

2011



USER GUIDE

Cascading Lookups

Cascading Lookups

ASP.NET web applications created with *Code On Time Generator* offer impressive lookup capabilities. Numerous lookup styles allow configuring sophisticated data lookup scenarios, such as cascading lookups. Cascading lookups occur when there is a data lookup scenario involving more than one lookup field.

Example of Cascading Lookups

Consider the fragment of *Adventure Works LT* database presented below.

Table *SalesOrderHeader* has three lookup fields. Field *CustomerID* references table *Customer*, while fields *ShipToAddressID* and *BillToAddressID* are referencing the table *Address*. Shipping and billing addresses may be represented by the same or different rows in the table *Address*. The many-to-many table *CustomerAddress* links a *Customer* and an *Address*.

If a new *Sales Order Header* is created and a customer is identified and associated with that header, then it will be logical to assist end users of your application in selecting a customer address by limiting the list of all addresses to those that are matching the selected customer. Selection of one lookup value (*CustomerID*) in a row must cascade as a filter to other dependent lookups (*ShipToAddressID* and *BillToAddressID*).



Configuring Cascading Lookups

If you generate a new *Web Site Factory* application from the database and start creating a new *Sales Order Header* then you will see a page similar to the one presented below. The screen shot shows that a customer has been selected, but the shipping and billing addresses are blank.

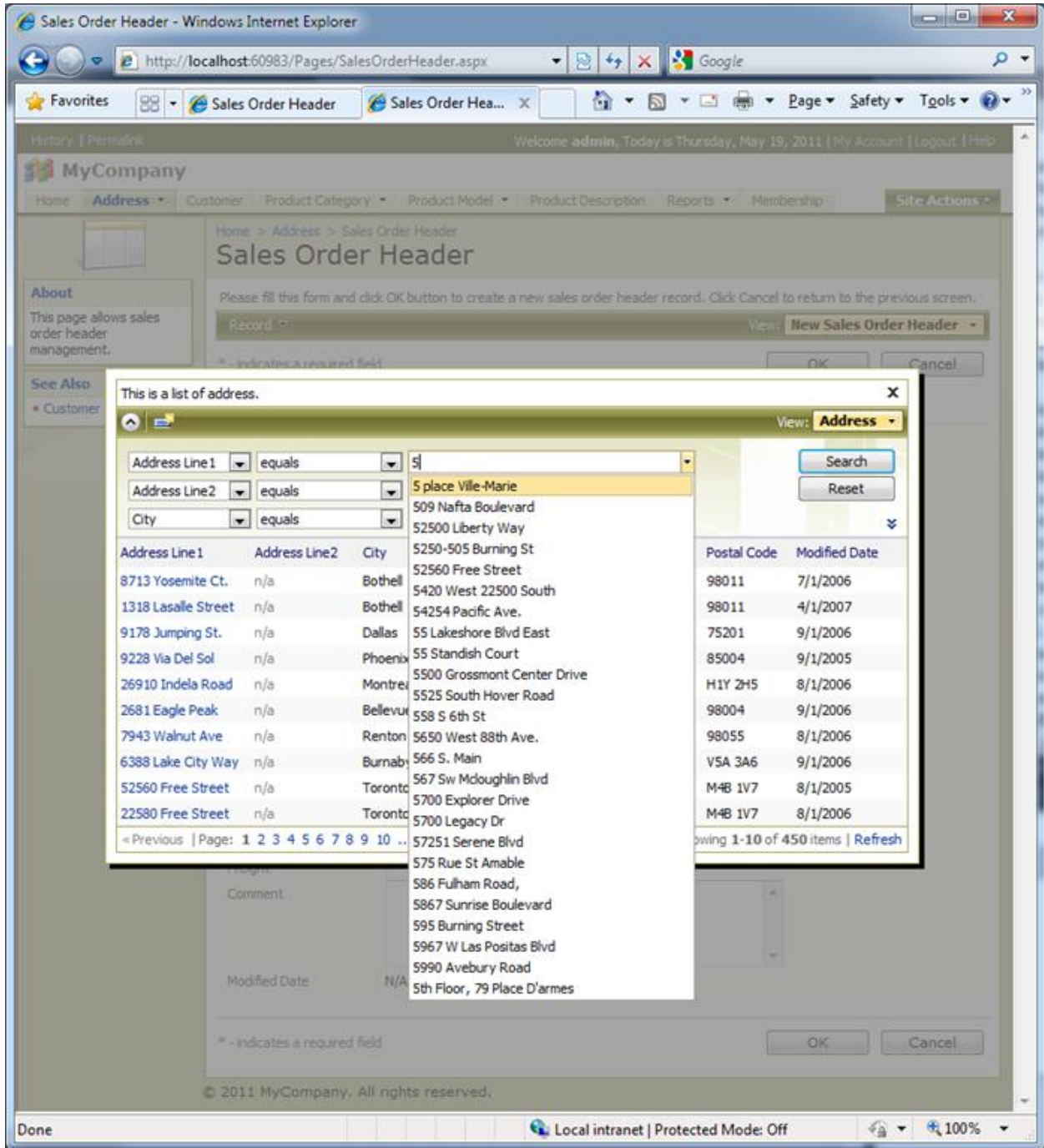
The screenshot shows a web browser window titled "Sales Order Header - Windows Internet Explorer" displaying a form for creating a new sales order header. The browser address bar shows the URL "http://localhost:60983/Pages/SalesOrderHeader.aspx". The page header includes "Welcome admin, Today is Thursday, May 19, 2011 | My Account | Logout | Help". The main content area is titled "Sales Order Header" and contains a form with the following fields:

- Revision Number *
- Order Date *
- Due Date *
- Ship Date
- Status *
- Online Order Flag * (select)
- Purchase Order Number
- Account Number
- Customer Company Name * (Bicycle Accessories and Kits)
- Ship To Address Line 1 (select)
- Bill To Address Line 1 (select)
- Ship Method *
- Credit Card Approval Code
- Sub Total *
- Tax Amt *
- Freight *
- Comment
- Modified Date: N/A

The form includes "OK" and "Cancel" buttons at the bottom right. A note at the bottom left states "© 2011 MyCompany. All rights reserved." The browser status bar at the bottom indicates "Local intranet | Protected Mode: Off" and "100%".

Click on *(select)* link in *Ship To Address Line1* or *Bill To Address Line1*. You will be presented with lookup window that allows address selection. All addresses stored in the database are available for selection.

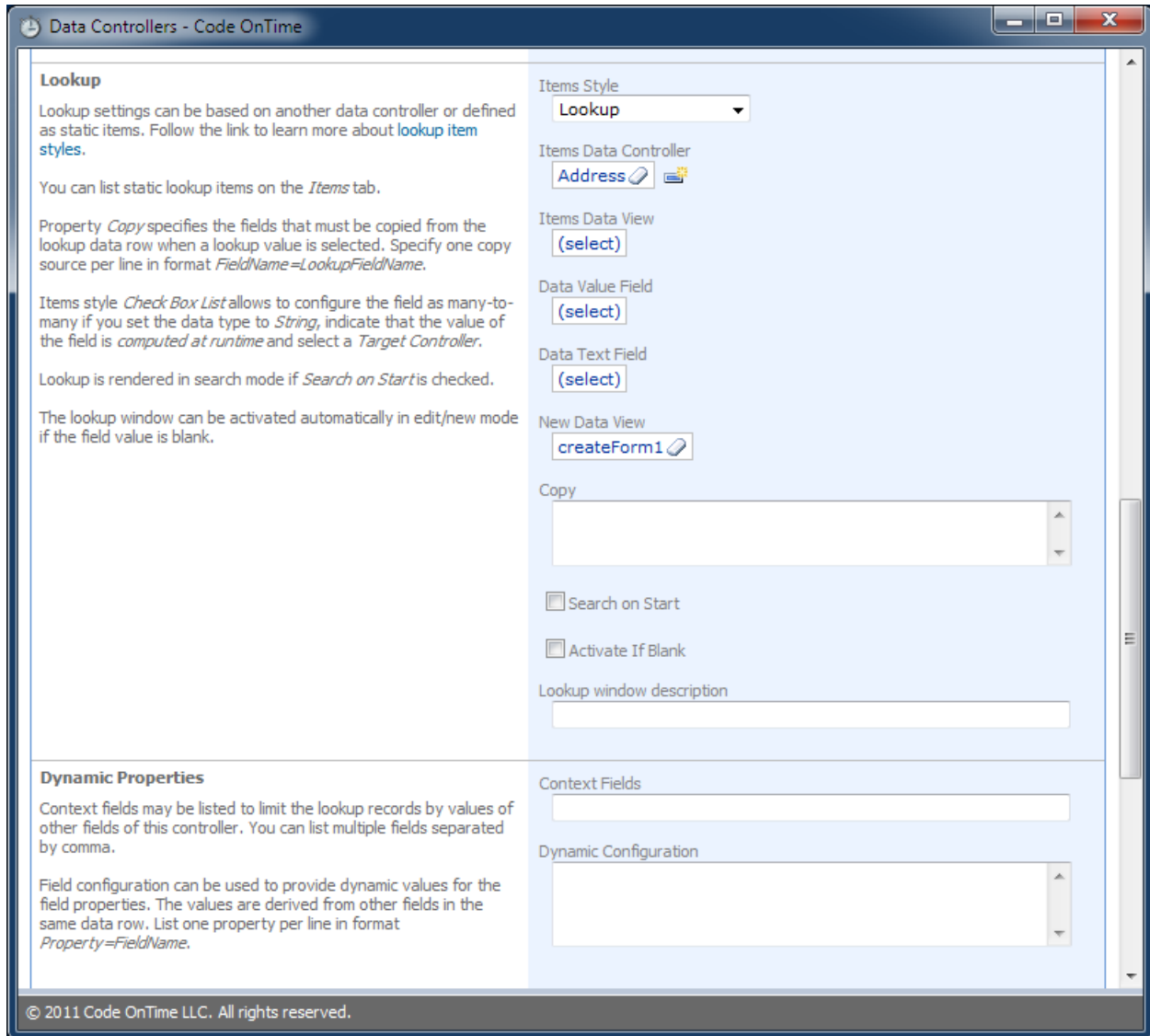
The screen shot shows that the application user has activated the search bar to find the right address and is trying to find an address using predictive input.



Let's make the task of selecting an address easier and limit the list of addresses to those that are associated with the user account.

First we will examine lookup configuration of *ShipToAddressID* and *BillToAddressID* fields.

Run the web application generator, select the project name and click *Design* to activate the project *Designer*. Select *All Controllers* tab and locate *SalesOrderHeader* data controller. Click on the controller name and select *Fields* tab. Select the *ShipToAddressID* field. Scroll to the *Lookup* section of the field properties. Change Items Data Controller to “Address”. This will instruct the generated web application to show the first grid view of *Address* data controller when a user clicks on the (*select*) link of the lookup field.



If you inspect configuration of *BillToAddressID* field then you will find that it is exactly the same. This type of configuration is automatically performed by *Code On Time* web application generator when the baseline application is constructed.

While the table *Address* does not offer any information that will help us to identify the relevant addresses, table *CustomerAddress* has a pointer to *Customer* and provides a perfect replacement for *Address* data controller.

By default only the first mandatory field from master table *Address* is included by the application generator in the data controller *CustomerAddress*. We suggest defining the following de-normalization map for your project to expand available fields borrowed from master tables in tables *CustomerAddress* and *SalesOrderHeader*.

```
SalesLT.SalesOrderHeader => SalesLT.Customer  
LastName  
FirstName  
CompanyName
```

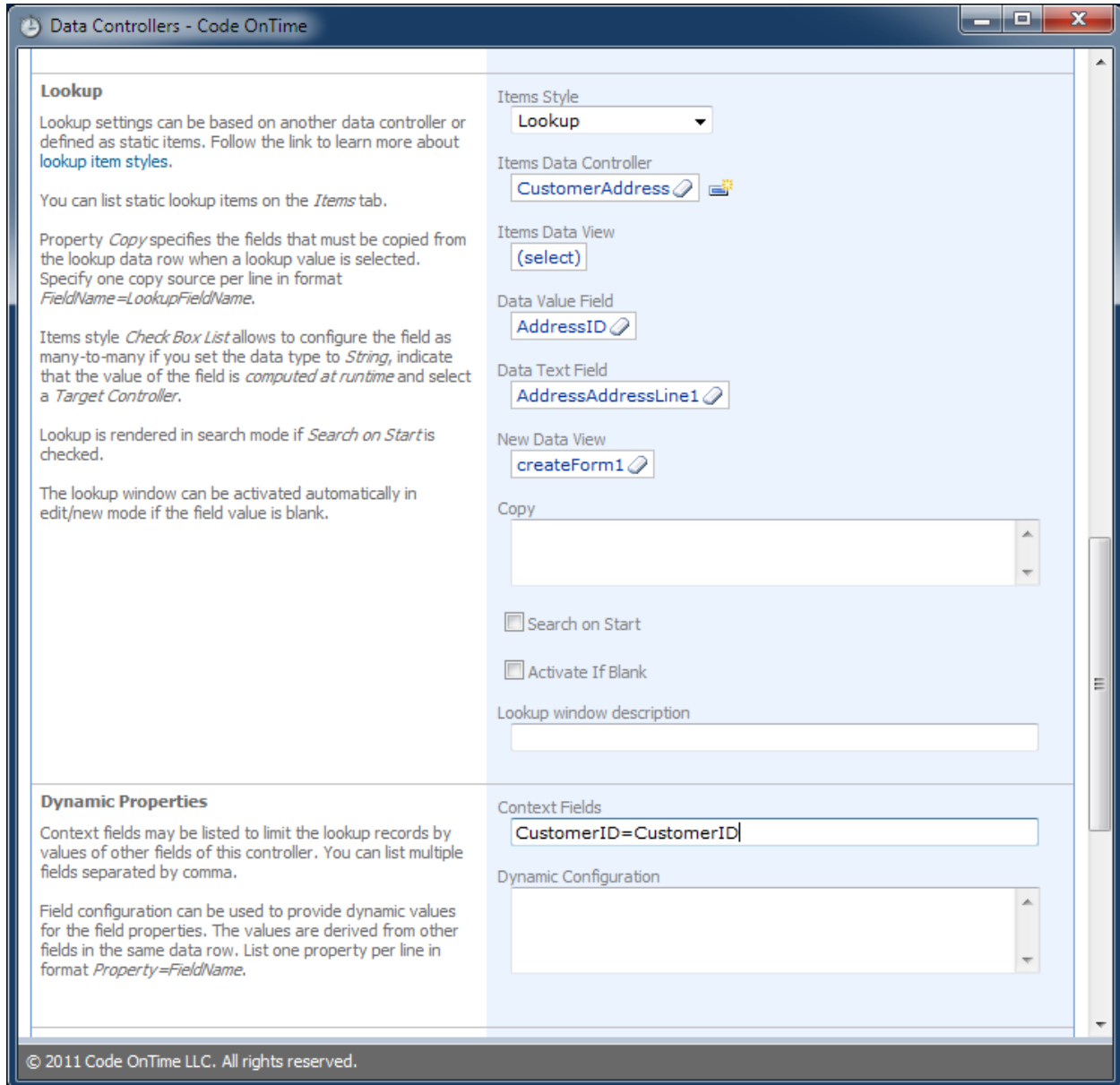
```
SalesLT.CustomerAddress => SalesLT.Customer  
CompanyName  
LastName  
FirstName
```

```
SalesLT.CustomerAddress => SalesLT.Address  
AddressLine1  
AddressLine2  
City  
StateProvince  
CountryRegion  
PostalCode
```

The application shown in the screenshots was generated with this de-normalization map. For more information on de-normalization maps, read this article: <http://codeontime.com/Documents/UGP1-BLL-P2.pdf>.

Let's change the configuration of field *ShipToAddressID* as follows:

1. Set *Data Controller* property to "CustomerAddress".
2. Set *Data Value Field* property to "AddressID".
3. Set *Data Text Field* property to "AddressAddressLine1".
4. Set *Content Fields* under *Dynamic Properties* section to "CustomerID=CustomerID".



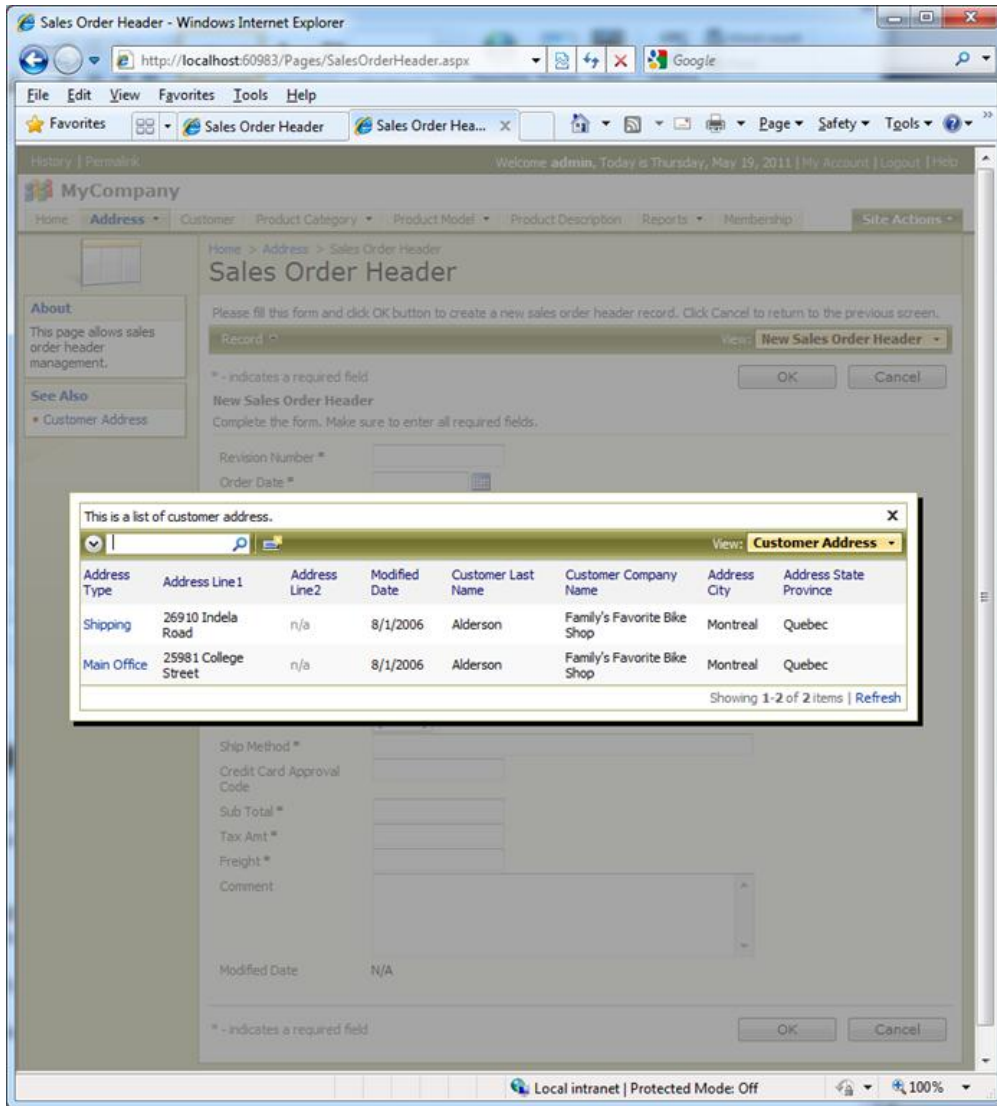
The text inserted in *Context Fields* will instruct the data controller *CustomerAddress* to filter data by field *CustomerID* (field on the left hand side of the equal sign) with the value stored in field *CustomerID* (field on right hand side of the equal sign) in the current row of *Sales Order Header* data controller.

This change does not affect the command text of the data controller *SalesOrderHeader* in any practical way. The lookup configuration will be taken into account only when a user selects a value for *ShipToAddressID* or *BillToAddressID* field.

Save the changes and modify the field *BillToAddressID* with the same settings as before.

Once completed, close the *Designer* and regenerate your application.

When the web page loads, navigate to the *Sales Order Header* page. Start creating a new *Sales Order Header* record. For *Customer Company Name* field, select "Family's Favorite Bike Shop". Try to select shipping or billing address by using the (*select*) link. Only two records will be available for selection. Both records match the selected customer.





Try changing a customer and observe that a different selection of addresses is presented each time.

Lookup Styles

You can change the style of lookups using the *Items Style* property. If you change the style of *CustomerID*, *ShipToAddressID*, and *BillToAddressID* then the cascading lookup behavior will still take place.

Lookup field *CustomerID* is presented as Lookup and the two address reference fields are presented as *Auto-Complete* below.

Account Number	<input type="text"/>
Customer Company Name *	Family's Favorite Bike Shop  
Ship To Address Line 1	25981 College Street
Bill To Address Line 1	<input type="text"/>
Ship Method *	N/A
Credit Card Approval Code	25981 College Street 26910 Indela Road
Sub Total *	\$320.60
Tax Amt *	\$0.00

A combination of *Auto-Complete*, *List Box* and *Radio Button List* lookup styles is shown next.

Account Number	<input type="text"/>
Customer Company Name *	Family's Favorite Bike Shop
Ship To Address Line 1	<input type="text" value="N/A"/> 25981 College Street 26910 Indela Road
Bill To Address Line 1	<input type="radio"/> N/A <input type="radio"/> 25981 College Street <input checked="" type="radio"/> 26910 Indela Road
Ship Method *	Mail
Credit Card Approval Code	<input type="text"/>
Sub Total *	\$320.60
Tax Amt *	\$0.00

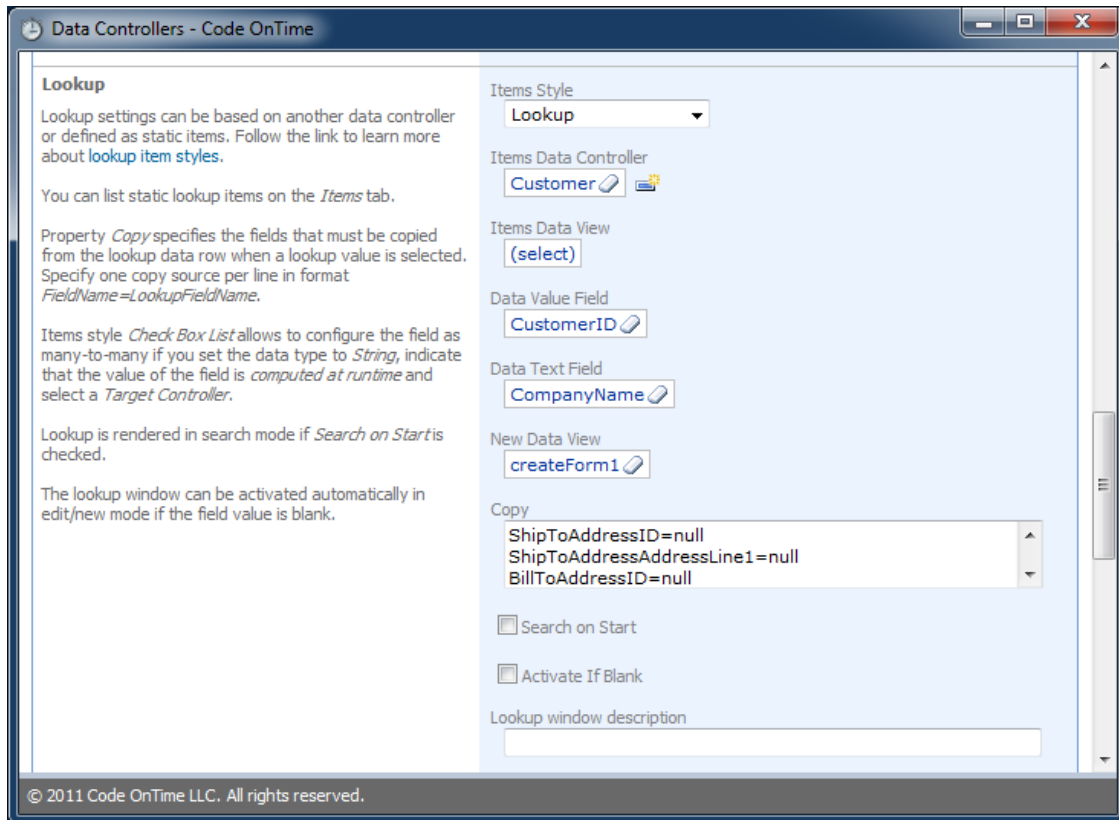
Clearing Dependent Lookup Fields

Most scenarios of cascading lookups will require a particular combination of master and detail selections in the lookup fields. If a master value has changed then a cascading clearing of dependent lookup fields may be called for. In fact, inconsistent selection of master and detail lookup fields may cause data integrity issues.

An extra step is involved in clearing dependent cascading lookup field values.

Select *CustomerID* master field in Designer and set its “Copy” property as follows:

```
ShipToAddressID=null  
ShipToAddressAddressLine1=null  
BillToAddressID=null  
BillToAddressAddressLine1=null
```



The primary purpose of “Copy” property is to allow specifying multi-field copy instructions executed upon lookup selection. Read about multi-field lookup at <http://blog.codeontime.com/2010/02/multi-field-lookup.html>.

If you enter “null” on the right-hand side of the equal sign then the field value will be cleared when the value of the lookup field has changed. In this particular case any changes to *CustomerID* field must cause clearing of fields *ShipToAddressID* and *BillToAddressID*. Both fields are aliased with *ShipToAddressAddressLine1* and *BillToAddressAddressLine1* fields accordingly. If you configure only the foreign key fields to be reset upon changes of the master field *CustomerID* then fields will get cleared but their visual representations will remain intact. Therefore we require clearing of alias fields as well.

Property “Copy” provides surgical level of control over resetting of dependent fields in a cascading lookup field groups. If you have more than one level of dependency then make sure to configure clearing of fields on all levels.