## Building a Web App with Access Services 2013

**Lab Time**: 90 minutes

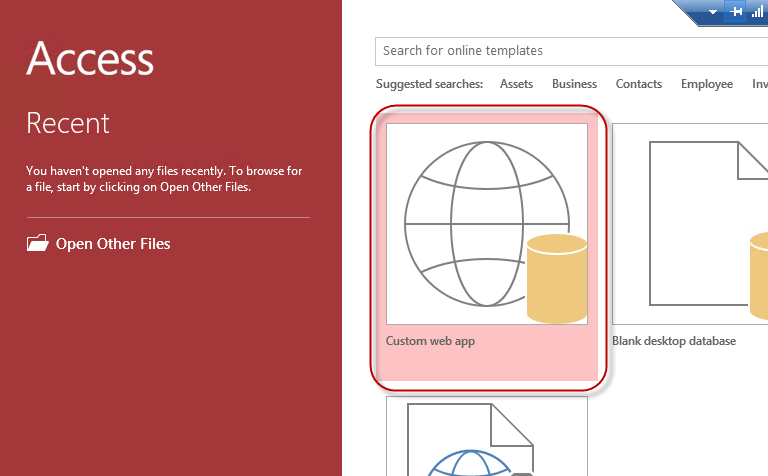
**Lab Folder**: C:\Student\Modules\AccessServices\Lab

**Lab Overview**: In this lab you will create a lightweight order entry system for Wingtip Toys using Access Services.

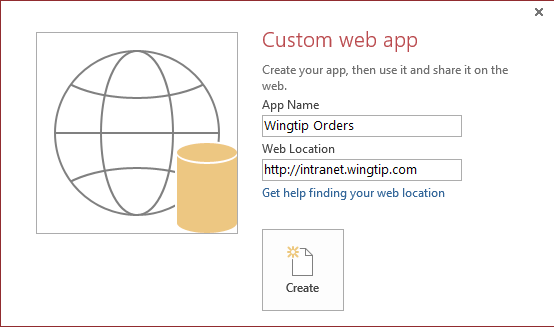
### Exercise 1: Creating a Custom Web App with Access Services

In this exercise you will create an Access Service Web Application that will enable users to enter orders for Wingtip Toys Products. In this exercise you will create a new “Custom web app” with Access 2013 and connect the app to the Wingtip Intranet. Adding this app to the Wingtip Intranet will enable other users to create and manage orders from SharePoint 2013.

1. Login to the Student VM using the login **WINGTIP\Administrator** and the appropriate password.
   1. If you’re using a local VM provided by the hosting training company, the password will be Password1.
   2. If your student VM is hosted by CloudShare, the password for the WINGTIP\Administrator account is going to be unique for each student, system-generated by CloudShare. Also note that the CloudShare VM configuration usually logs you into the VM automatically so you do not have to enter the user name and password.
2. From the Start Menu, launch **Access 2013**.
3. Chose Custom web app from the menu of choices for new projects.



1. In the Custom web app dialog enter the following information:
   1. App Name: **Wingtip Orders**
   2. Web Location: **http://intranet.wingtip.com**



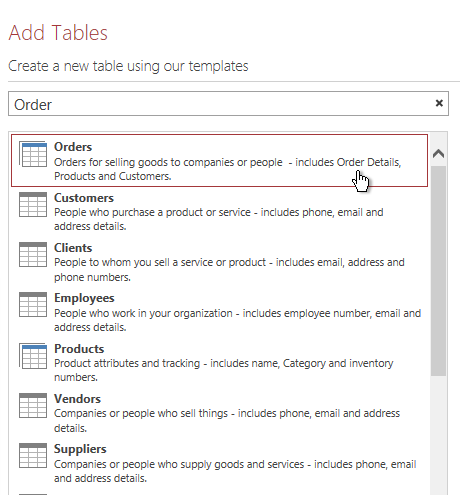
1. Click **Create**.

At this point you should be left on the **Add Tables** page of the **Wingtip Orders** tab in Access 2013. The initial steps to create the Custom web app in Access Services created a database for your project in Microsoft SQL 2012. In the next exercise you will add the tables needed to track your orders.

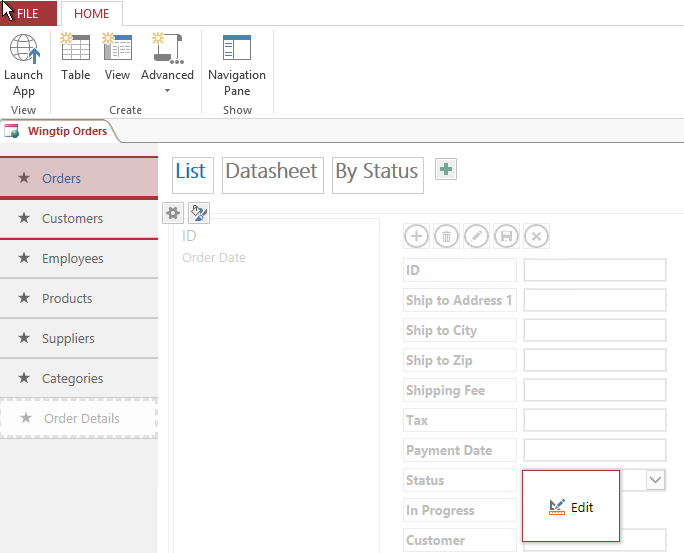
### Exercise 2: Creating Tables and Queries

In this exercise you will add the tables and queries to the App to track orders for Wingtip Toys. In this exercise you will use Microsoft Access 2013 to provision the necessary Customers, Orders, Items and Inventory tables for the orders application.

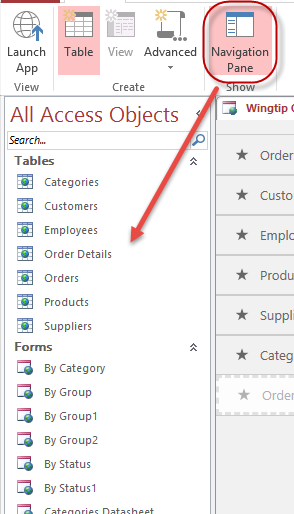
1. In the search box on the Add Tables page in Access 2013 enter the word Orders and press enter.
2. From the results list choose **Orders** and let Access build the associated tables.



1. Once the tables have been created you can browse the results and the default views by selecting the tables on the tabs to the left of the design surface.



1. You can view every object that was created for you by Access 2013 in the Navigation Pane. In the ribbon click **Navigation Pane**. This will show the All Access Objects navigation pane and help when you need to search for an object.



1. Click **Save** and then **Launch App**. A browser should open to the Orders Page of the App. There is no data in the App yet, but you should be able to browse around the App and see how sophisticated this app is even though it only took you a minute to build it.
2. Working with data driven apps is easier if you have data, so take a moment and add a few records to the following tables:

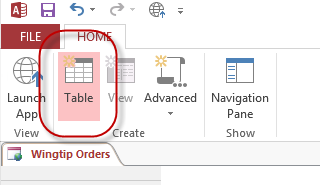
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Customers** | | | | | |
| First Name | Last Name | Company | City | State | Group |
| Bill | Gates | Microsoft | Seattle | WA | Business |
| Matthew | McDermott | Aptillon | Austin | TX | Family |
| Ted | Pattison | Critical Path Training | Tampa | FL | Business |

You have just created a stand-alone App in Access Services. This app is entirely contained within a single SQL Server Database and has no connection to the data from previous labs in class. While this may be a viable option for some solutions, it is not enough for us at Wingtip Toys. In our environment the real product numbers are maintained in SharePoint and need to be used in the Access App for validation. In the next steps you will create a data connection to the SharePoint Products list and then use that lookup to validate the correct Part Number and Image.

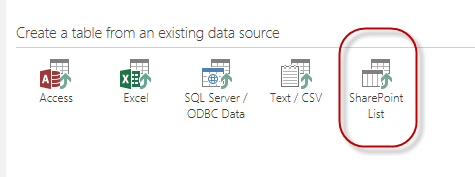
### Exercise 3: Connecting to SharePoint Lists

In this exercise you will create a lookup to a SharePoint list of Products that you created in previous labs. This lookup will ensure that only valid Product ID numbers are entered in the Product Inventory Table of the App.

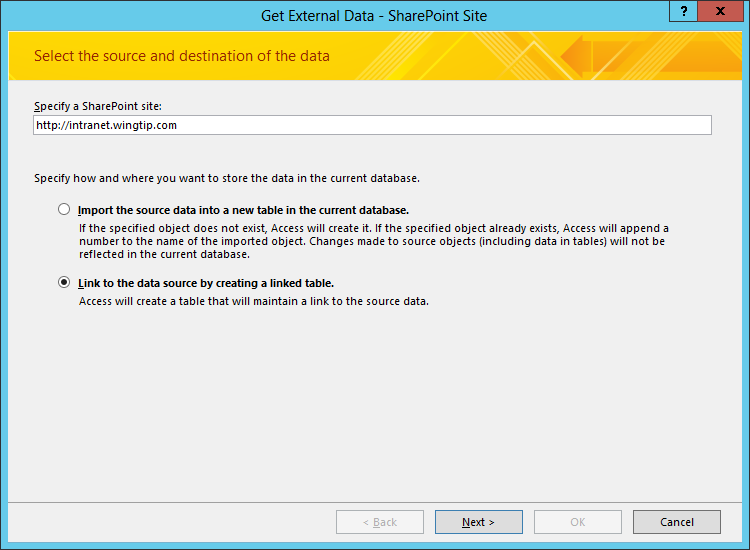
1. Return to Access 2013 and the Wingtip Orders tab. From the ribbon click **Create | Table**.



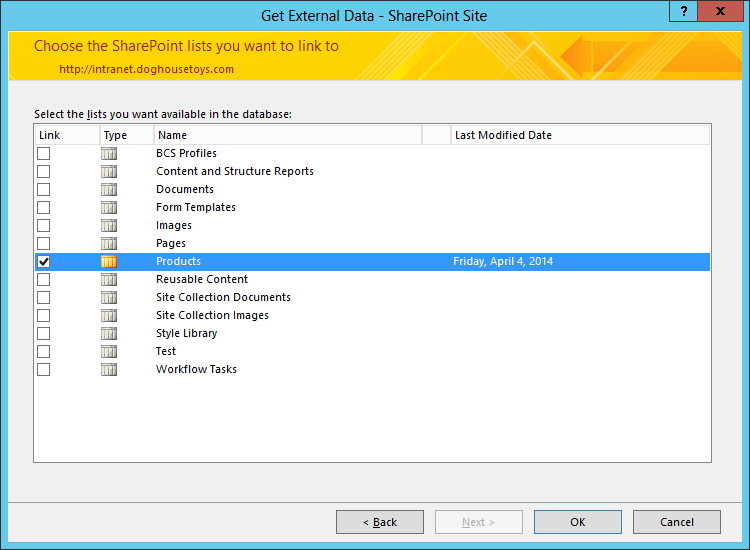
1. On the Add Tables page choose SharePoint List from the data sources at the bottom of the page.



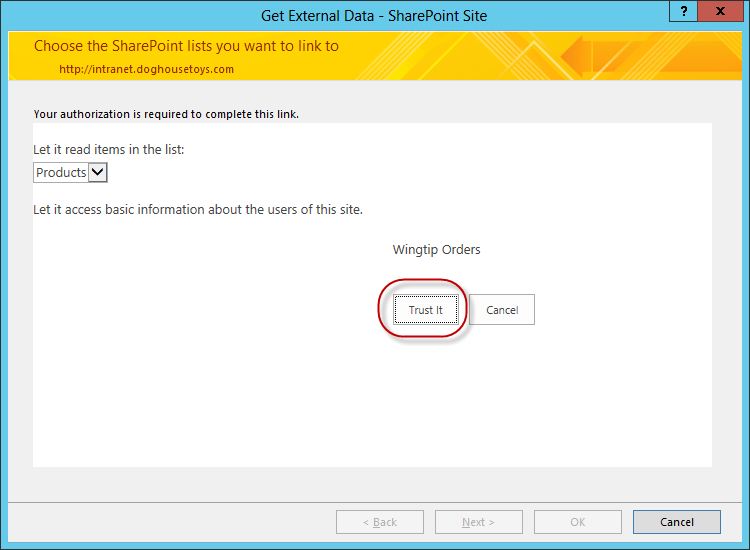
1. In the Get External Data dialog enter the address **http://intranet.wingtip.com**. Select **Link to the data source by creating a linked table**. Click **Next**.



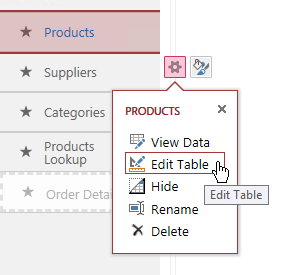
1. Choose the **Products** list and click **OK**.



1. In the Authorization dialog click **Trust It**.

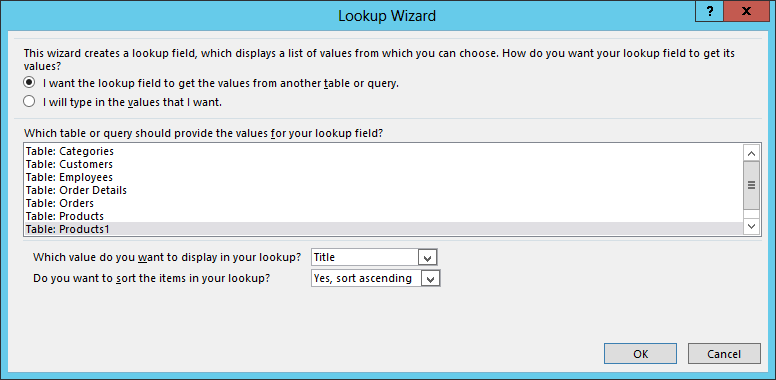


1. When the process completes you should now have a new table called **Products1**. Right click the table in the left navigation and choose **Rename**. Rename the table to **Products Lookup**.
2. Select the **Products** table. Right click and choose **Edit Table**.

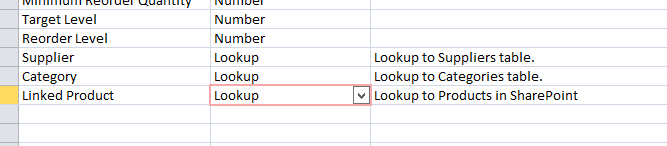


1. On the Table Design surface click after the last field and enter **Linked Product**. Change the field type to **Lookup**. In the **Lookup Wizard** select **I want the lookup field to get values from another table or query.**

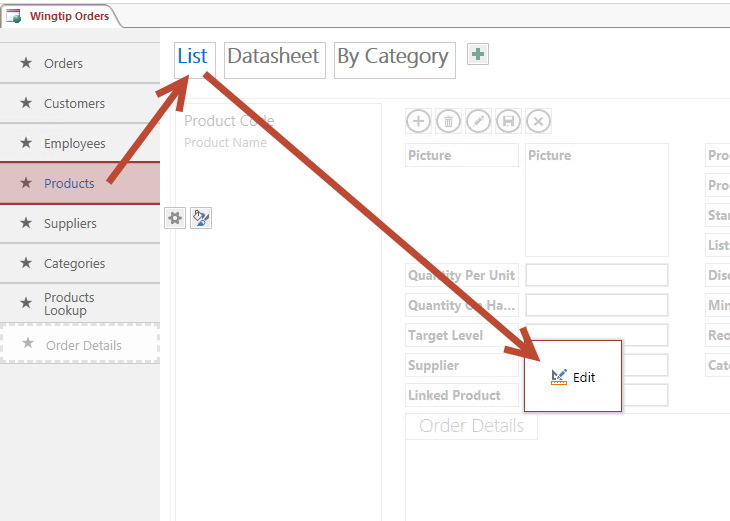
* Choose the **Products1** table
* Choose the **Title** field for display



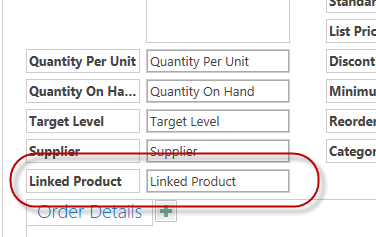
1. Click **OK**.
2. Change the Description column to **Lookup to Products in SharePoint**.



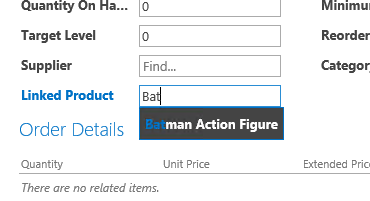
1. Click the **Save** button to save your changes. Close the **Products** tab.
2. Return to the **Products** table, choose the **List** view and click **Edit**.



1. Notice that Access automatically added the **Linked Product** field to the List view for you.



1. Close the Products List View and choose **Launch App** from the ribbon.
2. Choose the **Products** table and click in the **Linked Product** field. Type **“Bat”** and you should see that the field is looking up the data from your SharePoint list of Products.



1. Take a moment now and add a couple records to the table. The data in the **Discontinued** field will be used in a later exercise.

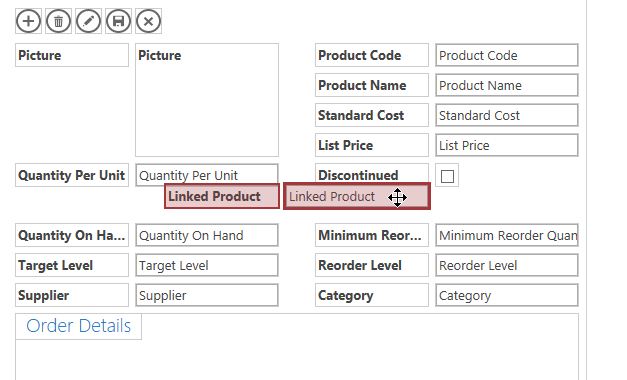
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Products** | | | | | |
| Linked Product | Product Code | Product Name | Standard Cost | List Price | Discontinued |
| Batman Action Figure | 10001 | Batman | $15.00 | $30.00 | Unchecked |
| Spiderman Action Figure | 10002 | Spiderman | $12.00 | $24.00 | Checked |
| Green Hornet | 10003 | Green Hornet | $10.00 | $20.00 | Unchecked |

In this Exercise you added a linked field to a SharePoint list. These links are read-only and enable you to manage content both in and outside of Access Services. Once the field was added to the table it was automatically added to the corresponding form. In the next exercise you will modify the forms to work the way you need to complete data entry.

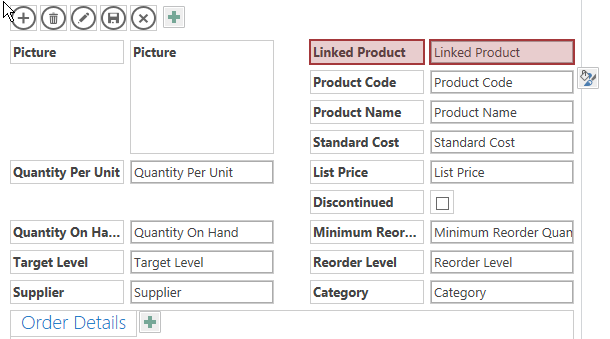
### Exercise 3: Designing Views with Access Services

In this exercise you will learn to modify the Access views that were automatically created while you were building your App. You will learn how to add and remove controls and how the designer works to keep everything looking great.

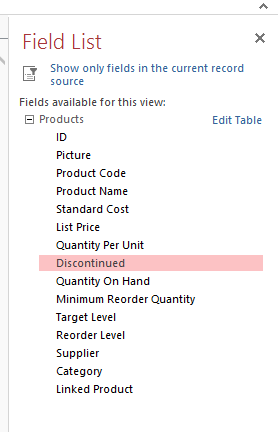
1. Return to the **Access Client** and click the **Products table** and **List view**.
2. Click **Edit** to begin editing the view.
3. Click the Linked Product label and then hold the [CTRL] key and click the Linked Product field to select them both. Drag them both to the new space created above the Product Code field. Notice how the fields move out of the way as you drag the fields together.



1. Once you drop the fields they should “snap” into the right location.



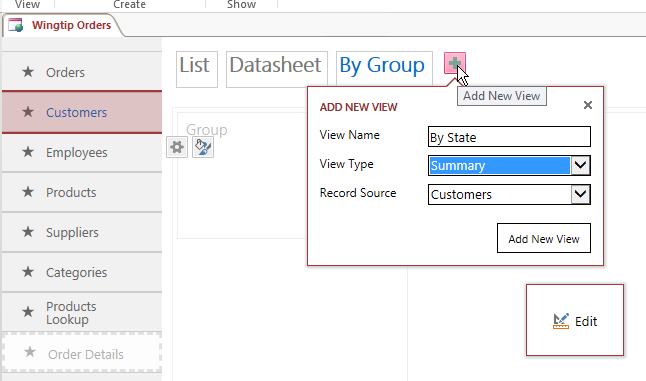
1. You can experiment with moving fields in the designer. If, for example, if you select the **Discontinued** field and press **Delete** it will be removed from the view, but not the table. To add it back, locate Discontinued in the field list and **double click** it. This will return the field to the form along with the label.



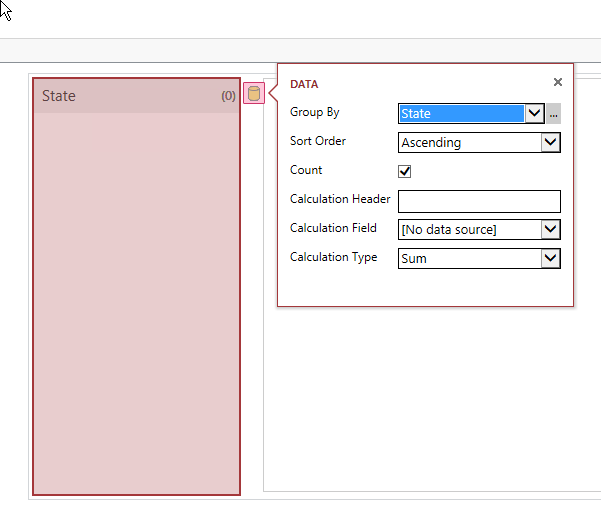
1. You can now drag the discontinued field to the bottom of the right column of the view. Once you are done editing the view, click **Save**.

#### Create a new View

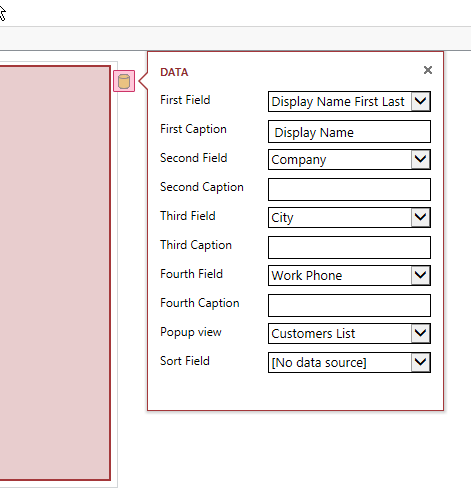
1. Click on the **Customers** table.
2. Click the **green plus mark** to create a **new View**.



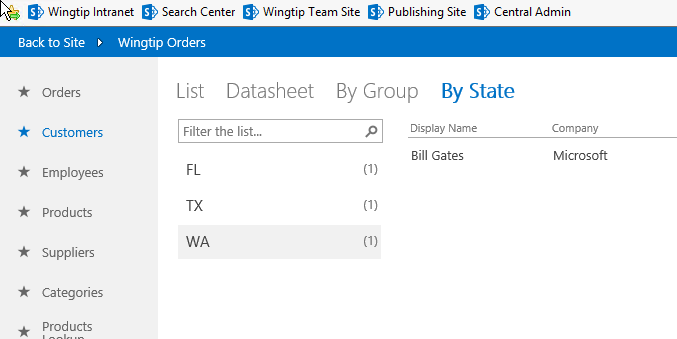
1. Name the view **By State** and choose the **Summary View Type**. Click **Add New View**.
2. Once the view is created click **Edit**.
3. Right click in the left pane titled **Group** and change the Group By field to **State**.



1. Right click the right pane and change the properties to display the third field as **City**.



1. Click **Save** to publish your work back to Access Services.
2. Click **Launch App** from the View section of the Home tab.
3. Click the **Customers** table and the **By State** view. You should now see a state group for each state you entered earlier. Clicking the **State** will show the Customers in that state.



In this exercise you worked with the view editing tools to change and create Access Services views. Views are the visual parts of the App that you are creating for your users. In the next exercise you will apply macros to the views and data to complete the app.

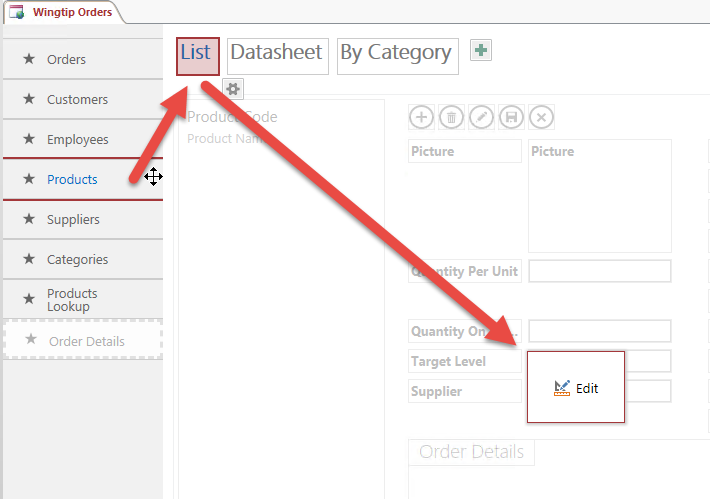
### Exercise 4: Working with Macros in Access Services

In this exercise you will work with UI Macros and Data Macros to learn to automate tasks and validate data. UI Macros run in the browser and can be used to provide additional information to the user. Data macros run behind the scenes to help maintain and validate the data in an Access Services App.

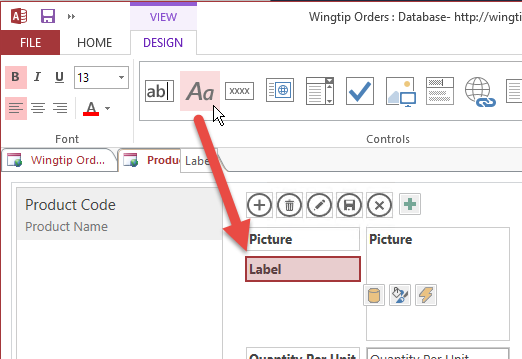
In this scenario there is a field on the Product record titled Discontinued. Users are not paying attention to the field when they place their orders. It is your job to provide feedback to the users visually when the field is set to indicate that the item is discontinued. Secondly, you need to prevent users from adding products to orders that are discontinued. Your approach is going to be to create a **UI Macro** to flag the item that is discontinued visually. To prevent the item from being added to an order you are going to create a **Data Macro** so that no matter how the user tried to add the item (from any view) the app will prevent the addition of discontinued items.

#### Create a User Interface Macro

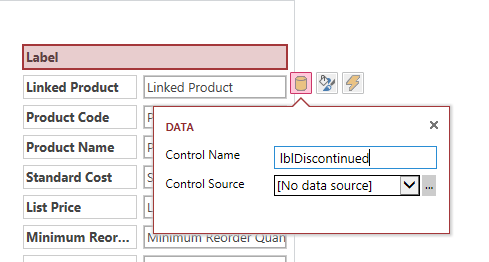
1. Return to **Access 2013** and click the **Products** table and the **List** view. Click **Edit** to begin editing the view.



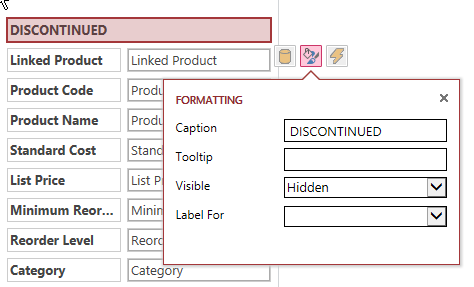
1. On the **Design** tab click the **Label** control to add it to the view.



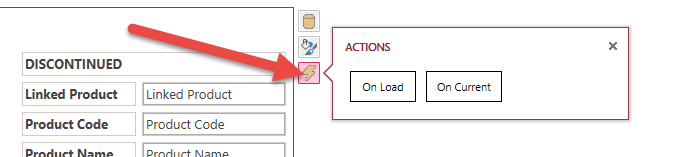
1. Drag the label above the **Linked Product** field and resize it to the width of the column.
2. Set the **Control Name** of the control to **lblDiscontinued**.



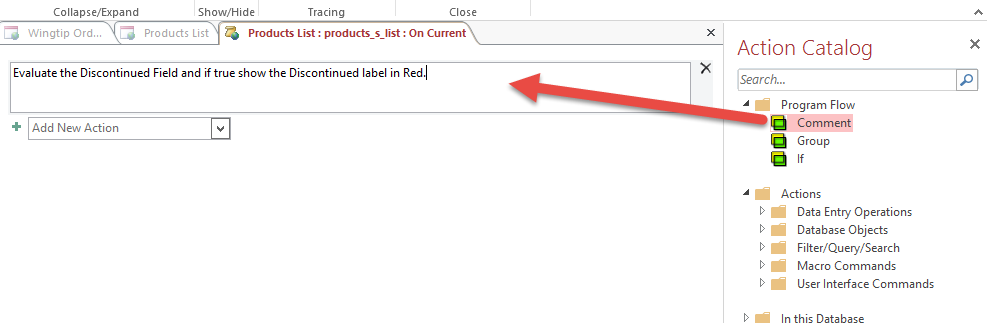
1. Set the **Formatting** of the **Caption** to **DISCONTINUED** and the **Visible** property to **Hidden**.



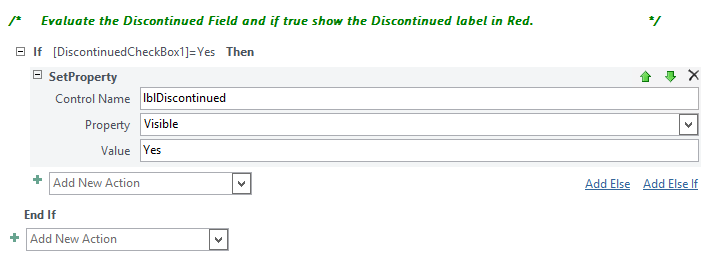
1. **Save** the view.
2. **Right click** in the **white space above** the **Discontinued** label and click the Actions button, choose **On Current** from the flyout menu. (If you don’t see the On Current Button ensure that you right clicked in the right place.)



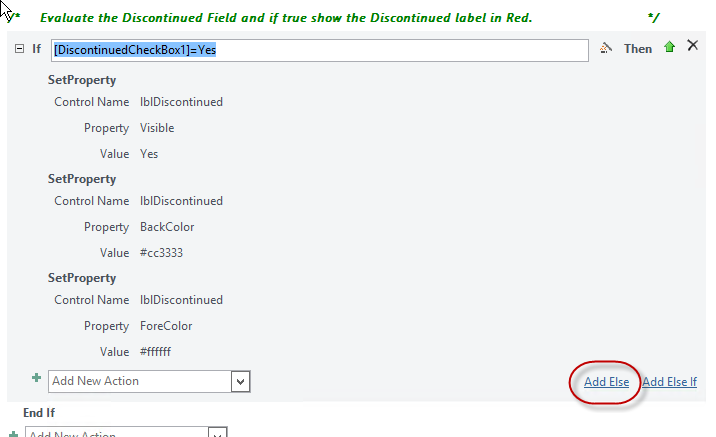
1. In the macro designer you will add the logic to evaluate the current record and trigger the visibility and formatting of the lblDiscontinued label. From the Action Catalog on the right, choose Comment. In the text field type: **Evaluate the Discontinued Field and if true show the Discontinued label in Red.**



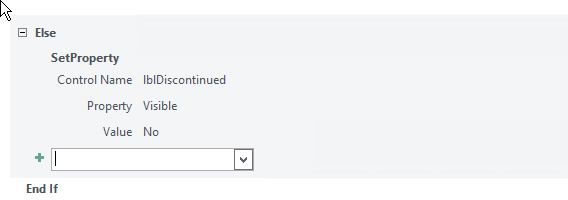
1. Click the **Add New Action** combobox and select **If**.
2. In the Conditional expression field type **“Dis”** and notice that intellisence displays both the Field and the Label associated with the Discontinued field. Choose **DiscontinuedCheckBox1** and add **“= Yes**”.
3. Add a new **SetProperty** action under the If condition. In the **Control Name** field type **lblDiscontinued**. Change the **Property** to **Visible** and the **Value** to **Yes**.



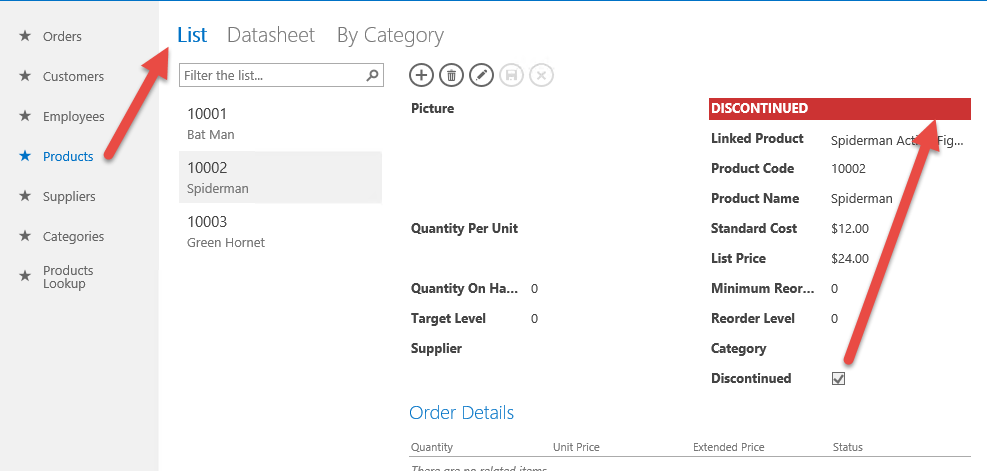
1. Add another **SetProperty** action above the **End If**.
2. Set the **Control Name** to **lblDiscontinued**, the **Property** to **BackColor** and the **Value** to **#cc3333**.
3. Add another **SetProperty** action above the **End If**.
4. Set the **Control Name** to **lblDiscontinued**, the **Property** to Fore**Color** and the **Value** to **#ffffff**.
5. Click **Add Else**. This will add a section for the condition when Discontinued is false.



1. In the **Else** block section add a **SetProperty** Action.
2. Set the **lblDiscontinued** control **Visible** property to **No**.



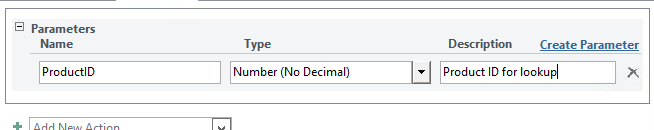
1. Click **Save.**
2. Click **Close** to return to the Product List tab.
3. Click **Save** from the tools menu.
4. Click Launch App to test your work.
5. Click on the Products table and (if you entered the data previously) you should see that moving from record to record triggers the On Current macro you created and causes the Discontinued flag to show based on the value of the Discontinued control.



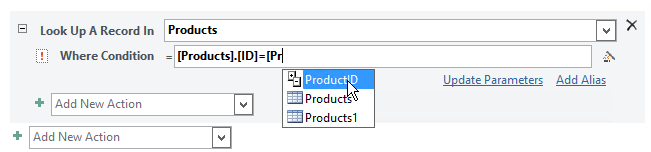
#### Create a Data Macro

Nothing we did previously will prevent the user from assigning discontinued items to an order. For that we can use a Data Macro. In this section we will create a data macro that prevents the use of discontinued items no matter how they are entered in to the Order Detail table. You will begin by creating the Data Macro and then “wire it” to the On Insert event of the Order Details table. The macro will look up a record based on the ProductID and then return a value indicating if the product is discontinued.

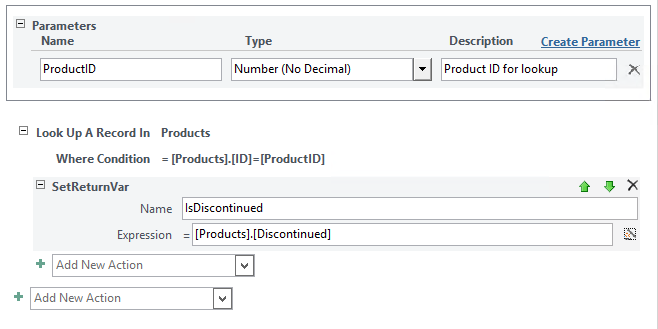
1. Return to **Access**. Close the **Products List** view.
2. From the ribbon on the Home tab in the **Create** group choose **Advanced | Data Macro.**
3. In the macro designer choose **Create Parameter**.
4. Provide the following properties:
   1. Name: **ProductID**
   2. Type: **Number (No Decimal)**
   3. Description: **Product ID for lookup**



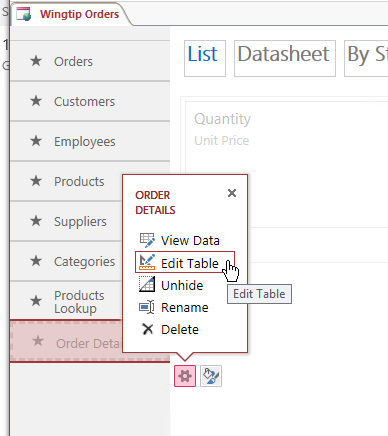
1. Use the **Add New Action** combobox to add a **LookupRecord** action.
2. Set the parameters of the Lookup action to the **Products** table **Where Condition = [Products].[ID]=[ProductID].** Notice that as you type Access will provide IntelliSense to look up valid values.



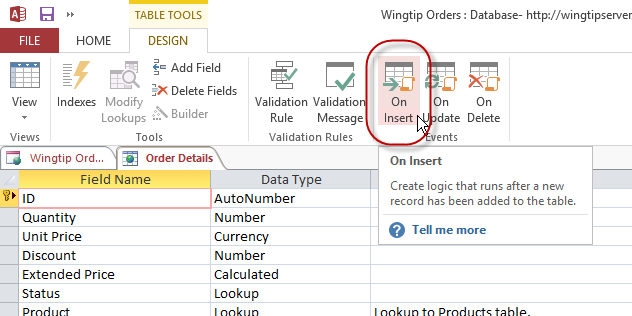
1. Add a **SetReturnVar** action to the action group.
2. Set the **Name** property to **IsDiscontinued** and the **Expression** to **[Products].[Discontinued]**. At this point your macro should look like the following:



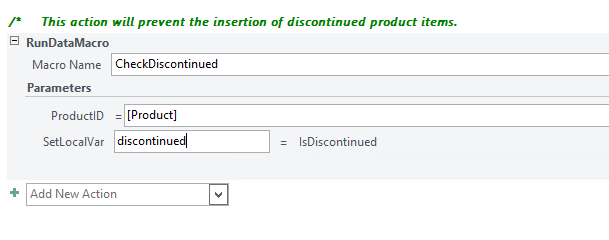
1. Click **Save** and name your macro **CheckDiscontinued**. If everything saves without error you can close the Macro designer.
2. Now you need to assign the data macro to the Order Details table. This is where ordered items are stored. The goal is to prevent the addition of discontinued items. **Click** the **Order Details** table and then **click** the **actions icon**. Choose **Edit Table**.



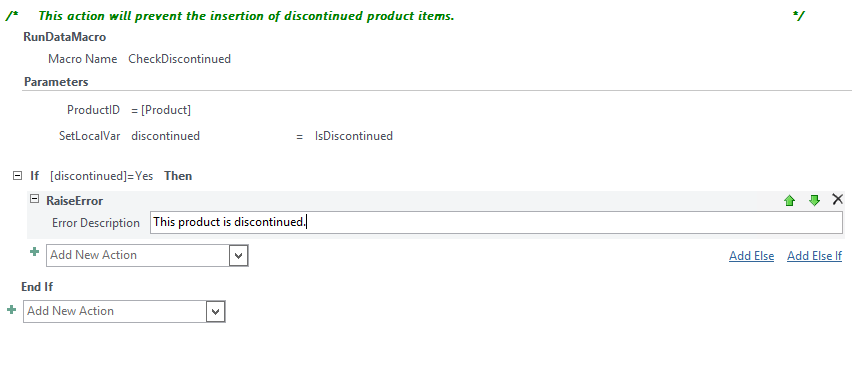
1. In the table designer, on the ribbon, in the **Events** group, click **On Insert**.



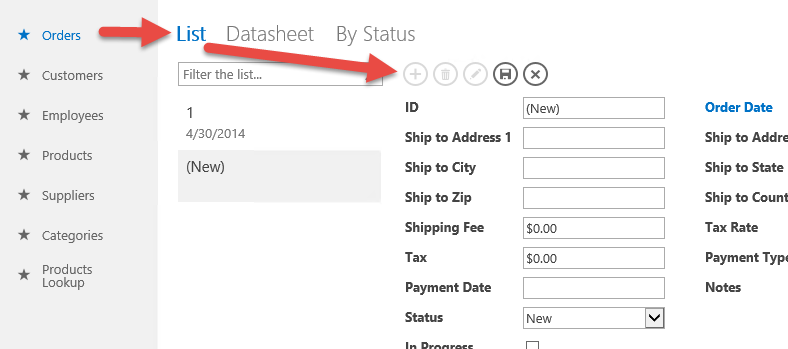
1. On the Macro design surface **Add** a **Comment** that says. **This action will prevent the insertion of discontinued product items.**
2. Under the comment, **Add** a **RunDataMacro** action.
3. Set the following properties of the RunDataMacro action
   1. Macro Name: **CheckDiscontinued**
   2. ProductID: **[Product]**
   3. SetLocalVar: **discontinued** = IsDiscontinued



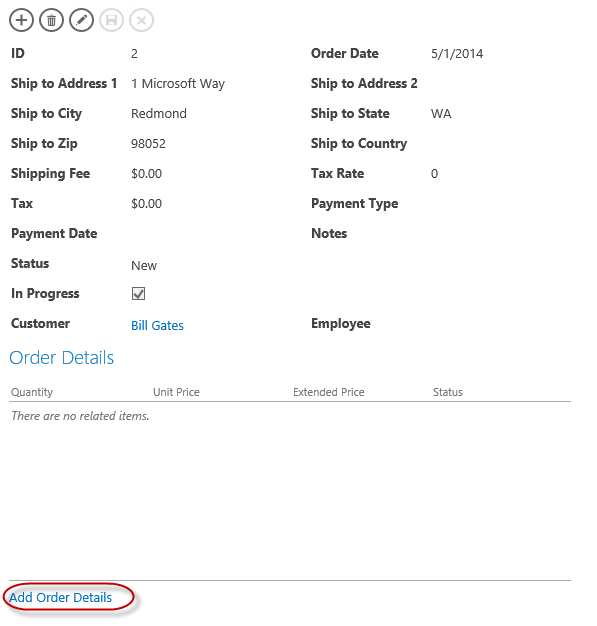
1. **Add** a new **If** action. Set the **Condition Expression** property to **[discontinued]=Yes.**
2. **Add** a new **RaiseError** action inside the **If** statement. Set the **Error Description** to **This product is discontinued.**



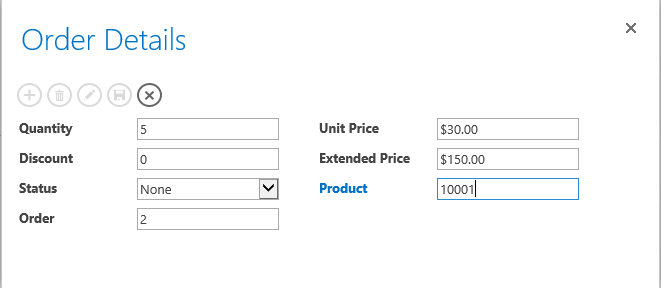
1. From the ribbon, click **Save** and **Close**.
2. **Close** the **Order Details** tab.
3. Click **Save** one last time and you are ready to test your work.
4. From the ribbon click **Launch App**.
5. Click on the **Orders** table. Click the **List** view.
6. Click the **+** icon to insert a new order record.



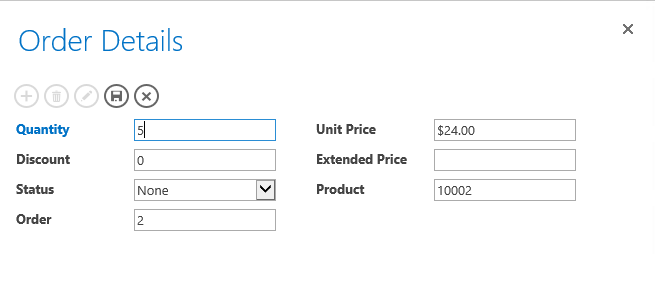
1. Supply a few details to set up the order record and click **Save**. Once the record is saved the Add Order Details link should appear at the bottom of the view.



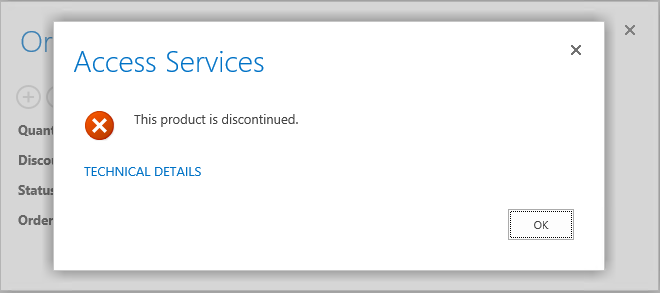
1. Click **Add Order Details**.
2. In the **Order Details** dialog add a Batman item **Product 10001**.
3. When you click **Save** there should be no error (if you entered the sample data from earlier in the lab).



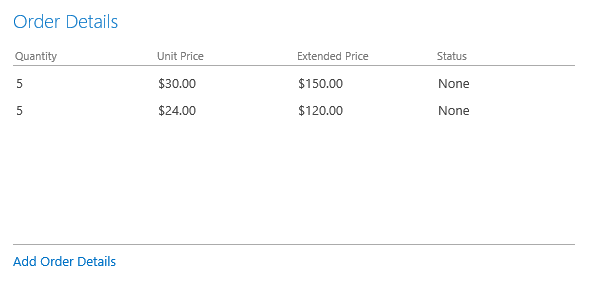
1. In the still open **Order Details** dialog, click **Add** (**+**).
2. Add a Spiderman Product 10002.



1. This time when you click Save you should receive an error that the product is discontinued.



1. Click **OK**.
2. Change the Product to **10003**. Click **Save** and you should not receive an error.
3. **Close** the **Order Details** dialog and you should see your two line items and the extended calculated prices.



In this exercise you added visual confirmation with a User Interface Macro and added data validation with a Data Macro. These two tools work together to create powerful business rules, data validation and stability for your App. Using these techniques you can create visually appealing functional applications without code.

You have completed this lab module.