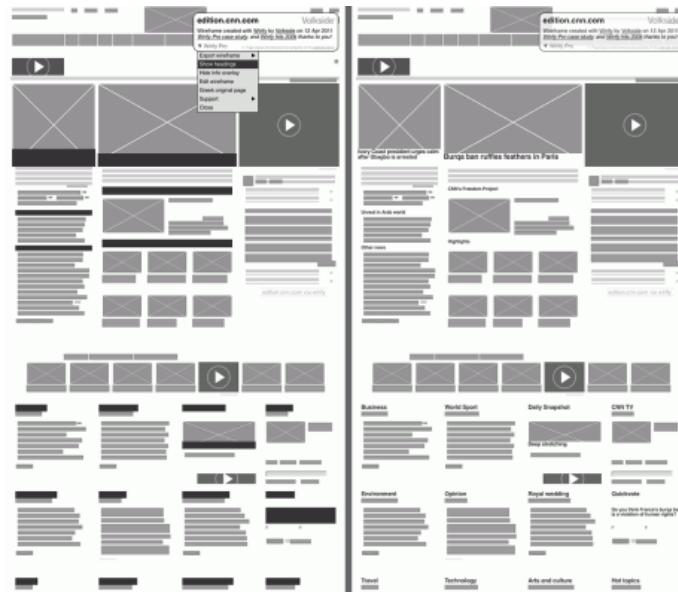
Projeto Interdisciplinar – Roteiro

Planejamento

Primeiro passo:

Levantamento de Requisitos do sistema.

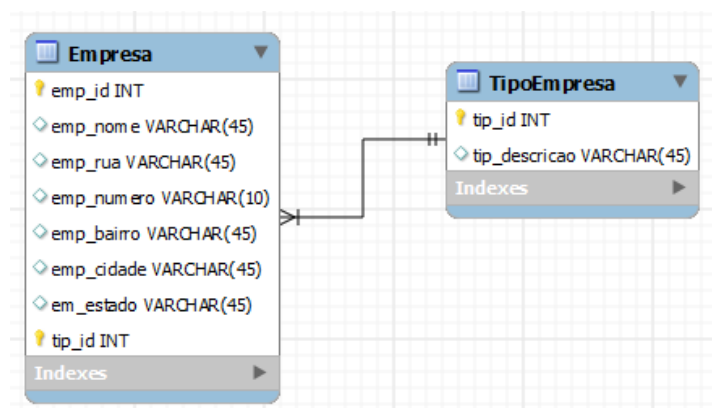
Prototipação do sistema (wireframe).



Segundo passo:

Modelagem e criação do Banco de Dados.

Iremos utilizar o exemplo abaixo, lembrando que este exemplo é meramente ilustrativo. Observe as chaves primárias e a chave estrangeira.



Disciplina de Script

Criando o exemplo acima:

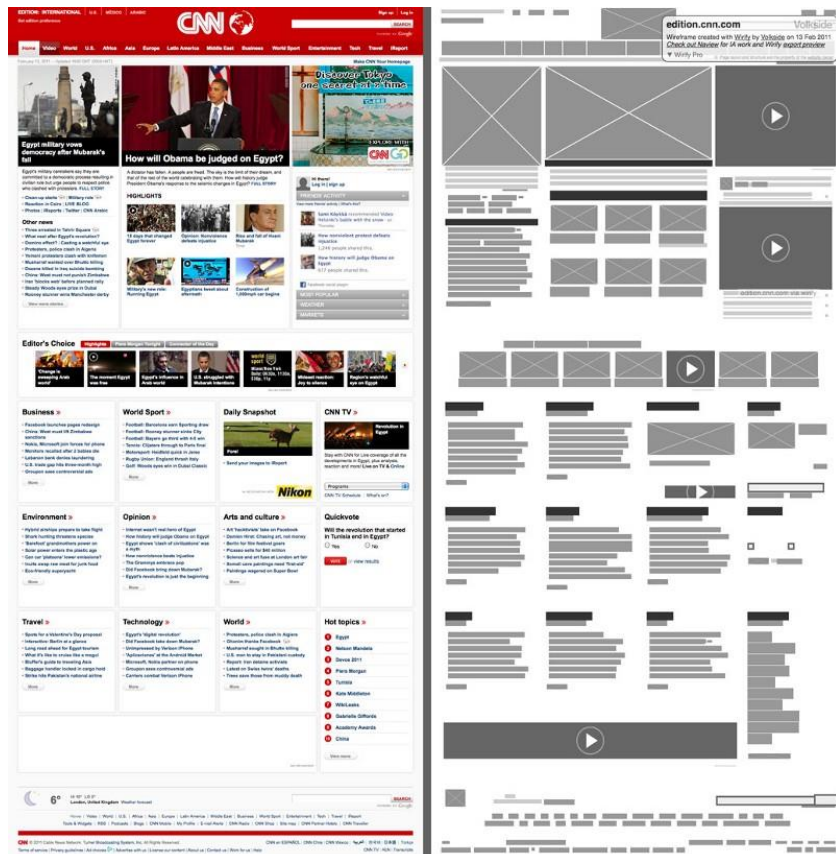
Create database projeto;
use projeto;

```
create table tip_tipoempresa(  
    tip_id integer primary key auto_increment,  
    tip_descricao varchar(120)  
);
```

```
create table emp_empresa(  
    emp_id integer primary key auto_increment,  
    emp_nome varchar(120),  
    emp_rua varchar(120),  
    emp_numero integer,  
    emp_bairro varchar(120),  
    emp_cidade varchar(120),  
    emp_estado varchar(5),  
    tip_id integer,  
    foreign key ( tip_id ) references tip_tipoempresa ( tip_id )  
);
```

Planejamento – Master Page (Layout)

Definir o que será comum à todas as páginas e criar a Master Page (Layout). Isso facilita a reutilização do código e a distribuição das tarefas entre os membros da equipe.

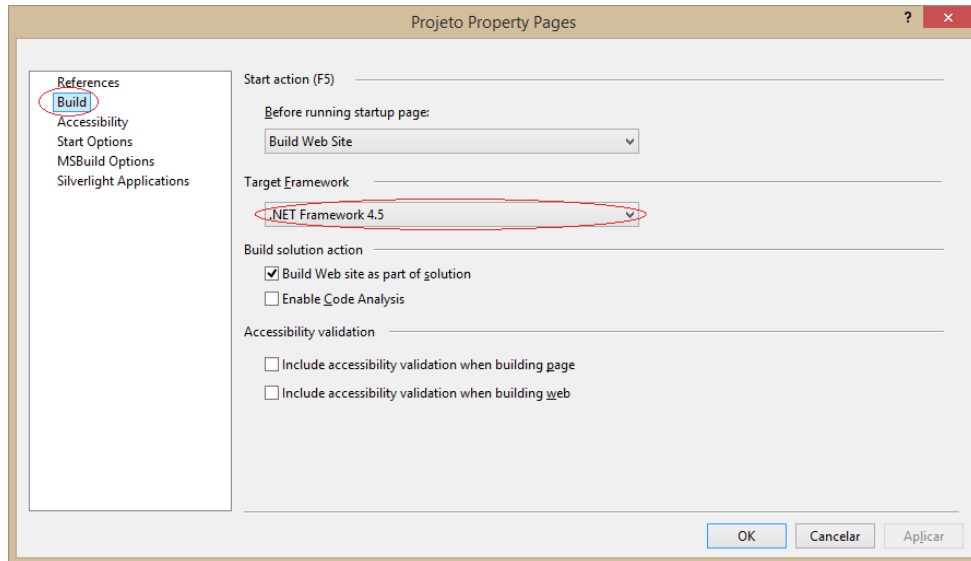


Verificando a versão do .NET Framework

Alterando a versão do .NET Framework

Botão direito sobre o projeto / Property Pages / Build

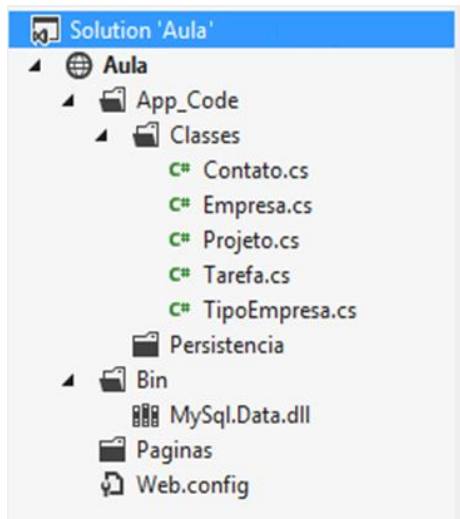
Em: "Target Framework" selecione a versão do .NET Framework desejado e clique em Ok.



Definir o padrão de pastas e organizar os arquivos

Aqui verificamos um exemplo simples.

Novas pastas podem ser criadas de acordo com a necessidade.



Definição

Pasta App_Code:

Armazenará todas as nossas classes, dentro da pasta App_Code criamos as pastas **Classe** e **Persistência**.

Pasta Classe:

As Classes são criadas de acordo com a modelagem do Banco de dados. Atenção para os tipos de dados, eles não podem contrariar o que foi modelado no banco de dados.

Devemos criar uma estrutura básica das classes.

Pasta Persistência:

Armazenará as classes de persistências, que são as classes que trabalharão com o acesso ao banco de dados (Select, Delete, Insert, Update).

Nem toda classe precisa obrigatoriamente de uma classe de persistência.

Pasta Bin:

Contém todas os arquivos de configuração (dll do projeto – por exemplo **MySqlData**).

dll MySqlData na pasta Bin

Este arquivo **MySqlData** é um arquivo de ponte, é ele quem fornece a conexão da aplicação com o banco de dados.

Sem ele não é possível acessar nossa base de dados.

Você pode copiar o arquivo e colar dentro desta pasta Bin.

Ou clique com o botão direito do mouse sobre a pasta Bin / Add References / Browser / Ache o local onde está o arquivo MySqlData.dll. / ok

Pode (ou não) ser exibido uma mensagem avisando que as configurações serão alteradas.

Atenção:

A pasta não pode ficar vazia, em algumas situações, mesmo executando a operação descrita acima a pasta permanece vazia. Se isso acontecer, localize o arquivo copie o cole o arquivo dentro da pasta.

Clique com o botão direito do mouse sobre a pasta Bin e dê um **refresh** para verificar se a pasta contém o arquivo MySqlData.dll.

String de Conexão

Configurar a String de Conexão no Web.Config

No Web.Config nós iremos criar nossa String de Conexão.

A String de conexão contém o nome do usuário do banco, senha do banco, onde o banco está armazenado e qual o nome do database que iremos utilizar.

Cada vez que a aplicação solicitar um serviço do banco de dados, ele vai validar esta String de conexão, se for válida ele acessa o banco de dados, executa suas operações e retorna essa validação.

Script - String de Conexão

```
<appSettings>
  <add key="strConexao" value="Database = projeto; Data Source = localhost; User id = root;
  Password=; pooling = false; "/>
</appSettings>
```

Onde:

key = nome sugestivo da String de Conexão.

Value = Dados para a conexão com o banco de dados.

Database = nome do banco de dados.

Data Source = Onde está armazenado o banco de dados, pode ser o Ip ou o endereço local.

User id = nome do usuário credenciado para acessar o banco de dados.

Password = senha de acesso do MySql para acessar o Banco de Dados.

Pooling = permite trabalhar com vários acessos utilizando a mesma as conexões;

Script Completo: Web.Config com a String de Conexão

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="strConexao" value="Database = projeto; Data Source = localhost; User id = root; Password=;
    pooling = false; "/>
  </appSettings>

  <system.web>
    <compilation debug="false" targetFramework="4.0">
      <assemblies>
        <add assembly="MySql.Data, Version=6.6.3.0, Culture=neutral,
        PublicKeyToken=C5687FC88969C44D"/>
      </assemblies>
    </compilation>
    <httpRuntime/>
  </system.web>
</configuration>
```


Classe Mapped

A classe **Mapped** é uma classe de mapeamento.

Ela pode ter qualquer nome.

Esta é a classe possui os métodos de **conexão**, **comando**, **mapeamento** e **execução** com o banco de dados.

Métodos Comuns:

Método de Conexão - Cria o objeto de Conexão

Método de Comandos SQL - Cria o objeto e valida o comando a ser executado

Método Adapter - Executa o comando

Método de Parametrização - Valida as entradas de dados antes de executar o comando Sql.

Criando a classe Mapped

Importar as seguintes Bibliotecas

```
//As bibliotecas MySql só aparecerão se a referência ao MySqlData estiver correta
using MySql.Data.MySqlClient;
using System.Configuration;
using System.Data;
```

Classe Mapped sem comentário

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using MySql.Data.MySqlClient;
using System.Configuration;
using System.Data;

public class Mapped{

    //Método para abrir a conexão
    public static IDbConnection Connection(){
        MySqlConnection objConexao = new MySqlConnection(ConfigurationManager.AppSettings["strConexao"]);
        objConexao.Open();
        return objConexao;
    }

    // Comandos SQL - Cria o objeto e valida o comando a ser executado
    public static IDbCommand Command(string query, IDbConnection objConexao){
        IDbCommand command = objConexao.CreateCommand();
        command.CommandText = query;
        return command;
    }

    // Funciona como uma ponte entre os dados desconexos e conexos
    public static IDataAdapter Adapter(IDbCommand command){
        MySqlDataAdapter adap = new MySqlDataAdapter();
        adap.SelectCommand = command;
        return adap;
    }

    // Parametrização
    // Valida as entradas de dados antes de executar o comando Sql
    public static IDbDataParameter Parameter(string nomeDoParametro, object valor){
        return new MySqlParameter (nomeDoParametro, valor);
    }
}
```

}

Comentando os métodos da classe Mapped

Método de Connection- Comentado

```
// IDbConnection - Representa uma fonte de conexão com banco de dados
public static IDbConnection Connection() {
    // A linha abaixo cria um objeto de conexão com o MySql
    // Observe que este objeto recebe a string de conexão "strConexao".
    // A string de conexão está dentro do "AppSettings"
    // A AppSettings está dentro do "Configuration" que está o Web.Config
    MySqlConnection objConexao = new
    MySqlConnection(ConfigurationManager.AppSettings["strConexao"]);
    // O objeto abre a conexão com o banco de dados
    objConexao.Open();
    // Executa e retorna a conexão
    return objConexao;
}
```

Método Command - Comentado:

// Comandos SQL - Cria o objeto e valida o comando a ser executado
// IDbCommand - Representa uma instrução SQL que é executado quando conectada a uma fonte de dados, e implementada pelos provedores de dados.NET Framework que acessam bancos de dados relacionais.

```
public static IDbCommand Command(string query, IDbConnection objConexao) {
    // O método recebe uma query que será uma sql a ser executada
    // Para executar um comando no Banco de Dados a Conexão deve estar aberta por isso ele recebe o objConexao
    // O objeto command recebe a conexão e cria um comando na conexão que está sendo executado
    IDbCommand command = objConexao.CreateCommand();
    // O objeto comando.CommandText recebe a query que é uma expressão SQL
    command.CommandText = query;
    //Executa e retorna o comando
    return command;
}
```

Método Adapter - Comentado

IDataAdapter - trabalha como ponte entre dados desconexos e conexos. Após carregado o DataSet, será possível uma manipulação desconexa destes dados em memória.

```
public static IDataAdapter Adapter( IDbCommand command) {
    // O método executa um commando validado, por isso ele recebe um IDbCommand
    IDataAdapter adap = new MySqlDataAdapter();

    // O objeto executa e recebe os dados do Banco de Dados
    adap.SelectCommand = command;
    return adap;
}
```

Método Parameter – Comentado

```
// Valida as entradas de dados antes de executar o comando Sql
public static IDbDataParameter Parameter ( string nomeDoParametro, object valor) {
    // Vai validar as entradas de dados e retornar os valores para serem trabalhado no sistema
    return new SqlParameter(nomeDoParametro, valor);
}
```

Criando as classes Básicas

```
// Classe TipoEmpresa
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class TipoEmpresa {
    private int tip_id;
    private string tip_descricao;
    public string Tip_descricao{
        get { return tip_descricao; }
        set { tip_descricao = value; }
    }
    public int Tip_id {
        get { return tip_id; }
        set { tip_id = value; }
    }
}
```

Observe que esta classe tem uma chave estrangeira em sua tabela no Banco de Dados

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class Empresa{
    private int emp_id;
    private string emp_nome;
    private string emp_rua;
    private string emp_numero;
    private string emp_bairro;
    private string emp_cidade;
    private string emp_estado;

    //Atenção para a chave estrangeira
    private TipoEmpresa tip_id;

    public global::TipoEmpresa Tip_id {
        get { return tip_id; }
        set { tip_id = value; }
    }

    public int Emp_id {
        get { return emp_id; }
        set { emp_id = value; }
    }

    public string Emp_nome {
        get { return emp_nome; }
        set { emp_nome = value; }
    }

    public string Emp_rua {
        get { return emp_rua; }
        set { emp_rua = value; }
    }

    public string Emp_numero {
        get { return emp_numero; }
        set { emp_numero = value; }
    }

    public string Emp_bairro {
        get { return emp_bairro; }
        set { emp_bairro = value; }
    }

    public string Emp_cidade {
        get { return emp_cidade; }
        set { emp_cidade = value; }
    }

    public string Emp_estado {
        get { return emp_estado; }
        set { emp_estado = value; }
    }
}
```

Classe de Persistência

Classes de persistências, que são as classes que trabalharão com o acesso ao banco de dados (Select, Delete, Insert, Update).

Nem toda classe precisa obrigatoriamente de uma classe de persistência.

```
// importar a bibliotecas
using System.Data;
```

Persistência da classe Tipo Empresa

Método Insert

```
public static int Insert(TipoEmpresa tipo) {
    int retorno = 0;
    try{
        IDbConnection objConexao; // Abre a conexao
        IDbCommand objCommand; // Cria o comando
        string sql = "INSERT INTO tip_tipoempresa (tip_descricao) "+"VALUES (?tip_descricao)";
        objConexao = Mapped.Connection();
        objCommand = Mapped.Command(sql, objConexao);
        objCommand.Parameters.Add(Mapped.Parameter("?tip_descricao", tipo.Tip_descricao));
        objCommand.ExecuteNonQuery(); // utilizado quando cdigo não tem retorno, como seria o caso do SELECT
        objConexao.Close();
        objCommand.Dispose();
        objConexao.Dispose();
    }catch (Exception e){
        retorno = -2;
    }
    return retorno;
}
```

Método Insert Comentado

// Insere todos os dados do objeto (Classe TipoEmpresa) no BD é muito mais prático passar um objeto todo de uma vez que passar atributo por atributo

```
public static int Insert(TipoEmpresa tipo){
    // 0 = caso de Sucesso
    // -2 = caso de falha de conexão
    int retorno = 0;

    try{
        IDbConnection objConexao; // Cria obj conexao
        IDbCommand objCommand; // Cria obj comando

        //SINTAXE: "INSERT INTO nomeDaTabela (nomeDasColunas) VALUE
        (?parametrização)";
        string sql = "INSERT INTO tipoempresa (tip_descricao) " +
```

Disciplina de Script

```
"VALUES (?tip_descricao)";

objConexao = Mapped.Connection(); // Abre a conexão

//atribui ao objeto comando a Sql e o objeto de conexão
objCommand = Mapped.Command(sql, objConexao);

// Parametriza os dados - Adiciona os dados para parametrização
objCommand.Parameters.Add(Mapped.Parameter("?tip_descricao", tipo.Tip_descricao));

// Executa o comando
// Comando ExecuteNonQuery - utilizado quando código não tem retorno, como seria o caso de
// SELECT
objCommand.ExecuteNonQuery();

objConexao.Close(); //Fecha a conexão

//Disponibiliza o objeto conexão e o objeto comando para serem utilizados novametne
objCommand.Dispose();
objConexao.Dispose();
}catch (Exception e){
    retorno = -2;
}
return retorno;
}
```

Testando - Insert

Código de Inserção ASPX

```
<asp:Label ID="lblTipoEmpresa" runat="server" Text="Tipo Empresa: "></asp:Label>
<asp:TextBox ID="txbTipoEmpresa" runat="server"></asp:TextBox>
<br />
<asp:Button ID="btnSalvar" runat="server" Text="Salvar" OnClick="btnSalvar_Click" />
<br />
<asp:Label ID="lblConfirmacao" runat="server" Text=""></asp:Label>
```

Tipo Empresa:

Salvar

OK

Código de Inserção ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Paginas_Insert1 : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e){ }

    protected void btnSalvar_Click(object sender, EventArgs e){

        TipoEmpresa tipo = new TipoEmpresa();
        tipo.Tip_descricao = txbTipoEmpresa.Text;
        switch(TipoEmpresaDB.Insert(tipo)){
            case 0:
                lblConfirmacao.Text = "OK";
```

Disciplina de Script

```
        break;
    case -2:
        lblConfirmacao.Text = "ZICOU";
        break;
    }
}
}
```

Método Insert da Tabela Empresa

Deve-se observar que na tabela Empresa temos uma chave estrangeira, esta é a única diferença que encontramos em relação aos outros métodos da persistência.

Os demais métodos seguem o mesmo padrão da Classe de Persistência TipoEmpresaDB.

```
public static int Insert(Empresa empresa){
    int errNumber = 0;
    try{
        IDbConnection objConexao;
        IDbCommand objCommand;
        string sql = "INSERT INTO empresa(";
        sql += "emp_nome,";
        sql += "emp_rua,";
        sql += "emp_numero,";
        sql += "emp_bairro,";
        sql += "emp_cidade,";
        sql += "emp_estado,";
        sql += "tip_id";
        sql += ")";
        sql += " VALUES (";
        sql += "?emp_nome,";
        sql += "?emp_rua,";
        sql += "?emp_numero,";
        sql += "?emp_bairro,";
        sql += "?emp_cidade,";
        sql += "?emp_estado,";
        sql += "?tip_id";
        sql += ")";
        objConexao = Mapped.Connection();
        objCommand = Mapped.Command(sql, objConexao);
        objCommand.Parameters.Add(Mapped.Parameter("?emp_nome", empresa.emp_nome));
        objCommand.Parameters.Add(Mapped.Parameter("?emp_rua", empresa.emp_rua));
        objCommand.Parameters.Add(Mapped.Parameter("?emp_numero",
            empresa.emp_numero));
        objCommand.Parameters.Add(Mapped.Parameter("?emp_bairro", empresa.emp_bairro));
        objCommand.Parameters.Add(Mapped.Parameter("?emp_cidade", empresa.emp_cidade));
        objCommand.Parameters.Add(Mapped.Parameter("?emp_estado", empresa.emp_estado));
        objCommand.Parameters.Add(Mapped.Parameter("?tip_id", empresa.Tip_id.Tip_id));
        // Chave Estrangeira

        objCommand.ExecuteNonQuery();
        objConexao.Close();
        objCommand.Dispose();
        objConexao.Dispose();
    }catch (Exception ex){
        errNumber = -2;
    }
    return errNumber;
}
}
```

```
Public static DataSet SelectAll() {  
  
    DataSet ds = new DataSet();  
    IDbConnection objConnection;  
    IDbCommand objCommand;  
    IDataAdapter objDataAdapter;  
    objConnection = Mapped.Connection();  
    objCommand = Mapped.Command("SELECT * FROM tipoempresa ORDER BY tip_descricao",  
    objConnection);  
    objDataAdapter = Mapped.Adapter(objCommand);  
  
    objDataAdapter.Fill(ds); // O objeto DataAdapter vai preencher o  
    DataSet com os dados do BD, O método Fill é o responsável por preencher o DataSet  
    objConnection.Close();  
    objCommand.Dispose();  
    objConnection.Dispose();  
    return ds;  
}  
}
```

Persistência – SELECT * From... Comentado

// O DataSet é um objeto que armazena grande quantidade de dados

```
Public static DataSet SelectAll() {
```

// Aqui não é necessário criar um try/catch, pois se um erro ocorrer com a conexão, o dataset ficará vazio. Se o dataset ficar vazio ou não temos dados no BD ou o código está errado

```
    DataSet ds = new DataSet();  
    IDbConnection objConnection;  
    IDbCommand objCommand;  
    IDataAdapter objDataAdapter; // trabalha como ponte entre dados desconexos e conexos. Após  
    carregado o DataSet, será possível uma manipulação desconexa destes dados em memória.  
  
    objConnection = Mapped.Connection();  
    objCommand = Mapped.Command("SELECT * FROM tipoempresa ORDER BY tip_descricao",  
    objConnection);  
    objDataAdapter = Mapped.Adapter(objCommand);  
  
    // O objeto DataAdapter vai preencher o DataSet com os dados do BD, O método Fill é o  
    responsável por preencher o DataSet  
    objDataAdapter.Fill(ds);  
    objConnection.Close();  
    objCommand.Dispose();  
    objConnection.Dispose();  
    return ds;  
}  
}
```

Carregar DropDownList

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

public partial class Paginas_CarregarDropDown : System.Web.UI.Page{

    protected void Page_Load(object sender, EventArgs e){
        if (!IsPostBack){
            //Carregar um DropDownList com o Banco de Dados
            DataSet ds = TipoEmpresaDB.SelectAll();
            ddl.DataSource = ds;
            ddl.DataTextField = "tip_descricao"; // Nome da coluna do Banco de dados
            ddl.DataValueField = "tip_id"; // ID da coluna do Banco
            ddl.DataBind();
            ddl.Items.Insert(0, "Selecione");
        }
    }
}
```

Carregar GridView

```
public partial class Paginas_CarregarGrid : System.Web.UI.Page {

    protected void Page_Load(object sender, EventArgs e) {
        if (!IsPostBack) {
            CarregarGrid();
        }
    }

    public void CarregarGrid() {
        //DataSet ds = new DataSet();
        DataSet ds = TipoEmpresaDB.SelectAll();
        int qtd = ds.Tables[0].Rows.Count;
        if (qtd > 0) {
            gdv.DataSource = ds.Tables[0].DefaultView;
            gdv.DataBind();
            gdv.Visible = true;
            lbl.Text = "Foram encontrados " + qtd + " de registros";
        } else {
            gdv.Visible = false;
            lbl.Text = "Não foram encontrado registros...";
        }
    }
}
```


Delete

```
public static int Delete(int id)
{
    int retorno = 0;
    try
    {
        IDbConnection objConexao;
        IDbCommand objComando;

        //SINTAXE: "DELETE FROM nomeDaTabela WHERE nomeDaColunaID = ?parametrização";
        string sql = "delete from tip_tipoempresa where tip_id = ?id";
        objConexao = Mapped.Connection();
        objComando = Mapped.Command(sql, objConexao);
        objComando.Parameters.Add(Mapped.Parameter("?id", id));
        objComando.ExecuteNonQuery();
        objConexao.Close();
        objComando.Dispose();
        objConexao.Dispose();
    }
    catch (Exception e)
    {
        retorno = -2;
    }
    return retorno;
}
```

Carregando vários elementos em uma div de maneira dinâmica

O Label receberá as informações do Banco.

Primeiro devemos trabalhar a persistência para trazer as informações desejadas do banco.

Este dados devem ser armazenados em um DataSet (igual ao SELECTALL).

Em nosso exemplo iremos carregar todos os tipos de empresa em um Label.

Código ASPX

```
<asp:Label ID="lblTipoEmpresa" runat="server"></asp:Label>
```

Code be Hidden

```
protected void Page_Load(object sender, EventArgs e){
    CarregarLabel();
}

private void CarregarLabel(){
    // Carregando todos os elementos da base de dados
    // em um DataSet

    DataSet ds = TipoEmpresaDB.SelectAll();

    // DataRow são linhas de dados
    // Usaremos o DataRow para percorre as linhas de nossa base de dados
    lblTipoEmpresa.Text = "";
    foreach (DataRow dr in ds.Tables[0].Rows){

        lblTipoEmpresa.Text += "<div class='alert-danger alert'
            style='width:15%;'> Identificação: " + dr["tip_id"];
        lblTipoEmpresa.Text += "<br /> Descrição: " + dr["tip_descricao"]+"</div>";

    }
}
```

Update

```
public static int Update(TipoEmpresa tipo) {
    int retorno = 0;
    try{
        IDbConnection objConexao; // Abre a conexao
        IDbCommand objCommand; // Cria o comando
        string sql = "UPDATE tipoempresa SET " + "tip_descricao = ?descricao WHERE
        tip_id = ?tip_id";
        objConexao = Mapped.Connection();
        objCommand = Mapped.Command(sql, objConexao);
        objCommand.Parameters.Add(Mapped.Parameter("?descricao", tipo.Tip_descricao));
        objCommand.Parameters.Add(Mapped.Parameter("?codigo", tipo.Tip_id));
        objCommand.ExecuteNonQuery();
        objConexao.Close();
        objCommand.Dispose();
        objConexao.Dispose();
    }catch (Exception e) {
        retorno = -2;
    }
    return retorno;
}
```

Update – Comentado

// Recebe todos os dados do objeto (Classe TipoEmpresa) e atualiza o BD.
// É muito mais prático passar um objeto todo de uma vez que passar atributo por atributo

```
public static int Update(TipoEmpresa tipo){
    // 0 = caso de Sucesso    -2 = caso de falha de conexão
    int retorno = 0;

    try {
        IDbConnection objConexao; // Abre a conexão
        IDbCommand objCommand;      // Cria o comando

        string sql = "UPDATE tipoempresa SET tip_descricao = ?descricao WHERE
        tip_id = ?tip_id";

        objConexao = Mapped.Connection(); // Abre a conexão

        //atribui ao objeto comando a Sql e o objeto de conexão
        objCommand = Mapped.Command(sql,objConexao);

        // Parametriza os dados
        // Adiciona os dados para parametrização
        objCommand.Parameters.Add(Mapped.Parameter("?descricao", tipo.Tip_descricao));
        objCommand.Parameters.Add(Mapped.Parameter("?codigo", tipo.Tip_id));

        // Executa o comando
        // Comando ExecuteNonQuery - utilizado quando o código não tem retorno, como seria o
        // caso de SELECT
        objCommand.ExecuteNonQuery();

        objConexao.Close(); //Fecha a conexão

        //Disponibiliza o objeto conexão e o objeto comando para serem utilizados
        // novamente
        objCommand.Dispose();
        objConexao.Dispose();
    }
    catch (Exception e) {
        retorno = -2;
    }
    return retorno;
}
```

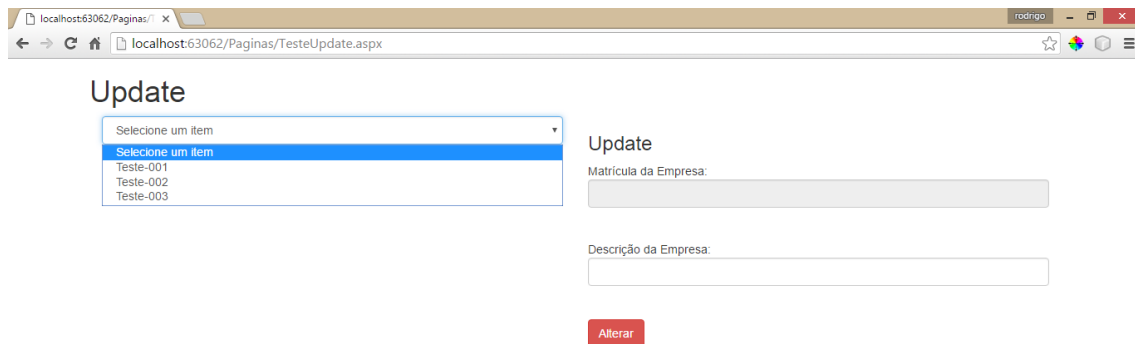
```
}
```

SelectId

```
public static DataSet SelectAId(int id)
{
    DataSet ds = new DataSet();
    IDbConnection objConnection;
    IDbCommand objCommand;
    string sql = "select * from tip_tipoempresa where tip_id = ?id;";
    IDataAdapter objDataAdapter;
    objConnection = Mapped.Connection();
    objCommand = Mapped.Command(sql, objConnection);
    objCommand.Parameters.Add(Mapped.Parameter("?id", id));
    objDataAdapter = Mapped.Adapter(objCommand);

    objDataAdapter.Fill(ds);
    objConnection.Close();
    objCommand.Dispose();
    objConnection.Dispose();
    return ds;
}
```

Visão do Browser



Código ASPX

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="TesteUpdate.aspx.cs" Inherits="Paginas_TestUpdate" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div class="container">
        <div class="row">
            <h1>Update</h1>
            <div class="col-lg-6">
                <asp:DropDownList ID="ddlTipoEmpresa" runat="server" AutoPostBack="true" CssClass="form-control"
OnSelectedIndexChanged="ddlTipoEmpresa_SelectedIndexChanged"> </asp:DropDownList>

                <br />
                <asp:Label ID="lbl" runat="server" Text="Label"></asp:Label>
            </div>
            <div class="col-lg-6">
                <h3>Update</h3>
                Matrícula da Empresa:
                <asp:TextBox ID="txtID" runat="server" CssClass="form-control" Enabled="false"></asp:TextBox>

                <br />
                <br />
                Descrição da Empresa:
                <asp:TextBox ID="txtDescicao" runat="server" CssClass="form-control">
                    </asp:TextBox>

                <br />
                <br />
            </div>
        </div>
    </div>
</asp:Content>
```

Disciplina de Script

```
                <asp:Button ID="btnAlterar" runat="server" Text="Alterar" CssClass="btn-danger btn"
                OnClick="btnAlterar_Click" />
            </div>
        </div>
    </div>
</asp:Content>
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Paginas_TestesUpdate : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e){
        if (!IsPostBack){
            CarregarDDL();
        }
    }

    // Faz o update
    protected void btnAlterar_Click(object sender, EventArgs e){
        TipoEmpresa tipo = new TipoEmpresa();
        tipo.Tip_id = Convert.ToInt32(txtID.Text);
        tipo.Tip_descricao = txtDescricao.Text;

        switch(TesteDB.Update(tipo)){
            case 0:
                lbl.Text = "<div class='alert-success alert'> >>> OK <<< </div>";
                break;
            case -2:
                lbl.Text = "<div class='alert-danger alert'> >>> ERRO <<< </div>";
                break;
        }

        txtDescricao.Text = "";
        txtID.Text = "";
        CarregarDDL();
    }

    // Evendo que vai carregar os formulários
    protected void ddlTipoEmpresa_SelectedIndexChanged(object sender, EventArgs e){
        DataSet ds = TesteDB.SelectAid(Convert.ToInt32(ddlTipoEmpresa.SelectedValue));

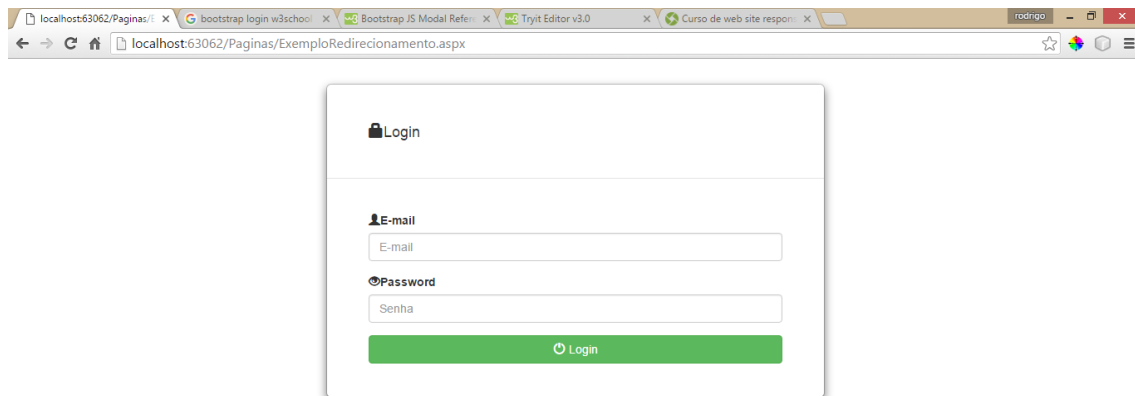
        if (ds.Tables[0].Rows.Count == 0){
            lbl.Text = "<script>alert('Elemento não encontrado');</script>";
        }else{
            txtID.Text = ds.Tables[0].Rows[0]["tip_id"].ToString();
            txtDescricao.Text = ds.Tables[0].Rows[0]["tip_descricao"].ToString();
        }
    }

    private void CarregarDDL(){
        DataSet ds = TipoEmpresaDB.SelectAll();
        ddlTipoEmpresa.DataSource = ds;
        ddlTipoEmpresa.DataTextField = "tip_descricao";
        ddlTipoEmpresa.DataValueField = "tip_id";
        ddlTipoEmpresa.DataBind();
        //-----
        ddlTipoEmpresa.Items.Insert(0, "Selecione um item");
    }
}
```

Redirecionamento - Exemplo:

No exemplo abaixo se a senha estiver errada será redirecionada para a página de erro:

Visão do browser



Código ASPX

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="ExemploRedirecionamento.aspx.cs" Inherits="Paginas_ExemploRedirecionamento" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="Server">

<!-- Modal -->
<div id="myModal" role="dialog">
  <div class="modal-dialog">

    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header" style="padding: 35px 50px;">
        <h4><span class="glyphicon glyphicon-lock"></span>Login</h4>
      </div>
      <div class="modal-body" style="padding: 40px 50px;">
        <div class="form-group">
          <label for="username"><span class="glyphicon glyphicon-user"></span>E-mail</label>
          <asp:TextBox ID="txtEmail" runat="server" CssClass="form-control" placeholder="E-mail"></asp:TextBox>
        </div>
        <div class="form-group">
          <label for="psw"><span class="glyphicon glyphicon-eye-open"></span>Password</label>
          <asp:TextBox ID="txtSenha" runat="server" CssClass="form-control" placeholder="Senha"></asp:TextBox>
        </div>
        <asp:LinkButton ID="btnLogin" runat="server" Text="<i class='glyphicon glyphicon-off'></i> Login"
          CssClass="btn btn-success btn-block" OnClick="btnLogin_Click"></asp:LinkButton>
      </div>
    </div>
  </div>
</div>

</div>
</div>
</asp:Content>
```

CódigoASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

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

    protected void btnLogin_Click(object sender, EventArgs e) {
        if (txtSenha.Text != "123"){
            Response.Redirect("Erro.aspx");
        }else{
            Response.Redirect("TipoEmpresa.aspx");
        }
    }
}
```

Aplicando mascaras nos controles

Primeiro devemos baixar e incluir os seguintes scripts em nosso arquivo:

```
<link href="../../css/bootstrap.min.css" rel="stylesheet" />
<script src="../../js/jquery-1.12.0.min.js"></script>
<script src="../../js/jquery.maskedinput.min.js"></script>
<script src="../../js/bootstrap.min.js"></script>
```

Depois devemos criar uma função anônima simplificada, o script abaixo pode ser colocado na página desejada, logo após os links descritos acima.

Observe que foram criadas as classes .data, .placa, .rg e .cpf.

Estas classes devem ser colocadas nos formulários desejados.

```
<script type="text/javascript">
    jQuery(function ($) {
        $(''.data').mask('99/99/9999');
        $(''.placa').mask('aaa*-9999');
        $(''.rg').mask('99.999.999-*');
        $(''.cpf').mask('999.999.999-99');
    });
</script>

// a – Representa um carácter alfa (AZ , az)
// 9 – Representa um carácter numérico (0-9)
// * – Representa um carácter alfanumérico (AZ , az ,0 -9)
```

Controle ASPX

Data:

```
<asp:TextBox ID="DATA" runat="server" CssClass="form-control data" placeholder="99/99/9999"> </asp:TextBox>
```

CPF:

```
<asp:TextBox ID="CPF" runat="server" CssClass="form-control cpf" placeholder="999.999.999-99"> </asp:TextBox>
```

Placa:

```
<asp:TextBox ID="PLACA" runat="server" CssClass="form-control placa" placeholder="AAA-9999"> </asp:TextBox>
```

RG:

```
<asp:TextBox ID="RG" runat="server" CssClass="form-control rg" placeholder="99.999.999-x"> </asp:TextBox>
```

Visão do Browser

Controles ASPX

Insira seu nome

Insira seu email

Insira seu telefone

Insira seu cep

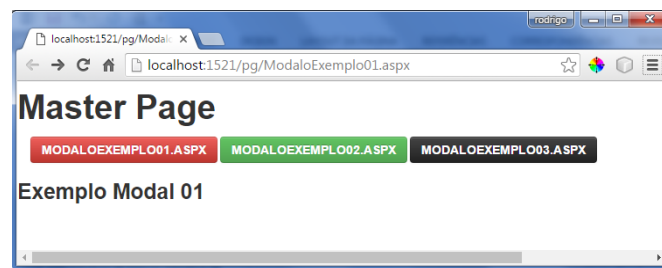
Links que perder a referência

No .NET é comum um link (como o exemplo abaixo) perder a referência.

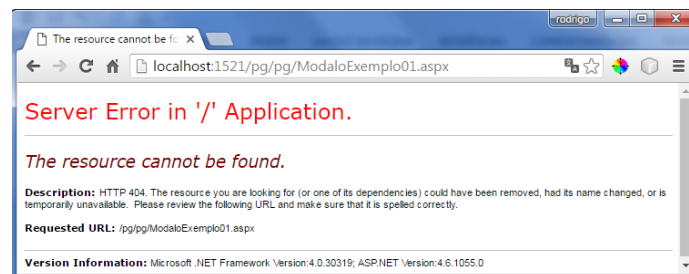
Código

```
<a href="pg/ModaloExemplo01.aspx" class="btn-danger btn">ModaloExemplo01.aspx</a>  
<a href="pg/ModaloExemplo02.aspx" class="btn-success btn">ModaloExemplo02.aspx</a>  
<a href="pg/ModaloExemplo03.aspx" class="btn-inverse btn">ModaloExemplo03.aspx</a>
```

Visão do Browser



Mensagem de erro



Para evitar este problema utilizamos o seguinte Script.

```
<a href='<%=ResolveUrl("pg/ModaloExemplo01.aspx") %>' class="btn-danger  
btn">ModaloExemplo01.aspx</a>
```

```
<a href='<%=ResolveUrl("pg/ModaloExemplo02.aspx") %>' class="btn-success btn">ModaloExemplo02.aspx</a>
```

```
<a href='<%=ResolveUrl("pg/ModaloExemplo03.aspx") %>' class="btn-inverse btn">ModaloExemplo03.aspx</a>
```

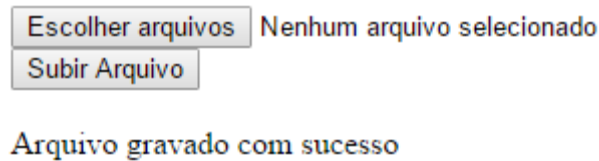
O mesmo pode acontecer com os links com **Script**, para evita este problema podemos utilizar o **ResolveURL** conforme modelo abaixo.

```
<script src='<%= ResolveUrl("bootstrap/js/jquery-1.12.3.min.js") %>'></script>  
<script src='<%= ResolveUrl("bootstrap/js/bootstrap.min.js") %>'></script>
```

Disciplina de Script

Upload Simples

Visão do Browser



Código ASPX

```
<asp:FileUpload ID="flpArquivo" runat="server" AllowMultiple="true" />

<asp:Button ID="btnArquivo" runat="server" Text="Subir Arquivo"
OnClick="btnArquivo_Click" /><br />

<asp:Label ID="lblMensagem" runat="server"></asp:Label>
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

// importar
using System.IO;

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

    protected void btnArquivo_Click(object sender, EventArgs e) {

        // Caminho que será salvo o arquivo
        string dir = Request.PhysicalApplicationPath + "pg\\uploads\\";

        // Verifica se o NÃO diretório existe
        if (!Directory.Exists(dir)) {

            // Se não existir criará o Diretório
            Directory.CreateDirectory(dir);
        }

        // Cria-se um for pois é possível carregar mais de um arquivo
        // flpArquivo - é o nome do botão
        // Estamos pegando todos os arquivos postados no botão e colocando em uma
        // Array

        foreach (HttpPostedFile flp in flpArquivo.PostedFiles) {
            try {

                // Limitando o tamanho do arquivo em kb
                double mp = 500;

                // Verifica se este arquivo existe (se foi enviado pelo botão)
                if (flpArquivo.HasFile) {
```


Disciplina de Script

```
//pega o nome do arquivo
string arq = Path.GetFileName(flp.FileName);

// pega a extensão do arquivo
string ext = Path.GetExtension(flp.FileName);

// verifica o tamanho do arquivo
double ta = flp.ContentLength / 1024;

//Verifica a extensão do arquivo
if (ext==".doc" || ext==".pdf" || ext==".txt" || ext==".jpg" ||
    ext==".png"){

//Verifica se o tamanho do arquivo não é maior que o tamanho
    definido
    if (ta <= mp) {

// Monta-se o nome do arquivo, neste exemplo (data hora .
    extensão)
    arq = DateTime.Now.ToString("yyyyMMddHHmmssfff") + ext;

//Verifica se o arquivo não existe
    if (!File.Exists(dir + arq)) {

        //Se o arquivo não existe ele é salvo
        flp.SaveAs(dir + arq);
        lblMensagem.Text = "Arquivo gravado com sucesso";

    }else{

        lblMensagem.Text = "Arquivo ja existe!";
    }
    }else {

        lblMensagem.Text = "tamanho maximo excedido - 500kb!";
    }
    } else {

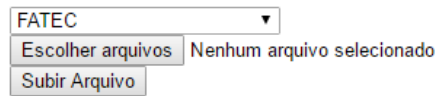
        lblMensagem.Text = "Extensão invalida";
    }
    } else {
        lblMensagem.Text = "Selecione um arquivo";
    }
    }

} catch {

    lblMensagem.Text = "Erro no upload!";
}
}
}
}
```

Upload - Criando um diretório com o ID da empresa e salvando o arquivo com o id da empresa

Visão do Browser



Arquivo gravado com sucesso

Código ASPX

```
<asp:DropDownList ID="ddl" runat="server"></asp:DropDownList>

<asp:FileUpload ID="flpArquivo" runat="server" AllowMultiple="true" />

<asp:Button ID="btnArquivo" runat="server" Text="Subir Arquivo" OnClick="btnArquivo_Click" /><br />

<asp:Label ID="lblMensagem" runat="server"></asp:Label>
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

// importar
using System.IO;
using System.Data;

public partial class pg_uploadID_UploadID : System.Web.UI.Page {

    protected void Page_Load(object sender, EventArgs e) {
        if (!IsPostBack) {
            CarregarDDL();
        }
    }

    public void CarregarDDL(){

        DataSet ds = TipoEmpresaDB.SelectAll();

        ddl.DataSource = ds;
        ddl.DataTextField = "tip_descricao";
        ddl.DataValueField = "tip_id";
        ddl.DataBind();

        ddl.Items.Insert(0, "Selecione");
    }

    protected void btnArquivo_Click(object sender, EventArgs e) {

        //Pegando o ID da Empresa
        string id_empresa = ddl.SelectedValue.ToString();

        // Caminho que será salvo o arquivo
        string dir = Request.PhysicalApplicationPath + "pg\\uploads\\" + id_empresa + "\\";

        // Verifica se o NÃO diretório existe
        if (!Directory.Exists(dir)) {
```

Disciplina de Script

```
//Se não existir criará o Diretório
Directory.CreateDirectory(dir);
}

// Cria-se um for pois é possível carregar mais de um arquivo
// flpArquivo - é o nome do botão
// Estamos pegando todos os arquivos postados no botão e colocando em uma
array
foreach (HttpPostedFile flp in flpArquivo.PostedFiles) {
    try{
        //Limitando o tamanho do arquivo em kb
        double mp = 500;

        //Verifica se este arquivo existe (se foi enviado pelo botão)
        if (flpArquivo.HasFile) {
            //pega o nome do arquivo
            string arq = Path.GetFileName(flp.FileName);

            // pega a extensão do arquivo
            string ext = Path.GetExtension(flp.FileName);

            // verifica o tamanho do arquivo
            double ta = flp.ContentLength / 1024;

            //Verifica a extensão do arquivo
            if (ext==".doc" || ext==".pdf" || ext==".txt" || ext==".jpg" || ext == ".png") {

                //Verifica se o tamanho do arquivo não é maior que o
                tamanho definido
                if (ta <= mp) {

                    // Monta-se o nome do arquivo, neste exemplo (data hora . extensão)
                    arq = id_empresa + "-" +
                        DateTime.Now.ToString("yyyyMMddHHmmssfff") + ext;

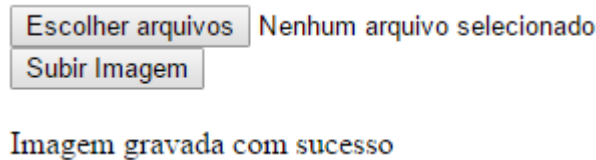
                    //Verifica se o arquivo não existe
                    if (!File.Exists(dir + arq)) {
                        //Se o arquivo não existe ele é salvo
                        flp.SaveAs(dir + arq);
                        lblMensagem.Text = "Arquivo gravado com sucesso";
                    } else {
                        lblMensagem.Text = "Arquivo ja existe!";
                    }
                } else {
                    lblMensagem.Text = "tamanho maximo excedido –
                        500kb!";
                }
            } else {
                lblMensagem.Text = "Extensão invalida";
            }
        } else {
            lblMensagem.Text = "Selecione um arquivo";
        }
    } catch {
        lblMensagem.Text = "Erro no upload!";
    }
}
}
```

Upload Simples de Imagens

Este código é parecido com o anterior, entretanto ele salva dois arquivos.

Um com o tamanho do arquivo e outro redimensionado (com um tamanho menor).

Visão do Browser



Código ASPX

```
<asp:FileUpload ID="flpArquivo" runat="server" AllowMultiple="true" />

<asp:Button ID="btnImagem" runat="server" Text="Subir Imagem"
OnClick="btnImagem_Click" /><br />

<asp:Label ID="lblMensagem" runat="server"></asp:Label>
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

// importar
using System.IO;

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

    protected void btnImagem_Click(object sender, EventArgs e) {
        string dir = Request.PhysicalApplicationPath + "pg\\uploads\\";

        if (!Directory.Exists(dir)) {
            Directory.CreateDirectory(dir);
        }

        foreach (HttpPostedFile flp in flpArquivo.PostedFiles) {

            try {
                double mp = 2000;

                if (flpArquivo.HasFile) {

                    string arq = Path.GetFileName(flp.FileName);
                    string ext = Path.GetExtension(flp.FileName);
                    ext = ext.ToLower();
                    double ta = flp.ContentLength / 1024;

                    if (ext == ".jpg" || ext == ".png" || ext == ".gif") {
```

Disciplina de Script

```
if (ta <= mp) {
    arq = DateTime.Now.ToString("yyyyMMddHHmmssfff") + ext;
    if (!File.Exists(dir + arq)) {
        flp.SaveAs(dir + arq);

        System.Drawing.Image imgOriginal =
            System.Drawing.Image.FromFile(Server.MapPath("~/pg/uploads/" +
                arq), true);

        System.Drawing.Image.GetThumbnailImageAbort miniatura = new
            System.Drawing.Image.GetThumbnailImageAbort(erro);

        System.Drawing.Image imgRedimensionada;

        int width, height;

        if (imgOriginal.Width > 200) {

            width = 200;
            height = (int)(width * imgOriginal.Height) / imgOriginal.Width;

        } else {

            width = imgOriginal.Width;
            height = imgOriginal.Height;
        }

        imgRedimensionada = imgOriginal.GetThumbnailImage(width, height,
            miniatura, IntPtr.Zero);

        //nova imagem gerada
        string novo_nome = arq.Substring(0, arq.Length - 4) + "-mini.png";

        //salve a miniatura em um formato
        imgRedimensionada.Save(dir + novo_nome,
            System.Drawing.Imaging.ImageFormat.Png);

        imgRedimensionada.Dispose();

        imgOriginal.Dispose();

        //visualize a original e a redimensionada
        //File.Delete(Server.MapPath("~/Uploads/" +
        Path.GetFileName(flp.FileName)));

        lblMensagem.Text = "Imagem gravada com sucesso";

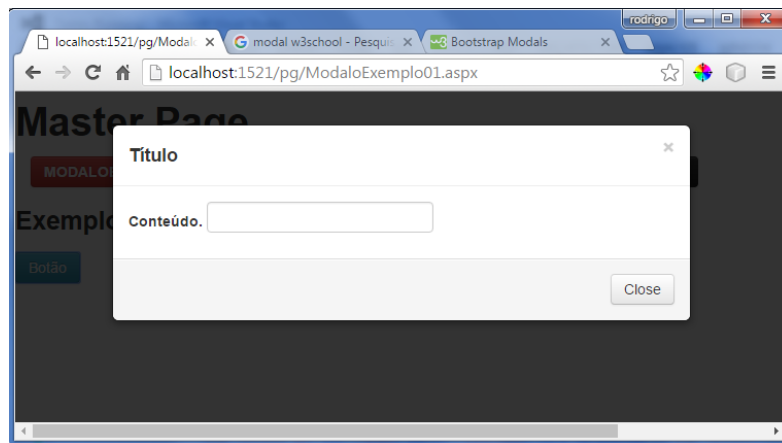
    } else {
        lblMensagem.Text = "Arquivo ja existe!";
    }
} else {
    lblMensagem.Text = "tamanho maximo excedido - 500kb!";
}
} else{
    lblMensagem.Text = "Extensão invalida";
}

} else{
    lblMensagem.Text = "Selecione um arquivo";
}

} catch {
    lblMensagem.Text = "Erro no upload!";
}
```

Disciplina de Script

```
    }  
}  
  
public bool erro()  
{  
    return false;  
}  
}
```



Código HTML / CSS

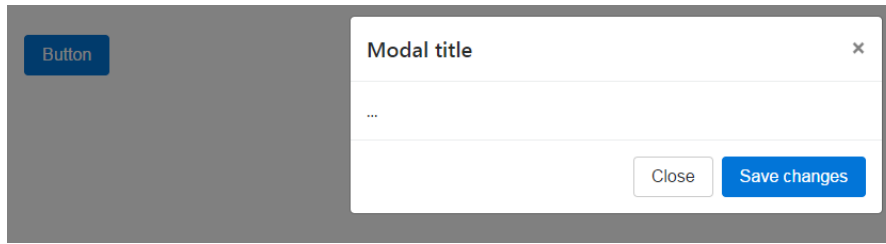
```
<!-- Gatilho do Modal-->
<button type="button" class="btn btn-info btn-lg" data-toggle="modal" data-
target="#myModal">Botão</button>

<!-- Início do Conteúdo - Modal -->
<div id="myModal" class="modal fade" role="dialog">
  <div class="modal-dialog">

    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal"> &times;</button>
        <h4 class="modal-title">Titulo</h4>
      </div>
      <div class="modal-body">
        <strong>Conteúdo.</strong>
        <asp:TextBox ID="txt" runat="server" CssClass="form-horizontal">
          </asp:TextBox>
        </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">
          Close</button>
        </div>
      </div>
    </div>
  </div>
</div>
<!-- Fim do Conteúdo - Modal -->
```

Chamar Modal Através do Code be Hidden

Visão do Browser



Código ASPX

```
<!-- Gatilho -->
<asp:Button ID="btn" runat="server" Text="Button" class="btn btn-primary" OnClick="btn_Click"/>

<!-- Início - Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="
exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
<!-- Fim - Modal -->
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class pg_Modal : System.Web.UI.Page{

    protected void Page_Load(object sender, EventArgs e) { }

    protected void btn_Click(object sender, EventArgs e) {

        Page.ClientScript.RegisterStartupScript(this.GetType(), "script",
        "<script>$('#myModal').modal('show');</script>", false);

    }
}
```


Controles de validação

A validação pode ser do lado do servidor (quando ocorre um PostBack) ou pode ser do lado do Cliente. As validações do lado do cliente são mais rápidas.

WebConfig:

```
<appSettings>
  <add key="ValidationSettings:UnobtrusiveValidationMode" value="None"/>
</appSettings>
```

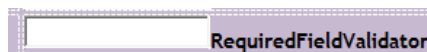
Aba VALIDATOR do toolbox

Required Field Validator:

Valida se o campo foi preenchido.

Propriedades Importantes:

- | | |
|--------------------------------------|---|
| ▪ ControlToValidate : | Nome do Web Control a que se refere a validação |
| ▪ Display: Static | Para mostrar o erro onde está localizado o próprio controle |
| ▪ ErrorMessage: | Descrição do erro |
| ▪ SetFocusOnError: true/false | focará no Controle que teve erro, muito usado quando temos muitos controles em uma página |



Exemplo:

- ControlToValidate: txb1;
- Display: Static
- ErrorMessage: Preenchimento Obrigatório
- ForeColor: red
- SetFocusOnError: True

Ranger Validator

Verifica se o usuário preencheu o campo dentro de um intervalo já determinado. Utiliza as mesmas propriedades de **“RequiredFieldValidation”** e mais:

- **MaximumValue:** Valor Máximo
- **MinumumValue:** Valor Mínimo
- **Type:** String / Integer / Double / Data / Currencu (moeda)

Compare Validator

Faz a comparação entre dois controles, por exemplo, verifica se o usuário digitou a senha em dois campos iguais.

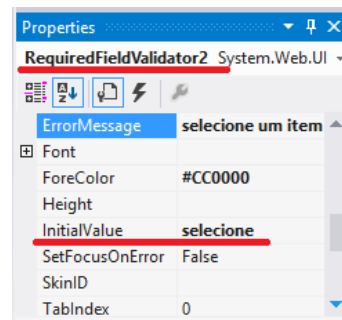
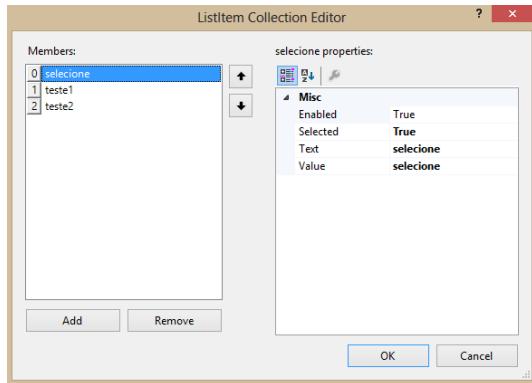
Mesmas propriedades das validações anteriores.

Propriedades:

- **ControlToCompare:** TextBox1
- **ControlToValidade:** TextBox2 [vai compara o textBox1 com o textBox2]
- **Operadores** equals / notEquals / etc.
- **Type** string / integer / etc.

Validando um DropDownList

Configure o primeiro campo como “Selezione” por exemplo.



No controle Validator.

Propriedades:

- **InitialValue:** Colocar o mesmo nome do campo inicial do DropDownList, em nosso exemplo (Selezione).

Controle CustomValidator (Validação do lado do servidor)

Devemos criar um evento, por exemplo, digitar um número de 8 dígitos, ou um texto de 8 caracteres.

Configure o **CustomValidator** como os comparadores anteriores.

Coloque um evento no CustomValidator.

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args){
    if (args.Value.Length == 8){
        args.IsValid = true;
        lbl.Visible = true;
    } else {
        args.IsValid = false;
        lbl.Visible = false;
    }
}
```

Observação: OPostBack vai funcionar sempre pois a validação ocorre do lado do servidor. Sempre que possível use a validação do lado do cliente.

Validation Summary

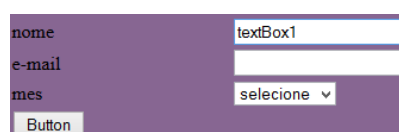
Exibe um sumário de todos os erros, ou seja, vai exibir uma lista de erros dos controles.

Propriedade:

- **DisplayMode:** modo que será exibido os erros (List / Bullet List / Single Paragraph)
Utilize também **ShowMessageBox** ou **ShowSummary**.
- Ambos true / false

Exemplo:

- ControlToValidator: textBox1
- ErrorMessage: Texto que será exibido no ValidationSummary
- Text: Texto que será exibido no próprio controle
- ForeColor: red



Sessão

A sessão permite armazenar qualquer tipo de dado na memória do servidor.

A informação é protegida porque nunca é exibida para o usuário, além de pertencer exclusivamente a uma sessão específica.

A sessão é ideal para o armazenamento de informações, como os itens de um carrinho de compras, enquanto o usuário navega entre as páginas.

Criar Sessão:

```
Session["nome"] = "Dionisio";
```

Persistir a Sessão:

```
lblSessao.Text = Convert.ToString(Session["nome"]);
```

Remover Sessão:

```
Session.Remove("nome");
```

Exemplo:

Neste exemplo Criaremos quatro páginas simples.

Na primeira página a **SESSÃO** será criada automaticamente.

Nas demais páginas a **SESSÃO** será persistida.

Na última página teremos um botão com a finalidade de **ENCERRAR A SESSÃO**.

Visão do Browser

[Home pg1 pg2 pg3](#)
[Home pg1 pg2 pg3](#)
[Home pg1 pg2 pg3](#)
[Home pg1 pg2 pg3](#)

HOME

Dionisio

PÁGINA 01

Dionisio

PÁGINA 02

Dionisio

PÁGINA 03

Dionisio

Remover Sessão

HOME

```
<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>HOME</h1>
```

```
<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label>
```

PÁGINA 01

```
<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>PÁGINA 01</h1>
```

```
<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label>
```

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

    Session["nome"] = "Dionisio";
    // lblSessao.Text = "Bem Vindo: " + Session["nome"];
    lblSessao.Text = Convert.ToString(Session["nome"]);

}
```

```
protected void Page_Load(object sender, EventArgs e) {
    if (Session["nome"] == null || Session["nome"] == "")
    {

        lblSessao.Text = "Sessão não encontrada –
        Direcionar para uma página de erro";

    } else {
        lblSessao.Text =
        Convert.ToString(Session["nome"]);
    }
}
```

Disciplina de Script

PÁGINA 02

```
<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>PÁGINA 02</h1>
```

```
<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label>
```

PÁGINA 03

```
<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>PÁGINA 03</h1>
```

```
<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label><br /><br />
```

```
<asp:Button ID="btn" runat="server" Text="Remover
Sessão" OnClick="btn_Click" />
```

```
protected void Page_Load(object sender, EventArgs e) {
    if (Session["nome"] == null || Session["nome"] == "")
    {

        lblSessao.Text = "Sessão não encontrada –
        Direcionar para uma página de erro";

    }else{

        lblSessao.Text =
        Convert.ToString(Session["nome"]);
    }
}
```

```
protected void Page_Load(object sender, EventArgs e) {
    if (Session["nome"] == null || Session["nome"] == "")
    {

        lblSessao.Text = "Sessão não encontrada –
        Direcionar para uma página de erro";

    } else {

        lblSessao.Text =
        Convert.ToString(Session["nome"]);
    }
}
```

```
protected void btn_Click(object sender, EventArgs e) {
    Session.Remove("nome");
    lblSessao.Text = "Sessão não encontrada –
    Direcionar para uma página de erro";
}
```

Exemplo:

Neste exemplo Criaremos quatro páginas simples.

Página de Login, Página de Erro, Pág1 (que só poderá ser acessada pelo perfil 1) e Pág2 (que só poderá ser acessada pelo perfil 2).

Persistência UsuárioDB.

Na última página teremos um botão com a finalidade de **ENCERRAR A SESSÃO**.

Visão do Browser.**Login**

E-mail:

Senha:

Perfil 1

Dionisio

Perfil 2

Rodrigo

Erro[Voltar para a HOME](#)**Banco**

usu_codigo	usu_email	usu_senha	per_codigo	usu_nome
1	d@gmail.com	123	3B	1 Dionisio
2	r@gmail.com	111	3B	2 Rodrigo

Persistência

```

public static DataSet SelectLOGIN(string email, string senha) {
    DataSet ds = new DataSet();
    IDbConnection objConexao;
    IDbCommand objCommand;

    IDataAdapter objDataAdapter;
    string sql = "select * from usu_usuario where usu_email = ?usu_email and usu_senha = ?usu_senha";

    objConexao = Mapped.Connection();
    objCommand = Mapped.Command(sql, objConexao);

    objCommand.Parameters.Add(Mapped.Parameter("?usu_email", email));
    objCommand.Parameters.Add(Mapped.Parameter("?usu_senha", senha));

    objDataAdapter = Mapped.Adapter(objCommand);
    objDataAdapter.Fill(ds);

    objConexao.Close();
    objConexao.Dispose();
    objCommand.Dispose();

    return ds;
}

```

Código ASPX – Login

```
<h1>Login</h1>
```

```
E-mail:<br />
```

```
<asp:TextBox ID="txtEmail" runat="server" type="email"></asp:TextBox>
```

```
Senha:<br />
```

```
<asp:TextBox ID="txtSenha" runat="server" type="password"></asp:TextBox>
```

```
<asp:Button ID="btn" runat="server" Text="Login" OnClick="btn_Click"/>
```

Código ASPX.CS – Login

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

public partial class pg_sessao02_Login : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) { }

    protected void btn_Click(object sender, EventArgs e) {

        DataSet ds = UsuarioDB.SelectLOGIN(txtEmail.Text, txtSenha.Text);

        if (ds.Tables[0].Rows.Count == 1) {

            Session["nome"] = ds.Tables[0].Rows[0]["usu_nome"].ToString();
            Session["perfil"] = ds.Tables[0].Rows[0]["per_codigo"].ToString();

            int perfil = Convert.ToInt32(Session["perfil"]);

            if (perfil == 1) {
                Response.Redirect("Pag1.aspx");
            }

            if (perfil == 2) {
                Response.Redirect("Pag2.aspx");
            }
        } else {
            Response.Redirect("Erro.aspx");
        }
    }
}
```

Código APSX – Página 1

```
<h1>Perfil 1</h1>
<asp:Label ID="lblSessao" runat="server" Text="---"></asp:Label>
<asp:Button ID="btnSair" runat="server" Text="Sair" OnClick="btnSair_Click" />
```

Código APSX.CS – Página 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class pg_sessao02_Pag1 : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {

        if (Session["nome"] == null || Session["perfil"] == null) {
            Response.Redirect("Erro.aspx");
        } else {
            int perfil = Convert.ToInt32(Session["perfil"]);

            if (perfil == 1) {
                lblSessao.Text = Session["nome"].ToString();
            } else {
                Response.Redirect("Erro.aspx");
            }
        }
    }
}
```

Disciplina de Script

```
protected void btnSair_Click(object sender, EventArgs e) {  
    Session.Remove("nome");  
    Session.Remove("perfil");  
    Response.Redirect("Login.aspx");  
}  
}
```

Código APSX – Página 2

```
<h1>Perfil 2</h1>  
<asp:Label ID="lblSessao" runat="server" Text="---"></asp:Label>  
<asp:Button ID="btnSair" runat="server" Text="Sair" OnClick="btnSair_Click" />
```

Código APSX.CS – Página 2

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
  
public partial class pg_sessao02_Pag2 : System.Web.UI.Page{  
  
    protected void Page_Load(object sender, EventArgs e) {  
        if (Session["nome"] == null || Session["perfil"] == null) {  
            Response.Redirect("Erro.aspx");  
        } else {  
            int perfil = Convert.ToInt32(Session["perfil"]);  
  
            if (perfil == 2) {  
                lblSessao.Text = Session["nome"].ToString();  
            } else {  
                Response.Redirect("Erro.aspx");  
            }  
        }  
    }  
  
    protected void btnSair_Click(object sender, EventArgs e) {  
        Session.Remove("nome");  
        Session.Remove("perfil");  
        Response.Redirect("Login.aspx");  
    }  
}
```

Código APSX – Página Erro

```
<h1>Erro</h1>  
<a href="Login.aspx">Voltar para a HOME</a>
```

Código APSX.CS – Página Erro

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
  
public partial class pg_sessao02_Erro : System.Web.UI.Page  
{  
    protected void Page_Load(object sender, EventArgs e)  
    {  
  
    }  
}
```


Criptografia Básica

Visão do Browser

Senha:

cd8c29b8deed323fe1538cfbdb46fc2a2ea61cfd67807f0629708ea2a6e13a2919def3c837c4e7f2c8e0067568e3236827defb05c9346e476b6e954440a908a7

Código ASPX

```
Senha:
<asp:TextBox ID="txtPassword" runat="server"></asp:TextBox>
<br />
<br />
<asp:Label ID="lblSenha" runat="server" Text="---"></asp:Label>
<br />
<br />
<asp:Button ID="btnSenha" runat="server" Text="Gerar Senha" OnClick="btnSenha_Click"/>
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

//-----
using System.Text;
using System.IO;
using System.Security.Cryptography;

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

    protected void btnSenha_Click(object sender, EventArgs e) {
        UnicodeEncoding UE = new UnicodeEncoding();
        byte[] HashValue, MessageBytes = UE.GetBytes(txtPassword.Text);
        SHA512Managed SHhash = new SHA512Managed();
        string strHex = "";

        HashValue = SHhash.ComputeHash(MessageBytes);
        foreach (byte b in HashValue) {
            strHex += String.Format("{0:x2}", b);
        }
        lblSenha.Text = strHex;
    }
}
```

Utilizando Sessão

A sessão é global e é acessível em todo o aplicativo para o usuário atual. No entanto, a sessão pode ser perdida em alguns casos:

- Se o usuário fechar ou reiniciar o navegador.
- Se o usuário acessar a mesma página web de outra janela do navegador. Se for a mesma janela, a sessão não será perdida, mas isso varia entre os navegadores.
- Se a sessão expirar.
- Se o código da página finalizar a sessão com o método **Session.Abandon()**.

Nos dois primeiros casos, a sessão ainda permanecerá inacessível na memória, até que finalmente será finalizada (normalmente, isso acontece depois de 20 minutos).

Exemplo 01:

Neste exemplo será criado duas páginas.

Na primeira página será criado duas sessões referente a nome e sobrenome.

Na segunda página há a possibilidade de encerrar a sessão.

Pagina 01: Criando a Sessão:

Código ASP

```
<asp:Label ID="lbl" runat="server" Text="Label"></asp:Label>
<br />
<a href="recebeSessao4b.aspx" class="btn-primary btn" target="_blank">Sessão B</a>
<asp:Button ID="btn" runat="server" Text="Encerrar Sessão" CssClass="btn-danger btn" OnClick="btn_Click"/>
```

Código CS

```
protected void Page_Load(object sender, EventArgs e) {
    Session["nome"] = "Rodrigo";
    Session["sobreNome"] = "Dionisio";

    lbl.Text = "Nome: " + Session["nome"] + " Sobrenomw: " +
    Session["sobrenome"];
}
```

Pagina 02: Encerrando a Sessão:

Código ASP

```
<asp:Label ID="lbl" runat="server" Text="Label"></asp:Label><br />
<a href="recebeSessao4b.aspx" class="btn-primary btn" target="_blank">Sessão B</a>
<asp:Button ID="btn" runat="server" Text="Encerrar Sessão" CssClass="btn-danger btn" OnClick="btn_Click"/>
```

Código CS

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

    if (Session["nome"] == null || Session["sobrenome"] == null) {
        lbl.Text = "Sem Usuário Logado";
    } else {
        lbl.Text = "Bem Vindo: " + Session["nome"].ToString() + " " + Session["sobrenome"].ToString();
    }
}
```

Disciplina de Script

```
protected void btn_Click(object sender, EventArgs e) {  
    Session.Remove("nome");  
    Session.Remove("sobreNome ");  
    lbl.Text = "Sem Usuário Logado";  
}
```

LOGIN

Exemplo 02:

Crie a classe funcionário:

```
public class Funcionario{
    private string nome;
    private int numero;
    private int idade;
    private string departamento;

    public Funcionario(string nome, int numero, int idade, string departamento){
        this.nome = nome;
        this.idade = idade;
        this.numero = numero;
        this.departamento = departamento;
    }

    public string Departamento{
        get { return departamento; }
        set { departamento = value; }
    }

    public int Idade{
        get { return idade; }
        set { idade = value; }
    }

    public int Numero{
        get { return numero; }
        set { numero = value; }
    }

    public string Nome{
        get { return nome; }
        set { nome = value; }
    }
}
```

Crie um WebForm com os itens ListBox, botão e dois Labels.

WebForm

Exemplo: Sessão

Funcionários

Carlos Silva Ana Santos Ricardo Magalhães Paula Moreira
--

Button

 Session ID: 4vzjrbb3nxbcupyrrdqfkie
 Número de objetos: 5
 Modo: InProc
 Rastreado por cookie: False
 Novo: True
 Timeout (em minutos): 20

Código WebForm

```

<div class="container">
  <h1>Exemplo: Sessão</h1>
  <div class="row">
    <div class="col-lg-4">

      <fieldset>
        <legend>Funcionários</legend>
        <asp:ListBox ID="lstFuncionarios" runat="server" CssClass="form-control" ></asp:ListBox><br />
        <asp:Button ID="btn" runat="server" Text="Button" CssClass="btn-danger btn" OnClick="btn_Click" /><br /><br />
        <asp:Label ID="lblDetalhes" runat="server" Text="---" CssClass="text-success"></asp:Label><br />
        <asp:Label ID="lblInfoSession" runat="server" Text="---" CssClass="text-danger"></asp:Label>
      </fieldset>

    </div>
  </div>
</div>
  
```

Código do Page_Load

```

protected void Page_Load(object sender, EventArgs e) {

  if (!IsPostBack) {

    // Criando os objetos
    // Poderia ser um DataSet com informações do Banco de dados

    Funcionario funcionario1 = new Funcionario("Carlos Silva", 1, 29, "Vendas");
    Funcionario funcionario2 = new Funcionario("Ana Santos", 2, 25, "Produção");
    Funcionario funcionario3 = new Funcionario("Ricardo Magalhães", 3, 40,
      "Gerência");
    Funcionario funcionario4 = new Funcionario("Paula Moreira", 3, 40,
      "Marketing");
    Funcionario funcionario5 = new Funcionario("Ana Maria", 5, 33, "Informática");

    // Criando uma SESSÃO para cada funcionário
    Session["funcionario1"] = funcionario1;
    Session["funcionario2"] = funcionario2;
    Session["funcionario3"] = funcionario3;
    Session["funcionario4"] = funcionario4;
    Session["funcionario5"] = funcionario5;

    // Adicionando o Objeto ao ListBox
    lstFuncionarios.Items.Add(funcionario1.Nome);
    lstFuncionarios.Items.Add(funcionario2.Nome);
    lstFuncionarios.Items.Add(funcionario3.Nome);
    lstFuncionarios.Items.Add(funcionario4.Nome);
    lstFuncionarios.Items.Add(funcionario5.Nome);
  }
}
  
```

Disciplina de Script

```
//Carregando e exibindo informações sobre a sessão:
lblInfoSession.Text = "<div class='alert-danger alert'>";
lblInfoSession.Text += "Session ID: " + Session.SessionID;
lblInfoSession.Text += "<br />Número de objetos: ";
lblInfoSession.Text += Session.Count.ToString();
lblInfoSession.Text += "<br />Modo: " + Session.Mode.ToString();
lblInfoSession.Text += "<br />Rastreado por cookie: ";
lblInfoSession.Text += Session.IsCookieless.ToString();
lblInfoSession.Text += "<br />Novo: ";
lblInfoSession.Text += Session.IsNewSession.ToString();
lblInfoSession.Text += "<br />Timeout (em minutos): ";
lblInfoSession.Text += Session.Timeout.ToString();
lblInfoSession.Text += "</div>";
}
```

Código do Botão

```
protected void btn_Click(object sender, EventArgs e){
    // Verificando se o ListBox foi selecionado
    // Se nenhum item foi selecionado ele vai retornar -1

    if (lstFuncionarios.SelectedIndex == -1){
        lblDetalhes.Text = "Nenhum funcionário selecionado!";
    }else{
        // Vai concatenar a palavra funcionário com o índice do ListBox
        // Observe que é somado 1 ao índice
        // isso ocorre por que a numeração do índice começa em 0
        // com isso teremos a sessão desejada (funcionario2, por exemplo)
        string chave = "funcionario" + (lstFuncionarios.SelectedIndex + 1).ToString();

        // Com as informações da sessão nós iremos introduzir os dados em um objeto
        // é necessário fazer o casting
        Funcionario funcionario = (Funcionario)Session[chave];

        // Transefe-se os dados do objeto para o Label
        lblDetalhes.Text = "<div class='alert-success alert'>";
        lblDetalhes.Text += "Nome: " + funcionario.Nome;
        lblDetalhes.Text += "<br />Número: " + funcionario.Numero;
        lblDetalhes.Text += "<br />Idade: " + funcionario.Idade;
        lblDetalhes.Text += "<br />Departamento: " + funcionario.Departamento;
        lblDetalhes.Text += "</div>";
    }
}
```

Está pronto o primeiro exemplo sobre Sessão

Exemplo: Sessão

Funcionários

Carlos Silva
Ana Santos
Ricardo Magalhães
Paula Moreira
...

Button

Nome: Paula Moreira
Número: 3
Idade: 40
Departamento: Marketing

Session ID: 4vzjrbb3nxbcupyrrdqfkie
Número de objetos: 5
Modo: InProc
Rastreado por cookie: False
Novo: False
Timeout (em minutos): 20

Disciplina de Script

PI

use projeto;

show tables;

create table per_perfil(

per_codigo integer primary key auto_increment,

per_descricao varchar(100)

);

insert into per_perfil(per_descricao) values('Administrador'),('Funcionário');

Select * from per_perfil;

create table usu_usuario(

usu_codigo integer primary key auto_increment,

usu_email varchar(120),

usu_senha text,

per_codigo integer,

usu_nome varchar(120),

foreign key(per_codigo) references per_perfil(per_codigo)

);

insert into usu_usuario (usu_nome, usu_email, usu_senha, per_codigo) values
('Dionisio','d@gmail.com','123',1), ('Rodrigo','r@gmail.com','111',2);

select * from usu_usuario;

public static TipoEmpresa Select(int id) {

try {

TipoEmpresa objTipo = null;

IDbConnection objConnection;

IDbCommand objCommnad;

IDataReader objDataReader;

objConnection = Mapped.Connection();

objCommnad = Mapped.Command("SELECT * FROM

Disciplina de Script

tipoempresa WHERE tip_id = ?id",

objConnection);

objCommnad.Parameters.Add(Mapped.Parameter

("?id", id));

objDataReader=objCommnad.ExecuteReader();

while (objDataReader.Read()) {

objTipo = new TipoEmpresa();'

objTipo.Tip_id = Convert.ToInt32

(objDataReader["tip_id"]);

objTipo.Tip_descricao = objDataReader

["tip_descricao"].ToString();

}

objDataReader.Close();

objConnection.Close();

objConnection.Dispose();

objCommnad.Dispose();

objDataReader.Dispose();

return objTipo;

}

catch (Exception e) {

return null;

}

}