The West-Wind Web Connection wwBusiness Class

by

Kevin J. Cully
Cully Technologies, LLC

Presented to the Atlanta FoxPro Users Group
March 21st, 2002

TOC

Introduction

Cully Technologies, LLC has been in existence since 2000, but Kevin Cully became a full time consultant at the beginning of 2002. Since that time, he has concentrated on building web sites, web applications, and fat client applications in a variety of commercial industries.

Kevin has been programming in FoxPro since FoxPro 2.0 for DOS in 1992 and has been programming using the Web Connection framework since 1997.

He utilizes Visual Interdev, Photoshop, Flash, ASP, VBScript, JavaScript, Access, Visual FoxPro, SQL Server, MySQL, and Web Connection.

This document is an expansion of the outline used in the March 21st presentation.

Assumptions

This presentation assumes that you have a full understanding of VFP and object oriented programming. It is also assumed that you have a foundational understanding of HTML, HTML form variables and a cursory knowledge of Web Connection.

wwBusiness Class Defined

From the West-Wind Web Connection Documentation:

> The wwBusiness object provides the core business object class. This class is a single level business object class that implements both the business and data tier in a single object. The base object provides core data access functionality (Load, Save, Query, New etc.) through its base class methods. Query based data access is also supported through native Fox syntax or using the class methods which can route to the appropriate backend. Fox or SQL Server are natively supported. SQL Server is supported with SQL Passthrough commands. The object also supports remote data access over the Web against a SQL Server backend (with some limitations a Fox backend can be used as well).
>
> Beause the framework consists only of a minimal set of methods on this object it's easy to get started with it. Because of the single level implementation the framework is also very efficient. In addition a Wizard is provided to help you create business objects by mapping to tables.

Example: Simple ERP

Starting on the first of the year 2002, I became a contractor full time. As such I had certain business needs to keep track of different data elements: companies, contacts, addresses, projects, contracts, documents, activities (meetings, to-do items, milestones, tasks, billed time, no charge items, and expenses), invoices, accounts receivable, accounts payable. That is a lot of information to track. In addition to tracking information I also need some sort of reporting / way of looking at my information that keeps me efficient and on track.

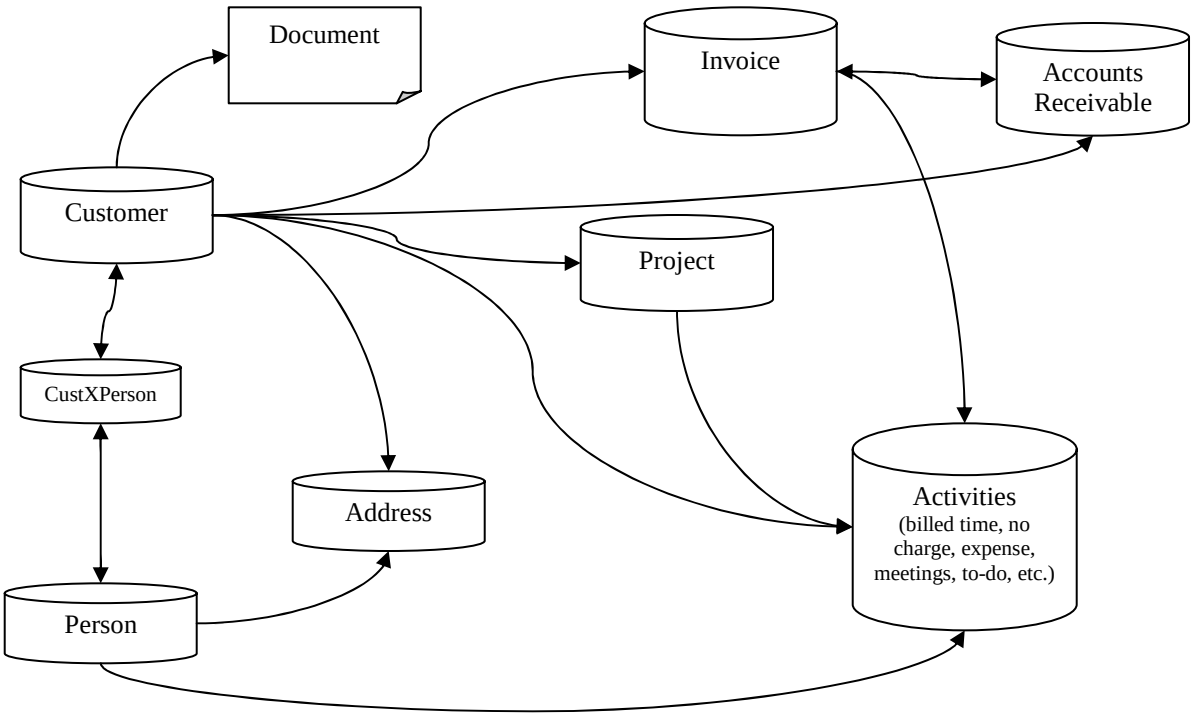
To start any process, I began by asking "what do I have already." I have Outlook. It has email, contact management, calendaring, reminders, tasks and notes. I don't really like the way that the contacts are structured in Contact where the company and contact are merged. It is also really easy to get multiple records per contact. If ever Cully Technologies, LLC gets big enough for multiple employees, it's a pain to share contact information even with an Exchange server. Don't get me started about Exchange server. Been there, done that, won't do it again. (Unless the money is really, really good.)

The accounting aspect makes me the most nervous because I'm not an accountant. I polled some of my peers and found that most people are using Quickbooks. Quickbooks basic costs $160 from Page Computers. Importing and exporting of the data from Outlook into this system would be a real hassle. Of course accounting systems are supposed to be a "solved problem" right? It'd be crazy to write another accounting package.

One of my past companies used ACT! which had the same problems with contact management that Outlook has; contacts and companies are stored in the same record. I'm still looking for my project management solution, document management solution.

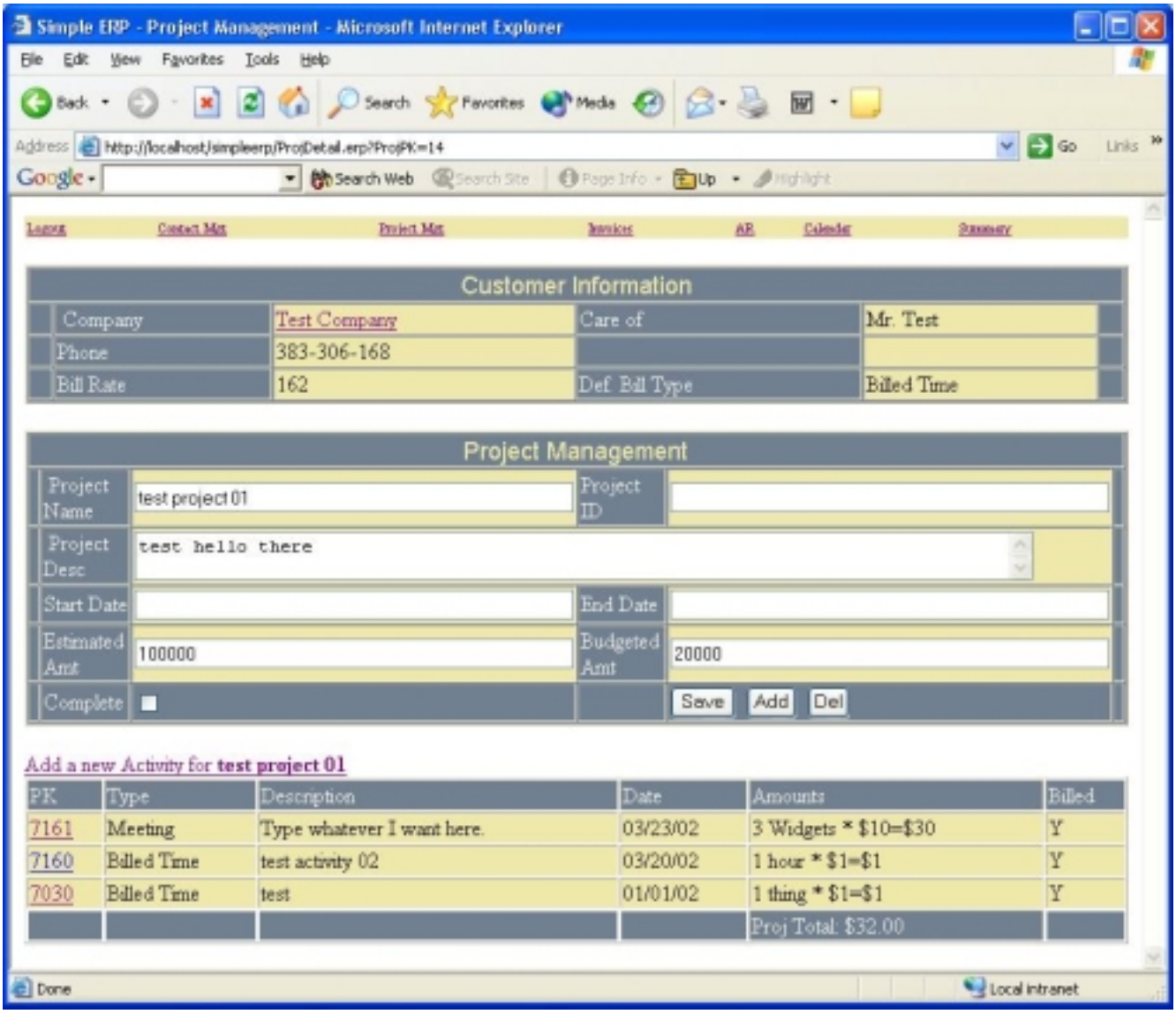
Wouldn't it be great to have one of those million dollar solutions where all of the information is integrated together and flows back and forth seamlessly? I've had a dream about one of these systems for a while now. I sat down and re-drew a diagram that I had drawn before but no one ever bought into. I wasn't sure that this would be a wise investment in my time but I was going to do it anyway.

SimpleERP – Relational Diagram



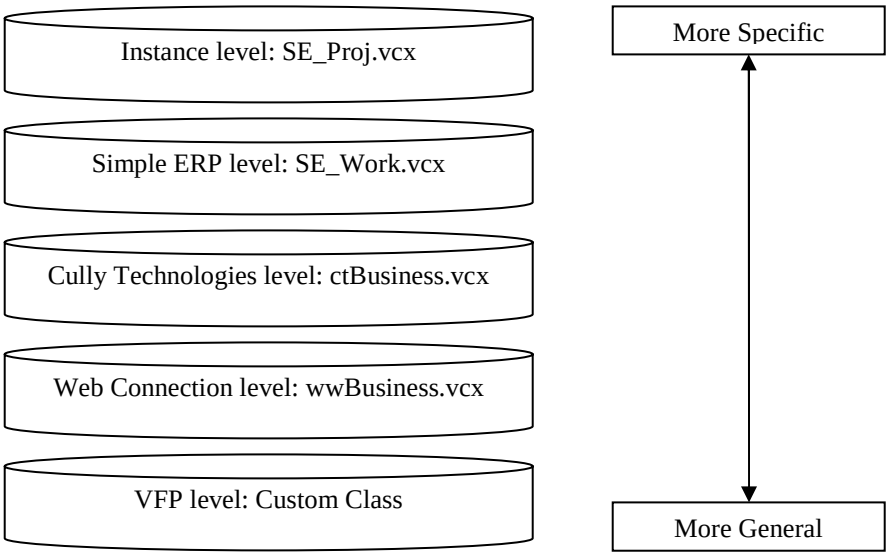
This diagram is of the current SimpleERP data model. There are some adjustments that I'd like to make but this model allows me to meet my current data needs necessary to run Cully Technologies.

You may notice that the Activies.dbf is key to linking many of the pieces of information together. Almost every detail action is stored into the Activities table. This allows me at a glance to view almost everything that is occurring with a customer. One stop shopping.

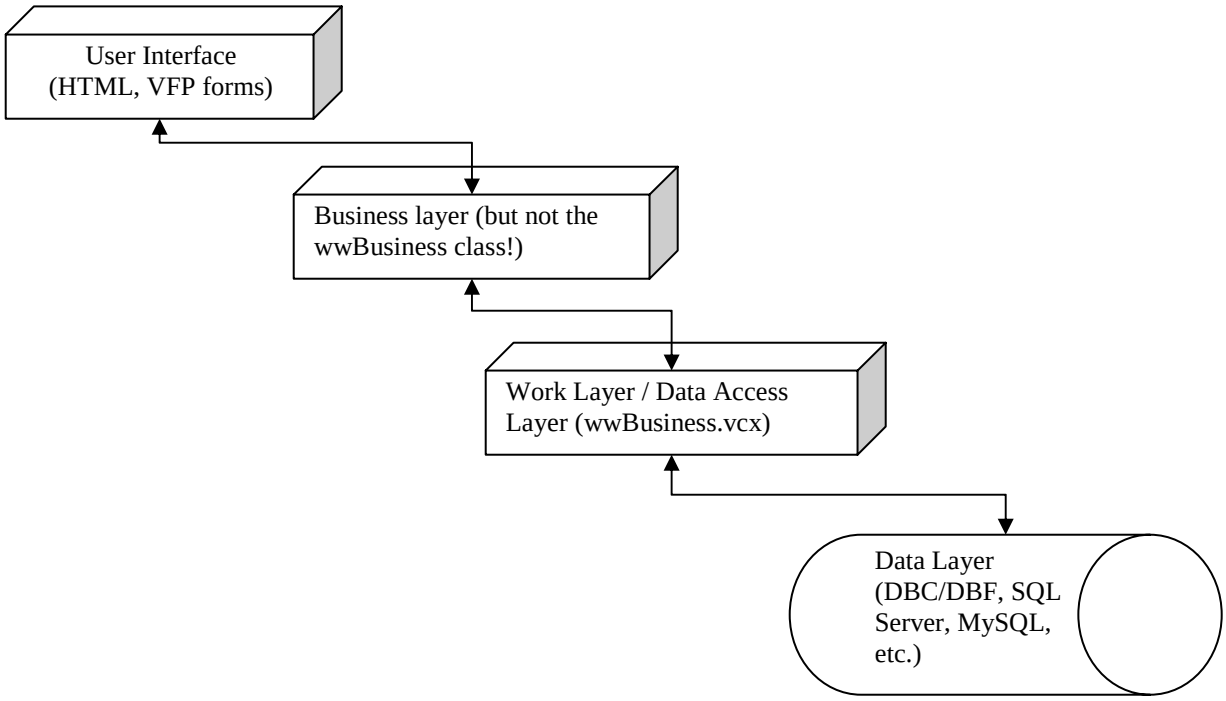

Simple ERP - Project Management - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/simpleerp/ProjDetail.erp?ProjPK=14
Go Links
Google Search Web Search Site Page Info Up Highlight
Logout Contact Mgt Project Mgt Invoices AR Calendar Summary
Customer Information
Company Test Company Care of Mr. Test
Phone 383-306-168
Bill Rate 162 Def. Bill Type Billed Time
Project Management
Project Name test project 01 Project ID
Project Desc test hello there
Start Date End Date
Estimated Amt 100000 Budgeted Amt 20000
Complete Save Add Del
Add a new Activity for test project 01
PK Type Description Date Amounts Billed
7161 Meeting Type whatever I want here. 03/23/02 3 Widgets * $10=$30 Y
7160 Billed Time test activity 02 03/20/02 1 hour * $1=$1 Y
7030 Billed Time test 01/01/02 1 thing * $1=$1 Y
Proj Total: $32.00
Done
Local intranet

wwBusiness Class – Subclassing for your enterprise use

To best utilize the business class, it is a good idea to subclass the originating class. Subclassing into a enterprise wide class, then an application level class, and then into the instance based classes allows for taking advantage of the best OOP hierarchy.



Also recommended is an evolution of the 3 tier and n-tier approach to application development: 4 tiers. Although the wwBusiness layer is designed to combine the business and data access (work) tiers, we recommend that you utilize it as a data access layer (work layer) only for maximum flexibility.



Instantiating the wwBusiness Instances in a Web Application

For efficiency, we instantiate the business classes on the init of the server object. This way the business classes can hold open tables if we're using VFP tables or hold open the connections into a SQL implementation of the business class.

In the SETSERVERPROPERTIES method of the server object is the following code:

```
*** Add any SET CLASSLIB or SET PROCEDURE code here
SET PROCEDURE TO CTUtils ADDITIVE
SET CLASSLIB TO SE_Work ADDITIVE

THIS.oSE_activity       = CREATEOBJECT('SE_activity')
THIS.oSE_addr           = CREATEOBJECT('SE_addr')
THIS.oSE_ar             = CREATEOBJECT('SE_ar')
THIS.oSE_cfg            = CREATEOBJECT('SE_cfg')
THIS.oSE_Cust           = CREATEOBJECT('SE_Cust')
THIS.oSE_custXperson    = CREATEOBJECT('SE_custXperson')
THIS.oSE_Person         = CREATEOBJECT('SE_Person')
THIS.oSE_proj           = CREATEOBJECT('SE_proj')
THIS.oSE_invoice        = CREATEOBJECT('SE_invoice')

THIS.oSE_Activity.Open()
THIS.oSE_Addr.Open()
THIS.oSE_Ar.Open()
THIS.oSE_Cust.Open()
THIS.oSE_Invoice.Open()
THIS.oSE_Proj.Open()

THIS.cMainMenu = FILE2VAR([HTML\MainMenu.htm])

*** And now create a browser that navigates to this application.
      lobrowse = create('InternetExplorer.Application')
      lobrowse.navigate( 'http://localhost/simpleerp')
      lobrowse.visible = .t.
```

Key Properties and Methods

We'll be using several of the methods and properties on the wwBusiness class. Taken from the WW help file, they're detailed here:

Name	Type	Description
createnewid	Method	Creates a new Primary Key ID number for the table using the ID table. **o.createnewid()**
delete	Method	Deletes the specified record from the underlying table. If no PK is passed the current PK of the loaded object is used. **o.delete(lnPk)**
open	Method	Opens the data source or remote connection. For ODBC commands the oSQL member holds the SQL connection. **o.open(lcFile, lcAlias, llForceReconnect)**
query	Method	High level query method that is aware of the data mode in operation. For porting data between different data layers use this method for SQL commands that can run against any backend. **o.query(lcSelect, lcCursor, lnResultmode)**
save	Method	Saves the current data member in oData to the underlying table. **o.save()**
seterror	Method	Sets the error flag and error message properties. **o.seterror(lcErrorMsg, lnError)**
calias	Property	The Alias of the master file.
cdatapath	Property	Location of the data
cerrormsg	Property	Any error messages that occurred.
lerror	Property	Error flag for the previous operation. Note this property must be managed by the developer for custom methods.
ndatamode	Property	Data access mode for the business object. 0 - Fox Data native, 2 - ODBC (SQL Server)
nupdatemode	Property	Determines whether the data member is in edit or new mode. 1-edit mode, 2 - New
odata	Property	Data member that receives all of the field names as property values.

Examples in retrieving data for display

In this example, we're looking at the function that displays the calendar. Basically this is just a list of activities joined with projects (if applicable) and customers (if applicable).

```
FUNCTION Calendar
      LOCAL lcCommand AS Character, lnRecCount AS Integer
      PRIVATE lcMainMenu AS Character, lcTemplate AS Character,
            lcExpandedHTML AS Character, oPerson AS Object

      lcMode      = UPPER(REQUEST.QueryString([cMode]))
      lcButton    = UPPER(REQUEST.FORM([btnSubmit]))
      lcDStart    = UPPER(REQUEST.FORM([dstart]))
      lcDaysOut   = UPPER(REQUEST.FORM([cDaysOut]))
      IF EMPTY(lcDStart)
            ldStart = DATE()
      ELSE
            ldStart = CTOD(lcDStart)
      ENDIF
      IF EMPTY(lcDaysOut)
            lnDaysOut = 7
      ELSE
            lnDaysOut = VAL(lcDaysOut)
      ENDIF

      lnRecCount = goWCServer.oSE_Activity.QueryByDateRange('c_Activity1',
ldStart, lnDaysOut)
      lcMainMenu = goWCServer.cMainMenu
      lcTemplate = File2Var("HTML\Calendar.HTM")
      lcExpandedhtml = ExpandHTML(lcTemplate)
      Response.write(lcExpandedHTML)
      =CloseDBF('c_Activity1')
ENDFUNC
```

A look within the QueryByDateRange method in the SE_Activity class shows some simple text manipulation and then a call to the Query method that is inherited from the wwBusiness class. Notice that in the lcCommand string that I'm assembling, I'm assuming that we're running against VFP data. With just a little effort, this could be re-written in such a way that either (1) the command would work both in VFP and SQL Server or (2) that I could check the nDataMode and assemble a string with a different syntax specific to the back end. Either way with out 4 tier approach, the business layer and UI layer are protected from out data layer changes.

```
*** SE_Activity.QueryByDateRange
LPARAMETERS tcTargetCursor AS Character, tdStart AS Date, tnDaysOut AS Integer

      LOCAL ldLow AS Date, ldHigh AS Date
      ldLow = tdStart
      ldHigh = tdStart + tnDaysOut
      IF lnDaysOut < 0
            ldLow = tdStart + tnDaysOut
            ldHigh = tdStart
      ENDIF
      lcCommand = [SELECT Se_activity.*,] + ;
                  [ Se_cust.ccompany, Se_proj.cprojname ] + ;
                  [ FROM  se_activity LEFT OUTER JOIN se_cust ] + ;
                  [  ON  Se_activity.custpk = Se_cust.pk] + ;
                  [  LEFT OUTER JOIN se_proj] + ;
                  [  ON  Se_activity.projpk = Se_proj.pk] + ;
                  [ WHERE BETWEEN(SE_Activity.DCreate, ] + StrictDTOC(ldLow)+ ;
                    [,] + StrictDTOC(ldHigh) + [)] + ;
                        [ ORDER BY SE_Activity.DCreate DESC]
      lnRecCount = THIS.Query(lcCommand, tcTargetCursor)
RETURN lnRecCount
*** / SE_Activity.QueryByDateRange
```

Examples in retrieving web data and then either inserting or updating records

No matter what project we're looking at, new or existing, this method is called when we click on SAVE. We determine what action to take and call the various methods within the wwBusiness class layer.

```
* Function ERPProcess :: ProjDetailAction
FUNCTION ProjDetailAction
LPARAMETERS tnPK AS Integer

	LOCAL lcCommand AS Character, lnRecCount AS Integer
	PRIVATE lcMainMenu AS Character, lcTemplate AS Character, ;
		lcExpandedHTML AS Character, oProj AS Object

	IF VARTYPE(tnPK) = "N"
		lnPK = tnPK
	ELSE
		lnPK = VAL(REQUEST.QueryString([ProjPk]))
	ENDIF
	lcAction = UPPER( Request.FORM([btnSubmit]) )
	DO CASE
		CASE lcAction = "SAVE"
			WITH goWCServer.oSE_Proj
				IF lnPK < 0
					.New()
					lnCustPK = VAL(REQUEST.QueryString([CustPK]))
					.oData.CustPK = lnCustPK
				ELSE
					lnResultCount = .Load(lnPK)
				ENDIF
				.oData = Request.FormVarsToObject(.oData)
				.Save()
				THIS.ProjDetail(.oData.Pk)
			ENDWITH
		CASE lcAction = "ADD"
			THIS.ProjDetail(-1)
		CASE lcAction = "DEL"
			goWCServer.oSE_Proj.Delete(lnPK)
			THIS.ProjQuery()
		OTHERWISE
			Response.StandardPage("Unable to determine appropriate action.")
	ENDCASE
ENDFUNC
* Function ERPProcess :: ProjDetailAction
```

Lets take a closer look at two of the methods on the wwBusiness class: NEW and SAVE

```
* FUNCTION wwBusiness::New
LOCAL lcPKField, lnPK

*** Not setting lError and cErrorMsg here - let worker methods do it for us. All
code here
THIS.SetError()

** Create a new record object
IF !THIS.getblankrecord()
   RETURN .F.
ENDIF

lnPK = THIS.CreateNewId()

IF lnPK < 1
   THIS.SetError("Couldn't create ID. " + THIS.cErrorMsg)
   RETURN .F.
ENDIF

lcPKField = THIS.cPKField
THIS.oData.&lcPKField = lnPK

THIS.nUpdateMode = 2 && New Record

RETURN .T.
* FUNCTION wwBusiness::New
```

In the SAVE method, we are at the point where we need to handle some data specific issues that rely on the nDataMode property of the class. Here's scaled down snippet of code from the SAVE method:

```
LOCAL lcPKField, llRetVal, loRecord
llRetVal = .T.
THIS.SetError()

IF THIS.lValidateOnSave AND !THIS.VALIDATE()
   RETURN .F.
ENDIF
loRecord = THIS.oData

DO CASE
   CASE THIS.ndatamode  = 0
      DO CASE
         CASE THIS.nupdatemode = 2      && New
            APPEND BLANK
            GATHER NAME loRecord MEMO
            THIS.nupdatemode = 1
         CASE THIS.nupdatemode = 1      && Edit
            lcPKField = THIS.cPKField
            LOCATE FOR &lcPKField = loRecord.&lcPKField
            IF FOUND()
               GATHER NAME loRecord MEMO
            ELSE
               APPEND BLANK
               GATHER NAME loRecord MEMO
            ENDIF
      ENDCASE
   CASE THIS.ndatamode = 2 OR THIS.nDataMode = 4
…  check the actual class for the rest of the code …
```

Other useful methods on the wwBusiness class

The following is taken from the WC documentation on the methods and properties of the wwBusiness class. I've pared down the list for the presentation. Check the documentation for the full list.

Name	Type	Description
Backup	Method	Takes the full content of the table and backs it up into the specified directory. **o.Backup(lcPath,llCreatePath)**
close	Method	Closes the currently open master table or releases the SQL connection. **o.close()**
convertdata	Method	Converts data from a cursor or the current object into the requested resultmode. **o.convertdata(lnResultMode, lcDocRoot, lcTable, lcRow)**
createchildobject	Method	Creates a child object instance that inherits the properties from the current object appropriate from the current object instance. **o.createchildobject(lcClass)**
createtable	Method	Create the primary table for this object. This method should always be subclassed and implemented by the developer. **o.createtable(lcFileName)**
execute	Method	Executes a fully qualified raw SQL statement through the class. Can be used for any backend data command using either VFP syntax or SQLExec() style SQL commands. **o.execute(lcSQL)**
find	Method	Tries to find a record based on a filter string and sets the oData member. Only the first item found is returned so you probably want to reserve this for known lookups of unique items. **o.find(lcFilter)**
getproperty	Method	Retrieves a property out of the XML field. Properties can be written with SetProperty. **o.getproperty(lcProperty)**
importdata	Method	Imports data using the same **o.importdata(lnMode,lcData,lcAlias)**
loadfromxml	Method	Imports object data from an XML string or XMLDOM object. **o.loadfromxml(lvXML,lnMode)**
reindex	Method	Used to PACK and Reindex the data files. **o.reindex(llSaveIndexString)**
setproperty	Method	Sets a property value in the XML field. Note: for performance reasons no checks are performed if the XML field exists. **o.setproperty(lcProperty, lvValue)**
setsqlobject	Method	Assigns existing SQL object to the oSQL object or creates one if a connection string is passed. Handles updating data mode and other admin tasks. **o.setsqlobject(loSQL)**

sqlbuildinsertstatement	Method	Builds a SQL string to perform insert operations from the oData member to the database (used for Remote data only). **o.sqlbuildinsertstatement(loData)**
sqlbuildupdatestatement	Method	Builds a SQL UPDATE statement from the current oData member (used for remote data only). **o.sqlbuildupdatestatement(loData, lcSkipFields)**
statusmessage	Method	Displays status information. Use this method to have the business object communicate with whatever output mechanism required. Default is a WAIT WINDOW... **o.statusmessage(lcMessage)**
updatestructure	Method	Update the structure of the file based on the CreateTable method's structure. **o.updatestructure()**
validate	Method	Virtual method that can be used to hook up validation logic. No implementation in the base class.This method typically contains code that checks the content of the oData member. **o.validate()**
calias	Property	The Alias of the master file.
cconnectstring	Property	SQL ConnectString if DataMode=2
cdatapath	Property	Location of the data
cresultxml	Property	Methods returning XML will return the XML in this property.
cServerUrl	Property	A URL on the server that's running the wwHTTPSQLServer component to process client requests.
cskipfieldsforupdates	Property	Property that holds fields that are skipped for Insert and Update statements built on the fly. Use this if you pull extra fields that aren't part of the base cursor used in Save operations.
csql	Property	SQL String for a query
csqlcursor	Property	Requests that return a Fox cursor will use this name for the cursor name.
lValidateOnSave	Property	Determines whether Save() automatically calls Validate().
oHTTPSQL	Property	An instance member for an wwHTTPSQL class that handles data access over the Web when nDataMode is set to 4.

osql	Property	An instance of a wwSQL object that handles data connection if nDatamode = 2. All SQL commands run through this object for remote data.
vresult	Property	Variable result property. Used at this point only for ADO recordsets returned as a result from queries.

Conclusion

The wwBusiness class is a great tool to work as a data access layer for interfacing between your data and a business layer. The class would work well whether serving up data from within a web application as well as a fat client application. It is hard to beat three lines of code to update a record when dealing with web data by using the Load, FormVarsToObject, and Save methods.

Of course, there are many other useful tools within the WC framework that all add up to an unbelievable value and unbelievable productivity boost for application development.