

2011



COOKBOOK

Table of Contents

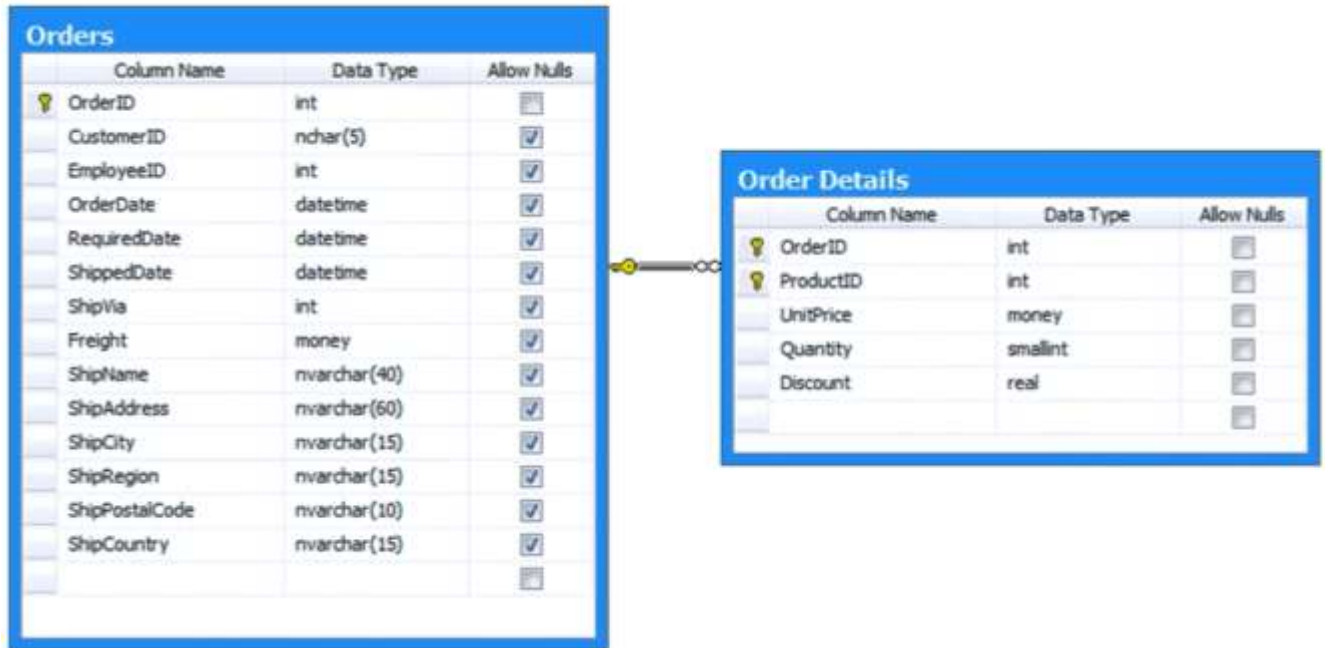
Order Form Page.....	3
Table Relationships	3
Objective	5
Sample.....	5
Implementation	5
Generate Northwind Sample	5
Add Page in Designer	7
Add Container to Page	9
Add Data View for “Orders”	9
Add Data View for “Order Details”	10
Customizing “Orders” Controller	11
Customizing “Order Details” Controller.....	18
Total and Subtotal.....	29
Calculating Freight	34
Custom Form Template	40
CreatedBy, CreatedOn, ModifiedBy, ModifiedOn	50
Many-To-Many Fields	58
Data Controller URL Parameters.....	61
Universal Lookup.....	63
Universal Lookup Database	63
Using the Standard Application	67
Modifying the Standard Application.....	69
View the Modifications.....	71
Further Modification.....	71
Role-based Security	74
Pages	74
Fields	77
Actions	78
Row-level Security.....	81
Role vs. Row-Level Security	81
Roles.....	82

Row-Level.....	82
Implementing Row-Level Security	82
Real World Example.....	82
Creating Roles	82
Creating User Accounts.....	83
Defining Role-Specific Views.....	85
Views in Action.....	88
Using OverrideWhen.....	89
Row-Level View Filters	90
Viewing the Results.....	91
Ideal Implementation.....	92
Custom Page Background	94
Creating Three Level Master Detail	97
Grouping Tabbed Data Views	103
Mixed Authentication	106
Chart View.....	108
Creating a Chart View	108
Displaying Multiple Values.....	112
Legend.....	113
Custom Charts.....	114
Standalone ASP.NET Membership Database	115
Learn More.....	119

Order Form Page

Table Relationships

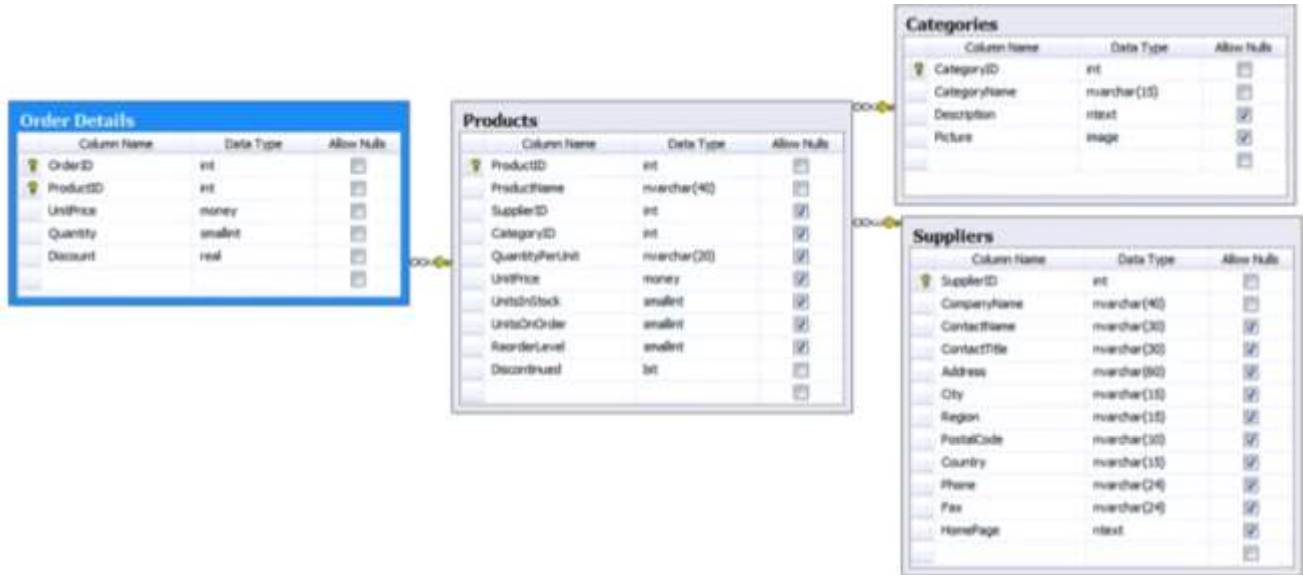
We have two tables, *Orders* and *Order Details*. Both tables are from the *Northwind* sample database. *Orders* is the master table, and *Order Details* is the details table.



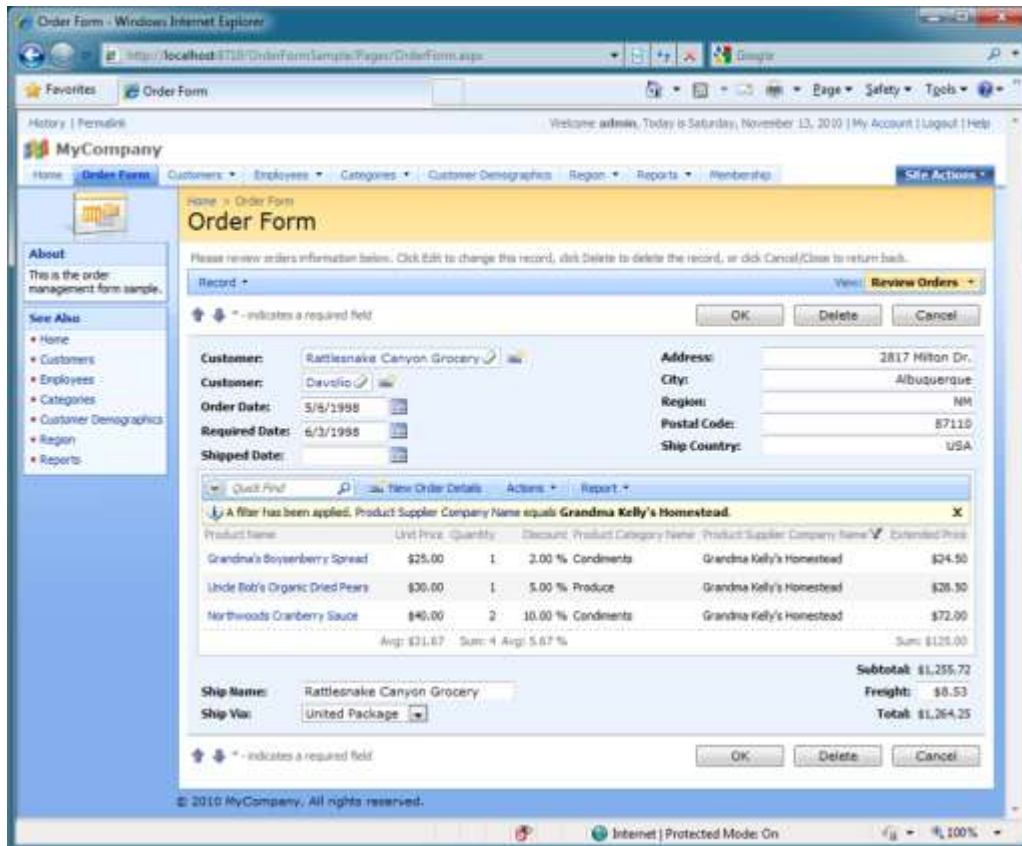
Each *Order* record references a *Customer*, an *Employee*, and a *Shipper*. We also know the *Order Date*, *Required Date*, *Shipped Date*, *Freight Amount*, and shipping information.



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.

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Cherg	\$19.00	10	15.00 %	Beverages	Beck's Liquids	\$361.00
Seagearl	\$12.00	30	15.00 %	Seafood	Lynghyllid	\$306.00
Lakkaikoon	\$18.00	2	15.00 %	Beverages	Karkki Oy	\$30.60
Avg: \$18.33 Sum: 42 Avg: 15.00 %						Sum: \$998.30

Subtotal: \$998.30
Freight: \$5.19
Total: \$104.29

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

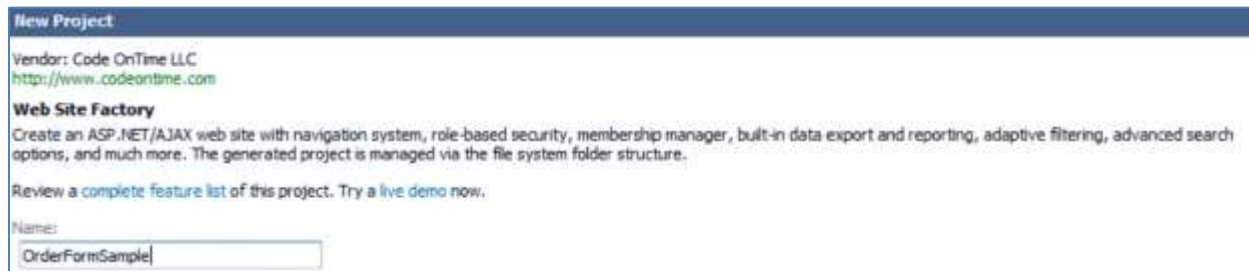
<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en> and download the database scripts.

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. The header includes the logo and navigation links for YouTube, Blog, and Newsgroup. The main content area is titled 'My Subscriptions' and 'My Projects'. Under 'My Projects', there is a 'New Project' section with two options: 'Web App Factory' and 'Web Site Factory'. The 'Web Site Factory' option is highlighted with a mouse cursor. Below the options, there are three bullet points providing recommendations and instructions for using the generator.

Give it the name of "OrderFormSample".



The screenshot shows the 'New Project' page for the 'Web Site Factory' project. The page includes the vendor information (Code On Time LLC) and a description of the project. A 'Name' field is present, which has been filled with the text '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.



The screenshot shows the 'Database Connection' page. It includes a 'Data Provider' dropdown menu set to '.NET Framework Provider for SqlServer (System.Data.SqlClient)'. Below this, there is a 'Connection String' field filled with the text 'Data Source=db;Initial Catalog=Northwind;Integrated Security=True;'. There are also two bullet points providing instructions on how to build the connection string and select database tables and views.

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 15 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 screens.

- 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 10 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.

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".

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

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

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 *Null* as custom style to eliminate the side bar on the page.

Security
Security settings for this page.
User ? to allow anonymous access to the page.

Name *
OrderForm

Index
1005

External Url

Title *
Order Form

Path
Order Form

Description
This is the order management form

Roles

OK Cancel

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.

Roles

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.

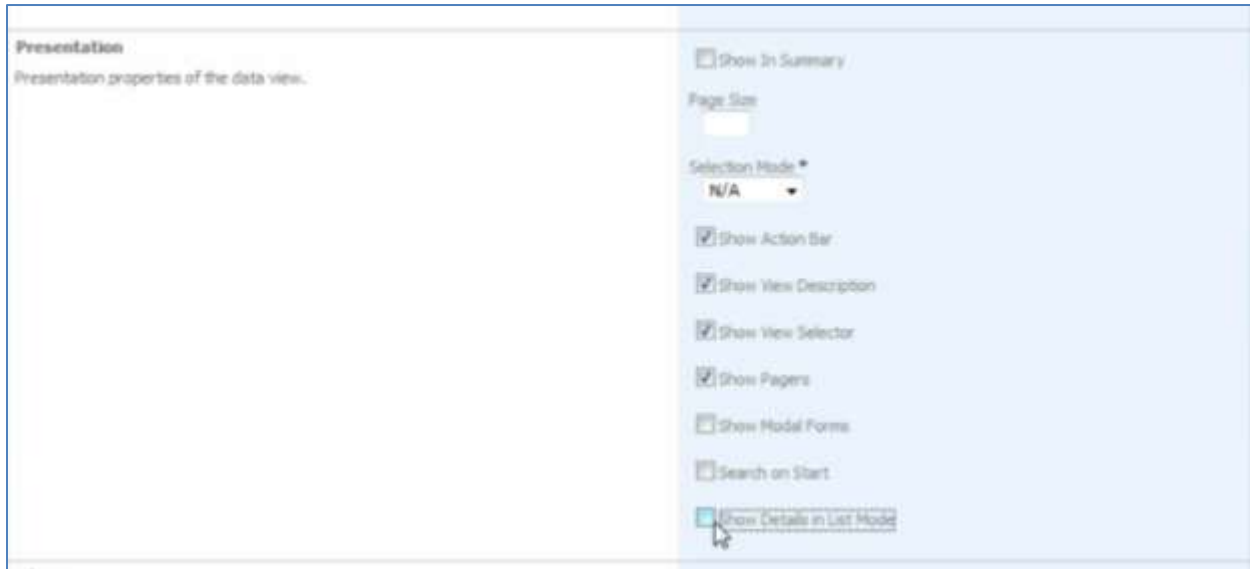
The screenshot shows a dialog box for creating a new container. It has a 'General' tab selected. The left pane contains instructions: 'A container can host multiple data views and user controls. At least one container must be declared for each page.' and 'Specify container flow rule and optional width. 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.' The right pane contains fields: 'Id' (N/A), 'Flow' (New Row), 'Width' (empty), 'CSS Class Name' (empty), and 'CSS Style Properties' (empty). At the bottom, there is a legend '* - indicates a required field' and 'OK' and 'Cancel' buttons.

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".

The screenshot shows a dialog box for creating a new data view. It has tabs for 'Page', 'Containers', 'Data Views', and 'Controls', with 'Data Views' selected. The top bar says 'Please fill this form and click OK button to create a new data view record. Click Cancel to return to the previous screen.' and 'View: New Data View'. Below is a legend '* - indicates a required field' and 'OK' and 'Cancel' buttons. The 'General' tab is active, with instructions: 'Page, container, controller, and controller view of the data view.' and 'Use Tag to enable conditional controller actions with matching whenTag property and to write custom business rules specific to tagged data views.' The right pane contains fields: 'Id' (N/A), 'Container' (c100), 'Controller' (Orders), and 'View' (grid1).

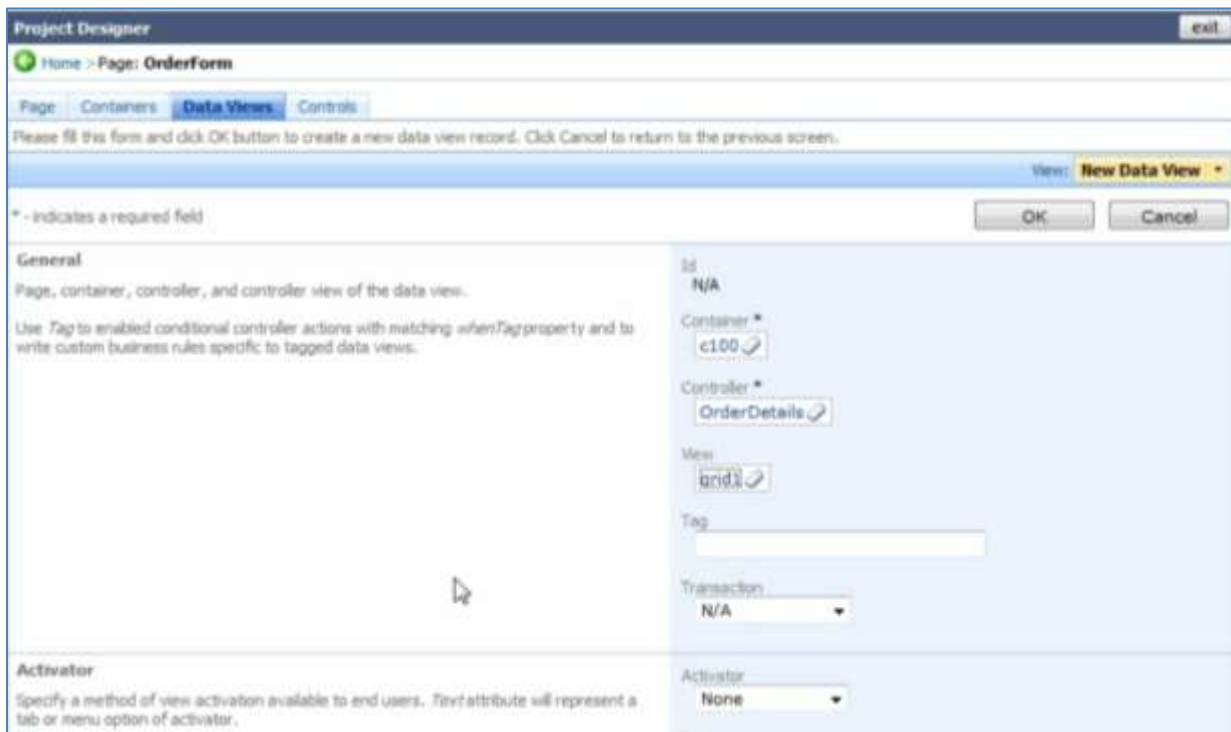
Scroll down to *Presentation* properties, and uncheck “Show Details in List Mode”. This way, no details will be shown next to master records in the list.



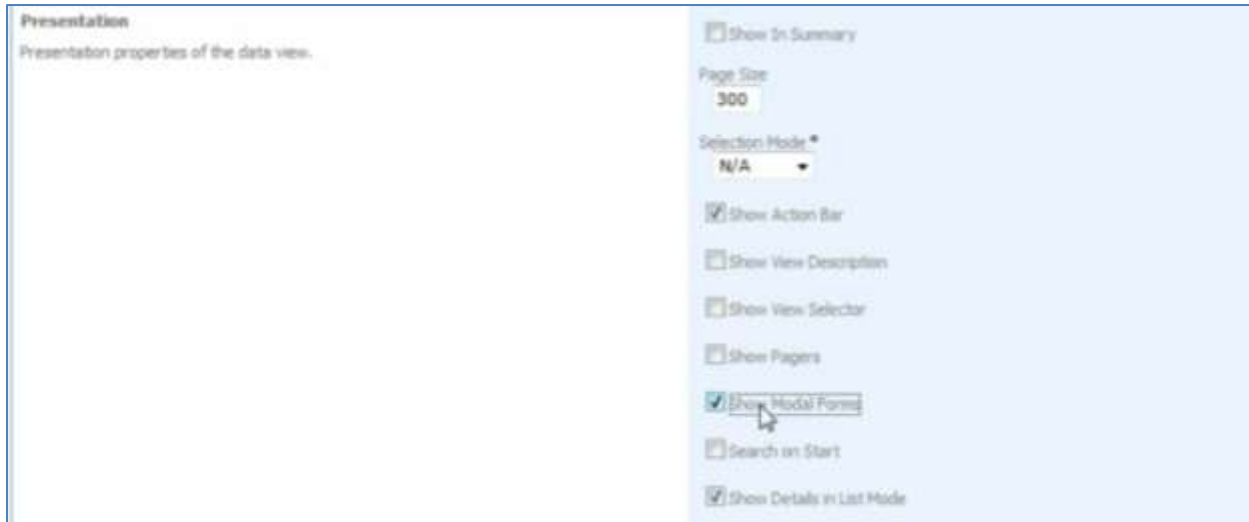
Don't forget to save the record.

Add Data View for “Order Details”

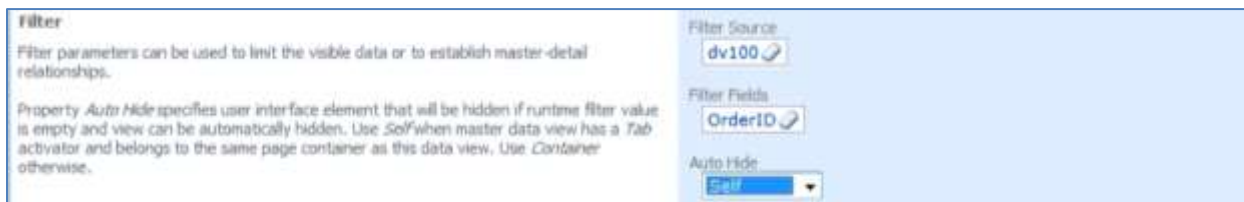
On the action bar, add another data view by pressing *New / New Data View*. *Container* will be “c100”, *Controller* will be “OrderDetails”, and *View* will be “grid1”.



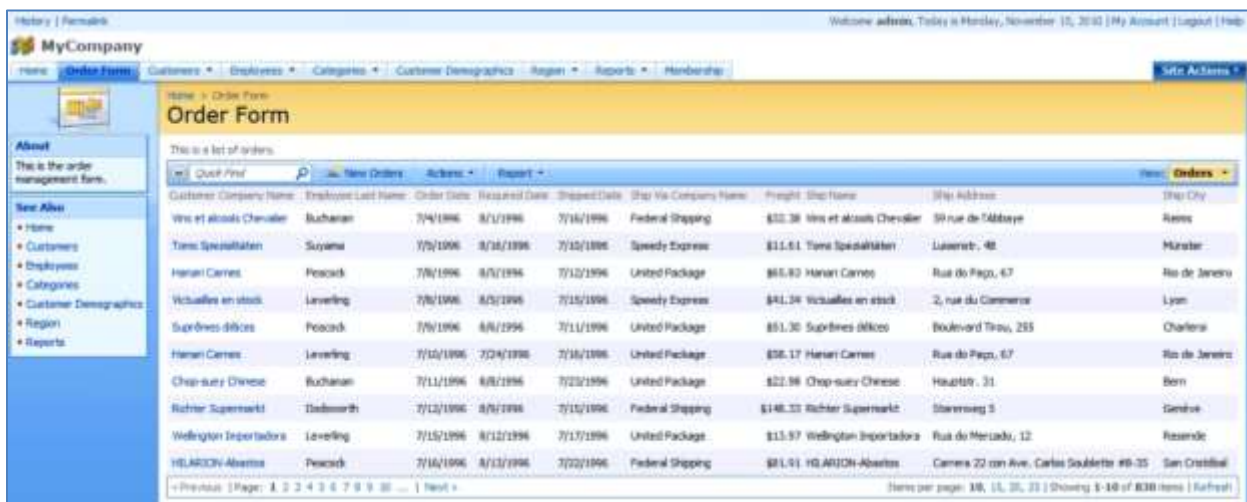
Let's set up a few other properties below. Disable "Show View Description", "Show View Selector", "Show Pagers", set *Page Size* to "300", and enable "Show Modal Forms".



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".



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.

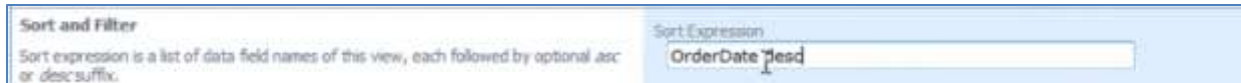


You can browse the list of orders. Select an order, and you will view its details, including order details below.

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*.



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.

The image shows a window titled 'This is a list of customers.' with a search filter section and a table of customer records. The search filters include Customer#, Company Name, Contact Name, Contact Title, and Address, each with a dropdown menu set to 'equals'. The table has columns for Customer#, Company Name, Contact Name, Contact Title, Address, City, Region, Postal Code, Country, and Phone.

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	Avenida 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	Berguvägen 8	Luleå	n/a	S-958 22	Sweden	0921-12 34 65
BLAUS	Blauser See Delikatessen	Hanna Moos	Sales Representative	Foersterstr. 57	Mannheim	n/a	68306	Germany	0621-09460
BONAP	Bonappetit	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	n/a	67000	France	88.60.15.31
BOLID	Bólido Comidas preparadas	Marín Sommer	Owner	C/ Araquil, 67	Madrid	n/a	28023	Spain	(91) 555 22 82
BONAP	Bon app'	Laurence Labihan	Owner	12, rue des Bouchers	Marseille	n/a	13008	France	91.24.45.40
BOTTM	Bottman Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T3P 8M4	Canada	(604) 555-4729

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”.

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.

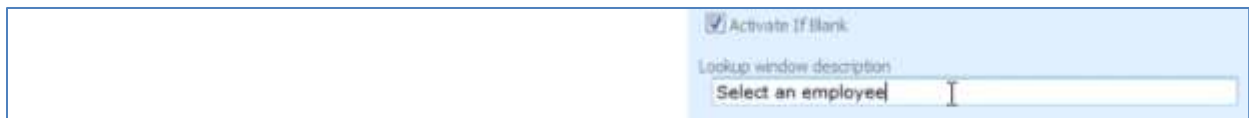
When you select a customer from the lookup, the shipping information will be copied over as well.

Configure "Employee ID" Lookup Field

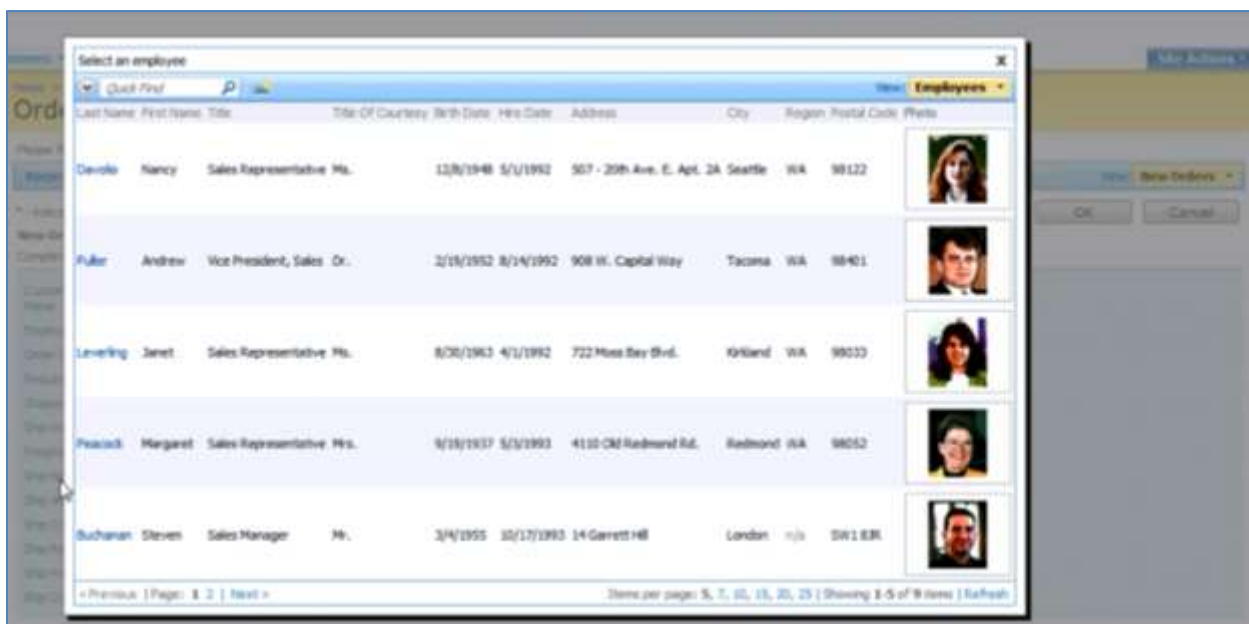
In the *Designer*, go to *All Controllers*. Select the *Employees* data controller. Switch to the *Views* tab. Select "grid1", and switch to *Data Fields* tab. On the action bar, press *New | New Data Field*. Set *Field Name* to *Photo*, and save the field.



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.



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



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.

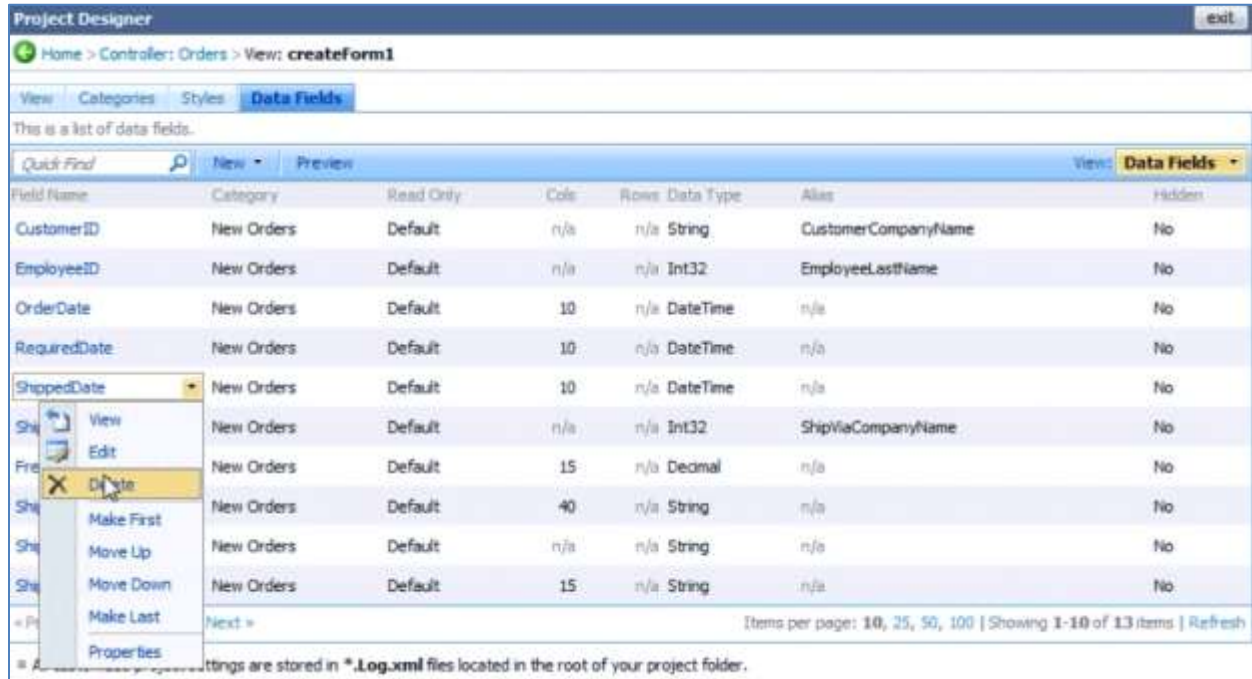
The screenshot shows the 'Project Designer' interface for editing the 'OrderDate' field. The breadcrumb path is 'Home > Controller: Orders > Field: OrderDate'. The 'Field' tab is selected. The 'Record' dropdown is set to 'Field'. The 'General' section contains instructions on field properties. On the right, the 'Name' is 'OrderDate', the 'Controller' is 'Orders', and the 'Type' is 'DateTime'. The 'Allow null values' checkbox is checked. The 'Code Default' field contains the text 'DateTime.Now'.

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

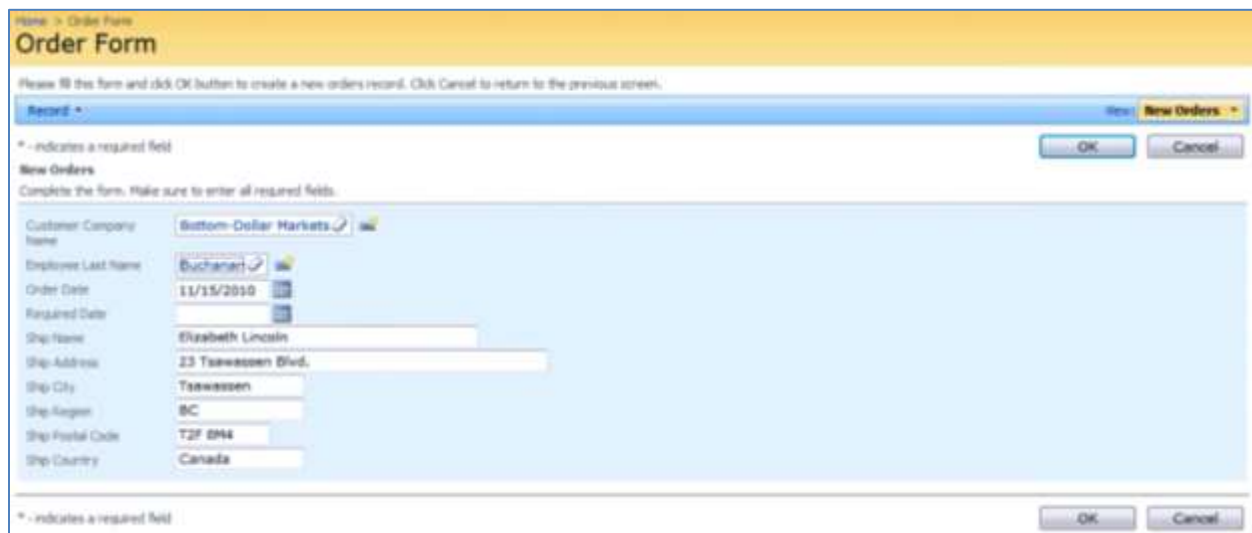
The screenshot shows the 'Order Form' in the application. The breadcrumb path is 'Home > Order Form'. The 'Record' dropdown is set to 'New Orders'. The form contains several fields: 'Customer Company Name' (Berglunds snabbköp), 'Employee Last Name' (Leverind), 'Order Date' (11/15/2010), 'Required Date', 'Shipped Date', 'Ship Via Company Name' (select), 'Freight', 'Ship Name' (Christina Berglund), 'Ship Address' (Berguvsvägen 11), 'Ship City' (Luleå), 'Ship Region', 'Ship Postal Code' (S-958 22), and 'Ship Country' (Sweden). The 'Order Date' field is highlighted, showing the date populated from the application.

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

editForm1 Insert Yes createForm1 Form

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

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)	

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

Header Text	View	Description	New Column	Tab	Floating	Collapsed
New Orders	createForm1	DefaultViewDescription	n/a	n/a	n/a	n/a
Order	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 the 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 Supermarket
 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: Granzacherweg 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.

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

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 *

Unit Price *

Quantity *

Discount *

* - indicates a required field

OK Cancel

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.

Project Designer

Home > Controller: OrderDetails > Field: **Quantity**

Field Items Validators Data Fields Field Outputs

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

Record * View: **Field**

* - indicates a required field

OK Delete Cancel

General

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 *

Controller **OrderDetails**

Type *

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:

Code Default:

Value is retrieved on demand

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

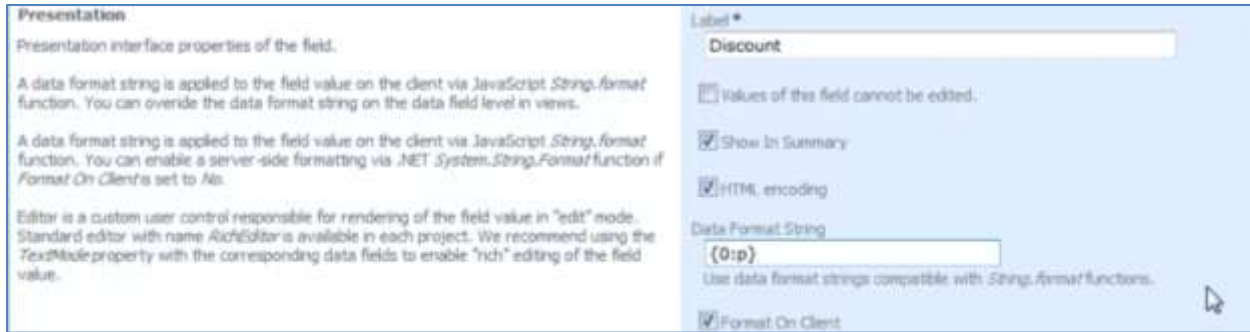
Code Default is an expression written in the programming language of your project. The expression is evaluated in an automatically created business rule to produce a default value for the field before it is presented in the user interface.

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.

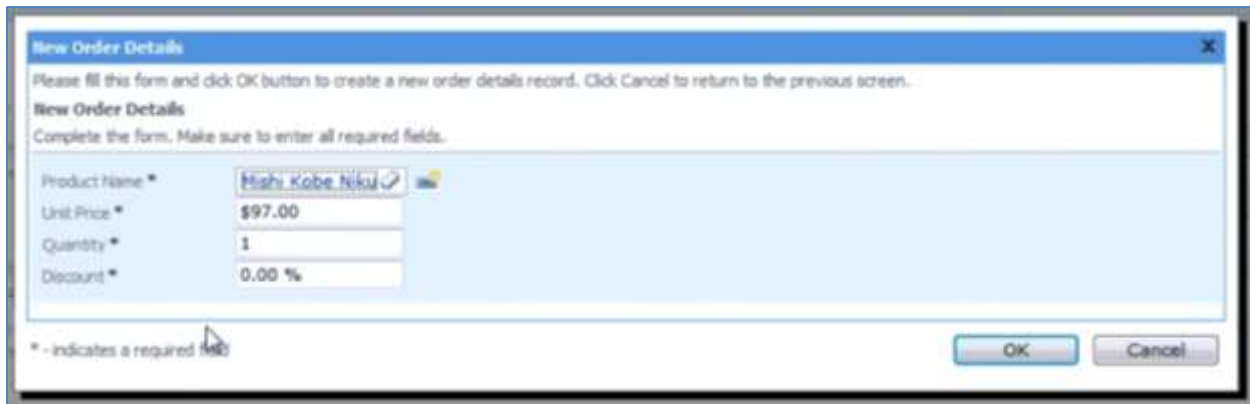
Server Default:

Code Default:

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}”.



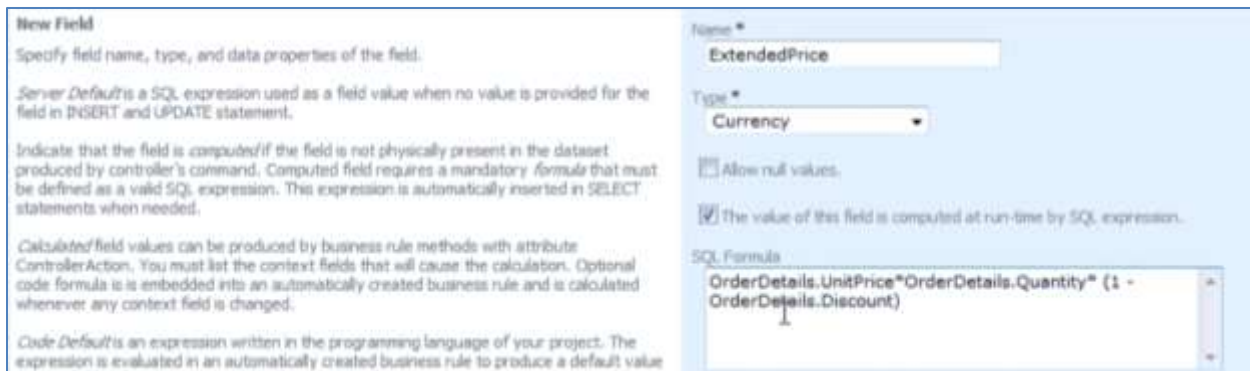
Now, when you create a new *Order Details* and select a product, *Unit Price*, *Quantity*, and *Discount* are automatically prepopulated, and *Discount* is formatted as a percentage.



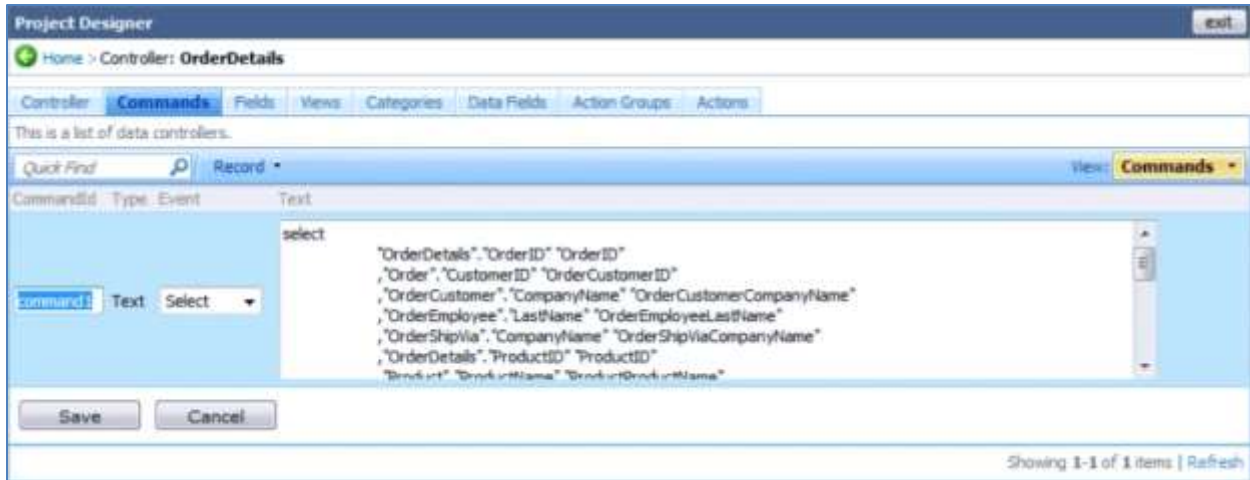
Add “Extended Price” Field

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.

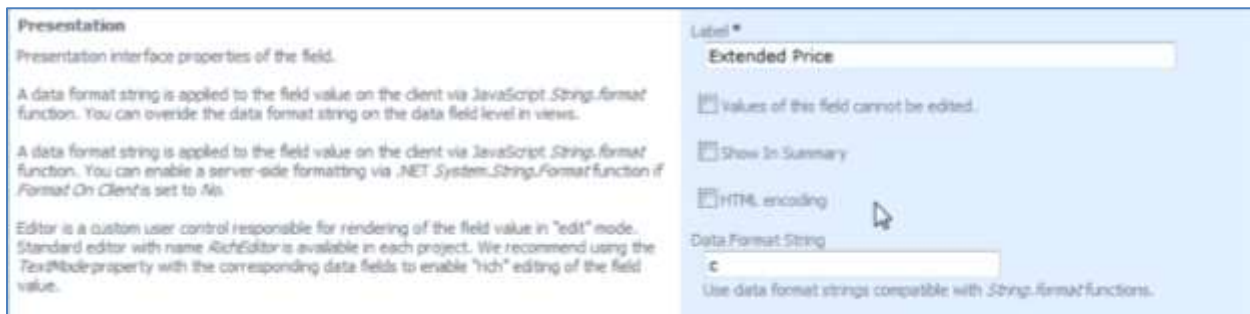
```
OrderDetails.UnitPrice*OrderDetails.Quantity*(1-OrderDetails.Discount)
```



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”.

Project Designer
Home > Controller: OrderDetails > Field: ExtendedPrice

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
Order Details

Alias
(select)

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

Project Designer
Home > Controller: OrderDetails > Field: ExtendedPrice

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 *
grid1

Category
(select)

Alias
(select)

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.

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**

Orders

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

Customer Company Name	Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via Company Name	Freight	Ship Name	Ship Address	Ship City	Ship Region	Ship Postal Code
Sinere betro	King	5/6/1998	6/3/1998	N/A	United Package	\$18.44	Sinere betro	Webbafet 34	Koberham	N/A	1734

Ship Country: Denmark

Order Details

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	3.00 %	Sinere betro	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*.

The screenshot shows a form titled "New Order Details" with the instruction "Complete the form. Make sure to enter all required fields." The fields are: Product Name (Northwoods Cranberry Sauce), Unit Price (\$40.00), Quantity (2), Discount (0.00 %), and 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 screenshot shows a dialog box with a checked option "The value of the field is calculated by a business rule expression." and a text area containing the 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”.

The screenshot shows a dialog box with two sections: "Dynamic Properties" and "Context Fields". The "Context Fields" section contains a text box with the value "ProductID,UnitPrice, Quantity, Discount".

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.

The screenshot shows the "New Order Details" form with the following values: Product Name (Uncle Bob's Organic Dried Pears), Unit Price (\$30.00), Quantity (5), Discount (0.00 %), and 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
```

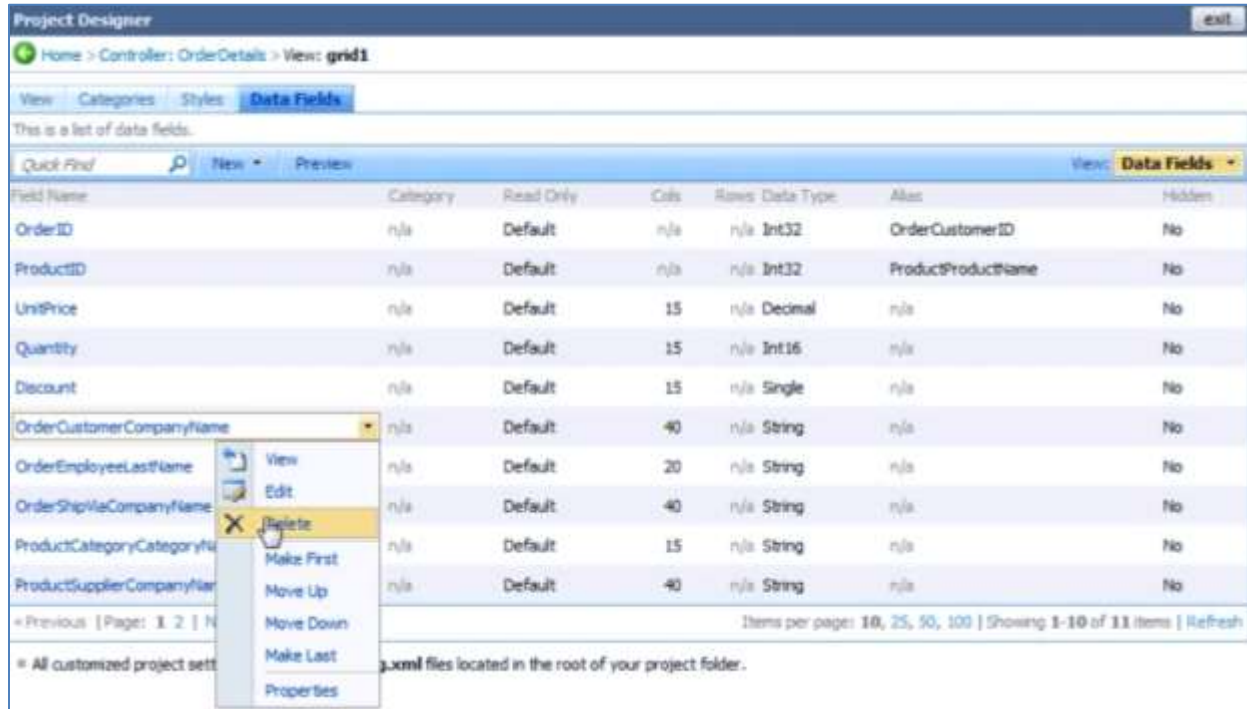
App Code/Rules/OrderDetails.Generated.cs

```
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*.



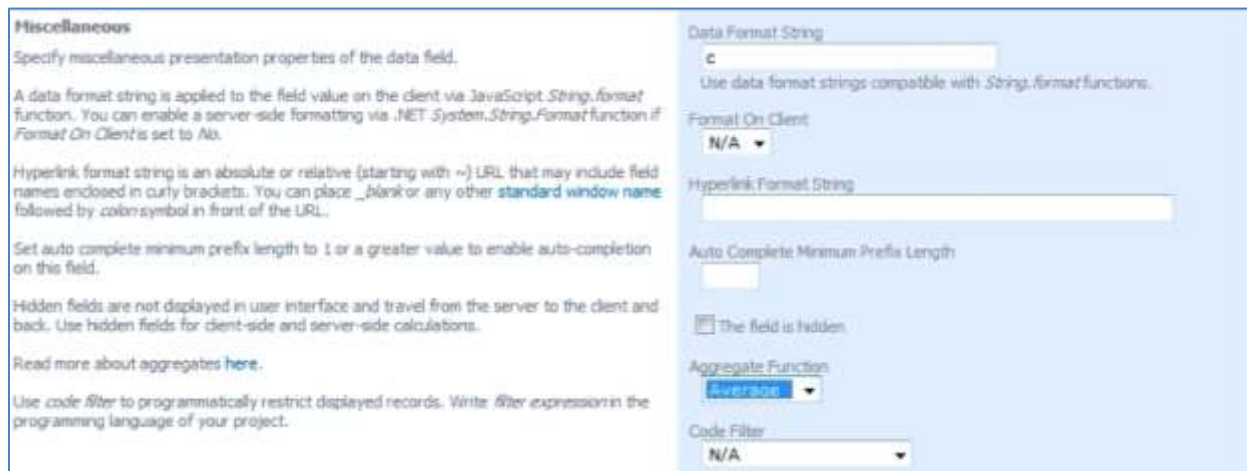
The screenshot shows the Project Designer interface for the *OrderDetails* controller, view *grid1*. The *Data Fields* tab is active, displaying a table of data fields. A context menu is open over the *OrderCustomerCompanyName* field, with the *Delete* option highlighted. The table lists the following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

- Data Format String:** c
- Format On Client:** N/A
- Hyperlink Format String:** (empty)
- Auto Complete Minimum Prefix Length:** (empty)
- The field is hidden:**
- Aggregate Function:** Average
- Code Filter:** N/A

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

<p>Miscellaneous</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 <code>String.Format</code> function. You can enable a server-side formatting via <code>.NET System.String.Format</code> 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 <code>~</code>) URL that may include field names enclosed in curly brackets. You can place <code>_blank</code> or any other standard window name followed by <code>coln</code> 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 here.</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</p> <input type="text"/> <p>Use data format strings compatible with <code>String.Format</code> functions.</p> <p>Format On Client</p> <p>N/A ▼</p> <p>Hyperlink Format String</p> <input type="text"/> <p>Auto Complete Minimum Prefix Length</p> <input type="text"/> <p><input type="checkbox"/> The field is hidden</p> <p>Aggregate Function</p> <p>Sum ▼</p> <p>Code Filter</p> <p>N/A ▼</p>
--	---

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

<p>Miscellaneous</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 <code>String.Format</code> function. You can enable a server-side formatting via <code>.NET System.String.Format</code> 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 <code>~</code>) URL that may include field names enclosed in curly brackets. You can place <code>_blank</code> or any other standard window name followed by <code>coln</code> 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 here.</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</p> <input type="text"/> <p>Use data format strings compatible with <code>String.Format</code> functions.</p> <p>Format On Client</p> <p>N/A ▼</p> <p>Hyperlink Format String</p> <input type="text"/> <p>Auto Complete Minimum Prefix Length</p> <input type="text"/> <p><input type="checkbox"/> The field is hidden</p> <p>Aggregate Function</p> <p>Average ▼</p> <p>Code Filter</p> <p>N/A ▼</p>
--	---

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

<p>Miscellaneous</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 <code>String.Format</code> function. You can enable a server-side formatting via <code>.NET System.String.Format</code> 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 <code>~</code>) URL that may include field names enclosed in curly brackets. You can place <code>_blank</code> or any other standard window name followed by <code>coln</code> 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 here.</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</p> <input type="text"/> <p>Use data format strings compatible with <code>String.Format</code> functions.</p> <p>Format On Client</p> <p>N/A ▼</p> <p>Hyperlink Format String</p> <input type="text"/> <p>Auto Complete Minimum Prefix Length</p> <input type="text"/> <p><input type="checkbox"/> The field is hidden</p> <p>Aggregate Function</p> <p>Sum ▼</p> <p>Code Filter</p> <p>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
Spegeidil	\$12.00	30	15.00 %	Seafood	Lyrngtynid	\$306.00
Lakukokoin	\$18.00	2	15.00 %	Beverages	Karkki Oy	\$30.60
Avg: \$15.33		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 *
Subtotal

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

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 *ViewMode* property with the corresponding data fields to enable "rich" editing of the field value.

Label *
Subtotal

Values of this field cannot be edited.

Show In Summary

HTML encoding

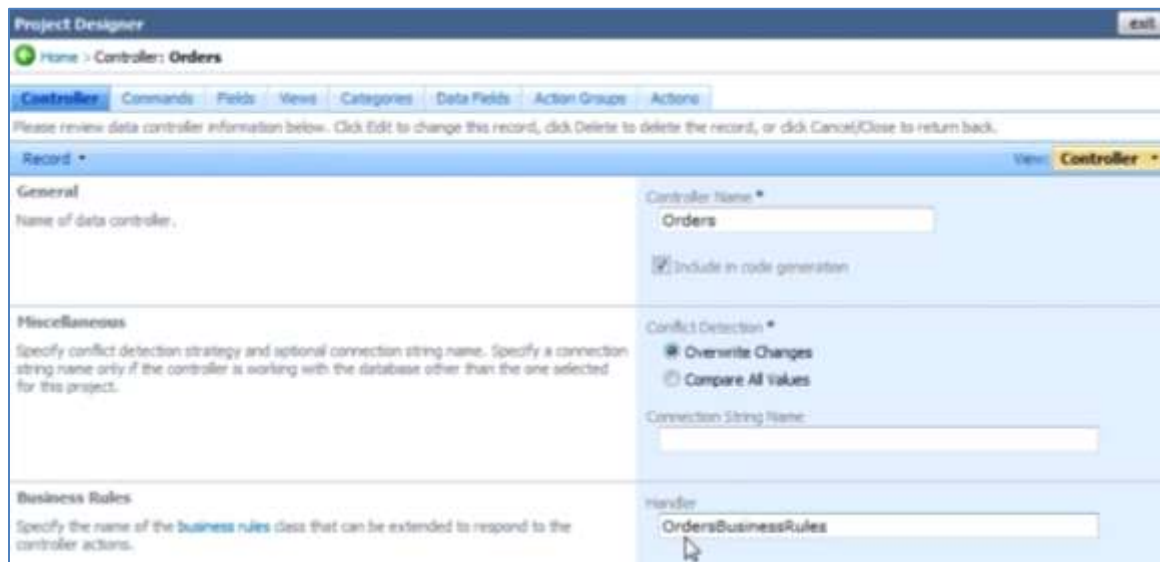
Data Format String
c

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

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”.



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);
            }
        }
    }
}
```

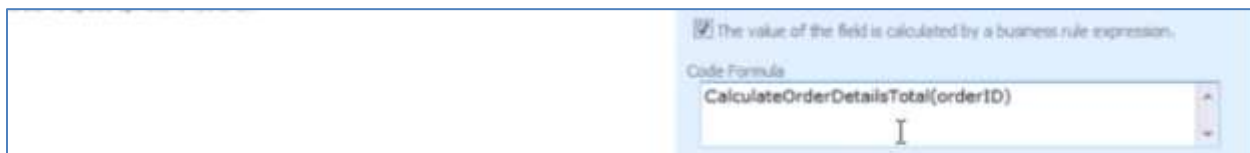
This function uses *SqlText* class to create an instance of a query connected to the project's database. This simple query selects a sum of *UnitPrice* multiplied by *Quantity* multiplied by one minus the *Discount*. Don't forget to save the file.

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
Orders

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.

The screenshot shows an 'Order Form' interface. At the top, it says 'Please review orders information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.' Below this is a 'Record' header with a 'Review Orders' button. A note indicates that a star icon indicates a required field. The main form contains several input fields: Customer Company Name (Richter Supermarkt), Employee Last Name (Fuler), Order Date (11/15/2010), Required Date, Shipped Date, Ship Via Company Name (N/A), Freight, Ship Address (Grenzacherweg 237), Ship City (Geneve), Ship Region, Ship Country (Switzerland), Ship Postal Code (1203), and a summary section with Subtotal (\$285.00) and Total (\$295.00). At the bottom, there is a table with columns: Product Name, Unit Price, Quantity, Discount, Product Category Name, Product Supplier Company Name, and Extended Price. The table contains one row for 'Uncle Bob's Organic Dried Pears' with a unit price of \$30.00, quantity of 10, and a 5.00% discount, resulting in an extended price of \$285.00. Summary statistics at the bottom of the table show an average unit price of \$30.00, a sum of 10, an average discount of 5.00%, and a total sum of \$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".

The screenshot shows a 'Miscellaneous' configuration panel. It contains a text box with the instruction: 'Specify if query-by-example and sorting is enabled in the context menu of the field when presented in grid views.' To the right of this text box are two checked checkboxes: 'Allow Query-by-Example' and 'Allow Sorting'.

Perform the same operation with *Total* field.

The screenshot shows a 'Miscellaneous' configuration panel, identical to the one above. It contains the same instruction text box and two checked checkboxes: 'Allow Query-by-Example' and '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)
```

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.

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

Code Formula
CalculateFreight(orderID, freight)

Server Default
{0}

Code Default

In *Context Fields*, enter “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.

Context Fields
OrderDetails

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: Ruchter Supermarkt
 Employee Last Name: Fuller
 Order Date: 11/15/2010
 Required Date:
 Shipped Date:
 Ship Via Company Name: N/A
 Freight: 11.95

Ship Name: Michael Holz

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

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

* - indicates a required field

OK Delete Cancel

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: Ruchter 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: \$75.95

* - indicates a required field

OK Delete Cancel

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);
        }

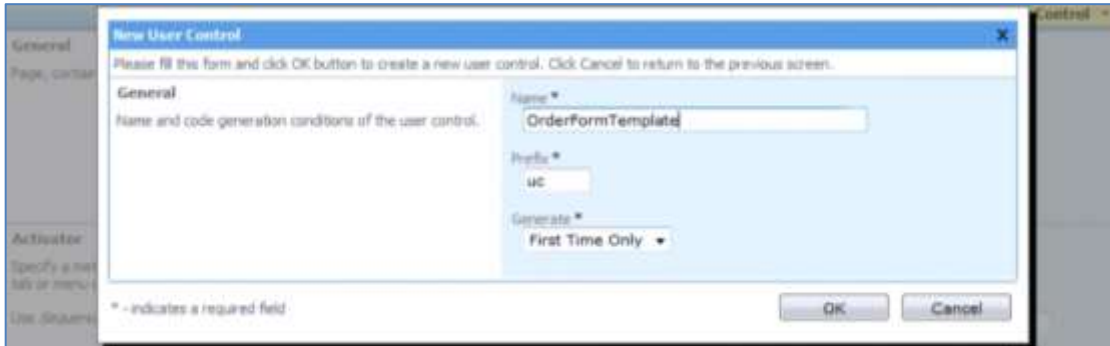
        [RowBuilder("Orders", RowKind.New)]
        public void BuildNewOrders()
        {
            UpdateFieldValue("OrderDate", DateTime.Now);
        }
    }
}
```

Custom Form Template

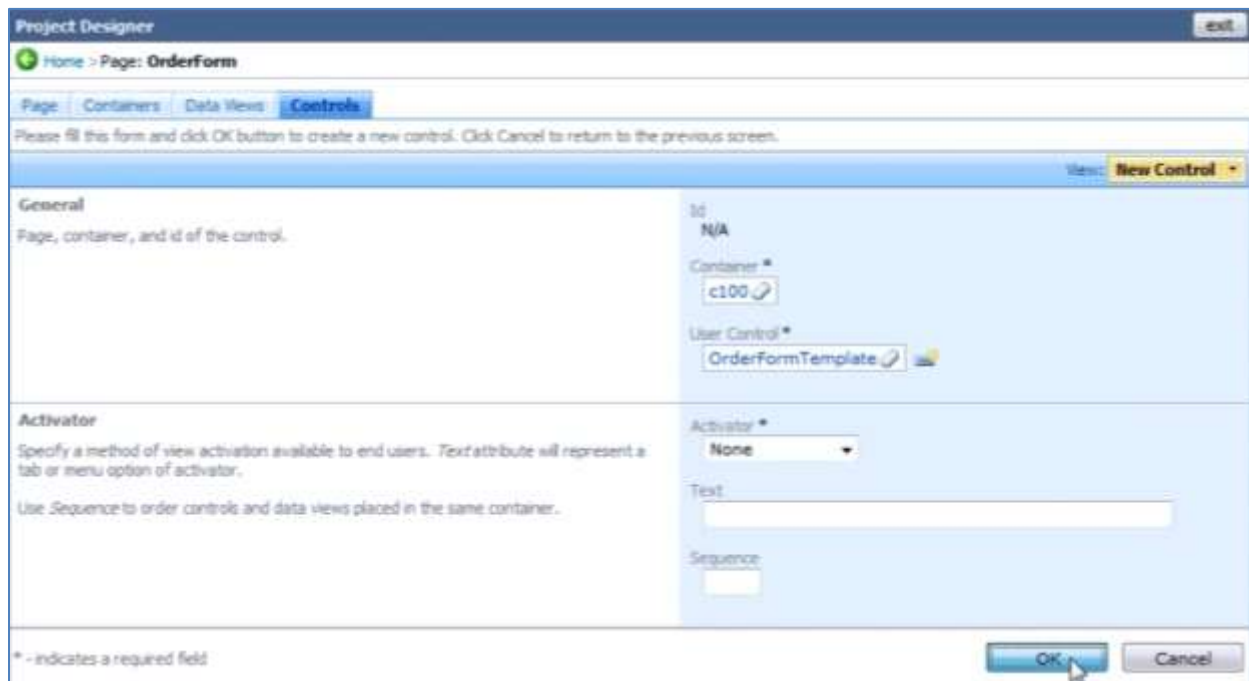
You will need to modify the form template, so that the *Order Form* is easier for the end user to interact with. First, you need to add *Order Form Template* user control to the page.

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

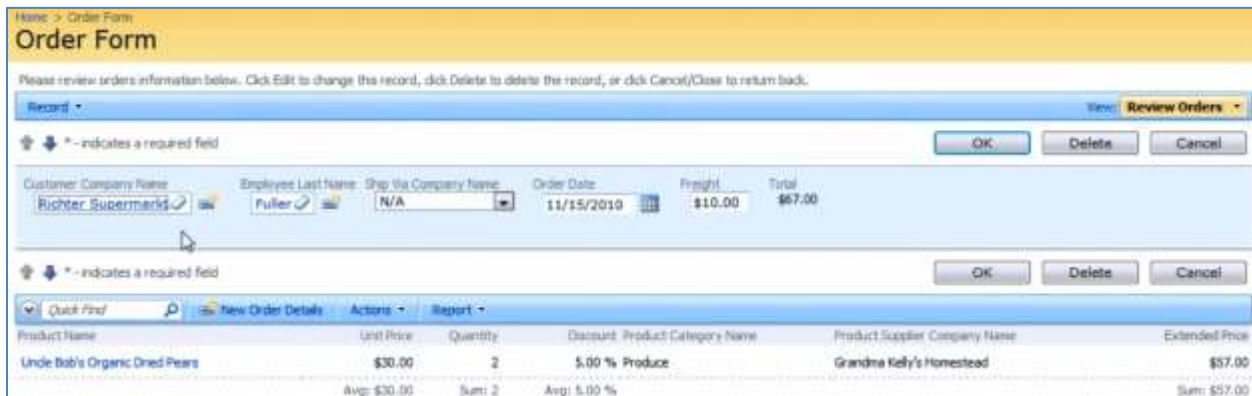
```
Partial Public Class Controls_OrderFormTemplate
    Inherits System.Web.UI.UserControl

    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.

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

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.

Order Form

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

Customer: Richard Supermarket
 Employee: Fuler
 Order Date: 11/15/2010
 Required Date: N/A
 Shipped Date: N/A

Address: Grenscheweg 217
 City: Gerbruc
 Region: N/A
 Postal Code: 1303
 Ship Country: Switzerland

Product Name	Unit Price	Quantity	Discount	Product Category Name	Product Supplier Company Name	Extended Price
Under Ball's Organic Seed Peas	\$28.00	2	5.00 % Product		Gardine Kelly's HomeFood	\$57.00
Avg: \$28.00 Sum: 2 Avg: 5.00 %						Sum: \$57.00

Ship Name: Michael Fuhr
 Ship Via: N/A

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

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.

The screenshot shows a 'Record' window with the following data:

Customer: Richter Supermarket
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*.

The screenshot shows a 'Record' window with the following data:

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

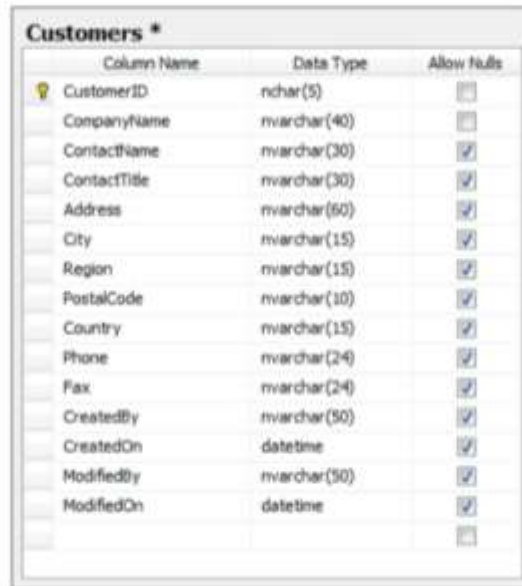
Filter: A filter has been applied. Product Supplier Company Name equals Grandma Kelly's Homestead.

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

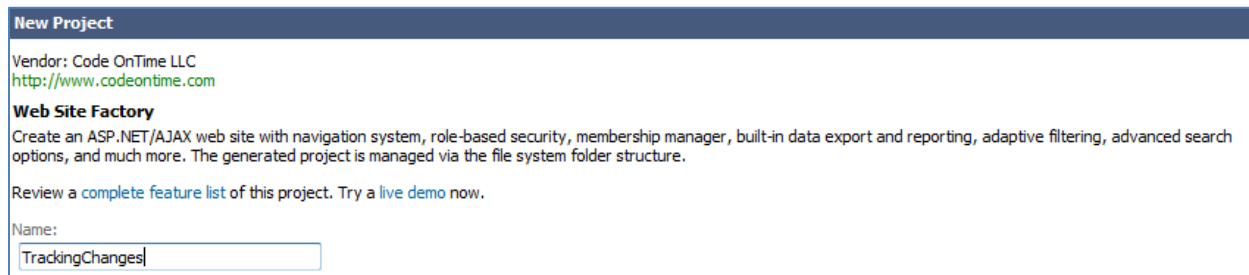
CreatedBy, CreatedOn, ModifiedBy, ModifiedOn

Let's add tracking of user activity in the *Customers* table of *Northwind* database. We'll start by adding four new fields: *CreatedBy*, *CreatedOn*, *ModifiedBy*, and *ModifiedOn*. This can be done in *SQL Server Management Studio*.



Column Name	Data Type	Allow Nulls
CustomerID	nchar(5)	<input type="checkbox"/>
CompanyName	nvarchar(40)	<input type="checkbox"/>
ContactName	nvarchar(30)	<input checked="" type="checkbox"/>
ContactTitle	nvarchar(30)	<input checked="" type="checkbox"/>
Address	nvarchar(60)	<input checked="" type="checkbox"/>
City	nvarchar(15)	<input checked="" type="checkbox"/>
Region	nvarchar(15)	<input checked="" type="checkbox"/>
PostalCode	nvarchar(10)	<input checked="" type="checkbox"/>
Country	nvarchar(15)	<input checked="" type="checkbox"/>
Phone	nvarchar(24)	<input checked="" type="checkbox"/>
Fax	nvarchar(24)	<input checked="" type="checkbox"/>
CreatedBy	nvarchar(50)	<input checked="" type="checkbox"/>
CreatedOn	datetime	<input checked="" type="checkbox"/>
ModifiedBy	nvarchar(50)	<input checked="" type="checkbox"/>
ModifiedOn	datetime	<input checked="" type="checkbox"/>

Now let's create a *Web Site Factory* application that will automatically populate these fields with relevant data. Give it the name of "TrackingChanges".



New Project

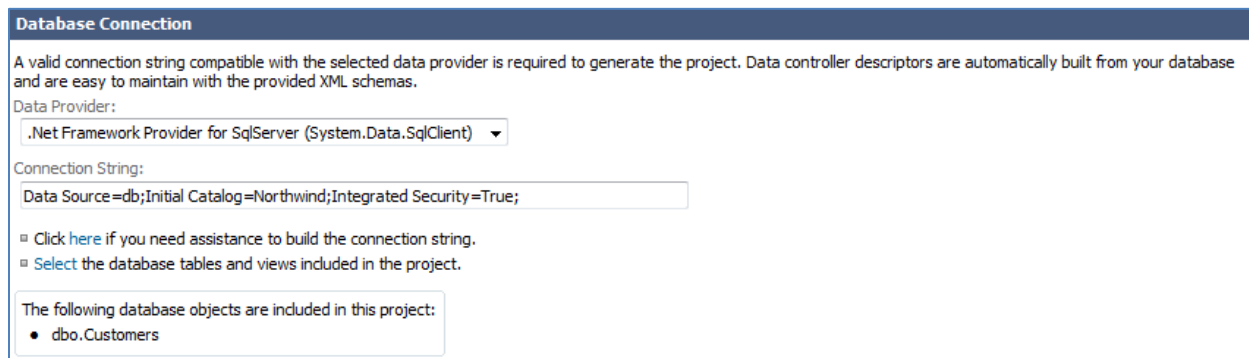
Vendor: Code OnTime LLC
<http://www.codeontime.com>

Web Site Factory
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.

Review a [complete feature list](#) of this project. Try a [live demo](#) now.

Name:

Connect to your server and select the *Northwind* database. Use the *Object Selector* to exclude all other tables from the project except the *Customers* table.



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:
Microsoft .Net Framework Provider for SqlServer (System.Data.SqlClient)

Connection String:

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

The following database objects are included in this project:

- dbo.Customers

Enable reporting for the application.

Reports:

Enable dynamic and static reports in my application.

Enable *ASP.NET Membership*. If necessary, you can connect to a standalone membership database for the application.

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.

Specify a valid connection string compatible with the selected data provider if you want to use a standalone membership database.

Data Provider:

Connection String:

If the connection string is left blank and membership is enabled then ASP.NET will expect that *Microsoft SQL Express* is installed on this computer.

Click [here](#) if you need assistance to build the connection string.

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.

Cancel Back Next

Continue with the project wizard and generate the application. When finished, a web page will appear with the fresh application. There are only three pages available, *Home*, *Customers*, and *Membership*. Navigate to *Customers*. You can see that the new fields are not visible in grid view, but can be viewed and edited from detail view.



Our goal here is to make the fields non-editable in grid and detail view, and not displayed in *New Customers* page. Switch back to the *Generator*, click on the project name, and press the *Design* button. From the list of *All Controllers*, select the *Customers* controller. Switch to *Fields* tab, and select the first field that was made, *CreatedBy*. Edit the field, and change the *Code Default* field to “Context.User.Identity.Name”. This will insert the name of the user creating the record.

Home > Controller: Customers > Field: **CreatedBy**

Field | Items | Validators | Data Fields | Field Outputs

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

Record View: **Field**

* - indicates a required field

General
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 of the new row before it is presented in the user interface.

Code Value is an expression written in the programming language of your project. This expression is evaluated every time a record is saved to the database.

Name *
CreatedBy

Controller
Customers

Type *
String

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

Code Default
Context.User.Identity.Name

Save this field, and click on the next field, *CreatedOn*. In the *Code Default* field, insert “DateTime.Now”. This will insert the date that the field was created on. Scroll down, and in the *Data Format String* field, type in “{0:G}” or “G”. This will show the date and time, up to the seconds. If lowercase “g” is used, seconds will not be displayed.

Code value is an expression written in the programming language of your project. This expression is evaluated every time a record is saved to the database.

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.

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.

Code Default
DateTime.Now

Code Value

Value is retrieved on demand

Label *
Created On

Values of this field cannot be edited.

Show In Summary

HTML encoding

Data Format String
G
Use data format strings compatible with *String.format* functions.

Now, save and move on to the *ModifiedBy* field. As this field needs to be updated every time the record is modified, we will use the *Code Value* field. Type in “Context.User.Identity.Name”, and save.

<p>General</p> <p>Specify field name, type, and data properties of the field.</p> <p><i>Server Default</i> is a SQL expression used as a field value when no value is provided for the field in INSERT and UPDATE statement.</p> <p>Indicate that the field is <i>computed</i> if the field is not physically present in the dataset produced by controller's command. Computed field requires a mandatory <i>formula</i> that must be defined as a valid SQL expression. This expression is automatically inserted in SELECT statements when needed.</p> <p><i>Calculated</i> field values can be produced by business rule methods with attribute <i>ControllerAction</i>. 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.</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 of the new row before it is presented in the user interface.</p> <p><i>Code Value</i> is an expression written in the programming language of your project. This expression is evaluated every time a record is saved to the database.</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>Name *</p> <input type="text" value="ModifiedBy"/> <p>Controller</p> <p>Customers</p> <p>Type *</p> <input type="text" value="String"/> <p><input checked="" type="checkbox"/> Allow null values.</p> <p><input type="checkbox"/> The value of this field is computed at run-time by SQL expression.</p> <p><input type="checkbox"/> The value of the field is calculated by a business rule expression.</p> <p>Server Default</p> <input type="text"/> <p>Code Default</p> <input type="text"/> <p>Code Value</p> <input type="text" value="Context.User.Identity.Name"/>
---	---

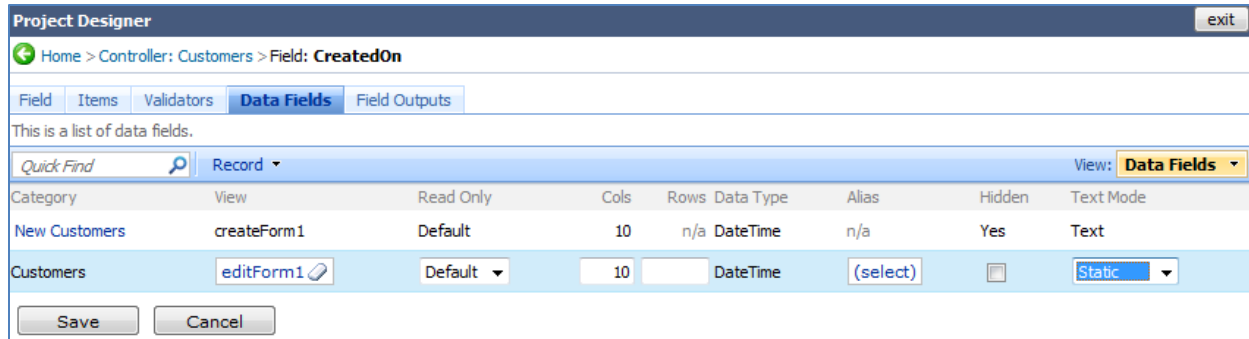
Next, go to the *ModifiedOn* field. In the *Code Value* property, type in “DateTime.Now”, and in the *Data Format String* property, type in “{0:G}” or “G”. These expressions work for both *Visual Basic.NET* and *C#*.

<p>order to speed up record retrieval.</p>	<p>Code Value</p> <input type="text" value="DateTime.Now"/> <p><input type="checkbox"/> Value is retrieved on demand</p>
<p>Presentation</p> <p>Presentation interface properties of the 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 override the data format string on the data field level in views.</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 <i>.NET System.String.Format</i> function if <i>Format On Client</i> is set to <i>No</i>.</p> <p>Editor is a custom user control responsible for rendering of the field value in "edit" mode. Standard editor with name <i>RichEditor</i> is available in each project. We recommend using the <i>TextMode</i> property with the corresponding data fields to enable "rich" editing of the field value.</p>	<p>Label *</p> <input type="text" value="Modified On"/> <p><input type="checkbox"/> Values of this field cannot be edited.</p> <p><input type="checkbox"/> Show In Summary</p> <p><input checked="" type="checkbox"/> HTML encoding</p> <p>Data Format String</p> <input type="text" value="G"/> <p>Use data format strings compatible with <i>String.format</i> functions.</p>

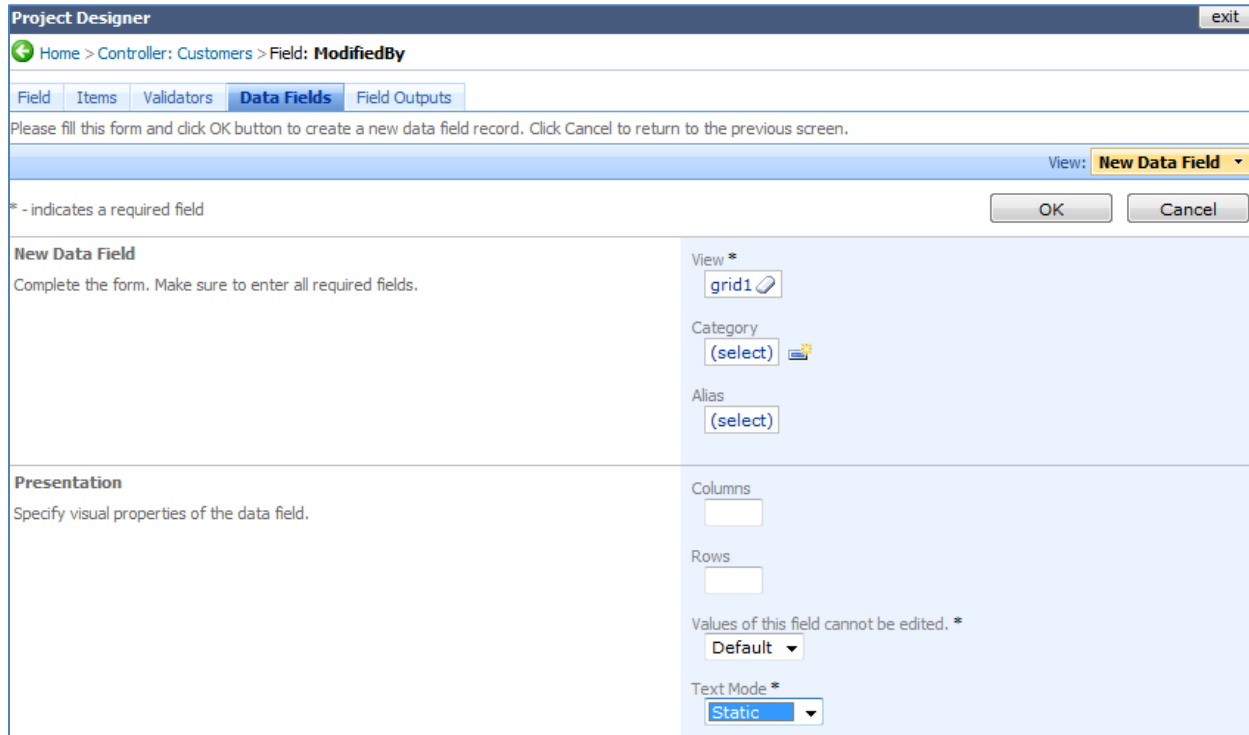
Next, we'll modify the bindings of each field to the respective data views. Select *CreatedBy*, and switch to the *Data Fields* tab. Edit the first binding so that the field is hidden in *createForm1*, and change *Text Mode* to “Static” for *editForm1*.

Category	View	Read Only	Cols	Rows	Data Type	Alias	Hidden	Text Mode
New Customers	createForm1	Default	50	n/a	String	n/a	Yes	Text
Customers	editForm1	Default	50		String	(select)	<input type="checkbox"/>	Static

Save, and go back to the list of fields. Select the *CreatedOn* field, and switch to the *Data Fields* tab. Perform the same changes as with *CreatedOn*.



Go back to the list of fields. The *Modified* fields will require an additional modification. Select the *ModifiedBy* field, and switch to *Data Fields* tab. On the action bar, press *New | New Data Field*. Specify the *View* field as “grid1”, change *Text Mode* to “Static”, and save.



Also, change *createForm1* data field to *Hidden* and change *editForm1* so that *Text Mode* is “Static”.

Category	View	Read Only	Cols	Rows	Data Type	Alias	Hidden	Text Mode
New Customers	createForm1	Default	50	n/a	String	n/a	Yes	Text
Customers	editForm1	Default	50	n/a	String	n/a	No	Static
n/a	grid1	Default	n/a	n/a	String	n/a	No	Static

Go back to the list of fields and select *ModifiedOn*. Switch to *Data Fields* tab, and perform the same operations as with *ModifiedBy*.

Project Designer exit

Home > Controller: Customers > Field: **ModifiedOn**

Field | Items | Validators | **Data Fields** | Field Outputs

This is a list of data fields.

Quick Find New ▾ Preview ▲ Up ▼ Down View: **Data Fields** ▾

Category	View	Read Only	Cols	Rows	Data Type	Alias	Hidden	Text Mode
New Customers	createForm1	Default	10	n/a	DateTime	n/a	Yes	Text
Customers	editForm1	Default	10	n/a	DateTime	n/a	No	Static
n/a	grid1	Default	n/a	n/a	DateTime	n/a	No	Static

Exit the *Designer*, and generate the application. When the web page opens, navigate to the *Customers* page. You can see that *ModifiedBy* and *ModifiedOn* fields are now present in grid view. However, if you edit the row in-line, the fields will be read-only.

MyCompany Welcome admin, Today is Wednesday, December 26, 2012 My Account Logout Help

Home > Customers > Customers

Summary

This is a list of customers.

Quick Find Search ▾ View: **Customers** ▾

Customer #	Company Name	Contact Name	Contact Title	Address	City	Region	Postal Code	Country	Phone	Modified By	Modified On
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	n/a	12209	Germany	030-0074321	n/a	n/a
ANATR	Ana Trujillo Emparedados y Helados	Ana Trujillo	Owner	Avenida de la Constitución 2222	México D.F.	n/a	05021	Mexico	(5) 555-4729	n/a	n/a
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	n/a	05023	Mexico	(5) 555-3932	n/a	n/a
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	n/a	WA1 1DP	UK	(171) 505-7788	n/a	n/a
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvägen 8	Luleå	n/a	S-958 22	Sweden	0921-12 34 65	n/a	n/a
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Foersterstr. 57	Mannheim	n/a	68306	Germany	0621-09400	n/a	n/a
BONAP	Bonappétit pâtisserie	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	n/a	67000	France	88.60.15.31	n/a	n/a
BOLID	Bólido Comidas preparadas	Marín Sommer	Owner	C/ Araquil, 67	Madrid	n/a	28023	Spain	(91) 555 22 82	n/a	n/a

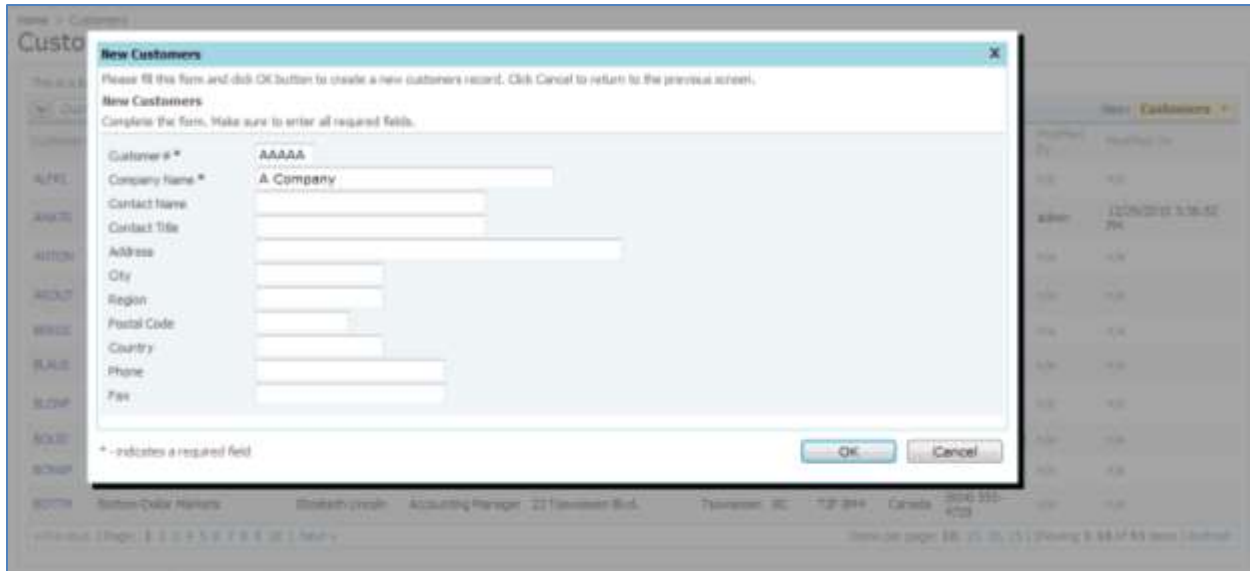
If you modify a record, you can see that the *Modified* fields will be updated as well.

This is a list of customers.

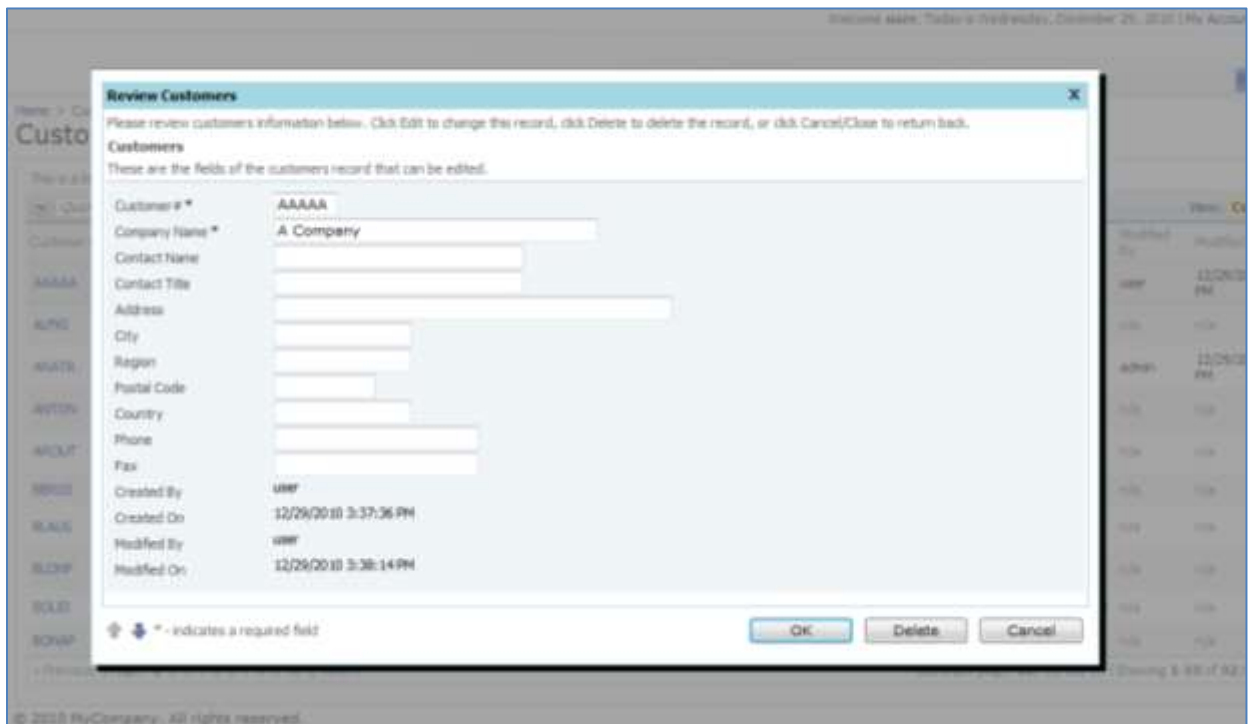
Quick Find New Customers Delete Actions ▾ Report ▾ View: **Customers** ▾

Customer #	Company Name	Contact Name	Contact Title	Address	City	Region	Postal Code	Country	Phone	Modified By	Modified On
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	n/a	12209	Germany	030-0074321	n/a	n/a
ANATR	Ana Trujillo Emparedados y Helados	Ana Trujillo*	Owner	Avenida de la Constitución 2222	México D.F.	n/a	05021	Mexico	(5) 555-4729	admin	12/29/2012 3:36:52 PM
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	n/a	05023	Mexico	(5) 555-3932	n/a	n/a
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	n/a	WA1 1DP	UK	(171) 505-7788	n/a	n/a
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvägen 8	Luleå	n/a	S-958 22	Sweden	0921-12 34 65	n/a	n/a
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Foersterstr. 57	Mannheim	n/a	68306	Germany	0621-09400	n/a	n/a
BONAP	Bonappétit pâtisserie	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	n/a	67000	France	88.60.15.31	n/a	n/a
BOLID	Bólido Comidas preparadas	Marín Sommer	Owner	C/ Araquil, 67	Madrid	n/a	28023	Spain	(91) 555 22 82	n/a	n/a
BONAP	Bon app	Laurence Letihen	Owner	12, rue des Bouchers	Marseille	n/a	13008	France	91.24.45.40	n/a	n/a

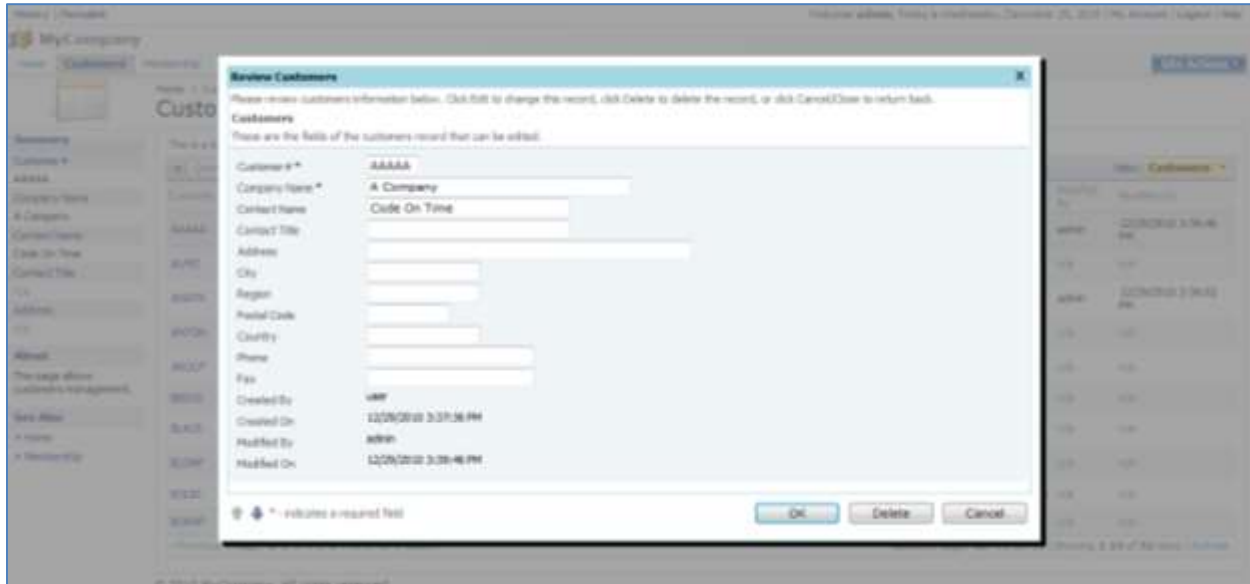
Now, log out of the application and log in as user (*User Name: user, Password: user123%*). Click on *New Customers* button on the action bar. The new fields are not visible in this view.



Fill in the required fields, and save the record. It will appear in the *Customers* list, with correct *ModifiedBy* and *ModifiedOn* information. If you select the record, you can view the *CreatedBy* and *CreatedOn* fields in detail view. When you edit the record, the Created and Modified fields will remain in read-only state.

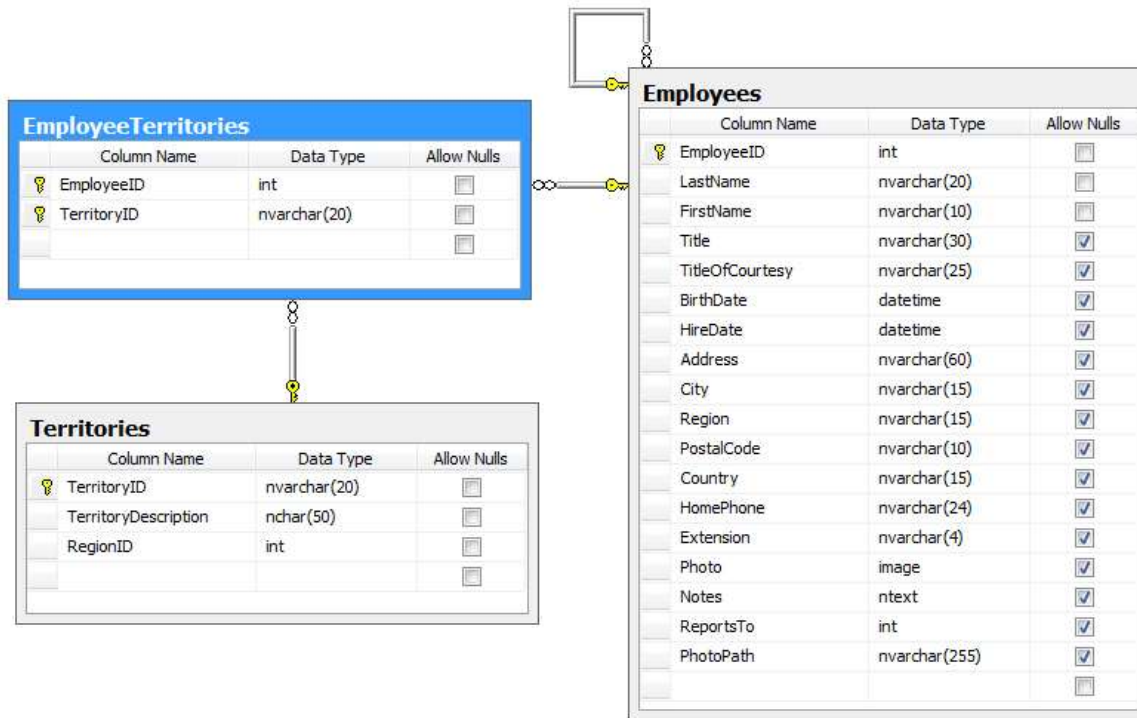


Log out of the application. Log in again as admin. Modify the record you just created, and the *ModifiedBy* and *ModifiedOn* fields will change accordingly. The *CreatedBy* and *CreatedOn* fields will stay the same, however.



Many-To-Many Fields

Let's set up a many-to-many field to the *Employees* page in a *Web Site Factory* application. As you can see below, the *EmployeeTerritories* junction table links together *Employees* and *Territories*.



Below, you can see how the *Employees* page is presented in a non-customized *Web Site Factory* application. There is a long list of fields going down the page, and several tabs below for the child record lists. One of these lists contain all the relevant territories to the selected employee.

The screenshot shows the **Employees** page in a web application. It displays a detailed record for an employee and a list of other employees.

Employee Record (Andrew Fuller):

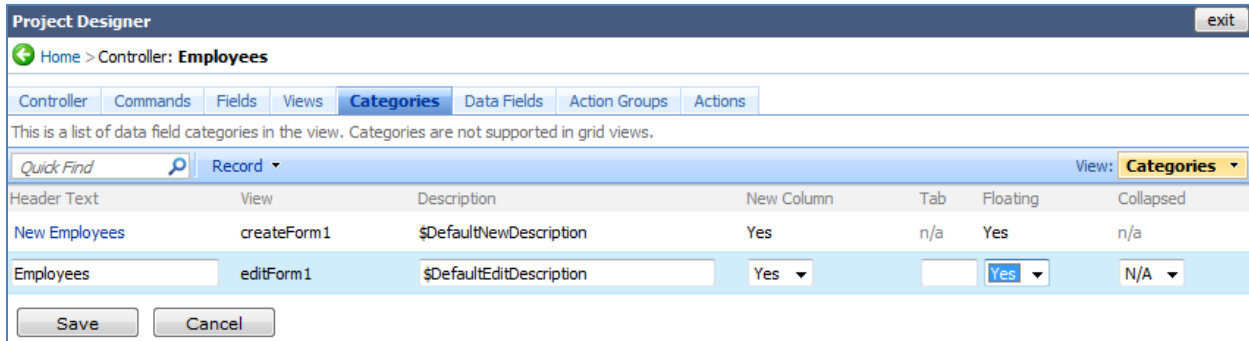
- Last Name: Fuller
- First Name: Andrew
- Title: Vice President, Sales
- Title Of Courtesy: Dr.
- Birth Date: 2/19/1952
- Hire Date: 8/14/1992
- Address: 908 W. Capital Way
- City: Tacoma
- Region: WA
- Postal Code: 98401
- Country: USA
- Home Phone: (206) 555-0482
- Extension: 3457
- Photo:
- Notes: Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.
- Reports To Last Name: N/A
- Photo Path: http://acweb/employees/Fuller.jpg

Employee List:

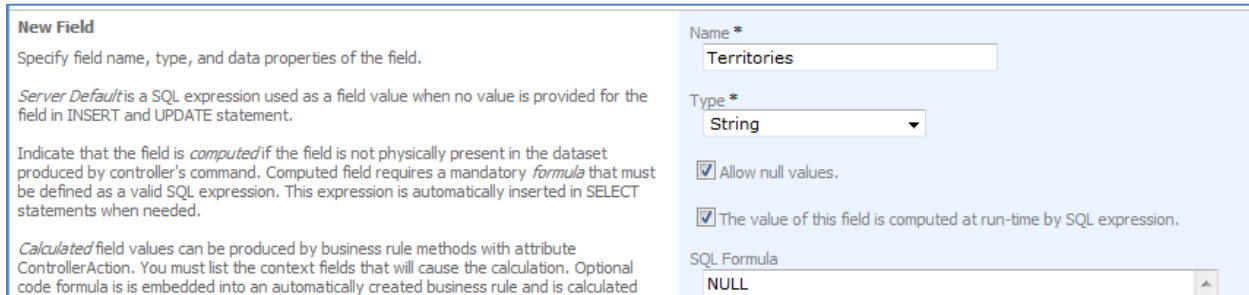
Last Name	First Name	Title	Title Of Courtesy	Birth Date	Hire Date	Address	City	Region	Postal Code
Devolio	Nancy	Sales Representative	Ms.	12/8/1948	8/1/1992	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122
Levelling	Janet	Sales Representative	Ms.	8/30/1963	4/1/1992	722 Moss Bay Blvd.	99164	WA	98033

We would like to make the form more compact, and move the *Territories* directly onto the *Employees* form, as well as make *Territories* more easily editable.

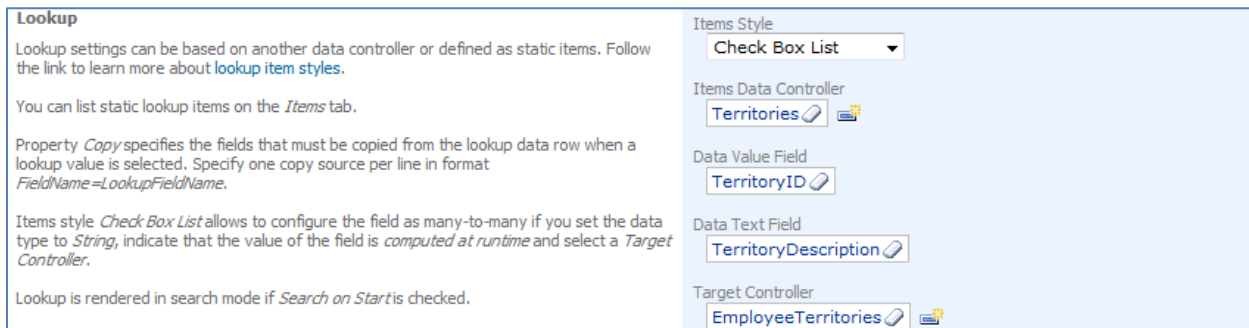
Bring up *Code On Time Generator*, select the project name, and press *Design*. From the list of *All Controllers*, select *Employees*. First, let's make the presentation of the data more compact by switching to *Categories* tab. Change both *New Column* and *Floating* to "Yes" for both categories.



Now, switch to *Fields* tab and create a new *Field*. Give it the *Name* "Territories", *Type* "String", and enable "Allow null values". Enable "The value of this field is computed at run-time", and specify the *SQL Formula* of "NULL". *Label* will be "Territories".



The last step will be to indicate how the items will be displayed. From *Items Style*, choose "Checkbox List". *Items Data Controller* will be "Territories", *Data Value Field* is "TerritoryId", and *Data Text Field* is "Territory Description". *Target Controller* will be "EmployeeTerritories". Save the field, and select it from the list.



Select the field you just created, *Territories*, and then click on *Data Fields* tab. We will need to bind the new field to several views. Create a new data field with *View* of “createForm1”, *Category* of “New Employees”, and set *Columns* to “5”.

The screenshot shows the 'Project Designer' window with the 'Data Fields' tab selected. The breadcrumb path is 'Home > Controller: Employees > Field: Territories'. The 'New Data Field' dialog is open, showing the following configuration:

- View ***: createForm1
- Category**: New Employees
- Alias**: (select)
- Columns**: 5
- Rows**: (empty)
- Values of this field cannot be edited. ***: Default
- Text Mode ***: N/A

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

Save this data field and create a new one. The next data field will be of *View* “editForm1”, *Category* of “Employees”, with *Columns* set to “5”.

The screenshot shows the 'Project Designer' window with the 'Data Fields' tab selected. The breadcrumb path is 'Home > Controller: Employees > Field: Territories'. The 'New Data Field' dialog is open, showing the following configuration:

- View ***: editForm1
- Category**: Employees
- Alias**: (select)
- Columns**: 5
- Rows**: (empty)
- Values of this field cannot be edited. ***: Default
- Text Mode ***: N/A

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

Save, and press the *Preview* button on the action bar.

The new *Employees* page will appear in a browser window. Select an employee and you can see that the detail view is much more compact. There is also a list of territories associated with that employee.

Home > Employees

Employees

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

New Employees | Actions | Report | View: Review Employees

↑ ↓ Edit Delete Close

Employees

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

Last Name	First Name	Title	Title Of Courtesy	Birth Date	Hire Date	Address	City	Region	Postal Code	Country	Home Phone
Fuller	Andrew	Vice President, Sales	Dr.	2/19/1952	8/14/1992	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482

Extension: 3457 | Photo:

Notes: Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.

Reports To Last Name: N/A | Photo Path: http://accweb/employees/fuller.bmp | Territories: Bedford, Boston, Braintree, Cambridge, Georgetown, Louisville, Westboro

↑ ↓ Edit Delete Close

If you press the *Edit* button, then the fields will become editable. You will notice that the new *Territories* field becomes a list of checkboxes. You can mark however many selections, and the application will save your selection when you press *Ok*.

Employees

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

Last Name * Fuller | First Name * Andrew | Title Vice President, Sales | Title Of Courtesy Dr. | Birth Date 2/19/1952 | Hire Date 8/14/1992

Address 908 W. Capital Way | City Tacoma | Region WA | Postal Code 98401 | Country USA

Home Phone (206) 555-9482 | Extension 3457 | Photo

Notes: Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales

Photo Path: http://accweb/employees/fuller.bmp

Territories:

<input type="checkbox"/> Atlanta	<input type="checkbox"/> Cary	<input type="checkbox"/> Hoffman Estates	<input type="checkbox"/> Philadelphia	<input type="checkbox"/> Santa Monica
<input type="checkbox"/> Austin	<input type="checkbox"/> Chicago	<input type="checkbox"/> Hialeah	<input type="checkbox"/> Phoenix	<input type="checkbox"/> Savannah
<input type="checkbox"/> Beachwood	<input type="checkbox"/> Colorado Springs	<input checked="" type="checkbox"/> Louisville	<input type="checkbox"/> Portsmouth	<input type="checkbox"/> Scottsdale
<input checked="" type="checkbox"/> Bedford	<input type="checkbox"/> Columbia	<input type="checkbox"/> Melville	<input type="checkbox"/> Providence	<input type="checkbox"/> Seattle
<input type="checkbox"/> Bellevue	<input type="checkbox"/> Dallas	<input type="checkbox"/> Menlo Park	<input type="checkbox"/> Racine	<input type="checkbox"/> Southfield
<input type="checkbox"/> Bentonville	<input type="checkbox"/> Denver	<input type="checkbox"/> Minneapolis	<input type="checkbox"/> Redmond	<input type="checkbox"/> Tampa
<input type="checkbox"/> Bloomfield Hills	<input type="checkbox"/> Edison	<input type="checkbox"/> Hammett	<input type="checkbox"/> Rockville	<input type="checkbox"/> Troy
<input checked="" type="checkbox"/> Boston	<input type="checkbox"/> Fairport	<input type="checkbox"/> New York	<input type="checkbox"/> Roseville	<input checked="" type="checkbox"/> Westboro
<input checked="" type="checkbox"/> Braintree	<input type="checkbox"/> Findlay	<input type="checkbox"/> New York	<input type="checkbox"/> San Francisco	<input type="checkbox"/> Wilson
<input checked="" type="checkbox"/> Cambridge	<input checked="" type="checkbox"/> Georgetown	<input type="checkbox"/> Newark	<input type="checkbox"/> Santa Clara	
<input type="checkbox"/> Campbell	<input type="checkbox"/> Greensboro	<input type="checkbox"/> Orlando	<input type="checkbox"/> Santa Cruz	

Click [here](#) to upload or clear employees photo file. | Reports To Last Name (select)

* - indicates a required field

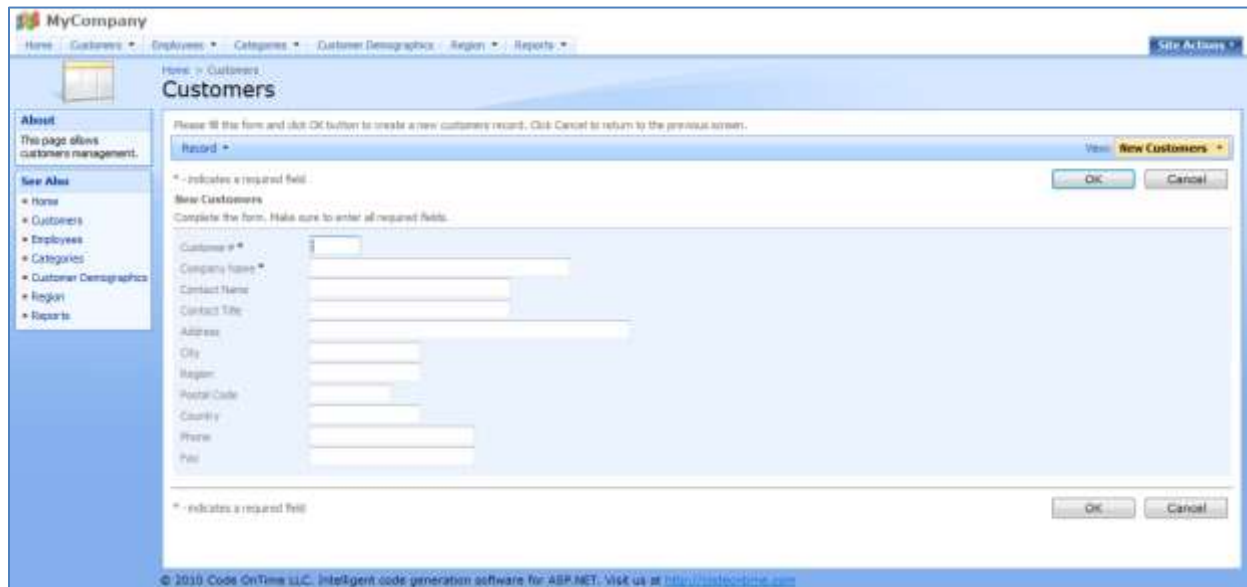
OK Delete Cancel

Data Controller URL Parameters

Code On Time applications recognize several URL parameters that allow constructing simple URL actions to open a multiple-purpose page in "new", "edit", or "view" mode.

The following URL will start a record in "new" mode.

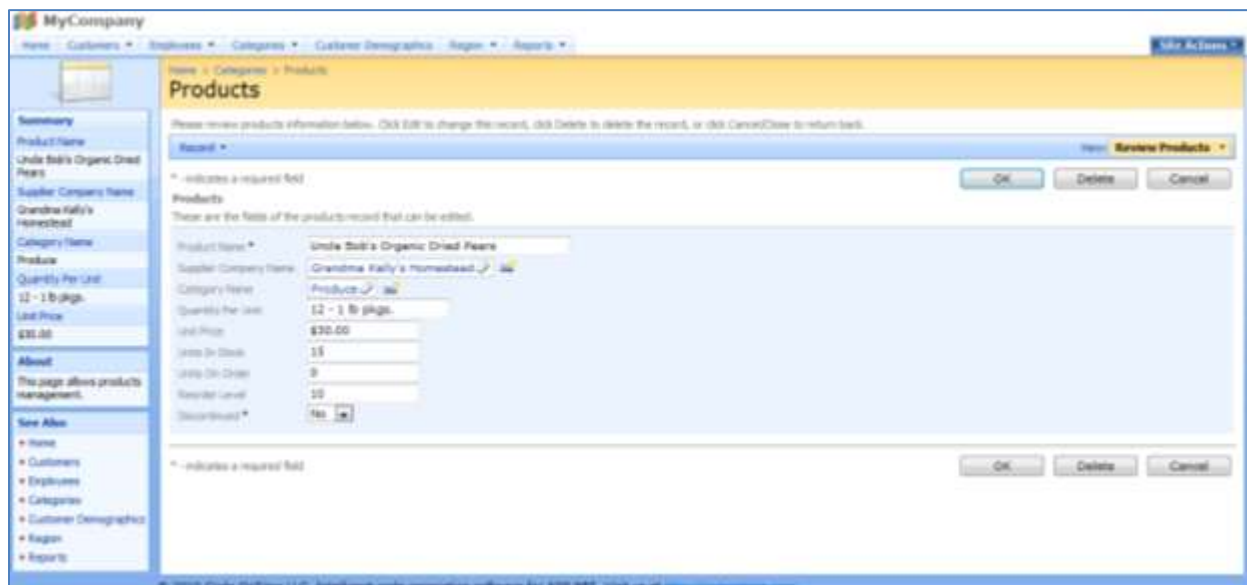
http://dev.codeontime.com/demo/websitefactory1/Pages/Customers.aspx?_controller=Customers&_commandName=New&_commandArgument=createForm1



The screenshot shows the 'MyCompany' web application interface. The main content area is titled 'Customers' and displays a 'New Customers' form. The form includes fields for Customer ID, Company Name, Contact Name, Contact Title, Address, City, Region, Postal Code, County, Phone, and Fax. A 'New Customers' button is visible in the top right corner. The left sidebar contains a navigation menu with options like Home, Customers, Employees, Categories, Customer Demographics, Region, and Reports. The footer indicates the application is powered by Code OnTime LLC.

The following URL will start the *Products* page with product #7 displayed in "edit" mode.

http://dev.codeontime.com/demo/WebSiteFactory1/Pages/Products.aspx?ProductID=7&_controllerName=Products&_commandName=Edit&_commandArgument=editForm1



The screenshot shows the 'MyCompany' web application interface. The main content area is titled 'Products' and displays a 'Review Products' form. The form includes fields for Product Name, Supplier Company Name, Category Name, Quantity Per Unit, Unit Price, Units In Stock, Units On Order, Reorder Level, and Discounted. A 'Review Products' button is visible in the top right corner. The left sidebar contains a navigation menu with options like Home, Customers, Employees, Categories, Customer Demographics, Region, and Reports. The footer indicates the application is powered by Code OnTime LLC.

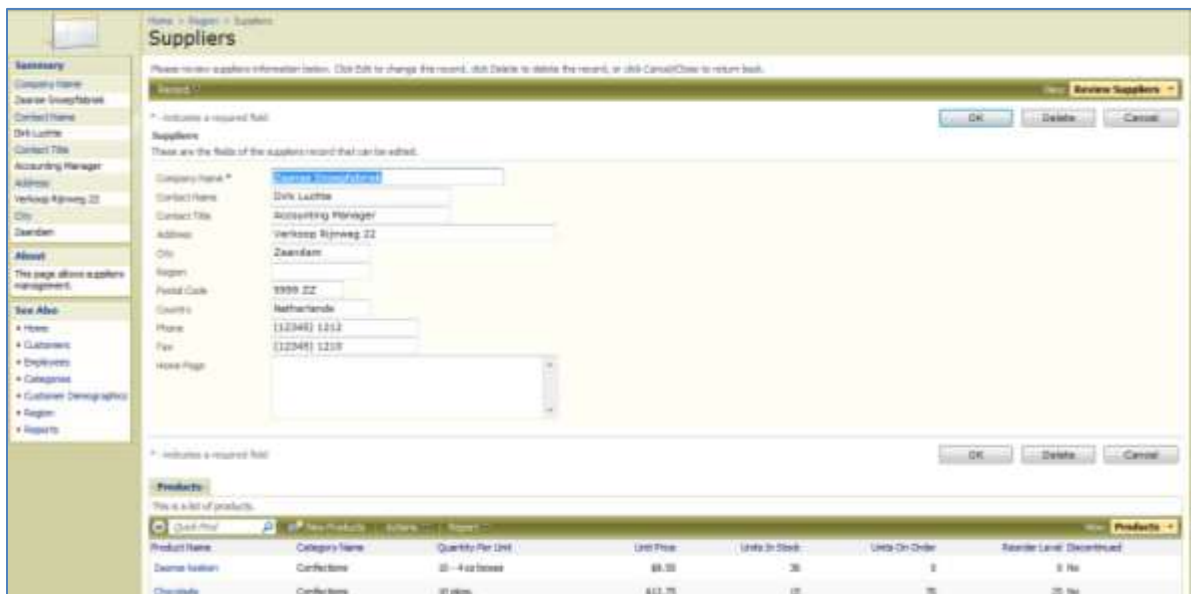
The following URL will select a Product record with ID = 35 in "view" mode:

http://dev.codeontime.com/demo/WebSiteFactory1/Pages/Products.aspx?ProductID=35&_controllerName=Products&_commandName=Select&_commandArgument=editForm1



The “_controller” parameter is optional if your page presents a single data controller. You have to use this parameter to ensure that master-detail pages will respond correctly. The following URL will navigate to the protected *Suppliers* page and will select supplier #22 in “edit” mode. You will have to sign in to access the page.

http://dev.codeontime.com/demo/WebSiteFactory6/Pages/Suppliers.aspx?SupplierID=22&_controllerName=Suppliers&_commandName=Edit&_commandArgument=editForm1

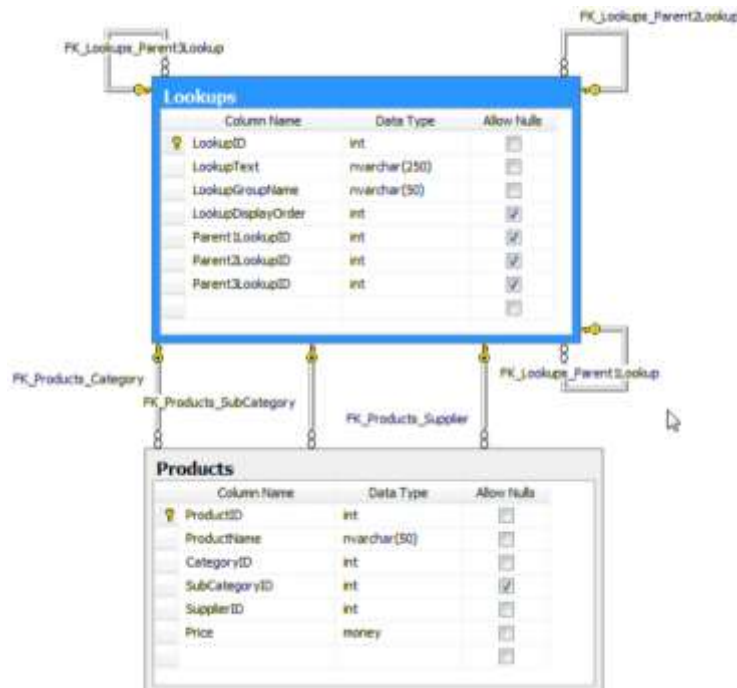


Notice that cancellation or successful Insert, Update, or Delete action will automatically navigate back to the previous page that was loaded in the browser prior to navigation with data controller URL parameters. This convenient behavior will save time that is typically needed to write redirection logic in similar situations.

Universal Lookup

Universal Lookup Database

Universal Lookup functionality allows extension of otherwise static database structures with custom fields, and simplifies maintenance of lookup records in a fixed set of tables. Below is a sample database diagram with a universal lookup. *Lookup* table in the middle has integer *LookupID*, string *LookupText*, and *LookupGroupName* field. Lookup records in the same group constitute a virtual lookup dataset.



Below is a list of records available in this table. There are several records with the group *Categories*, some other ones with the groups *Sub Categories*, *Suppliers*, and *Comm Methods*.

	LookupID	LookupText	LookupGroupN...	LookupDisplay...	Parent1LookupID	Parent2LookupID	Parent3LookupID
▶	16	Category 1	Categories	1	NULL	NULL	NULL
	17	Category 2	Categories	2	NULL	NULL	NULL
	18	Category 3	Categories	3	NULL	NULL	NULL
	19	Sub Category 1	Sub Categories	1	16	NULL	NULL
	20	Sub Category 2	Sub Categories	2	16	NULL	NULL
	21	Sub Category 3	Sub Categories	3	17	NULL	NULL
	22	Sub Category 4	Sub Categories	4	18	NULL	NULL
	23	Sub Category 5	Sub Categories	5	18	NULL	NULL
	24	Sub Category 6	Sub Categories	6	18	NULL	NULL
	25	Sub Category 7	Sub Categories	7	18	NULL	NULL
	26	Sub Category 8	Sub Categories	8	18	NULL	NULL
	27	Supplier 1	Suppliers	NULL	NULL	NULL	NULL
	28	Supplier 2	Suppliers	NULL	NULL	NULL	NULL
	29	Supplier 3	Suppliers	NULL	NULL	NULL	NULL
	30	Comm Method 1	Comm Methods	NULL	NULL	NULL	NULL
	31	Comm Method 2	Comm Methods	NULL	NULL	NULL	NULL
	32	Comm Method 3	Comm Methods	NULL	NULL	NULL	NULL

The lookup table also has three self-referring keys, *Parent1LookupID*, *Parent2LookupID*, and *Parent3LookupID*. This can be used to refer to the category of a subcategory. You can see which subcategories belong to which category in the field list. The references in our example are under *Parent2LookupID* field.

The products table has *CategoryID*, *SubCategoryID*, and *SupplierID*, all of whom are referring to the same lookups table. You will have to set up foreign keys to allow *Code On Time Generator* to detect the relationship that the universal lookup table creates.

This script creates the database tables for *Microsoft SQL Server*.

```
USE [UniversalLookups]
GO
/***** Object: Table [dbo].[Lookups] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Lookups] (
    [LookupID] [int] IDENTITY(1,1) NOT NULL,
    [LookupText] [nvarchar](250) NOT NULL,
    [LookupGroupName] [nvarchar](50) NOT NULL,
    [LookupDisplayOrder] [int] NULL,
    [Parent1LookupID] [int] NULL,
    [Parent2LookupID] [int] NULL,
    [Parent3LookupID] [int] NULL,
    CONSTRAINT [PK_Lookups] PRIMARY KEY CLUSTERED
(
    [LookupID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[Products] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Products] (
    [ProductID] [int] NOT NULL,
    [ProductName] [nvarchar](50) NOT NULL,
    [CategoryID] [int] NOT NULL,
    [SubCategoryID] [int] NULL,
    [SupplierID] [int] NOT NULL,
    [Price] [money] NOT NULL,
    CONSTRAINT [PK_Products] PRIMARY KEY CLUSTERED
(
    [ProductID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: ForeignKey [FK_Lookups_Parent1Lookup] *****/
```



```

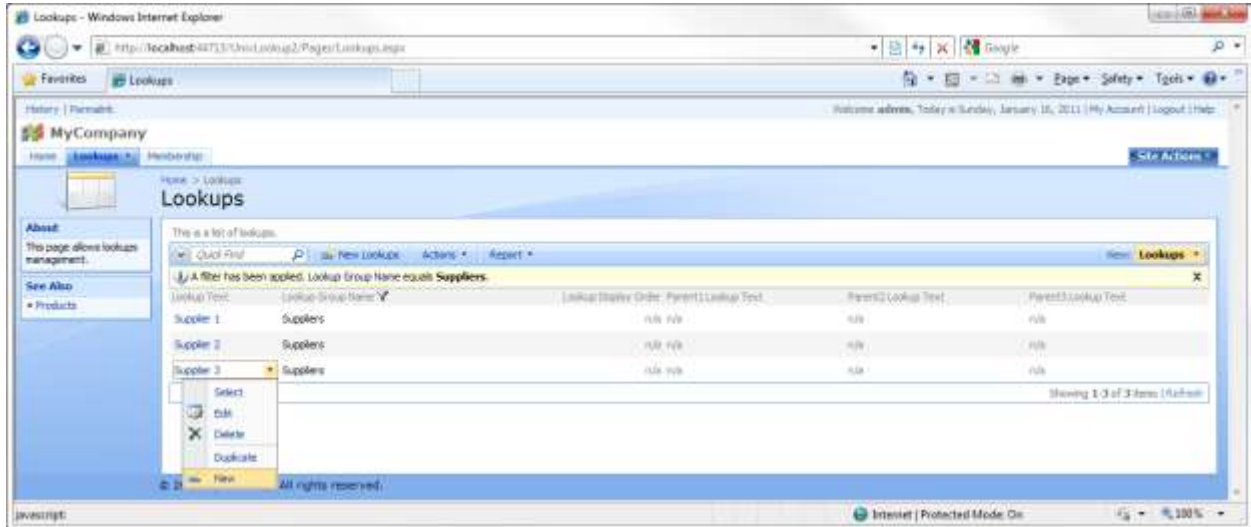
ALTER TABLE [dbo].[Lookups] WITH CHECK ADD CONSTRAINT
[FK_Lookups_Parent1Lookup] FOREIGN KEY([Parent1LookupID])
REFERENCES [dbo].[Lookups] ([LookupID])
GO
ALTER TABLE [dbo].[Lookups] CHECK CONSTRAINT [FK_Lookups_Parent1Lookup]
GO
/***** Object: ForeignKey [FK_Lookups_Parent2Lookup] *****/
ALTER TABLE [dbo].[Lookups] WITH CHECK ADD CONSTRAINT
[FK_Lookups_Parent2Lookup] FOREIGN KEY([Parent2LookupID])
REFERENCES [dbo].[Lookups] ([LookupID])
GO
ALTER TABLE [dbo].[Lookups] CHECK CONSTRAINT [FK_Lookups_Parent2Lookup]
GO
/***** Object: ForeignKey [FK_Lookups_Parent3Lookup] *****/
ALTER TABLE [dbo].[Lookups] WITH CHECK ADD CONSTRAINT
[FK_Lookups_Parent3Lookup] FOREIGN KEY([Parent3LookupID])
REFERENCES [dbo].[Lookups] ([LookupID])
GO
ALTER TABLE [dbo].[Lookups] CHECK CONSTRAINT [FK_Lookups_Parent3Lookup]
GO
/***** Object: ForeignKey [FK_Products_Category] *****/
ALTER TABLE [dbo].[Products] WITH CHECK ADD CONSTRAINT
[FK_Products_Category] FOREIGN KEY([CategoryID])
REFERENCES [dbo].[Lookups] ([LookupID])
GO
ALTER TABLE [dbo].[Products] CHECK CONSTRAINT [FK_Products_Category]
GO
/***** Object: ForeignKey [FK_Products_SubCategory] *****/
ALTER TABLE [dbo].[Products] WITH CHECK ADD CONSTRAINT
[FK_Products_SubCategory] FOREIGN KEY([SubCategoryID])
REFERENCES [dbo].[Lookups] ([LookupID])
GO
ALTER TABLE [dbo].[Products] CHECK CONSTRAINT [FK_Products_SubCategory]
GO
/***** Object: ForeignKey [FK_Products_Supplier] *****/
ALTER TABLE [dbo].[Products] WITH CHECK ADD CONSTRAINT
[FK_Products_Supplier] FOREIGN KEY([SupplierID])
REFERENCES [dbo].[Lookups] ([LookupID])
GO
ALTER TABLE [dbo].[Products] CHECK CONSTRAINT [FK_Products_Supplier]
GO

```

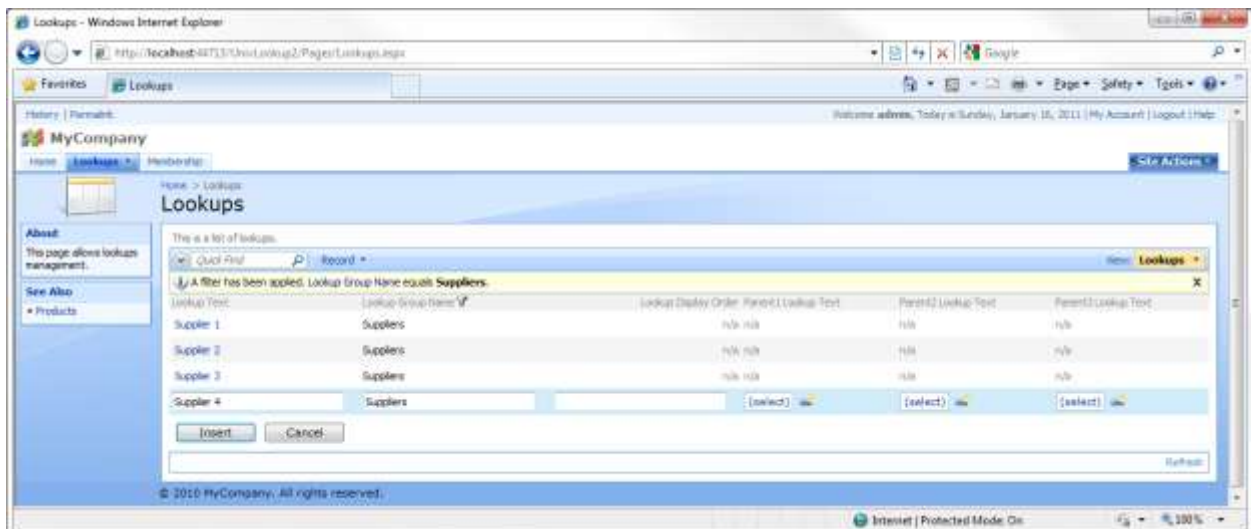
Using the Standard Application

Here is the standard *Web Site Factory* project that is generated straight from the *UniversalLookup* database. Navigate to the *Lookups* page, and you will see a list of records similar to what *SQL Management Studio* provides (with some more advanced functionality, such as adaptive filtering), and you can easily add new records to the table.

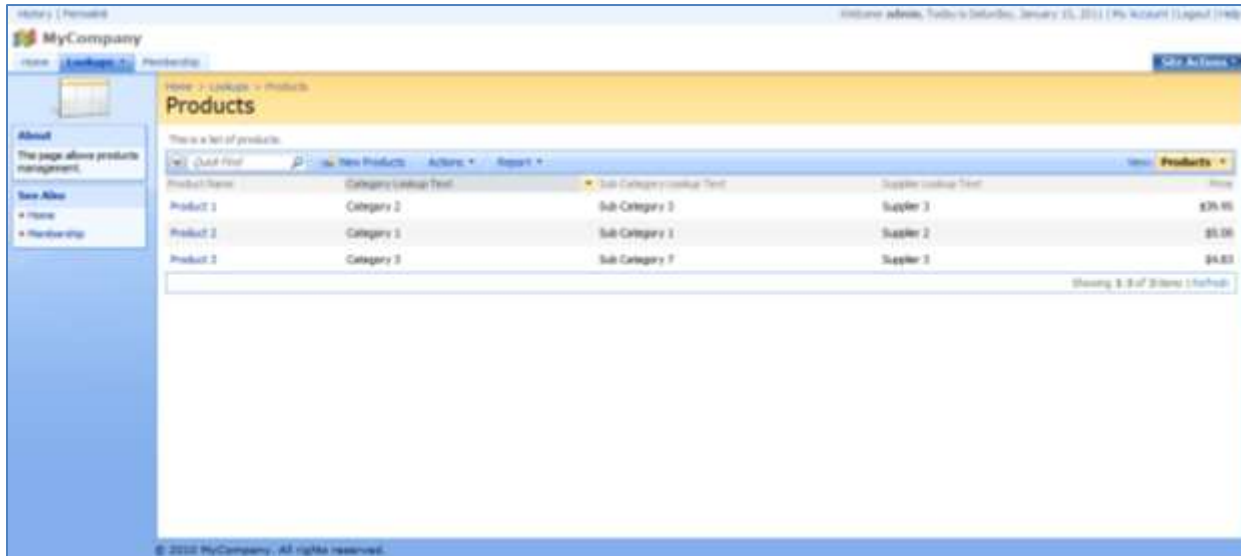
Set *Lookup Group Name* filter to *Suppliers* (using the column header), click on the dropdown menu next to *Supplier 3*, and press *New*.



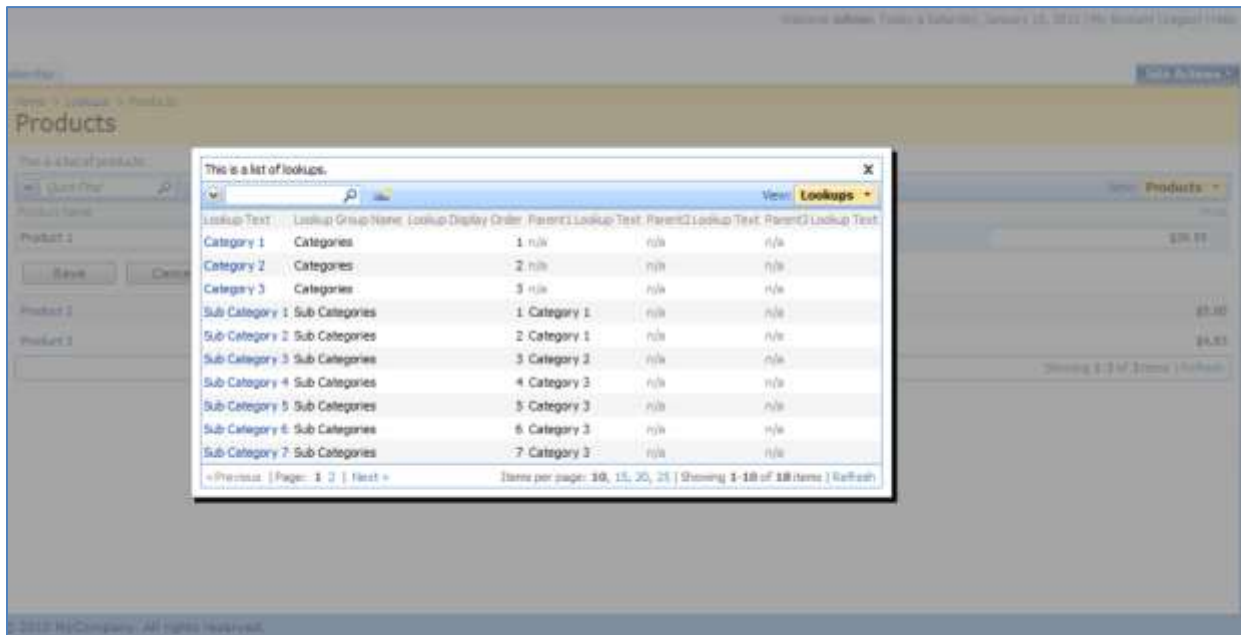
A new inline row will appear. *Lookup Text* field will be called "Supplier 4", with *Lookup Group Name* of "Suppliers". Press *Insert*, and the record will be saved.



Now, switch to the *Products* page. The product records are displayed, but with a few deficiencies. The “Text” suffix is included for three of the fields (*Category Lookup*, *Sub Category Lookup*, *Supplier Lookup*).



When a record is edited, the lookup values for those fields are not limited by *Lookup Group Name*. All values are presented.



Modifying the Standard Application

Bring up *Code On Time Generator*, select the project name, and press *Design*. Select the *Lookups* controller and switch to the *Views* tab. Edit View *grid1*, and change *Sort Expression* to

LookupGroupName, LookupDisplayOrder

Project Designer exit

Home > Controller: Lookups > View: **grid1**

View Categories Styles Data Fields

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

Record ▾ View: **View** ▾

* - indicates a required field

General
Id and type of the view.

Id *
grid1

Controller
Lookups

Type *
 Grid
 Form

Command, Label & Header Text
Specify the command, label and header text for this view.

Command *
command1

Label *
Lookups

Header Text
\$DefaultGridViewDescription

Sort and Filter
Sort expression is a list of data field names of this view, each followed by optional asc or desc suffix.

Sort Expression
LookupGroupName, LookupDisplayOrder

OK Delete Cancel

Now, go back to *All Controllers*, and select *Products* controller. Switch to *Fields* tab. Edit the fields with “Lookup Text” in the *Label*, and remove the relevant text from their respective *Label*.

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

This is a list of fields.

Quick Find 🔍 Record ▾ View: **Fields** ▾

Name	Index	Type	Allow Nulls	Is Primary Key	Read Only	QBE	Sort	LEV	Label
ProductID	1	Int32	No	Yes	No	Yes	Yes	No	Product#
ProductName	2	String	No	No	No	Yes	Yes	No	Product Name
CategoryID	3	Int32	No	No	No	Yes	Yes	No	Category#
CategoryLookupText	4	String	Yes	No	Yes	Yes	Yes	No	Category
SubCategoryID	5	Int32	Yes	No	No	Yes	Yes	No	Sub Category#
SubCategoryLookupText	6	String	Yes	No	Yes	Yes	Yes	No	Sub Category
SupplierID	7	Int32	No	No	No	Yes	Yes	No	Supplier #
SupplierLookupText	8	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Supplier Lookup Text
Price	9	Decimal	No	No	No	Yes	Yes	No	Price

Save Cancel

We will also need to make some modifications to a few fields (*CategoryID*, *SubCategoryID*, and *SupplierID*) to make sure the lookup values are filtered properly. Edit the *CategoryID* field, and scroll down to the *Dynamic Properties* section. Change *Context Fields* value to

LookupGroupName='Categories'

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 <i>Property=FieldName</i> .	Context Fields LookupGroupName='Categories' Dynamic Configuration
---	---

Save the record, and select *SubCategoryID*. Scroll down to the *Dynamic Properties* section, and in *Context Fields*, insert

LookupGroupName='Sub Categories', Parent1LookupID=CategoryID

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 <i>Property=FieldName</i> .	Context Fields LookupGroupName='Sub Categories', Parent1LookupID=CategoryID Dynamic Configuration
---	---

Save the change, and edit *SupplierID*. Scroll down to *Dynamic Properties*, and type in

LookupGroupName='Suppliers'

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 <i>Property=FieldName</i> .	Context Fields LookupGroupName='Suppliers' Dynamic Configuration
---	--

Save the record, close the Designer, and generate the application.

View the Modifications

The columns are now displaying proper headers, with no extraneous “Lookup Text”. The records are now sorted according to *Lookup Group Name* and *Lookup Display Order*.

Lookup Text	Lookup Group Name	Lookup Display Order	ParentID	ParentID
Category 1	Categories	1	nil	nil
Category 2	Categories	2	nil	nil
Category 3	Categories	3	nil	nil
Case Method 1	Case Methods	100	nil	nil
Case Method 2	Case Methods	101	nil	nil
Case Method 3	Case Methods	102	nil	nil
Sub-Category 1	Sub-Categories	1	Category 1	nil
Sub-Category 2	Sub-Categories	2	Category 1	nil
Sub-Category 3	Sub-Categories	1	Category 2	nil
Sub-Category 4	Sub-Categories	2	Category 2	nil

If you navigate to the *Products* page, you can see the extraneous text has been removed here as well.

Product Name	Category	Sub-Category	Supplier	Price
Product 1	Category 1	Sub-Category 1	Supplier 1	\$20.00
Product 2	Category 1	Sub-Category 1	Supplier 2	\$5.00
Product 3	Category 2	Sub-Category 7	Supplier 3	\$4.00

If you edit a record, and select a *Category* with the lookup, you will only see *Categories*. If you open the *Sub Category* lookup, only sub categories relevant to the selected *Category* will be shown. *Supplier* lookup will only show suppliers.

Lookup Text	Lookup Display Order	ParentID	ParentID
Sub-Category 4	4	nil	nil
Sub-Category 5	5	nil	nil
Sub-Category 6	6	nil	nil
Sub-Category 7	7	nil	nil
Sub-Category 8	8	nil	nil

Further Modification

Nevertheless, lookups for those fields are not really that convenient for the end user. Let's modify the presentation to display a dropdown list for *Category*, radio button list for *Sub Category*, and a list box for *Supplier*.

Open the *Designer*, select the *Products* controller, switch to *Fields* tab, and edit *CategoryID*. Scroll down to *Lookup* section, and select "Drop Down List" for *Items Style*. *Data Value Field* is "LookupID" and *Data Text Field* is "LookupText".

The screenshot shows the 'Lookup' configuration panel for the *CategoryID* field. On the left, there is a 'Lookup' section with explanatory text. On the right, the configuration options are: 'Items Style' set to 'Auto Complete', 'Items Data Controller' set to 'Lookups', 'Data Value Field' set to 'LookupID', and 'Data Text Field' set to 'LookupText'. There is also a 'Copy' field at the bottom.

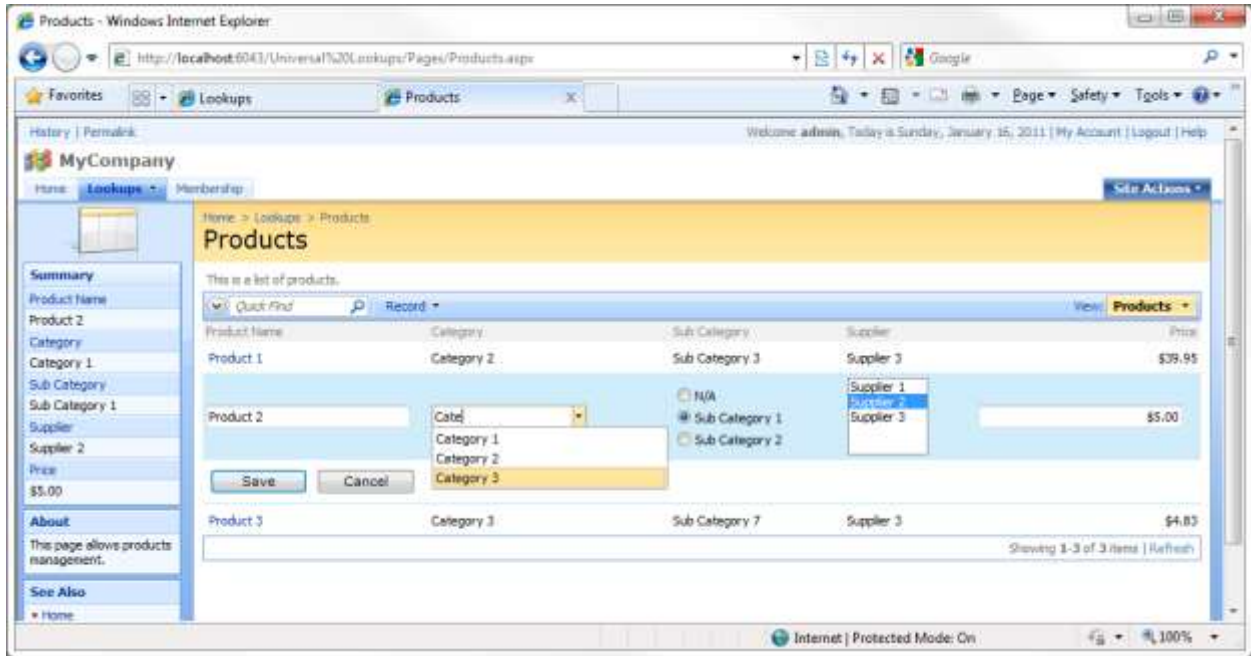
Save, and edit *SubCategoryID* field. Change *Items Style* to "Radio Button List".

The screenshot shows the 'Lookup' configuration panel for the *SubCategoryID* field. The 'Items Style' is now set to 'Radio Button List'. The 'Data Value Field' and 'Data Text Field' are both set to '(select)'. The 'Copy' field is empty.

Save, and edit *SupplierID* field. Change *Items Style* to "List Box".

The screenshot shows the 'Lookup' configuration panel for the *SupplierID* field. The 'Items Style' is now set to 'List Box'. The 'Data Value Field' and 'Data Text Field' are both set to '(select)'. The 'Copy' field is empty.

Save the field, close the *Designer*, and generate the application. When the web page opens, navigate to *Products* page. If you edit a record, you can see the new item lookup styles in action.

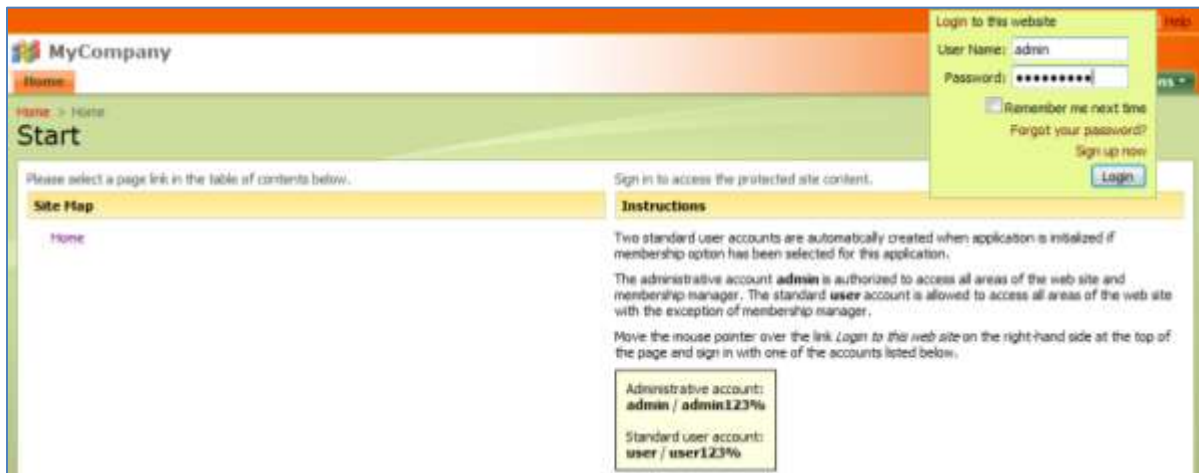


Role-based Security

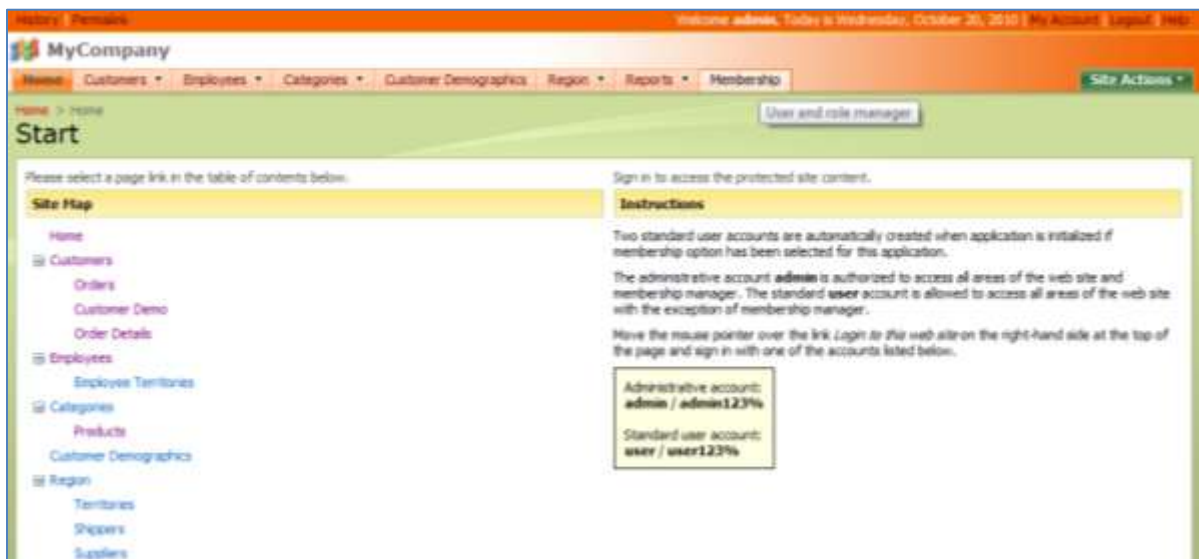
Web Site Factory applications rely on *Microsoft ASP.NET* for security infrastructure. In *ASP.NET* applications, each user is assigned one or several roles. The roles that a user has determine what a user is able to do. For example, only users with the *Administrator* role can access the *Membership Manager*. You can secure your pages, fields, and the various actions through roles.

Pages

The current page being displayed does not have a user logged in. The only page available is *Home*, both on the *Site Map* and on the menu bar. The membership bar allows us to sign in. By default, there are two accounts available for use, administrator and standard user. Let's log in as administrator.

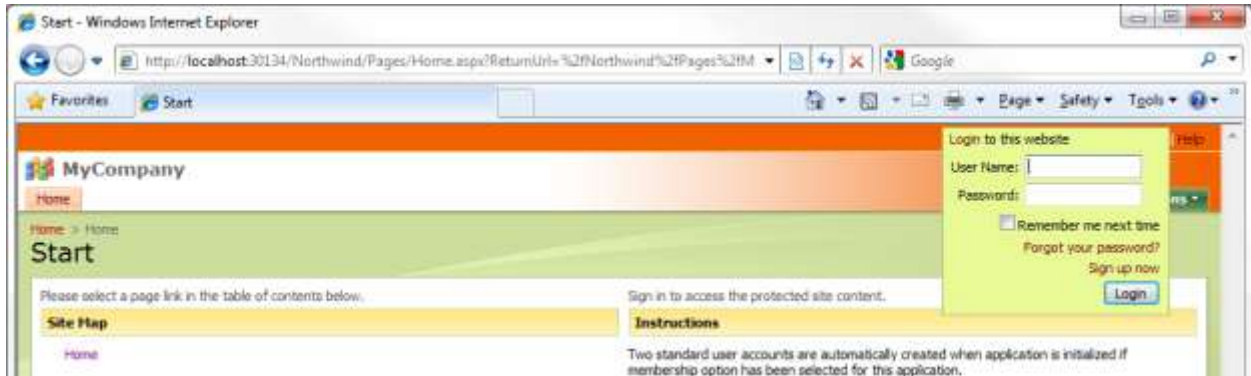


You can see that after logging in, the sitemap and menu bar has been expanded. More options are available, including the *Membership* page.

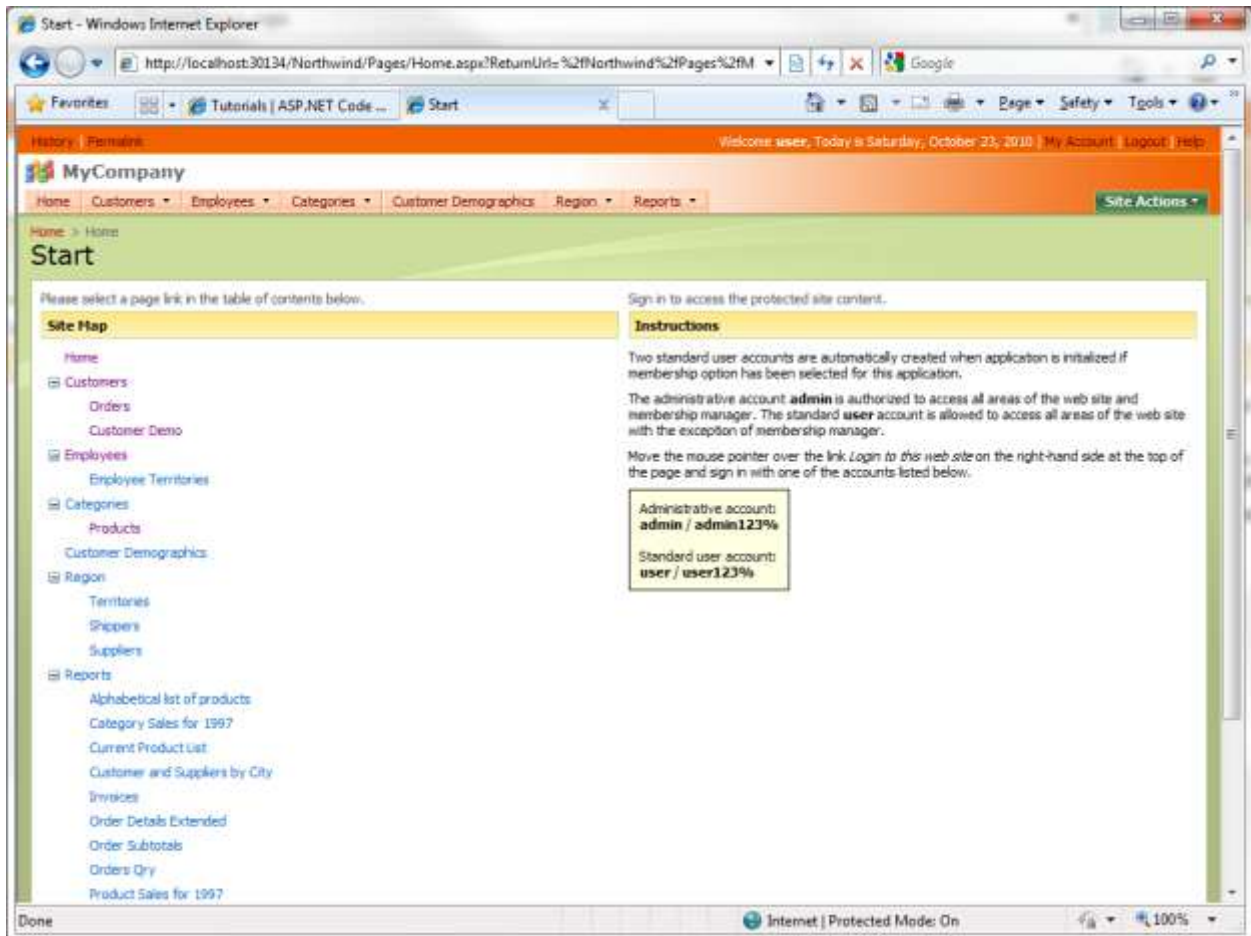


Navigate to the *Membership* page. On this page, we can create, modify, and delete *Users* and *Roles*. This enables a high degree of control over who can access the application. However, this page is only

accessible to those users with administrative privileges. When we sign out, you can see that we have been given a redirect URL, and taken back to the home page and prompted to log in.

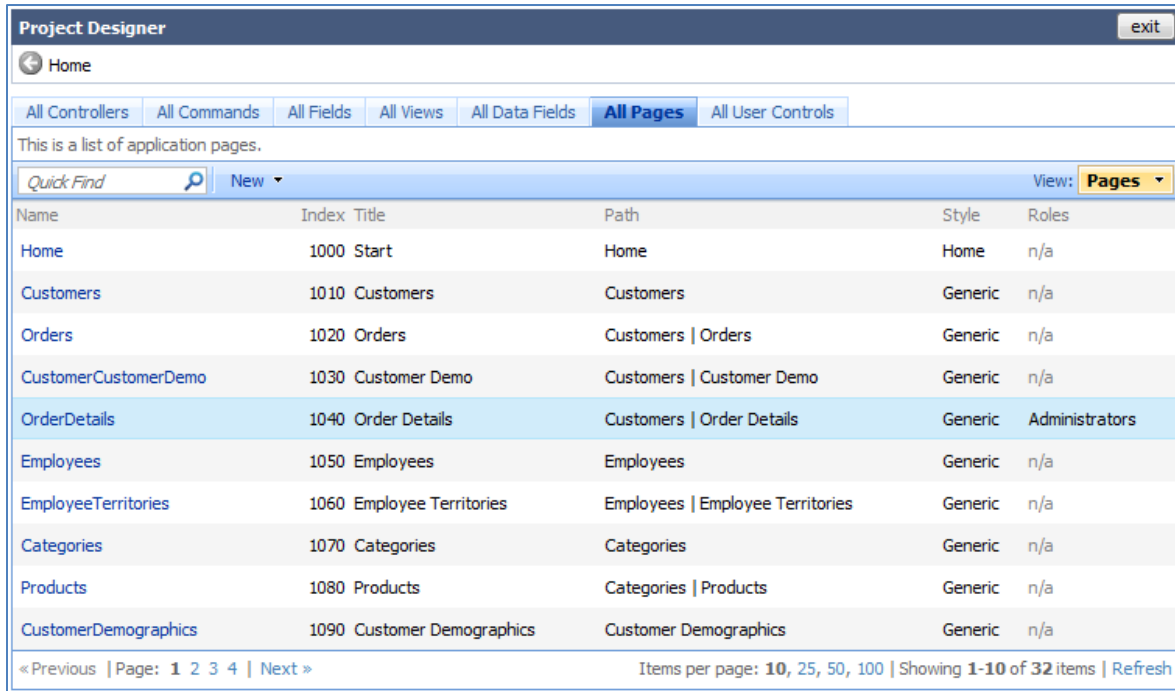


If we log in as a standard user, then we will be redirected again. The navigation enables access to all of the pages in the application except the *Membership* page, which is absent.

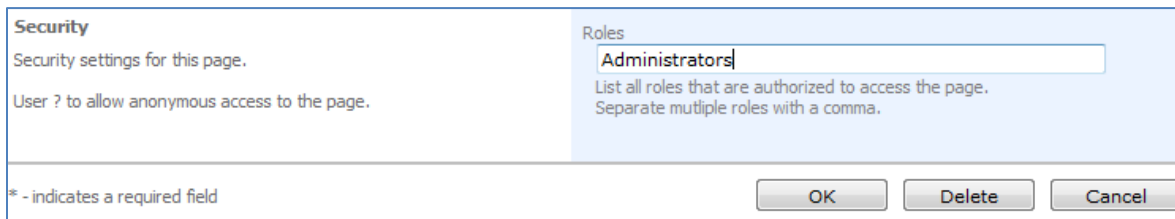


Let's change things up a bit. We will change the security so that only those users with administrative privileges will be able to access the *Order Details* page. Run *Code On Time Generator*, click on the project

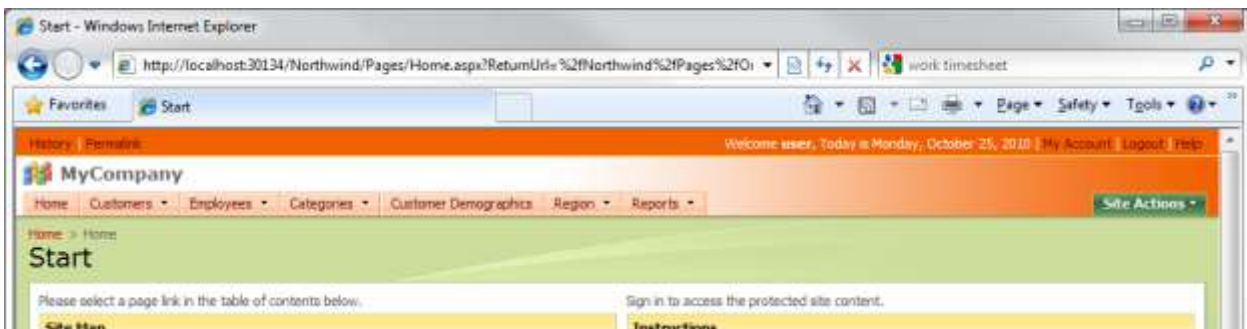
name, and then press the *Design* button. Click on the *All Pages* tab at the top, and select *Order Details* page.



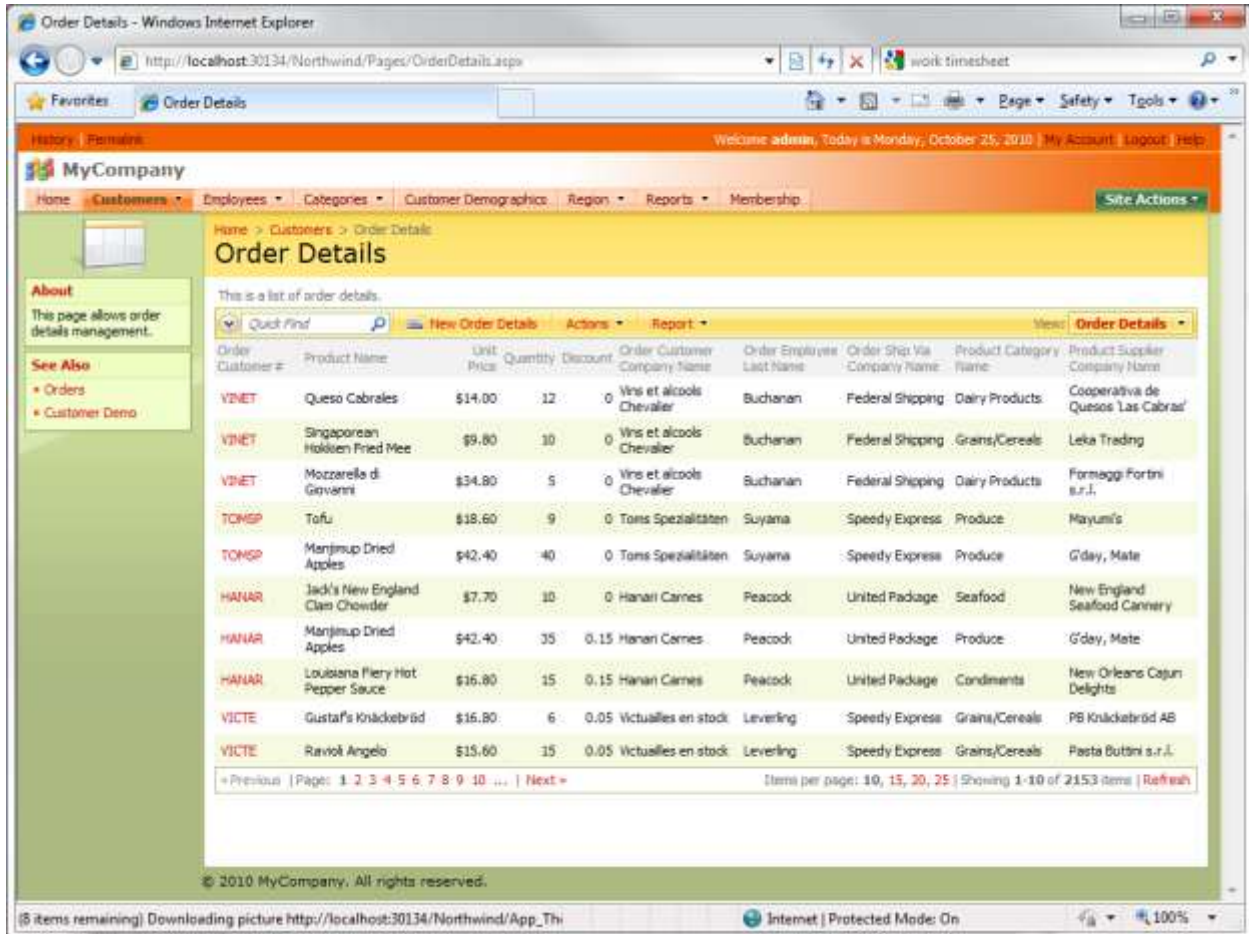
Press the *Edit* button, and change the *Roles* field to “Administrators”. This will insure that only users with the administrator’s role can see the page. Press *Ok*, then *Close*, and generate the application.



When the web application loads in your browser, you can see that you are still logged in as a standard user. The *Order Details* page is no longer available in the navigation. You can try entering the URL for *Order Details* page. This will only redirect you back to the home page.



If you logout, and log in as admin, the page will be available.



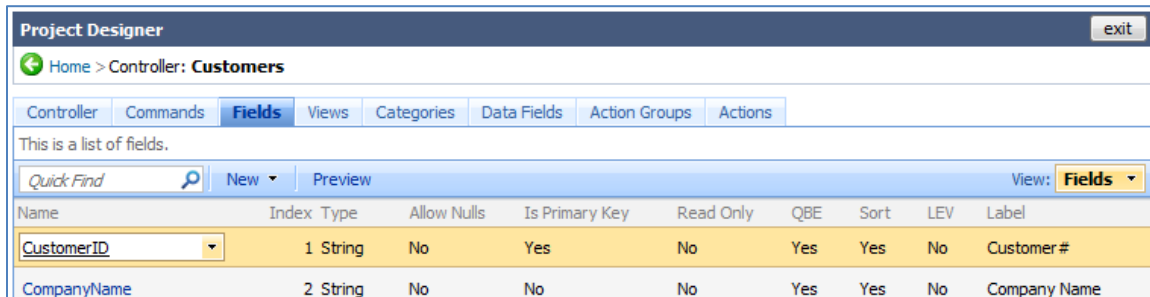
Fields

Fields can also be secured through roles. You can specify some roles to only view the field as read-only, and specify other roles to not view the field at all.

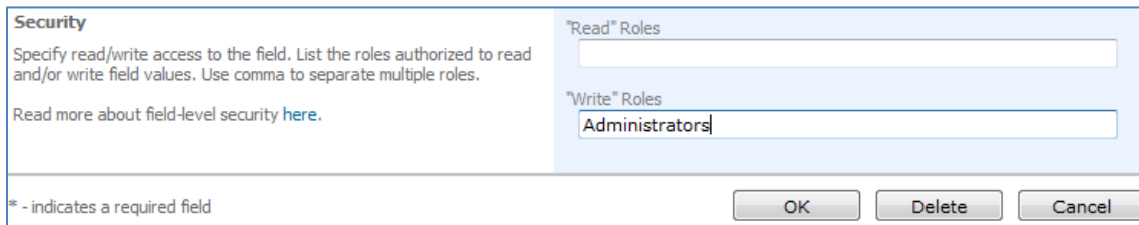
Suppose that you need to protect certain fields from modification and/or viewing by unwanted users. *Northwind* allows editing of the *CustomerID* field, which could be disastrous if changed. Let's make this a read only field for standard users.



Run *Code On Time Generator*, click on the name of the project, and press the *Design* button. Click on the *Customers* controller in the *All Controllers* list. Navigate to the *Fields* tab, and select the *CustomerID* field.



Edit *CustomerID*. Change the *Write Roles* field to "Administrators". If you wanted the field completely hidden to standard users, you can write "Administrators" in the *Read Field*. Save the field, press *Close*, and generate the application.



When the web application loads in the browser, log in as standard user. When you edit a customer record, the *CustomerID* field will be read only.



If we log in as administrator, the *CustomerID* field will be editable again.



Actions

Suppose that you have a need to prevent some users from executing certain actions. By default, the standard user account can create, edit, and delete products. Let's change this so that the standard user can only view the products. Administrator will be the only role that can create, edit, and delete. There are several locations we will have to modify, such as the dropdown, the buttons on the action bar, and the buttons in detail view.

This will be done by switching to *Code On Time Generator*, clicking on the project name, and pressing on the *Designer* button. Select the *Products* controller. Switch to the *Action Groups* tab.

Id	Scope	Header Text	Flat Rendering
ag1	Grid	n/a	No
ag2	Form	n/a	No
ag3	Action Bar	New	Yes
ag4	Action Bar	Edit/Delete	Yes
ag5	Action Bar	Actions	No
ag6	Action Bar	Record	No
ag7	Action Bar	Report	No
ag8	Row	n/a	No

Click on *ag1*, or *Action Group 1*. Click on the actions tab. You can see that there is *Select*, *Edit*, a blank action, *Duplicate*, and *New*. These actions correspond to the dropdown next to the product name in the application.

Edit the *Edit* action. In the *Roles* field, type in "Administrators". This will insure that only users with the *Administrators* role can edit. Perform the same with *Delete*, *Duplicate*, and *New*.

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
Select	editForm1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Grid
Edit	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Administrators	Grid
Delete	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Administrators	Grid
n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Grid
Duplicate	createForm1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Administrators	Grid
New	grid1		N/A		N/A				Administrators	Grid

Go back to the *Action Group* list. Navigate to *ag2*. Here is a long list of actions. Most of these are activated when a previous action has been selected, such as *Edit*, *New*, or *Duplicate*. In this case, the only actions that need modification would be the first two, without a *When Last Command Name* argument. Insert “Administrators” in their *Role* fields.

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	Administrators	Form
Delete			N/A		N/A				Administrators	Form
Cancel	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Form

Go back to the *Action Group* list, and click on *ag3*. This action is the *New Product* button on the action bar. Change the *Role* field to “Administrators”.

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
New	createForm1		N/A		N/A				Administrators	Action Bar

On the *Action Group* list, navigate to *ag4*. These are the buttons that appear on the action bar when you select a record, *Edit* and *Delete*. Add “Administrators” to the *Role* fields of these actions. Now, generate the application.

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	editForm 1	n/a	n/a	n/a	Yes	n/a	grid 1	n/a	Administrators	Action Bar
Delete			N/A		Yes		grid 1		Administrators	Action Bar

Login as user, and navigate to *Products*. First thing you will notice is that there is no *New Products* button. When you activate the dropdown menu, the only option available is *Select*. When you go into detail view, the only button available is *Close*.

The screenshot shows the MyCompany user interface for a user. The top navigation bar includes 'Home', 'Customers', 'Employees', 'Categories', 'Customer Demographics', 'Region', 'Reports', and 'Site Actions'. The main content area is titled 'Products' and contains a table of product information. A dropdown menu is open over the 'Chang' product row, showing only a 'Select' option. The table columns include Product Name, Supplier Company Name, Category Name, Quantity Per Unit, Unit Price, Units In Stock, Units On Order, Reorder Level, and Discontinued.

Product Name	Supplier Company Name	Category Name	Quantity Per Unit	Unit Price	Units In Stock	Units On Order	Reorder Level	Discontinued
Chai	Exotic Liquids	Beverages	10 boxes x 20 bags	\$18.00	39	0	10	No
Chang	Exotic Liquids	Beverages	24 - 12 oz bottles	\$19.00	17	40	25	No
Anseed Syru	Exotic Liquids	Condiments	12 - 550 ml bottles	\$10.00	13	70	25	No
Chef Ant's Cajun Seasoning	New Orleans Cajun Delights	Condiments	48 - 6 oz jars	\$22.08	53	0	0	No
Chef Ant's Gumbo Mix	New Orleans Cajun Delights	Condiments	36 boxes	\$21.36	0	0	0	Yes
Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments	12 - 8 oz jars	\$25.00	120	0	25	No
Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce	12 - 1 lb pkgs.	\$30.05	15	0	10	No
Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments	12 - 12 oz jars	\$40.00	6	0	0	No
Mishi Kobe Niku	Tokyo Traders	Meat/Poultry	18 - 500 g pkgs.	\$97.00	29	0	0	Yes
Ika	Tokyo Traders	Seafood	12 - 200 ml jars	\$31.00	31	0	0	No

Logout, and login as administrator. You can see that the *New*, *Edit*, and *Delete* buttons have returned.

The screenshot shows the MyCompany user interface for an administrator. The top navigation bar includes 'Home', 'Customers', 'Employees', 'Categories', 'Customer Demographics', 'Region', 'Reports', 'Membership', and 'Site Actions'. The main content area is titled 'Products' and contains the same table of product information. A dropdown menu is open over the 'Chang' product row, showing options for 'New Products', 'Select', 'Edit', 'Delete', 'Duplicate', and 'New'. The table columns are the same as in the previous screenshot.

Product Name	Supplier Company Name	Category Name	Quantity Per Unit	Unit Price	Units In Stock	Units On Order	Reorder Level	Discontinued
Chai	Exotic Liquids	Beverages	10 boxes x 20 bags	\$18.00	39	0	10	No
Chang	Exotic Liquids	Beverages	24 - 12 oz bottles	\$19.00	17	40	25	No
Anseed S	Exotic Liquids	Condiments	12 - 550 ml bottles	\$10.00	13	70	25	No
Chef Ant's Seasoning	New Orleans Cajun Delights	Condiments	48 - 6 oz jars	\$22.08	53	0	0	No
Chef Ant's	New Orleans Cajun Delights	Condiments	36 boxes	\$21.36	0	0	0	Yes
Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments	12 - 8 oz jars	\$25.00	120	0	25	No
Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce	12 - 1 lb pkgs.	\$30.05	15	0	10	No
Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments	12 - 12 oz jars	\$40.00	6	0	0	No
Mishi Kobe Niku	Tokyo Traders	Meat/Poultry	18 - 500 g pkgs.	\$97.00	29	0	0	Yes
Ika	Tokyo Traders	Seafood	12 - 200 ml jars	\$31.00	31	0	0	No

Row-level Security

Role vs. Row-Level Security

Let's compare role and row-level security. Roles are controlling access to shared resources of your application, such as pages, data fields, and actions. Row-level security further restricts access to individual items (rows) presented in the application data views.

Roles

Roles are equivalent to vertical restrictions in your application. Let's consider a few examples of vertical restrictions. Sales representatives are allowed to see a page with a list of *Orders*. Users from *Accounting* are allowed to see the *Commission* data field in the list of orders. Users with role *Admin* can access the *Membership Manager* and can create, edit, and delete any user.

Row-Level

Row-level security is equivalent to horizontal restrictions, applied on top of vertical restrictions. User *andrew.fuller* can only view his respective orders that are marked with his last name. User *federal.shipping* can see all orders that are "not shipped" or "shipped this or last week" if orders are assigned to shipper *Federal Shipping*.

Implementing Row-Level Security

How do you go about implementing row-level security?

1. Define application roles, such as *Sales*, *Shippers*, etc.
2. Assign roles to user accounts.
3. Create custom views available for each role.
4. Define when a custom view is displayed.

These four steps will provide row-level security. Step five will be to impose vertical security on the data by defining row-level filters for each custom view.

Real World Example

Let's consider a real world example. We'll use *Northwind* database. We'll set up row-level security with role *Sales*, *Shippers*, and *Customers*.

Creating Roles

Log in to the application as *Administrator*. Navigate to the *Membership Manager*, and switch to the roles tab. On the action bar, press the *New* button, and then *New Role*.



You will be taken to the role creation screen. In the *Role Name* field, type *Sales*. Press *Ok* to save the new role.

Home > Membership
Membership Manager

Users Roles

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

View: **New Role**

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

Role Name *
Sales

* - indicates a required field

OK Cancel

Create two more roles. One will have the name of *Shippers*, and the other will be called *Customers*. When finished, the list of roles should look like the image below.

Home > Membership
Membership Manager

Users Roles

This is a list of membership roles.

Quick Find New Actions View: **Roles**

Role Name	Description
Administrators	n/a
Customers	n/a
Sales	n/a
Shippers	n/a
Users	n/a

Showing 1-5 of 5 items | Refresh

Creating User Accounts

We need to set up several user accounts. In the *Membership Manager*, switch to *Users* tab. On the action bar, click on *New*, and press *New User*.

History | Permissions
MyCompany
Home Customers Employees Categories Customer Demographics Region Reports Membership Site Actions

Home > Membership
Membership Manager

Users Roles

This is a list of membership user records.

Quick Find New Actions View: **All Users**

New User
Create a new user record.

User Name	Is Approved	Is Locked Out	Create Date	Last Login Date	Comment
admin	Yes	No	4/25/2010 4:05 PM	5/25/2010 9:30 PM	n/a
user	Yes	No	4/25/2010 4:35 PM	5/25/2010 4:57 PM	n/a

Showing 1-2 of 2 items | Refresh

Create the user *Fuller*. The password will be “user123%”, which fulfills standard restrictions on the password imposed by ASP.NET. Give Mr. Fuller the roles of *Users* and *Sales*, and fill in the password recovery information.

Home > Membership
Membership Manager

Users Roles

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

View: **New User**

* - indicates a required field

New User Information
Please enter user name and password. Note that password must be at least 7 characters long and include one non-alphanumeric character. Only approved users will be able to login into the website.

User Name *
Fuller

Password *

Confirm Password *

This user account will be created as approved.

Roles
Please select user roles that most closely match user's responsibilities. Roles control access to the areas of this web site. Please contact system administrator if role access restrictions must be changed.

Roles:
 Administrators Sales Users
 Customers Shoppers

Password Recovery
These fields are required to help a user to recover a forgotten password. During the recovery process the user will be asked to enter a user name. If a user account exists then a security question is requested to be answered. A correct answer will trigger an email with a temporary password sent to the user.

Email *
user@user.com

Password Question *
Code On Time

Password Answer *
LLC

OK Cancel

Next, create the account *Federal Shipping*. It will have the same password. This account will have the roles *Users* and *Shippers*. Don't forget to fill in the password recovery information.

Home > Membership
Membership Manager

Users Roles

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

View: **New User**

* - indicates a required field

New User Information
Please enter user name and password. Note that password must be at least 7 characters long and include one non-alphanumeric character. Only approved users will be able to login into the website.

User Name *
Federal Shipping

Password *

Confirm Password *

This user account will be created as approved.

Roles
Please select user roles that most closely match user's responsibilities. Roles control access to the areas of this web site. Please contact system administrator if role access restrictions must be changed.

Roles:
 Administrators Sales Users
 Customers Shippers

Password Recovery
These fields are required to help a user to recover a forgotten password. During the recovery process the user will be asked to enter a user name. If a user account exists then a security question is requested to be answered. A correct answer will trigger an email with a temporary password sent to the user.

Email *
user@user.com

Password Question *
Code On Time

Password Answer *
LLC

OK Cancel

AROUT will be the last account created. AROUT is the *customer ID* of the company *Around The Horn*. This account will have the roles *Users* and *Customers*.

Home > Membership

Membership Manager

Users Roles

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

View: **New User** ▾

* - indicates a required field

New User Information

Please enter user name and password. Note that password must be at least 7 characters long and include one non-alphanumeric character. Only approved users will be able to login into the website.

User Name *

Password *

Confirm Password *

This user account will be created as approved.

Roles

Please select user roles that most closely match user's responsibilities. Roles control access to the areas of this web site. Please contact system administrator if role access restrictions must be changed.

Roles

Administrators Sales Users

Customers Shippers

Defining Role-Specific Views

Now we need to define role-specific views. *Run Code On Time Generator* and select the project from the project list. Click on the *Design* button. Select *Orders* data controller from the controller list.

YouTube | Blog | Newsgroup
http://www.codeontime.com

Review and modify properties of the project items and select *Close* to return to code generator.

Project Designer Close

← Home

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

This is a list of data controllers. View [tutorial](#) that shows how to work with the data controllers.

Quick Find ▾ View: **Controllers** ▾

Name	Generate	Conflict Detection	Handler	Annotations
Invoices	Yes	Overwrite Changes	n/a	n/a
OrderDetails	Yes	Overwrite Changes	n/a	n/a
OrderDetailsExtended	Yes	Overwrite Changes	n/a	n/a
Orders	Yes	Overwrite Changes	n/a	n/a
OrdersQry	Yes	Overwrite Changes	n/a	n/a
OrderSubtotals	Yes	Overwrite Changes	n/a	n/a
Products	Yes	Overwrite Changes	n/a	n/a
ProductsAboveAveragePrice	Yes	Overwrite Changes	n/a	n/a
ProductSalesfor1997	Yes	Overwrite Changes	n/a	n/a
ProductsbyCategory	Yes	Overwrite Changes	n/a	n/a

« Previous | Page: 1 2 3 | Next » Items per page: 10, 25, 50, 100 | Showing 11-20 of 29 items | Refresh

All customized project settings are stored in *.Log.xml files located in the root of your project folder.

Click on the *Views* tab at the top of the page. You should be on the page as shown below. On the action bar, press *New*, and click on *New View*.

Review and modify properties of the project items and select *Close* to return to code generator.

Project Designer Close

Home > Controller: **Orders**

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

This is a list of data controller views.

Quick Find New Preview View: **Views**

Id	Type	Command	Label	Header Text
createForm1			New Orders	\$DefaultCreateViewDescription
editForm1	Form	command1	Review Orders	\$DefaultEditViewDescription
grid1	Grid	command1	Orders	\$DefaultGridViewDescription

Showing 1-3 of 3 items | Refresh

▣ All customized project settings are stored in *.Log.xml files located in the root of your project folder.

Create a new *View* with the *Id* of *salesGrid1*. In the *Command* field, click on *(select)* and choose *command1* from the list. For *Label* field, type “My Orders”. The *Header Text* will be “This is a list of my orders”.

Project Designer Close

Home > Controller: **Orders**

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

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

View: **New View**

* - indicates a required field OK Cancel

General
Id and type of the view.

Id *
salesGrid1

Type *
 Grid
 Form

Command, Label & Header Text
Specify the command, label and header text for this view.

Command *
command1

Label *
My Orders

Header Text
This is a list of my orders.


Sort the orders by *OrderDate* in descending order by typing in “OrderDate desc” in the *Sort Expression* field. For *Base View ID*, select *grid1*. This will copy all data fields from *grid1*. Press Ok to save the view.

<p>Sort and Filter</p> <p>Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix.</p> <p>Filter expression must be compatible with the back-end database server syntax. Data field names used in filter are automatically expanded into appropriate SQL expressions as defined in command.</p> <p>Parameters must be prefixed by "@" or ":" symbol. You must implement a business rules class for the data controller with a property or field that matches the parameter name.</p>	<p>Sort Expression OrderDate desc</p> <p>Filter Expression</p>
<p>Virtualization</p> <p>Define the virtual view ID and a condition for virtualization to occur. The virtual view will be automatically replaced by this view when the condition is met.</p> <p>Specify a base view to inherit its data fields, categories, description, and label.</p>	<p>Virtual View Id (select)</p> <p>Override When</p> <p>Base View Id grid1</p>

Next, create a *View* with the *ID* *shippersGrid1*, with *Command* of *command1*. The *Label* will be “Orders To Ship”, and the *Header Text* will be “These orders must be shipped”. For *Sort Expression*, sort by *ShippedDate* in descending order. *Base View ID* will be *grid1*. Press Ok to save.

<p>Project Designer Close</p>	
<p>Home > Controller: Orders > View: shippersGrid1</p>	
<p>View Categories Styles Data Fields</p>	
<p>Please review view information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.</p>	
<p>Record View: View</p>	
<p>* - indicates a required field OK Delete Cancel</p>	
<p>General</p> <p>Id and type of the view.</p>	<p>Id * shippersGrid1</p> <p>Controller Orders</p> <p>Type Grid</p>
<p>Command, Label & Header Text</p> <p>Specify the command, label and header text for this view.</p>	<p>Command * command1</p> <p>Label * Orders To Ship</p> <p>Header Text These orders must be shipped.</p>
<p>Sort and Filter</p> <p>Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix.</p>	<p>Sort Expression ShippedDate desc</p>

The last view will go by the *ID* of *customersGrid1*. *Command* will be *command1*, *Label* will be “My Recent Orders”, and *Header Text* will be “Orders that were placed this year”. Sort all orders by *OrderDate* in descending order by typing in “OrderDate desc” in *Sort Expression*. *Base View ID* will be *grid1* as well.



[YouTube](#) | [Blog](#) | [Newsgroup](#)
<http://www.codeontime.com>

Review and modify properties of the project items and select *Close* to return to code generator.

Project Designer Close

← Home > Controller: Orders > View: **customersGrid1**

View
Categories
Styles
Data Fields

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

Record ▾
View: View ▾

OK
Delete
Cancel

* - indicates a required field

General Id and type of the view.	Id * <input style="width: 100%;" type="text" value="customersGrid1"/> Controller Orders Type Grid
Command, Label & Header Text Specify the command, label and header text for this view.	Command * <input style="width: 100%;" type="text" value="command1"/> Label * <input style="width: 100%;" type="text" value="My Recent Orders"/> Header Text <input style="width: 100%; height: 20px;" type="text" value="Orders that were placed this year."/>
Sort and Filter Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix. Filter expression must be compatible with the back-end database server syntax. Data field names used in filter are automatically expanded into appropriate SQL expressions as defined in command. Parameters must be prefixed by "@" or ":" symbol. You must implement a business rules class for the data controller with a property or field that matches the parameter name.	Sort Expression <input style="width: 100%;" type="text" value="OrderDate desc"/> Filter Expression <div style="border: 1px solid gray; height: 60px; width: 100%;"></div>
Virtualization Define the virtual view ID and a condition for virtualization to occur. The virtual view will be automatically replaced by this view when the condition is met. Specify a base view to inherit its data fields, categories, description, and label.	Virtual View Id <input style="width: 100%; border: 1px solid gray;" type="text" value="(select)"/> Override When <div style="border: 1px solid gray; height: 30px; width: 100%;"></div> Base View Id <input style="width: 100%;" type="text" value="grid1"/>

Views in Action

Let's take a look at these views in action. Regenerate the application using *Code On Time Generator*. When it finishes, navigate to the *Orders* page. If you select the *My Orders* view from the *View Selector*, you can see the custom description at the top, and that the *Order Date* field has been sorted in descending order. This list of orders is designed for salespeople.

The screenshot shows a web application interface for viewing orders. At the top, there is a breadcrumb trail: Home > Customers > Orders. Below this is a yellow header with the title 'Orders'. A message states 'This is a list of my orders.' Below the message is a toolbar with a search box, a 'New Orders' button, and an 'Actions' dropdown menu. A 'View Selector' dropdown is set to 'My Orders'. The main content is a table with the following columns: Customer Company Name, Employee Last Name, Order Date, Required Date, Shipped Date, Ship Via Company Name, Freight, Ship Name, Ship Address, and a 'View' column. The table contains 10 rows of order data. At the bottom, there is a pagination bar showing 'Items per page: 10, 15, 20, 25' and 'Showing 1-10 of 800 items | Refresh'.

Customer Company Name	Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via Company Name	Freight	Ship Name	Ship Address	View
Smors Istros	King	5/6/1998	6/3/1998	<=>	United Package	\$18.44	Smors Istros	Widewell 34	My Orders
Richter Supermarkt	Callahan	5/6/1998	6/3/1998	<=>	United Package	\$5.19	Richter Supermarkt	Stammweg 5	Orders To Ship
Bon app'	Peacock	5/6/1998	6/2/1998	<=>	United Package	\$38.28	Bon app'	12, rue des Bouchers	My Recent Orders
Raffinavale Canyon Grocery	Deviko	5/6/1998	6/3/1998	<=>	United Package	\$6.53	Raffinavale Canyon Grocery	2817 Milton Cr.	Albuquerque
Lehmans Marktstand	Fuller	5/5/1998	6/2/1998	<=>	Speedy Express	\$136.00	Lehmans Marktstand	Hagenstrweg 7	Frankfurt a.M.
LLA-Supermercado	Deviko	5/5/1998	6/2/1998	<=>	Speedy Express	\$0.93	LLA-Supermercado	Carrera 52 con Ave. Bolívar #65-98	Berqueto
Ernst Handel	Peacock	5/5/1998	6/2/1998	<=>	United Package	\$298.64	Ernst Handel	Kirchgasse 6	Graz
Pericles Comidas clásicas	Fuller	5/5/1998	6/2/1998	<=>	United Package	\$24.85	Pericles Comidas clásicas	Calle Dr. Jorge Cebal 321	México D.F.
Drachenbrot Dekontamin	Deviko	5/4/1998	5/18/1998	5/6/1998	United Package	\$7.98	Drachenbrot Dekontamin	Waterweg 21	Aachen
Queen Cozinha	Callahan	5/4/1998	6/1/1998	<=>	United Package	\$85.75	Queen Cozinha	Alameda dos Canários, 891	Sao Paulo

The view *Orders To Ship* lists all of the most recent orders in descending order. The view *My Recent Orders* is designed for customers. We do not expect end users to sign in, go to the *Orders* page, and select the correct view from the selector, so we need to indicate when custom views will override *grid1* and present themselves to the end users. Custom views will replace *grid1* at runtime. We'll use *Virtual View ID* to configure replacement, and use *Override When* to configure condition for replacement to take place.

Using OverrideWhen

In *Code On Time Generator*, click on the project name in the project list and press the *Design* button. Select the *Orders* data controller from the controller list, and switch to the *Views* tab.

The screenshot shows the 'Project Designer' interface. At the top, it says 'Home > Controller: Orders'. Below this is a tabbed interface with tabs for 'Controller', 'Commands', 'Fields', 'Views', 'Categories', 'Data Fields', 'Action Groups', and 'Actions'. The 'Views' tab is selected. A message states 'This is a list of data controller views.' Below this is a toolbar with a search box, a 'New' button, and a 'Preview' button. A 'View Selector' dropdown is set to 'Views'. The main content is a table with the following columns: Id, Type, Command, Label, and Header Text. The table contains 6 rows of view data. The 'salesGrid1' row is highlighted. At the bottom, there is a pagination bar showing 'Showing 1-6 of 6 items | Refresh'.

Id	Type	Command	Label	Header Text
createForm1	Form	command1	New Orders	\$DefaultCreateViewDescription
customersGrid1	Grid	command1	My Recent Orders	Orders that were placed this year.
editForm1	Form	command1	Review Orders	\$DefaultEditViewDescription
grid1	Grid	command1	Orders	\$DefaultGridViewDescription
salesGrid1	Grid	command1	My Orders	This is a list of my orders.
shippersGrid1	Grid	command1	Orders To Ship	These orders must be shipped.

Click on *salesGrid1*, and press *Edit*. Scroll down to the *Virtualization* section. Change *Virtual View ID* to *grid1*. In the *OverrideWhen* field, type in “Context.User.IsInRole(“Sales”)”. Save the view.

The screenshot shows a dialog box titled "Virtualization". On the left, there is explanatory text: "Define the virtual view ID and a condition for virtualization to occur. The virtual view will be automatically replaced by this view when the condition is met." and "Specify a base view to inherit its data fields, categories, description, and label." On the right, there are three input fields: "Virtual View Id" with the value "grid1", "Override When" with the value "Context.User.IsInRole(“Sales”)", and "Base View Id" with the value "grid1". At the bottom right are "OK", "Delete", and "Cancel" buttons. A footer note states: "All customized project settings are stored in *.Log.xml files located in the root of your project folder."

The view *shippersGrid1* will have a similar expression. Change *Virtual View ID* to *grid1*, and for *OverrideWhen*, type in “Context.User.IsInRole(“Shippers”)”.

The screenshot shows a dialog box titled "Virtualization". On the left, there is explanatory text: "Define the virtual view ID and a condition for virtualization to occur. The virtual view will be automatically replaced by this view when the condition is met." and "Specify a base view to inherit its data fields, categories, description, and label." On the right, there are three input fields: "Virtual View Id" with the value "grid1", "Override When" with the value "Context.User.IsInRole(“Shippers”)", and "Base View Id" with the value "grid1". At the bottom right are "OK", "Delete", and "Cancel" buttons. A footer note states: "All customized project settings are stored in *.Log.xml files located in the root of your project folder."

Perform the same operation for *customersGrid1*. Change *Virtual View ID* to *grid1*, and *OverrideWhen* to “Context.User.IsInRole(“Customers”)”.

The screenshot shows a dialog box titled "Virtualization". On the left, there is explanatory text: "Define the virtual view ID and a condition for virtualization to occur. The virtual view will be automatically replaced by this view when the condition is met." and "Specify a base view to inherit its data fields, categories, description, and label." On the right, there are three input fields: "Virtual View Id" with the value "grid1", "Override When" with the value "Context.User.IsInRole(“Customers”)", and "Base View Id" with the value "grid1". At the bottom right are "OK", "Delete", and "Cancel" buttons. A footer note states: "All customized project settings are stored in *.Log.xml files located in the root of your project folder."

Row-Level View Filters

Now we need to define row-level view filters. These will insure that sales and customers will see their respective orders, and that shippers will see orders that were not shipped or shipped this or last week.

For *salesGrid1*, type in the *Filter Expression* of “EmployeeLastName = \$UserName()”.

Sort and Filter Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix. Filter expression must be compatible with the back-end database server syntax. Data field names used in filter are automatically expanded into appropriate SQL expressions as defined in command. Parameters must be prefixed by "@" or ":" symbol. You must implement a business rules class for the data controller with a property or field that matches the parameter name.	Sort Expression <input type="text" value="OrderDate desc"/> Filter Expression <input type="text" value="EmployeeLastName = \$UserName()"/>
---	---

For *shippersGrid1*, type in the *Filter Expression* of:

```
ShipViaCompanyName = $UserName() and  
(ShippedDate is null or  
  (  
    $ThisWeek(ShippedDate) or  
    $LastWeek(ShippedDate)  
  )  
)
```

Sort and Filter Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix. Filter expression must be compatible with the back-end database server syntax. Data field names used in filter are automatically expanded into appropriate SQL expressions as defined in command. Parameters must be prefixed by "@" or ":" symbol. You must implement a business rules class for the data controller with a property or field that matches the parameter name.	Sort Expression <input type="text" value="ShippedDate desc"/> Filter Expression <input type="text" value="ShipViaCompanyName = \$UserName() and
(ShippedDate is null or
 (
 \$ThisWeek(ShippedDate) or
 \$LastWeek(ShippedDate)
)
)"/>
---	--

Lastly, set the *Filter Expression* of *customersGrid1* to

```
CustomerID = $UserName() and  
$YearToDate(OrderDate)
```

Sort and Filter Sort expression is a list of data field names of this view, each followed by optional <i>asc</i> or <i>desc</i> suffix. Filter expression must be compatible with the back-end database server syntax. Data field names used in filter are automatically expanded into appropriate SQL expressions as defined in command. Parameters must be prefixed by "@" or ":" symbol. You must implement a business rules class for the data controller with a property or field that matches the parameter name.	Sort Expression <input type="text" value="OrderDate desc"/> Filter Expression <input type="text" value="CustomerID = \$UserName() and
\$YearToDate(OrderDate)"/>
---	---

Close the designer, and regenerate the application.

Viewing the Results

First, we sign in as Fuller. The password is *user123%*. Click *Login*, and navigate to the *Orders* page. You can see that all the orders listed are those with the *Employee Last Name* of *Fuller*, and there are no other filtering options available. The orders are sorted according to *OrderDate*.

This is a list of my orders.

Customer Company Name	Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via Company Name	Freight	Ship Name	Ship Address	Ship City
		5/5/1998	6/2/1998	n/a	Speedy Express	\$136.00	Lehmanns Marktstand	Magazinweg 7	Frankfurt a.M.
		5/5/1998	6/2/1998	n/a	United Package	\$24.95	Perides Comidas clásicas	Calle Dr. Jorge Cash 321	México D.F.
		4/30/1998	5/28/1998	5/4/1998	United Package	\$10.98	Franchi S.p.A.	Via Monte Bianco 34	Torino
		4/29/1998	6/10/1998	n/a	United Package	\$85.80	Ricardo Adocicados	Av. Copacabana, 267	Rio de Janeiro
Piccolo und mehr	Fuller	4/27/1998	5/25/1998	4/29/1998	United Package	\$53.05	Piccolo und mehr	Geislweg 14	Salzburg
Comércio Mineiro	Fuller	4/22/1998	5/6/1998	5/1/1998	Speedy Express	\$29.99	Comércio Mineiro	Av. dos Lusíadas, 23	Sao Paulo
Suprêmes délices	Fuller	4/20/1998	5/18/1998	4/24/1998	United Package	\$0.17	Suprêmes délices	Boulevard Tirou, 255	Charleroi
White Clover Markets	Fuller	4/17/1998	5/15/1998	4/23/1998	Federal Shipping	\$606.19	White Clover Markets	1029 - 12th Ave. S.	Seattle
Königlich Essen	Fuller	4/16/1998	5/14/1998	4/22/1998	Speedy Express	\$29.59	Königlich Essen	Maubelstr. 90	Brandenburg
Ottlies Käseladen	Fuller	4/14/1998	5/12/1998	4/16/1998	United Package	\$43.30	Ottlies Käseladen	Mehrheimerstr. 369	Köln

« Previous | Page: 1 2 3 4 5 6 7 8 9 10 | Next » Items per page: 10, 15, 20, 25 | Showing 1-10 of 96 items | Refresh

Let's sign out, and sign in again as *Federal Shipping*, with the password of *user123%*. On the *Orders* page, you can see that all of the orders have *Federal Shipping* as *Ship Via Company Name*, and no other filtering options are available. It is sorted in descending order by *Shipped Date*. You can see that the orders listed are those shipped this week, last week, or were never shipped.

These orders must be shipped.

Customer Company Name	Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via Company Name	Freight	Ship Name	Ship Address	Ship City
Ernst Handel	King	4/8/1998	5/6/1998	n/a	Federal Shipping	\$79.46	Ernst Handel	Kirchgasse 6	Graz
Rancho grande	Suyama	4/13/1998	5/11/1998	n/a	Federal Shipping	\$3.17	Rancho grande	Av. del Libertador 900	Buenos Aires
Great Lakes Food Market	Peacock	4/22/1998	5/20/1998	n/a	Federal Shipping	\$18.84	Great Lakes Food Market	2732 Baker Blvd.	Eugene
La maison d'Asie	King	4/27/1998	5/25/1998	n/a	Federal Shipping	\$2.79	La maison d'Asie	1 rue Alsace-Lorraine	Toulouse
Blauer See Delikatessen	Dodsworth	4/29/1998	5/27/1998	n/a	Federal Shipping	\$31.14	Blauer See Delikatessen	Forsterstr. 57	Mannheim
Great Lakes Food Market	Peacock	4/30/1998	6/11/1998	n/a	Federal Shipping	\$14.01	Great Lakes Food Market	2732 Baker Blvd.	Eugene

Showing 1-6 of 6 items | Refresh

This time, sign in as *AROUT*, with the password *user123%*. You can see that the list of orders has only one record in it, which belongs to the company of *AROUT*, and it is sorted in descending order of *Order Date*.

Home > Customers > Orders

Orders

Orders that were placed this year.

Customer Company Name	Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via Company Name	Freight	Ship Name	Ship Address	Ship City
Around the Horn	Fuller	9/7/2010	9/9/2010	9/10/2010	Federal Shipping	\$0.00	n/a	n/a	n/a

Showing 1-1 of 1 items | Refresh

Ideal Implementation

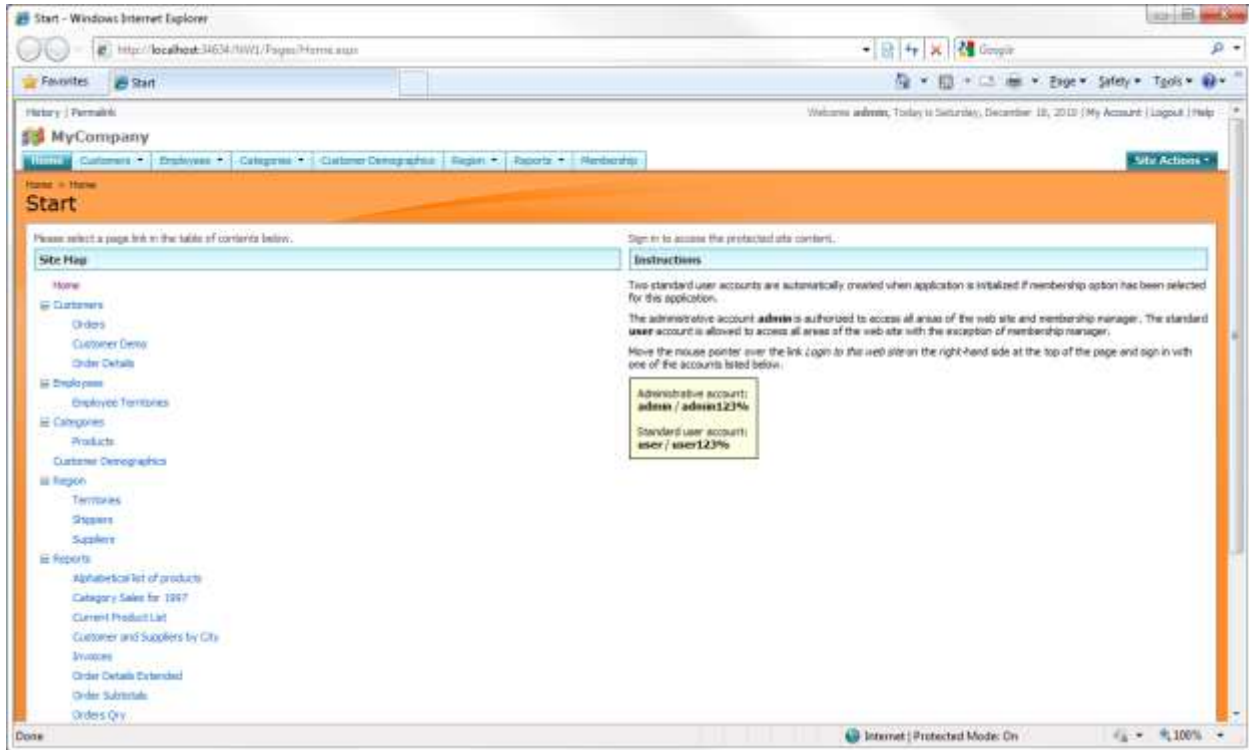
The implementation that we have created is not ideal. *Northwind* database was not designed for row-level security. Ideally, *UserName* or *UserId* column would be available in the tables *Employees*, *Shippers*, and *Customers*. Filters with the *\$UserName()* or *\$UserId()* functions would be matched to the corresponding data fields.

If you were to have the *UserName* field in the specified tables, then a de-normalization field map will need to be set up to automatically include *UserName* in the respective data controllers.

You would also need to change your filters by replacing the data fields *EmployeeLastName*, *ShipViaCompanyName*, and *CustomerCompanyName*.

Custom Page Background

Below is a picture of the standard home page of a *Web Site Factory* application generated by *Code On Time Generator*. You can quickly add a custom background to this or any other page with a custom CSS stylesheet.

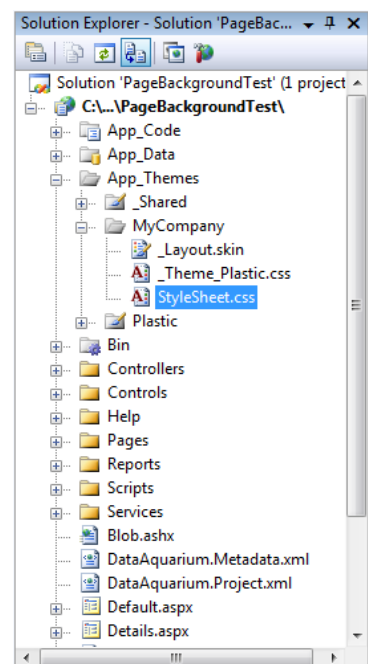


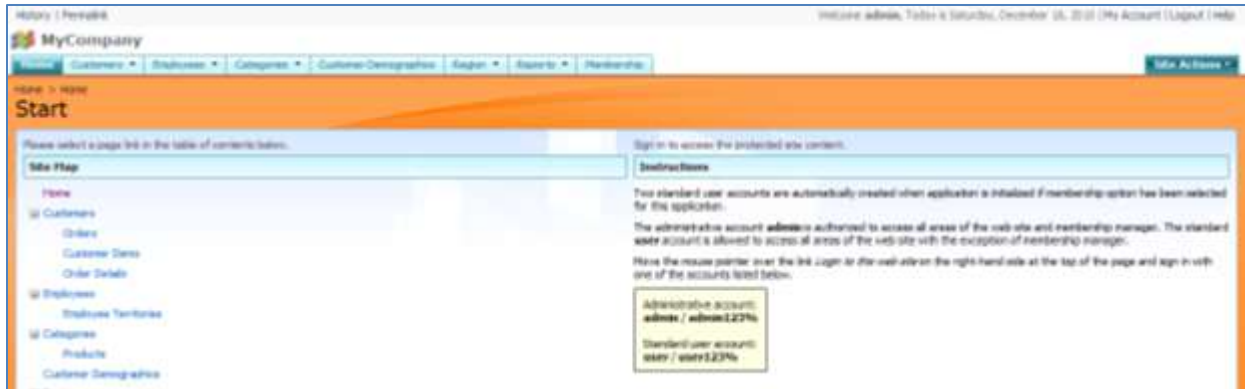
Open your project in *Visual Studio* or *Visual Web Developer* and create a new CSS style sheet in the same `~/App_Themes` folder of your project that contains the `_Layout.skin` file.

Type the following CSS rules in the style sheet:

```
.pages_home_aspx #PageContent
{
    background-image: url(../_Shared/SettingsGraphic.jpg);
    background-repeat: repeat-x;
}
```

The rule will change the home page content to display the standard `SettingsGraphics.jpg` image as the background image across the top of the page, as shown on the next page.

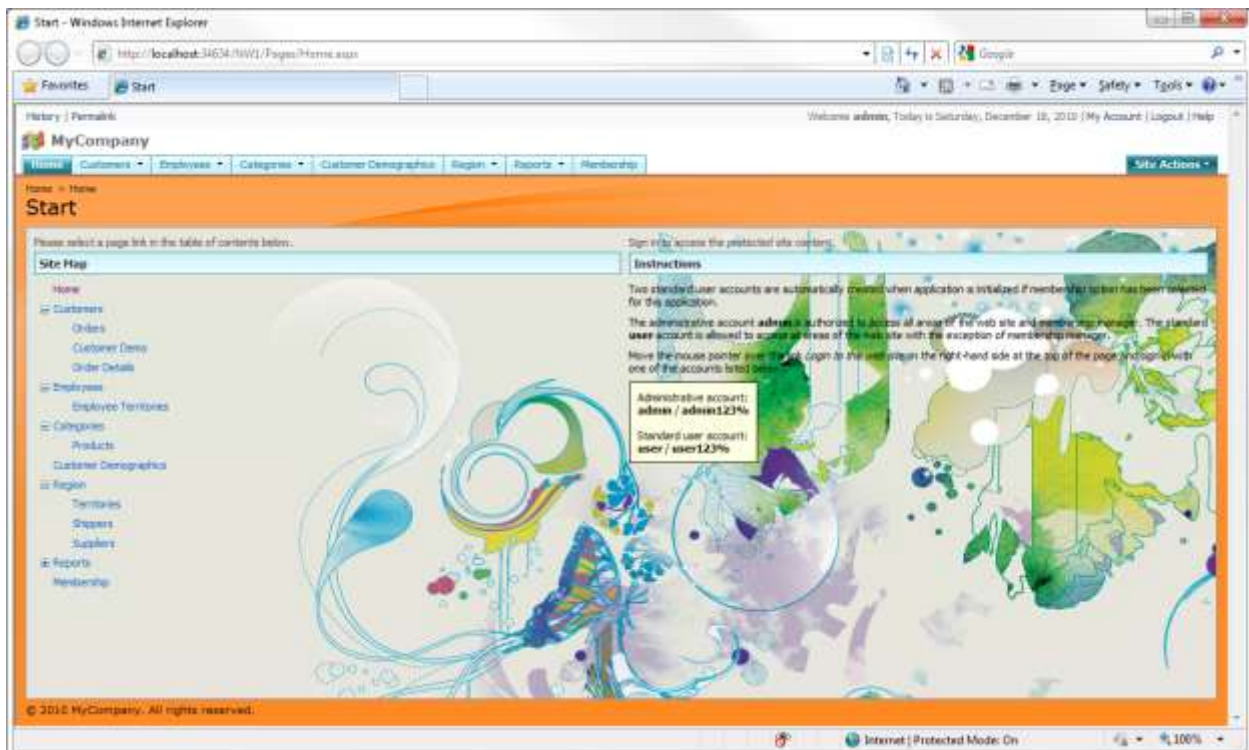




In this example, we copied the standard Windows 7 background wallpaper *img24.jpg* into the folder with the CSS file and changed the CSS rule to:

```
.pages_home_aspx #PageContent
{
    background-image: url(img24.jpg);
    background-repeat: repeat-x;
}
```

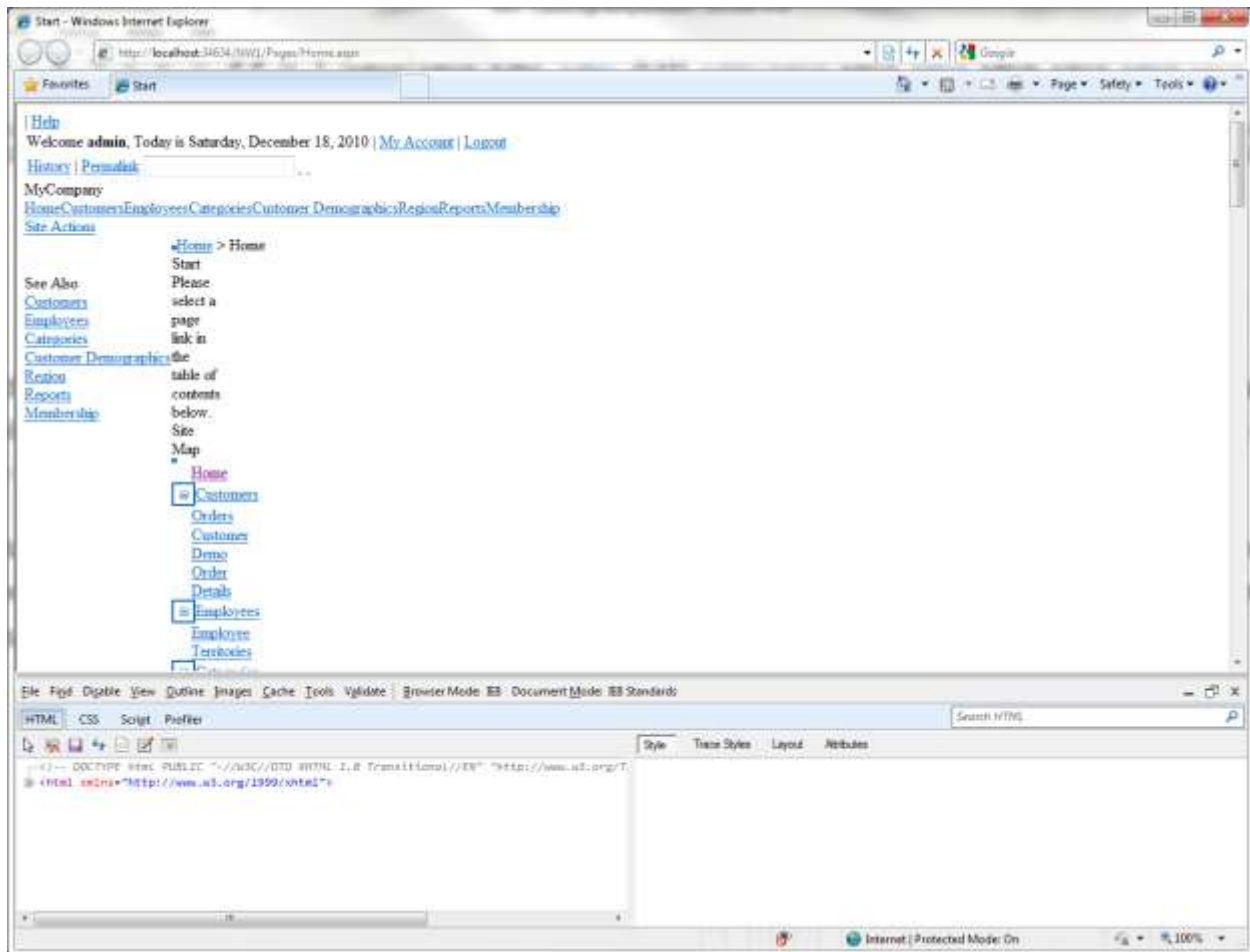
The home page of the application has changed, as shown below.



The CSS class *pages_home_aspx* is automatically assigned to the home page by the application framework. In fact, every page of a *Web Site Factory* application automatically assigns its own class name to the content container element. This allows creating CSS rules with precise targeting of

individual pages. The styling of all pages in the generated web applications is controlled entirely by a collection of CSS stylesheets that make your application look like *Microsoft SharePoint Services* web site by default.

Disable the stylesheet and it turns into a “black and white” canvas. This can be done by using *Developer Tools* in *Internet Explorer 8*. On your keyboard, press *F12*. In the menu of the screen that appears, press *Disable | CSS*.



If you feel an inspiration, then go ahead and create your own masterpiece!

Creating Three Level Master Detail

Here we have a three level master detail page. In its initial state you can only see *Customers*. If you select a record by clicking on its row, a second view will appear underneath the list of *Customers*. This list shows *Orders* relevant to the selected *Customer*. If you select an *Order* from the list, you will see a list of *Details* appear underneath. When you scroll down the page, a summary of the selected customer will stay visible in the top left corner of your screen. This three level master detail layout is a very quick and efficient way of going through the records.

The screenshot displays a web application interface for a 3-Level Master-Detail layout. The browser window title is "3-Level Master-Detail - Windows Internet Explorer". The address bar shows "http://localhost:43830/MyCompanyDemo/Pages/ThreeLevelMasterDetail.aspx". The page title is "3-Level Master-Detail".

The interface includes a navigation bar with the following items: Home, 3-Level Master-Detail, Customers List, Employees, Categories, Customer Demographics, Regions, Reports, and New Page. A sidebar on the left contains a "Summary" section for the selected customer (ALFRI) and a "See Also" section with links to Home, Customers, Employees, Categories, Customer Demographics, Region, and Reports.

The main content area is divided into three sections:

- Customers:** This section displays a list of customers. The table has columns: Customer #, Company Name, Contact Name, Contact Title, Address, City, Region, Postal Code, Country, and Phone. The data rows are:

Customer #	Company Name	Contact Name	Contact Title	Address	City	Region	Postal Code	Country	Phone
ALFRI	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) 555-4728
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 3DP	UK	(171) 555-7788
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvägen 8	Luleå	n/a	S-958 22	Sweden	0921-12 34 65
- Orders:** This section displays a list of orders. The table has columns: Employee Last Name, Order Date, Required Date, Shipped Date, Ship Via, Company Name, Freight, Ship Name, Ship Address, and Ship City. The data rows are:


Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via	Company Name	Freight	Ship Name	Ship Address	Ship City
Suyama	8/25/1997	9/22/1997	9/2/1997	Speedy Express	Alfreds Futterkiste	\$25.46	Alfreds Futterkiste	Obere Str. 57	Berlin
Peacock	10/3/1997	10/31/1997	10/13/1997	United Package	Alfreds Futterkiste	\$61.02	Alfreds Futterkiste	Obere Str. 57	Berlin
Peacock	10/13/1997	11/24/1997	10/21/1997	Speedy Express	Alfreds Futterkiste	\$23.94	Alfreds Futterkiste	Obere Str. 57	Berlin
Davolio	1/15/1998	2/12/1998	1/21/1998	Federal Shipping	Alfreds Futterkiste	\$69.53	Alfreds Futterkiste	Obere Str. 57	Berlin
Davolio	3/16/1998	4/27/1998	3/24/1998	Speedy Express	Alfreds Futterkiste	\$40.42	Alfreds Futterkiste	Obere Str. 57	Berlin
- Order Details:** This section displays a list of order details. The table has columns: Product Name, Unit Price, Quantity, Discount, Order Customer Company Name, Order Employee Last Name, Order Ship Via Company Name, Product Category Name, and Product Supplier Company Name. The data row is:

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
Veggie-spread	\$41.90	20	0	Alfreds Futterkiste	Peacock	United Package	Condiments	Pavlova, Ltd.

The page footer includes "© 2010 MyCompany. All rights reserved." and "Internet | Protected Mode: On".

To start creating a three level master detail data layout, open *Code On Time Generator*, select the project name, and press *Design*. Switch to the *All Pages* tab. On the action bar, press *New / New Page*.

Give the page the *Name* of “ThreeLevelMasterDetail”, with *Index* of “1005”, and *Title* and *Path* of “3-Level Master-Detail”. Change *Style* to “Miscellaneous”, and *About This Page* will be “This page will demo a three level master-detail data layout.” Make the *Roles* field blank.


YouTube | Blog | Newsgroup
http://www.codeontime.com

Review and modify properties of the project items and select *Close* to return to code generator.

Project Designer
exit

Home

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

<p>General</p> <p>Name and index of the page. The address of the generated ASP.NET page is <code>~/Pages/Name.aspx</code> where Name is the specified name.</p> <p>Use <i>External Url</i> to create a menu link to an external web site. No physical application page is generated if <i>External Url</i> is not blank.</p> <p>If <i>External Url</i> is equal to <code>about:blank</code> then a site map node without a Url is created.</p>	<p>Name *</p> <input type="text" value="ThreeLevelMasterDetail"/> <p>Index</p> <input type="text" value="1005"/> <p>External Url</p> <input type="text"/>
<p>Presentation</p> <p>Page title is displayed in the title of the browser window.</p> <p>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 than the index of this page.</p> <p>If <i>Path</i> is left blank then there will be no menu option to access the page.</p> <p>Page description is displayed as a tool tip of the corresponding menu option.</p> <p>Custom style is one or more CSS classes. Use <i>Wide</i> as custom style to eliminate the side bar on the page.</p> <p>The page will feature <i>About</i> box on the side bar if specified.</p>	<p>Title *</p> <input type="text" value="3-Level Master-Detail"/> <p>Path</p> <input type="text" value="3-Level Master-Detail"/> <p>Description</p> <input style="height: 30px;" type="text"/> <p>Style *</p> <input type="text" value="Miscellaneous"/> <p>Custom Style</p> <input type="text"/> <p>About This Page</p> <input style="height: 30px;" type="text" value="This page will demo a three level master-detail data layout."/>
<p>Security</p> <p>Security settings for this page.</p> <p>User ? to allow anonymous access to the page.</p>	<p>Roles</p> <input type="text"/> <p>List all roles that are authorized to access the page. Separate multiple roles with a comma.</p>

* - indicates a required field

All customized project settings are stored in *.Log.xml files located in the root of your project folder.

Save the page, and select the new page from the list of *All Pages*. Switch to the *Containers* tab, and create a new container. It will have *Flow* of “New Row”.

The screenshot shows the 'Project Designer' interface. At the top, it says 'Home > Page: ThreeLevelMasterDetail'. Below that, there are tabs for 'Page', 'Containers', 'Data Views', and 'Controls'. A message reads: 'Please fill this form and click OK button to create a new container record. Click Cancel to return to the previous screen.' The 'View' dropdown is set to 'New Container'. The 'General' section contains instructions: '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.' The form fields are: 'Id' (N/A), 'Flow*' (New Row), 'Width' (empty), 'CSS Class Name' (empty), and 'CSS Style Properties' (empty). At the bottom, there are 'OK' and 'Cancel' buttons and a note: '* - indicates a required field'.

Save, and create another container with *Flow* of “New Row”. In *CSS Style Properties*, write “padding-top:8px;”.

This screenshot is identical to the previous one, but the 'CSS Style Properties' field now contains the text 'padding-top:8px;'.

Now, create a third container with the same settings as before.

This screenshot is identical to the previous one, showing the 'New Container' dialog box with 'padding-top:8px;' in the 'CSS Style Properties' field.

Now, switch to the *Data Views* tab. Create a new *Data View*, and place it in *Container* “c100”, with *Controller* of “Customers”, and *View* of “grid1”. *Text* will be “Customers”, and *Page Size* will be “5”. Enable “Show in Summary”.

Review and modify properties of the project items and select *Close* to return to code generator.

Project Designer exit

Home > Page: **ThreeLevelMasterDetail**

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

<p>General</p> <p>Page, container, controller, and controller view of the data view.</p> <p>Use <i>Tag</i> to enabled conditional controller actions with matching <i>whenTag</i> property and to write custom business rules specific to tagged data views.</p>	<p>Id N/A</p> <p>Container * c100</p> <p>Controller * Customers</p> <p>View grid1</p> <p>Tag <input type="text"/></p> <p>Transaction N/A</p>
<p>Activator</p> <p>Specify a method of view activation available to end users. <i>Text</i> attribute will represent a tab or menu option of activator.</p> <p>Use <i>Sequence</i> to order controls and data views placed in the same container.</p>	<p>Activator None</p> <p>Text Customers</p> <p>Sequence <input type="text"/></p>
<p>Presentation</p> <p>Presentation properties of the data view.</p>	<p><input checked="" type="checkbox"/> Show In Summary</p> <p>Page Size 5</p> <p>Selection Mode * N/A</p> <p><input checked="" type="checkbox"/> Show Action Bar</p> <p><input checked="" type="checkbox"/> Show View Description</p> <p><input checked="" type="checkbox"/> Show View Selector</p> <p><input checked="" type="checkbox"/> Show Paggers</p> <p><input type="checkbox"/> Show ModalForms</p>

Save, and create another data view. This one will have *Container* of “c101”, *Controller* of “Orders”, *View* of “grid1”, *Text* of “Orders”, and *Page Size* of “5”. Disable “Show View Selector”. The *Filter Source* will be “dv100” and *Filter Fields* will be “CustomerID”. Set *Auto Hide* to “Container”.

<p>General</p> <p>Page, container, controller, and controller view of the data view.</p> <p>Use <i>Tag</i> to enabled conditional controller actions with matching <i>whenTag</i> property and to write custom business rules specific to tagged data views.</p>	<p>Id N/A</p> <p>Container * c101 </p> <p>Controller * Orders </p> <p>View grid1 </p> <p>Tag <input type="text"/></p> <p>Transaction N/A ▼</p>
<p>Activator</p> <p>Specify a method of view activation available to end users. <i>Text</i> attribute will represent a tab or menu option of activator.</p> <p>Use <i>Sequence</i> to order controls and data views placed in the same container.</p>	<p>Activator None ▼</p> <p>Text Orders <input type="text"/></p> <p>Sequence <input type="text"/></p>
<p>Presentation</p> <p>Presentation properties of the data view.</p>	<p><input type="checkbox"/> Show In Summary</p> <p>Page Size 5 <input type="text"/></p> <p>Selection Mode * N/A ▼</p> <p><input checked="" type="checkbox"/> Show Action Bar</p> <p><input checked="" type="checkbox"/> Show View Description</p> <p><input type="checkbox"/> Show View Selector</p> <p><input checked="" type="checkbox"/> Show Paggers</p> <p><input type="checkbox"/> Show Modal Forms</p> <p><input type="checkbox"/> Search on Start</p> <p><input checked="" type="checkbox"/> Show Details in List Mode</p>
<p>Filter</p> <p>Filter parameters can be used to limit the visible data or to establish master-detail relationships.</p> <p>Property <i>Auto Hide</i> specifies user interface element that will be hidden if runtime filter value is empty and view can be automatically hidden. Use <i>Self</i> when master data view has a <i>Tab</i> activator and belongs to the same page container as this data view. Use <i>Container</i> otherwise.</p>	<p>Filter Source dv100 </p> <p>Filter Fields CustomerID </p> <p>Auto Hide Container ▼</p>

Create one more data view. *Container* will be “c102”, *Controller* will be “OrderDetails”, *View* will be “grid1”, *Text* will be “Details”, and *Page Size* will be “5”. Disable “Show View Selector”. *Filter Source* will be “dv101” and *Filter Fields* will be “OrderID”. Set *Auto Hide* to “Container”.

<p>General Page, container, controller, and controller view of the data view.</p> <p>Use <i>Tag</i> to enabled conditional controller actions with matching <i>whenTag</i> property and to write custom business rules specific to tagged data views.</p>	<p>Id N/A</p> <p>Container * c102</p> <p>Controller * OrderDetails</p> <p>View grid1</p> <p>Tag <input type="text"/></p> <p>Transaction N/A</p>
<p>Activator Specify a method of view activation available to end users. <i>Text</i> attribute will represent a tab or menu option of activator.</p> <p>Use <i>Sequence</i> to order controls and data views placed in the same container.</p>	<p>Activator None</p> <p>Text Details</p> <p>Sequence <input type="text"/></p>
<p>Presentation Presentation properties of the data view.</p>	<p><input type="checkbox"/> Show In Summary</p> <p>Page Size 5</p> <p>Selection Mode * N/A</p> <p><input checked="" type="checkbox"/> Show Action Bar</p> <p><input checked="" type="checkbox"/> Show View Description</p> <p><input type="checkbox"/> Show View Selector</p> <p><input checked="" type="checkbox"/> Show Paggers</p> <p><input type="checkbox"/> Show Modal Forms</p> <p><input type="checkbox"/> Search on Start</p> <p><input checked="" type="checkbox"/> Show Details in List Mode</p>
<p>Filter Filter parameters can be used to limit the visible data or to establish master-detail relationships.</p> <p>Property <i>Auto Hide</i> specifies user interface element that will be hidden if runtime filter value is empty and view can be automatically hidden. Use <i>Self</i> when master data view has a <i>Tab</i> activator and belongs to the same page container as this data view. Use <i>Container</i> otherwise.</p>	<p>Filter Source dv101</p> <p>Filter Fields OrderID</p> <p>Auto Hide Container</p>

Save the data view, close the *Designer*, and generate the application. When the page loads, sign in and navigate to the *3-Level Master-Detail* page to see your new page in action.

Grouping Tabbed Data Views

We learned how to create a three level master detail data layout on a new page in your application. If you wanted to use tabs to group your data views, you will have to take different steps.

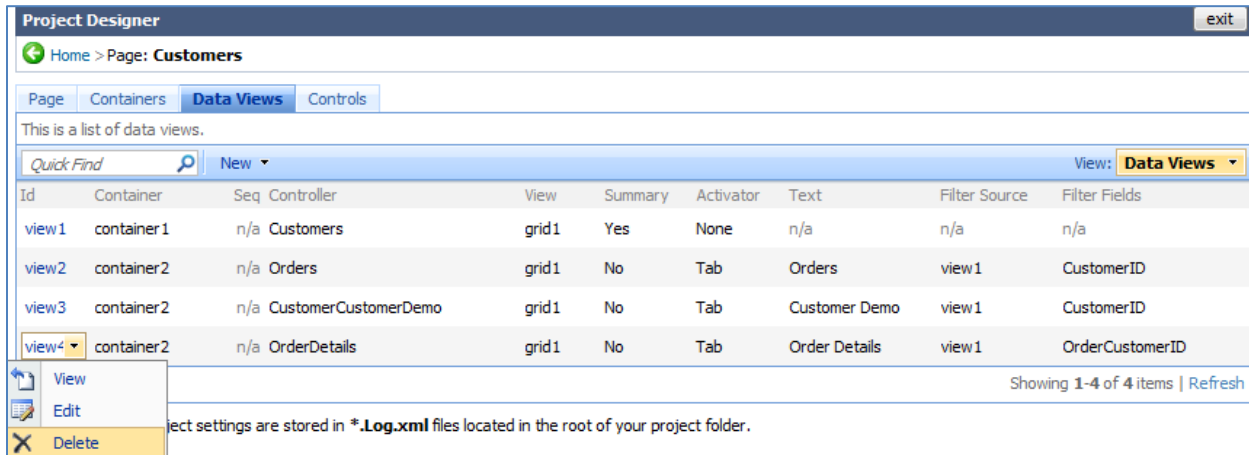
If you generate a *Code On Time* application from the *Northwind* database using *Classic* layout, then the *Customers* page will look like the picture below. When you select a customer from the list, a tabbed view will appear underneath with relevant information, from the *Orders*, *Customer Demo*, and *Order Details* tables. The default page provides only a two level master detail, and displays all relevant *Order Details* to the selected customer. We would like a list of *Order Details* to appear underneath *Orders* list, and to be filtered according to the selected *Order*.

The screenshot shows a web browser window displaying a web application. The page title is "Customers - Windows Internet Explorer". The address bar shows "http://localhost:35180/GroupTabbedDataViews/Pages/Customers.aspx". The page has a navigation menu with "Home", "Customers", "Employees", "Categories", "Customer Demographics", "Region", "Reports", and "Membership". The main content area is titled "Customers" and contains a list of customers. Below the list, there are tabs for "Orders", "Customer Demo", and "Order Details". The "Orders" tab is active, showing a list of orders.

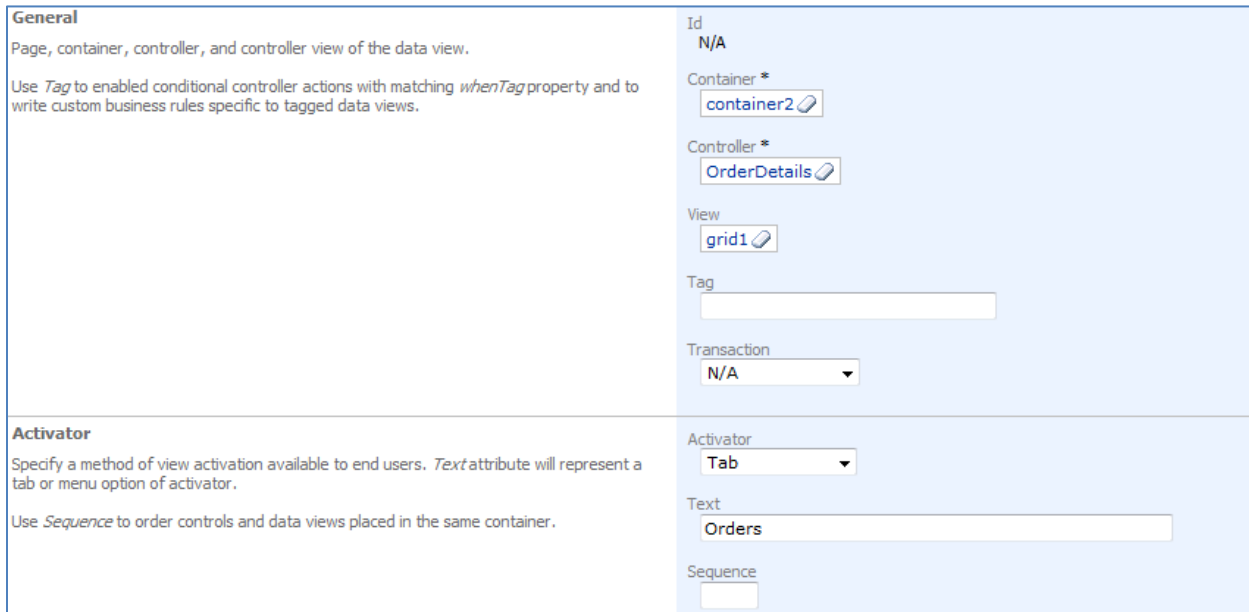
Customer #	Company Name	Contact Name	Contact Title	Address	City	Region	Postal Code	Country	Phone
ALFKJ	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	n/a	12209	Germany	030-0074321
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	n/a	05021	Mexico	(5) 555-4729
ANTON	Antonio Moreno Taquerías	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 Moore	Sales Representative	Frankestr. 57	Hannheim	n/a	68306	Germany	0621-08460
BLOPP	Blondies père et fils	Pierrick Citeaux	Marketing Manager	24, place Kléber	Strasbourg	n/a	67000	France	88.60.15.31
BOLID	Bólido Comidas preparadas	Martin Sommer	Owner	C/ Araquil, 67	Madrid	n/a	28023	Spain	(91) 355 22 82
BONAP	Bon app'	Laurence Labihan	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	TJF 8M4		Canada	(604) 555-4729

Employee Last Name	Order Date	Required Date	Shipped Date	Ship Via	Company Name	Freight	Ship Name	Ship Address	Ship City
Suyans	11/15/1996	12/13/1996	11/20/1996	Speedy Express	Around the Horn	\$41.95	Around the Horn	Brook Farm Stratford St. Mary	Colchester
Callahan	12/16/1996	1/13/1997	12/18/1996	Federal Shipping	Around the Horn	\$34.24	Around the Horn	Brook Farm Stratford St. Mary	Colchester
Davidi	2/21/1997	3/21/1997	2/26/1997	United Package	Around the Horn	\$25.36	Around the Horn	Brook Farm Stratford St. Mary	Colchester
Davidi	6/4/1997	7/2/1997	6/10/1997	United Package	Around the Horn	\$72.97	Around the Horn	Brook Farm Stratford St. Mary	Colchester
Peacock	10/16/1997	10/30/1997	10/23/1997	Federal Shipping	Around the Horn	\$21.74	Around the Horn	Brook Farm Stratford St. Mary	Colchester

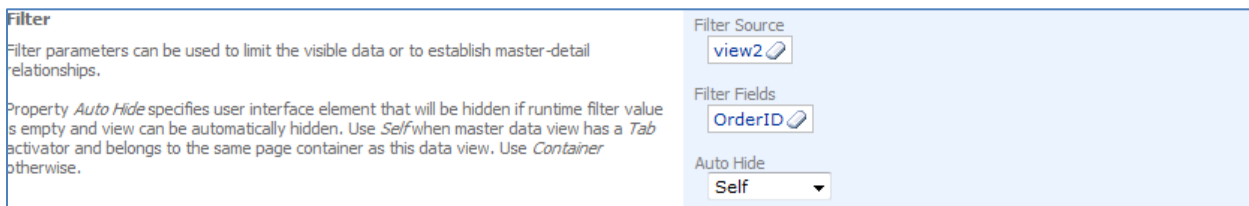
Open *Code On Time Generator*, select the project name, and press *Design*. Switch to the *All Pages* tab. Select the *Cusitomers* page from the list, and switch to *Data Views*. Using the dropdown menu next to *view4* (that holds *OrderDetails* controller), press *Delete*, as this view is not needed.



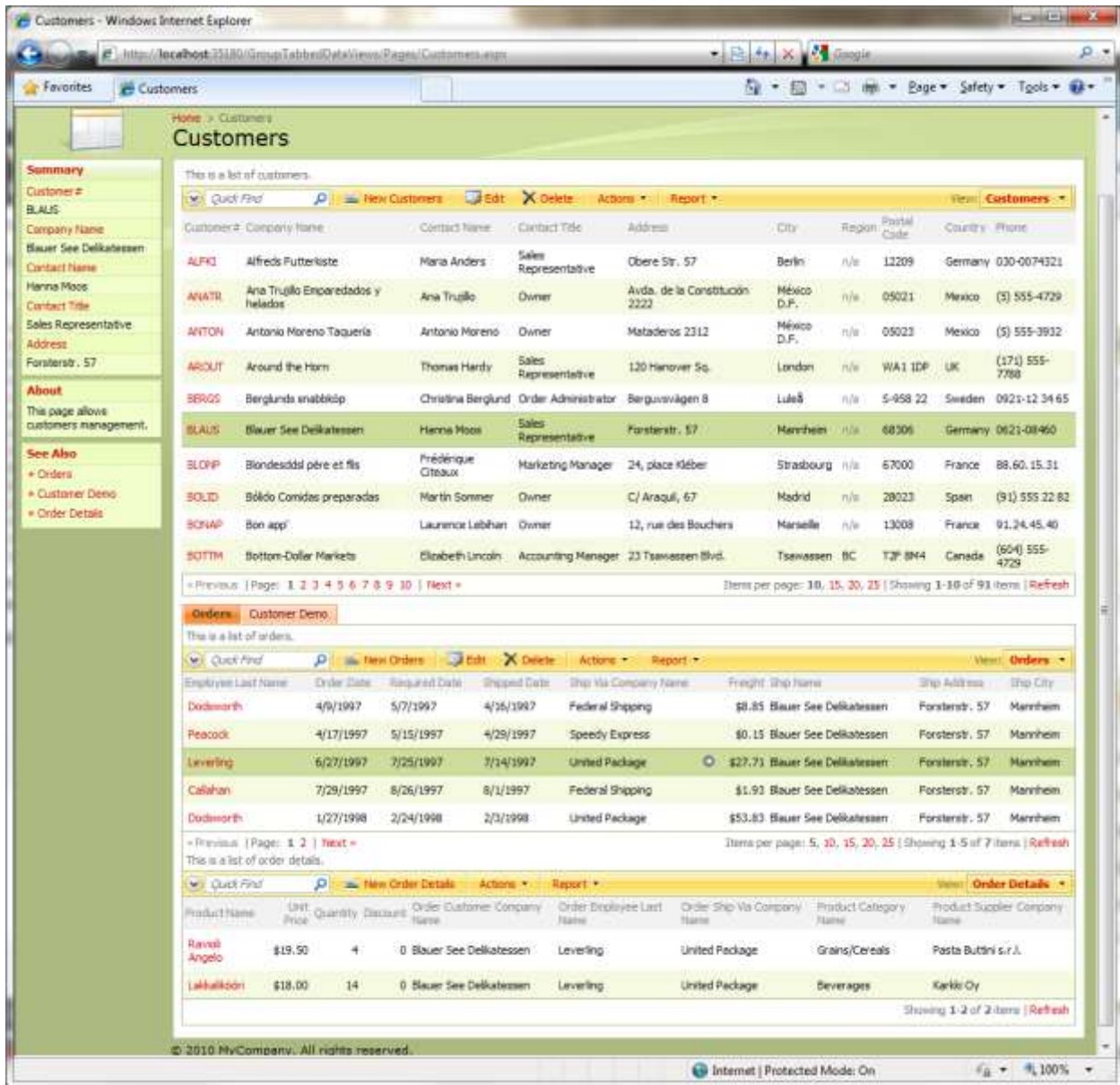
Now, using the action bar, press *New | New Data View*. The *Container* will be “*container2*”, *Controller* will be “*OrderDetails*”, and *View* “*grid1*”. Change *Activator* to “*Tab*”. The *Text* must be the same as the *Orders* tab to insure that both data views will be on the same tab, so write “*Orders*”.



The *Filter Source* will be “*view2*”, and *Filter Fields* will be “*OrderID*”. Indicate that *Auto Hide* is “*Self*”.



Close the *Designer*, and generate the application. When the web page appears with the modified application, navigate to the *Customers* tab. Now, when you select a *Customer*, tabs will appear underneath the *Customers* list displaying *Orders* and *Customer Demo*. When you select an order, its respective *Order Details* will appear underneath, within the tab.



Mixed Authentication

Code On Time web application generator supports ASP.NET Membership and several other authentication mechanisms. ASP.NET Membership is an attractive option for Internet applications and can be also successfully used in *intranet* applications deployed within network boundaries of an organization for use by a specific group of business users.

Web application administrator can use the advanced user manager provided with each generated application to create user accounts and manage roles.

Large organizations frequently mandate the need for a single sign-on mechanism to eliminate the need to manage multiple passwords and users accounts.

1. Typically a user name token is created and validated by the authentication software deployed to the local network. The user name token is embedded into each web request coming to a server. The authenticated user name can be found in a page request header variable.
2. Another option is to use the active directory identity name that can be available if Windows Authentication is enabled in your web application.

You can take advantage of either option to implement a mixed authentication based on the ASP.NET Membership option available in Code On Time database web application. Only the users registered in the ASP.NET Membership database of your application can access the application. User roles will also be derived from the membership database.

Users can self-register to use the application and will be able to access the application page when the user account is approved by administrator. Administrator can also create all authorized user accounts and assign the same “secret” password to all users.

Single sign-on is enabled through changes to the login user control. Open file `~/App_Code/Controls/Login.ascx` and modify the code-behind file as shown on the next page.

C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.Security;
using System.Security.Principal;

public partial class Controls_Login : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Mixed authentication sample
        if (!Page.User.Identity.IsAuthenticated)
        {
            string userName = null;
            // 1. read the identity from the page request header variable
            userName = Request.Headers["UserName"];
            // 2. read the identity from the identity of the current Windows user
            userName = WindowsIdentity.GetCurrent().Name;
            // simulate the user name and ignore methods (1) and (2)
            userName = "admin";
            if (!String.IsNullOrEmpty(userName))
            {
                MembershipUser user = Membership.GetUser(userName);
                if (user != null)
                    FormsAuthentication.RedirectFromLoginPage(user.UserName, false);
            }
        }
    }
}
```

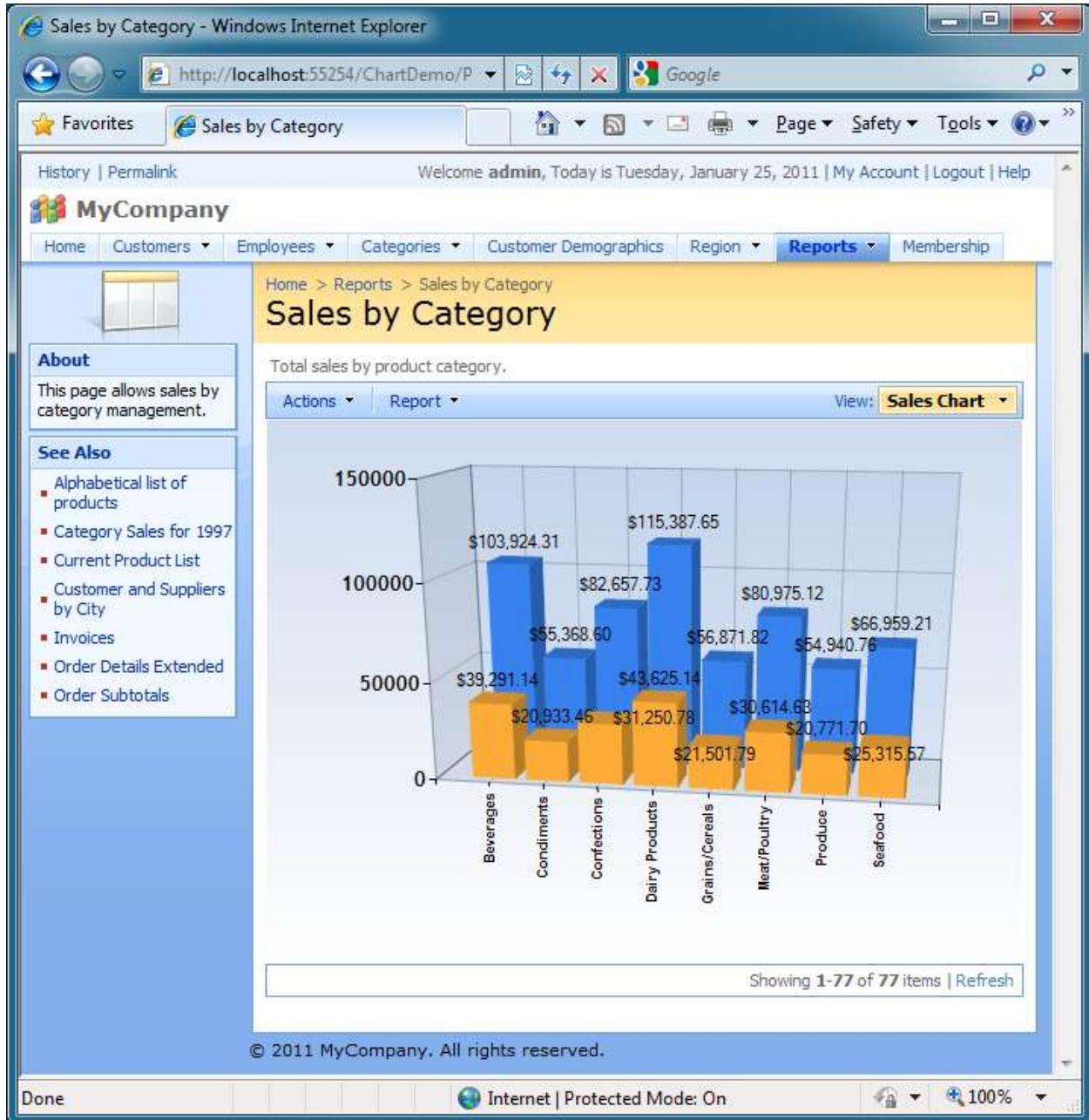
Methods of “silent” authentication are marked as (1) and (2). This particular example ignores the obtained information and simply assigns explicitly the user name “admin” to variable “userName”. Application makes a lookup request to identify the user as a valid ASP.NET Membership user. If that is the case then the user is automatically signed into the web application.

The login control is generated the first time only. Your changes to the code-behind file will stay intact with subsequent code generations.

You can adjust the sample to reflect your actual single sign-on method.

Chart View

Code On Time applications now offer a new type of data rendering – “Chart” view. Chart view is just another way of presenting a set of data records retrieved from the database. Chart view supports many end-user features including sorting and adaptive filtering.



Creating a Chart View

Generate a new web application from the Northwind database. Browse the generated web site and select *Reports | Sales by Category* menu option. The following data view will be displayed.

Columns *CategoryID*, *Category Name*, *Product Name*, and *Product Sales* are visible in the grid view.

The screenshot shows a web browser window displaying a report titled "Sales by Category". The report content is as follows:

Category #	Category Name	Product Name	Product Sales
1	Beverages	Chai	\$4,887.00
1	Beverages	Chang	\$7,038.55
1	Beverages	Chartreuse	\$4,475.70
1	Beverages	Côte de Blay	\$49,198.09
1	Beverages	Guaraná Far	\$1,630.13
1	Beverages	Ipoh Coffee	\$11,069.90
1	Beverages	Lakkalikööri	\$7,379.10
1	Beverages	Laughing Lu	\$5,468.40
1	Beverages	Outback Lag	\$4,485.54
1	Beverages	Rhönbräu Kl	

The data controller is based on the database view *dbo.[Sales by Category]*. This view is a part of the Northwind database and is defined as follows.

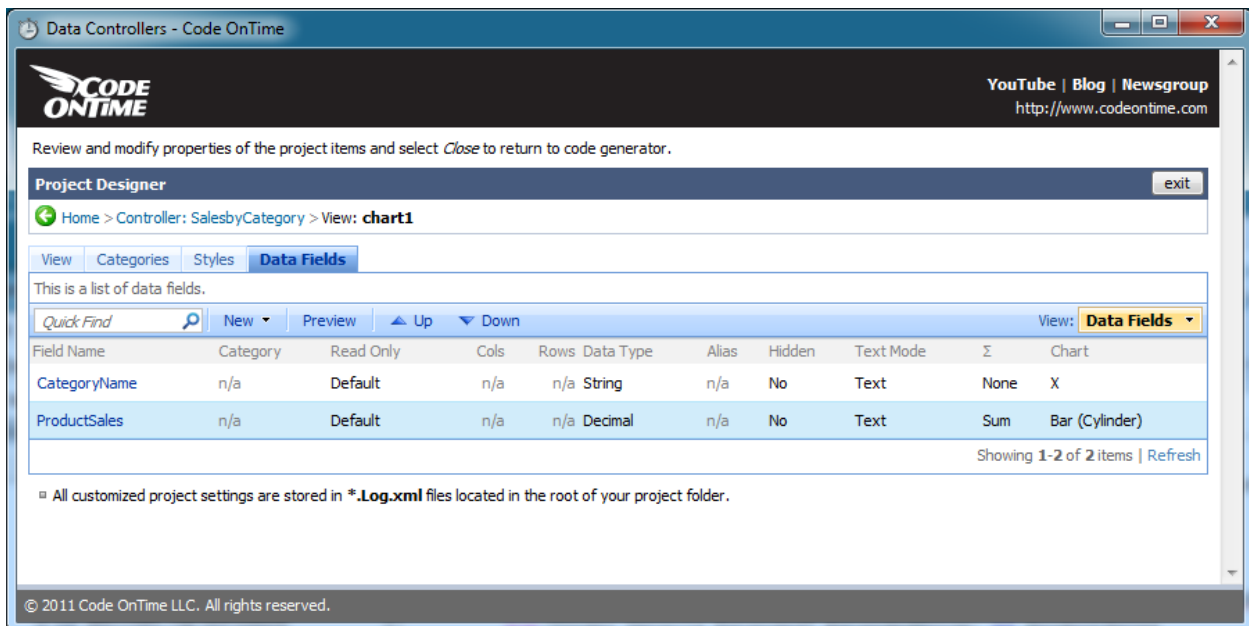
```
create view [dbo].[Sales by Category] AS
SELECT Categories.CategoryID, Categories.CategoryName, Products.ProductName,
       Sum("Order Details Extended".ExtendedPrice) AS ProductSales
FROM   Categories INNER JOIN
       (Products INNER JOIN
       (Orders INNER JOIN "Order Details Extended" ON
       Orders.OrderID = "Order Details Extended".OrderID)
       ON Products.ProductID = "Order Details Extended".ProductID)
       ON Categories.CategoryID = Products.CategoryID
WHERE  Orders.OrderDate BETWEEN '19970101' And '19971231'
GROUP BY Categories.CategoryID, Categories.CategoryName, Products.ProductName
```

Start the code generator, select the project name, and click *Design* button. Select the data controller *SalesbyCategory* and click on *Views* tab.

Add a new view, set its *Id* to *chart1*, select *Chart* as view type, and select *command1* as command. Set label to *Sales Chart*. Enter “*Total sales by product category.*” in the header text.

Save the view and click on its name in the list of available data controller views, select *Data Fields* tab. Add new data field with the field name set to *CategoryName*. Set its *Chart* property under *Miscellaneous* section to *X*. Save the field.

Add another data field with the field name set to *ProductSales*. Enter letter “*c*” without double quotes into *Data Format String*. Set the *Aggregate Function* property of the data field to *Sum*. Set its *Chart* property to *Bar (Cylinder)*. The list of data views in *Designer* will look as follows.



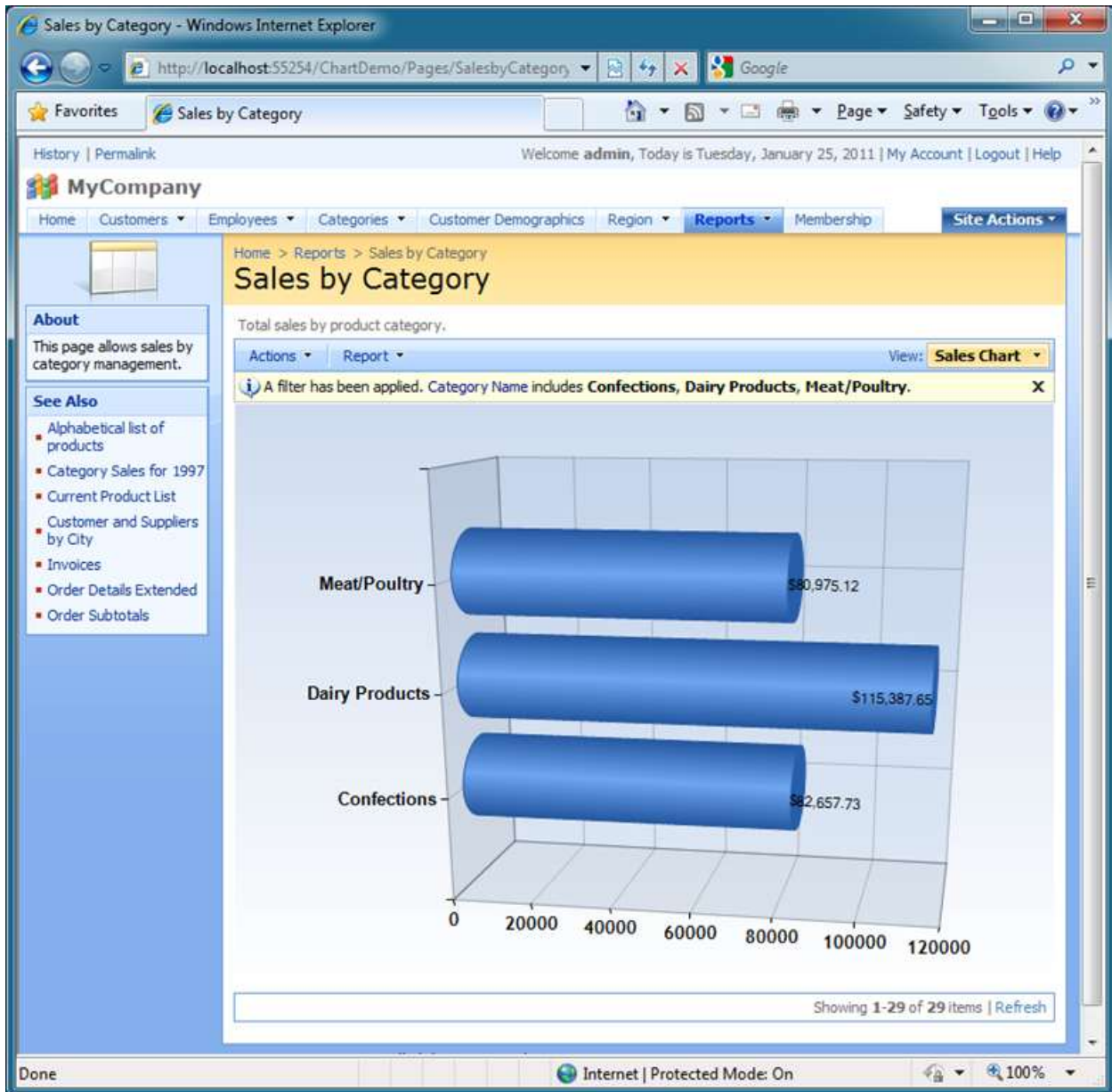
Exit the *Designer* and generate your application. Activate the same page and select *Sales Chart* option in the view selector in the right hand corner of the action bar. The following chart on the next page will be displayed.



Activate the filter in the view selector and select “Filter...” item in the popup menu of the *Category Name* option.



Select several filter options to review subset of data presented in the chart.



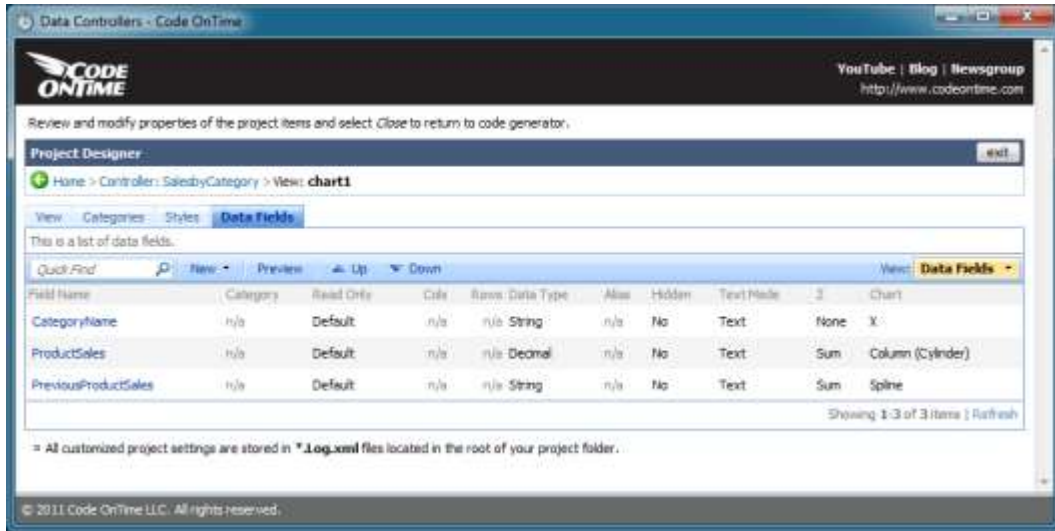
Displaying Multiple Values

The chart view is capable of displaying multiple data series. Let's add a calculated field to the same data controller to simulate the "Previous Product Sales". Select the data controller in *Designer* and activate *Fields* tab. Add a new field with name *PreviousProductSales*, indicate that the field value is calculated by SQL formula and enter the following SQL formula into *SQL Formula* text box:

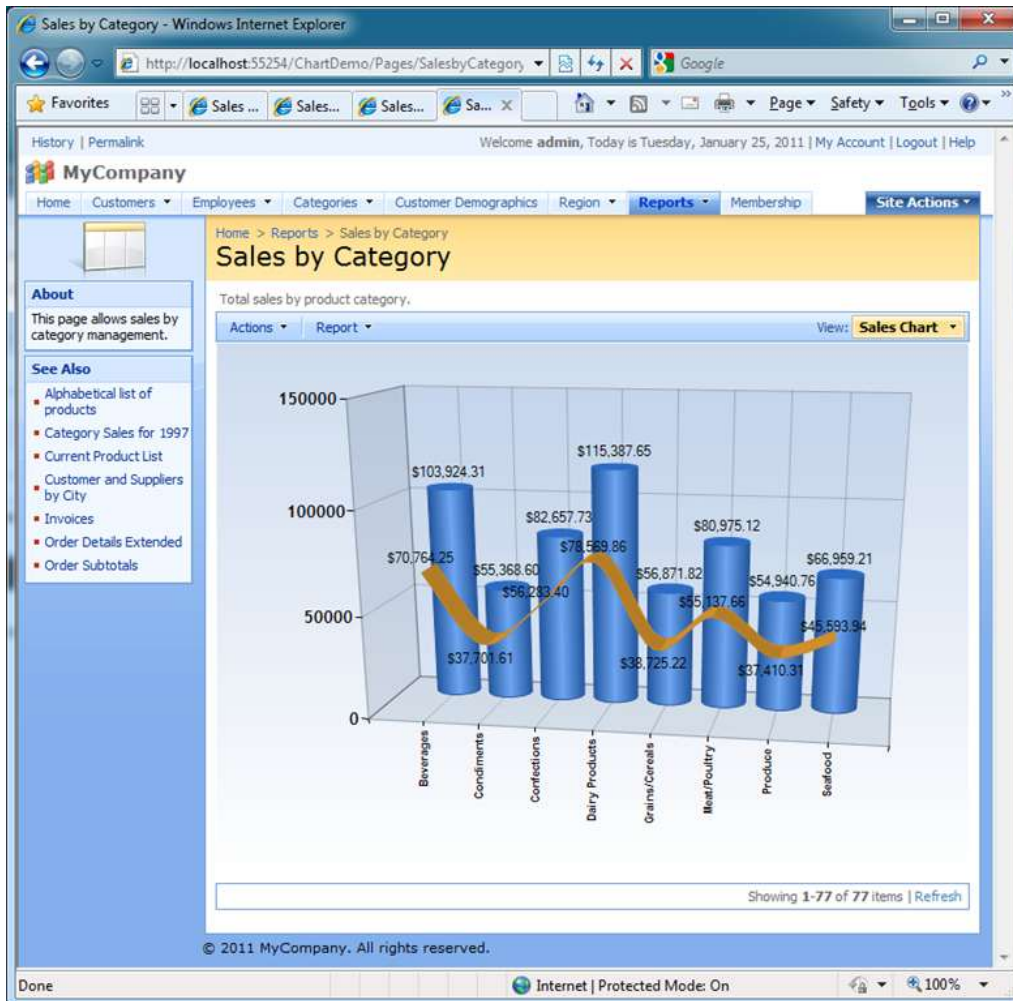
```
cast(ProductSales * Rand() as Numeric(10,2))
```

Set the label of the field to "Previous Product Sales". Set its *Data Format String* to "c" without quotes. Save the field and select *Views* tab. Select *chart1* in the list of available views. Bind the new field to

the *chart1* view and set its properties to make them look as shown in the screenshot. Notice that we are using a different *Chart* type *Column(Cylinder)* for *ProductSales*.



Run the generated application. The following chart view will be presented if you activate *Sales Chart* in the view selector. The actual spline that you will see may look different due to randomization factor of the formula that we have specified to simulate the previous sales.



Legend

You can activate a legend if you select the chart view in *Designer* and mark the check box “*Enable legend in the chart area*”. The data field header will be used as the text displayed in the chart legend.



Custom Charts

Chart views are based on the standard Microsoft Data Visualization component included with ASP.NET 4.0. Unlimited customization options are available to developers. You can quickly customize a chart view if you select “*Custom*” as *Chart*property of the data field.

All charts are generated as ASP.NET user controls stored in `~/Controls` folder of your web application. For example, the name of the chart in this sample is `~/Controls/Chart_SalesbyCategory_chart1.ascx`. The

name of a chart user control always starts with *Chart* and includes the name of the data controller and the chart view ID.

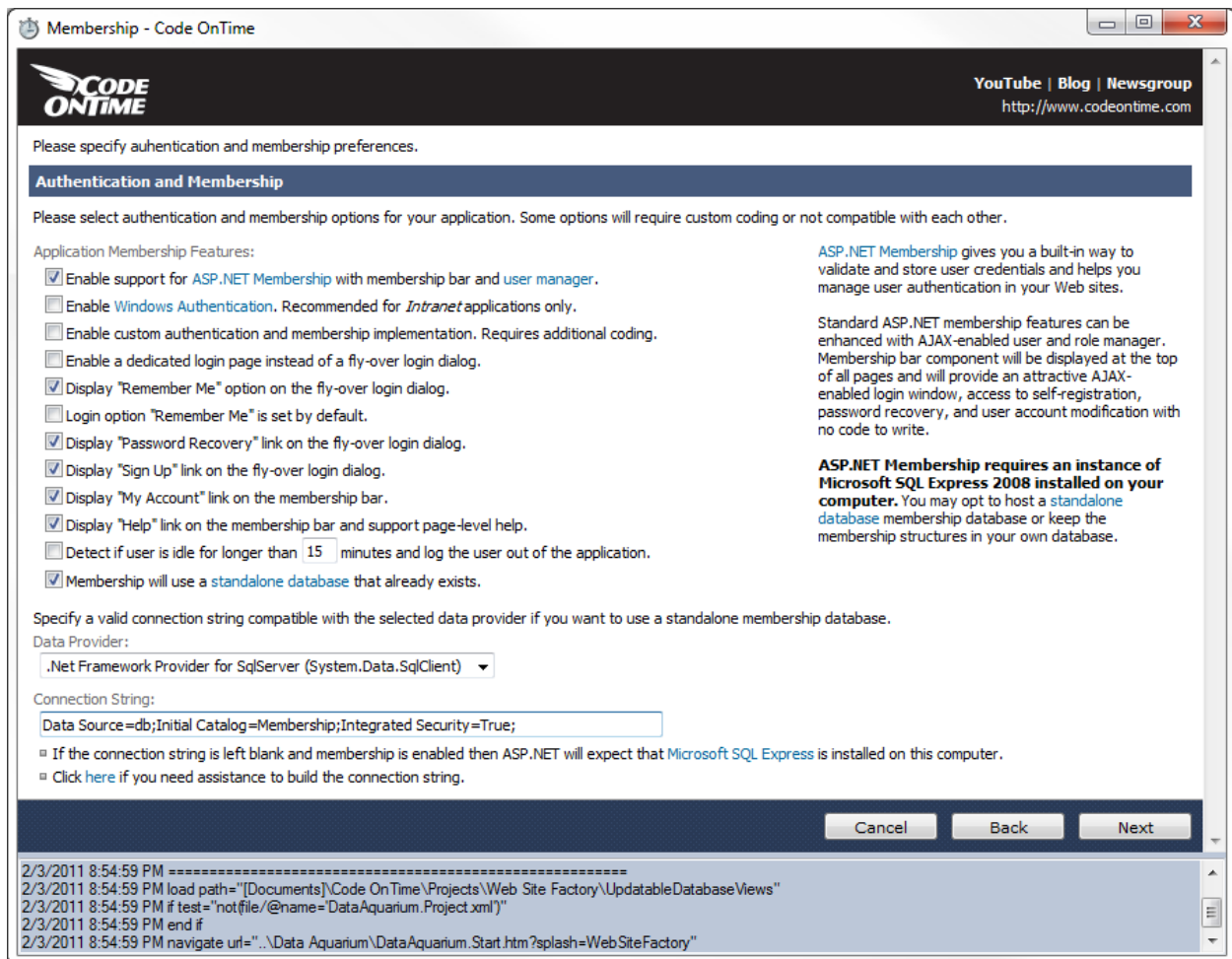
“Custom” charts are generated once only. If a “Custom” chart exists then the code generator will not make an attempt to generate the chart again. You can safely modified hundreds of the chart control properties.

Standalone ASP.NET Membership Database

Web Site Factory and other premium projects integrate *ASP.NET Membership*, a built-in way to store and validate user credentials. You can enable *ASP.NET Membership* by selecting the membership option in the code generator project wizard. This will enable numerous membership features including a fly-over sign-in window, self-service membership enrollment, membership bar, and membership manager.

The configuration of your project will be automatically changed to support the default membership provider available in ASP.NET. This provider defines a connection string that points to a local instance of *Microsoft SQL Server Express*. The provider will automatically connect to the server and dynamically create a database to maintain users, roles, and other membership features. The database will be created under `~/App_Data` folder of your project.

This works great on a development machine with installed *SQL Server Express*. There are many situations when you want to use a standalone membership database or store ASP.NET membership data structures directly in your own database.



Project wizard offers an option that will enable a standalone membership database configuration. Here is the screen shot of the project wizard with the standalone membership database enabled.

The connection string in the screen shot looks as follows:

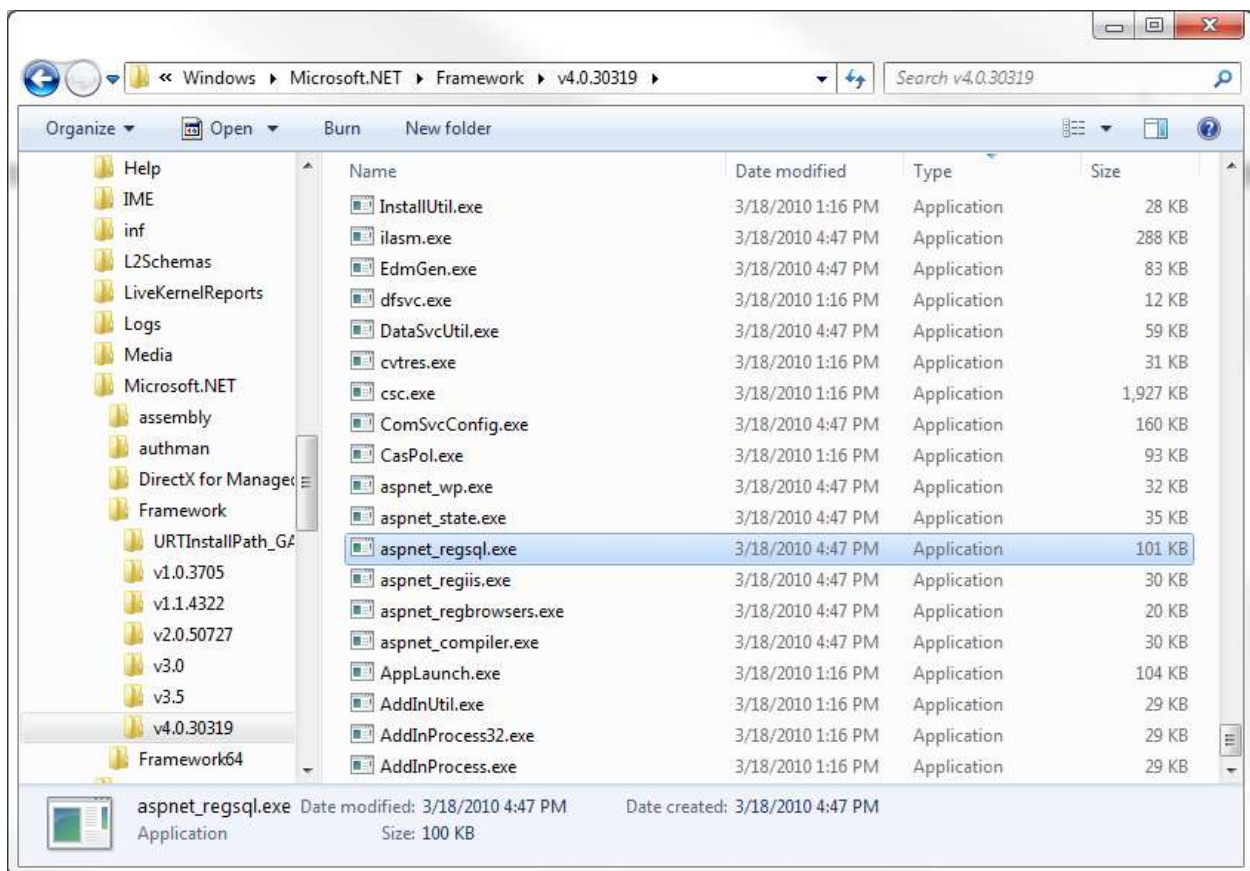
```
Data Source=db;Initial Catalog=Membership;Integrated Security=True;
```

We have configured the standalone membership database with the name *aspnetdb*.

You can read more about the configuration process at [http://msdn.microsoft.com/en-us/library/ms229862\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/ms229862(VS.80).aspx).

These are the steps that we have taken to create the *aspnetdb* database:

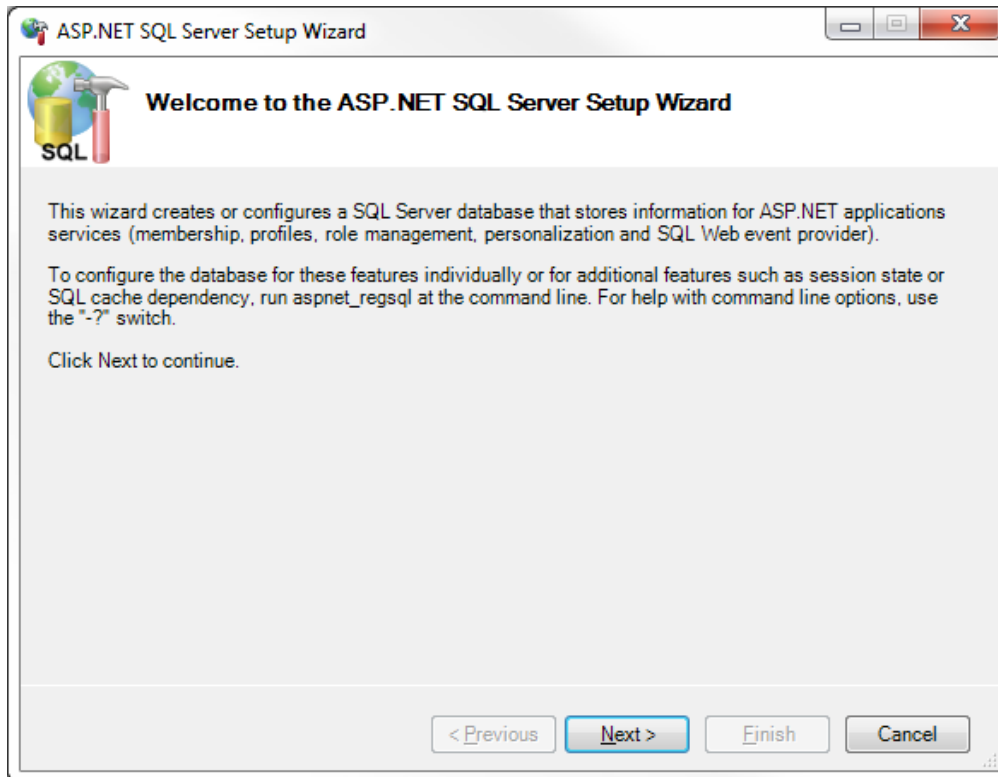
1. We have started *aspnet_regsql.exe* from *Windows Explorer* as shown in picture.



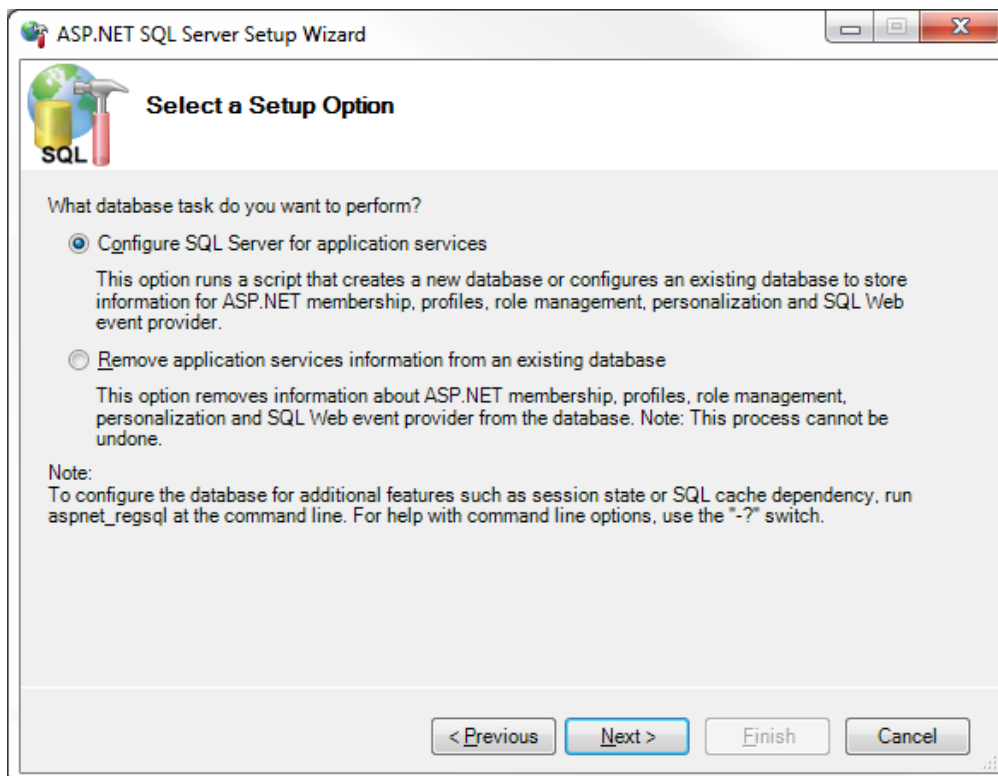
The path to your instance of *aspnet_regsql.exe*:

```
C:\%windir%\Microsoft.NET\Framework\<versionNumber>\aspnet_regsql.exe
```

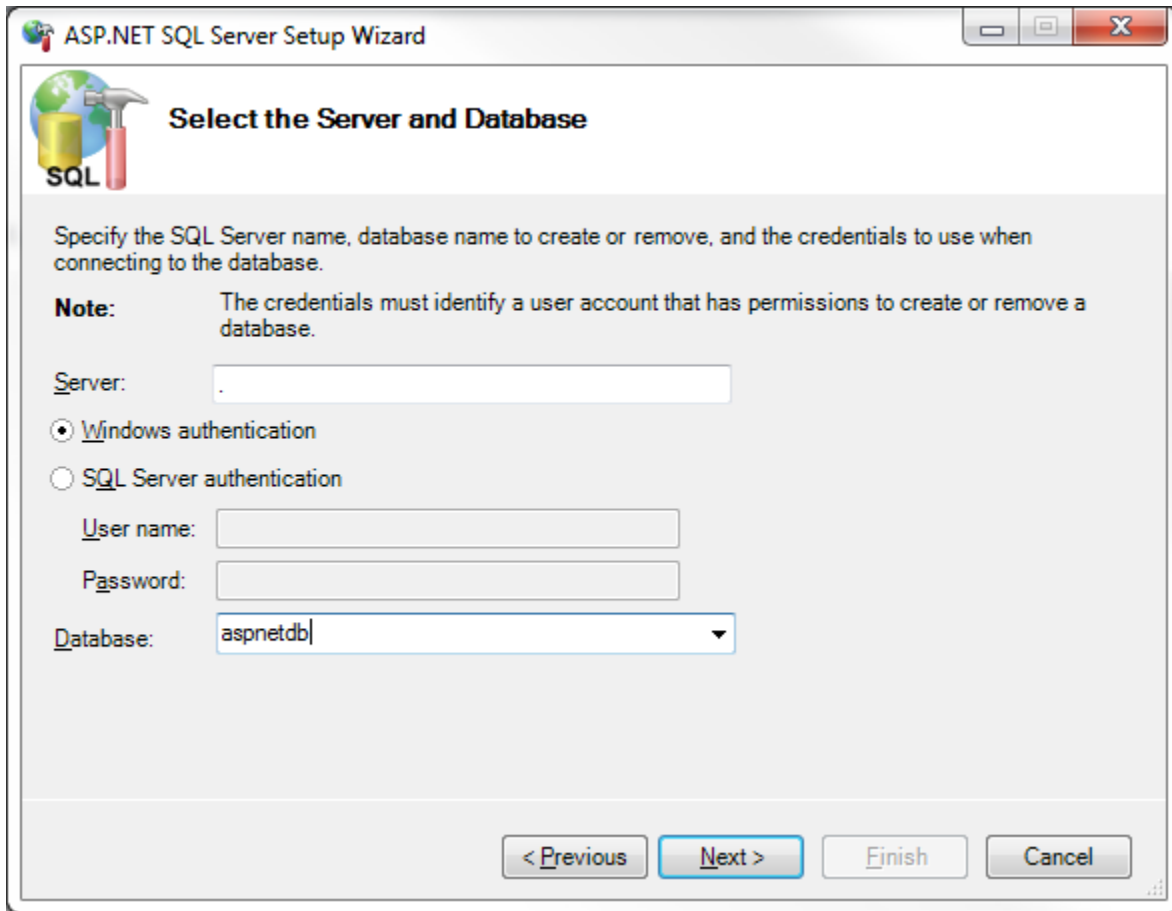
2. We have clicked *Next* button in *ASP.NET SQL Server Setup Wizard*:



3. We have continued to the next step to configure *SQL Server* for application services:



4. We have entered "." as a server name and "aspnetdb" as database name.



A few more clicks on the *Next* button have done the job for us. The database has been created. We have returned to the project wizard of our code generation project and configure the provider name and connection string of the newly created membership database.

Learn More

Visit our website at: <http://codeontime.com/>

More Tutorials: <http://codeontime.com/Tutorials.aspx>

Blog: <http://blog.codeontime.com/>

YouTube Channel: <http://youtube.com/user/codeontime>

Download Code On Time Generator at <http://codeontime.com/Download.aspx>

Get a subscription at <http://codeontime.com/Subscriptions.aspx>.

