

2010



# **COOKBOOK**

Creating an Order Form

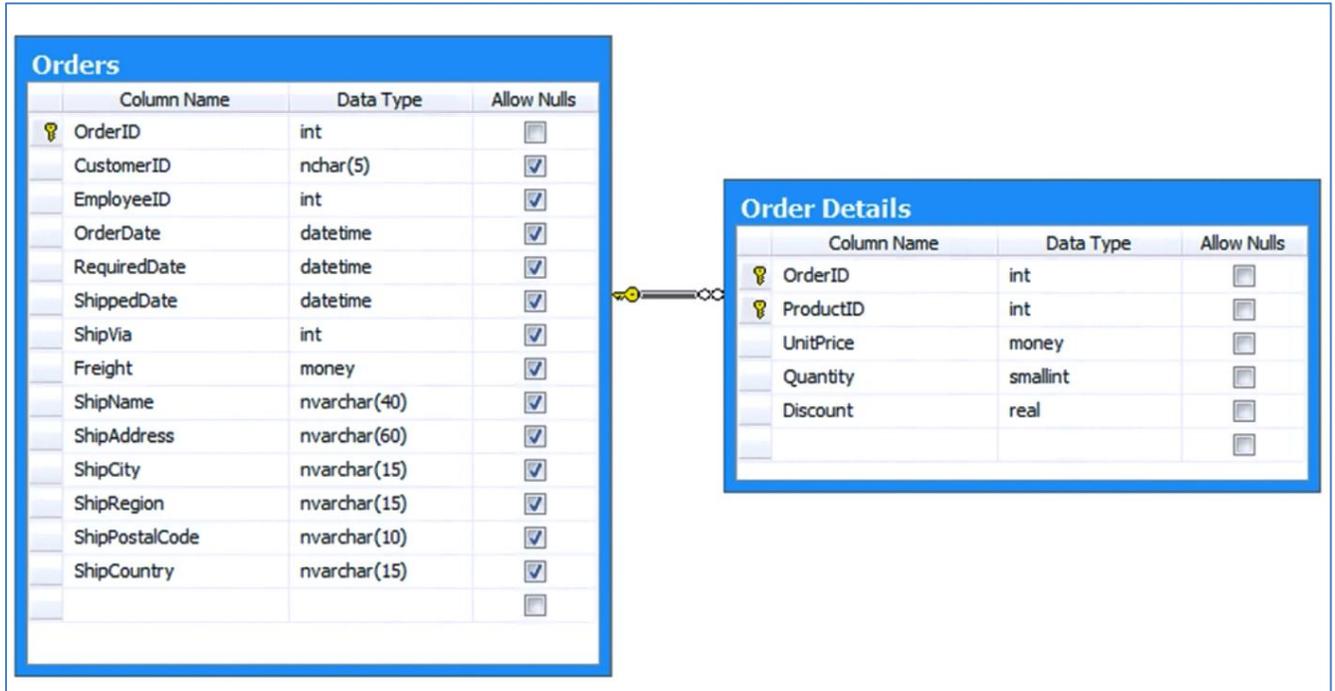
## Table of Contents

Understanding the Project.....	2
Table Relationships .....	2
Objective .....	4
Sample.....	4
Implementation .....	4
Generate Northwind Sample .....	5
Order Form Page.....	8
Add Page in Designer .....	8
Add Container to Page .....	9
Add Data View for “Orders” .....	9
Add Data View for “Order Details” .....	10
Customizing “Orders” Controller .....	12
Customizing “Order Details” Controller.....	19
Total and Subtotal.....	28
Calculating Freight .....	33
Custom Form Template .....	39

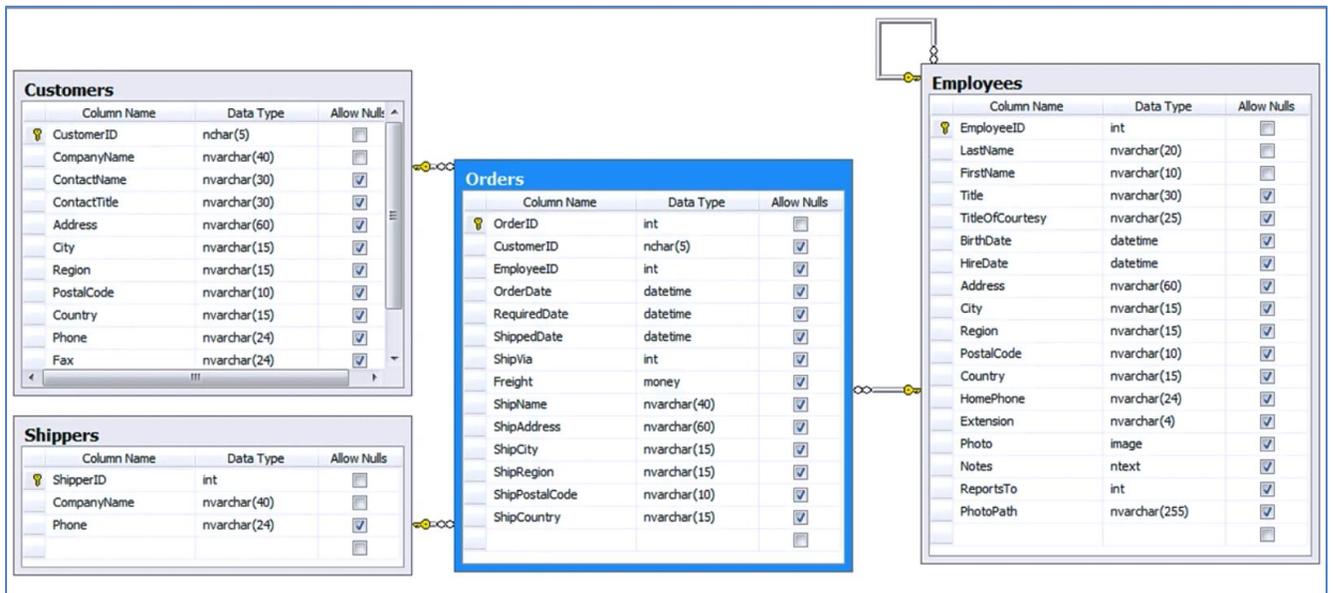
# Understanding the Project

## Table Relationships

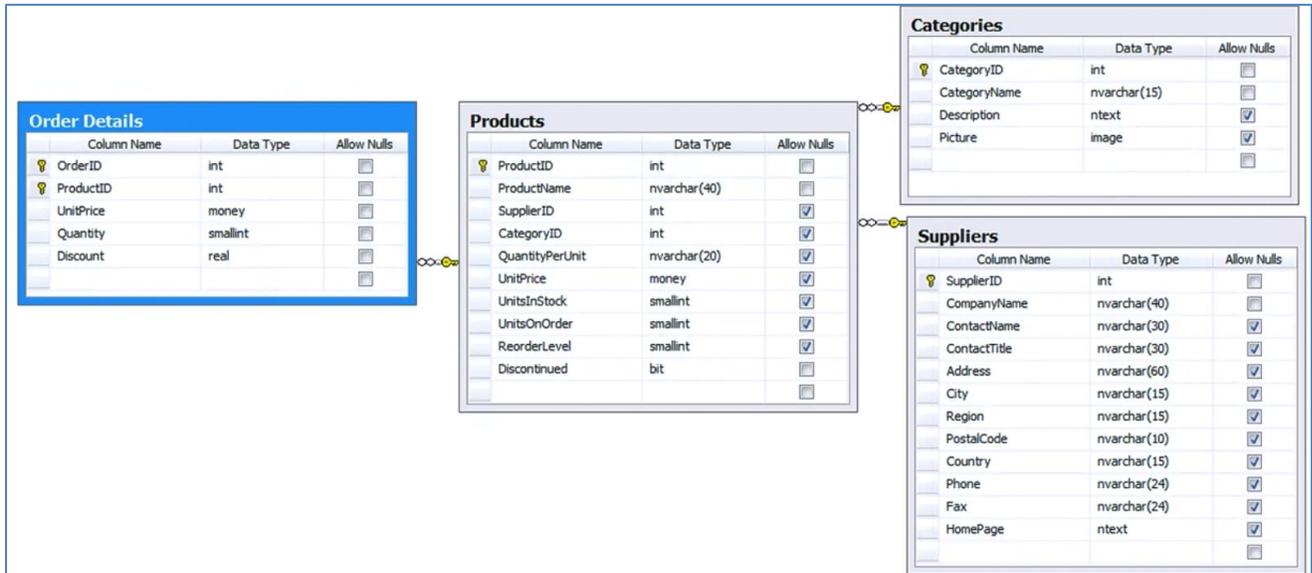
We have two tables, *Orders* and *Order Details*. Both tables are from the *Northwind* sample database. *Orders* is the master table, and *Order Details* is the details table.



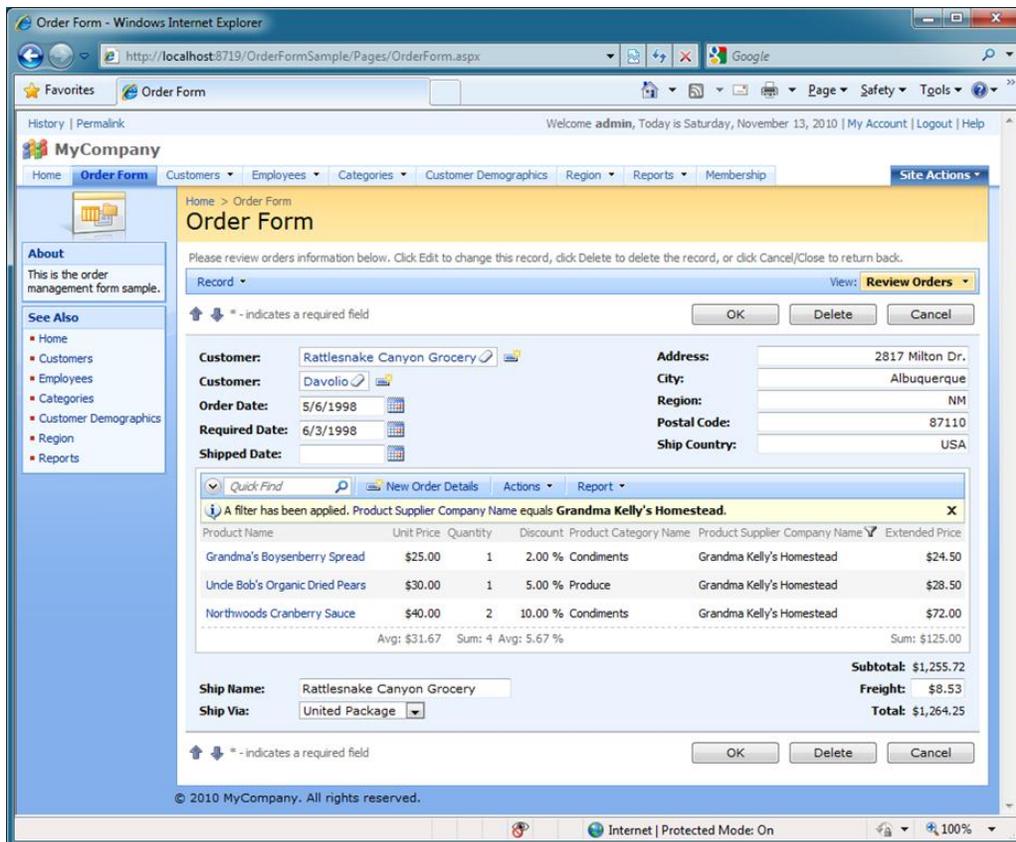
Each *Order* record references a *Customer*, an *Employee*, and a *Shipper*. We also know the *Order Date*, *Required Date*, *Shipped Date*, *Freight Amount*, and shipping information.



Order Details table features Unit Price, Quantity, Discount, and a pointer to Products. This also references Categories and Suppliers.



We want both Orders and Order Details to be presented as shown in the picture.



## Objective

The objective of this tutorial is to create an order detail form that allows the following:

1. Browsing a list of orders
2. Creating new orders
3. Editing existing orders
4. Calculating order freight
5. Displaying order subtotal and total

## Sample

Below is a picture of the sample order form in action. You can navigate through orders using the buttons with up and down arrows. Details of the current order will be displayed in the list inside of the order form template. The order subtotal and total are calculated based on the total extended price of all items. The total is composed of the freight added to the subtotal. The dynamic aggregate line automatically updates values based on the filter selected in the order details. It shows average unit price, sum of quantity, average discount and sum of extended price of line items.

Home > Order Form

### Order Form

Please review orders information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

Record ▾ View: **Review Orders** ▾

⬆ ⬇ \* - indicates a required field

**Customer:** Richter Supermarkt

**Employee:** Callahan

**Order Date:** 5/6/1998

**Required Date:** 6/3/1998

**Shipped Date:**

**Address:**  Starenweg 5

**City:**  Genève

**Region:**

**Postal Code:**  1204

**Ship Country:**  Switzerland

OK Delete Cancel

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Chang	\$19.00	10	15.00 %	Beverages	Exotic Liquids	\$161.00
Spegesild	\$12.00	30	15.00 %	Seafood	Lyngbysild	\$306.00
Lakkaliköön	\$18.00	2	15.00 %	Beverages	Karkko Oy	\$30.60
Avg: \$16.33		Sum: 42	Avg: 15.00 %			Sum: \$498.10

**Ship Name:** Richter Supermarkt

**Ship Via:** United Package

**Subtotal:** \$498.10

**Freight:** \$6.19

**Total:** \$504.29

OK Delete Cancel

⬆ ⬇ \* - indicates a required field

## Implementation

These are the steps we need to go through to implement an Order Form.

1. Generate sample *Northwind* web application
2. Add new page called **Order Form**
3. Customize *Orders* data controller
4. Customize *Order Details* data controller
5. Add *Total* and *Subtotal* to *Orders* controller
6. Calculate *Freight* based on order *Subtotal*
7. Create custom template for **Order Form**

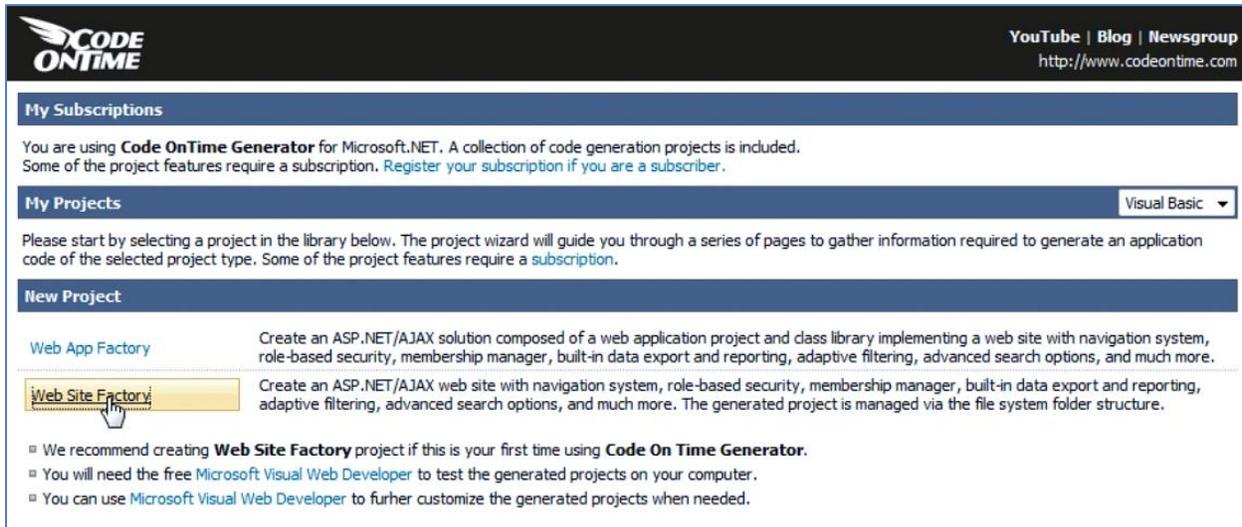
The steps are explained in further detail below.

## Generate Northwind Sample

If you don't have the *Northwind* database, navigate to

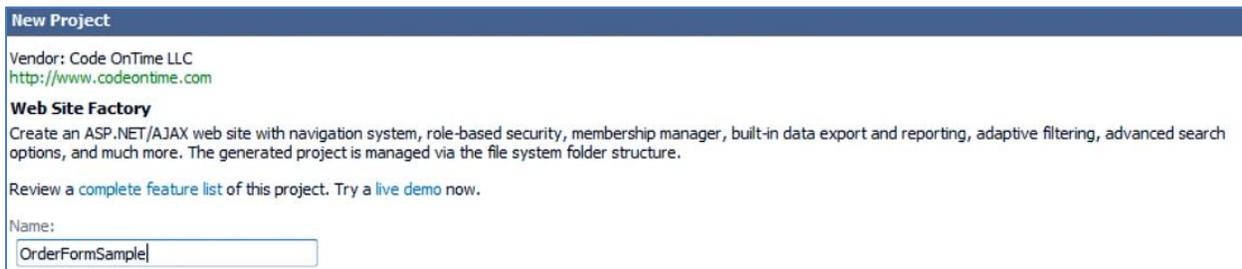
<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en> and download the database scripts.

Next, generate a *Web Site Factory* application using *Code On Time Generator* straight from the *Northwind* database.



The screenshot shows the Code On Time Generator website interface. At the top left is the Code On Time logo. At the top right are links for YouTube, Blog, and Newsgroup, along with the URL http://www.codeontime.com. Below the header is a 'My Subscriptions' section with a message about using the Code On Time Generator for Microsoft.NET and a link to register. This is followed by a 'My Projects' section with a 'Visual Basic' dropdown menu and a message about selecting a project. The 'New Project' section is highlighted, showing two options: 'Web App Factory' and 'Web Site Factory'. The 'Web Site Factory' option is selected, and a mouse cursor is pointing at it. Below the options are three bullet points: 'We recommend creating Web Site Factory project if this is your first time using Code On Time Generator.', 'You will need the free Microsoft Visual Web Developer to test the generated projects on your computer.', and 'You can use Microsoft Visual Web Developer to further customize the generated projects when needed.'

Give it the name of "OrderFormSample".



The screenshot shows the 'New Project' form in the Code On Time Generator website. It displays the vendor information: Code On Time LLC, http://www.codeontime.com. The 'Web Site Factory' project is selected. The description reads: 'Create an ASP.NET/AJAX web site with navigation system, role-based security, membership manager, built-in data export and reporting, adaptive filtering, advanced search options, and much more. The generated project is managed via the file system folder structure.' Below the description is a link to 'Review a complete feature list of this project. Try a live demo now.' The 'Name:' field is filled with 'OrderFormSample'.

For the database connection, access the connection string assistant by clicking on the link below the field, write in your server name, and select the *Northwind* database.

**Database Connection**

A valid connection string compatible with the selected data provider is required to generate the project. Data controller descriptors are automatically built from your database and are easy to maintain with the provided XML schemas.

Data Provider:

Connection String:

- Click [here](#) if you need assistance to build the connection string.
- Select the database tables and views included in the project.

Make sure to enable reporting.

**Reporting**

Support for [Microsoft Reporting Services](#) technology can be embedded into your application. Dynamic reports are automatically created on-the-fly without a need for a standalone reporting server. Static reports are supported as well.

Application users will be able to render reports as *Adobe PDF*, *Microsoft Office Excel*, or *Tiff* image file.

The Microsoft report viewer is required to run the reports. The *Microsoft Report Viewer Redistributable Package* includes Windows Forms and ASP.NET Web server controls for viewing reports designed using Microsoft reporting technology.

- Download the redistributable report viewer package for Microsoft.NET 3.5.
- Download the redistributable report viewer package for Microsoft.NET 4.0.

**Attention:**  
 If you have Microsoft Visual Studio 2008/2010 installed on your computer then the viewer is likely installed already. Users of Visual Web Developer Express 2008/2010 will have to install the report viewer if reports are enabled for this project.

You have to install the viewer on the server machine when the generated application is deployed.

Reports:  
 Enable dynamic and static reports in my application.

Enable ASP.NET membership.

**Authentication and Membership**

Please select authentication and membership options for your application. Some options will require custom coding or not compatible with each other.

Application Membership Features:

- Enable support for ASP.NET Membership with membership bar and user manager.
- Enable Windows Authentication. Recommended for *Intranet* applications only.
- Enable custom authentication and membership implementation. Requires additional coding.
- Enable a dedicated login page instead of a fly-over login dialog.
- Display "Remember Me" option on the fly-over login dialog.
- Login option "Remember Me" is set by default.
- Display "Password Recovery" link on the fly-over login dialog.
- Display "Sign Up" link on the fly-over login dialog.
- Display "My Account" link on the membership bar.
- Display "Help" link on the membership bar and support page-level help.
- Detect if user is idle for longer than  minutes and log the user out of the application.
- Membership will use a standalone database that already exists.

ASP.NET Membership gives you a built-in way to validate and store user credentials and helps you manage user authentication in your Web sites.

Standard ASP.NET membership features can be enhanced with AJAX-enabled user and role manager. Membership bar component will be displayed at the top of all pages and will provide an attractive AJAX-enabled login window, access to self-registration, password recovery, and user account modification with no code to write.

**ASP.NET Membership requires an instance of Microsoft SQL Express 2008 installed on your computer.** You may opt to host a standalone database membership database or keep the membership structures in your own database.

And finally, enable *Permalinks* and *Interactive History*.

**Features**

Specify the text displayed at the top of all pages in the page header. Project namespace is displayed if left blank.

Page Header:

Specify the text displayed at the bottom of all pages in the page footer. A standard copyright message is displayed if left blank.

Copyright:

**Annotations**  
A standard annotations plug-in allows to enhance all data controllers of a generated application with unlimited number of free-form notes and file attachments that can be associated with any records by end-users at run-time. Requires support for ASP.NET Membership option to be enabled.

Enable global record annotations and store attachment and note files in  folder.

**Form Layout**  
Standard form layout displays category information on the left side of the screen. The data fields are listed on the right side of the screen.

Start each data field category in the new column with category information displayed on top.

Float data fields in view categories from left to right to fill the entire space available.

Show modal forms in master data views without children and in child data views.

**Grid Layout**  
All grid views will present up to  data fields. Use *Designer* to modify, add, and remove the data fields of individual views.

Activate search mode in master grid views by default.

Data lookup windows must always open in search mode.

Enable multi-selection in all grid views. Only *Delete* action is automatically supported on multiple rows.

Enable batch editing in all data controllers. Requires multi-selection mode.

**Miscellaneous**

Enable relationship explorer hyperlinks in lookup fields of all data controllers.

Enable permalinks to allow bookmarking of master records selected by end users.

Enable interactive history of most-recent-used data objects.

Leave the rest of the options with their default values and generate the application.

## Order Form Page

### Add Page in Designer

Now it's time to create a new page in the *Designer*, with the name of "Order Form".

In *Code On Time Generator*, click on the name of the project, and press the *Design* button. Go to the *All Pages* tab. On the action bar, press *New | New Page*. The name will be "OrderForm", with *Index* of "1005", *Title* and *Path* of "Order Form", and *Description* of "This is the order management form".

All Controllers All Commands All Fields All Views All Data Fields All Pages All User Controls

Please fill this form and click OK button to create a new page record. Click Cancel to return to the previous screen.

View: **New Controller**

\* - indicates a required field

OK Cancel

**General**

Name and index of the page. The address of the generated ASP.NET page is `~/Pages/Name.aspx` where **Name** is the specified name.

Use *External Url* to create a menu link to an external web site. No physical application page is generate if *External Url* is not blank.

If *External Url* is equal to `about:blank` then a site map node without a Url is created.

Name \*  
OrderForm

Index  
1005

External Url

**Presentation**

Page title is displayed in the title of the browser window.

Use symbol "[|" in the page path to define a multi-level menu option that selects the page. Make sure that any segment of the path is matched to a path of an existing page that has an index less then the index of this page.

If *Path* is left blank then there will be no menu option to access the page.

Page description is displayed as a tool tip of the corresponding menu option.

Custom style is one or more CSS classes. Use *Wide* as custom style to eliminate the side bar on the page.

Title \*  
Order Form

Path  
Order Form

Description  
This is the order management form.

The *Style* will be "Miscellaneous", and *About This Page* will be the same as *Description*. Remove "\*" from *Roles* to hide the menu option for anonymous users.

The page will feature *About* box on the side bar if specified.

Style \*  
Miscellaneous

Custom Style

About This Page  
This is the order management form.

**Security**

Security settings for this page.

User ? to allow anonymous access to the page.

Roles

List all roles that are authorized to access the page. Separate multiple roles with a comma.

## Add Container to Page

Click on the new page in the *All Pages* list, and navigate to the *Containers* tab. On the action bar, press *New | New Container*. Leave the properties as default and save the new container.

**General**  
A container can host multiple data views and user controls. At least one container must be declared for each page.  
Specify container flow rule and optional width.  
Optional container width must be expressed as a percent of the total page width or as an exact width in pixels. The examples of width are *40%* and *300px*.

Id: N/A  
Flow\*: New Row  
Width:   
CSS Class Name:   
CSS Style Properties:

\* - indicates a required field

OK Cancel

## Add Data View for "Orders"

Navigate to the *Data Views* tab, and press *New | New Data View*. The *Container* will be "c100", *Controller* will be "Orders", and *View* will be "grid1".

Page Containers **Data Views** Controls

Please fill this form and click OK button to create a new data view record. Click Cancel to return to the previous screen.

View: New Data View

\* - indicates a required field

OK Cancel

**General**  
Page, container, controller, and controller view of the data view.  
Use *Tag* to enable conditional controller actions with matching *whenTag* property and to write custom business rules specific to tagged data views.

Id: N/A  
Container\*: c100  
Controller\*: Orders  
View: grid1

Scroll down to *Presentation* properties, and uncheck “Show Details in List Mode”. This way, no details will be shown next to master records in the list.

**Presentation**  
Presentation properties of the data view.

- Show In Summary
- Page Size:
- Selection Mode \*:
- Show Action Bar
- Show View Description
- Show View Selector
- Show Paggers
- Show Modal Forms
- Search on Start
- Show Details in List Mode

Don't forget to save the record.

### Add Data View for “Order Details”

On the action bar, add another data view by pressing *New / New Data View*. *Container* will be “c100”, *Controller* will be “OrderDetails”, and *View* will be “grid1”.

**Project Designer** exit

Home > Page: **OrderForm**

Page Containers **Data Views** Controls

Please fill this form and click OK button to create a new data view record. Click Cancel to return to the previous screen.

View: **New Data View**

\* - indicates a required field

**General**  
Page, container, controller, and controller view of the data view.  
Use *Tag* to enabled conditional controller actions with matching *whenTag* property and to write custom business rules specific to tagged data views.

**Activator**  
Specify a method of view activation available to end users. *Text* attribute will represent a tab or menu option of activator.

Id: N/A

Container \*: c100

Controller \*: OrderDetails

View: grid1

Tag:

Transaction: N/A

Activator: None

OK Cancel

Let's set up a few other properties below. Disable "Show View Description", "Show View Selector", "Show Pagers", set *Page Size* to "300", and enable "Show Modal Forms".

**Presentation**  
Presentation properties of the data view.

Show In Summary  
 Page Size  
  
 Selection Mode \*  
  
 Show Action Bar  
 Show View Description  
 Show View Selector  
 Show Pagers  
 Show Modal Forms  
 Search on Start  
 Show Details in List Mode

Next, set *Filter Source* to be the *Orders* data controller from the data view "dv100". The *Filter Field* will be "OrderID". Set *Auto-Hide* field to "Self".

**Filter**  
Filter parameters can be used to limit the visible data or to establish master-detail relationships.

Property *Auto Hide* specifies user interface element that will be hidden if runtime filter value is empty and view can be automatically hidden. Use *Self* when master data view has a *Tab* activator and belongs to the same page container as this data view. Use *Container* otherwise.

Filter Source  
  
 Filter Fields  
  
 Auto Hide

Close the *Designer* and regenerate the project. (Note: You only need to regenerate the application to view the latest changes). When you sign into the web application, you can see that the *Order Form* page has been added to the menu navigation and sitemap. Navigate to the page, and you can see the list of orders.

History | Permalink Welcome admin, Today is Monday, November 15, 2010 | My Account | Logout | Help

**MyCompany**

Home **Order Form** Customers Employees Categories Customer Demographics Region Reports Membership Site Actions

**About**  
This is the order management form.

**See Also**

- Home
- Customers
- Employees
- Categories
- Customer Demographics
- Region
- Reports

Home > Order Form

This is a list of orders.

Customer Company Name	Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via Company Name	Freight	Ship Name	Ship Address	Ship City
Vins et alcools Chevalier	Buchanan	7/4/1996	8/1/1996	7/16/1996	Federal Shipping	\$32.38	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims
Toms Spezialitäten	Suyama	7/5/1996	8/16/1996	7/10/1996	Speedy Express	\$11.61	Toms Spezialitäten	Luisenstr. 48	Münster
Hanari Carnes	Peacock	7/8/1996	8/5/1996	7/12/1996	United Package	\$65.83	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro
Victualles en stock	Leverling	7/8/1996	8/5/1996	7/15/1996	Speedy Express	\$41.34	Victualles en stock	2, rue du Commerce	Lyon
Suprêmes délices	Peacock	7/9/1996	8/6/1996	7/11/1996	United Package	\$51.30	Suprêmes délices	Boulevard Tirou, 255	Charleroi
Hanari Carnes	Leverling	7/10/1996	7/24/1996	7/16/1996	United Package	\$58.17	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro
Chop-suey Chinese	Buchanan	7/11/1996	8/8/1996	7/23/1996	United Package	\$22.98	Chop-suey Chinese	Hauptstr. 31	Bern
Richter Supermarkt	Dodsworth	7/12/1996	8/9/1996	7/15/1996	Federal Shipping	\$148.33	Richter Supermarkt	Starenweg 5	Genève
Wellington Importadora	Leverling	7/15/1996	8/12/1996	7/17/1996	United Package	\$13.97	Wellington Importadora	Rua do Mercado, 12	Resende
HILARION-Abastos	Peacock	7/16/1996	8/13/1996	7/22/1996	Federal Shipping	\$81.91	HILARION-Abastos	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal

« Previous | Page: 1 2 3 4 5 6 7 8 9 10 ... | Next » Items per page: 10, 15, 20, 25 | Showing 1-10 of 830 items | Refresh

You can browse the list of orders. Select an order, and you will view its details, including order details below.

## Customizing “Orders” Controller

### Set Sort Expression

In the *Designer*, select the *Orders* controller from the list of *All Controllers*. Switch to the *Views* tab, and select “grid1”. Edit *Sort Expression* field so that it reads “OrderDate desc”. The grid will be ordered in descending order by *Order Date*.

<b>Sort and Filter</b> Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix.	Sort Expression OrderDate desc
---	-----------------------------------

### Configure “Customer ID” Lookup Field

If you create a new order in the current application, the *Customer Company Name* needs to be selected using the lookup. You can also use advanced search to find the records by a specific field. It would be nice if advanced search opened by default. It would also be nice if the shipping information of the selected customer would be pasted into the order information.

This is a list of customers.

View: Customers

Customer# equals  
Company Name equals  
Contact Name equals  
Contact Title equals  
Address equals

Search  
Reset

Customer#	Company Name	Contact Name	Contact Title	Address	City	Region	Postal Code	Country	Phone
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	n/a	12209	Germany	030-0074321
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	n/a	05021	Mexico	(5) 555-4729
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	n/a	05023	Mexico	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	n/a	WA1 1DP	UK	(171) 555-7788
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	n/a	S-958 22	Sweden	0921-12 34 65
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	n/a	68306	Germany	0621-08460
BLONP	Blondesdls père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	n/a	67000	France	88.60.15.31
BOLID	Bólide Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid	n/a	28023	Spain	(91) 555 22 82
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	n/a	13008	France	91.24.45.40
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729

« Previous | Page: 1 2 3 4 5 6 7 8 9 10 | Next »

Items per page: 10, 15, 20, 25 | Showing 1-10 of 91 items | Refresh

This can be done in *Designer*. Select the *Orders* data controller from the list of all controllers. Navigate to the *Fields* tab, and click on the *CustomerID* field. Press *Edit*, and scroll down to the *Lookup* section. Change the *Data Value Field* to *CustomerID*, and the *Data Text Field* to *CompanyName*. The *Copy* field will specify which fields are copied from the selected customer into the orders record. In this field, write:

```
ShipName=ContactName
ShipAddress=Address
ShipCity=City
ShipRegion=Region
ShipPostalCode=PostalCode
ShipCountry=Country
```

Enable “Search on Start” and “Activate If Blank”. In *Lookup window description*, type “Select a customer”.

**Lookup**

Lookup settings can be based on another data controller or defined as static items. Follow the link to learn more about [lookup item styles](#).

You can list static lookup items on the *Items* tab.

Property *Copy* specifies the fields that must be copied from the lookup data row when a lookup value is selected. Specify one copy source per line in format *FieldName=LookupFieldName*.

Items style *Check Box List* allows to configure the field as many-to-many if you set the data type to *String*, indicate that the value of the field is *computed at runtime* and select a *Target Controller*.

Lookup is rendered in search mode if *Search on Start* is checked.

The lookup window can be activated automatically in edit/new mode if the field value is blank.

Items Style: Lookup

Items Data Controller: Customers

Data Value Field: CustomerID

Data Text Field: CompanyName

New Data View: createForm1

Copy: ShipRegion=Region, ShipPostalCode=PostalCode, ShipCountry=Country

Search on Start

Activate If Blank

Lookup window description: Select a customer.

Close the *Designer*, and regenerate the application. Navigate to the *Order Form* page in the web application. While creating a new order, if you activate the lookup for *Customer Company Name*, the lookup will be in advanced search mode.

New Orders

Complete the form. Make sure to enter all required fields.

Customer Company Name: Richter Supermark

Employee Last Name: (select)

Order Date: [calendar icon]

Required Date: [calendar icon]

Shipped Date: [calendar icon]

Ship Via Company Name: (select)

Freight: [input field]

Ship Name: Michael Holz

Ship Address: Grenzacherweg 237

Ship City: Genève

Ship Region: [input field]

Ship Postal Code: 1203

Ship Country: Switzerland

Select a customer

View: Customers

Customer # equals [input field] Search

Company Name equals [input field] Reset

Contact Name equals [input field] Refresh

When you select a customer from the lookup, the shipping information will be copied over as well.

New Orders

Complete the form. Make sure to enter all required fields.

Customer Company Name: Richter Supermark

Employee Last Name: (select)

Order Date: [calendar icon]

Required Date: [calendar icon]

Shipped Date: [calendar icon]

Ship Via Company Name: (select)

Freight: [input field]

Ship Name: Michael Holz

Ship Address: Grenzacherweg 237

Ship City: Genève

Ship Region: [input field]

Ship Postal Code: 1203

Ship Country: Switzerland

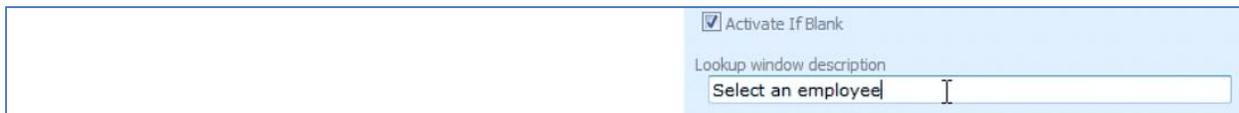
## Configure "Employee ID" Lookup Field

In the *Designer*, go to *All Controllers*. Select the *Employees* data controller. Switch to the *Views* tab. Select "grid1", and switch to *Data Fields* tab. On the action bar, press *New | New Data Field*. Set *Field Name* to *Photo*, and save the field.



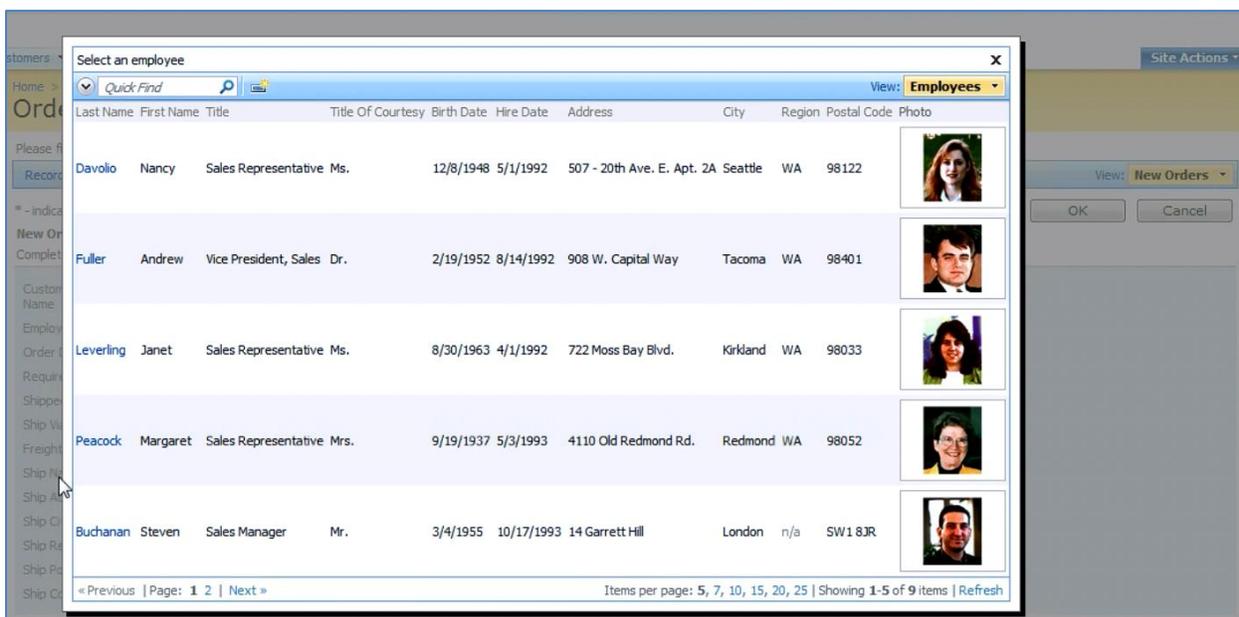
The screenshot shows the 'Project Designer' interface. The breadcrumb path is 'Home > Controller: Employees > View: grid1'. The 'Data Fields' tab is active. A message at the top says 'Please fill this form and click OK button to create a new data field record. Click Cancel to return to the previous screen.' The 'View' dropdown is set to 'New Data Field'. Below this, there are 'OK' and 'Cancel' buttons. The main form area is titled 'New Data Field' and contains the instruction 'Complete the form. Make sure to enter all required fields.' On the right side of the form, there are three fields: 'Field Name \*' with the value 'Photo', 'Category' with a '(select)' dropdown, and 'Alias' with a '(select)' dropdown.

Now, go back to the list of *All Controllers*, and select the *Orders* data controller. Navigate to the *Fields* tab and click on *EmployeeID*. Edit, and scroll down to the *Lookup* section. Enable "Activate If Blank" and type "Select an employee" for *Lookup* window description.



The screenshot shows a configuration window for a field. It has a checkbox labeled 'Activate If Blank' which is checked. Below it is a text input field for 'Lookup window description' containing the text 'Select an employee'.

Now in the regenerated application, when you select a customer for a new order, the *Employee* lookup will automatically appear.



The screenshot shows a 'Select an employee' lookup window. It has a search bar with 'Quick Find' and a 'View: Employees' dropdown. Below the search bar is a table with columns: Last Name, First Name, Title, Title Of Courtesy, Birth Date, Hire Date, Address, City, Region, Postal Code, and Photo. The table contains five rows of employee data. At the bottom, there are navigation controls: '<< Previous | Page: 1 2 | Next >>' and 'Items per page: 5, 7, 10, 15, 20, 25 | Showing 1-5 of 9 items | Refresh'.

Last Name	First Name	Title	Title Of Courtesy	Birth Date	Hire Date	Address	City	Region	Postal Code	Photo
Davolo	Nancy	Sales Representative	Ms.	12/8/1948	5/1/1992	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	
Fuller	Andrew	Vice President, Sales	Dr.	2/19/1952	8/14/1992	908 W. Capital Way	Tacoma	WA	98401	
Leverling	Janet	Sales Representative	Ms.	8/30/1963	4/1/1992	722 Moss Bay Blvd.	Kirkland	WA	98033	
Peacock	Margaret	Sales Representative	Mrs.	9/19/1937	5/3/1993	4110 Old Redmond Rd.	Redmond	WA	98052	
Buchanan	Steven	Sales Manager	Mr.	3/4/1955	10/17/1993	14 Garrett Hill	London	n/a	SW1 8JR	

## Set Default Value for "Order Date" Field

In the list of *All Controllers*, select the *Orders* controller. Switch to the *Fields* tab, and click on *OrderDate*. Press *Edit*, and enter "DateTime.Now" in the *Code Default* field.

The screenshot shows the Project Designer window with the following configuration for the OrderDate field:

- Name:** OrderDate
- Controller:** Orders
- Type:** DateTime
- Allow null values.
- The value of this field is computed at run-time by SQL expression.
- The value of the field is calculated by a business rule expression.
- Server Default:** (empty field)
- Code Default:** DateTime.Now
- Value is retrieved on demand

The General tab on the left provides instructions for field configuration, including how to use Server Default, Code Default, and computed fields.

Save, and regenerate the application. When you create a new order, the current date will be automatically entered into *Order Date*.

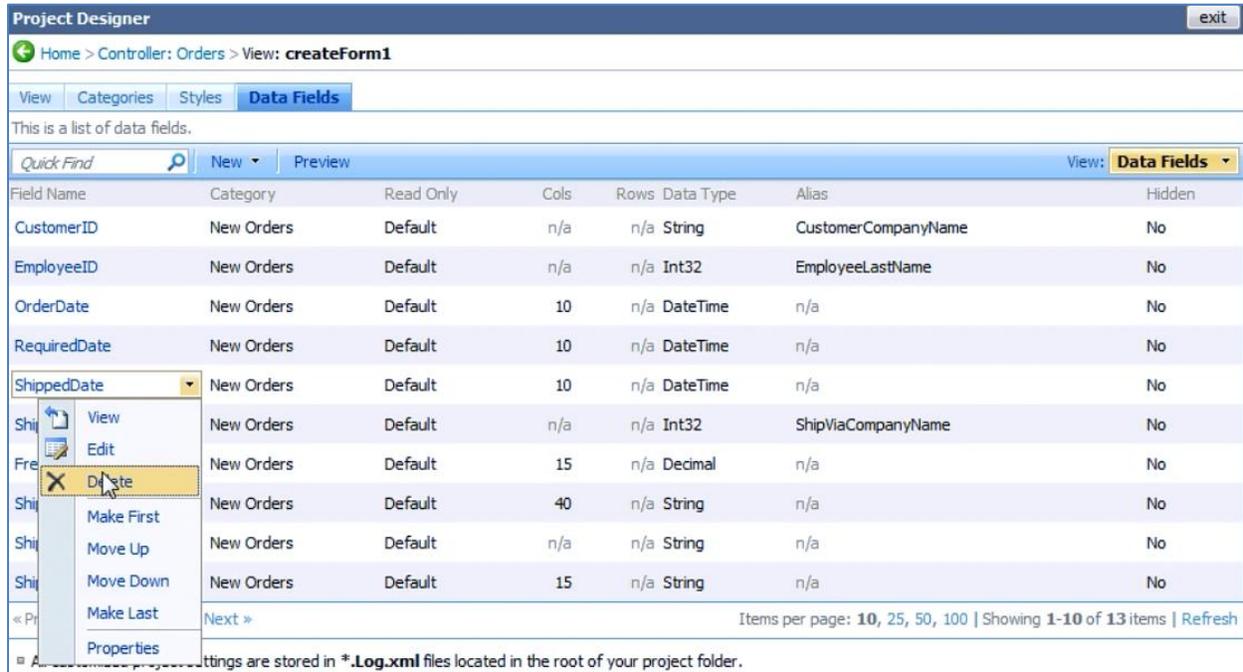
The screenshot shows the Order Form application with the following data entered:

- Customer Company Name:** Berglunds snabbköp
- Employee Last Name:** Leverling
- Order Date:** 11/15/2010
- Required Date:** (empty)
- Shipped Date:** (empty)
- Ship Via Company Name:** (select)
- Freight:** (empty)
- Ship Name:** Christina Berglund
- Ship Address:** Berguvsvägen 8
- Ship City:** Luleå
- Ship Region:** (empty)
- Ship Postal Code:** S-958 22
- Ship Country:** Sweden

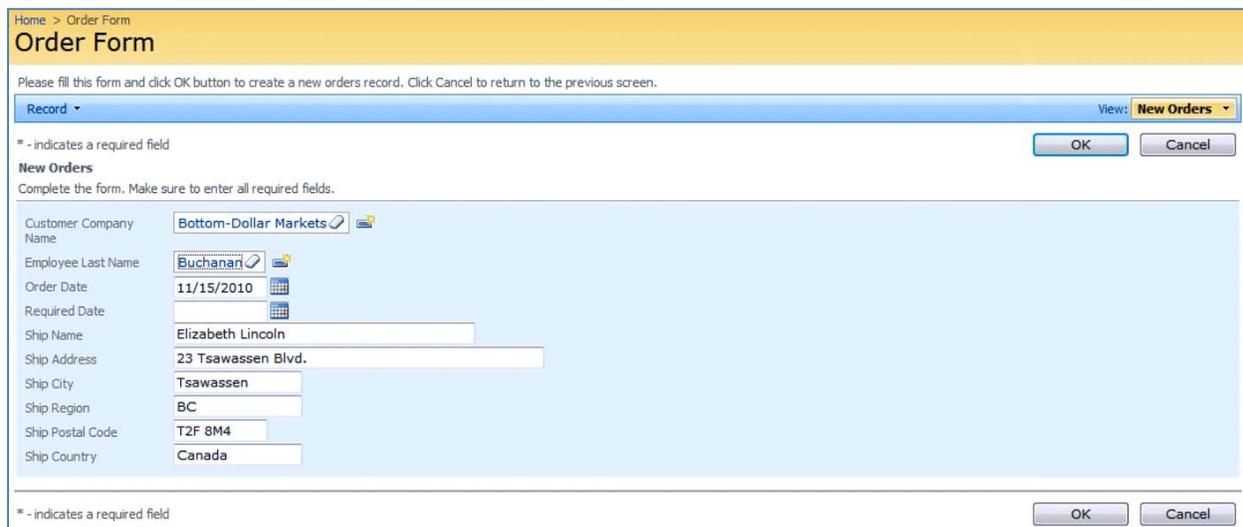
The application title is "Order Form" and the view is "New Orders".

## Delete Fields From “createForm1” View

Select the *Orders* controller from the list of *All Controllers*. Navigate to the *Views* tab, and click on “createForm1”. Switch to the *Data Fields* tab. By using the dropdown menu next to *ShippedDate*, press *Delete*.



Delete the fields *ShipVia* and *Freight* as well. Save, and regenerate the application. Below, you can see the compact version of *createForm1* without the fields *ShippedDate*, *ShipVia*, and *Freight*.



When the record is saved, it will be automatically selected, and *Order Details* will be displayed below the *Order* record. It would be nice if the master record would be in edit mode right after the insertion.

### Display Inserted Master Record in Edit Mode

In the list of *All Controllers*, select *Orders*. Navigate to the *Action Groups* tab, and select “ag2” from the list. Click on the *Actions* tab at the top of the page. The very last action in the list is *Select*. Using the context menu, edit the action and change the *Command Name* to “Edit”.

Command Name	Command Argument	Header Text	When Last Command Name	When Last Command Argument	When Key Selected	When HRef (Regex)	When View (Regex)	When Tag (Regex)	Roles	Scope
Edit	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Form
Delete	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Form
Cancel	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Form
Update	n/a	n/a	Edit	n/a	n/a	n/a	n/a	n/a	n/a	Form
Delete	n/a	n/a	Edit	n/a	n/a	n/a	n/a	n/a	n/a	Form
Cancel	n/a	n/a	Edit	n/a	n/a	n/a	n/a	n/a	n/a	Form
Insert	n/a	n/a	New	n/a	n/a	n/a	n/a	n/a	n/a	Form
Cancel	n/a	n/a	New	n/a	n/a	n/a	n/a	n/a	n/a	Form
Insert	n/a	n/a	Duplicate	n/a	n/a	n/a	n/a	n/a	n/a	Form
Cancel	n/a	n/a	Duplicate	n/a	n/a	n/a	n/a	n/a	n/a	Form

Context menu for 'Edit':

- Edit (selected)
- editForm1
- Insert
- Yes
- createForm1
- Form

Buttons: Save, Cancel

Save the action, and regenerate the application. When you save a new record in the *Order Form* page, it will still be editable without having to press *Edit*.

### Set Size of “Shipping” Data Fields

In the list of *All Controllers*, select the *Orders* controller. Switch to the *Views* tab. Select *editForm1*, and switch to the *Data Fields* tab. Edit the *Freight* field, and change *Columns* to “5”.

Field Name	Category	Read Only	Cols	Rows	Data Type	Alias	Hidden
CustomerID	Orders	Default	n/a	n/a	String	CustomerCompanyName	No
EmployeeID	Orders	Default	n/a	n/a	Int32	EmployeeLastName	No
OrderDate	Orders	Default	10	n/a	DateTime	n/a	No
RequiredDate	Orders	Default	10	n/a	DateTime	n/a	No
ShippedDate	Orders	Default	10	n/a	DateTime	n/a	No
ShipVia	Orders	Default	n/a	n/a	Int32	ShipViaCompanyName	No
Freight	Orders	Default	5		Decimal	(select)	
ShipName	Orders	Default	40	n/a	String	n/a	No

Buttons: Save, Cancel

Change the number of *Columns* for all *shipping* fields to “30”, as shown below.

ShipName	Orders	Default	30	n/a	String	n/a	No
ShipAddress	Orders	Default	30	n/a	String	n/a	No
ShipCity	Orders	Default	30	n/a	String	n/a	No
ShipRegion	Orders	Default	30	n/a	String	n/a	No
ShipPostalCode	Orders	Default	30	n/a	String	n/a	No
ShipCountry	Orders	Default	30		String	(select)	<input type="checkbox"/>

Save Cancel

### Change “Ship Via Company Name” Lookup to Dropdown

Select the *Orders* controller from the *All Controllers* list. On the *Fields* tab, select *ShipVia*. Click *Edit*, and scroll down to *Lookup* section. Change the *Items Style* to “Drop Down List”.

**Lookup**

Lookup settings can be based on another data controller or defined as static items. Follow the link to learn more about [lookup item styles](#).

Items Style: Drop Down List

Now, go to the *Categories* tab and edit *Orders* category. Change the *Floating* field to “Yes”, so that the fields will float.

Project Designer

Home > Controller: Orders

Controller Commands Fields Views **Categories** Data Fields Action Groups Actions

This is a list of data field categories in the view. Categories are not supported in grid views.

Quick Find Record View: Categories

Header Text	View	Description	New Column	Tab	Floating	Collapsed
New Orders	createForm1	\$DefaultNewDescription	n/a	n/a	n/a	n/a
Orders	editForm1	\$DefaultEditDescription	N/A		Yes	N/A

Save Cancel

If you save and regenerate the application, the *Order Form* page will look like the image below.

Please review orders information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

Record View: Review Orders

\* - indicates a required field

OK Delete Cancel

**Orders**

These are the fields of the orders record that can be edited.

Customer Company Name: Richter Supermarkt  
 Employee Last Name: Fuller  
 Order Date: 11/15/2010  
 Required Date:  
 Shipped Date:  
 Ship Via Company Name: N/A  
 Freight: \$0.00  
 Ship Name: Michael Holz  
 Ship Address: Grenzacherweg 237  
 Ship City: Genève  
 Ship Region:  
 Ship Postal Code: 1203  
 Ship Country: Switzerland

\* - indicates a required field

OK Delete Cancel

Quick Find New Order Details Actions Report

Product Name Unit Price Quantity Discount Order Customer Company Name Order Employee Last Name Order Ship Via Company Name Product Category Name Product Supplier Company Name

No records found.

## Customizing “Order Details” Controller

### Customize “Product Id” Lookup

Select Order Details controller from the All Controllers list, and switch to the Fields tab. Click on “ProductID”, and press Edit. Scroll down to the Lookup section. In the Copy field, write “UnitPrice=UnitPrice”, so that the unit price of the product will be pasted into the unit price of the order. Enable “Search on Start” and “Activate if Blank”. Lookup window description will be “Select a product”.

**Lookup**

Lookup settings can be based on another data controller or defined as static items. Follow the link to learn more about [lookup item styles](#).

You can list static lookup items on the *Items* tab.

Property *Copy* specifies the fields that must be copied from the lookup data row when a lookup value is selected. Specify one copy source per line in format *FieldName=LookupFieldName*.

Items style *Check Box List* allows to configure the field as many-to-many if you set the data type to *String*, indicate that the value of the field is *computed at runtime* and select a *Target Controller*.

Lookup is rendered in search mode if *Search on Start* is checked.

The lookup window can be activated automatically in edit/new mode if the field value is blank.

Items Style: **Lookup**

Items Data Controller: **Products**

Data Value Field: **(select)**

Data Text Field: **(select)**

New Data View: **createForm1**

Copy: **UnitPrice=UnitPrice**

Search on Start

Activate If Blank

Lookup window description: **Select a product**

Save and regenerate the application. Now, when an order is selected in the Order Form page, and you create a new Order Detail, a prompt will immediately open requiring you to select a product.

**New Order Details**

Please fill this for **New Order Det**  
Complete the for

Product Name \* | equals | | Search

Supplier Company Name \* | equals | | Reset

Category Name \* | equals | | Refresh

View: **Products**

OK Cancel

When you select a product, the unit price will automatically be copied into the *Order Details* record.

### Assign Default Values to “Quantity” and “Discount”

Now select the field *Quantity*, and press Edit. You can see that the standard default value is ((1)), assigned as part of the SQL expression. In the Code Default field, type “1” and save the field. The expression will be in either C# or VB, depending on the language of the project.

Perform the same operation on *Discount* field. Provide a Code Default of “0”.

For the *Discount* field, scroll down to the Presentation section, and change Data Format String to “p” to format the field as a percentage. You can also write “{0:p}”.

**Presentation**  
Presentation interface properties of the field.

A data format string is applied to the field value on the client via JavaScript *String.format* function. You can override the data format string on the data field level in views.

A data format string is applied to the field value on the client via JavaScript *String.format* function. You can enable a server-side formatting via .NET *System.String.Format* function if *Format On Client* is set to *No*.

Editor is a custom user control responsible for rendering of the field value in "edit" mode. Standard editor with name *RichEditor* is available in each project. We recommend using the *TextMode* property with the corresponding data fields to enable "rich" editing of the field value.

Label \*  
Discount

Values of this field cannot be edited.

Show In Summary

HTML encoding

Data Format String  
{0:p}

Use data format strings compatible with *String.format* functions.

Format On Client

Now, when you create a new *Order Details* and select a product, *Unit Price*, *Quantity*, and *Discount* are automatically prepopulated, and *Discount* is formatted as a percentage.

**New Order Details**

Please fill this form and click OK button to create a new order details record. Click Cancel to return to the previous screen.

**New Order Details**  
Complete the form. Make sure to enter all required fields.

Product Name \* Mishi Kobe Niku

Unit Price \* \$97.00

Quantity \* 1

Discount \* 0.00 %

\* - indicates a required field

OK Cancel

### Add “Extended Price” Field

An *Extended Price* field is necessary to calculate the price of each line item. In *All Controllers*, select *OrderDetails*. Switch to *Fields* tab, and on the action bar, press *New | New Field*. Give this field the name “*ExtendedPrice*”, of *Type* “*Currency*”. Enable “*The value of this field is computed at run-time by SQL Expression*”, and paste in the code below in the *SQL Formula* field.

```
OrderDetails.UnitPrice*OrderDetails.Quantity*(1-OrderDetails.Discount)
```

**New Field**  
Specify field name, type, and data properties of the field.

*Server Default* is a SQL expression used as a field value when no value is provided for the field in INSERT and UPDATE statement.

Indicate that the field is *computed* if the field is not physically present in the dataset produced by controller's command. Computed field requires a mandatory *formula* that must be defined as a valid SQL expression. This expression is automatically inserted in SELECT statements when needed.

*Calculated* field values can be produced by business rule methods with attribute *ControllerAction*. You must list the context fields that will cause the calculation. Optional code formula is embedded into an automatically created business rule and is calculated whenever any context field is changed.

*Code Default* is an expression written in the programming language of your project. The expression is evaluated in an automatically created business rule to produce a default value

Name \*  
ExtendedPrice

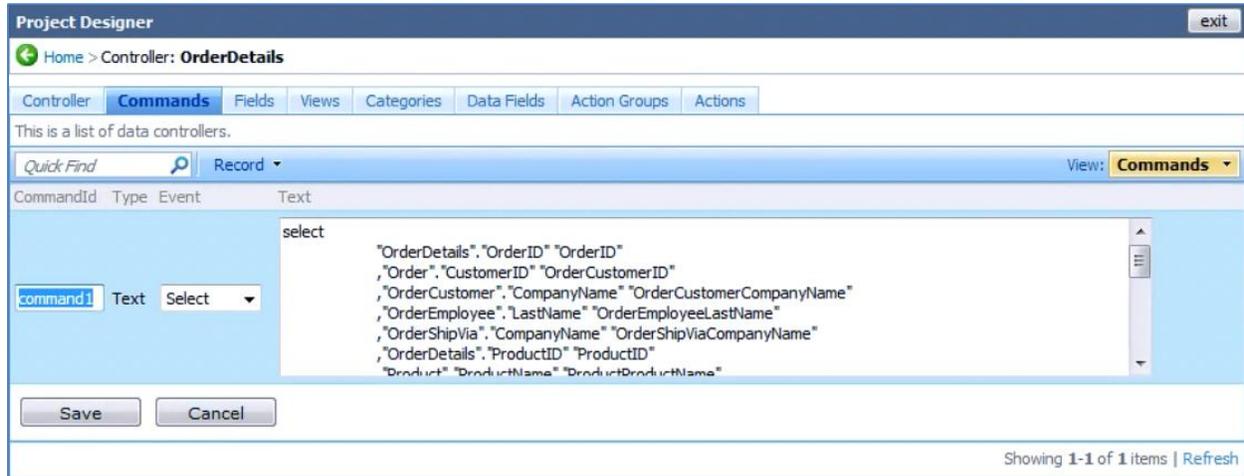
Type \*  
Currency

Allow null values.

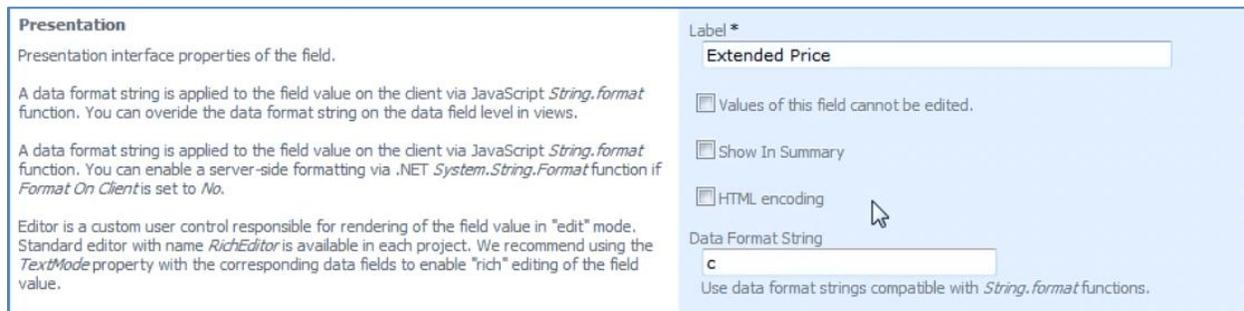
The value of this field is computed at run-time by SQL expression.

SQL Formula  
OrderDetails.UnitPrice\*OrderDetails.Quantity\*(1 - OrderDetails.Discount)

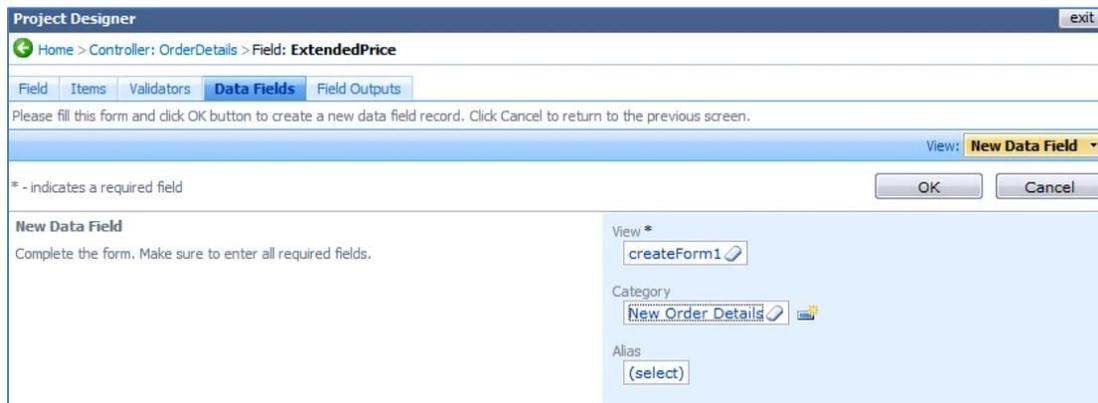
The *OrderDetails* alias used in the previous expression is referring to *command1* of the controller *OrderDetails*. The “select” statement provides a dictionary of fields for the data controller.



Scroll down to the *Presentation* section of the field, set *Label* as “Extended Price”, and enter “c” for the *Data Format String* to make sure the value appears as a currency. Enable “Values of this field cannot be edited”, as it is a calculated field. Save the field.



To make sure that the field is displayed in the application, you need to bind the new field to the data view. Select the field in the field list, and click on the *Data Fields* tab. This list is empty, as the field is not bound to any controller. On the action bar, press *New | New Data Field*. Bind this data field to “createForm1” *View*, and “New Order Details” *Category*.



Save, and create one more data field. This one will have *View* of “editForm1”, and *Category* of “Order Details”.

The screenshot shows the 'Project Designer' window with the 'Data Fields' tab selected. The 'New Data Field' dialog is open, showing the following configuration:

- View \***: editForm1
- Category**: Order Details
- Alias**: (select)

Buttons for 'OK' and 'Cancel' are visible at the top right of the dialog.

The last data field will have *View* of “grid1”, with no *Category*.

The screenshot shows the 'Project Designer' window with the 'Data Fields' tab selected. The 'New Data Field' dialog is open, showing the following configuration:

- View \***: grid1
- Category**: (select)
- Alias**: (select)

Buttons for 'OK' and 'Cancel' are visible at the top right of the dialog.

Now, if you regenerate and select an order in the *Order Form* page, you can see the *Extended Price* field displayed in the *Order Details* grid.

The screenshot shows the 'Order Form' page with the following details:

**Order Information:**

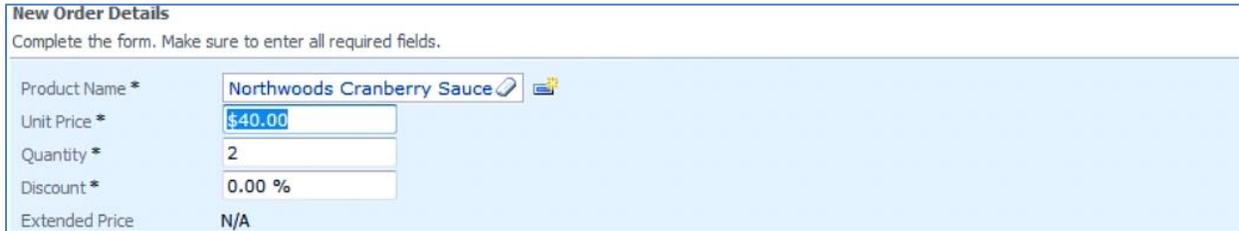
- Customer Company Name: Simons bistro
- Employee Last Name: King
- Order Date: 5/6/1998
- Required Date: 6/3/1998
- Shipped Date: N/A
- Ship Via Company Name: United Package
- Freight: \$18.44
- Ship Name: Simons bistro
- Ship Address: Vinbæltet 34
- Ship City: Kobenhavn
- Ship Region: N/A
- Ship Postal Code: 1734
- Ship Country: Denmark

**Product Grid:**

Product Name	Unit Price	Quantity	Discount	Order Customer Company Name	Order Employee Last Name	Order Ship Via Company Name	Product Category Name	Product Supplier Company Name	Extended Price
Pavlova	\$17.45	14	5.00 %	Simons bistro	King	United Package	Confections	Pavlova, Ltd.	\$232.09

## Update “Extended Price” Field

When you add a new *Order Detail*, *Extended Price* will show up as “N/A”. The calculation is executed on the server, as part of the *SQL Expression*. Let’s have the field be updated to reflect changes in *Product ID*, *Quantity*, *Price*, and *Discount*.



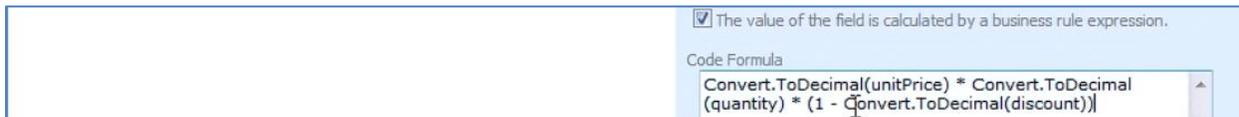
**New Order Details**  
Complete the form. Make sure to enter all required fields.

Product Name *	Northwoods Cranberry Sauce
Unit Price *	\$40.00
Quantity *	2
Discount *	0.00 %
Extended Price	N/A

In the list of *All Controllers*, select *OrderDetails* controller. In the *Fields* tab, select *ExtendedPrice* field. Press *Edit*, and indicate that “The value of the field is calculated by a business rule expression”. In the *Code Formula* box that appears, write in the following code below:

```
Convert.ToDecimal(unitPrice) * Convert.ToDecimal(quantity) * (1 - Convert.ToDecimal(discount))
```

This expression is reminiscent of SQL Formula, but it is written in the language that the project was generated in. In this case, it is Visual Basic.

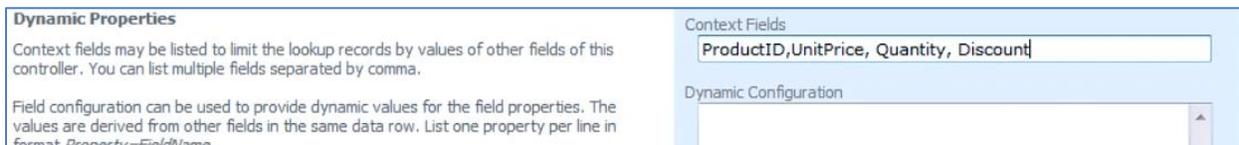


The value of the field is calculated by a business rule expression.

Code Formula

```
Convert.ToDecimal(unitPrice) * Convert.ToDecimal(quantity) * (1 - Convert.ToDecimal(discount))
```

The calculation will be performed when the specified *Context Fields* are modified. These *Context Fields* will be “ProductID, UnitPrice, Quantity, Discount”.



**Dynamic Properties**

Context fields may be listed to limit the lookup records by values of other fields of this controller. You can list multiple fields separated by comma.

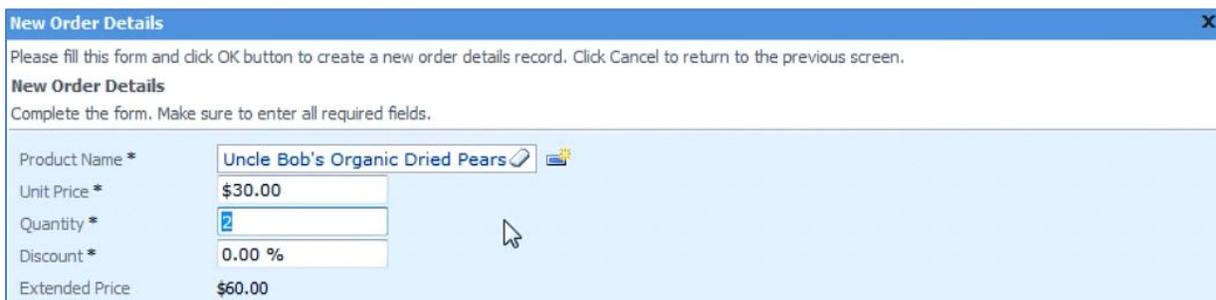
Field configuration can be used to provide dynamic values for the field properties. The values are derived from other fields in the same data row. List one property per line in format *Property=FieldName*.

Context Fields

ProductID,UnitPrice, Quantity, Discount

Dynamic Configuration

Now, when you create a new *Order Details* record, the *Extended Price* field will be updated when any of the fields are changed. The calculation will be performed when you hit *Enter* on your keyboard.



**New Order Details**  
Please fill this form and click OK button to create a new order details record. Click Cancel to return to the previous screen.

**New Order Details**  
Complete the form. Make sure to enter all required fields.

Product Name *	Uncle Bob's Organic Dried Pears
Unit Price *	\$30.00
Quantity *	2
Discount *	0.00 %
Extended Price	\$60.00

The source of the automatically generated business rules class that performs calculation of *Extended Price* is presented below.

### App Code/Rules/OrderDetails.Generated.vb

```
Namespace MyCompany.Rules

    Partial Public Class OrderDetailsBusinessRules
        Inherits MyCompany.Data.BusinessRules

        <ControllerAction("OrderDetails", "Calculate", "ExtendedPrice")> _
        Public Sub CalculateOrderDetails(ByVal orderID As Nullable(Of Integer), _
            ByVal orderCustomerID As String, _
            ByVal orderCustomerCompanyName As String, _
            ByVal orderEmployeeLastName As String, _
            ByVal orderShipViaCompanyName As String, _
            ByVal productID As Nullable(Of Integer), _
            ByVal productProductName As String, _
            ByVal productCategoryCategoryName As String, _
            ByVal productSupplierCompanyName As String, _
            ByVal unitPrice As Nullable(Of Decimal), _
            ByVal quantity As Nullable(Of Short), _
            ByVal discount As Nullable(Of Single))
            UpdateFieldValue("ExtendedPrice", Convert.ToDecimal(unitPrice) * _
                Convert.ToDecimal(quantity) * (1 - Convert.ToDecimal(discount)))
        End Sub

        <RowBuilder("OrderDetails", RowKind.New)> _
        Public Sub BuildNewOrderDetails()
            UpdateFieldValue("Quantity", 1)
            UpdateFieldValue("Discount", 0)
        End Sub
    End Class
End Namespace
```

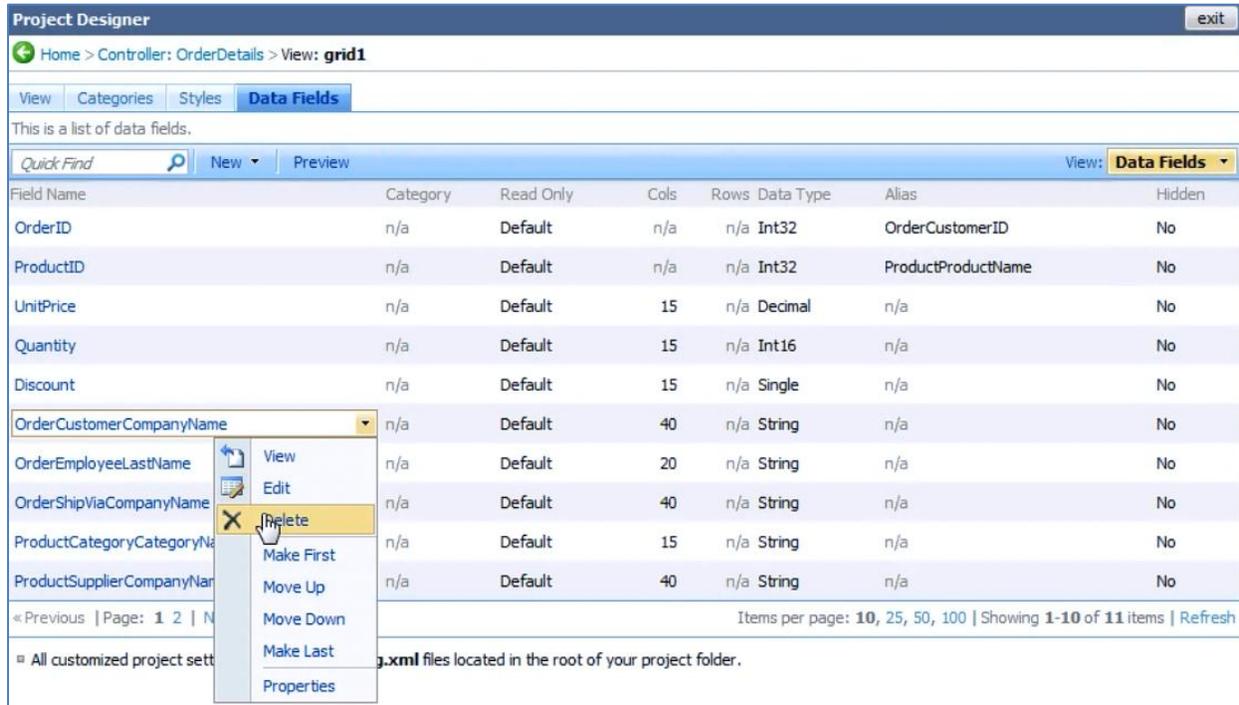
### App Code/Rules/OrderDetails.Generated.cs

```
namespace MyCompany.Rules
{
    public partial class OrderDetailsBusinessRules : MyCompany.Data.BusinessRules
    {
        [ControllerAction("OrderDetails", "Calculate", "ExtendedPrice")]
        public void CalculateOrderDetails(Nullable<int> orderID, string orderCustomerID,
            string orderCustomerCompanyName, string orderEmployeeLastName, string orderShipViaCompanyName,
            Nullable<int> productID, string productProductName, string productCategoryCategoryName,
            string productSupplierCompanyName, Nullable<decimal> unitPrice, Nullable<short> quantity,
            Nullable<float> discount)
        {
            UpdateFieldValue("ExtendedPrice", Convert.ToDecimal(unitPrice) *
                Convert.ToDecimal(quantity) * (1 - Convert.ToDecimal(discount)));
        }

        [RowBuilder("OrderDetails", RowKind.New)]
        public void BuildNewOrderDetails()
        {
            UpdateFieldValue("Quantity", 1);
            UpdateFieldValue("Discount", 0);
        }
    }
}
```

## Delete “Order XXXX” Fields from “grid1” View

Select *OrderDetails* from the list of *All Controllers*. Switch to the *Views* tab. Click on *grid1*, navigate to the *Data Fields* tab, and delete all the fields that start with the word “Order.” This includes *OrderCustomerCompanyName*, *OrderEmployeeLastName*, and *OrderShipViaCompanyName*.



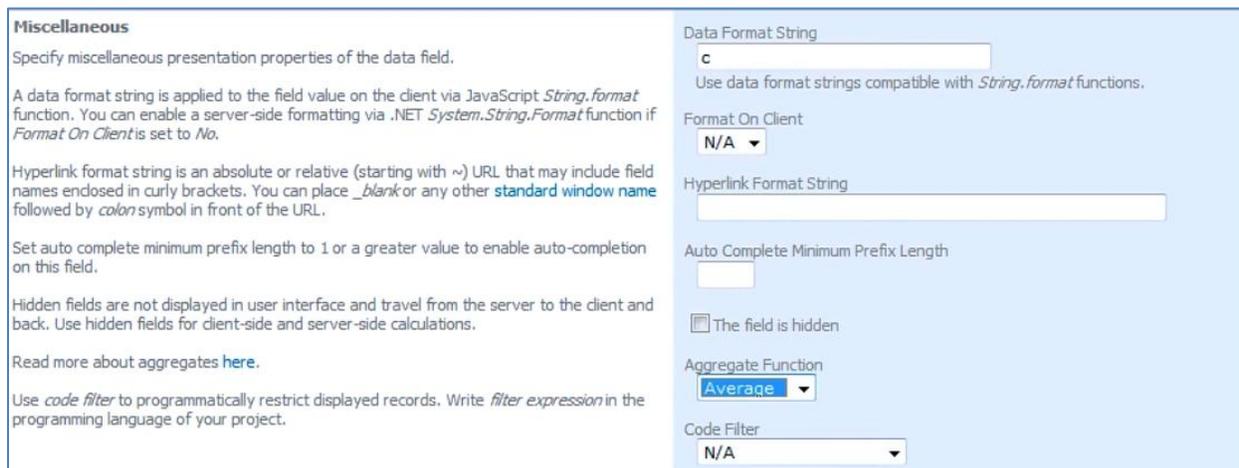
The screenshot shows the Project Designer interface for the *OrderDetails* controller, *grid1* view. The *Data Fields* tab is active, displaying a table of data fields. A context menu is open over the *OrderCustomerCompanyName* field, with the *Delete* option selected. The table lists the following fields:

Field Name	Category	Read Only	Cols	Rows	Data Type	Alias	Hidden
OrderID	n/a	Default	n/a	n/a	Int32	OrderCustomerID	No
ProductID	n/a	Default	n/a	n/a	Int32	ProductProductName	No
UnitPrice	n/a	Default	15	n/a	Decimal	n/a	No
Quantity	n/a	Default	15	n/a	Int16	n/a	No
Discount	n/a	Default	15	n/a	Single	n/a	No
OrderCustomerCompanyName	n/a	Default	40	n/a	String	n/a	No
OrderEmployeeLastName	n/a	Default	20	n/a	String	n/a	No
OrderShipViaCompanyName	n/a	Default	40	n/a	String	n/a	No
ProductCategoryCategoryName	n/a	Default	15	n/a	String	n/a	No
ProductSupplierCompanyName	n/a	Default	40	n/a	String	n/a	No

## Assign Aggregates

The new *Order Form* page is much cleaner, without unnecessary duplicate master fields in details. The next step would be to add a summary that shows total price, average discount, total quantity, and average price.

Select the *OrderDetails* controller from the *All Controllers* list. Switch to *Views* and select *grid1*. On the *Data Fields* tab, first select *Unit Price*. *Edit*, and change *Aggregate Function* to “Average”.



The screenshot shows the *Miscellaneous* properties window for a data field. The *Aggregate Function* is set to *Average*. The *Data Format String* is *c*. The *Format On Client* is *N/A*. The *Hyperlink Format String* is empty. The *Auto Complete Minimum Prefix Length* is empty. The *The field is hidden* checkbox is unchecked. The *Code Filter* is *N/A*.

Next, edit *Quantity* field and change *Aggregate Function* to “Sum”.

<p><b>Miscellaneous</b></p> <p>Specify miscellaneous presentation properties of the data field.</p> <p>A data format string is applied to the field value on the client via JavaScript <i>String.format</i> function. You can enable a server-side formatting via .NET <i>System.String.Format</i> function if <i>Format On Client</i> is set to <i>No</i>.</p> <p>Hyperlink format string is an absolute or relative (starting with ~) URL that may include field names enclosed in curly brackets. You can place <i>_blank</i> or any other <b>standard window name</b> followed by <i>colon</i> symbol in front of the URL.</p> <p>Set auto complete minimum prefix length to 1 or a greater value to enable auto-completion on this field.</p> <p>Hidden fields are not displayed in user interface and travel from the server to the client and back. Use hidden fields for client-side and server-side calculations.</p> <p>Read more about aggregates <a href="#">here</a>.</p> <p>Use <i>code filter</i> to programmatically restrict displayed records. Write <i>filter expression</i> in the programming language of your project.</p>	<p>Data Format String <input type="text"/></p> <p>Use data format strings compatible with <i>String.format</i> functions.</p> <p>Format On Client N/A ▼</p> <p>Hyperlink Format String <input type="text"/></p> <p>Auto Complete Minimum Prefix Length <input type="text"/></p> <p><input type="checkbox"/> The field is hidden</p> <p>Aggregate Function Sum ▼</p> <p>Code Filter N/A ▼</p>
--	--

Edit *Discount* field, and change *Aggregate Function* to “Average”.

<p><b>Miscellaneous</b></p> <p>Specify miscellaneous presentation properties of the data field.</p> <p>A data format string is applied to the field value on the client via JavaScript <i>String.format</i> function. You can enable a server-side formatting via .NET <i>System.String.Format</i> function if <i>Format On Client</i> is set to <i>No</i>.</p> <p>Hyperlink format string is an absolute or relative (starting with ~) URL that may include field names enclosed in curly brackets. You can place <i>_blank</i> or any other <b>standard window name</b> followed by <i>colon</i> symbol in front of the URL.</p> <p>Set auto complete minimum prefix length to 1 or a greater value to enable auto-completion on this field.</p> <p>Hidden fields are not displayed in user interface and travel from the server to the client and back. Use hidden fields for client-side and server-side calculations.</p> <p>Read more about aggregates <a href="#">here</a>.</p> <p>Use <i>code filter</i> to programmatically restrict displayed records. Write <i>filter expression</i> in the programming language of your project.</p>	<p>Data Format String <input type="text"/></p> <p>Use data format strings compatible with <i>String.format</i> functions.</p> <p>Format On Client N/A ▼</p> <p>Hyperlink Format String <input type="text"/></p> <p>Auto Complete Minimum Prefix Length <input type="text"/></p> <p><input type="checkbox"/> The field is hidden</p> <p>Aggregate Function Average ▼</p> <p>Code Filter N/A ▼</p>
--	--

Lastly, the *ExtendedPrice* field will have *Aggregate Function* of “Sum”.

<p><b>Miscellaneous</b></p> <p>Specify miscellaneous presentation properties of the data field.</p> <p>A data format string is applied to the field value on the client via JavaScript <i>String.format</i> function. You can enable a server-side formatting via .NET <i>System.String.Format</i> function if <i>Format On Client</i> is set to <i>No</i>.</p> <p>Hyperlink format string is an absolute or relative (starting with ~) URL that may include field names enclosed in curly brackets. You can place <i>_blank</i> or any other <b>standard window name</b> followed by <i>colon</i> symbol in front of the URL.</p> <p>Set auto complete minimum prefix length to 1 or a greater value to enable auto-completion on this field.</p> <p>Hidden fields are not displayed in user interface and travel from the server to the client and back. Use hidden fields for client-side and server-side calculations.</p> <p>Read more about aggregates <a href="#">here</a>.</p> <p>Use <i>code filter</i> to programmatically restrict displayed records. Write <i>filter expression</i> in the programming language of your project.</p>	<p>Data Format String <input type="text"/></p> <p>Use data format strings compatible with <i>String.format</i> functions.</p> <p>Format On Client N/A ▼</p> <p>Hyperlink Format String <input type="text"/></p> <p>Auto Complete Minimum Prefix Length <input type="text"/></p> <p><input type="checkbox"/> The field is hidden</p> <p>Aggregate Function Sum ▼</p> <p>Code Filter N/A ▼</p>
--	--

Below, you can see the *Order Details* list with aggregates at the bottom. These aggregates will change to reflect any changes as you navigate between orders, change order details, or filter order details.

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Chang	\$19.00	10	15.00 %	Beverages	Exotic Liquids	\$161.50
Spegesild	\$12.00	30	15.00 %	Seafood	Lyngbysild	\$306.00
Lakkalkööri	\$18.00	2	15.00 %	Beverages	Karikki Oy	\$30.60
Avg: \$16.33		Sum: 42	Avg: 15.00 %		Sum: \$498.10	

## Total and Subtotal

### SQL Expression for Subtotal

From *All Controllers*, select *Orders*. Switch to *Fields*, and on the action bar, press *New | New Field*. *Field Name* is "Subtotal", of *Type* "Currency". Enable "The value of this field is computed at run-time by SQL Expression". In the *SQL Formula* field that appears, type the expression below:

```
Select sum(unitprice*quantity*(1-discount)) from "order details"
where "Order Details".OrderID = Orders.OrderID
```

This will be pasted verbatim into the output expression which retrieves values for the *Orders* table.

**New Field**  
Specify field name, type, and data properties of the field.

*Server Default* is a SQL expression used as a field value when no value is provided for the field in INSERT and UPDATE statement.

Indicate that the field is *computed* if the field is not physically present in the dataset produced by controller's command. Computed field requires a mandatory *formula* that must be defined as a valid SQL expression. This expression is automatically inserted in SELECT statements when needed.

*Calculated* field values can be produced by business rule methods with attribute *ControllerAction*. You must list the context fields that will cause the calculation. Optional code formula is embedded into an automatically created business rule and is calculated whenever any context field is changed.

*Code Default* is an expression written in the programming language of your project. The expression is evaluated in an automatically created business rule to produce a default value for the field before it is presented in the user interface.

Name \*

Type \*

Allow null values.

The value of this field is computed at run-time by SQL expression.

SQL Formula

The *Label* field will be "Subtotal", enable "Values of this field cannot be edited", and type "c" in *Data Format String*.

**Presentation**  
Presentation interface properties of the field.

A data format string is applied to the field value on the client via JavaScript *String.format* function. You can override the data format string on the data field level in views.

A data format string is applied to the field value on the client via JavaScript *String.format* function. You can enable a server-side formatting via *.NET System.String.Format* function if *Format On Client* is set to *No*.

Editor is a custom user control responsible for rendering of the field value in "edit" mode. Standard editor with name *RichEditor* is available in each project. We recommend using the *TextMode* property with the corresponding data fields to enable "rich" editing of the field value.

Label \*

Values of this field cannot be edited.

Show In Summary

HTML encoding

Data Format String

Use data format strings compatible with *String.format* functions.

## Add Business Rules to “Orders” Controller and Code Expression for “Subtotal” Field

The *Subtotal* field is now present in the application. However, it does not update to reflect changes in the *Order Details*. This can be solved by adding a business rule to *Orders* controller and adding a code expression for *Subtotal* that will use this rule to calculate the subtotal.

Select the *Orders* controller from *All Controllers* list. Edit the controller, and in the *Handler* field, type “OrdersBusinessRules”.

The screenshot shows the Project Designer window for the 'Orders' controller. The 'General' tab is active, showing the Controller Name as 'Orders' and the Handler as 'OrdersBusinessRules'. The 'Miscellaneous' tab shows the Conflict Detection strategy set to 'Overwrite Changes'. The 'Business Rules' tab is also visible, showing the Handler field.

Regenerate the project, and open it in *Microsoft Visual Studio* or *Visual Web Developer*. Navigate to *App\_Code | Rules | OrdersBusinessRules.vb*. Enter the *CalculateOrderDetailsTotal* function.

### App\_Code/Rules/OrdersBusinessRules.vb

```
Imports MyCompany.Data
Imports System
Imports System.Collections.Generic
Imports System.Data
Imports System.Linq

Namespace MyCompany.Rules

    Partial Public Class OrdersBusinessRules
        Inherits MyCompany.Data.BusinessRules

        Public Function CalculateOrderDetailsTotal(ByRef orderID As Nullable(Of Integer)) As Decimal
            Using calc As SqlText = New SqlText( _
                "select sum(unitprice * quantity * (1 - discount)) from [Order Details] where OrderID=@OrderID")
                calc.AddParameter("@OrderID", orderID)
                Dim total As Object = calc.ExecuteScalar()
                If DBNull.Value.Equals(total) Then
                    Return 0
                Else
                    Return Convert.ToDecimal(total)
                End If
            End Using
        End Function
    End Class
End Namespace
```

### App Code/Rules/OrdersBusinessRules.cs

```
using MyCompany.Data;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;

namespace MyCompany.Rules
{
    public partial class OrdersBusinessRules : MyCompany.Data.BusinessRules
    {
        public decimal CalculateOrderDetailsTotal(int? orderID)
        {
            using (SqlText calc = new SqlText(@"select sum(unitprice * quantity * (1 - discount)) from
[Order Details] where OrderID= @OrderID"))
            {
                calc.AddParameter("@OrderID", orderID);
                object total = calc.ExecuteScalar();
                if (DBNull.Value.Equals(total))
                    return 0;
                else
                    return Convert.ToDecimal(total);
            }
        }
    }
}
```

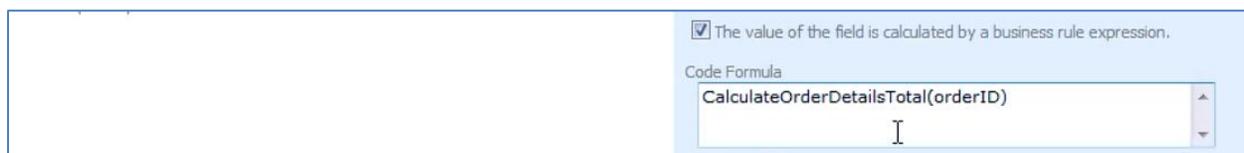
This function uses *SqlText* class to create an instance of a query connected to the project's database. This simple query selects a sum of *UnitPrice* multiplied by *Quantity* multiplied by one minus the *Discount*. Don't forget to save the file.

Note that *SqlText* utility class is generated as a part of the code base of your application. It uses the default database connection string and ADO.NET to execute the query.

Switch to the *Designer*, navigate to the *Fields* tab of the *Orders* controller, and select *Subtotal*. Edit, and enable "The value of this field is calculated by a business rule expression". In the *Code Formula* field that appears, type in the code below:

```
CalculateOrderDetailsTotal(orderID)
```

This is the method that was defined in *Visual Studio*.



To make sure that the calculation will occur when details are changed, change *Context Fields* to "OrderDetails".

**Dynamic Properties**  
Context fields may be listed to limit the lookup records by values of other fields of this controller. You can list multiple fields separated by comma.  
Field configuration can be used to provide dynamic values for the field properties. The values are derived from other fields in the same data row. List one property per line in format *Property=FieldName*.

Context Fields  
OrderDetails

Dynamic Configuration

### Add Total Field, Configure SQL Expression and Context Fields

To handle the *Total* calculation, you will need to configure an SQL expression similar to the one used in *Subtotal*, except that *Freight* will be included. From *All Controllers*, select *Orders*, and switch to *Fields* tab. Press *New | New Field*. Give this field the *Name* of "Total", of *Type* "Currency". Enable "The value of this field is computed at run-time by SQL Expression", and in the *SQL Formula*, type in the following expression:

```
(Select sum(unitprice*quantity*(1-discount)) from "order details" where "Order Details".OrderID = Orders.OrderID) + Orders.Freight
```

Also, enable "The value of the field is calculated by a business rule expression", and type in:

```
CalculateOrderDetailsTotal(orderID) + freight
```

Controller Commands **Fields** Views Categories Data Fields Action Groups Actions

Please fill this form and click OK button to create a new field record. Click Cancel to return to the previous screen.

View: **New Field**

\* - indicates a required field

**New Field**  
Specify field name, type, and data properties of the field.

*Server Default* is a SQL expression used as a field value when no value is provided for the field in INSERT and UPDATE statement.

Indicate that the field is *computed* if the field is not physically present in the dataset produced by controller's command. Computed field requires a mandatory *formula* that must be defined as a valid SQL expression. This expression is automatically inserted in SELECT statements when needed.

*Calculated* field values can be produced by business rule methods with attribute ControllerAction. You must list the context fields that will cause the calculation. Optional code formula is embedded into an automatically created business rule and is calculated whenever any context field is changed.

*Code Default* is an expression written in the programming language of your project. The expression is evaluated in an automatically created business rule to produce a default value for the field before it is presented in the user interface.

The field must be marked as *on-demand* if the field is a large binary object (BLOB) or text in order to speed up record retrieval.

Name \*  
Total

Type \*  
Currency

Allow null values.

The value of this field is computed at run-time by SQL expression.

SQL Formula  
(select sum(unitprice\*quantity\*(1-discount)) from "order details" where "Order Details".OrderID = Orders.OrderID) + Orders.Freight

The value of the field is calculated by a business rule expression.

Code Formula  
CalculateOrderDetailsTotal(orderID) + freight

OK Cancel

The *Label* will be “Total”, and *Data Format String* is “c”. Enable “Values of this field cannot be edited”.

**Presentation**  
Presentation interface properties of the field.

A data format string is applied to the field value on the client via JavaScript *String.format* function. You can override the data format string on the data field level in views.

A data format string is applied to the field value on the client via JavaScript *String.format* function. You can enable a server-side formatting via *.NET System.String.Format* function if *Format On Client* is set to *No*.

Editor is a custom user control responsible for rendering of the field value in "edit" mode. Standard editor with name *RichEditor* is available in each project. We recommend using the *TextMode* property with the corresponding data fields to enable "rich" editing of the field value.

Label \*  
Total

Values of this field cannot be edited

Show In Summary

HTML encoding

Data Format String  
c

Use data format strings compatible with *String.format* functions.

Format On Client

Editor

In the *Context Fields*, type “OrderDetails, Freight”.

**Dynamic Properties**  
Context fields may be listed to limit the lookup records by values of other fields of this controller. You can list multiple fields separated by comma.

Context Fields  
OrderDetails, Freight

Now we need to bind the field *Total* to the views. Click on the field, and switch to *Data Fields* tab. On the action bar, press *New | New Data Field*. For *View*, select “editForm1”. *Category* will be “Orders.”

Field Items Validators **Data Fields** Field Outputs

Please fill this form and click OK button to create a new data field record. Click Cancel to return to the previous screen.

View: **New Data Field**

\* - indicates a required field

OK Cancel

**New Data Field**  
Complete the form. Make sure to enter all required fields.

View \*  
editForm1

Category  
Orders

Alias  
(select)

Create another field. The *View* will be “grid1”, with no *Category*.

**New Data Field**  
Complete the form. Make sure to enter all required fields.

View \*  
editForm1

Category  
(select)

Alias  
(select)

If you regenerate the application, you can see this new field in action. It will calculate the total, including the cost of freight for the order.

Home > Order Form  
**Order Form**

Please review orders information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

Record ▾ View: **Review Orders** ▾

⬆ ⬇ \* - indicates a required field

**Orders**  
 These are the fields of the orders record that can be edited.

Customer Company Name: Richter Supermarkt  
 Employee Last Name: Fuller  
 Order Date: 11/15/2010  
 Required Date:   
 Shipped Date:   
 Ship Via Company Name: N/A  
 Freight: 10

Ship Name: Michael Holz

Ship Address: Grenzacherweg 237  
 Ship City: Genève  
 Ship Region:   
 Ship Postal Code: 1203

Ship Country: Switzerland  
 Subtotal: \$285.00  
 Total: \$295.00

⬆ ⬇ \* - indicates a required field

Quick Find 🔍 New Order Details Actions ▾ Report ▾

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Uncle Bob's Organic Dried Pears	\$30.00	10	5.00 %	Produce	Grandma Kelly's Homestead	\$285.00
Avg: \$30.00		Sum: 10	Avg: 5.00 %			Sum: \$285.00

### Enable Sorting and Filtering

The new *Subtotal* and *Total* fields do not allow sorting or filtering, unlike the other fields in the view. Let's enable this feature. Select the *Orders* controller from the list of *All Controllers*. Switch to *Fields*, and select *Subtotal*. Enable "Allow Query-by-example" and "Allow Sorting".

**Miscellaneous**  
 Specify if query-by-example and sorting is enabled in the context menu of the field when presented in grid views.

Allow Query-by-Example  
 Allow Sorting

Perform the same operation with *Total* field.

**Miscellaneous**  
 Specify if query-by-example and sorting is enabled in the context menu of the field when presented in grid views.

Allow Query-by-Example  
 Allow Sorting

### Calculating Freight

The calculation will analyze *Order ID* and current *Freight* value. If the order total is greater than \$100, then *Freight* will be \$19.95 flat. Otherwise, *Freight* is \$3.95. User can also override the *Freight* value.

Below is the updated version of the *Orders* business rules class. There is an added method called *CalculateFreight*. It takes nullable integers *orderID* and *freight*, and returns a decimal value. It will call *CalculateOrderDetailsTotal* method. If *Freight* is equal to blank, 0, 3.95, or 19.95, then it will be returned as 19.95 for *Total* greater than \$100, or 3.95 for *Total* under \$100. If the conditions are not met, then *Freight* will not be affected.

Modify *OrdersBusinessRules.vb(cs)* to support the calculation of freight. The sample implementation of *CalculateFreight* is presented next.

*App Code/Rules/OrdersBusinessRules.vb*

```
Namespace MyCompany.Rules

    Partial Public Class OrdersBusinessRules
        Inherits MyCompany.Data.BusinessRules

        Public Function CalculateOrderDetailsTotal(ByRef orderID As Nullable(Of Integer)) As Decimal
            Using calc As SqlText = New SqlText( _
                "select sum(unitprice * quantity * (1 - discount)) from [Order Details] where OrderID=@OrderID")
                calc.AddParameter("@OrderID", orderID)
                Dim total As Object = calc.ExecuteScalar()
                If DBNull.Value.Equals(total) Then
                    Return 0
                Else
                    Return Convert.ToDecimal(total)
                End If
            End Using
        End Function

        Public Function CalculateFreight(ByRef orderID As Nullable(Of Integer), _
            ByRef freight As Nullable(Of Decimal)) As Decimal
            Dim total As Decimal = CalculateOrderDetailsTotal(orderID)
            If Not freight.HasValue Or freight.Value = 0 Or freight.Value = 3.95 Or _
                freight.Value = 19.95 Then
                If total >= 100 Then
                    Return 19.95
                Else
                    Return 3.95
                End If
            Else
                Return freight.Value
            End If
        End Function
    End Class
End Namespace
```

## App Code/Rules/OrdersBusinessRules.vs

```
namespace MyCompany.Rules
{
    public partial class OrdersBusinessRules : MyCompany.Data.BusinessRules
    {
        public decimal CalculateOrderDetailsTotal(int? orderID)
        {
            using (SqlText calc = new SqlText(@"select sum(unitprice * quantity * (1 - discount)) from
[Order Details] where OrderID= @OrderID"))
            {
                calc.AddParameter("@OrderID", orderID);
                object total = calc.ExecuteScalar();
                if (DBNull.Value.Equals(total))
                    return 0;
                else
                    return Convert.ToDecimal(total);
            }
        }

        public decimal CalculateFreight(int? orderID, decimal? freight)
        {
            decimal total = CalculateOrderDetailsTotal(orderID);
            if (!freight.HasValue || freight.Value == 0 || freight.Value == 3.95m ||
                freight.Value == 19.95m)
            {
                if (total > 100)
                    return 19.95m;
                else
                    return 3.95m;
            }
            else
                return freight.Value;
        }
    }
}
```

Go back to the *Designer*, and select *Orders* from the list *All Controllers*. Switch to *Fields* tab, and select *Freight*. Enable “The value of the field is calculated by a business rule expression”, and in the *Code Formula* field that appears, type the following code:

```
CalculateFreight(orderID, freight)
```

<p>whenever any context field is changed.</p> <p><i>Code Default</i> is an expression written in the programming language of your project. The expression is evaluated in an automatically created business rule to produce a default value for the field before it is presented in the user interface.</p> <p>The field must be marked as <i>on-demand</i> if the field is a large binary object (BLOB) or text in order to speed up record retrieval.</p>	<p><input checked="" type="checkbox"/> The value of the field is calculated by a business rule expression.</p> <p>Code Formula <input type="text" value="CalculateFreight(orderID, freight)"/></p> <p>Server Default <input type="text" value="((0))"/></p> <p>Code Default <input type="text"/></p>
---	--

In *Context Fields*, enter “OrderDetails”.

<p><b>Dynamic Properties</b></p> <p>Context fields may be listed to limit the lookup records by values of other fields of this controller. You can list multiple fields separated by comma.</p>	<p>Context Fields <input type="text" value="OrderDetails"/></p>
---	---

If you save and regenerate the application, you can see *Freight* field in action. When you change *Freight* to 0, and hit *Enter* on your keyboard, the field will be calculated.

**Orders**  
These are the fields of the orders record that can be edited.

Customer Company Name: Richter Supermarkt  
Employee Last Name: Fuller  
Order Date: 11/15/2010  
Required Date:   
Shipped Date:   
Ship Via Company Name: N/A  
Freight: \$19.95

Ship Name: Michael Holz

Ship Address: Grenzacherweg 237  
Ship City: Genève  
Ship Region:   
Ship Postal Code: 1203

Ship Country: Switzerland  
Subtotal: \$285.00  
Total: \$285.00

↑ ↓ \* - indicates a required field

OK Delete Cancel

Quick Find New Order Details Actions Report

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Uncle Bob's Organic Dried Pears	\$30.00	10	5.00 %	Produce	Grandma Kelly's Homestead	\$285.00
Avg: \$30.00		Sum: 10	Avg: 5.00 %		Sum: \$285.00	

If you were to change the size of an *Order Detail* so that the *Subtotal* is under \$100, *Freight* will change to \$3.95.

**Orders**  
These are the fields of the orders record that can be edited.

Customer Company Name: Richter Supermarkt  
Employee Last Name: Fuller  
Order Date: 11/15/2010  
Required Date:   
Shipped Date:   
Ship Via Company Name: N/A  
Freight: \$3.95

Ship Name: Michael Holz

Ship Address: Grenzacherweg 237  
Ship City: Genève  
Ship Region:   
Ship Postal Code: 1203

Ship Country: Switzerland  
Subtotal: \$57.00  
Total: \$76.95

↑ ↓ \* - indicates a required field

OK Delete Cancel

Quick Find New Order Details Edit Delete Actions Report

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Uncle Bob's Organic Dried Pears	\$30.00	2	5.00 %	Produce	Grandma Kelly's Homestead	\$57.00
Avg: \$30.00		Sum: 2	Avg: 5.00 %		Sum: \$57.00	

Let's take a quick look at the *Orders* business rules class that was automatically created by the code generator for us. You can see that we have a partial class *OrdersBusinessRules* with method *CalculateOrders* adorned with attributes *ControllerAction*, which respond to *Calculate* action. The method calculates *Freight*, *Subtotal*, and *Total* fields by calling *CalculateOrderDetailsTotal* and *CalculateFreight* with *orderID* passed as an argument.

App Code/Rules/Orders.Generated.vb

```
Imports MyCompany.Data
Imports System
Imports System.Collections.Generic
Imports System.Data
Imports System.Linq
Imports System.Text.RegularExpressions
Imports System.Web

Namespace MyCompany.Rules

    Partial Public Class OrdersBusinessRules
        Inherits MyCompany.Data.BusinessRules

        <ControllerAction("Orders", "Calculate", "Freight"), _
        ControllerAction("Orders", "Calculate", "Subtotal"), _
        ControllerAction("Orders", "Calculate", "Total")> _
        Public Sub CalculateOrders( _
            ByVal orderID As Nullable(Of Integer), _
            ByVal customerID As String, _
            ByVal customerCompanyName As String, _
            ByVal employeeID As Nullable(Of Integer), _
            ByVal employeeLastName As String, _
            ByVal orderDate As Nullable(Of DateTime), _
            ByVal requiredDate As Nullable(Of DateTime), _
            ByVal shippedDate As Nullable(Of DateTime), _
            ByVal shipVia As Nullable(Of Integer), _
            ByVal shipViaCompanyName As String, _
            ByVal freight As Nullable(Of Decimal), _
            ByVal shipName As String, _
            ByVal shipAddress As String, _
            ByVal shipCity As String, _
            ByVal shipRegion As String, _
            ByVal shipPostalCode As String, _
            ByVal shipCountry As String, _
            ByVal subtotal As Nullable(Of Decimal), _
            ByVal total As Nullable(Of Decimal))
            UpdateFieldValue("Freight", CalculateFreight(orderID, freight))
            UpdateFieldValue("Subtotal", CalculateOrderDetailsTotal(orderID))
            UpdateFieldValue("Total", CalculateOrderDetailsTotal(orderID) + freight)
        End Sub

        <RowBuilder("Orders", RowKind.New)> _
        Public Sub BuildNewOrders()
            UpdateFieldValue("OrderDate", DateTime.Now)
        End Sub
    End Class
End Namespace
```

## App\_Code/Rules/Orders.Generated.cs

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;
using MyCompany.Data;

namespace MyCompany.Rules
{
    public partial class OrdersBusinessRules : MyCompany.Data.BusinessRules
    {
        [ControllerAction("Orders", "Calculate", "Freight")]
        [ControllerAction("Orders", "Calculate", "Subtotal")]
        [ControllerAction("Orders", "Calculate", "Total")]
        public void CalculateOrders(
            Nullable<int> orderID,
            string customerID,
            string customerCompanyName,
            Nullable<int> employeeID,
            string employeeLastName,
            Nullable<DateTime> orderDate,
            Nullable<DateTime> requiredDate,
            Nullable<DateTime> shippedDate,
            Nullable<int> shipVia,
            string shipViaCompanyName,
            Nullable<decimal> freight,
            string shipName,
            string shipAddress,
            string shipCity,
            string shipRegion,
            string shipPostalCode,
            string shipCountry)
        {
            UpdateFieldValue("Freight", CalculateFreight(orderID, freight));
            UpdateFieldValue("Subtotal", CalculateOrderDetailsTotal(orderID));
            UpdateFieldValue("Total", CalculateOrderDetailsTotal(orderID) + freight);
        }

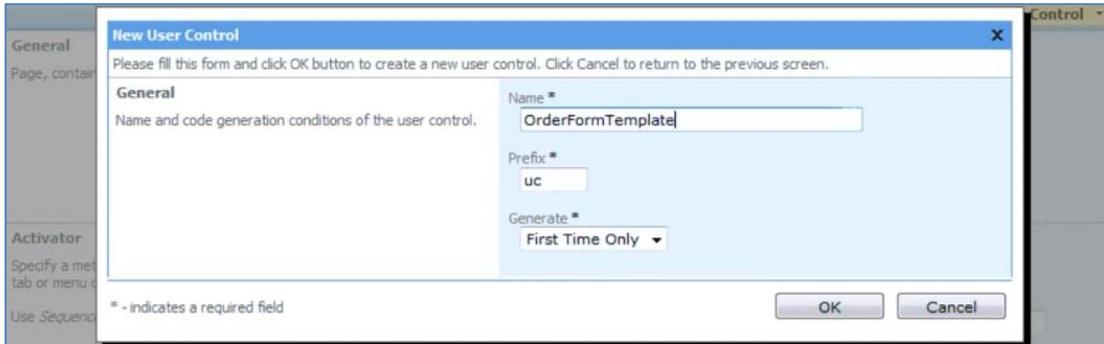
        [RowBuilder("Orders", RowKind.New)]
        public void BuildNewOrders()
        {
            UpdateFieldValue("OrderDate", DateTime.Now);
        }
    }
}
```

## Custom Form Template

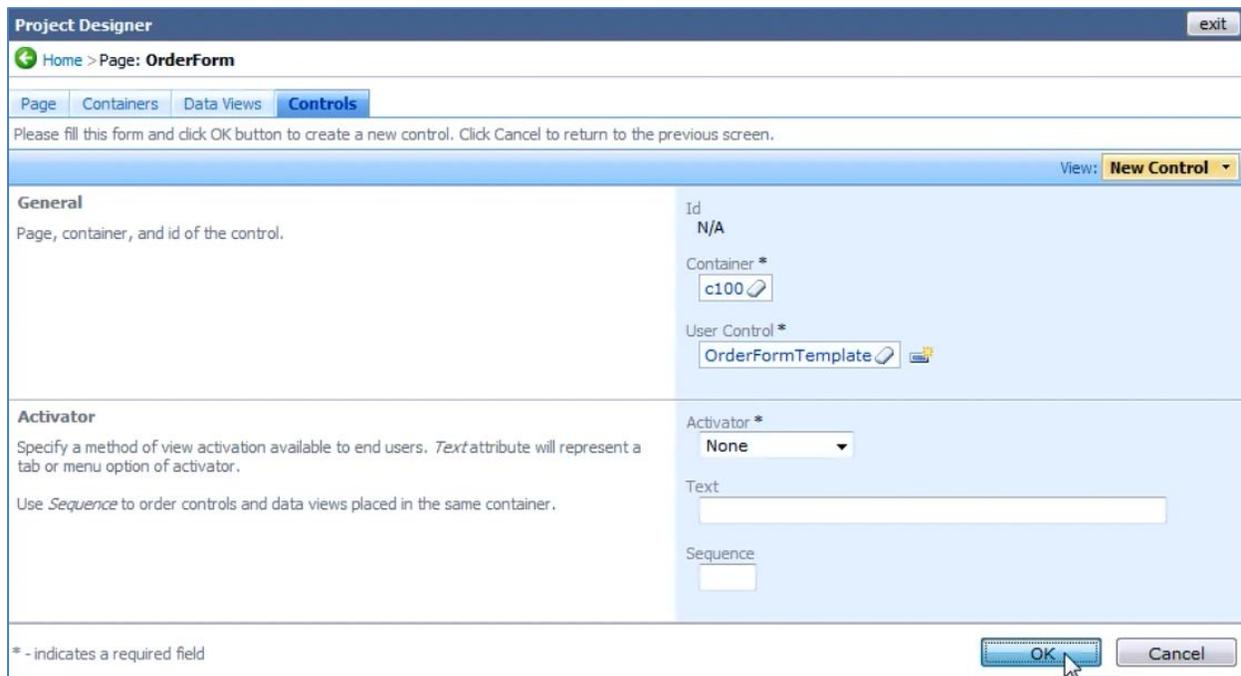
You will need to modify the form template, so that the *Order Form* is easier for the end user to interact with. First, you need to add *Order Form Template* user control to the page.

### Add “Order Form Template” User Control

In the *Designer*, click on the *All Pages* tab. Select “OrderForm”, and switch to *Controls* tab. On the action bar, press *New | New Control*. Press the *New User Control* icon next to the *User Control* field. It will have the *Name* of “OrderFormTemplate”.



Save, and this will insert the new *User Control* into the *Control*. Select “c100” for *Container*, and save.



### Define the Template Placeholder

Open the project in *Visual Studio* (or *Visual Web Developer*), and press the *Refresh* button. Navigate to *App\_Code/Controls/OrderFormTemplate.ascx*. Open this document, and format using *Edit | Format*

Document. Currently, there is just an *UpdatePanel* present, which can be eliminated. Use the template below:

App Code/Controls/OrderFormTemplate.ascx

```
<div id="FormTemplate1" runat="server">
  <div id="Orders_editForm1">
    <div class="FieldPlaceholder">
      {CustomerID}
    </div>
    <div class="FieldPlaceholder">
      {EmployeeID}
    </div>
    <div class="FieldPlaceholder">
      {ShipVia}
    </div>
    <div class="FieldPlaceholder">
      {OrderDate}
    </div>
    <div class="FieldPlaceholder">
      {Freight}
    </div>
    <div class="FieldPlaceholder">
      {Total}
    </div>
  </div>
</div>
```

There is a new element defined, *div* with *id* of "FormTemplate1". Underneath is another *div* element with *id* "Orders\_editForm1". This element instructs the client-side application to present the contents of *editForm1*, rendered by *Orders* data controller, using the template. Underneath this are several more *div* elements, of class "FieldPlaceholder". Inside each, there is just the field name in curly brackets, to get started.

If you were to save and refresh the application, only the field names will appear in brackets above the list.



This isn't quite the effect we're going for, so view code for the file by pressing the *View Code* button in the *Solution Explorer*, and add a line to the method.

*App\_Code/Controls/OrderFormTemplate.ascx.vb*

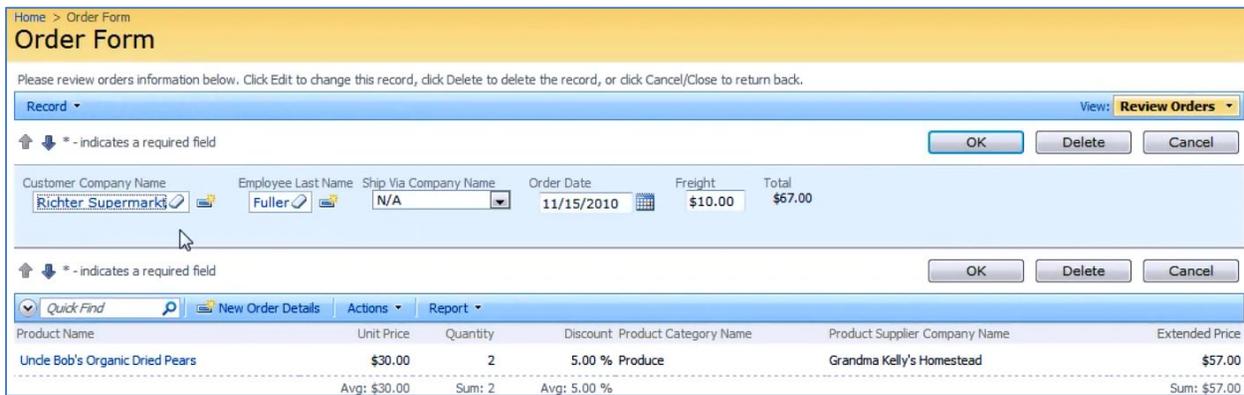
```
Partial Public Class Controls_OrderFormTemplate
    Inherits System.Web.UI.UserControl

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handles Me.Load
        FormTemplate1.Style("display") = "none"
    End Sub
End Class
```

*App\_Code/Controls/OrderFormTemplate.ascx.cs*

```
public partial class Controls_OrderFormTemplate : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        FormTemplate1.Style["display"] = "none";
    }
}
```

This line will dictate that *FormTemplate1* will have a special *Style* that changes "display" to "none", so that the template will not be displayed when the application runs. If you switch to *Design* mode, you can still see the controls and interact with them visually. Save, and refresh the web application. You can see that no field names in brackets will appear, and that only the fields specified in the template are presented in the detail view.

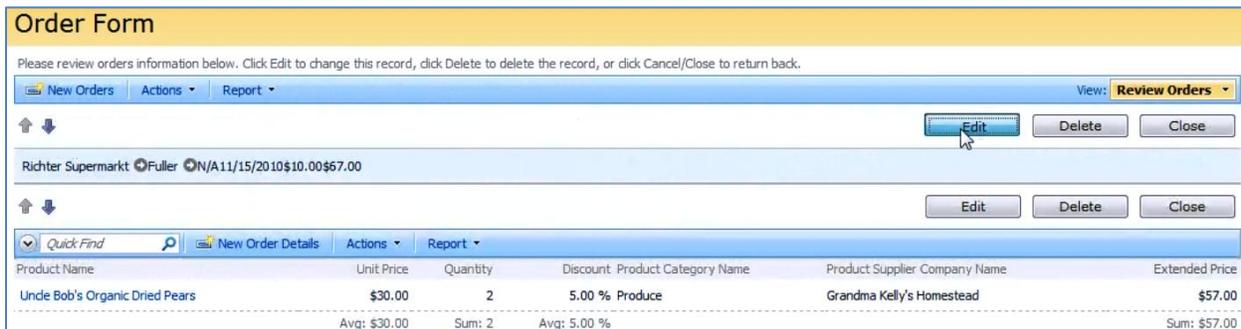


Let's make a more sophisticated design for the template, which includes the rest of the fields. In order to build a completely custom template and retain the data functionality of the client side library, you need to get rid of the labels. Switch back to *Visual Studio*, and add the class "DataOnly" to each field.

App Code/Controls/OrderFormTemplate.ascx

```
<div id="FormTemplate1" runat="server">
  <div id="Orders_editForm1">
    <div class="FieldPlaceholder DataOnly">
      {CustomerID}
    </div>
    <div class="FieldPlaceholder DataOnly">
      {EmployeeID}
    </div>
    <div class="FieldPlaceholder DataOnly">
      {ShipVia}
    </div>
    <div class="FieldPlaceholder DataOnly">
      {OrderDate}
    </div>
    <div class="FieldPlaceholder DataOnly">
      {Freight}
    </div>
    <div class="FieldPlaceholder DataOnly">
      {Total}
    </div>
  </div>
</div>
```

When you save and refresh the application, you can see that labels are no longer present, but the formatting is terribly off.



**Create Custom HTML Table Layout**

You will need to add a custom HTML table layout that uses field placeholders to position the data fields. The new layout code is displayed below.

Here is the new version of the template, which is much longer than the previous version. You can see that there is a *style* element with a few defined CSS rules, *.FieldLabel* and *.RightAlignedInputs*.

You can see that there are several *div* and *table* elements that hold all of the fields referenced in curly brackets.

## App\_Code/Controls/OrderFormTemplate.ascx

```
<%@ Control Language="VB" AutoEventWireup="false" CodeFile="OrderFormTemplate.ascx.vb"
    Inherits="Controls_OrderFormTemplate" %>
<style type="text/css">
    .FieldLabel
    {
        font-weight: bold;
        padding: 4px;
        width: 90px;
    }

    .RightAlignedInputs input
    {
        text-align: right;
    }
</style>
<div id="FormTemplate1" runat="server">
    <div id="Orders_editForm1">
        <table style="width: 100%">
            <tr>
                <td valign="top">
                    <table>
                        <tr>
                            <td class="FieldLabel">
                                Customer:
                            </td>
                            <td>
                                <div class="FieldPlaceholder DataOnly">
                                    {CustomerID}</div>
                                </td>
                            </tr>
                            <tr>
                            <td class="FieldLabel">
                                Employee:
                            </td>
                            <td>
                                <div class="FieldPlaceholder DataOnly">
                                    {EmployeeID}</div>
                                </td>
                            </tr>
                            <tr>
                            <td class="FieldLabel">
                                Order Date:
                            </td>
                            <td>
                                <div class="FieldPlaceholder DataOnly">
                                    {OrderDate}</div>
                                </td>
                            </tr>
                            <tr>
                            <td class="FieldLabel">
                                Required Date:
                            </td>
                            <td>
                                <div class="FieldPlaceholder DataOnly">
                                    {RequiredDate}</div>
                                </td>
                            </tr>
                            <tr>
                            <td class="FieldLabel">
                                Shipped Date:
                            </td>
                            <td>
                                <div class="FieldPlaceholder DataOnly">
                                    {ShippedDate}</div>
                                </td>
                            </tr>
                        </tr>
                    </table>
                </td>
            </tr>
        </table>
    </div>
</div>
```

```

        </table>
    </td>
    <td valign="top">
        <table style="float: right" class="RightAlignedInputs">
            <tr>
                <td class="FieldLabel">
                    Address:
                </td>
                <td>
                    <div class="FieldPlaceholder DataOnly" style="float: right">
                        {ShipAddress}</div>
                </td>
            </tr>
            <tr>
                <td class="FieldLabel">
                    City:
                </td>
                <td>
                    <div class="FieldPlaceholder DataOnly" style="float: right">
                        {ShipCity}</div>
                </td>
            </tr>
            <tr>
                <td class="FieldLabel">
                    Region:
                </td>
                <td>
                    <div class="FieldPlaceholder DataOnly" style="float: right">
                        {ShipRegion}</div>
                </td>
            </tr>
            <tr>
                <td class="FieldLabel">
                    Postal Code:
                </td>
                <td>
                    <div class="FieldPlaceholder DataOnly" style="float: right">
                        {ShipPostalCode}</div>
                </td>
            </tr>
            <tr>
                <td class="FieldLabel">
                    Ship Country:
                </td>
                <td>
                    <div class="FieldPlaceholder DataOnly" style="float: right">
                        {ShipCountry}</div>
                </td>
            </tr>
        </table>
    </td>
</tr>
<tr>
    <td colspan="2">
        {dv101Extender}
    </td>
</tr>
<tr>
    <td valign="bottom">
        <table>
            <tr>
                <td class="FieldLabel">
                    Ship Name:
                </td>
                <td>
                    <div class="FieldPlaceholder DataOnly">
                        {ShipName}</div>
                </td>
            </tr>
        </table>
    </td>

```

```

        <tr>
            <td class="FieldLabel">
                Ship Via:
            </td>
            <td>
                <div class="FieldPlaceholder DataOnly">
                    {ShipVia}</div>
                </td>
        </tr>
    </table>
</td>
<td align="right">
    <table style="margin-right: 4px;" class="RightAlignedInputs">
        <tr>
            <td class="FieldLabel">
                Subtotal:
            </td>
            <td align="right">
                <div class="FieldPlaceholder DataOnly" style="float: right">
                    {Subtotal}</div>
                </td>
        </tr>
        <tr>
            <td class="FieldLabel">
                Freight:
            </td>
            <td align="right">
                <div class="FieldPlaceholder DataOnly" style="float: right">
                    {Freight}</div>
                </td>
        </tr>
        <tr>
            <td class="FieldLabel">
                Total:
            </td>
            <td>
                <div class="FieldPlaceholder DataOnly" style="float: right">
                    {Total}</div>
                </td>
        </tr>
    </table>
</td>
</tr>
</table>
</div>
</div>

```

The C# version of the file will feature a different page directive:

```

<%@ Control Language="C#" AutoEventWireup="true" CodeFile="OrderFormTemplate.ascx.cs"
    Inherits="Controls_OrderFormTemplate" %>

```

Switch to *Design* view, and you can see how the layout appears. There is a label next to each field. Visual tools can be used to rearrange the fields to whatever order you would like.

<b>Customer:</b> {CustomerID}		<b>Address:</b> {ShipAddress}
<b>Employee:</b> {EmployeeID}		<b>City:</b> {ShipCity}
<b>Order Date:</b> {OrderDate}		<b>Region:</b> {ShipRegion}
<b>Required Date:</b> {RequiredDate}		<b>Postal Code:</b> {ShipPostalCode}
<b>Shipped Date:</b> {ShippedDate}		<b>Ship Country:</b> {ShipCountry}
{dv101Extender}		
		<b>Subtotal:</b> {Subtotal}
<b>Ship Name:</b> {ShipName}		<b>Freight:</b> {Freight}
<b>Ship Via:</b> {ShipVia}		<b>Total:</b> {Total}

One key element is the `{dv101Extender}` in the middle of the layout. This refers to *Details View* with ID of "dv101". Open the *Designer*, switch to *All Pages* tab, and click on the *OrderForm* page. If you switch to *Data Views* tab, you can see that "dv101" does exist, and it presents *OrderDetails*.

Home > Page: **OrderForm**

Page Containers **Data Views** Controls

This is a list of data views.

Quick Find  New View: **Data Views**

Id	Container	Seq	Controller	View	Summary	Activator	Text	Filter Source	Filter Fields
dv100	c100	n/a	Orders	grid1	No	None	n/a	n/a	n/a
dv101	c100	n/a	OrderDetails	grid1	No	None	n/a	dv100	OrderID

Save the template, and refresh the web application. Select an order and you can see the new template at work.

Home > Order Form

## Order Form

Please review orders information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

[New Orders](#) [Actions](#) [Report](#) View: **Review Orders**

<b>Customer:</b> Richter Supermarket	<b>Address:</b> Grenzacherweg 237
<b>Employee:</b> Fuller	<b>City:</b> Genève
<b>Order Date:</b> 11/15/2010	<b>Region:</b> N/A
<b>Required Date:</b> N/A	<b>Postal Code:</b> 1203
<b>Shipped Date:</b> N/A	<b>Ship Country:</b> Switzerland

Quick Find  [New Order Details](#) [Actions](#) [Report](#)

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Uncle Bob's Organic Dried Pears	\$30.00	2	5.00 % Produce		Grandma Kelly's Homestead	\$57.00
Avg: \$30.00		Sum: 2	Avg: 5.00 %		Sum: \$57.00	

**Subtotal:** \$57.00  
**Freight:** \$10.00  
**Total:** \$67.00

**Ship Name:** Michael Holz  
**Ship Via:** N/A

The *Customer*, *Employee*, and *Date* fields are presented on the left side. *Shipping Information* is displayed on the right side. The *Details* grid is automatically inserted in the next row of the template. *Ship Name* and *Ship Via* are displayed in the bottom left, and *Subtotal*, *Freight*, and *Total* are in the

bottom right, underneath the *Extended Price* row of *Order Details*. If you edit the record, you can see that the fields have modified lengths. If you use the up and down arrows to move through *Orders*, you can see the information change.

Record View: Review Orders

\* - indicates a required field

Customer: Richter Supermarkt  
 Employee: Fuller  
 Order Date: 11/15/2010  
 Required Date:  
 Shipped Date:

Address: Grenzacherweg 237  
 City: Genève  
 Region:  
 Postal Code: 1203  
 Ship Country: Switzerland

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Uncle Bob's Organic Dried Pears	\$30.00	2	5.00 %	Produce	Grandma Kelly's Homestead	\$57.00
Avg: \$30.00 Sum: 2 Avg: 5.00 %						Sum: \$57.00

Subtotal: \$57.00  
 Freight: \$10.00  
 Total: \$67.00

Ship Name: Michael Holz  
 Ship Via: N/A

If you have a lot of *Order Detail* records, you can sort and filter using the columns. You can also search specific products with *Quick Find*. The *Sum* will show a sum of the filtered fields, while *Subtotal* will be calculated for all fields relevant to the *Order*.

Record View: Review Orders

\* - indicates a required field

Customer: Rattlesnake Canyon Grocery  
 Employee: Davolio  
 Order Date: 5/6/1998  
 Required Date: 6/3/1998  
 Shipped Date:

Address: 2817 Milton Dr.  
 City: Albuquerque  
 Region: NM  
 Postal Code: 87110  
 Ship Country: USA

A filter has been applied. Product Supplier Company Name equals Grandma Kelly's Homestead.

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Grandma's Boysenberry Spread	\$25.00	1	2.00 %	Condiments	Grandma Kelly's Homestead	\$24.50
Uncle Bob's Organic Dried Pears	\$30.00	1	5.00 %	Produce	Grandma Kelly's Homestead	\$28.50
Northwoods Cranberry Sauce	\$40.00	2	10.00 %	Condiments	Grandma Kelly's Homestead	\$72.00
Avg: \$31.67 Sum: 4 Avg: 5.67 %						Sum: \$125.00

Subtotal: \$1,255.72  
 Freight: \$8.53  
 Total: \$1,264.25

Ship Name: Rattlesnake Canyon Grocery  
 Ship Via: United Package