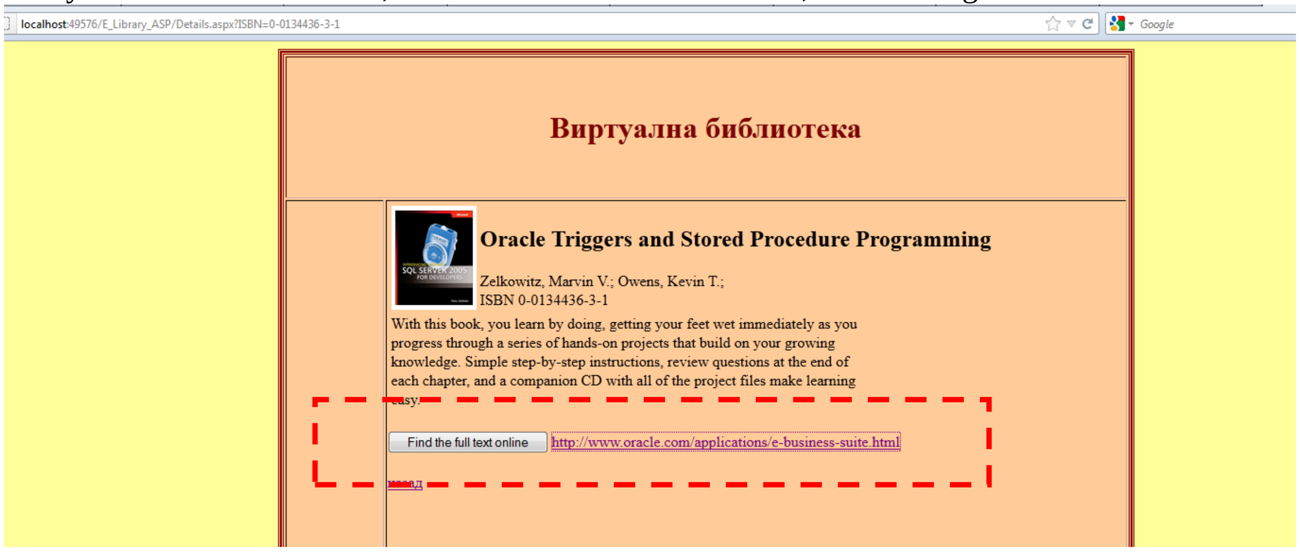
Exercise 11

Phase III-B. Converting eLibrary_ASP in consumer of a web-service

Purposes

Add a button in the eLibrary_ASP site's page for detailed information about a book. On the button click, search and display the URL for the case when a book is available in electronic format.
In the database ***biblio.mdf***, used by the website through eLibraryServer, there is no such information. Therefore, when you click the new button, the site will call eBookServices, which will give the URL of the book.


localhost:49576/E_Library_ASP/Details.aspx?ISBN=0-0134436-3-1

Виртуална библиотека

Oracle Triggers and Stored Procedure Programming

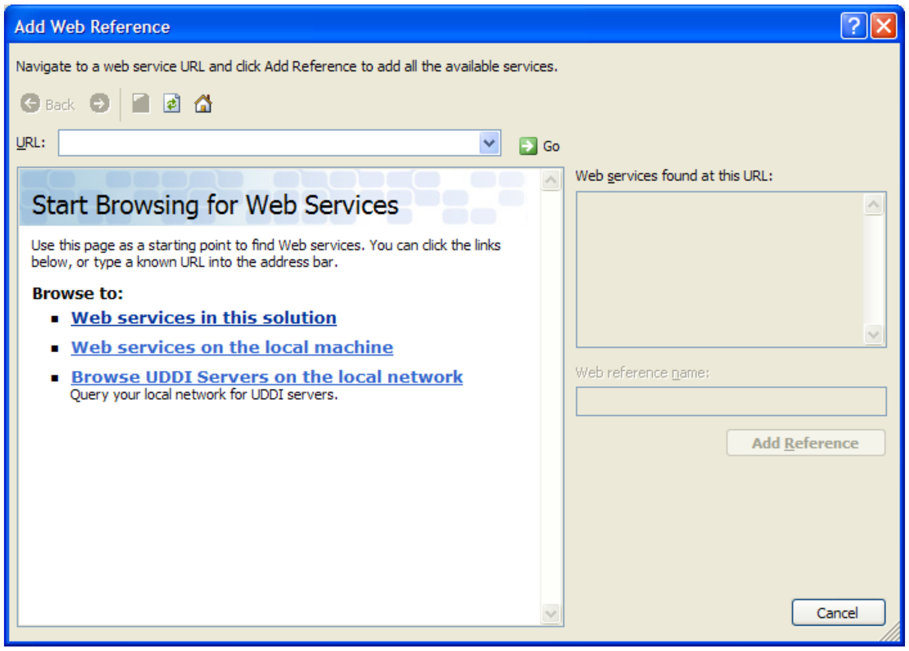
Zelkowitz, Marvin V.; Owens, Kevin T.;
ISBN 0-0134436-3-1

With this book, you learn by doing, getting your feet wet immediately as you progress through a series of hands-on projects that build on your growing knowledge. Simple step-by-step instructions, review questions at the end of each chapter, and a companion CD with all of the project files make learning easy.

Find the full text online
http://www.oracle.com/applications/e-business-suite.html
назад

Step1

1 a) Open the context menu for the eLibrary_ASP site, and then click ***Add Web Reference***
The following panel will open:


Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

Start Browsing for Web Services

Use this page as a starting point to find Web services. You can click the links below, or type a known URL into the address bar.

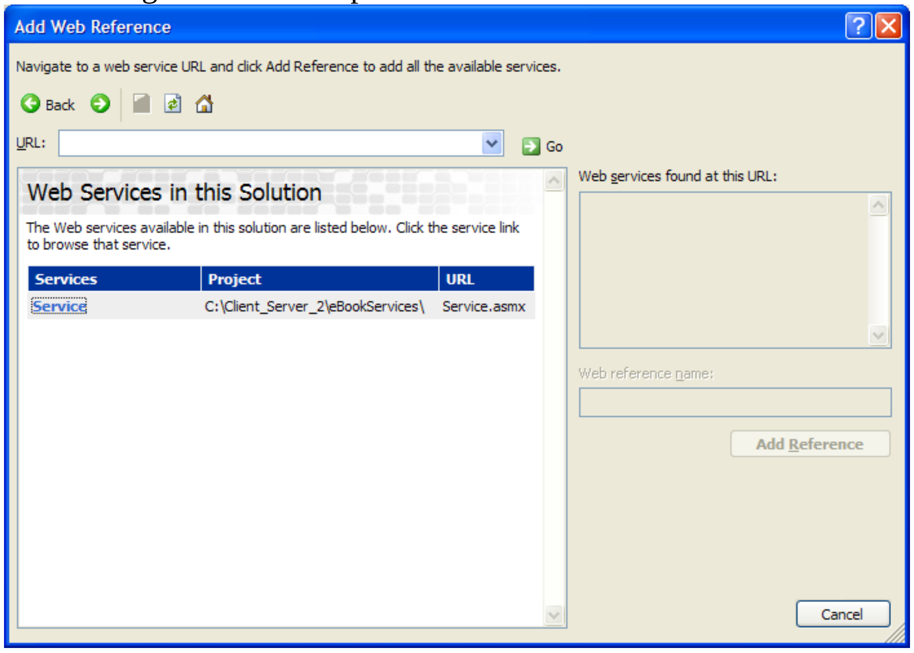
Browse to:

- Web services in this solution
- Web services on the local machine
- Browse UDDI Servers on the local network
  Query your local network for UDDI servers.

1 б)

Choose ***Web services in this solution.***

The following panel for selecting services will open:



Navigate to a web service URL and click Add Reference to add all the available services.

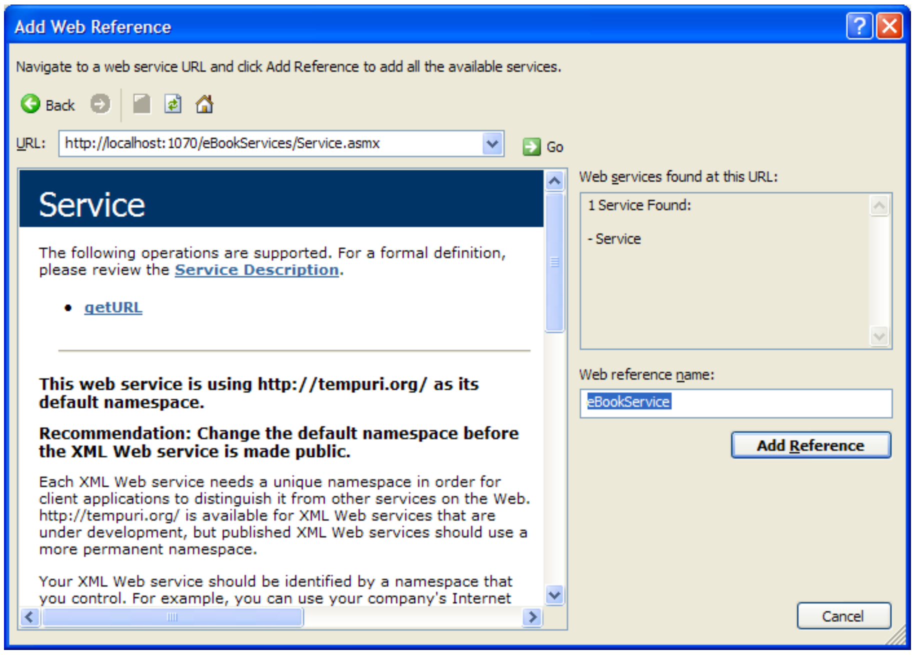
Web Services in this Solution

The Web services available in this solution are listed below. Click the service link to browse that service.

Services	Project	URL
Service	C:\Client_Server_2\eBookServices\	Service.asmx

Choose ***Service***

In the *web service name*: Specify the name, which we will refer to this external website providing services when we refer to it in eLibrary_ASP. Change the name by default "***localhost***" to eBookService, for example.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:1070/eBookServices/Service.asmx

Web services found at this URL:

1 Service Found:

- Service

Web reference name: eBookService

Service

The following operations are supported. For a formal definition, please review the Service Description.

- getURL

This web service is using http://tempuri.org/ as its default namespace.

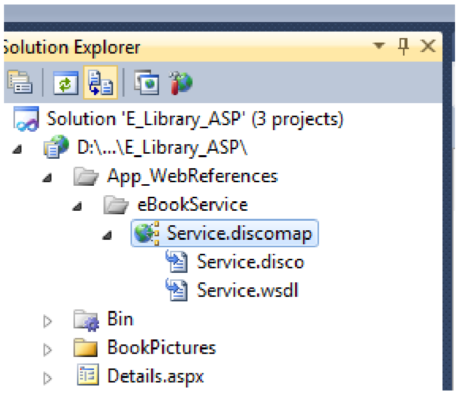
Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet

Click the button ***Add Reference***

As a result, in the web site a folder ***App_WebReferences*** with the following files is added:



Step 2

Viewing files that describe the provider of the web site service

NOTE: This step is optional. Visual Studio has read these files for us. Using the methods available as Web services - within Visual Studio - is not different from the way we use the methods of any other external class.

Service.discomap

Service.discomap *is a help file that contains the URL of two files describing the site provider of web services*:

- Service.disco – `DiscoveryDocumentReference` –
  Obtained from the Service.asmx using parameter disco
  `In this case:  url="http://localhost:49421/Services/Service.asmx?disco"`

- Service.wsdl – `ContractReference`
  Obtained from the Service.asmx using parameter wsdl
  `In this case:  url= http://localhost:49421/Services/Service.asmx?wsdl`

Service.disco

The file Service.disco *[disco =* ***disco****very] contains 2 URLs:*

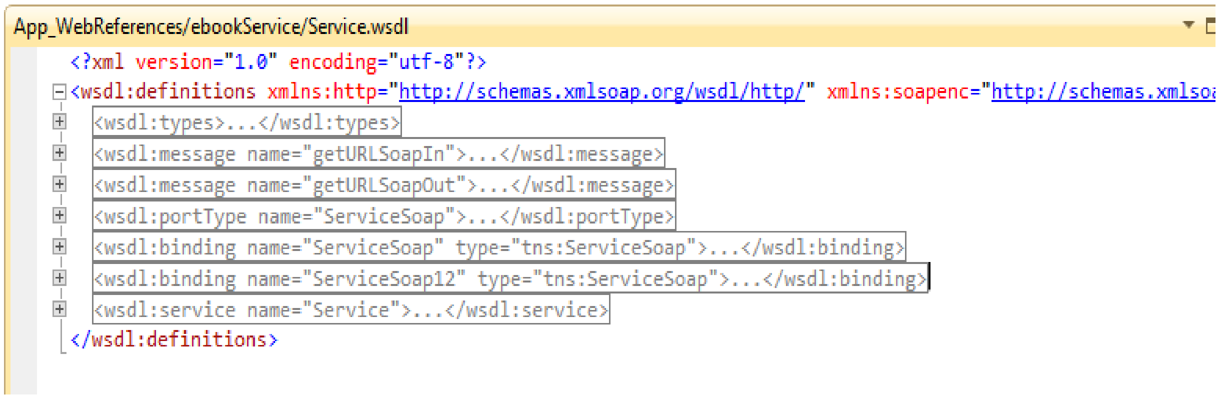
- *Address for services invoking* `< soap address= …>`
- *Address file WSDL with description of services* `<contractRef ref= …>`

`In this case:`

```
<contractRef ref="http://localhost:49421/Services/Service.asmx?wsdl"
              docRef="http://localhost:49421/Services/Service.asmx"
              xmlns="http://schemas.xmlsoap.org/disco/scl/" />
  <soap address="http://localhost:49421/Services/Service.asmx"
        xmlns:q1="http://tempuri.org/" binding="q1:ServiceSoap"
        xmlns="http://schemas.xmlsoap.org/disco/soap/" />
```

Service.wsdl

Service.wsdl is a file with standard WSDL description of the service website eBookService.
The figure represented the first level tags**.**


```
App_WebReferences/ebookService/Service.wsdl
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soapenc="http://schemas.xmlsoa
  <wsdl:types>...</wsdl:types>
  <wsdl:message name="getURLSoapIn">...</wsdl:message>
  <wsdl:message name="getURLSoapOut">...</wsdl:message>
  <wsdl:portType name="ServiceSoap">...</wsdl:portType>
  <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">...</wsdl:binding>
  <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">...</wsdl:binding>
  <wsdl:service name="Service">...</wsdl:service>
</wsdl:definitions>
```

The element <**definitions**> is a main (root) element in the document WSDL. All other items are placed inside.

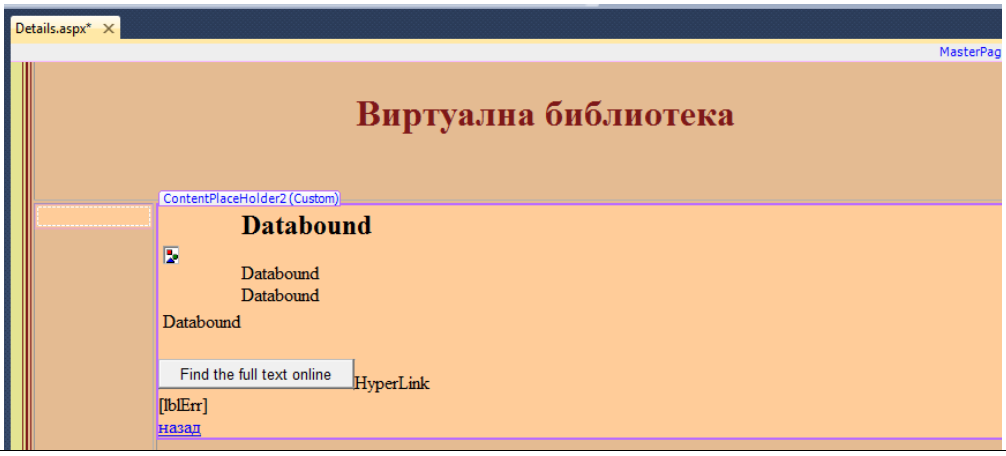
The element **<service>** specifies the name of the web service. In this element the different ports on which the web service is available are set.

The element**<message>** reflects a process of communication with the service. Each service includes sending a request and receiving a response. For each function described in WSDL (in this case only getURL) there are In and Out.

The element **<binding>** indicates the way in which the data should be encoded. In this case, there is a dependent specification of SOAP, which contains a predefined set of rules for such situations.

Step 3

Open the page **Details.aspx** in design mode and add:
- a button named `btnFindURL`
- a hyperlink named `HyperLinkFound` with property `Target="_blank"`



```
<asp:Button ID="btnFindURL" runat="server" Text="Find the full text online"
        Width="170px" />
    <asp:HyperLink ID="HyperLinkFound" runat="server"
Target="_blank">HyperLink</asp:HyperLink>
```

In this site there is no textbox for ISBN, because there is a label with id = "ISBN" and it is not necessary.

```
<asp:Label id="ISBN" Runat="Server" Text='<%# "ISBN  " & Eval("ISBN") %>'/>
```

Step 4

In ***Details.aspx.cs*** page add a mehod **btnFindURL_Click** for the click event of the button btnFindURL.

```
protected void btnFindURL_Click(object sender, EventArgs e)
    {   eBookService.Service oDAL=new eBookService.Service();
        String strURL ;
        try
        {   strURL = oDAL.getURL(strISBN);
        }
        catch(Exception)
        {
            strURL = "No connection to Database";
        }
        if (strURL !="")
        {  HyperLinkFound.Text = strURL;
           HyperLinkFound.NavigateUrl = strURL;
        }
        else
        {   HyperLink1.Text = "Sorry, we have not found an e-book version!";
            HyperLink1.NavigateUrl ="";
        }
    }
```

Step 5

Comparison of the two technologies – ***Net Remoting*** and ***Web Services***
Let’s compare the functions in ***Details.aspx.cs***:

- **btnFindURL_Click** and
- **Page_Load**

In the function **btnFindURL_Click there** is defined an object *oDAL* from *class eBookService.Service*.
In the function **form1_Load there** is defined an object *oDAL* from class *eLibraryServer.DAL.*
Compared to the web form, the two sources have the same role - delivering data for visualization. The two objects are used in same way.

However, there are some restrictions on the **types of data** that can be passed as parameters and the resultant value of the functions of the web services. It can be only of basic data types.
Using Net remoting, where the client and the server are applications in the environment .NET. there can be transmitted objects of any type(each class) that is defined serialization. (for example: can be passed complex objects as ***Dataset)***

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
[WebMethod]
    public String getURL(String pISBN)
    {   String strReturn;
        strReturn = SearchInDB.getURL(pISBN);
        return strReturn;
    }
}
```

The public method available as a web service is getURL(). It calls already established method ***SearchInDB.getURL***(pISBN).