Exercise 7

Creating a web-based application "Virtual Library" Connection with other applications using the communication technologies NET Remoting and Web Services

PHASE I. Creating a website "Virtual Library"

1. Create a new folder Client_Server_2. Copy the final project eLibrary_v3 and the database file - biblio.mdf from the first part of the semester. Only the server part of the project will be used - class for remote access to data 'DAL'. This class has to be changed a little, as the new application will be web-based it is necessary to extract only part of the data, not the entire table with 9000 entries, like in desktop applications.

2. Open **eLibraryClient.sln** with the three projects and from the server-side application, open class DAL.

Add the following:

- A method **getBooksByCategory**, which is similar to the existing getBookByISBN (it can be copied from the existing), but selects only some of the fields (no Description)

```
  public DataTable getBooksByCategory(String pCategory)
        {
            DataTable dt = new DataTable();
            SqlConnection connection = new SqlConnection();
            SqlCommand cmd = new SqlCommand();

            connection.ConnectionString =
eLibraryServer.Properties.Settings.Default.biblioConnectionString;
            try
            {
                using (connection)
                {
                    connection.Open();
                    cmd.Connection = connection;
cmd.CommandText = "Select Authors, Title, ISBN from Titles WHERE BookTypeID ='" + pCategory + "'";
                    dt.Load(cmd.ExecuteReader());
                }
            }
            catch (SqlException e) { connection.Close(); }
            return dt;
        }
```

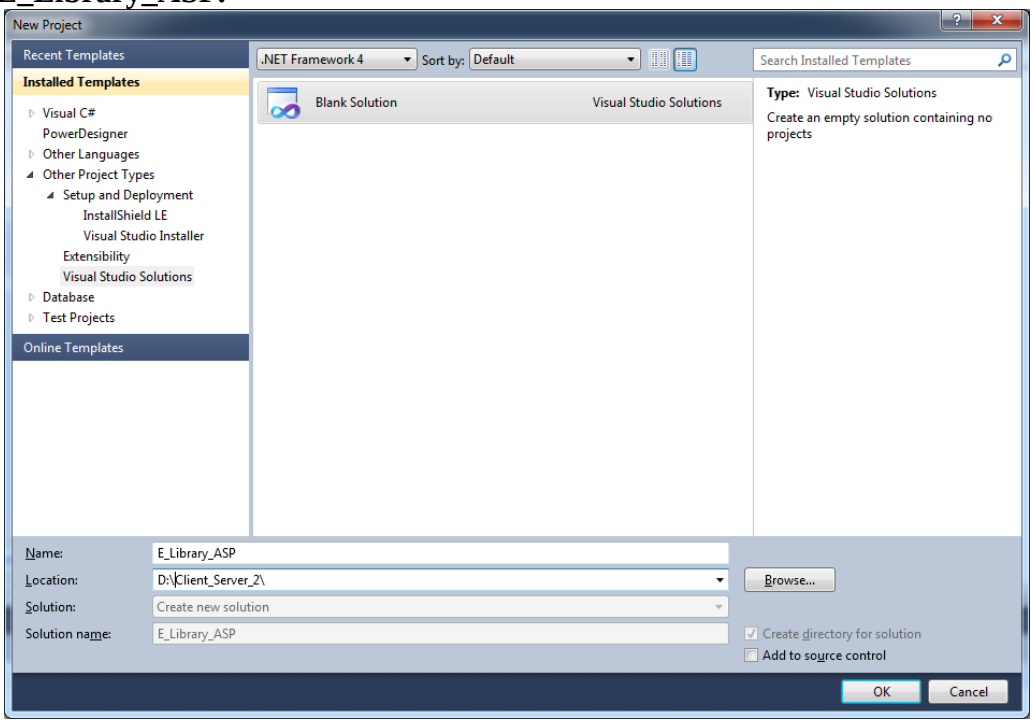
- A method getBookTypes, which returns only the types of the books.

```
public DataTable getBookTypes()
        {   DataTable dt = new DataTable();
            SqlConnection connection = new SqlConnection();
            SqlCommand cmd = new SqlCommand();
```

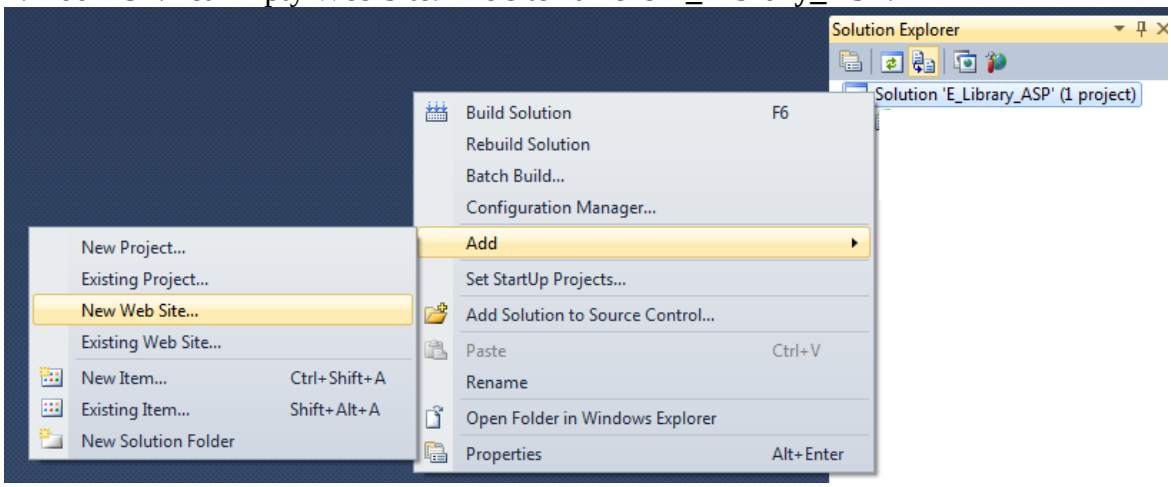
```
    connection.ConnectionString = eLibraryServer.Properties.Settings.Default.biblioConnectionString;
            try
            {
                using (connection)
                {
                    connection.Open();
                    cmd.Connection = connection;
                    cmd.CommandText = "Select * from BookTypes ";
                    dt.Load(cmd.ExecuteReader());
                }
            }
            catch (SqlException e) { connection.Close(); }
            return dt;
        }
```

Close the application.

3. Create a new blank solution, locate it in the folder **Client_Server_2** and give it a name **E_Library_ASP.**



4. Add ASP.Net Empty Web Site. The site name is **E_Library_ASP**.

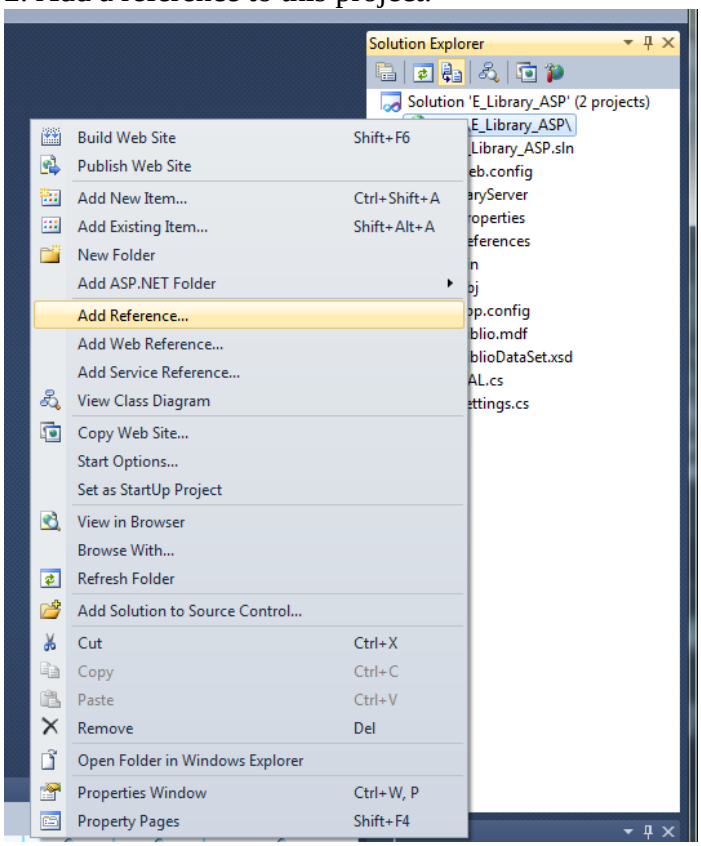


Adding the eLibraryServer project

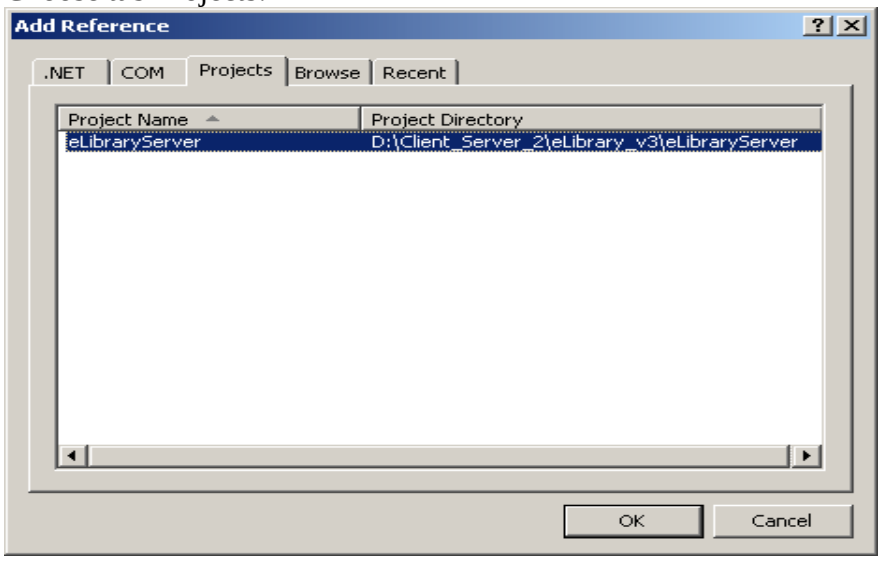
1. Add the existing project eLibraryServer from eLibrary_v3 to the created (empty for now) site solution:

File->Add->Existing Project

2. Add a reference to this project.



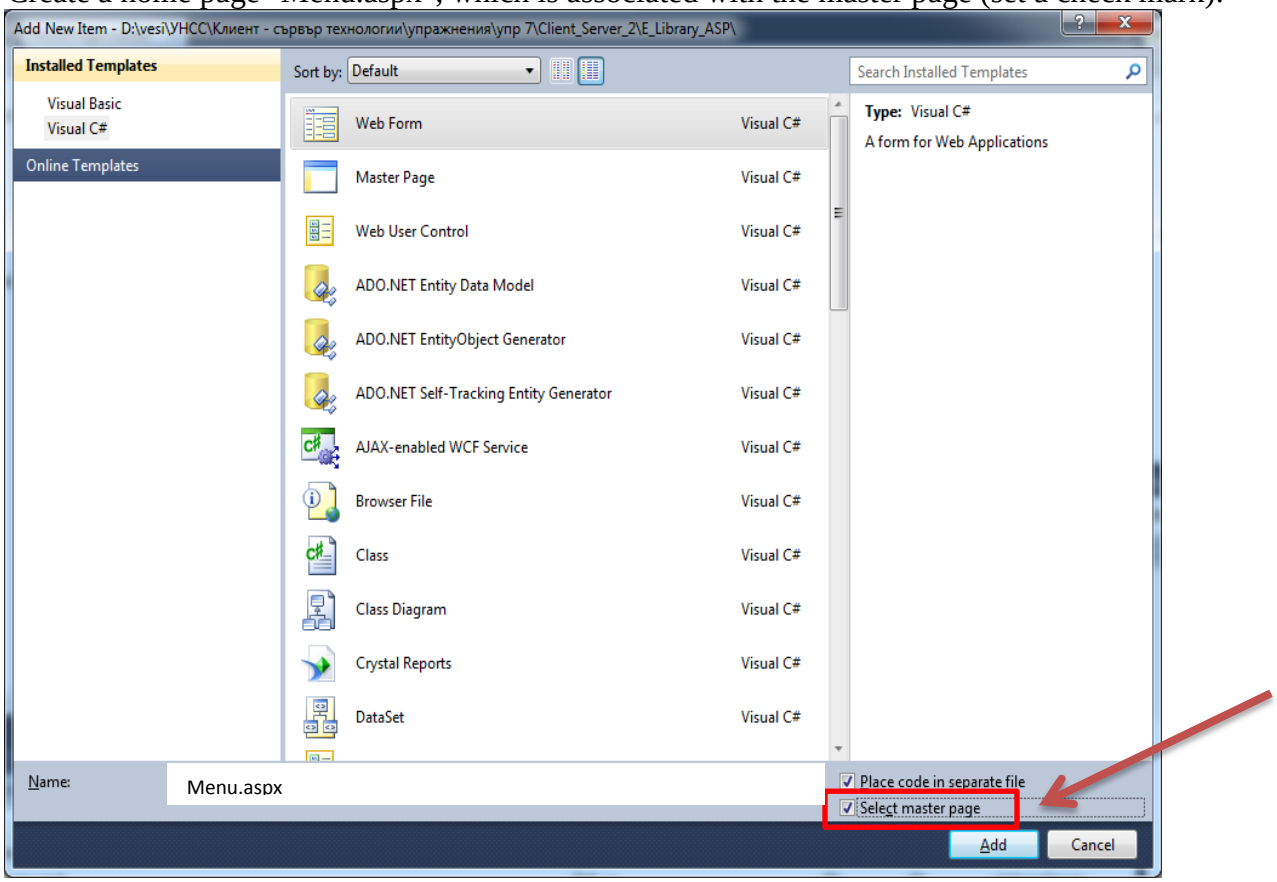
Choose tab Projects:



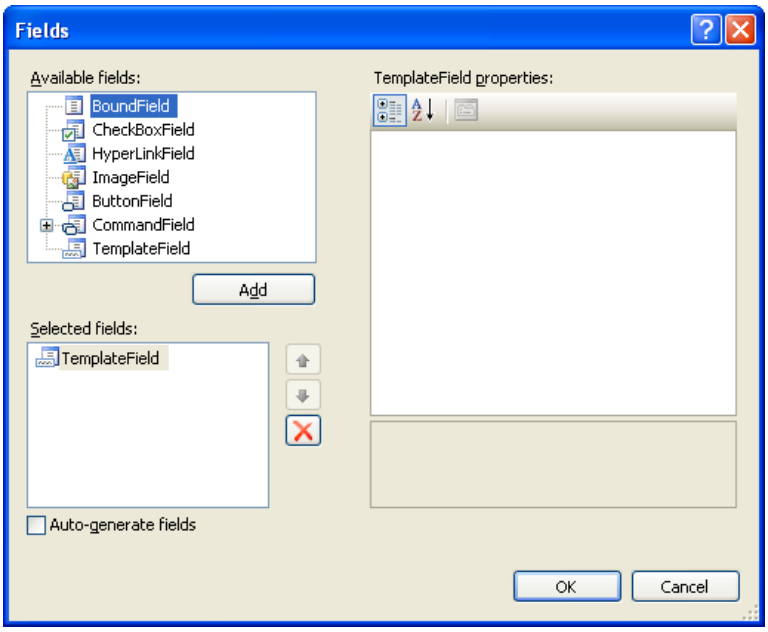
CREATING A WEBSITE "VIRTUAL LIBRARY"

Create a project of the site, which should look something like desktop application from the first part of the semester.
Create a Master page to organize the application interface. Let the page be divided into three parts, two rows, on the second row - 2 columns.
Write "Virtual Library" in the first row using the appropriate control and apply a style on your own.
put a `ContentPlaceHolder` in the second row in the both parts.

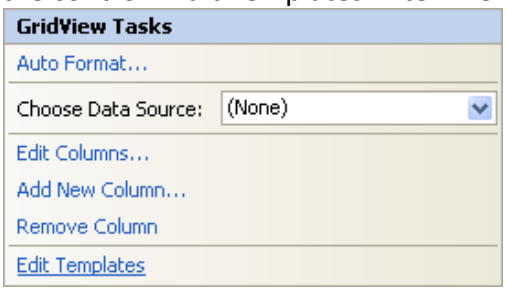
1. Create a home page "Menu.aspx", which is associated with the master page (set a check mark):



Put the following controls in the left side of the page:
2. A label with text "menu and categories."
3. A gridview control. From the GridView, choose Edit Columns, then add TemplateFields and turn off Auto-generate fields.



4. Choose from the smart tag of the control - Edit Templates > Item Template



5. Add a control - LinkButton.

6. Select the Link Button in command mode. Add the Text property and enter the following commands:

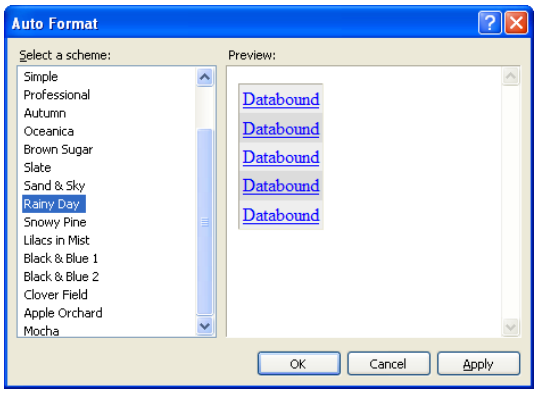
```
<asp:LinkButton ID="LinkButton1"
                runat="server"
                Text='<%# Eval("BookTypeName")%>'
                OnCommand="Get_Category"
                CommandName='<%#Eval("BookTypeID")%>'>
            </asp:LinkButton>
```

7. At the bottom of the page put a label, and use it to display a message in case of error.

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

8. Set the GridView ShowHeader setting to False.

9. Choose an appropriate formatting from the smart tag on the GridView , Auto Format.



10. Add the following code in **Menu.aspx.cs**

```
protected void Page_Load(object sender, EventArgs e)
    {
        DataTable bookTypes;
        eLibraryServer.DAL  oDAL= new eLibraryServer.DAL();

        try
        {
            bookTypes = oDAL.getBookTypes();
            GridView1.DataSource = bookTypes.DefaultView;
            GridView1.DataBind();
        }
        catch (Exception ex){ LabelErr.Text = "No link to the Data Server!";}

    }
    public void Get_Category(Object Src, CommandEventArgs Args)
    {
        Response.Redirect("Menu.aspx?Category=" + Args.CommandName);
    }
```

When the menu is loading, there is a new object oDAL from the class eLibraryServer.DAL, which is located in the server application. The menu's data is loaded using this class, which communicates with the database.

Test the menu.