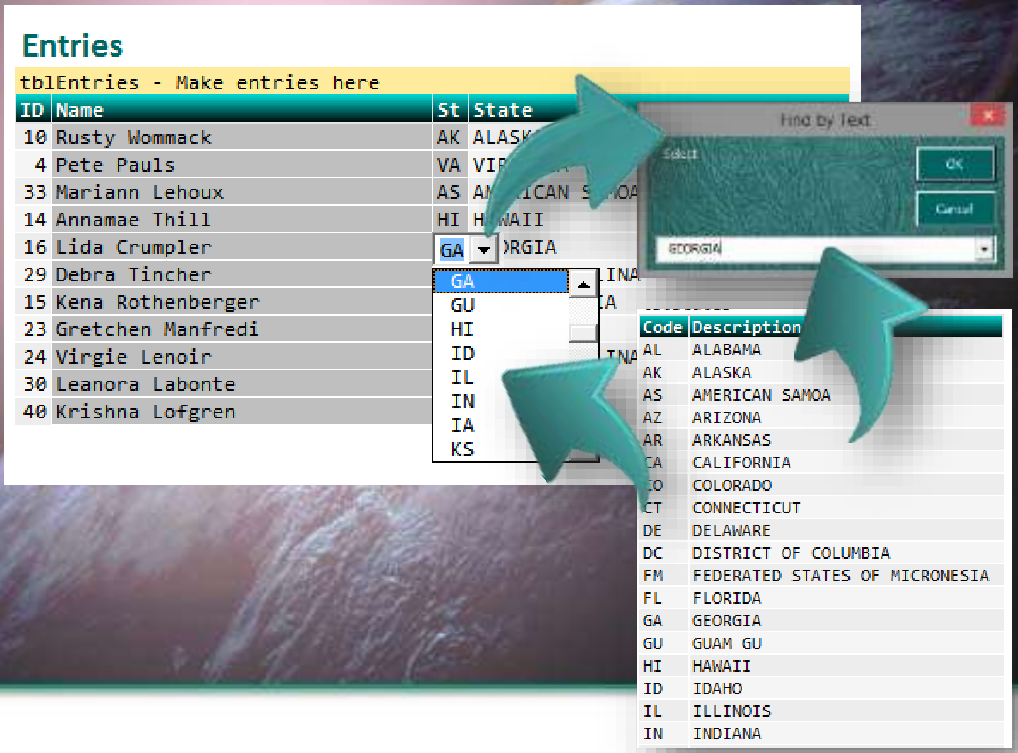
Beyond Excel

Add-In

ComboBox with Search by Description



Entries

tblEntries - Make entries here

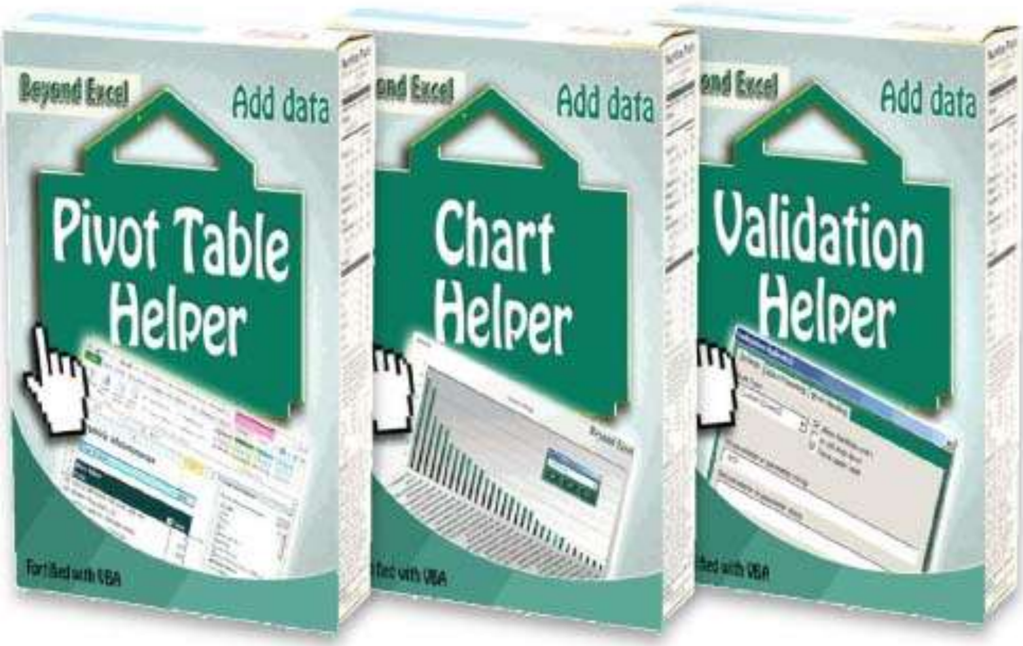
ID	Name	St	State
10	Rusty Wommack	AK	ALASK
4	Pete Pauls	VA	VIR
33	Mariann Lehoux	AS	AM ICAN S OA
14	Annamae Thill	HI	H WAII
16	Lida Crumpler	GA	RGIA
29	Debra Tincher		LINA
15	Kena Rothenberger		IA
23	Gretchen Manfredi		
24	Virgie Lenoir		INA
30	Leanora Labonte		
40	Krishna Lofgren		

GEORGIA

Code	Description
AL	ALABAMA
AK	ALASKA
AS	AMERICAN SAMOA
AZ	ARIZONA
AR	ARKANSAS
CA	CALIFORNIA
CO	COLORADO
CT	CONNECTICUT
DE	DELAWARE
DC	DISTRICT OF COLUMBIA
FM	FEDERATED STATES OF MICRONESIA
FL	FLORIDA
GA	GEORGIA
GU	GUAM GU
HI	HAWAII
ID	IDAHO
IL	ILLINOIS
IN	INDIANA

WHAT IS BEYOND EXCEL?

Beyond Excel is a free "cookbook" approach to extending Excel with **VBA**. Practical solutions are offered instead of abstract theory. Those desiring more theory and/or a **VBA** language reference are encouraged to seek out the many superior resources available for purchase or free on the internet. Links to some of those resources are provided inside.



"Cookbook approach" is an analogy to recipes that provide instructions on how to prepare a final product, such as a cake or casserole, instead of theory on how to cook in general.

Table of Contents

Why Read This?

Make data entry easier.

Problem

Data validation's dropdown lists do a good job of preventing bad entries and for small lists they help find the right code. Dropdowns, however, fall short for larger lists or longer codes. We need what dropdowns lack. We need autocomplete. We need lists positioned by partial entries. We need to search for codes by description.

Autocomplete

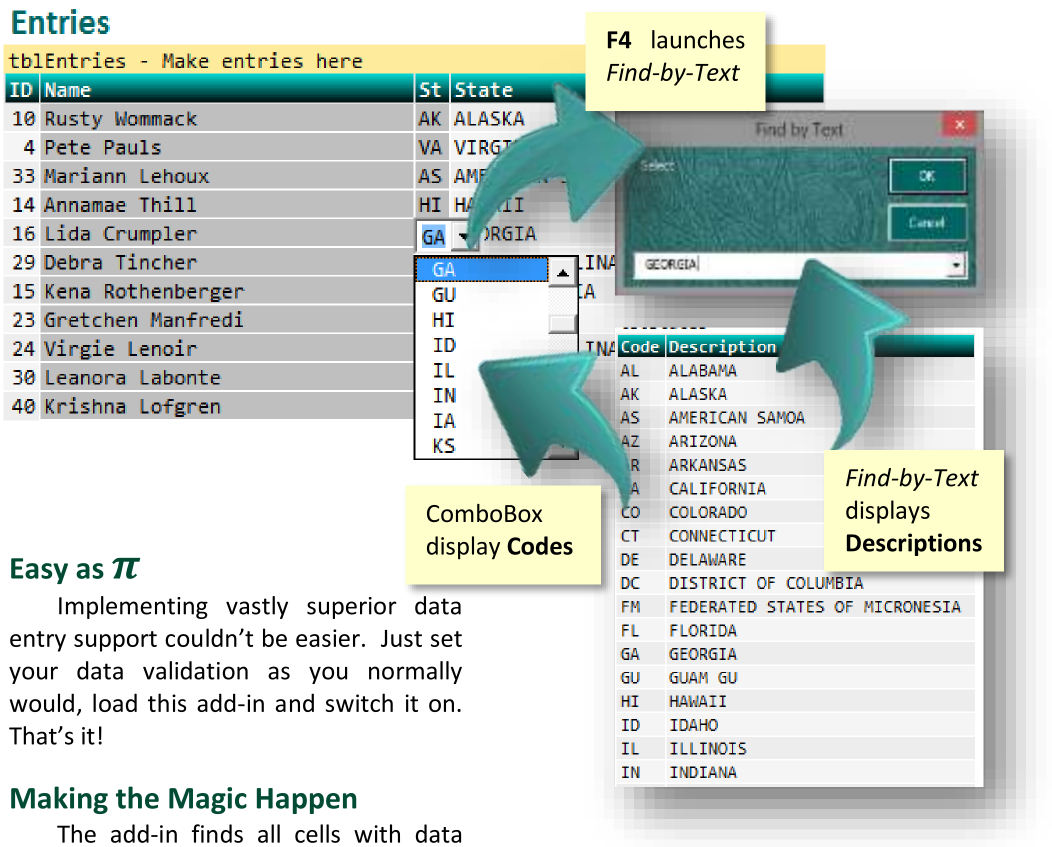
ComboBoxes find matching entries as we type and they often autocomplete with the correct entry after just a few keystrokes. This greatly improves data entry efficiency ***if*** we know the correct spelling. What if we don't?

Position by Partial Entry

Like dropdowns we can scroll through ComboBoxes' value list to pick a code that looks right. This helps if we have some idea of what the code is. What if we don't?

Search for Code by Description

This add-in provides another very useful feature, *find-by-text*. With the ComboBox displayed, press **F4** to display and search by code descriptions.



Entries

tblEntries - Make entries here

ID	Name	St	State
10	Rusty Wommack	AK	ALASKA
4	Pete Pauls	VA	VIRGI
33	Mariann Lehoux	AS	AME
14	Annamae Thill	HI	HA II
16	Lida Crumpler	GA	RGIA
29	Debra Tincher		INA
15	Kena Rothenberger		A
23	Gretchen Manfredi		
24	Virgie Lenoir		INA
30	Leanora Labonte		
40	Krishna Lofgren		

Code	Description
AL	ALABAMA
AK	ALASKA
AS	AMERICAN SAMOA
AZ	ARIZONA
R	ARKANSAS
A	CALIFORNIA
CO	COLORADO
CT	CONNECTICUT
DE	DELAWARE
DC	DISTRICT OF COLUMBIA
FM	FEDERATED STATES OF MICRONESIA
FL	FLORIDA
GA	GEORGIA
GU	GUAM GU
HI	HAWAII
ID	IDAHO
IL	ILLINOIS
IN	INDIANA

F4 launches *Find-by-Text*

ComboBox display **Codes**

Find-by-Text displays **Descriptions**

Easy as *π*

Implementing vastly superior data entry support couldn't be easier. Just set your data validation as you normally would, load this add-in and switch it on. That's it!

Making the Magic Happen

The add-in finds all cells with data validation set to ***list***. When users select one of those cells, the add-in loads a ComboBox from the cell's validation formula and displays the ComboBox over the cell. When the user exits the ComboBox the add-in hides It keeping our data entry area clean.

Seeing is believing

Install the demo. Try it out. See how easy data entry can be. And if you like coding, open the VBE, examine the code, and use this guide to understand how to add feature rich ComboBoxes to your own projects without the add-in.

Installing the Demo

Requirements

This demo requires **MS Excel 2007®** or later. VBA (macros) must be enabled.

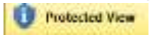
Installing

IMPORTANT! Follow these steps. Don't skip any.

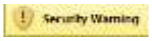
1. Download ComboBox.Zip
2. Open your **Downloads** folder, NOT the zip file.
3. Right click ComboBox.Zip and select *Extract All*.
4. Make sure *Show extracted files when complete* is checked.
5. Make note of the extracted folder's path.

Running

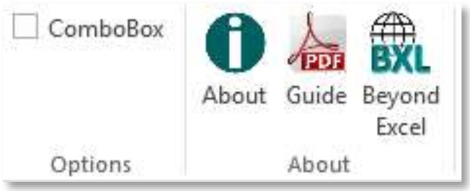
1. Open the extracted folder
2. Double click **AddInDemo.xls**
3. If *Protected View* displays click *Enable Editing*.



4. If *Security Warning* displays click *Enable Content*.



Find the ComboBox tab in the Ribbon and check ComboBox checkbox



ComboBox.zip's URL is:

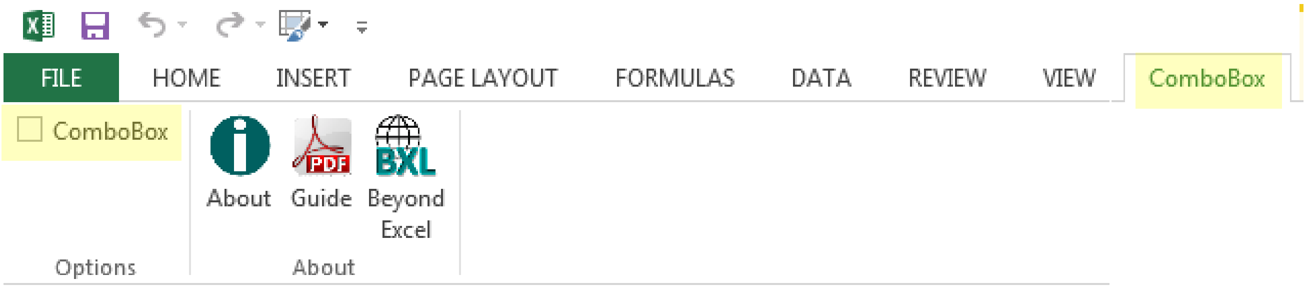
https://dl.dropboxusercontent.com/u/13737137/Projects/ComboBox/ComboBox.zip

Demo Walkthrough

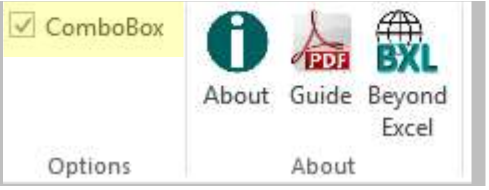
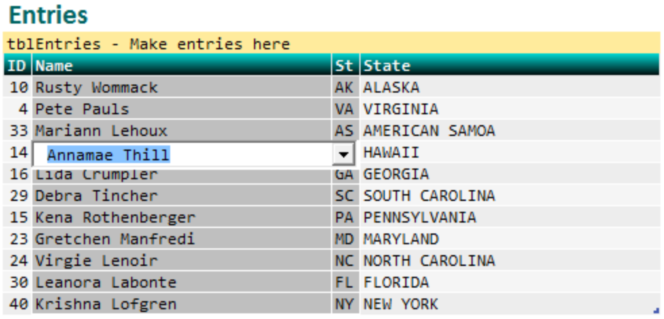
This section walks us through the demo so we can see what this app does.

Basic Concept

Set up data validation as we would normally (see Data Validation Tips for more on setting Excel's data validate) then check the Ribbon's ComboBox. At that point, all dropdown data validation uses a ComboBox instead of a dropdown listbox and a form to search for codes by description is accessible using function key F4.



Walkthrough

Instruction	Figure
1. Find the ComboBox tab and check the ComboBox checkbox.	
2. Click on a name. **NOTE**! The dropdown is now a ComboBox!	

Entries

tblEntries - Make entries here

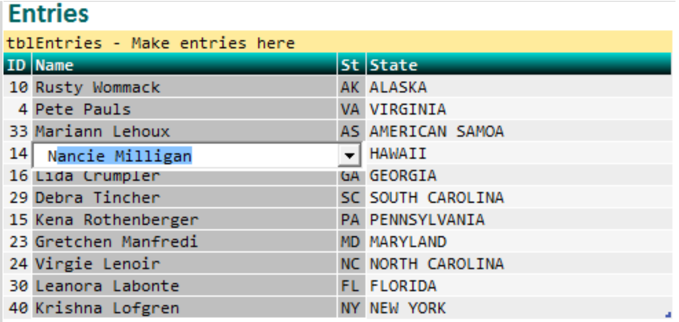
ID	Name	St	State
10	Rusty Wommack	AK	ALASKA
4	Pete Pauls	VA	VIRGINIA
33	Mariann Lehoux	AS	AMERICAN SAMOA
14	Annamae Thill		HAWAII
16	Lida Crumpler	GA	GEORGIA
29	Debra Tincher	SC	SOUTH CAROLINA
15	Kena Rothenberger	PA	PENNSYLVANIA
23	Gretchen Manfredi	MD	MARYLAND
24	Virgie Lenoir	NC	NORTH CAROLINA
30	Leanora Labonte	FL	FLORIDA
40	Krishna Lofgren	NY	NEW YORK

Instruction

Figure

3. Type an "N".

NOTE! Nancie's name appears. Dropdowns don't do that.



Entries

tblEntries - Make entries here

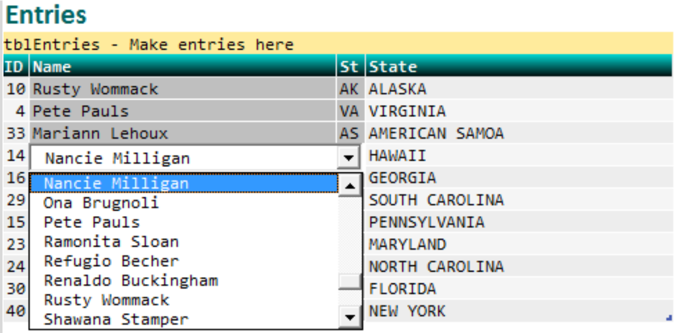
ID	Name	St	State
10	Rusty Wommack	AK	ALASKA
4	Pete Pauls	VA	VIRGINIA
33	Mariann Lehoux	AS	AMERICAN SAMOA
14	Nancie Milligan		HAWAII
16	Lida Crumpler	GA	GEORGIA
29	Debra Tincher	SC	SOUTH CAROLINA
15	Kena Rothenberger	PA	PENNSYLVANIA
23	Gretchen Manfredi	MD	MARYLAND
24	Virgie Lenoir	NC	NORTH CAROLINA
30	Leanora Labonte	FL	FLORIDA
40	Krishna Lofgren	NY	NEW YORK

4. Click the expansion triangle.

NOTE! The list positions to Nancie. Dropdowns don't do that.

As you can see, ComboBoxes make data entry where longer lists are involved much more manageable and efficient.

In this particular case, I decided to ask our data entry clerk to select by name and use a formula to retrieve the code. Now let's try an example where we ask our data entry clerk to enter a code.



Entries

tblEntries - Make entries here

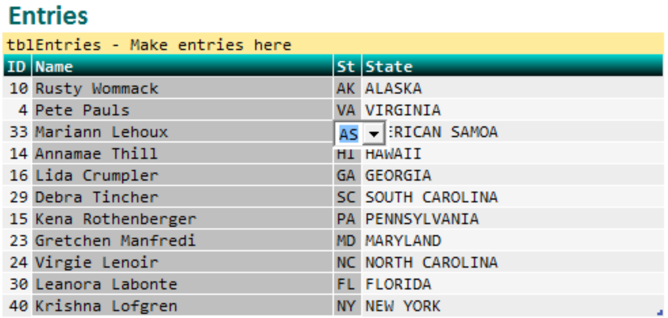
ID	Name	St	State
10	Rusty Wommack	AK	ALASKA
4	Pete Pauls	VA	VIRGINIA
33	Mariann Lehoux	AS	AMERICAN SAMOA
14	Nancie Milligan		HAWAII
16			GEORGIA
29			SOUTH CAROLINA
15			PENNSYLVANIA
23			MARYLAND
24			NORTH CAROLINA
30			FLORIDA
40			NEW YORK

Nancie Milligan
Ona Brugnoli
Pete Pauls
Ramonita Sloan
Refugio Becher
Renaldo Buckingham
Rusty Wommack
Shawana Stamper

5. Click one of the State codes (column **St**)

NOTE! This is the same ComboBox. The add-in moved it from our previous entry to this entry. It behaves exactly the same way no matter where the add-in positions it.

If you are like me, we sometimes get abbreviations for Alaska and Arkansas mixed up. No worries. This app helps us. See next.

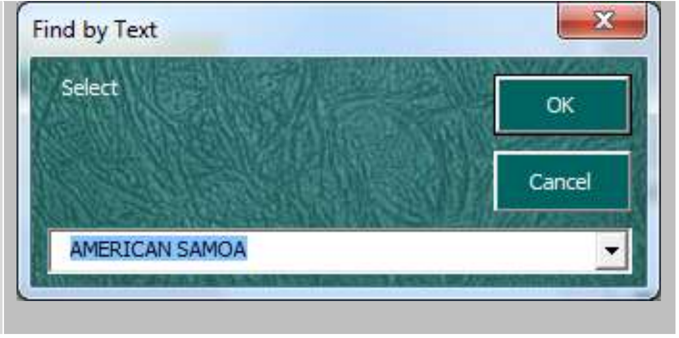


Entries

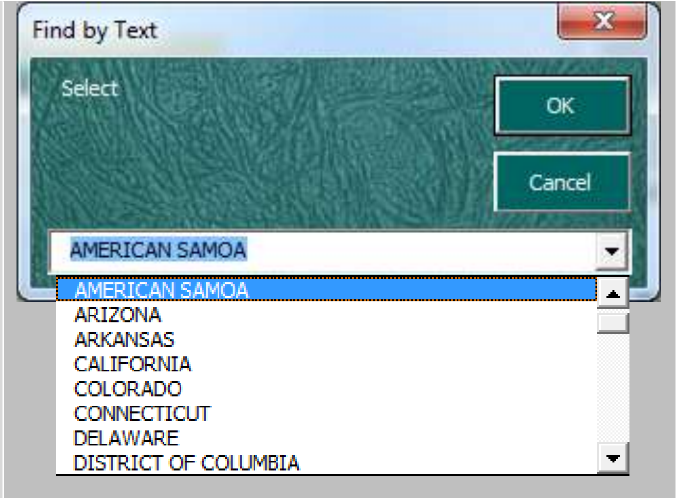
tblEntries - Make entries here

ID	Name	St	State
10	Rusty Wommack	AK	ALASKA
4	Pete Pauls	VA	VIRGINIA
33	Mariann Lehoux	AS	:RICAN SAMOA
14	Annamae Thill	HI	HAWAII
16	Lida Crumpler	GA	GEORGIA
29	Debra Tincher	SC	SOUTH CAROLINA
15	Kena Rothenberger	PA	PENNSYLVANIA
23	Gretchen Manfredi	MD	MARYLAND
24	Virgie Lenoir	NC	NORTH CAROLINA
30	Leanora Labonte	FL	FLORIDA
40	Krishna Lofgren	NY	NEW YORK

6. Press your computer keyboard's F4 function key to display this



AMERICAN SAMOA

Instruction	Figure
8. The Find-by-Text's ComboBox works just like all ComboBoxes. We can click the Expansion triangle to display a positioned list, or just start typing and it will autocomplete. Whatever we select from here is properly translated to our code entry. **NOTE**! This is useful also for data entry tables that, due to lack of screen real estate, display only the code. Data entry clerks can display the code's description, thus confirming it is the appropriate code, simply by selecting the code and pressing the F4 key.	 Find by Text Select OK Cancel AMERICAN SAMOA AMERICAN SAMOA ARIZONA ARKANSAS CALIFORNIA COLORADO CONNECTICUT DELAWARE DISTRICT OF COLUMBIA

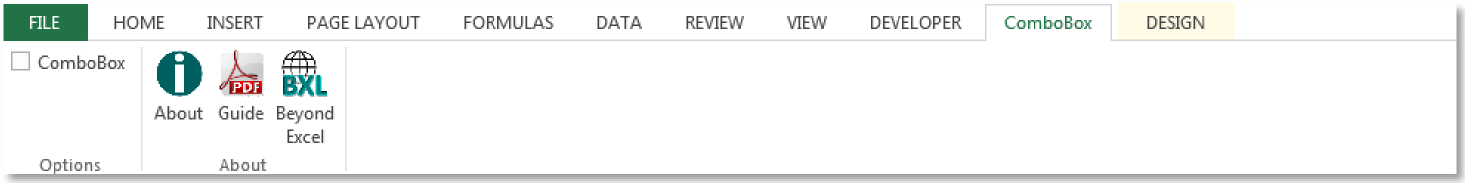
User Instructions

Using ComboBox is automatic. Simply select any cell with **Data Validation** set to **Allow *List*** and the add-in instantly swaps out the dropdown listbox for a ComboBox.

If we want the old dropdown's to return, simply uncheck the ComboBox checkbox in the ComboBox tab.

User Interface Elements

This add in installs its own tab on Excel's Ribbon. This is only automatic with the add-in (there is no menu tab when incorporating the modules directly into your project).



Ribbon Buttons

Button	Description
□ ComboBox ***On/Off Checkbox***	Turn ComboBox feature on or off. When checked all data validation dropdown lists are replaced with ComboBoxes for the current worksheet.
About	**About** this add in.
Guide	**Guide** (this document) display
BXL ***Website***	**Beyond Excel Website** display

Data Validation Tips

Here are some tips for setting up data validation for lists.

Tables – Glorious Tables!

Tables are not required; but, if you have not started using tables yet, you are in for a treat. They are marvelous. I particularly like them for **Data Validation** even though **Data Validation** is a bit clueless when it comes to tables.

Instruction

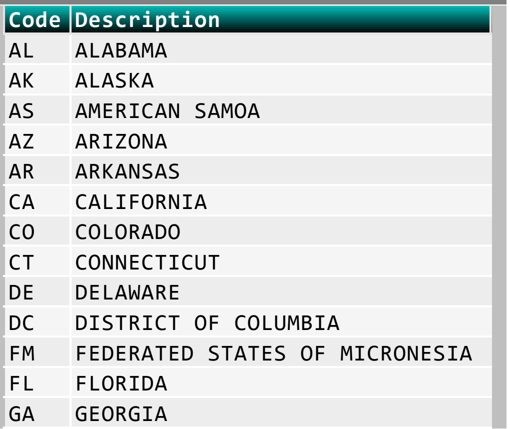
Figure

If you have XL2007 or later, you can take advantage of tables and tables' structured references. Tables are automatic dynamic named ranges. As they expand and contract all table references automatically expand and contract.

Setting up tables is easy. Enter your data in columns with column headings.

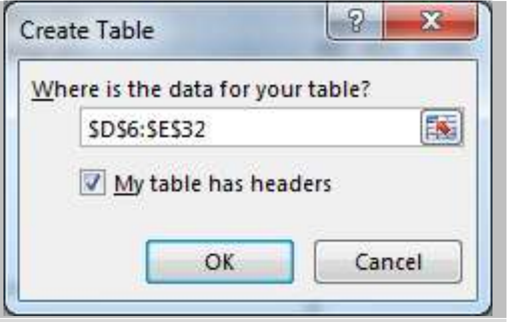
NOTE! Adding a "Description" column enables the **F4** *Find-by-Text* feature.

NOTE! It does not matter what the column headings are. The **F4** *Find-by-Text* feature assumes code descriptions are immediately right of the code column.



Code	Description
AL	ALABAMA
AK	ALASKA
AS	AMERICAN SAMOA
AZ	ARIZONA
AR	ARKANSAS
CA	CALIFORNIA
CO	COLORADO
CT	CONNECTICUT
DE	DELAWARE
DC	DISTRICT OF COLUMBIA
FM	FEDERATED STATES OF MICRONESIA
FL	FLORIDA
GA	GEORGIA

Click anywhere inside the list then press **CTRL-T**. The **Create Table** dialog appears. Make sure the **Create Table** dialog has properly assumed you table's range, check **My table has headers**, then click **OK**



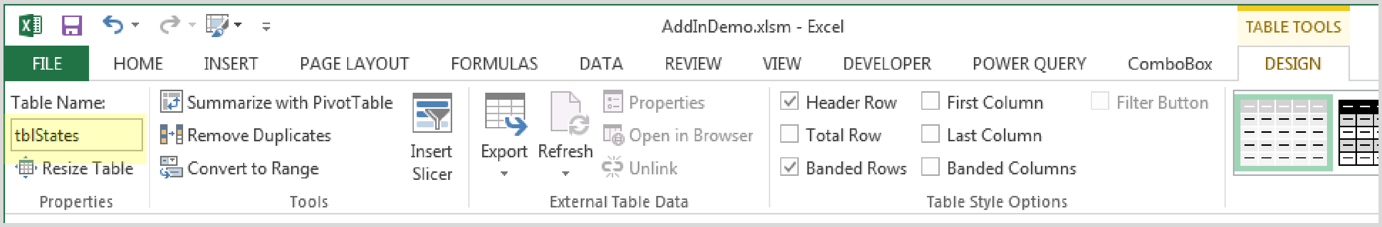
Create Table

Where is the data for your table?

$D$6:$E$32

☑ My table has headers

A new ribbon tab appears, **Table Tools**. On the far left of this tab is the **Table Name** box. Change the name to something meaningful. Personally, I like to name my tables with a "tbl" prefix, but that's totally a matter or preference.



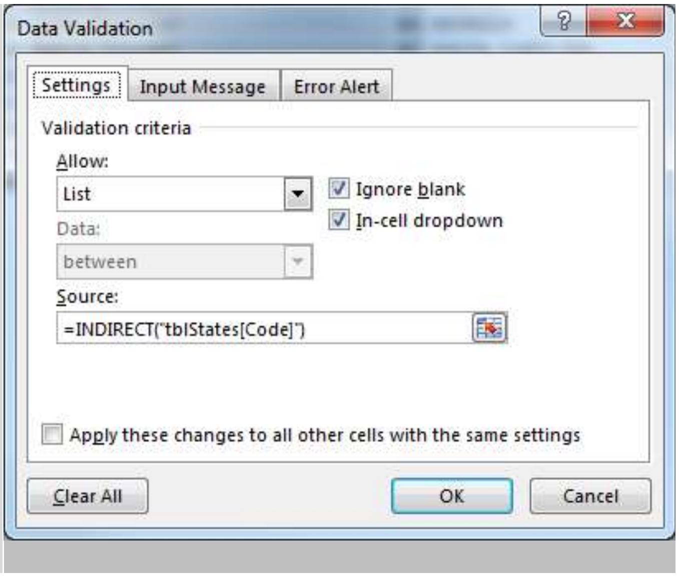
AddInDemo.xlsm - Excel

Table Name: tblStates

☑ Header Row
☐ Total Row
☑ Banded Rows
☐ First Column
☐ Last Column
☐ Banded Columns
☐ Filter Button

We can now use our **Table** in **Data Validation** without having to know precisely where it is, or how large it is, and never having to worry about changes to either position or size. There is just one drawback.

Excel's **Tables** feature is newer than Excel's **Data Validation** feature and, so, Excel's **Data Validation** needs some assistance from us to use tables and tables' structured references. All we need to do is surround our table reference with the old Excel function **INDIRECT**. **INDIRECT** converts our table reference to a reference **Data Validation** understands.



Allow: List

☑ Ignore blank

☑ In-cell dropdown

Data: between

Source: =INDIRECT("tblStates[Code]")

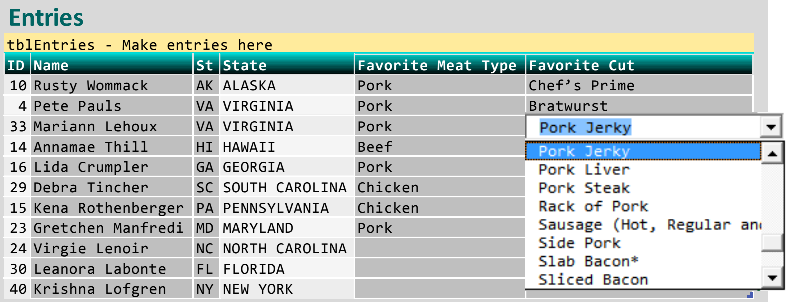
□ Apply these changes to all other cells with the same settings

Sort Carefully

Sort by how people search. I sorted **tblNames** by the **Name** column because data entry clerks will search by **Name**. They won't search by **ID**. I sorted the **tblStates** by **Code** because people who use such tables expect them sorted that way.

Dependent Lists

We often find data element choices depends on earlier choices. As an example **AddInDemo.xlsm** has two columns: **Favorite Meat Type**, and **Favorite Cut**. If users select **Pork**, under **Favorite Meat Type** we want to display only Pork cuts in **Favorite Cut.**



Entries

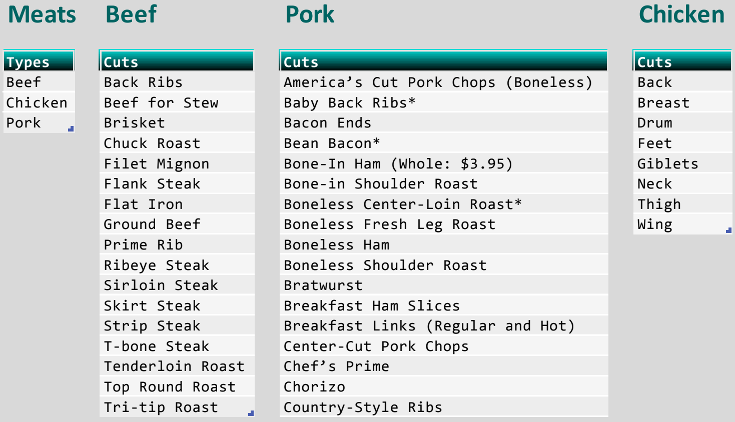
tblEntries - Make entries here

ID	Name	St	State	Favorite Meat Type	Favorite Cut
10	Rusty Wommack	AK	ALASKA	Pork	Chef's Prime
4	Pete Pauls	VA	VIRGINIA	Pork	Bratwurst
33	Mariann Lehoux	VA	VIRGINIA	Pork	Pork Jerky
14	Annamae Thill	HI	HAWAII	Beef	
16	Lida Crumpler	GA	GEORGIA	Pork	
29	Debra Tincher	SC	SOUTH CAROLINA	Chicken	
15	Kena Rothenberger	PA	PENNSYLVANIA	Chicken	
23	Gretchen Manfredi	MD	MARYLAND	Pork	
24	Virgie Lenoir	NC	NORTH CAROLINA		
30	Leanora Labonte	FL	FLORIDA		
40	Krishna Lofgren	NY	NEW YORK		

Pork Jerky
Pork Liver
Pork Steak
Rack of Pork
Sausage (Hot, Regular an
Side Pork
Slab Bacon*
Sliced Bacon

The easiest way to address dependent lists is to name dependent lists using the primary list's values. At right we have a primary list, **Meats**. **Meats** contains three values: *Beef*, *Chicken* and *Pork*. We also have three corresponding dependent lists: **Beef**, **Pork**, and **Chicken**.

NOTE! I like to name tables with the prefix "tbl". Thus these tables are named: **tblMeats**, **tblBeef**, **tblPork**, and **tblChicken**



Meats

Types
Beef
Chicken
Pork

Beef

Cuts
Back Ribs
Beef for Stew
Brisket
Chuck Roast
Filet Mignon
Flank Steak
Flat Iron
Ground Beef
Prime Rib
Ribeye Steak
Sirloin Steak
Skirt Steak
Strip Steak
T-bone Steak
Tenderloin Roast
Top Round Roast
Tri-tip Roast

Pork

Cuts
America's Cut Pork Chops (Boneless)
Baby Back Ribs*
Bacon Ends
Bean Bacon*
Bone-In Ham (Whole: $3.95)
Bone-in Shoulder Roast
Boneless Center-Loin Roast*
Boneless Fresh Leg Roast
Boneless Ham
Boneless Shoulder Roast
Bratwurst
Breakfast Ham Slices
Breakfast Links (Regular and Hot)
Center-Cut Pork Chops
Chef's Prime
Chorizo
Country-Style Ribs

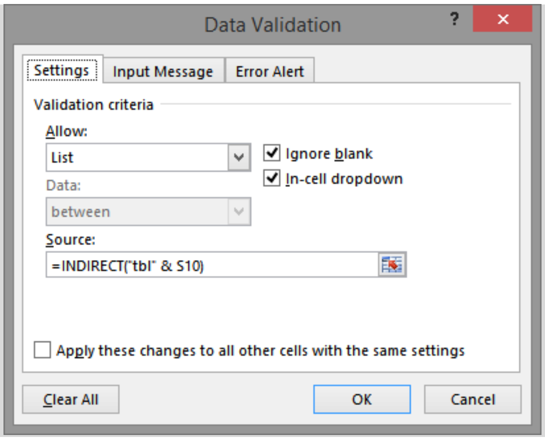
Chicken

Cuts
Back
Breast
Drum
Feet
Giblets
Neck
Thigh
Wing

To set data validation to display dependent dropdowns we use **INDIRECT** as shown at right. In English, the Source formula says:

Add *tbl* to the value in cell **S10** and convert that name to an address.

If cell **S10** contains *Pork*, the formula will read **=INDIRECT("tblPork")**. **tblPork** contains our Pork Cuts list and that will be loaded into our dropdown.



Validation criteria

Allow:
List

☑ Ignore blank

☑ In-cell dropdown

Data:
between

Source:
=INDIRECT("tbl" & S10)

□ Apply these changes to all other cells with the same settings

NOTE! In this example I opened the Data Validation dialog with cell **T10** selected; thus, **S10** is the cell immediately left of cell **T10**.

NOTE! The absence of **$** make **S10** a *relative reference* instead of an *absolute reference*. As a relative reference Excel knows we don't mean cell **S10**, we mean the cell left of **T10**. If we then copy this validation rule to other cells, the **Source** formula will always point to the cell immediately left of the cell containing this validation rule.

NOTE! I normally don't use cell references. I normally use structured references. Instead of cell address **S10** we could use the structured reference **tblEntries[@[Favorite Meat Type]]** except there is a problem with Excel's **Evaluate** function in VBA and this specific type of structured reference. The **ComboBox Add-in** uses Excel's **Evaluate** function and so we cannot use table row structured references with this add-in at this time.

Technical Documentation

The following section is for VBA Developers only. It is mind-numbing to all others.

You do not need to read ANY of this section if you just want to use this feature. If you are not interested in how the code works, skip this.

Adding ComboBox to Workbooks

Load and play with **AddInDemo.xls.** This automatically installs the add-in into your PC and adds it to the **Add-Ins** dialog box (see below right). From this point on, you can add it to your projects by following these simple steps:

1. Use the Add-Ins dialog and place a check next to **ComboBox** (see right).

   -OR-

2. Add the following code to your **ThisWorkbook** module to automatically load and unload **ComboBox** with your project.

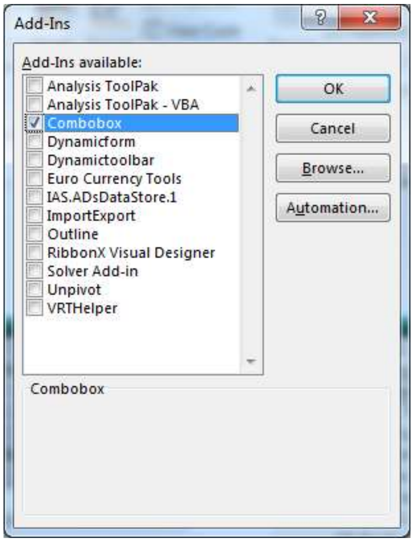
```
    Private Sub Workbook_Open()
        Application.AddIns("ComboBox").Installed = True
    End Sub

    Private Sub Workbook_BeforeClose(Cancel As Boolean)
        Application.AddIns("ComboBox").Installed = False
    End Sub
```

   -OR-

3. Drag and drop modules **clsCBO**, **modGeneral**, **clsSettings** and **modAddIn** from **ComboBox.xlam** to your project

   When either of these options is completed, **ComboBox** is ready for use with your XL apps.



Add-Ins available:

☐ Analysis ToolPak
☐ Analysis ToolPak - VBA
☑ Combobox
☐ Dynamicform
☐ Dynamictoolbar
☐ Euro Currency Tools
☐ IAS.ADsDataStore.1
☐ ImportExport
☐ Outline
☐ RibbonX Visual Designer
☐ Solver Add-in
☐ Unpivot
☐ VRTHelper

Combobox

Bypassing the Add-In

clsCBO (the main class module) is NOT limited to replacing data validation dropdown list boxes. **clsCBO** was created as part of a package to make custom data validation possible, custom data validation that far exceeds the limits of Excel's data validation. This add-in is merely a means to provide **clsCBO's** functionality as easily as possible. To get the most out of **clsCBO** we should incorporate this apps components directly into our projects.

Components

We can incorporate this functionality directly into our projects to simplify distribution by packaging everything into a single workbook requiring no external add-ins. The components we must include are:

modGeneral	BXL's general purpose function library
frmInputBox	BXL's InputBox replacement used for Find-by-Text functionality
clsForm	BXL's Userform formatting class (for frmInputBox)
clsCBO	BXL's ComboBox class used to setup, fill, and display the ComboBox

The other add-in components are only used by the add-in. With the components listed above loaded into our workbook, we need to write just a few lines of code to implement the class.

Description	Code
Create a variable to hold the class	`Dim oMyClass As clsCBO`
Create a new instance of the class	`Set oMyClass = New clsCbo`
Set the class's worksheet	`Set oMyClass.Worksheet = ActiveSheet`

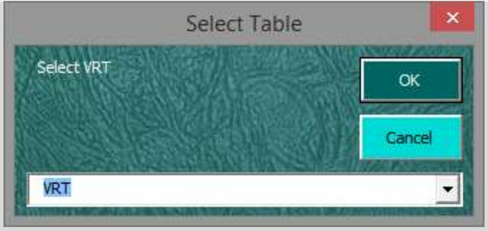
Save the class to the class server. The class server will keep the class instance in memory so it can respond to events.	`modGeneral.SavCls _`
	`    sSaveAs:="ComboBox:" & Thisworkbook.Name & "!" & ActiveSheet.Name, _`
	`    oClass:=oMyClass`

NOTE! Each component's coding contains its own documentation. See the source for a more detailed description of what is happening within each routine. I believe documenting the ***intent*** of each block of code to be beneficial. I believe documenting each line of a code's ***function*** is of no value. It clutters code making it harder to understand. If a coder runs across an XL object or a VBA function they do not understand, Googling the XL object or VBA function with the added keyword "MSDN" will provide a much better and complete understanding than anything I could write.

Modules

Component	Description
modGeneral	This is BXL's general purpose library of routines common to all BXL apps. Because this is a general purpose library, please see the module for documentation on each routine. Not all are used in this project.

Forms

Form	Description
frmInputBox	This a general purpose, formatted InputBox. 

Classes (Event Handlers)

<table>
<tr><th>Component</th><th>Description</th></tr>
<tr><td><em><strong>clsForm</strong></em></td><td><strong>clsForm</strong> provides glowing effects and other user form formatting.
For more on this see:
https://dl.dropboxusercontent.com/u/13737137/Projects/Hover/FormColorClass.pdf
 </td></tr>
<tr><td><em><strong>clsCBO</strong></em></td><td>
<table>
<tr><th>Public Properties</th><th>Description</th></tr>
<tr><td><em><strong>Worksheet</strong></em>
(Write Only)</td><td>If we do not want to use Setup we can use this to tell the class which worksheet to monitor. It also adds a ComboBox to the worksheet if one has not already been setup for it.
Here is an example code snippet setting this property.<pre>
Public Sub InstantiateClass()
    Set oMyClass = New clsCbo
    Set oMyClass.Worksheet = ActiveSheet
End Sub
</pre></td></tr>
<tr><td><em><strong>ComboBox</strong></em>
(Write Only)</td><td>If we do not want to use Setup or Set Worksheet we can use this to assign an existing ComboBox to the class.</td></tr>
<tr><td><em><strong>AutoReplaceValidationDropdowns</strong></em>
(Write Only)</td><td>If we do not want to use Setup we can set this to TRUE to instruct the class to take over all dropdown data validation.
Here is an example code snippet setting this property.<pre>
Public Sub InstantiateClass()
    Set oMyClass = New clsCbo
    Set oMyClass.Worksheet = ActiveSheet
    Set oMyClass.AutoReplaceValidationDropdowns = TRUE
End Sub
</pre></td></tr>
<tr><td><em><strong>Value</strong></em>
(Read Only)</td><td>Use this to manually retrieve the ComboBox’s current value</td></tr>
<tr><td><em><strong>Cell</strong></em>
(Read Only)</td><td>Use this to manually set the cell that the ComboBox is for</td></tr>
<tr><td><em><strong>Visible</strong></em>
(Read Only)</td><td>Use this to manually make the ComboBox visible</td></tr>
</table>
</td></tr>
</table>

Component	Description

Private Event Handlers	Description
Class_Initialize	Turns on AutoReplaceValidationDropdown feature
Class_Terminate	Makes sure the ComboBox is hidden when the class terminates
oWks_SelectionChange	Monitors the worksheet for cell selection and sets up the ComboBox if the activecell uses Allow List Data Validation
oCbo_KeyDown	Monitors key presses within the ComboBox to determine if the user wants to exit the ComboBox, and in which direction, or if the user wants to use the *Find-By-Text* feature

Public Methods	Description

Setup	Fills a ComboBox with items and displays it over the target cell
	`Public Sub InstantiateClass()`
	`    Set oMyClass = New clsCbo`
	`    oMyClass.Setup “A,B,C,D”, Range(“A1”)`
	`End Sub`
	NOTE! This method automatically sets the **Worksheet** property from the target cell.
	NOTE! If we omit the target range parameter, **Setup** defaults to **ActiveCell**.
	NOTE! See Fill(vList) for notes on **vList**.

Clear	Use this to manually clear the ComboBox list.
Fill	Use this to manually Fill the ComboBox list NOTE! **vList** can be a comma delimited list, an array, a range, or one of several special names: CHARTS All Charts in **oTarget’s** workbook NAMES All Names (excludes table names) in **oTarget’s** workbook PIVOTTABLES All Pivot Tables in **oTarget’s** workbook QUERYTABLES All Query Tables in **oTarget’s** workbook SHAPES All Shapes in **oTarget’s** workbook STYLES All Styles in **oTarget’s** workbook PIVOTTABLESTYLES All Pivot Table Styles in **oTarget’s** workbook TABLES AND NAMES All Names with Table Names in **oTarget’s** workbook WORKBOOKS All currently open workbooks WORKSHEETS All worksheets in **oTarget’s** workbook
Show	Use this to manually display the ComboBox
Hide	Use this to manually hide the ComboBox

<table>
<tr><th>Component</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Private Functions</th><th>Description</th></tr>
<tr><td><strong><em>AddOLEObject</em></strong></td><td>Adds an ActiveX ComboBox to the worksheet in a way that does not reset the Visual Basic Project.</td></tr>
<tr><td><strong><em>FindByDesc</em></strong></td><td>Displays the <em>Find-by-Text</em> feature</td></tr>
</table>
</td></tr>
</table>

Appendix

Jargon and Terms

Topic	Description
Structured References	https://support.office.com/en-us/article/Using-structured-references-with-Excel-tables-f5ed2452-2337-4f71-bed3-c8ae6d2b276e

Additional References and Documentation

Subject	Description	
BXL Error Handling	This explains how BXL apps handle errors including theory and best practices. Click image to download or go to: https://dl.dropboxusercontent.com/u/13737137/Starters/Error_Handling.pdf	Beyond Excel
BXL Coding Best Practices	This explains some of BXL's philosophies regarding development in VBA. https://dl.dropboxusercontent.com/u/13737137/Starters/Coding%20Best%20Practices.pdf	Beyond Excel