Desenvolvendo uma aplicação camadas

Nesta aplicação vamos partir do início, criaremos primeiramente o banco de dados, passaremos pelas regras de negócio e o desenvolvimento da interface em si.

Criando o banco de dados

Crie o banco de dados do script a seguir:

Listagem 1 script do banco de dados

```
CREATE DATABASE ProjetoDI
GO
USE ProjetoDI
GO

CREATE TABLE Pessoa (
  IDPessoa int identity (1,1) NOT NULL,
  Nome varchar(50) NULL,
  DataNascimento datetime NULL,
  Sexo bit NULL,
  CONSTRAINT PK_IDPesso PRIMARY KEY CLUSTERED (IDPessoa ASC)
  )
  GO

CREATE TABLE Revista
(
   RevistaID CHAR(10) not null primary key,
   Titulo varchar(30) not null,
   Ano tinyint not null,
   Edicao tinyint not null,
   Capa varchar(35) not null,
   Publicador varchar(30) not null,
   Valor decimal(12,2)
)
GO

CREATE TABLE Sumario
(
  SumarioID int identity(1,1) not null primary key,
  RevistaID char(10),
  Artigo varchar(50) not null,
  Pagina tinyint not null,
  Constraint fk_RevSum FOREIGN KEY (RevistaID) REFERENCES
Revista(RevistaID)
)
GO

CREATE TABLE Artigo
(
  ArtigoID int identity(1,1) not null primary key,
  TituloArtigo varchar(50) not null,
  Subtitulo varchar(35),
  Resumo text
```

```
)
GO

CREATE TABLE RevistaArtigo
(
   RevistaID Char(10),
   ArtigoID int,
   Constraint FK_REVRevArtigo FOREIGN KEY (RevistaID) REFERENCES
Revista(RevistaID),
   Constraint FK_ArtRevistaArtigo FOREIGN KEY (ArtigoID) REFERENCES
Artigo(ArtigoID)
)
GO

CREATE PROCEDURE [dbo].[uspCadastrarPessoa]
@Excluir AS BIT = NULL,
@IDPessoa AS INT = NULL,
@Nome AS VARCHAR(50) = NULL,
@DataNascimento AS DATETIME = NULL,
@Sexo AS BIT = NULL
AS
BEGIN
    SET NOCOUNT ON
   BEGIN TRY
      BEGIN TRAN
      IF (@Excluir = 1)
         BEGIN
             DELETE FROM Pessoa WHERE IDPessoa = @IDPessoa
         END
      ELSE IF (@IDPessoa IS NULL)
         BEGIN
             INSERT INTO Pessoa (Nome, DataNascimento, Sexo) VALUES
(@Nome,@DataNascimento,@Sexo)
         END
      ELSE
         BEGIN
             UPDATE Pessoa SET  Nome = @Nome,
                                DataNascimento = @DataNascimento,
                                Sexo  = @Sexo
                    WHERE IDPessoa  = @IDPessoa
          END
          COMMIT TRAN
   END TRY
   BEGIN CATCH
      ROLLBACK TRAN
      SELECT ERROR_MESSAGE() AS Retorno
   END CATCH
   RETURN
END


CREATE PROCEDURE [dbo].[uspConsultarPessoa]
@IDPessoa AS INT = NULL,
@Nome AS VARCHAR(50) = NULL,
@DataNascimento AS DATETIME = NULL,
@Sexo AS BIT = NULL
AS
BEGIN
SELECT IDPessoa, Nome, DataNascimento, Sexo,
       CASE WHEN Sexo = 0 THEN 'FEMININO' ELSE 'MASCULINO' END AS
SexoDesc
```

```
FROM Pessoa
WHERE ((IDPessoa = @IDPessoa) OR (@IDPessoa IS NULL)) AND
      ((Nome LIKE '%' + @Nome + '%') OR (@Nome IS NULL)) AND
      ((DataNascimento = @DataNascimento) OR ( @DataNascimento IS
NULL)) AND
      ((Sexo = @Sexo) OR (@Sexo IS NULL)) AND
      ((@IDPessoa IS NOT NULL)OR(@Nome IS NOT NULL) OR
(@DataNascimento IS NOT NULL)OR(@Sexo IS NOT NULL))
END
```

Veja que no script acima, foram criados dois procedimentos (Stored procedures) que usaremos para acessar os dados da tabela Pessoa. O primeiro é responsável por inserir, alterar e excluir dados na tabela. Este procedimento funciona da seguinte forma: é verificado se o parâmetro excluir é verdadeiro, então exclui-se o registro. Se não for exclusão, o parâmetro IDPessoa é verificado, e caso seja diferente de NULL, o registro será incluído.

Na consulta deve ser enviado pelo menos um dos parâmetros e é retornado uma coleção de pessoa(s). O interessante deste procedimento é que ele pode ser consultado por qualquer um dos parâmetros.

Criando a solução

A solução é utilizada para agrupar projetos que estão relacionados e guardar as informações das dependências dos projetos que são utilizados no processo de construção (build). Na nossa aplicação a solução será composta pelos seguintes projetos: **acesso a dados**, **camada de negócios**, **objeto de transferência**, **aplicação web site** e **Silverlight**.

Passos para criar a solução:

Abra o Visual Studio, acesse o menu “File/New” e então selecione “Project”. Será exibida a janela “New Project”. Localize “Project types”, clique sobre o item “Other Project Types” e então selecione o item “Visual Studio Solutions”. Em templates selecione o item “Blank Solution”. Na parte inferior da janela informe o nome (Name) e a localização “Location” onde a solução será salva.

No Visual Studio, selecione menu View/Solution Explorer. Note que foi criada uma solução, onde vamos adicionar nossos projetos.

Criando o projeto de acesso a dados

O primeiro projeto que vamos criar será uma biblioteca de classes (“Class Library”) chamada “AcessoDados”. Aqui é onde será implementada a

interação com a base de dados, ou seja, é neste projeto que vamos inserir, alterar e excluir informações do banco de dados.

Para criar um novo projeto, abra a solução, acesse o menu "File/Add" e então selecione "New Project...". na janela "Add new Project" selecione o template "Class Library", altere o campo "Name" para "AcessoDados" e na propriedade "Location" aponte para a pasta onde está sua solução. Exclua a classe que foi criada automaticamente e, adicione uma nova classe e atribua o nome "clsAcessoDados".

Listagem 2: Classe de acesso a dados

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace AcessoDados
{
    public class clsAcessoDados
    {
        private SqlParameterCollection sqlParametros;

        public clsAcessoDados()
        {
            sqlParametros = new SqlCommand().Parameters;
        }

        private SqlConnection GetConnection()
        {
            return new SqlConnection(@"Data Source=localhost;Initial
Catalog=ProjetoDI;Integrated Security=True;");
        }

        public void AdicionarParametro(SqlParameter sqlParameter)
        {
            sqlParametros.Add(sqlParameter);
        }

        public void LimparParametros()
        {
            sqlParametros.Clear();
        }

        public void SetarParametros(SqlCommand sqlCommand)
        {
            sqlCommand.Parameters.Clear();
            foreach (SqlParameter sqlParam in sqlParametros)
            {
                sqlCommand.Parameters.Add(new
SqlParameter(sqlParam.ParameterName, sqlParam.Value));
            }
        }
        public object ExecuteScalar(string strComando, CommandType cmdType)
        {
```

```
            using (SqlConnection sqlCon = GetConnection())
            {
                sqlCon.Open();
                SqlCommand sqlCommand = sqlCon.CreateCommand();
                sqlCommand.CommandText = strComando;
                sqlCommand.CommandType = cmdType;
                sqlCommand.CommandTimeout = 7200;
                SetarParametros(sqlCommand);
                return sqlCommand.ExecuteScalar();
            }
        }

        public DataTable GetDataTable(String strComando, CommandType cmdType)
        {
            using (SqlConnection sqlCon = GetConnection())
            {
                sqlCon.Open();
                SqlCommand sqlCommand = sqlCon.CreateCommand();
                sqlCommand.CommandText = strComando;
                sqlCommand.CommandType = cmdType;
                sqlCommand.CommandTimeout = 7200;
                SetarParametros(sqlCommand);
                SqlDataAdapter dtaDataAdapter = new SqlDataAdapter(sqlCommand);
                DataTable dtbDataTable = new DataTable();
                dtaDataAdapter.Fill(dtbDataTable);
                return dtbDataTable;
            }
        }
    }
}
```

Note que é uma classe bem simples, que contém o método "GetConnection" que retorna a string de conexão, outros três métodos ("AdicionarParametros", "LimparParametros" e "SetarParametros") que controlam os parâmetros, o método "ExecutarScalar" que será utilizado para enviar dados para a base de dados e o método "GetDataTable" que será utilizado para recuperar os dados da base de dados.

Criando o projeto "objetos de transferência"

O projeto de objetos de transferência é onde são implementados os objetos que trafegam entre as camadas de apresentação e de negócio.

Vá no menu "File/Add" e então selecione o template "Class Library", altere o campo "Name" para "ObjetoTransferencia" e aponte para sua solução em "Location". Exclua a classe que foi adicionada automaticamente e em seguida adicione uma nova classe "Pessoa.cs".

Listagem 3: Classe Pessoa

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

```
using System.Text;

namespace ObjetoTransferencia
{
    public class Pessoa
    {
        public int IDPessoa { get; set; }
        public string Nome { get; set; }
        public DateTime DataNascimento { get; set; }
        public bool Sexo { get; set; }
        public string SexoDesc { get; set; }
    }

    public class PessoaCollection : List<Pessoa> { }

}
```

Esta classe contém todas as suas propriedades. É importante salientar que ela contém o modificador "public", ou seja, é uma classe pública que pode ser acessada por outros projetos na solução. Note que também foi criada uma segunda classe que é uma coleção de pessoas.

Criando o projeto de negócio

No projeto de negócios são implementados os métodos de consultas, inserção, alteração, exclusão e outros métodos relacionados. Acesse o menu "File/Add" e então selecione "New Project...". na janela "Add New Project" selecione o template "Class Library", altere o "Name" para "Negocios" e na propriedade "Location" aponte para a pasta ode está sal solução. Exclua a classe criada automaticamente e adicione uma nova, veja a **Listagem 4**.

Atenção! Não esqueça de adicionar referências dos objetos de transferência e do objeto de AcessoDados.

Listagem 4

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using ObjetoTransferencia;
using AcessoDados;


namespace Negocios
{
    class clsPessoa
    {
        clsAcessoDados objAcessoDados = new clsAcessoDados();
```

```
        public void InserirPessoa(Pessoa objPes)
        {
            objAcessoDados.LimparParametros();
            objAcessoDados.AdicionarParametro(new SqlParameter("@Nome",
objPes.Nome));
            objAcessoDados.AdicionarParametro(new SqlParameter("@DataNascimento",
objPes.DataNascimento));
            objAcessoDados.AdicionarParametro(new SqlParameter("@Sexo",
objPes.Sexo));
            objAcessoDados.ExecuteScalar("uspCadastrarPessoa",
CommandType.StoredProcedure);
        }

        public void AlterarPessoa(Pessoa objPes)
        {
            objAcessoDados.LimparParametros();
            objAcessoDados.AdicionarParametro(new SqlParameter("@IDPessoa",
objPes.IDPessoa));
            objAcessoDados.AdicionarParametro(new SqlParameter("@Nome",
objPes.Nome));
            objAcessoDados.AdicionarParametro(new SqlParameter("@DataNascimento",
objPes.DataNascimento));
            objAcessoDados.AdicionarParametro(new SqlParameter("@Sexo",
objPes.Sexo));
            objAcessoDados.ExecuteScalar("uspCadastrarPessoa",
CommandType.StoredProcedure);
        }

        public void ExcluirPessoa(Pessoa objPes)
        {
            objAcessoDados.LimparParametros();
            objAcessoDados.AdicionarParametro(new SqlParameter("@Excluir",
Convert.ToBoolean("True")));
            objAcessoDados.AdicionarParametro(new SqlParameter("@IDPessoa",
objPes.IDPessoa));
            objAcessoDados.ExecuteScalar("uspCadastrarPessoa",
CommandType.StoredProcedure);
        }

        public PessoaCollection ConsultarPessoa(int? intIDPessoa, string strNome,
DateTime? datDataNascimento, bool? booSexo)
        {
            PessoaCollection objPessoas = new PessoaCollection();
            objAcessoDados.LimparParametros();
            objAcessoDados.AdicionarParametro(new SqlParameter("@IDPessoa",
intIDPessoa.HasValue ? (object)intIDPessoa : (object)DBNull.Value));
            objAcessoDados.AdicionarParametro(new SqlParameter("@Nome", strNome
!= null ? (object)strNome : (object)DBNull.Value));
            objAcessoDados.AdicionarParametro(new SqlParameter("@DataNascimento",
datDataNascimento.HasValue ? (object)datDataNascimento : (object)DBNull.Value));
            objAcessoDados.AdicionarParametro(new SqlParameter("@Sexo",
booSexo.HasValue ? (object)booSexo : (object)DBNull.Value));

            using (DataTable dtbPessoas =
objAcessoDados.GetDataTable("uspConsultarPessoa", CommandType.StoredProcedure))
            {
                foreach (DataRow dtrPessoa in dtbPessoas.Rows)
                {
                    objPessoas.Add (new Pessoa {
                        IDPessoa=Convert.ToInt32(dtrPessoa["IDPessoa"]),
                        Nome = dtrPessoa["Nome"].ToString(),
```

```
                        DataNascimento =
Convert.ToDateTime(dtrPessoa["DataNascimento"]),
                        SexoDesc = dtrPessoa["SexoDesc"].ToString()
                    });
                }
            }
            return objPessoas;
        }

}
```

Criando o Web Site

Até o momento nós criamos os projetos que ficam na retaguarda, ou seja, as camadas de acesso a dados e de negócios. Ainda, criamos o projeto de objeto de transferência que é responsável por levar os dados de um lado para o outro em forma de objeto. Agora vamos iniciar a parte de camada de apresentação, isto é, a parte que proporciona a interatividade entre o usuário e a aplicação.

Adicione um novo projeto, clique em "File/Add" e selecione "New Web Site...". Na janela "Add New Web Site" escolha o template "ASP.NET Web Site". Na propriedade "Location" aponta para a pasta onde se encontra sua solução.

Este projeto será utilizado para executar nossa aplicação Silverlight e também é neste Web Site que vamos criar um Web Service que será consumido pela aplicação Silverlight.

Criando o Web Service

Nossa aplicação Silverlight fará chamada a um Web Service (WS), ou seja, a aplicação não terá acesso diretamente à base de dados, isto é, vamos criar uma interface responsável por receber as solicitações da aplicação Silverlight com o banco de dados e desenvolver o resultado para o Silverlight.

Para adicionar um Web Service no Web Site, clique com o botão direito no Web Site e selecione "Add New Item...". Na janela "Add New Item..." selecione o template "Web Service", altere o campo "Name" para "WSProDWeb", marque a opção "Place code in separate file" e clique em "Add". Note que foi criado um Web Service na raiz do Web Site e dentro da pasta App_Code foi inserido o code-behind do Web Service.

Antes de iniciar a implementação, adicione as referências dos projetos de negócios e objetos de transferência no projeto Web Site.

Clique com o botão direito no Web Service "WSProDWeb" e selecione a opção "View Code". Adicione então o código da **Listagem 5**.

Listagem 5. Implementação do Web Service

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using ObjetoTransferencia;
using Negocios;

/// <summary>
/// Summary description for WSPessoa
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX,
uncomment the following line.
// [System.Web.Script.Services.ScriptService]
public class WSPessoa : System.Web.Services.WebService {

   clsPessoa objClsPessoa = new clsPessoa();

    public WSPessoa () {    }

    [WebMethod]
    public void InserirPessoa(Pessoa objPes) {
        objClsPessoa.InserirPessoa(objPes);
    }

    [WebMethod]
    public void AlterarPessoa(Pessoa objPes)
    {
        objClsPessoa.AlterarPessoa(objPes);
    }
    [WebMethod]
    public void ExcluirPessoa(Pessoa objPes)
    {
        objClsPessoa.ExcluirPessoa(objPes);
    }

    [WebMethod]
    public PessoaCollection ConsultarPessoa(int? intIDPessoa, String strNome,
DateTime? datDataNascimento, bool? booSexo)
    {
        return objClsPessoa.ConsultarPessoa(intIDPessoa, strNome,
datDataNascimento, booSexo);
    }
}
```

Note que o Web Service faz a chamada aos métodos do projeto de negócio, ou seja, sua função é receber os parâmetros da camada de apresentação e passar para o projeto de negócio.

Antes de entrar na aplicação Silverlight, vamos testar o Web Service. Marque o Web Site como projeto de inicialização (clique com o botão direito do mouse no Web Site e selecione a opção "Set as Startup Project") e execute o projeto.

Note que será exibida uma página com os métodos do Web Service. Observe o endereço que está sendo exibido na barra do navegador (no estilo http://localhost:49374/InterfaceWeb/WSProDWeb.asmx), pois é este endereço que será utilizado para referenciar o Web Service na aplicação Silverlight.

Criando a aplicação Silverlight

Agora temos toda a estrutura necessária para que nossa aplicação Silverlight possa consumir dados de uma base de dados. Para adicionar um projeto Silverlight, abra a solução, vá no menu "File", localize o sub-menu "Add" e então selecione "New Project...". Na janela "Add New Project" selecione o template "Silverlight Application", altere o campo "Name" para "SilverlightAplicação", na propriedade "Location" aponte para a pasta onde está sua solução e clique em OK.

Na janela "Add Silverlight Application" selecione a opção para que a aplicação Silverlight seja vinculada com o Web Site onde vamos exibir a aplicação e desmarque a opção para criar uma página de teste.

Note que foi adicionado um projeto "SilverlightAplicacao" na solução e no Web Site foi criada a pasta ClientBin. Antes de prosseguir com a aplicação Silverlight, vamos preparar o Web Site para exibir a aplicação. Selecione a página "Default.aspx" no Web Site, clique com o botão direito e selecione a opção "View Markup".

O primeiro passo é adicionar o elemento "object", que permite adicionar e configurar o plug-in Silverlight, conforma a **Listagem 6**.

Listagem 6. Configurando o elemento object

```
<body>
    <form id="form1" runat="server" style="height:100%">
    <div id="silverlightControlHost">
        <object data="data:application/x-silverlight-2," type="application/x-
silverlight-2" width="100%" height="100%">
                <param name="source"
value="ClientBin/SilverlightApplicatio3ch2.xap"/>
                <param name="onError" value="onSilverlightError" />
                <param name="background" value="white" />
                <param name="minRuntimeVersion" value="3.0.40818.0" />
                <param name="autoUpgrade" value="true" />
                <a
href="http://go.microsoft.com/fwlink/?LinkID=149156&v=3.0.40818.0" style="text-
decoration:none">
```

```
                    <img src="http://go.microsoft.com/fwlink/?LinkId=161376"
alt="Get Microsoft Silverlight" style="border-style:none"/>
                </a>
            </object><iframe id="_sl_historyFrame"
style="visibility:hidden;height:0px;width:0px;border:0px"></iframe></div>
    </form>
</body>
```

Note que o elemento object possui atributos e elementos filhos (<param.../>). Os atributos "width" e "height" são a altura e a largura do plug-in. Já o atributo "type" utiliza o MIME Type do silverlight para identificar o plug-in e a versão requerida. Note que o indicador MIME Type é usado com silverlight 2 e todas as versões mais novas. O atributo "data" é recomendado para evitar problemas de desempenho em alguns navegadores. Note a vírgula no valor deste atributo, ela indica um segundo parâmetro que está com valor vazio.

O elemento "param" com nome "source" é obrigatório, pois indica a localização e o nome do arquivo da aplicação silverlight. Note que o value aponta para o arquivo com extensão".xap", que por padrão está localizado na pasta "ClientBin".

O elemento "param" com o nome "background" indica a cor de fundo. O elemento "param" com o nome "minRuntimeVersion" indica a versão mínima exigida e o elemento "param" com nome "onError" permite informar uma função JavaScript que será executada em caso de erro.

O elemento <a.../>, adicionado após os elementos "param" é utilizado para especificar um HTML alternativo que será exibido quando o silverlight não estiver instalado.

Já o elemento <iframe.../> é utilizado para garantir compatibilidade entre navegadores. A presença deste elemento impede que o navegador Safari faça cachê das páginas, por exemplo. Adicione o JavaScript da **Listagem 7** no cabeçalho da página Default.aspx.

Listagem 7. CSS e JavaScript da página

```
<head>
<title>SilverlightApplicatio3ch2</title>
    <style type="text/css">
    html, body {
            height: 100%;
            overflow: auto;
    }
    body {
            padding: 0;
            margin: 0;
    }
    #silverlightControlHost {
            height: 100%;
            text-align:center;
```

```
    }
    </style>
    <script type="text/javascript" src="Silverlight.js"></script>
    <script type="text/javascript">
        function onSilverlightError(sender, args) {
            var appSource = "";
            if (sender != null && sender != 0) {
              appSource = sender.getHost().Source;
            }

            var errorType = args.ErrorType;
            var iErrorCode = args.ErrorCode;

            if (errorType == "ImageError" || errorType == "MediaError") {
              return;
            }

            var errMsg = "Unhandled Error in Silverlight Application " +
appSource + "\n" ;

            errMsg += "Code: "+ iErrorCode + "    \n";
            errMsg += "Category: " + errorType + "       \n";
            errMsg += "Message: " + args.ErrorMessage + "     \n";

            if (errorType == "ParserError") {
                errMsg += "File: " + args.xamlFile + "     \n";
                errMsg += "Line: " + args.lineNumber + "     \n";
                errMsg += "Position: " + args.charPosition + "     \n";
            }
            else if (errorType == "RuntimeError") {
                if (args.lineNumber != 0) {
                    errMsg += "Line: " + args.lineNumber + "     \n";
                    errMsg += "Position: " +  args.charPosition + "     \n";
                }
                errMsg += "MethodName: " + args.methodName + "     \n";
            }

            throw new Error(errMsg);
        }
    </script>
</head>
```

O código JavaScript contém a função que será utilizada para tratar os erros do Silverlight. Basicamente informamos dados específicos do objeto de exceção, como código, tipo, mensagem etc.

Vamos agora iniciar o desenvolvimento da aplicação Silverlight. Quando uma nova aplicação Silverlight é criada, são adicionados na aplicação os arquivos App.xaml e MainPage.xaml. o arquivo App.xaml é a aplicação, ou seja, é semelhante ao arquivo “Program.cs” de uma aplicação Windows Forms.

Clique com o botão direito sobre o arquivo App.xaml e selecione “View Code”. Note que existem vários métodos, entre eles, “Application_Startup” e “Application_Exit”. Como os próprios nomes dizem, nestes métodos você pode implementar ações que ocorrem quando a aplicação inicia e termina.

Observe que no método “Application_Startup” está sendo atribuído um valor para a propriedade “RootVisual”. Este valor é o controle que será exibido quando a página for iniciada.

Já o arquivo MainPage.xaml é um UserControl, ou seja, um controle com extensão “.xaml” que pode ser implementado pelos desenvolvedores, permitindo a implementação de funcionalidades de interface dentro de controles re-utilizados.

Como estamos criando uma aplicação do zero, vamos excluir o arquivo MainPage.xaml e criar um novo controle que será exibido quando a aplicação for iniciada. Clique sobre o arquivo MainPage.xaml e selecione a opção “Exclude Form Project”.

Para adicionar um novo controle na aplicação Silverlight, clique com o botão direito na aplicação e selecione “Add/New Item...”. Na janela “Add New Item...” selecione o template “Silverlight User Control”, altere o campo “Name” para “FormPrincipal” e clique em “Add”. Para este controle ser exibido quando a aplicação for iniciada, vá no código do arquivo App.xaml e altere a propriedade “RootVisual” para

“**this.RootVisual = new FormPrincipal;**”.

Gerenciandores de layout

O Silverlight é composto basicamente por três gerenciadores de layout que são : StackPanel, Canvas e Grid:

- **StackPanel**: é um container que permite colocar controles um ao lado do outro (orientação horizontal) ou abaixo do outro (orientação vertical);

Canvas: é um container onde os controles são posicionados com base nas coordenadas (esquerda e topo – lefet e top);

Grid: é muito semelhante à tabelas utilizadas em aplicações Web, ou seja, cria-se o Grid e define-se o número de linhas e colunas.

Vamos criar nossa aplicação criando um Grid que contém uma coluna e três linhas. Na primeira linha vamos criar os filtros, na segunda será adicionado o DataGrid e na terceira serão adicionados os botões.

Clique sobre o controle FormPrincipal.xaml com o botão direito e selecione Open. Deixe o código como mostra **a Listagem 8**.

Listagem 8. Layout com Grid

```
<UserControl x:Class="SilverlightAplicacao.FormPrincipal"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="grdPrincipal">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>
    </Grid>
</UserControl>
```

UserControls

Até aqui apenas criamos o Grid e preparamos o container para receber os UserControls. Vamos agora inserir os controles propriamente ditos. Primeiramente, adicione uma referência aos assembly´s **Systema.Windows.Controls** e **System.Windows.Controls.Data** no projeto Silverlight. Em seguida, substitua o código do arquivo Formprincipal.xaml pelo da **Listagem 11**, que produzirá um resultado semelhante ao **da Figura 1**.

Listagem 11. XAML do UserControl FormPrinciapl

```
<UserControl x:Class="SilverlightAplicacao.FormPrincipal"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:basics="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls"
    xmlns:data="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" Loaded="UserControl_Loaded">

    <Grid x:Name="grdPrincipal" HorizontalAlignment="Left">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
```

```
        </Grid.ColumnDefinitions>

        <Border Grid.Row="0" Grid.Column="0" CornerRadius="4"
                BorderBrush="Gray" BorderThickness="1"
                Background="#E0E0E0" Height="30" Margin="0,0,0,4">
            <TextBlock Text="Controle de Pessoas" FontWeight="Bold"
                       FontSize="14" TextAlignment="Left"
                       VerticalAlignment="Center" Margin="10,0,2,0"/>
        </Border>
        <Border Grid.Row="1" Grid.Column="0" CornerRadius="4"
                BorderBrush="Gray" BorderThickness="1">
            <StackPanel Orientation="Vertical" Margin="2">
                <StackPanel Orientation="Horizontal" Margin="0,0,0,5">
                    <TextBlock Text="Nome:" FontWeight="Bold"
                               TextAlignment="Right" VerticalAlignment="Center"
                               Margin="0,0,2,0"/>
                    <TextBox x:Name="txtNome" Padding="0" Height="20" Width="445"
HorizontalAlignment="Left" />
                </StackPanel>
                <StackPanel Orientation="Horizontal" >
                    <TextBlock Text="Data Nascimento:" FontWeight="Bold"
TextAlignment="Right"
                               VerticalAlignment="Center" Margin="0,0,2,0"/>
                    <basics:DatePicker x:Name="txtDataNascimento" Padding="0"
Height="20" Width="120" HorizontalAlignment="Left" />
                    <TextBlock Text="Sexo:" FontWeight="Bold"
TextAlignment="Right" VerticalAlignment="Center" Margin="10,0,2,0"/>
                    <ComboBox x:Name="cboSexo" Height="20" Width="120"
HorizontalAlignment="Left" >
                        <ComboBoxItem Content="FEMININO"/>
                        <ComboBoxItem Content="MASCULINO"/>
                    </ComboBox>
                    <Button x:Name="btnAtualizar" Content="Atualizar" Height="20"
Width="80" Margin="10,0,2,0"></Button>
                </StackPanel>
            </StackPanel>
        </Border>

        <data:DataGrid x:Name="dgrRevista" Grid.Row="2" Grid.Column="0"
Margin="0,6,0,6" Height="200" Width="500" AutoGenerateColumns="False">
            <data:DataGrid.Columns>
                <data:DataGridTextColumn Header="Nome" Binding="{Binding Nome}"
Width="265"/>
                <data:DataGridTextColumn Header="Data Nasc." Binding="{Binding
DataNascimento}" Width="100"/>
                <data:DataGridTextColumn Header="Sexo" Binding="{Binding Sexo}"
Width="100"/>
            </data:DataGrid.Columns>
        </data:DataGrid>

        <Border x:Name="botoes" Grid.Row="4" Grid.Column="0" CornerRadius="4"
BorderThickness="1" BorderBrush="Gray" >
            <StackPanel Orientation="Horizontal" HorizontalAlignment="Right" >
                <Button x:Name="btnInserir" Content="Inserir" Height="20"
Width="65"/>
                <Button x:Name="btnEditar" Content="Editar" Height="20"
Width="60" Margin="5,0,0,0"/>
                <Button x:Name="btnExcluir" Content="Excluir" Height="20"
Width="60" Margin="5,0,2,0"/>
            </StackPanel>
        </Border>
```

```
    </Grid>
</UserControl>
```

O primeiro passo é adicionar a tag Boder, que cria uma borda com um fundo cinza onde será inserido o título do formulário. Dentro da tag Border foi inserido um rótulo (TextBlock) com o título. Os atributos Grid.Row e Grid.Column informam onde a borda será adicionada. Observe que os atributos Grid.Row e Grid.Column foram configurados para que o título seja exibido na primeira e coluna do Grid.

O segundo passo foi adicionar uma Border para os filtros. Observe o atributo CornerRadius, que define o canto arredondado e, o atributo BorderBrush define a cor e o atributo BorderTickness define a espessura da borda. Dentro da tag Border utilizamos o StackPanel para agrupar os ítens.

O terceiro passo é adicionar o DataGrid onde serão exibidos os dados retornados pelo Web Service. Para utilizar o controle DataGrid é necessário a referência ao assembly "System.Windows.Controls.Data".

O quarto passo é adicionar os botões no final do controle. Para isto foi criado uma nova borda e um StackPanel para acomodar os botões.

Adicionando referência ao serviço

O próximo passo, antes de começar a implementação do código C#, é adicionar uma referência ao Web Service. Clique com o botão direito no projeto Silverlight e selecione "Add Service Reference...". Informe o endereço do Web Service (se não souber clique no botão Discover e selecione "Service in Solution"), altere a propriedade Namespace para WSProDIntegrado e clique em OK. Será criada uma pasta chamada "Service References", que contém o Web Service.

Implementando o código C# do controle principal

primeiramente, vamos explicar como funciona no Silverlight chamadas a métodos de um Web Service. Quando adicionamos uma referência ao serviço, o Visual Studio gera um código que implementa os métodos descritos pelo WSDL do Web Service. Para cada método do Web Service é gerada uma estrutura que contém um método que é responsável por iniciar a requisição e um evento que será disparado quando a chamada for concluída (callback), retornando os dados da solicitação. Por exemplo, para consultar as pessoas, é preciso invocar o método ConsultarPessoaAsync que consulta os dados no Web Service e implementa o evento ConsultarPessoaCompleted. O evento ConsultarPessoaCompleted é

executado quando a consulta dos dados é concluída. Ou seja, as coisas acontecem de forma assíncrona, o que significa que a aplicação não para e fica aguardando até que os dados sejam consultados, mas apenas dispara um pedido para o Web Service e continua trabalhando normalmente até que receba alguma resposta (neste caso o evento ConsultarPessoaCompleted).

Clique com o botão direito no controle FormPrincipal e escolha a opção "View Code". Instancie o Web Service no início da classe dando o nome de wsRevista, conforme a **Listagem 12**.

Vamos programar o evento Loaded do UserControl. Na tag UserControl, logo após as declarações, adicione o atributo (evento) Loaded conforme segue:

```
<UserControl x:Class="SilverlightAplicacao.FormPrincipal"
    …  Loaded="UserControl_Loaded">
```

Listagem 12. Código C# do formulário principal

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.ComponentModel;
using SilverlightAplicacao.ServiceReference1;

namespace SilverlightAplicacao
{
    public partial class FormPrincipal : UserControl
    {
        WSPessoaSoapClient wsrPessoa = new WSPessoaSoapClient();

        public FormPrincipal()
        {
            InitializeComponent();
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {
```

```
            wsrPessoa.ConsultarPessoaCompleted += new
EventHandler<ConsultarPessoaCompletedEventArgs>(wsrPessoa_ConsultarPessoaComplete
d);
            wsrPessoa.ExcluirPessoaCompleted += new
EventHandler<AsyncCompletedEventArgs>(wsrPessoa_ExcluirPessoaCompleted);
            cboSexo.SelectedIndex = 0;
        }

        void wsrPessoa_ExcluirPessoaCompleted(object sender,
AsyncCompletedEventArgs e)
        {
            btnAtualizar_Click(null, null);
            MessageBox.Show("Registro excluído com sucesso!", "Aviso",
MessageBoxButton.OK);
        }

        void wsrPessoa_ConsultarPessoaCompleted(object sender,
ConsultarPessoaCompletedEventArgs e)
        {
            btnAtualizar.IsEnabled = true;
            if (e.Error == null)
            {
                dgrPessoa.ItemsSource = e.Result;
            }
        }

        private void btnAtualizar_Click(object sender, RoutedEventArgs e)
        {
            btnAtualizar.IsEnabled = false;
            wsrPessoa.ConsultarPessoaAsync (null,
txtNome.Text.Equals(string.Empty) ? txtNome.Text,
                txtDataNascimento.Text.Equals(string.Empty) ? (DateTime?)null :
(DateTime)Convert.ToDateTime(txtDataNascimento.Text),
                cboSexo.SelectedIndex.Equals(0) ? false : true);
        }

        private void btnExcluir_Click(object sender, RoutedEventArgs e)
        {
            if (dgrPessoa.SelectedIndex != null)
            {
                if (MessageBox.Show("Tem certeza?", "Alerta",
MessageBoxButton.OKCancel) == MessageBoxResult.OK)
wsrPessoa.ExcluirPessoaAsync(new Pessoa
                {
                    IDPessoa = (dgrPessoa.SelectedItem as Pessoa).IDPessoa
                });

            }
            else
                MessageBox.Show("nenhum registro seecionado", "Aviso",
MessageBoxButton.OK);
        }

        private void btnInserir_Click(object sender, RoutedEventArgs e)
        {
            FormCadastrar frmCadstrar = new FormCadastrar();
            this.Content = frmCadstrar;
        }

        private void btnEditar_Click(object sender, RoutedEventArgs e)
        {
            if (dgrPessoa.SelectedItem != null)
```

```
            {
                FormCadastrar frmCadastrar = new
FormCadastrar((dgrPessoa.SelectedItem as Pessoa));
                this.Content = frmCadastrar;

            }
            else
            {
                MessageBox.Show("Nenhum registro selecionado.", "Aviso",
MessageBoxButton.OK);
            }
        }

        private void btnAtualizar_Click_1(object sender, RoutedEventArgs e)
        {

        }

        private void btnInserir_Click_1(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

Pronto, a interface do controle principal está completa, vamos agora criar o segundo controle que será responsável por inserir e alterar registros. Este formulário é bem simples, será composto apenas pelo nome, data nascimento e sexo.

Adicione um novo controle e altere o nome para "FormCadastro.xaml". a **Listagem 13** mostra o código XAML.

Listagem 13. XAML do UserControl FormCadastrar

```
<UserControl x:Class="SilverlightAplicacao.FormCadastrar"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:basics="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls">

    <Grid HorizontalAlignment="Left" >
        <StackPanel Orientation="Vertical" Width="425">
            <Border CornerRadius="4" BorderBrush="Gray" BorderThickness="1"
Background="#E0#0#0" Height="30" Margin="0,0,0,4">
                <TextBlock x:Name="txtTitulo" Text="Cadastrar Pessoas"
FontWeight="Bold" FontSize="14" TextAlignment="Left" VerticalAlignment="Center"
Margin="10,0,2,0"/>
            </Border>
            <Border CornerRadius="4" BorderBrush="Gray" BorderThickness="1"
Margin="0,0,0,4">
                <Grid Margin="4">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
```

```
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <TextBlock Text="Nome:" Grid.Row="0" Grid.Column="0"
FontWeight="Bold" TextAlignment="Right" VerticalAlignment="Center"
Margin="0,4,2,0"/>
                <TextBox x:Name="txtNome" Grid.Row="0" Grid.Column="1"
Padding="0" Height="20" Width="300" HorizontalAlignment="Left" Margin="0,4,0,0"/>
                <TextBlock Text="Data Nascimento:" Grid.Row="1" Grid.="0"
FontWeight="Bold" TextAlignment="Right" VerticalAlignment="Center"
Margin="0,4,2,0"/>
                <basics:DatePicker x:Name="txtDataNascimento" Grid.Row="1"
Grid.Column="1" Padding="0" Height="20" Width="120" HorizontalAlignment="Left"
Margin="0,4,0,0"/>
                <TextBlock Text="Sexo:" Grid.Row="2" Grid.Column="0"
FontWeight="Bold" TextAlignment="Right" VerticalAlignment="Center"
Margin="10,4,2,0"/>
                <ComboBox x:Name="cboSexo" Grid.Row="2" Grid.Column="1"
Height="20" Width="120" HorizontalAlignment="Left" Margin="0,4,0,0">
                    <ComboBoxItem Content="FEMININO"/>
                    <ComboBoxItem Content="MASCULINO"/>
                </ComboBox>
            </Grid>
        </Border>
        <StackPanel Orientation="Horizontal" Grid.Row="2" Grid.Column="0"
HorizontalAlignment="Right" >
            <Button x:Name="btnSalvar" Content="Salvar" Height="20"
Width="65"/>
            <Button x:Name="btnCancelar" Command="Cancelar" Height="20"
Width="65" Margin="5,0,0,0"/>
        </StackPanel>
    </StackPanel>
</Grid>
</UserControl>
```

Implementando o código do controle de cadastro

O código C# do formulário FormCadastrar deve ser implementado conforme a **Listagem 14**.

Instanciamos o Web service logo no início da classe dando o nome wsrPessoa. Então criamos uma propriedade do tipo Pessoa e damos o nome de PessoaCadastrada. Esta propriedade vai receber o objeto pessoa quando o usuário for realizar uma edição. Adicionamos um segundo construtor que recebe um objeto pessoa e seta este objeto com o valor da variável PessoaCadastrada. Então codificamos o evento Loaded do UserControl. Na tag UserControl, logo após as declarações, adicione o atributo (evento) Loaded conforme segue:

```
<UserControl x:Class="SilverlightAplicacao.FormCadastrar"
   ...  Loaded="UserControl_Loaded">
```

Listagem 14. Código C# do formulário de cadastro

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using SilverlightAplicacao.ServiceReference1;
using System.ComponentModel;

namespace SilverlightAplicacao
{
    public partial class FormCadastrar : UserControl
    {
        WSPessoaSoapClient wsrPessoa = new WSPessoaSoapClient();
        private Pessoa pessoaCadastrada;

        public FormCadastrar()
        {
            InitializeComponent();
        }

        public FormCadastrar(Pessoa objPessoa)
        {
            InitializeComponent();
            pessoaCadastrada = objPessoa;
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {
            wsrPessoa.InserirPessoaCompleted += new
EventHandler<AsyncCompletedEventArgs>(wsrPessoa_InserirPessoaCompleted);
            wsrPessoa.AlterarPessoaCompleted += new
EventHandler<AsyncCompletedEventArgs>(wsrPessoa_AlterarPessoaCompleted);

            if (pessoaCadastrada != null)
            {
                txtNome.Text = pessoaCadastrada.Nome;
                txtDataNascimento.Text =
pessoaCadastrada.DataNascimento.ToString("dd/MM/YYYY");
                cboSexo.SelectedIndex = pessoaCadastrada.Sexo == false ? 0 : 1;
            }
            else
            {
                cboSexo.SelectedIndex = 0;
            }
        }

        void wsrPessoa_AlterarPessoaCompleted(object sender,
AsyncCompletedEventArgs e)
        {
            MessageBox.Show("Registro alterado com sucesso!", "Aviso",
MessageBoxButton.OK);
            FormPrincipal frmPrincipal = new FormPrincipal();
            this.Content = frmPrincipal;
        }
```

```
        void wsrPessoa_InserirPessoaCompleted(object sender,
AsyncCompletedEventArgs e)
        {
            MessageBox.Show("Registro inserido com
sucesso!","Aviso",MessageBoxButton.OK);
            FormPrincipal frmPrincipal = new FormPrincipal();
            this.Content = frmPrincipal;
        }

        private void btnSalvar_Click(object sender, RoutedEventArgs e)
        {
            if (VerificarCamposEstaoPreenchidos())
            {
                if (pessoaCadastrada != null)
                    wsrPessoa.AlterarPessoaAsync(new Pessoa
                    {
                        IDPessoa = pessoaCadastrada.IDPessoa,
                        Nome = txtNome.Text,
                        DataNascimento =
Convert.ToDateTime(txtDataNascimento.Text),
                        Sexo = cboSexo.SelectedIndex == 0 ? false : true});
                 else
                    wsrPessoa.InserirPessoaAsync(new Pessoa {Nome = txtNome.Text,
                                                             DataNascimento =
Convert.ToDateTime(txtDataNascimento.Text),

Sexo=cboSexo.SelectedIndex == 0 ? false : true });
            }
        }

        private bool VerificarCamposEstaoPreenchidos()
        {
            if (txtNome.Text.Equals(string.Empty))
            {
                MessageBox.Show("O campo Nome é obrigatório.", "Aviso",
MessageBoxButton.OK);
                txtNome.Focus();
                return false;
            }
            else if (txtDataNascimento.Text.Equals(string.Empty))
            {
                MessageBox.Show("O campo Data de Nascimento é
obrigatório.","Aviso",MessageBoxButton.OK);
                txtDataNascimento.Focus();
                return false;
            }
            return true;
        }

        private void btnCancelar_Click(object sender, RoutedEventArgs e)
        {
            FormPrincipal frmPrincipal = new FormPrincipal();
            this.Content = frmPrincipal;
        }
    }
}
```

Primeiramente implementamos um “Event Handler” para o método InserirPessoaCompleted, ou seja, informamos que será executado o método wsrPessoa_InserirPessoaCompleted quando o evento InserirPessoaCompledt for disparado. Logo abaixo foi implementado o evento “Event Handler” para o método AlterarPessoaCompleted, ou seja, informamos que será executado o método wsrPessoa_AlterarPessoaComplted quando o evento AlterarPessoaCompleted for disparado. Ainda no evento Loaded será verificado se a propriedade PessoaCadastrada não é nula, o que significa que se a propriedade PessoaCadastrada for diferente de nula o usuário vai alterar uma pessoa. Neste caso, os controles devem ser carregados com os dados do objeto PessoaCadastrada. No método wsrPessoa_AlterarPessoaCompleted exibimos uma mensagem de confirmação para o usuário quando o registro for alterado com sucesso. O método wsrPessoa_InserirPessoaCompleted exibe uma mensagem de confirmação para o usuário e volta para o controle principal.

Agora podemos fazer um teste, compile e execute sua aplicação. Clique no botão Inserir do formulário principal para cadastrar algumas pessoas. Em seguida faça uma busca preenchendo os filtros, selecione uma pessoa no DataGrid, altere os dados exclua a pessoa.