

We Translate
Business Processes

from the Mind
to the Computer
to the Bottom Line.

BUSINESS & COMPUTERS, Inc.

13839 Mur-Len Rd, Suite M
OLATHE, KANSAS 66062

Phone: (913) 764-2311

Fax: 764 7515

larryg@kcnet.com

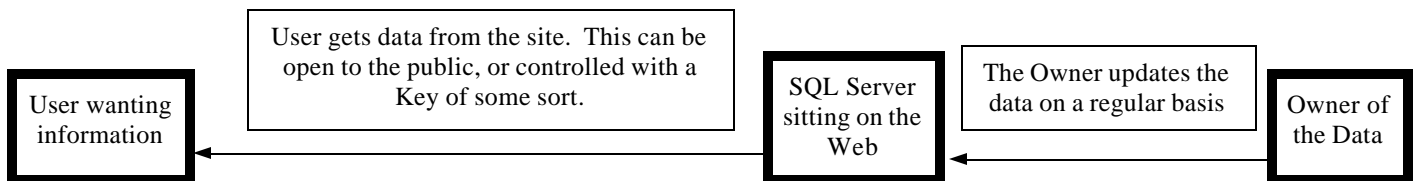
September 11, 2003

Transmitting Data to and from SQL Server by Way of a Website using VB.NET and ADO.NET

Copyright© 2003 Business & Computers, Inc.

A note – the below is my humble opinion – with testing – If you use my ideas
please test them and if you have problems or learn more let me know.

1) What is the most simple form of transmitting data:



Example 1: Make sure you have the right address

The screenshot shows a web application interface on the left and a browser window on the right. The interface has three columns: 'Enter Address', 'Push the button', and 'Get the correct address'. Under 'Enter Address', there are input fields for Street (13839 Murlen Suite M), City (olathe), and State (ks). Under 'Push the button', there is a 'Get Corrected Address' button. Under 'Get the correct address', there are output fields for Street (13839 S MURLEN RD STE M), City (OLATHE), State (KS), Short ZIP (66062), and Full Zip (66062-1662). A 'Close Form' button is at the bottom. The browser window shows the URL <http://www.dev1.eraserver.net/WebServices/zipcoderesolver/zipcoderesolver.asmx/CorrectedAddressXml?accessCode=9999&address=13839+Murlen+suite+M&city=Olathe&state=ks> and the following XML response:

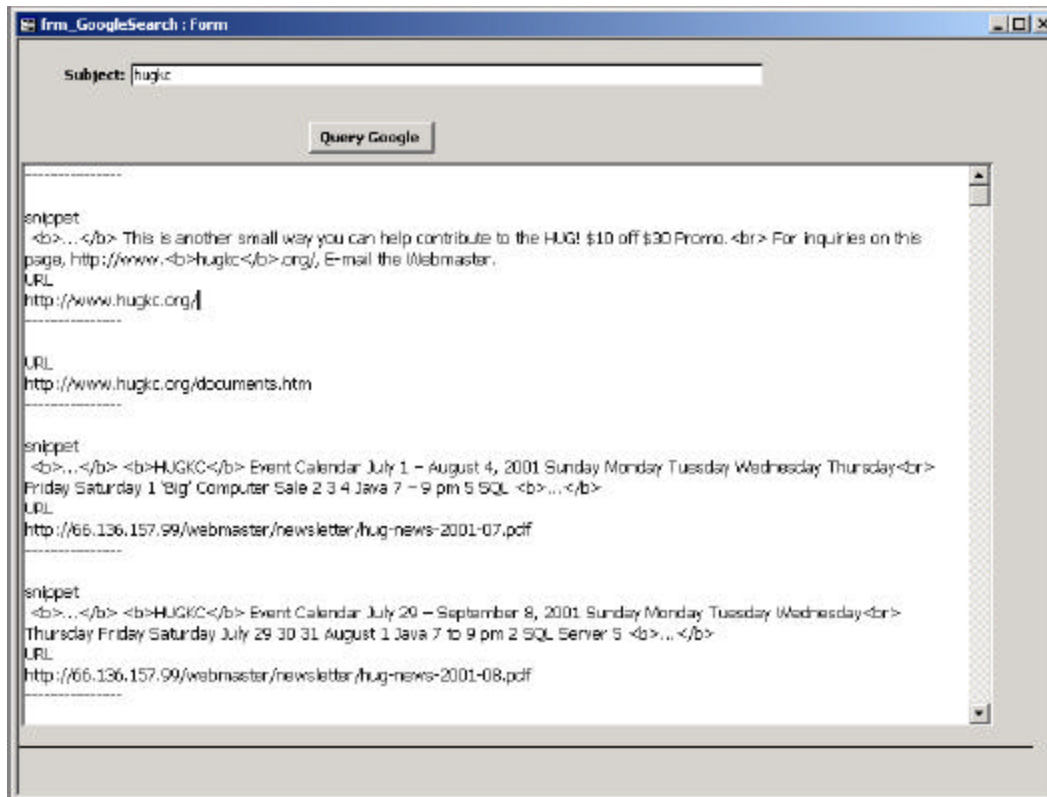
```

<?xml version='1.0' encoding='utf-8' ?>
<USPSAddress
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xmlns='http://webservices.eraserver.net/'>
  <Street>1441 E SLEEPY HOLLOW DR</Street>
  <City>OLATHE</City>
  <State>KS</State>
  <ShortZIP>66062</ShortZIP>
  <FullZIP>66062-2283</FullZIP>
</USPSAddress>
  
```

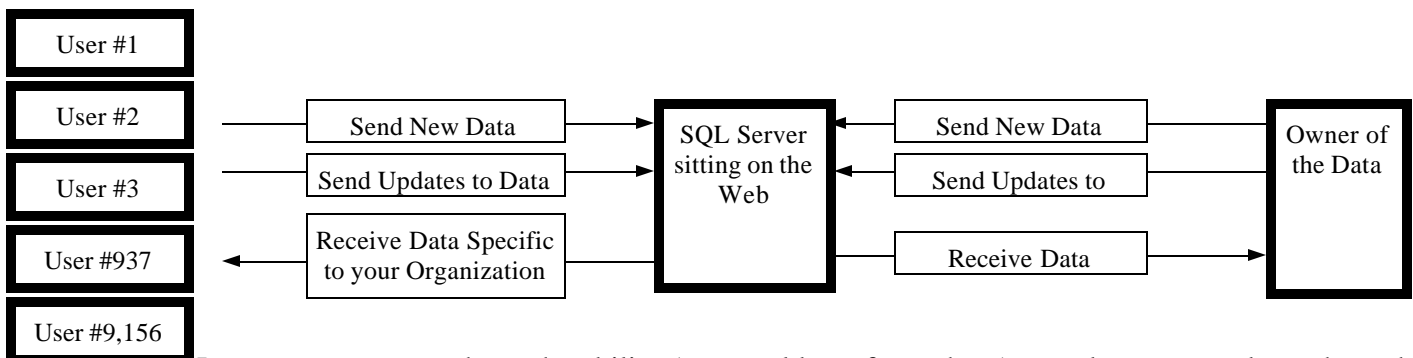
The above form is just using a couple of tools to go get this information from a database on the web.
<http://www.dev1.eraserver.net/WebServices/zipcoderesolver/zipcoderesolver.asmx/CorrectedAddressXml?accessCode=9999&address=13839+Murlen+suite+M&city=Olathe&state=ks>

Example 2: Do a Google Search

In this you must register with Google and get a Key --> <http://www.google.com/apis/>
The concept is to have a key so the owner has some control. The user uses the key to access the data and they can use it any way they want in their application.
(Note: Google limits 1 key to 1,000 queries per day.)



2) Here is a bit more complicated form of transmitting data:



Example 3: Have your customers have the ability (we would not force them) to order your products through XML over the web.

Your customers could set up their existing purchase order system to get your product/price list. Once they had the list in their system they could setup a purchase order for your products, send it to you with a push of a button, and get the status of their order with a push of another button. They would do this instead of ordering over the web because this order is their data - they don't care if you have the information, but for sure they want the information on their network, not something they need to go to a website to lookup.

- * Order Received
- * Order and Pricing Approved
- * Delivery Scheduled
- * Material in Transit
- * Material Delivered

3) Some of the Unique Things About the ADO and Stored Procedures we will Look at

- A) Each company, including the owner of the data, has a key and a company Id. We send the Key (e.g. ABCDEFGHIJ1234567890) to the web along with a description of the process (e.g. “Send Returns to the Web”) to get back a **Process Id**. This process Id is good for 15 minutes. The Key must match a company. In later processes we will confirm the company we are dealing with matches the company the process was started for.
- B) We send a XML string (recordset) to the web.
- C) We have a “cycle number” in a table (e.g. 20) that indicates how many records we send to the web at a time. (Note if multiple tables = 1 record - e.g. Invoice and line items, we send a shaped recordset) We set up a process to send 20 records at a time to .Net and SQL Server. If everything works, .Net sends back a recordset with 1 field with the value of “Success”. If success, we can mark those 20 records as being sent.
- D) We do all processing in SQL server in a transaction, so if we send 20 invoices with 50 line items and 1 line item cannot be inserted, the transaction is rolled back. We would then send an error message indicating what function, what stored procedure if applicable, what position, and the exact SQL Server or .Net error message. If the client application receives an error, we send the 20 records to a subroutine that takes all the data (as a string that is an XML document) and runs it through a process that gets rid of all ASCII characters below 9 and above 122. We then send the records 1 at a time.

If we get success on all 20, we can mark them sent, if an error, we mark the record(s) as no being sent, and notify the user what record(s) were not sent if any.

If 20% of the records are sent through 1 record at a time, we lower (if >1) the “cycle number” in a table by 1. So if 20% of the records or more were sent 1 record at a time, next time and all times after we will send 19 records per cycle instead of 20.

- E) We use Soap to send the data to the .Net web service.

```
Dim SoapClient_1 As MSSOAPLib30.SoapClient30
Dim ElemList As MSXML2.IXMLDOMNodeList

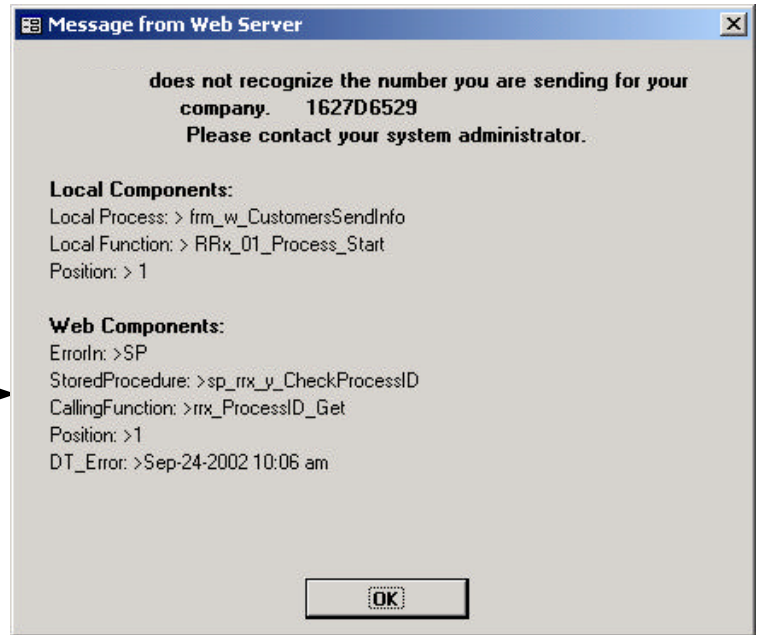
'Set up Soap and WSDL File
Set SoapClient_1 = New MSSOAPLib30.SoapClient30
SoapClient_1.mssoapinit strWSDLFile

'Go to the Web Service
Set ElemList = SoapClient_1.rxx_ProcessID_Get(strGwidLike, strProcessName)
```

.Net Web Services:

F) We have very small web methods (See next two pages). We do most of the processing on SQL Server.

G) There are times when the actual error is in SQL Stored Procedures, however because of how SQL Server works, it never runs the stored procedure and the error appears in dot net. We send back information to the client application that will allow us to see where the error is. _____→



H) Time out and Speed with Soap

I tested downloading records from the web with a slow modem, with the following results

- * The Connection was 21.6 kb
- * I was able to receive from the web 200 records = 110 kb in 20 seconds
- * I was able to receive from the web 2000 records = 1,091 kb in 3 minutes and 10 seconds
- * I thought I would have problems with timing out - I did not.

J) I did not need to set the time out property

a) You can set the timeout property in soap, (the default is 30 seconds) however with the above test results, I did not. The way you would set it is below. (each number = a millisecond - 30,000 = 3 seconds)

```
SoapClient_1.ConnectorProperty("Timeout") = 30000 '
```

```

<WebMethod ()> _
    Public Function Y_ReturnsToWeb(ByVal strXMLDoc As String, ByVal strProcessIdt As String, _
        ByVal intStep As Int32) As DataSet
        Purpose: Put records in or update tbl_FromYellowReturns and tbl_FromYellowReturnsLineItem
        Add ids and Processes or update processes in tbl_Returns_Id and tbl_ReturnsLineItem_Id

        Required_Elements: sp_Y_ReturnsToWeb tbl_FromYellowReturns tbl_FromYellowReturnsLineItem tbl_Processes
        tbl_Returns_Id and tbl_ReturnsLineItem_Id
        Example: Y_ReturnsToWeb(strXMLDoc, strProcessIdt, 5, "10")
        In_Parameter: strXMLDoc = Doc with a new or update to table
        strProcessIdt = The current process we are using (must be less than 3 hours Old)
        intStep = The Step we put in tbl_Processes once completed

        Returns: dataset Name ---> Process
        Return Value Dataset ---> Success - Only Possible value = True
        Error Value Dataset ---> ErrorIn - StoredProcedure - CallingFunction - Position - ClientErrMsg
        SP or Function Name of SP Name of Function Position Error msg for client
        in SP or Function

    Dim strCallingFunction As String = "Y_ReturnsToWeb"
    Dim strRetData As String = "ReturnValue" 'Table in dataset

    Dim dsl As New DataSet()
    Dim dataAdp1 As New SqlDataAdapter()
    Dim cmd1 As New SqlCommand()
    Dim intReturnValue As Int32

    'Connection
    Dim cmn As New SqlConnection(ConfigurationSettings.AppSettings("CmnString"))

    Try
        cmn.Open()

        With cmd1
            .Connection = cmn
            .CommandType = CommandType.StoredProcedure
            .CommandText = "sp_Y_ReturnsToWeb"

            .Parameters.Add("@txtXMLDoc", SqlDbType.Text).Value = strXMLDoc
            .Parameters.Add("@vcProcess_Idt", SqlDbType.VarChar, 10).Value = strProcessIdt
            .Parameters.Add("@intProcessStepAtCompletion", SqlDbType.Int).Value = intStep
            .Parameters.Add("@vcCallingFunction", SqlDbType.VarChar, 50).Value = strCallingFunction

            .Parameters.Add("RETURN_VALUE", SqlDbType.Int).Direction = ParameterDirection.ReturnValue

        End With
    End Try

```

```

' Set the Adapter to the recordset
dataAdpt1.SelectCommand = cmd1

' Fill the DataSet
dataAdpt1.Fill(ds1, strRetData)

' Check Return value 0=Success 1=Failure - Note strRetData is set to --> ErrorValue
intReturnValue = cmd1.Parameters("RETURN_VALUE").Value
If intReturnValue <> 0 Then
    ds1.Tables(0).TableName = "ErrorValue"
End If

' Close the connection
cmn.Close()
Catch
' Put error message in table for dataset
ds1.Tables.Add(SF_ErrorTable(strCallingFunction, Err.Description))
End Try

Return ds1
End Function

```

Alter Procedure sp_YS_HazMatSend

Typical SQL Stored Procedure for my .Net Web Services

```
@txtXMLDoc text,  
@vcProcess_Idt varchar(10),  
@IntProcessStepAtCompletion as int,  
@vcCallingFunction varchar(50)
```

AS

set nocount on

/* Documentation Below

--> Purpose: Put records in tbl_HazMat_Descriptions <- Purpose

--> Required_Elements: tbl_HazMat_Descriptions, sp_rtx_ProcessCheck <- Required_Elements

--> Example: sp_YS_tbl_HazMat_strXMLDoc, '1234567', 33, 'Function test' <- Example

--> Out_Parameter: <- Out_Parameter

--> In_Parameter: @txtXMLDoc = XML Doc with a record for tbl_Customers

@vcProcess_Idt = The current Process Id

@IntProcessStepAtCompletion as int = The step# we put in tbl_Process at completion of this SO

@vcCallingFunction = Function that called this SP <- In_Parameter

-->Returns: 0=Success 1=Failure

Documentation Above */

declare @IntXMLDocNbr int,

@vcCompany_Idt as varchar(9),

@return int,

@vcCompany_Idt_Proc as varchar(9)

-- Below for error message -----

declare @IntErrorPosition int,

@vcProcedureName varchar(50),

@vcUserErrMsg varchar(8000),

@vcErrorIn varchar(2),

@vcProgrammerMsg varchar(2000)

Set @IntErrorPosition=0

set @vcProcedureName = 'sp_YS_HazMat_tbl'

set @vcErrorIn = 'SP'

```

set @vcProgrammerMsg = 'vcProcess_Idt=' + @vcProcess_Idt + ' |X|@IntProcessStepAtCompletion=' +
                        convert(varchar, @IntProcessStepAtCompletion) +
                        '|X|@vcCallingFunction=' + @vcCallingFunction +
                        '|X|@txtXMLDoc=' + convert(varchar(2000), @txtXMLDoc)

set @vcProgrammerMsg = Left(@vcProgrammerMsg, 2000)

--Get the @vcCompany_Idt based on the passed in ProcessIdt
--(Note a -1 will come back if it's been 3 hours or longer or the Step > 98
exec @return = sp_rx_ProcessCheck @vcProcess_Idt
-- If not matching - stop now - fail
if @return < > 0
    Begin
        set @IntErrorPosition = 1
        set @vcUserErrMsg = 'Could not verify company based on Process Id. -->' + @vcProcess_Idt

        ----- Put Error in Table -----
        INSERT INTO [dbo].[tbl_ProcessProblems] ([ProcessIdt], [ErrorIn], [StoredProcedure],
        [CallingFunction], [Position],
        [ClientErrMsg], [dt_Error], [ProgrammerMsg])
        Select @vcProcess_Idt, @vcErrorIn, @vcProcedureName,
        @vcCallingFunction, @IntErrorPosition,
        @vcUserErrMsg, getdate(), @vcProgrammerMsg

        ----- Send Back Error to Client -----

        Select @vcErrorIn, ErrorIn, @vcProcedureName as StoredProcedure, @vcCallingFunction as CallingFunction,
        @IntErrorPosition as Position, @vcUserErrMsg as ClientErrMsg, getdate() as DT_Error

        return 1
    end

--Create an internal representation of the XML document.
exec sp_xml_preparedocument @intXMLDocNbr OUTPUT, @txtXMLDoc

```


Begin Transaction

```
INSERT INTO tbl_HazMat_Descriptions (HazMat_Id, UNNA, UNNA_Idt, Full_Desc, Poison_Ask_YN, Seq, Pack_Group,
Prime_Class, Second_Class, Tert_Class,
Inhale_YN, Zone, NOS_Ind_YN, Prop_Ship, TimeStamp, Yellow_Haul_YN, Last_Update_Nbr)
select id, UNNA, UNNA_Idt, Full_Desc, Poison_Ask_YN, Seq, Pack_Group, Prime_Class, Second_Class,
Inhale_YN, Zone, NOS_Ind_YN, Prop_Ship, TimeStamp, Yellow_Haul_YN, Last_Update_Nbr
FROM OPENXML (@intXMLDocNbr, '/HazMat/Details', 2)
WITH([id] int,
[UNNA] varchar(50),
[UNNA_Idt] varchar(50),
[Full_Desc] varchar(50),
[Poison_Ask_YN] bit,
[Seq] float],
[Pack_Group] varchar(50),
[Prime_Class] varchar(50),
[Second_Class] varchar(50),
[Tert_Class] varchar(50),
[Inhale_YN] bit,
[Zone] varchar(50),
[NOS_Ind_YN] bit,
[Prop_Ship] varchar(50),
[TimeStamp] datetime,
[Yellow_Haul_YN] bit,
[Last_Update_Nbr] int)

If (@@Error <> 0) or @@ROWCOUNT = 0
Begin
Rollback Transaction
Exec sp_xml_removedocument @intXMLDocNbr

set @IntErrorPosition=2
set @vUserErrMsg = 'Could not put the information in tbl_HazMat_Descriptions'

----- Put Error in Table -----
INSERT INTO [dbo].[tbl_ProcessProblems]([ProcessId], [ErrorIn], [StoredProcedure],
[CallingFunction], [Position],
[ClientErrMsg], [dt_Error],[ProgrammerMsg])
Select @vcProcess_Idt, @vcErrorIn, @vcProcedureName,
@vcCallingFunction, @IntErrorPosition,
@vUserErrMsg, getdate(), @vcProgrammerMsg
```

```

----- Send Back Error to Client -----
Select @vcErrorIn_ErrorIn, @vcProcedureName as StoredProcedure, @vcCallingFunction as CallingFunction,
@IntErrorPosition as Position, @vcUserErrMsg as ClientErrMsg, getdate() as DT_Error
return 1
End
else
Commit Transaction

-- Remove XML document from memory
Exec sp_xml_removedocument @IntXMLDocNbr

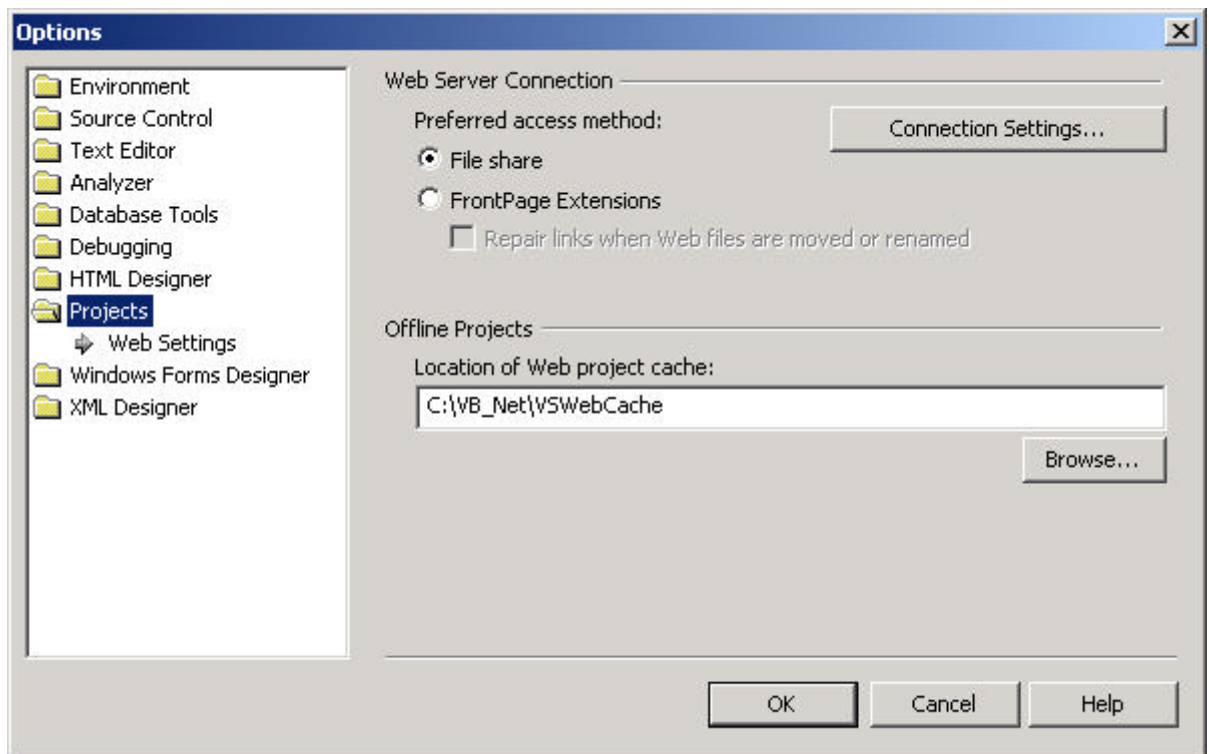
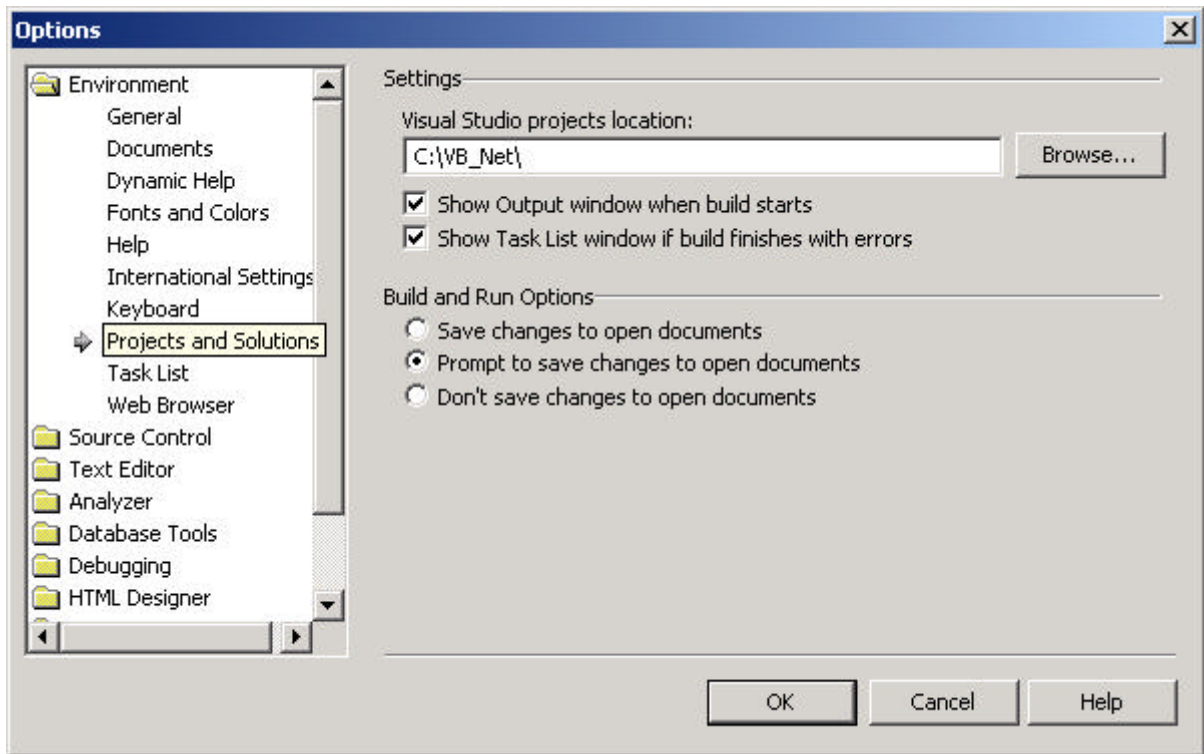
--Update Process
exec @return = sp_rtx_ProcessUpdate @vcProcess_Idt, @IntProcessStepAtCompletion

Select 'True' as Success

return 0 ---<-- 0=Success 1=Failure

```

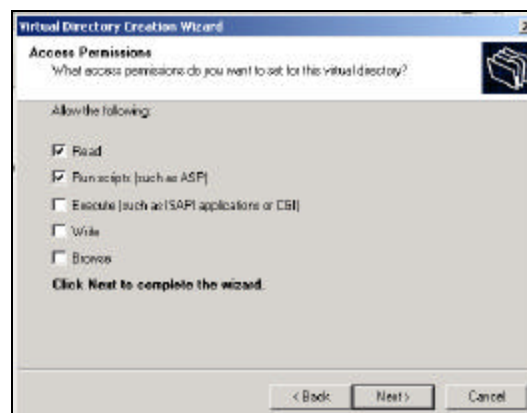
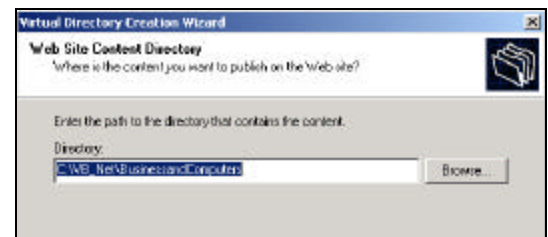
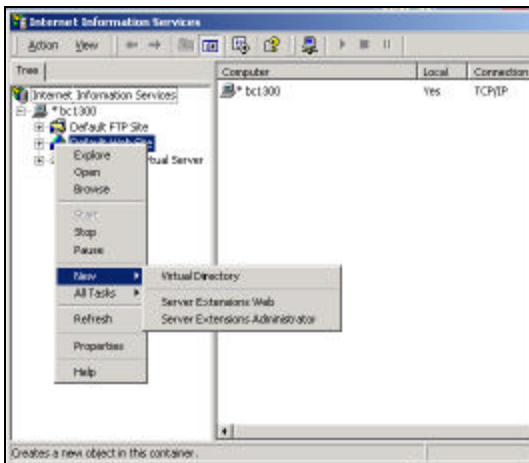
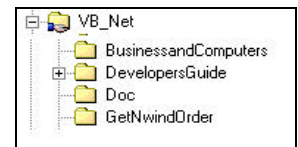
You might want to set the properties to a different sub directory
Tools/Option



Creating XML Web Service on Your Machine

1) Set up your Virtual Directory

- Create a directory on your hard drive that your web service will sit in. (Note this is going to match up with your website. So you might create a directory with the same name of you web site. E.g. if you have a website BusinessandComputers.com you might want to create a directory BusinessandComputers. Your XML Web Services will sit in a directories below this directory.
- Bring up Send IIS (Internet Information Services) and right click on default Web Site, New, Virtual Directory.
- Follow the directions.



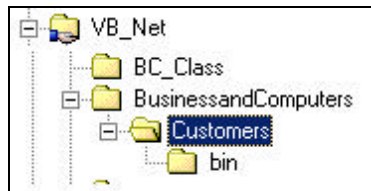
2) Create a new Web Service

- From VS.Net (Visual Studio .Net) interface click New Project
- Use `http://localhost/BusinessandComputers/Customers`

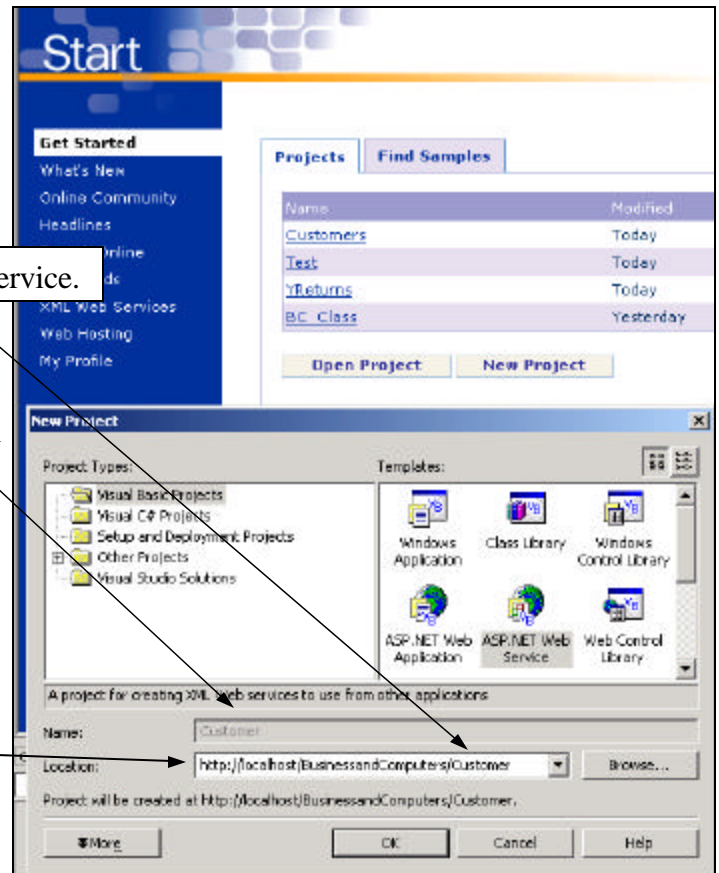
IIS directory from

The name of the XML Web Service.

- VS.Net will create the Virtual directory and put the files in the directory. It might also create a directory under your default directory with a .sln and .suo. If it does just the files to the directory you are using.

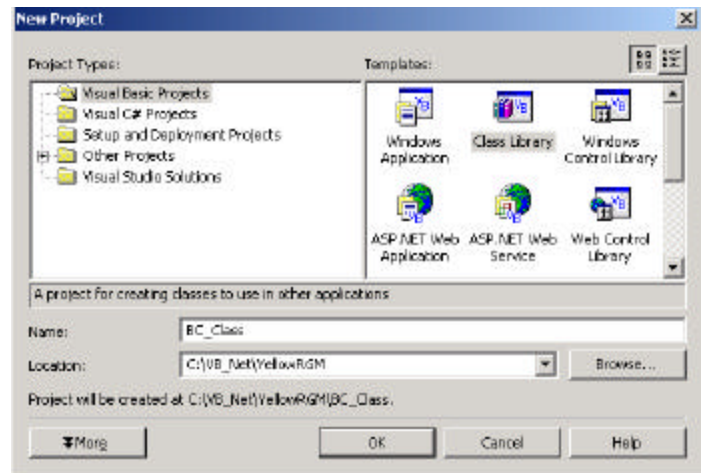


Notice forward slash



3) Having functions that can be used by many web services. (Creating and using a .dll)

- From VS.Net (Visual Studio .Net) interface click New Project
- Pick Project Types = "Visual Basic Projects" and Templates = "Class Library"
- Put the name of the class in the name field.
- Pick a Location (Note VS.Net will create a directory with the name of the class under the location you pick)
- In the Class1.vb put your functions (You might want to rename Class1.vb to something more friendly.)

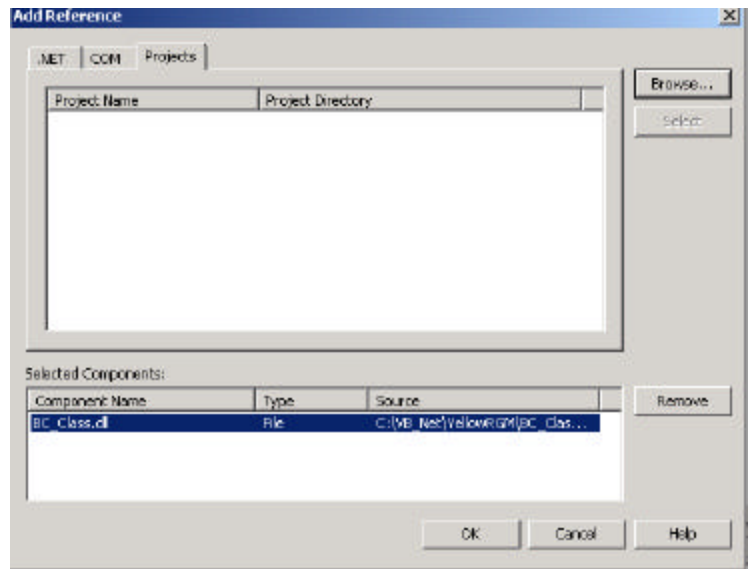


```
Public Class BC_Standard
    Public Function HundredDollarBills(ByVal Number As Double) As Long
        Dim lngBills As Long
        lngBills = Int(Number / 100)
        Return lngBills
    End Function

    Public Function ThousandDollarBills(ByVal Number As Double) As Long
        Dim lngBills As Long
        lngBills = Int(Number / 1000)
        Return lngBills
    End Function
End Class
```

- f) From the your XML web service right click on references and pick "Add Reference".
- g) Go to the Projects Tab, click Browse button. Go find the .dll and then click OK.
- h) It is now loaded with your project and you can call it in your web methods.
- i) Each time you compile your XML web service VS.Net will bring the dll down to the bin folder for your web service.

Note: If you change the functions in your dll you will need to recompile each of your web services and re-deploy them.



```
<WebMethod()> Public Function TestHDollars(ByVal dol As Double) As String
    Dim bc As New BC_Class.BC_Standard()
    Dim HDollars As Double
    HDollars = bc.HundredDollarBills(dol)
    Return HDollars
End Function

<WebMethod()> Public Function TestTDollars(ByVal dol As Double) As String
    Dim bc As New BC_Class.BC_Standard()
    Dim HDollars As Double
    HDollars = bc.ThousandDollarBills(dol)
    Return HDollars
End Function
```

H) We put the connection string in the Web.config file, and refer to it in the web method.

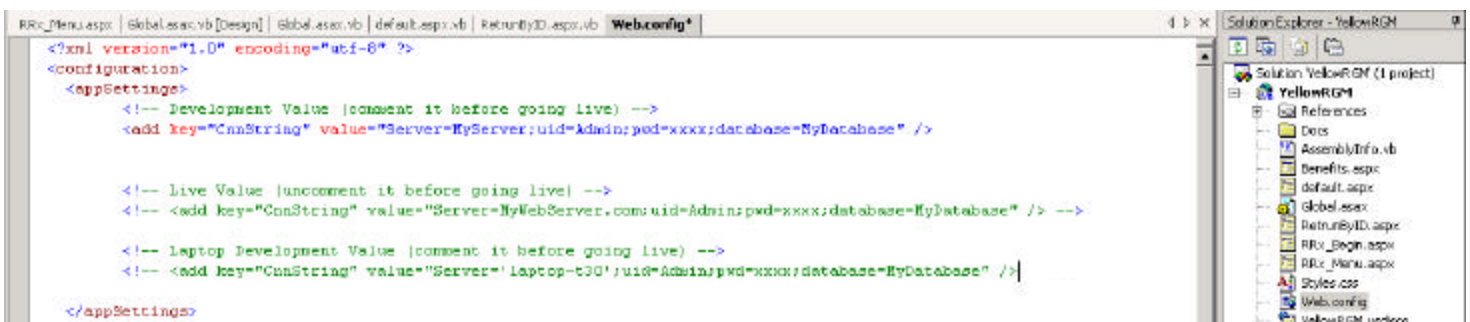
Web.Config

```
<appSettings>
  <!-- Development Value (comment it before going live) -->
  <add key="CnnString" value="Server=localhost;uid=SomeUser;pwd=SomePassWord;database=MyDatabase" />

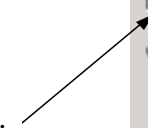
  <!-- Live Value (uncomment it before going live) -->
  <!-- <add key="CnnString" value="Server=SQL2.MyWeb.com;uid=SomeUser;pwd=SomePassWord;;database=MyDatabase" /
-->
</appSettings>
```

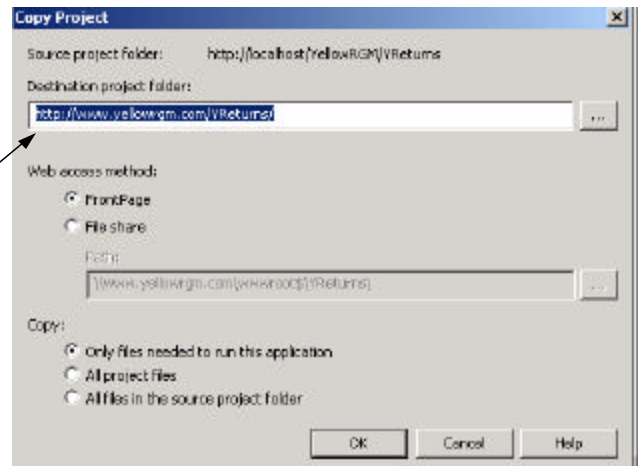
Web Method

```
'Connection
    Dim cnn As New SqlConnection(ConfigurationSettings.AppSettings("CnnString"))
```

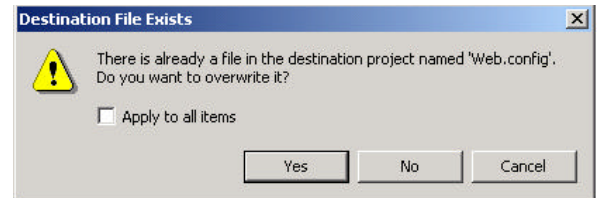


Copying an XML Web Service to the Internet.

- 1) Go to the website and create a folder for the XML Web Service.
- 2) Bring up the project in VS.Net
 - a) Go to Projects/Copy Project.
 - b) The form will appear.
 - c) Put your website and the appropriate folder. 
 - d) It will over write the files that are currently on the web.
 - e) Test
 - f) Your done.



Do not overwrite the Web.Config once you got it right —>



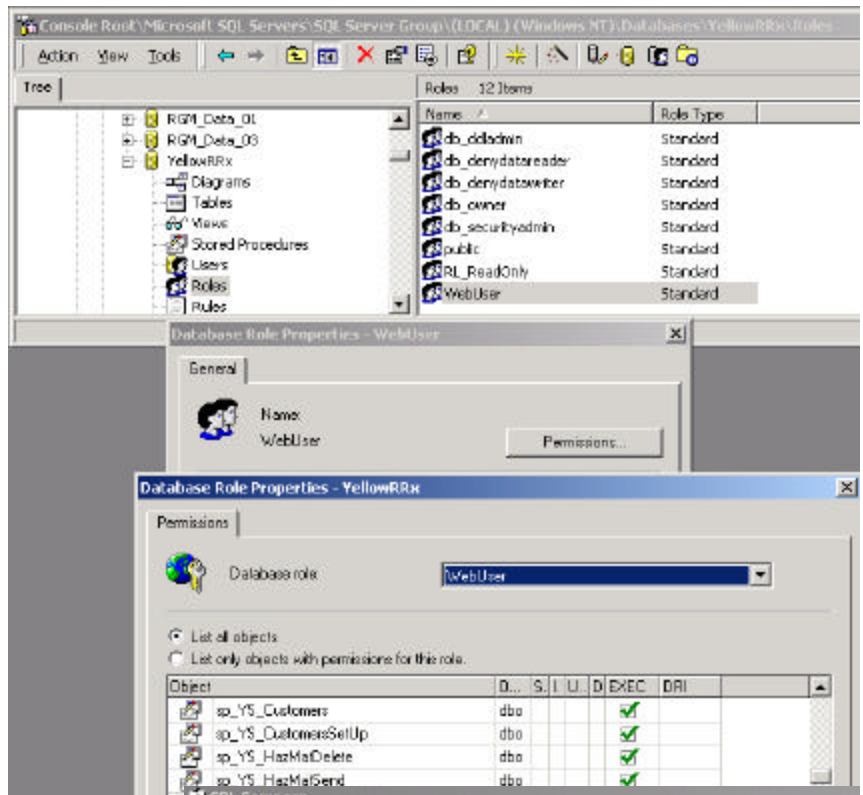
Copying an SQL Server Objects to the Internet SQL Server.

1) Make sure all functions begin with dbo. Like:

ALTER Procedure dbo.sp_Cust_CustFieldsToWeb

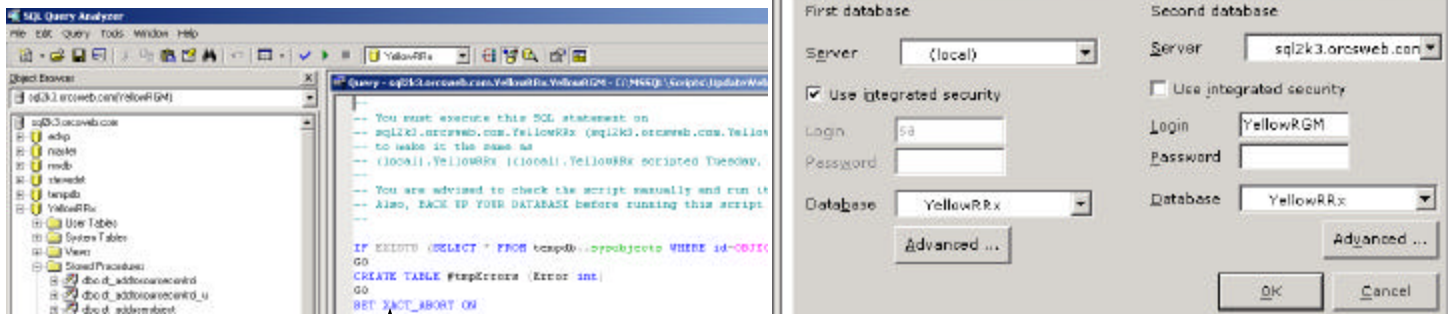
2) Deal with Role

- a) Go to Enterprise Mgr/Roles/
WebUser and make sure all Stored
Procedures the user will use have
Execute permission.



3) Create Scripts

- a) Run SQL Compare against your
local database and the web database.
- b) Create Individual scripts for each object
changed. Save on C:\MSSQL\Scripts
\UpdateWebServer as the object name (eg → sp_GetId).



4) Run the scripts (one at a time) in the query Analyzer

5) Check the web and confirm the objects exist.

6) Check the WebUser Role on the web (See #2 above).

Site	Setup Including SQL Server	Microsoft .Net Available	Monthly Fee	SQL Server	Total Monthly	Data Transfer (GB)	Disk Storage (MB)	SQL Server (MB)	Windows 2000	30 Day Money Back Guarantee
http://www.innerhost.com	\$99.95	Yes	\$49.95	\$50.00	\$99.95	10	200	50	Yes	Yes
				ASP .Net(Microsoft) - 4 Guy's <--- Recommended By:						
http://www.orcsweb.com/shared.asp	\$0.00	Yes	\$65.00	Included	\$65.00	3	200	25	Yes	???????
				ASP .Net(Microsoft) - 4 Guy's <--- Recommended By:						
http://www.flarehosting.com/services/hosting_compare.asp	\$0.00	Yes	\$24.95	Included	\$24.95	10	200	100	Yes	Yes
				ASP .Net(Microsoft) - developersdex <--- Recommended By:						
http://www.alentus.com/hosting/compare.asp	\$30.00	Yes	\$49.95	\$0.00	\$49.95	10	1,000	50	Yes	Yes
				ASP .Net(Microsoft) - developersdex <--- Recommended By:						
http://www.brinkster.com/	\$23.90	Yes	\$12.95	\$32.85	\$45.80			SQL Server 7.0 Only		