

Using Treeviews with MS Access Applications

Pete Poppe (login MajP on Access Forums)

7 August 2024

Background and Access

- Career Marine Officer with multiple assignments in the supporting force combat systems acquisition, test and evaluation, experimentation, life cycle management
 - Introduced to access databases around 1999 to support Operational Testing
 - User surveys, event data, reliability data, test incident reports
 - Uniquely suited: Multiple events, remote test sites and ranges, disconnected from network
 - Realized I could do it better and cheaper
 - Read Litwin, Getz, Gunderloy Desktop Developer handbook
- After retiring continued to actively develop Access applications as a defense contractor to provide the same type of support
- Master of Science in Operations Research from Naval Postgraduate
 - Have demonstrated OR techniques in conjunction with Access (optimization modeling, shortest path, graph theory/networks, routing, assignment modeling, order fulfillment, scheduling)
- Not a developer, content creator, but a power user
 - Active on forums because I enjoy the problem solving
 - Disclaimer if demo databases fail

Objectives

- Generate some interest and take out some mystery of working with Treeviews
- Provide basic familiarization with capabilities and features of treeviews in Access and show what is possible
- Demonstrate the utility of Treeviews, especially when working with certain data structures
- Demonstrate a tips and tricks I use when dealing with Treeviews
- Provide and demonstrate a class module that hopefully makes working with treeviews and tying to a database easier

Agenda

- Intro
 - What a Treeview is
 - Examples
 - Basic Terminology
- Two types of Treeview Controls
 - How to get and install
- Utility / Value of Treeview examples
- Self Referencing Tables

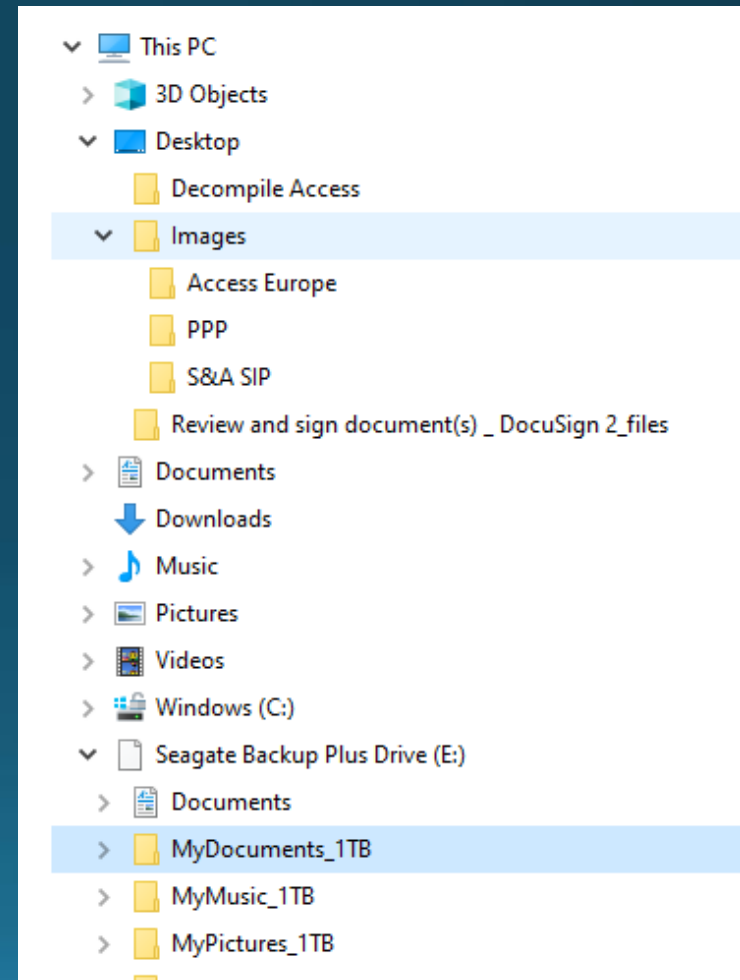
Agenda

- Recursion
- Tying the view to the data
- Basic Treeview functions
- Common Query
- Wrapper Class
- Image List

What is a Treeview Control

- The tree view control enables a hierarchical list using expanding and collapsing nodes

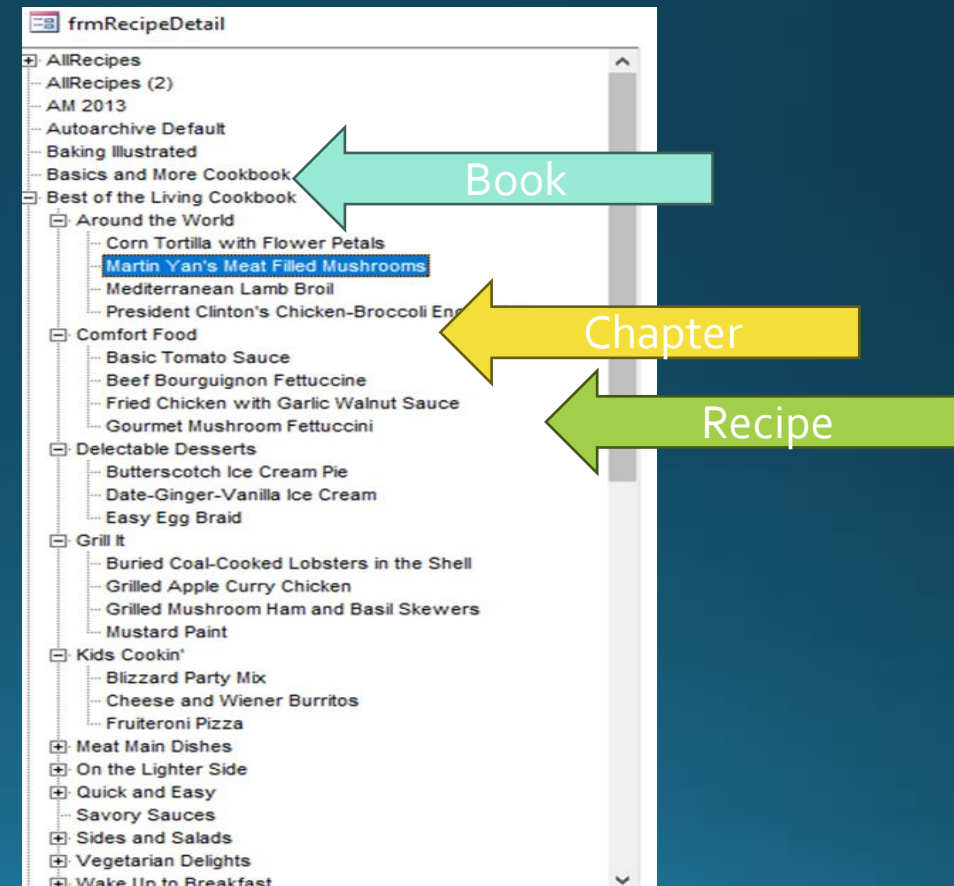
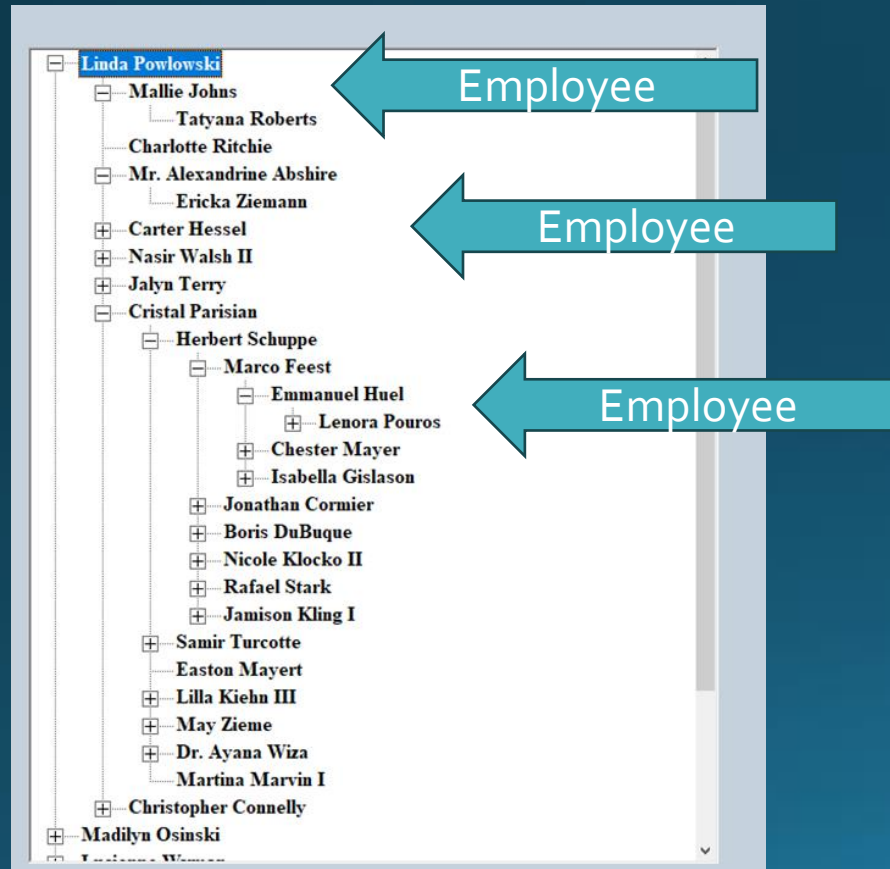
Windows File Explorer



Two Types of Data

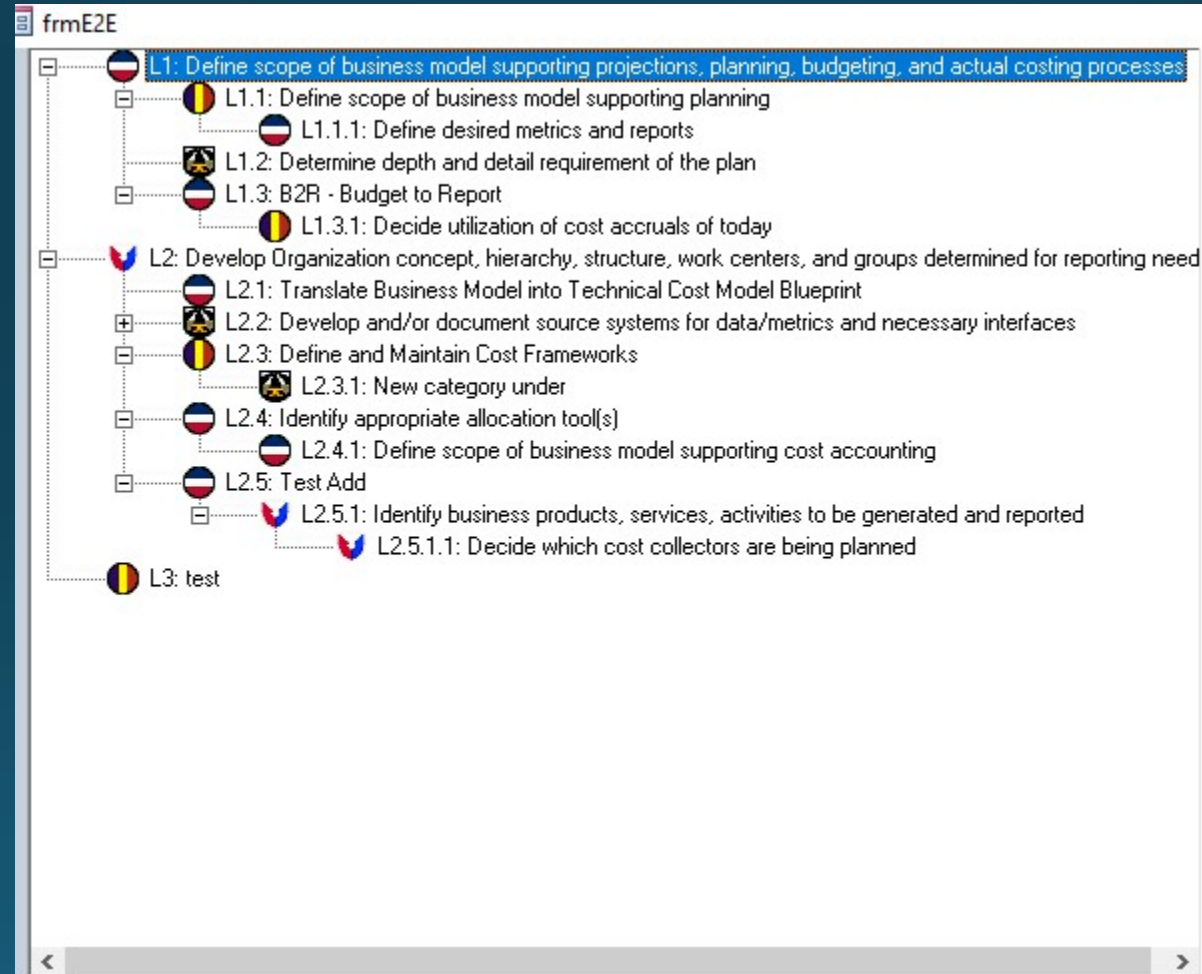
Hierarchical - Entities at each level are the same, but have a precedence

Organized Data - Entities differ at between level.
More traditional database related records



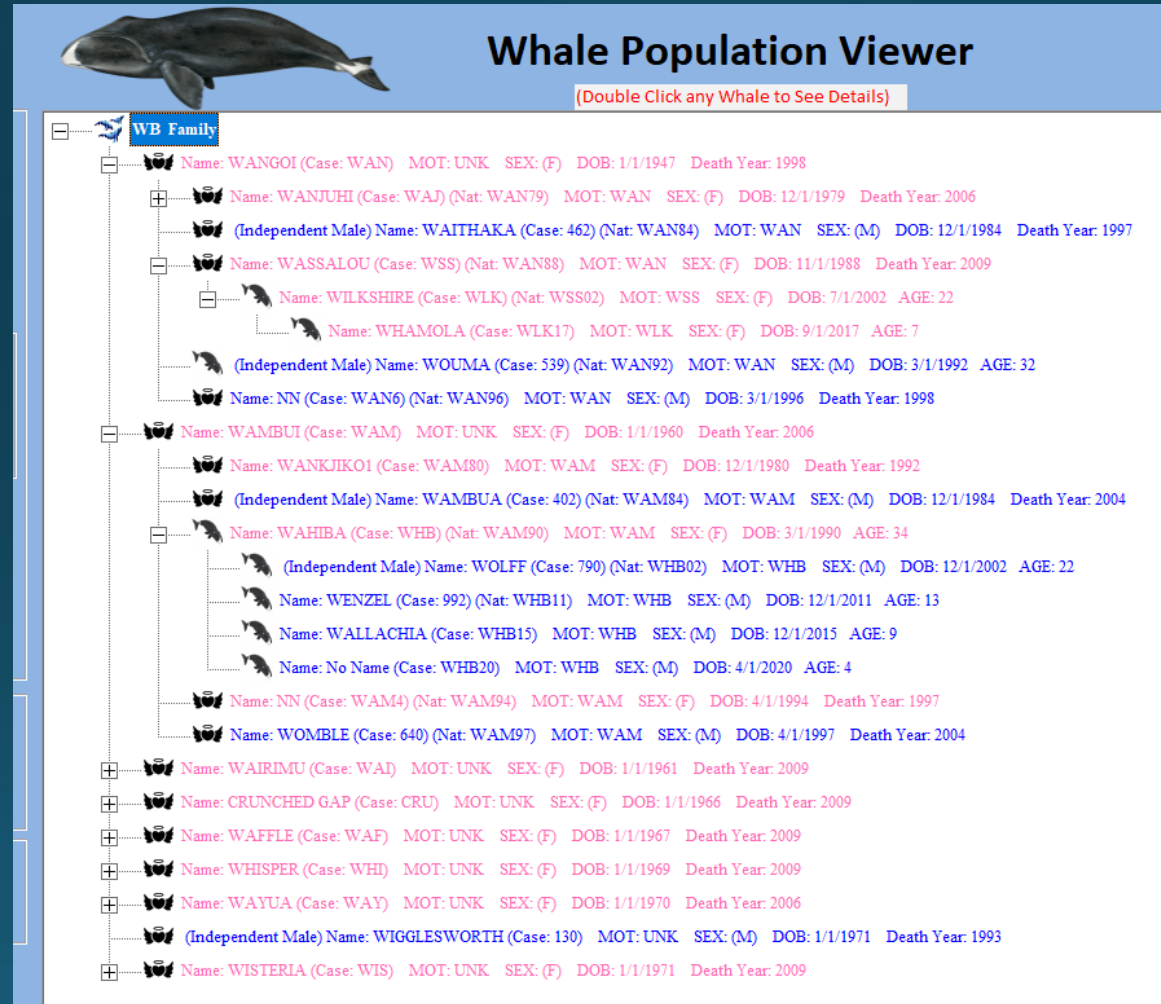
Custom Access Database Examples

Operational Tasks-SubTasks with Icons (Hierarchical)



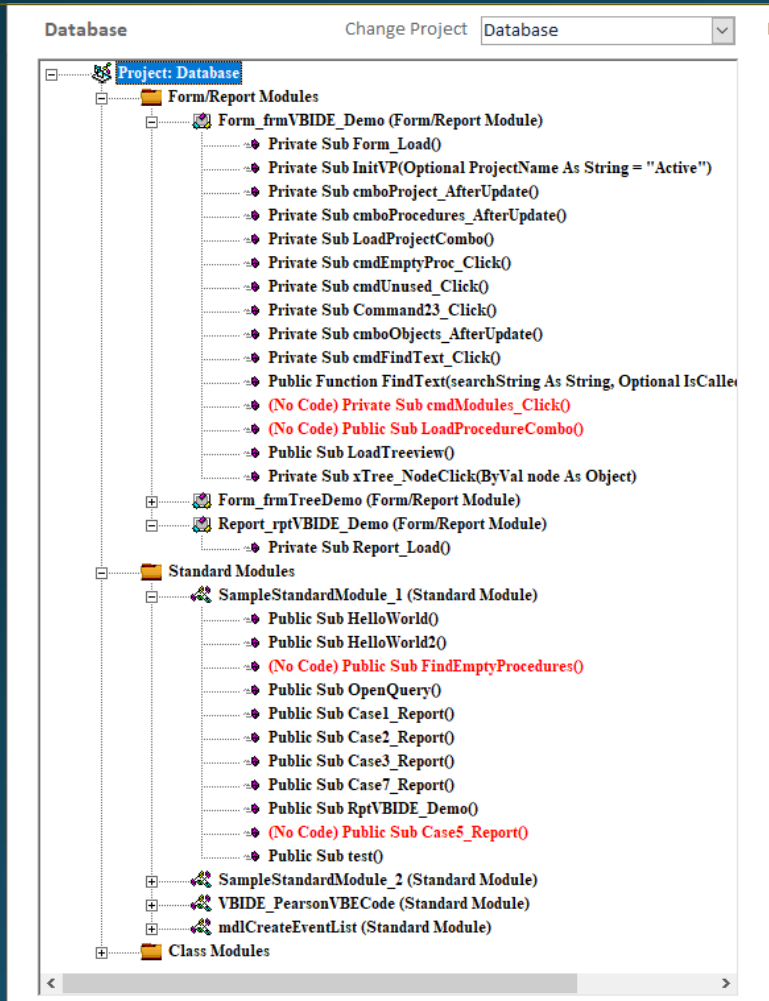
Custom Database Examples (2 of 3)

Whale Pod Genealogy icons and colors (Hierarchical)

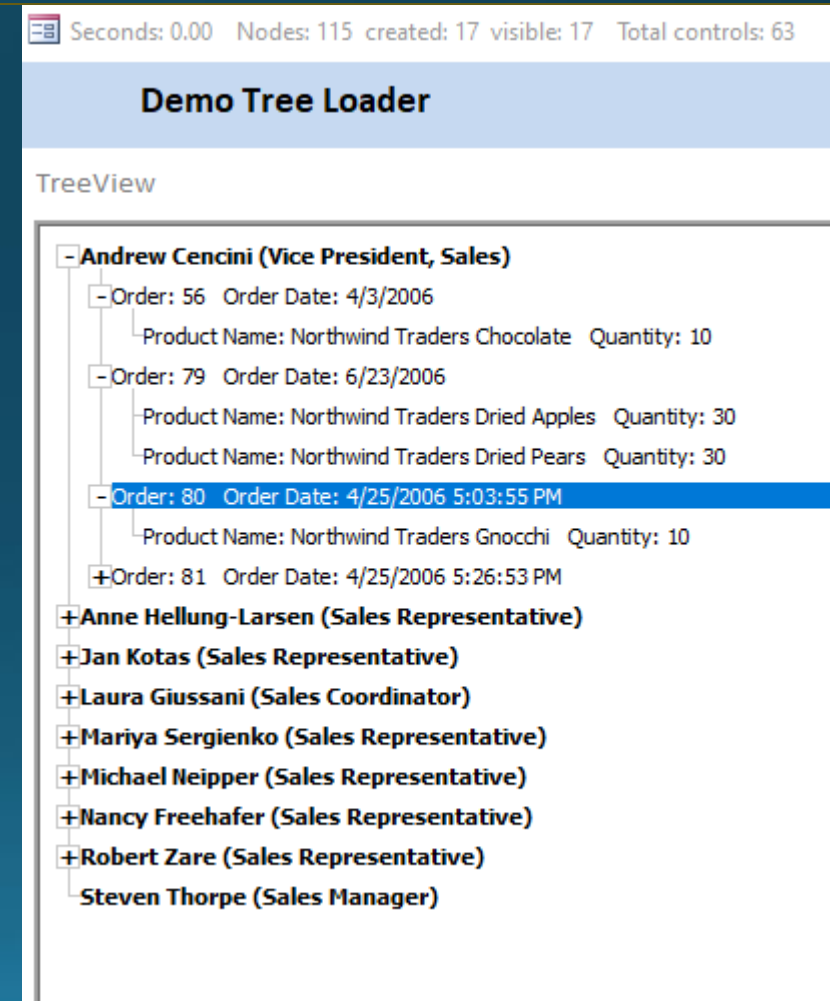


Custom Database Examples (3 of 3)

Mapping Database objects using VBA Extensibility (organized)

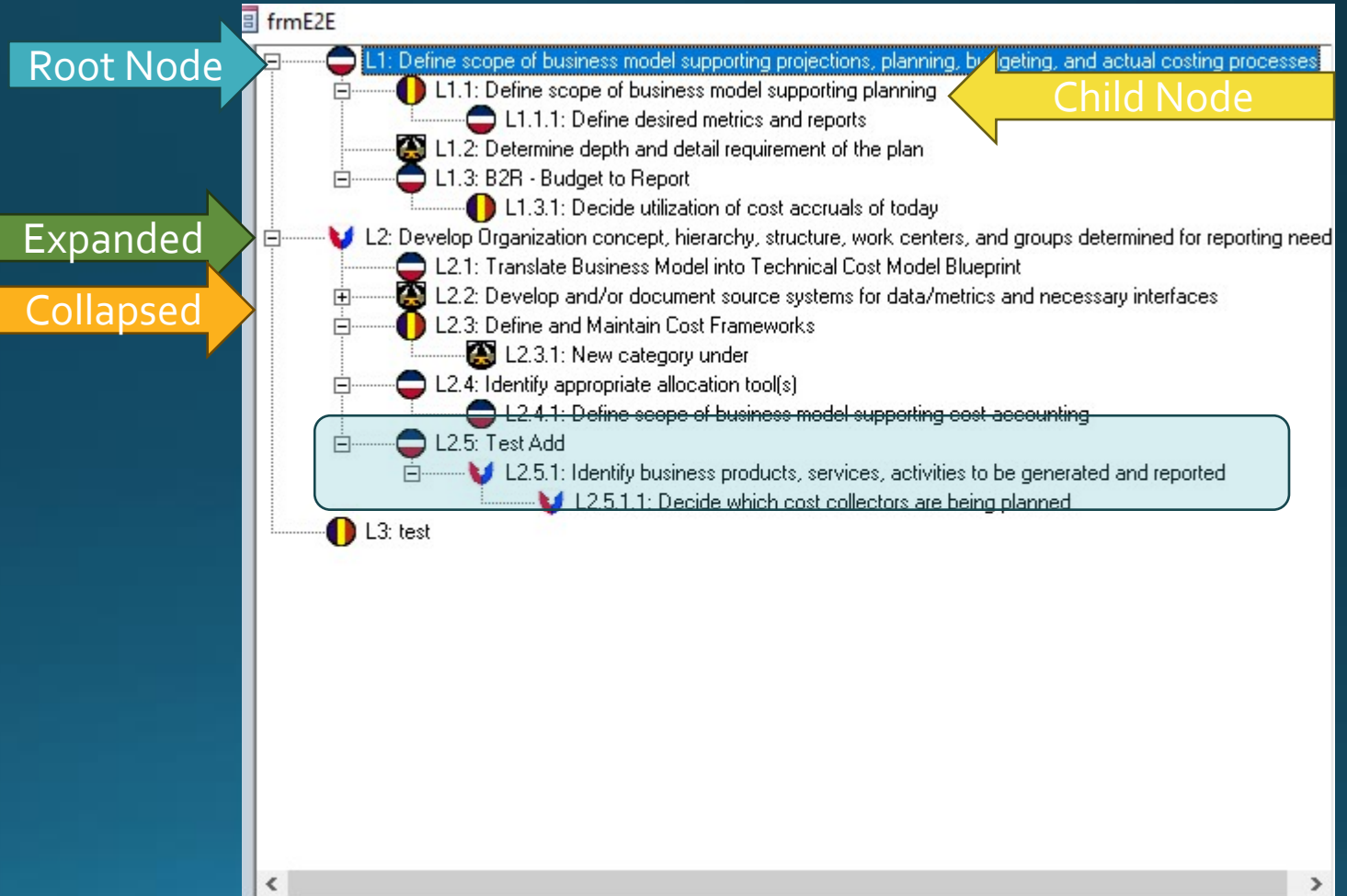


Northwind Employees, Orders, Products (organized)



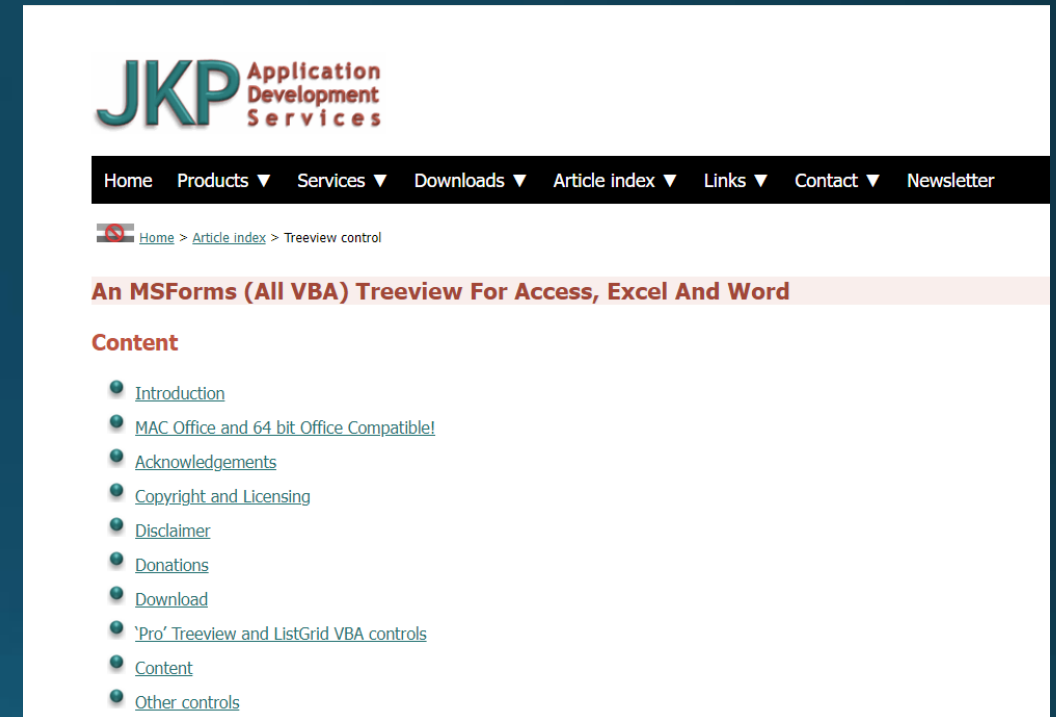
Terminology

- Node
 - Root Node
 - Child Node
 - Important Properties
 - Node Key (must be unique)
 - Node Text (visible text)
 - Node Level
- Branch
- Expanded / Collapsed



Two Free Treeview Controls exist

- Unfortunately, no native Access Control
 - Can be unstable in development and in use
 - Not easily portable
 - Not well documented
 - No native way to bind data (roll your own)
- **Active X (Microsoft Common Controls 6.0)**
 - Now supported in both 32 and 64 bit
 - Not exchangeable (must be built in correct version)
- **JKP Applications**
 - Based on MS Forms vba controls
 - <https://jkp-ads.com/articles/treeview.asp>
- **Which to Use**
 - Extremely similar in methods, properties, look
 - More existing code for Active X
 - More portable with JKP (Used MS Forms same as Excel and other Office Apps)
 - JKP does not support drag and drop AFAIK
 - JKP missing some events
 - (double click, drag/drop)



Late Entry -Access UI Ribbon and Tree builder

- Have not played with it so cannot comment
- <https://accessui.com/Products/RibbonTreeBuilder>



Custom Ribbons

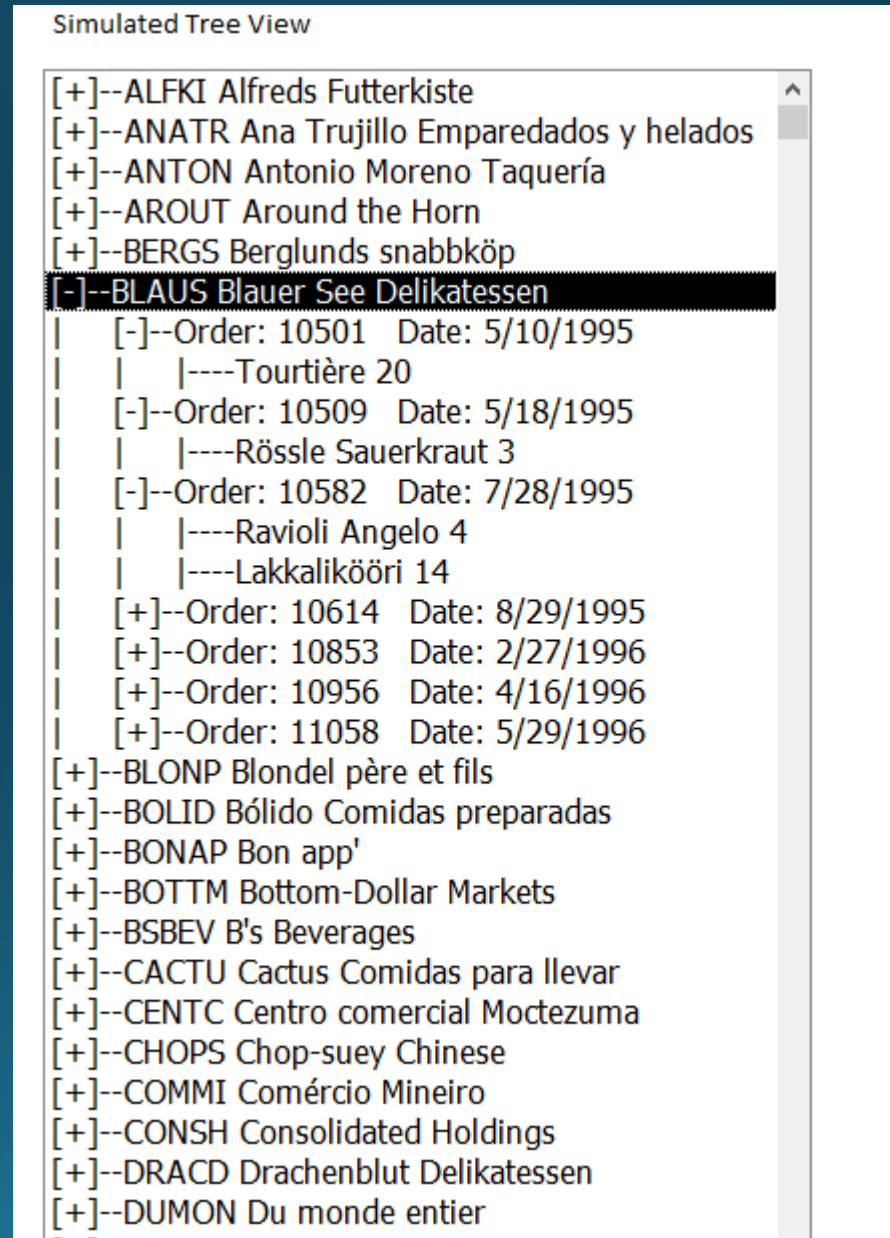
The AccessUI Ribbon & Tree Builder allows you to quickly create complex Ribbon menus in your application. It requires no knowledge of XML; you simply edit records in a form. You can respond to user actions, such as clicking a button, by creating callback methods. These callback methods are simple VBA functions that the Ribbon & Tree Builder can help you create.

The screenshot displays the AccessUI Ribbon Builder interface. At the top, a preview of the 'Northwind Traders' ribbon is shown with tabs for File, Reports, and Tools. Below this, the 'AccessUI Ribbon Builder' window is open, showing a tree view of the ribbon structure on the left and a 'Control Setup' form on the right. The tree view shows a hierarchy starting with 'Northwind Traders', followed by 'Employees' (containing 'Organization (Button)', 'All Employees (Button)', and 'Complex Tree (Button)'), 'Customers' (containing 'Customers by Country (Button)'), 'Products' (containing 'Products by Category (Button)' and 'Suppliers and Products (Button)'), 'Orders' (containing 'Orders by Employee (Button)', 'Shippers (Button)', and 'Sales Report (Button)'), 'Legacy' (containing 'Old Switchboard (Button)'), and 'Reports' (containing 'Sales By Category (Button)' and 'Sales By Year (Button)'). The 'Control Setup' form on the right is for a 'Button' control. It includes fields for 'Custom ID', 'Label' (set to 'All Employees'), 'idMsoImage' (set to 'people-40'), 'Image' (set to 'people-40'), 'Large' (checked), 'Description', 'Screen Tip', 'Super Tip', and 'Visible' (checked). A 'Callbacks' table is also present, listing methods like 'getLabel', 'getImage', 'getSize', 'getDescription', 'getScreenTip', 'getSupertip', 'getVisible', and 'getEnabled', each with a corresponding input field and a 'Pro' indicator.

Control Type	Custom ID	Label	idMsoImage	Image	Large	Description	Screen Tip	Super Tip	Visible	Callbacks	Pro	
Button		All Employees		people-40	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>	getLabel	<input type="text"/>	<input checked="" type="checkbox"/>
										getImage	<input type="text"/>	<input checked="" type="checkbox"/>
										getSize	<input type="text"/>	<input checked="" type="checkbox"/>
										getDescription	<input type="text"/>	<input checked="" type="checkbox"/>
										getScreenTip	<input type="text"/>	<input checked="" type="checkbox"/>
										getSupertip	<input type="text"/>	<input checked="" type="checkbox"/>
										getVisible	<input type="text"/>	<input checked="" type="checkbox"/>
										getEnabled	<input type="text"/>	<input checked="" type="checkbox"/>

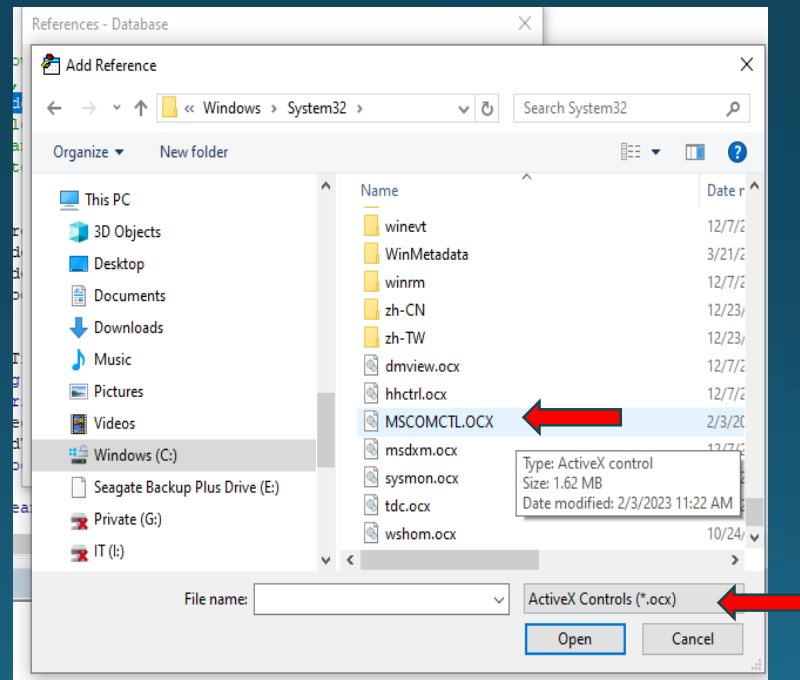
Late Entry 2 - Beta MajP Fake ListBox Treeview

- Uses standard listbox
- Class module that requires single line of code to initialize and load
- Demo on AWF
- <https://www.access-programmers.co.uk/forums/threads/help-on-locating-text-by-position-fake-treeview.331880/post-1931443>



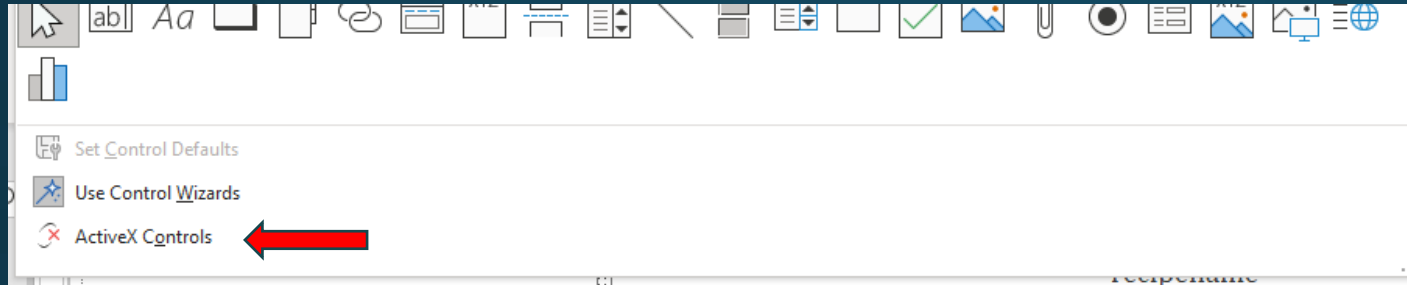
To Use Active X

- Need reference to Microsoft Common Controls 6.0
 - From references browse to Windows – System32
 - Make sure you select ActiveX Controls (*.OCX)

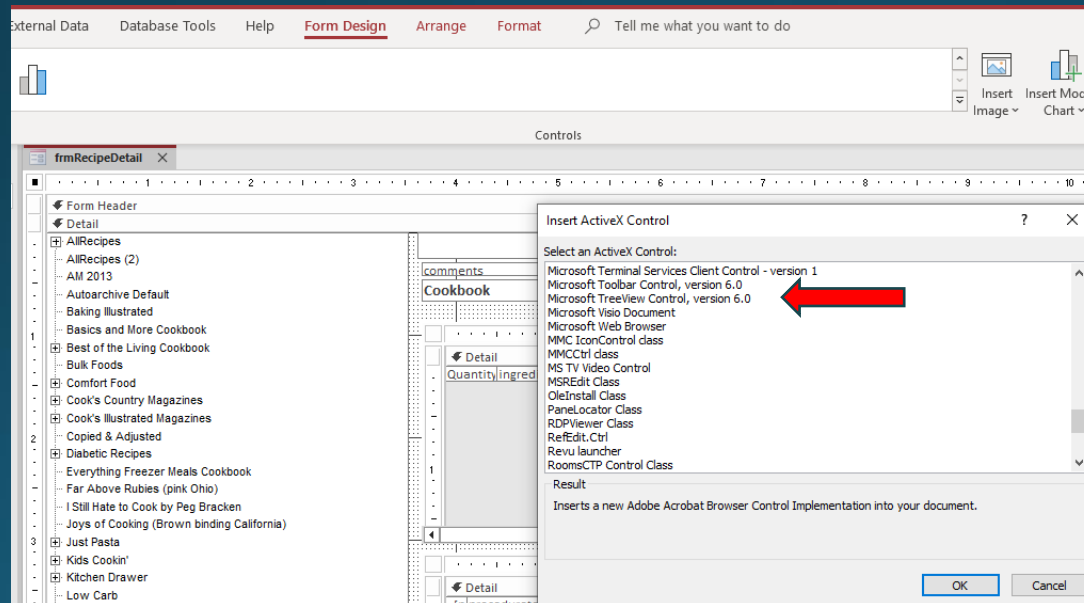


To Use Active X (2 of 2)

- On a form select Active X in more controls



- Browse to Microsoft Treeview Control Version 6.0



Need / Utility

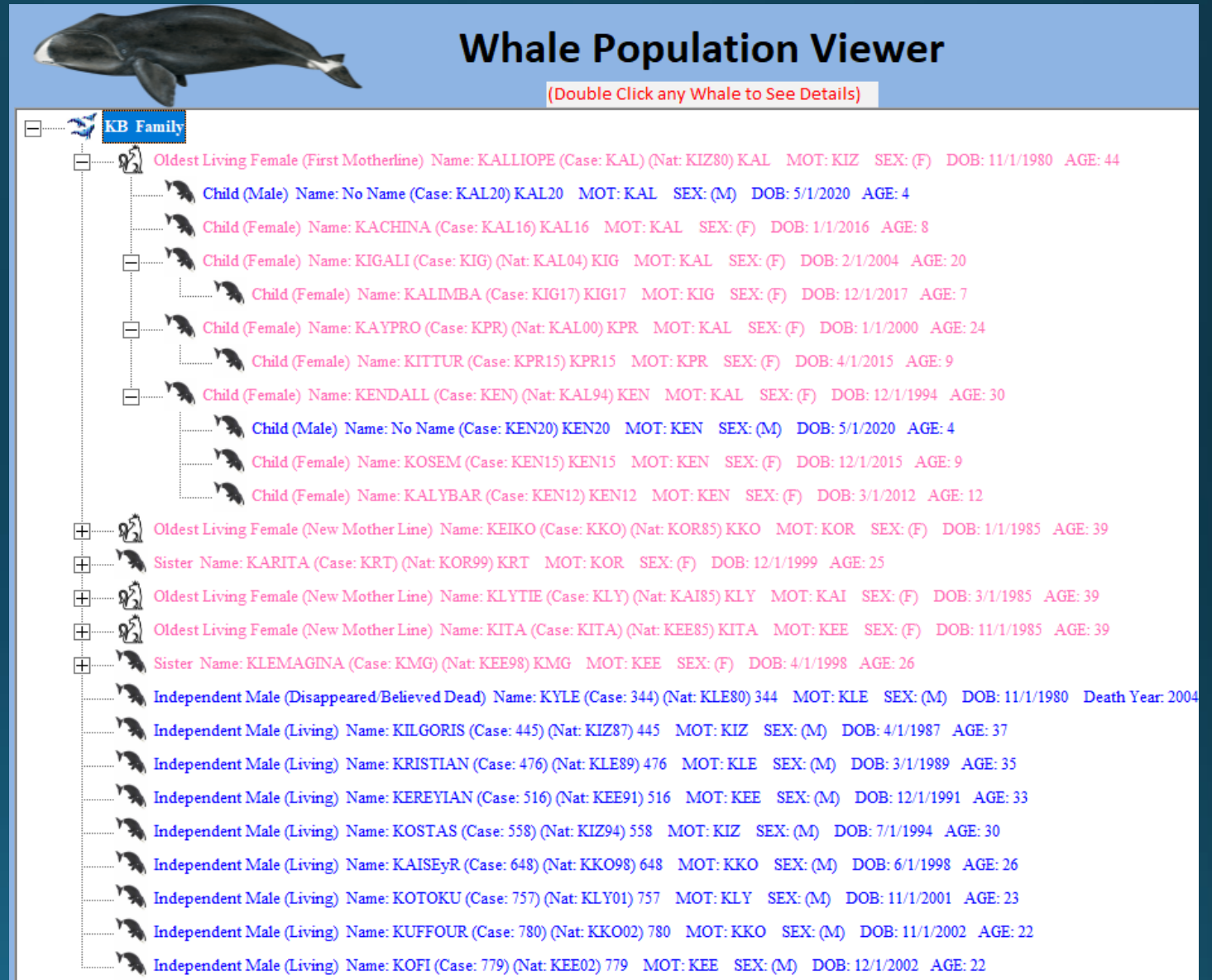
- **DATA VISUALIZATION:** Visualize data that is almost impossible in native Access controls
 - Ability to show data where the number of levels is undetermined and no set limit
 - File Structure
 - Family Trees
 - Systems-sub systems
 - Tasks – sub tasks
- **DATA MANAGEMENT / NAVIGATION:** Add, enter, reorganize easily that would be near impossible
 - Collapsing and expanding very large list
- **UI CONTROL**
 - Can replace multiple controls (cascading combos)

Once you implement it within certain data structures, you will wonder how you lived without it

Ability to Visualize Data and Relationships

Request to sort data by POD Oldest living female, then sisters by age, female cousins by age, then by dependent males (living in pod), then independent males (living away from pod). BTW females break away and start new pods so need to track old and new Pods

The above is a “living” pod and extremely hard to visualize without seeing the whole family tree



As a Navigation Control or Combo/List Control

frmRecipeDetail

I Still Hate to Cook by Peg Bracken

Joys of Cooking (Brown binding California)

Just Pasta

Kids Cookin'

Kitchen Drawer

Low Carb

Low Carb Diner

Newspaper

Not Your Mother's

Paper Slips

Quarantined Duplicates

Quick and Easy

Appetizers

Beef

BBQ Beef Sandwiches

Beef and Eggplant Casserole

Beef and Salsa Topped Potatoes

Beef Choufleur

Beef Satay

Beef Stroganoff Bake

Beefy Green Chile and Cheese Bake

Biscuit and Beef Cups

Broiled Steaks With Herb-Cheese Potatoes

Cheeseburger Casserole

French Onion Beef

Glazed Corned Beef

Lone Star Steak and Pasta

Provençal Beef Stew

Quick 'n' Easy Beef Quesadillas

Six-Can Slow Cooked Chili

Tangy Beef Barbecue

Beverages

Breads

Canapes

Casserole

Cheese

Chicken

Broiled Steaks With Herb-Cheese Potatoes

When unexpected guests show up, broil frozen steaks with potato wedges for a delicious dinner in no time! Microwaved or steamed

Quick and Easy

306

Amount	Ingredient
4	(4-oz.) frozen beef tenderloin or top loin petite steaks, cut 1-inch t
2	baking potatoes, cut lengthwise into 6 wedges each
1 tsp	crushed dried basil
1 Tbs	grated Parmesan cheese
1/4 tsp	ground black pepper
1 Tbs	olive oil
*	

Index	Procedure
2	Place frozen steaks on one side of rack in broiler pan; arrange potato wedges on the o
3	Combine oil, basil and pepper. Brush on potatoes. Broil meat and potatoes 5 to 6 inch
4	Sprinkle cheese over potatoes. Season with salt, if desired.
*	

Record: 10 of 64 No Filter Search

Ability to add, enter, and edit Hierarchical data

- IMO it is extremely difficult to build a UI using native Access controls that lets you add, edit, relate, delete Hierarchical Data.
- Think how this could be done in native controls
 - Ability to visualize Hierarchical data
 - Ability to easily add, delete and move
 - Move complete branch
 - Order within branch

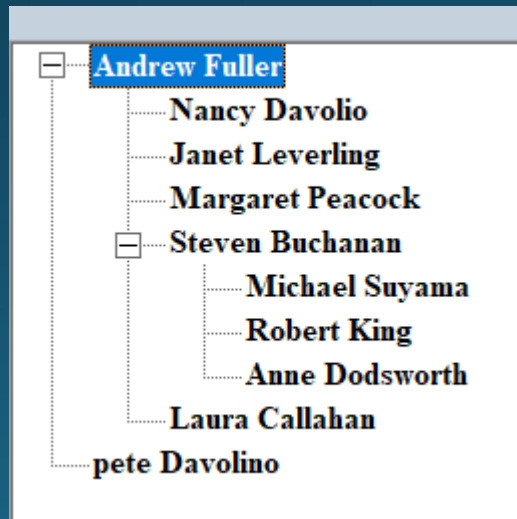
Demo 1

Demo 2
Systems
Sub Systems

Self Referencing Tables

- Often TreeViews can illustrate Hierarchical data
- Hierarchical data – All entities are the same thing but have a precedence.
 - All entities have the same properties can be stored in the same table
 - All entities have a parent
 - Examples Employees have a boss that they report to (Parent ID)

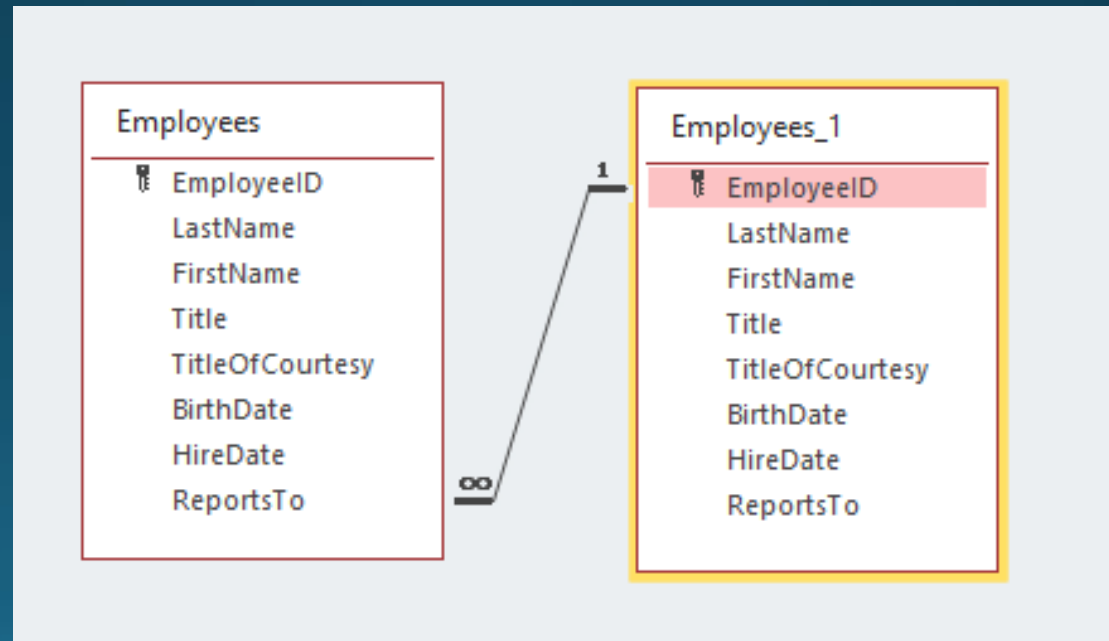
Employee ID	Last Name	First Name	Reports To
1	Davolio	Nancy	2
2	Fuller	Andrew	
3	Leverling	Janet	2
4	Peacock	Margaret	2
5	Buchanan	Steven	2
6	Suyama	Michael	5
7	King	Robert	5
8	Callahan	Laura	2
9	Dodsworth	Anne	5
10	Davolino	pete	



- I make top level nodes Parent NULL
 - Some people will make the Parent ID same as record ID
 - Some people will make a dummy record

Self Referencing Tables Can Enforce Referential Integrity

- You can enforce Referential Integrity to include Cascade Deletes
- This is not a trivial requirement because if you need to recurse the data, a bad ID may break a major branch and you lose visibility on many records.



Recursion

- Recursive code can support Hierarchical data and in turn loading a tree view.
 - Not required but self referencing tables, hierarchical data, recursion and Treeviews go hand in hand
 - IMO recursion is extremely confusing
- Example of Recursive Code: Factorial $N!$ is $N * n-1 * n-2 \dots 1$
 - $5! = 5 * 4 * 3 * 2 * 1$
 - Yes, This example can be done in a simple loop but others cannot
 - Determine number of employees reporting to boss
 - Return all Children, Grand Children...
 - Get all files in a folder, subfolders
 - Return totals from branch

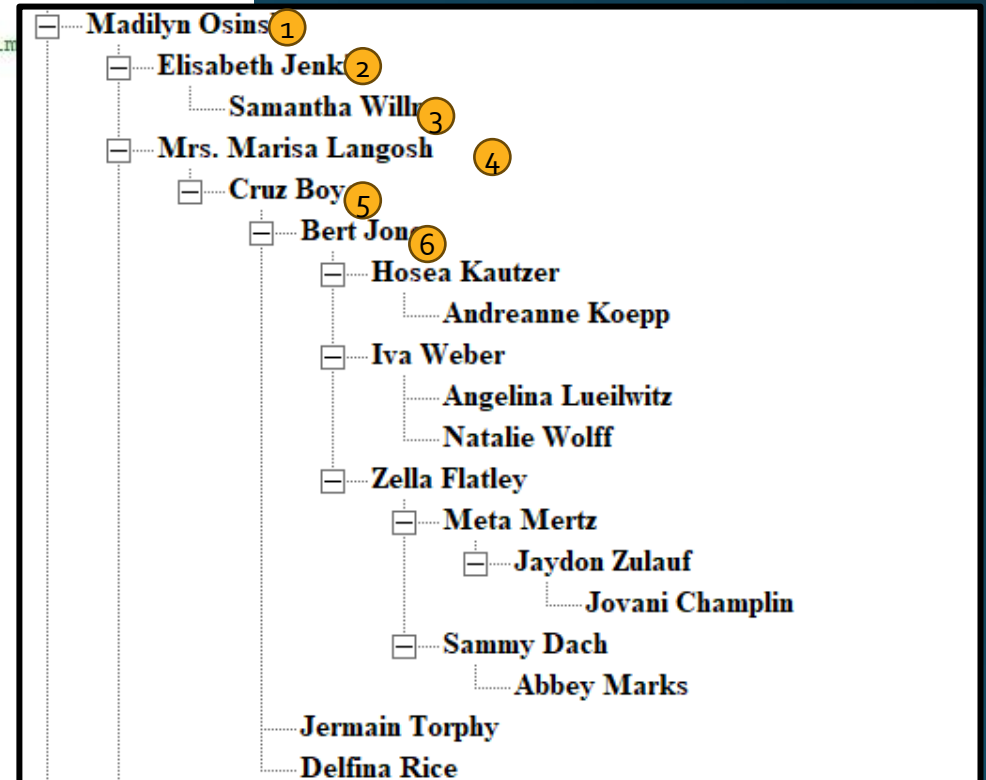
```
Public Function Factorial(ByVal N As Long) As Long
    If (N = 1) Then
        Factorial = N 'Must ensure there is something to terminate the recursive call
    Else
        Factorial = N * Factorial(N - 1)
    End If
End Function
```


Loading a Tree Using Recursion

```
Public Sub addBranch(parentIDValue As Variant, identifier)
    On Error GoTo errLabel
    Dim strCriteria As String
    Dim bk As String
    Dim currentID As Variant
    Dim currentText As String
    Dim currentNode As Node
    Dim parentNode As Node
    Dim strKey As String
    Dim currentImage As String
    Dim rsTree As Recordset
    Set rsTree = Me.Recordset
    strCriteria = Parent_ID_Field & " = '" & parentIDValue & "'"

    'Since it is not a root node must determine the parent node. All node keys are "ID " & the prim
    Set parentNode = Me.Nodes(parentIDValue)
    rsTree.FindFirst (strCriteria)
    Do Until rsTree.NoMatch
        currentID = rsTree.Fields(ID_Field)
        currentText = rsTree.Fields(Node_Text_Field)
        identifier = rsTree.Fields(Identifier_Field)
        Set currentNode = Me.Nodes.Add(parentNode, tvwChild, currentID, currentText)
        currentNode.Tag = identifier
        If mLoadImages Then
            currentImage = Nz(rsTree.Fields(Image_Name_Field), "")
            If currentImage <> "" Then currentNode.Image = currentImage
        End If
        bk = rsTree.Bookmark
        Call addBranch(currentID, identifier)
        rsTree.Bookmark = bk
        rsTree.FindNext (strCriteria)
    Loop
Exit Sub
errLabel:
    MsgBox Err.Number & " " & Err.Description & " In addBranch"
    If MsgBox("Do you want to exit the loop?", vbYesNo, "Error In Loop") = vbYes Then
        Exit Sub
    Else
        Resume Next
    End If
End Sub
```

- Pass in a parent
 - Loop children
 - If Child exists recurse
 - Until no child exists then start falling out and working backwards



Tying a Treeview to the Data

- There is no native link between the data and the view
- The trick is to build your own methods and use the Treeview object to “pseudo bind” the data
- I have attempted to do much of the with a “wrapper” class called TreeviewForm
 - Automates the loading by providing a standard query
 - Single line of code to load any tree
 - Allows “light load”
 - Loads images
 - Supports drag and drop

Tying a Treeview to the Data (2 of 2)

- Modify the data, must Update the Tree
- Modify the tree, must Update the Data
- The corresponding record determined from the Node Key and Node Tag (Can determine PK and Table)
- Still requires the use to add database method
 - Node Delete event – User must write delete query
 - Add Record – Update treeview
 - Modify Treeview – User must write update query
 - Modify Record – Update treeview
 - Drag Drop Event – User writes update query to update Parent Record of dropped node to dropped Target

Common Query

- Benefits
 - Allows reusable code (field names and content always the same)
 - Works for both Hierarchical and organized
- Fields
 - Identifier – Something that tells which table the record is from so if it comes from a Union query. This can be the name of the table.
 - Important with organized data coming from two tables (EMP10 and CUST10)
 - ID – Concatenate an Identifier with the actual PK
 - ParentID – Concatenate the parent Identifier with the parent PK
 - NodeText – What you want to display. Often concatenated data
 - Image – Optional if using an icon. This is the key used by the imagelist

Example (Self Referencing)

Person_ID	Full_Name	Boss_ID
0	Linda Powlowski	
1	Madilyn Osinski	
4	Joshuah Parisian	3
5	Forest Oberbrunner	4
9	Gwen Wunsch	8
10	Ona Christiansen II	11

Root Node

Source Table

Reports to Josh

ID	ParentID	NodeText	Identifier
TD0	TD	Linda Powlowski	TD
TD1	TD	Madilyn Osinski	TD
TD10	TD11	Ona Christiansen II	TD
TD100	TD1	Elisabeth Jenkins	TD
TD1000	TD23	Maryam Erdman	TD
TD1001	TD534	Lance Ward	TD

Root Nodes

Transformed Query

Parent is Madilyn

- Save the ID as the Node Key (Ensures Key is a String, Unique with union tables)
- Save the Identifier in the Node Tag
- To get the PK, Remove the identifier from the Node Key (link to the database)
- The Identifier can be the Table Name or you can use a select case to determine the table from the Identifier

Example (Organized)

- Create query for each table (level) as shown
- Combine the queries in a Union Query
- Remember the Parent ID is the parent Identifier & PK

ID	parentID	nodeText	identifier
CustGOURL		GOURL Gourmet Lanchonetes	Cust
CustGREAL		GREAL Great Lakes Food Market	Cust
CustGROSR		GROSR GROSELLA-Restaurante	Cust
Ord10254	CustGOURL	Order: 10254 Date: 8/11/1994	Ord
Ord10258	CustERNSH	Order: 10258 Date: 8/17/1994	Ord
Ord10259	CustGREAL	Order: 10259 Date: 8/18/1994	Ord

Tips / Tricks

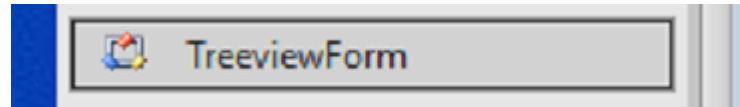
- When working with Properties Access shows only generic Active X properties
- AFAIK there is not Custom Properties display like other Active X
- So, I do most formatting and setting of properties in code
- However, if you have a Treeview Control you need to reference the Treeview Object (if someone understands this please identify in chat)
 - Example Treeview control on your form Named "Xtree"
 - Me.Xtree.Font.Size = 12 works but no Intellisense
 - Need to:
 - Dim TVW as Treeview
 - Set TVW = me.xtree.object
 - TVW.... Intellisense works

Key Treeview Methods

- Node Properties
 - Key – Must be unique and a String. (Super sensitive)
 - Tag – Extra property can be used to help relate a record to a node
 - Node Text – What you see
- Node Add Method:
 - Add([Relative], [Relationship], [Key], [Text], [Image], [SelectedImage]) As Node
 - Relative: Another node object. Either left blank (default) or most of the time it is the Parent Node
 - Relationship: If the Relative is the Parent node, then it is the child (twvChild)
 - Key – **MUST be unique! Must be Numeric!**

TreeviewForm – Wrapper class

```
Const ID_Field = "ID"  
Const Parent_ID_Field = "ParentID"  
Const Node_Text_Field = "NodeText"  
Const Identifier_Field = "Identifier"  
Const Image_Name_Field = "NodeImage"
```



```
Private WithEvents mTVW As TreeView  
Private mQueryName As String  
Private mRootParentID As String  
Private mRecordSet As DAO.Recordset  
Private mLoadType As lt_LoadType  
Private mAllowDragDrop As Boolean  
Private mLoadImages As Boolean
```

Treeview where you can trap its events

Recordset based on your Query

```
Public Event TreeviewNodeClick(SelectedNode As Node)  
Public Event TreeviewNodeCheck(CheckedNode As Node, Checked As Boolean)  
Public Event TreeviewNodeDragDrop(DragPK As Variant, DropPK As Variant, DragNode As Node, DropNode As Node)  
Public Event TreeviewNodeDbClick(SelectedNode As Node)  
Public Event RightClickOnNode(SelectedNode As Node)  
Public Event RightClickOffNode()  
Public Event NodeIterated(Node As Node)  
Public Event ExpandedBranch(ExpandedNode As Node)  
Public Enum lt_LoadType  
    lt_lightload = 1  
    lt_fullload = 2  
End Enum
```


Initialize Event

```
Public Sub Initialize(  
    objTreeView As TreeView,  
    QueryName As String,  
    ParentStartID As String,  
    Optional LoadImages As Boolean = False,  
    Optional LoadType As It_LoadType = It_lightload,  
    Optional AllowDragDrop As Boolean = False)
```

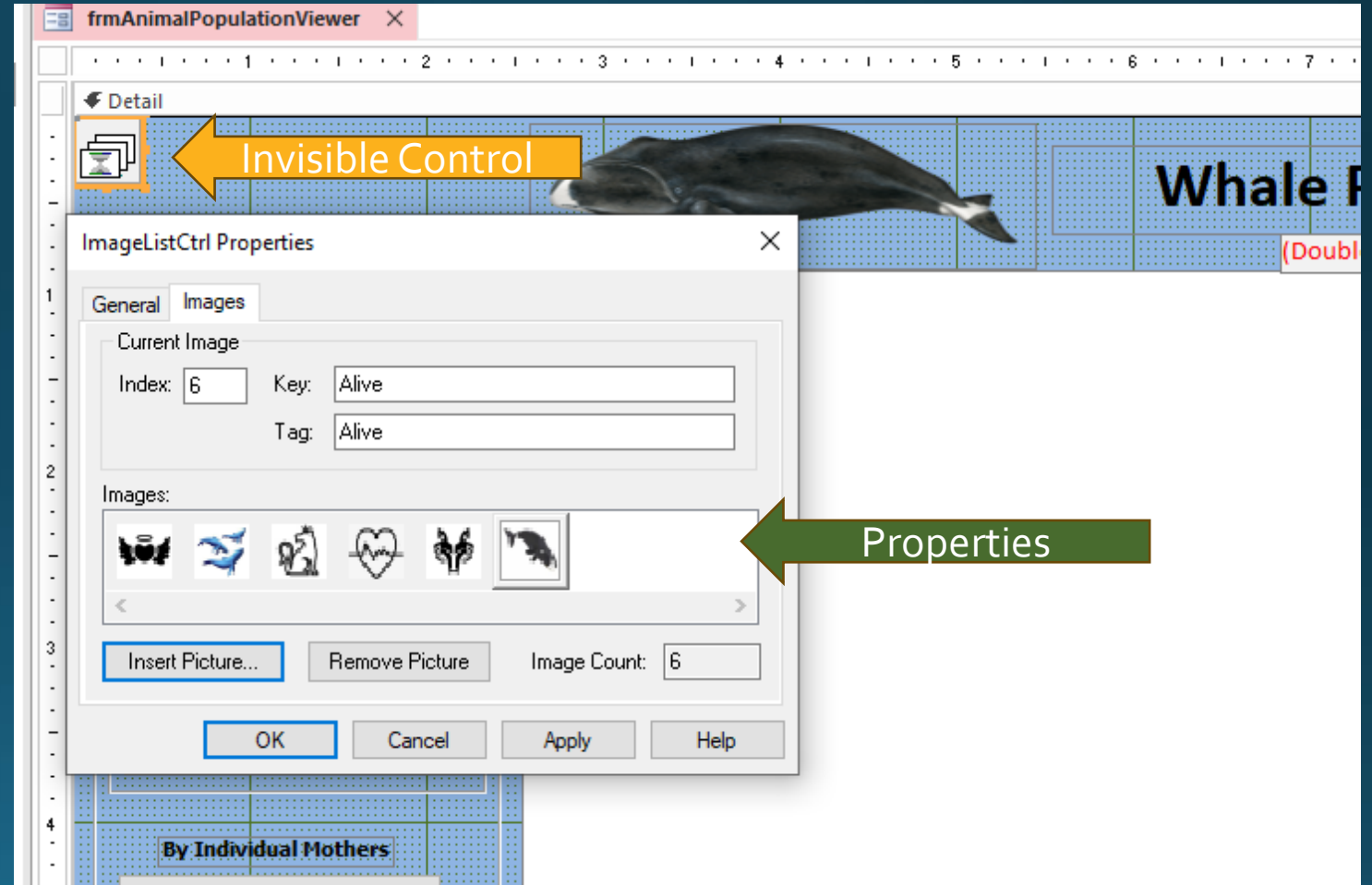
- Load Type
 - Full Load
 - Light Load

Key Methods

- Initialize and Recursively load methods / Light load
- Public Property Let / Get AllowDragDrop() As Boolean
- Public Property Get Nodes() As Nodes
- Public Property Get Recordset() As DAO.Recordset
- Public Property Get SelectedNode() As Node
- Public Property Get SelectedNodeIdentifier() As String
- Public Property Get SelectedNodeLevel() As Integer
- Public Property Get SelectedNodePK() As Variant
- Public Property Get TreeView() As TreeView
- Public Sub ClearTree()
- Public Sub CollapseBranch()
- Public Sub CollapseTree()
- Public Sub DeleteNode(strIndex As String)
- Public Sub deleteSelectedNode()
- Public Sub ExpandTree()
- Public Sub ExpandBranch()
- Private Sub ExpandChildren(theNode As Node) Recursive
- Public Sub IterateBranch(ParentNode As Node) Recursive
- Public Sub MoveDown()
- Public Sub MoveUp()
- Public Function GetDescendants(theNode As Node, Optional theDescendants As Collection = Nothing) As Collection
- Public Function getNode(strKey As String) As Node
- Public Function childNodes(ByVal TheParentNode As Node) As Collection
- Public Function getNodeIdentifier(theNode As Node) As String
- Public Function GetNodeLevel(myNode As Node) As Integer
- Public Function getNodePK(theNode As Node) As String
- Public Function GetSortOrder() As Collection

Image List

- Also, part of MS Common Controls
- Invisible control that holds images
- You place it on the form with the Treeview and is basically the container to hold images for use in the Treeview
- Once you associate the image list with the tree you simply call the image by its Key to load
- I do not know if you can load icons other ways, Never tried.
- JKP has its own techniques



Try To Build from Scratch Using Wrapper Class

- Show Employee Query
- Show Active X
- Show Instantiate initialize.

Links

- JKP Treeview: <https://jkp-ads.com/articles/treeview.asp>
- Examples and Discussions by MajP on AWF: <https://www.access-programmers.co.uk/forums/threads/hierarchical-data-recursion-tree-views-and-a-custom-class-to-assist.309753/>
- Access UI Ribbon Tree Builder: <https://accessui.com/Products/RibbonTreeBuilder>
- Picoware Treeview <https://www.picoware.de/info/treeview.htm>