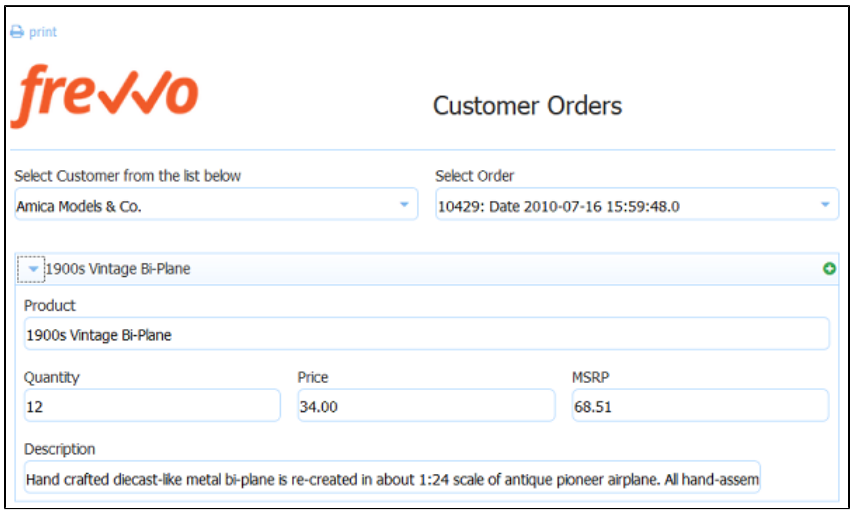
DB Connector Tutorial

> The database connector maps between HTTP requests issued from Live Forms forms to SQL queries executed against your database. The connector enables you to use the database of your choice to save and load your form data. You can create a new database or you can automatically generate forms that connect to your existing database tables.



To successfully complete this tutorial, you will need:

- An installed database ex: MySQL
- Live Forms configured to work with your database
- To install and configure the database connector
- To Install the sample database

You can download the completed application and then upload it to your account. This will provide a reference for the example forms described below.

> There are additional resources designed to help you learn how to use the power of Live Forms forms with your database. Refer to Database Connector Examples for more information.

On this page:

What are we going to build ?

We will first implement a simple master-detail application to view orders for a customer and line item details for any order in a single form. This demonstrates how to populate a web form with data from the database.

- When the form loads, we will initialize a drop down with the list of customers from the database.
- Select a customer and the drop down below it populates with the orders for that customer.
- Select an order and order details will appear below.

Later, we will also write data back to the database creating new records and updating existing ones. This tutorial does not cover advanced areas such as autocreate, autodelete, date/timestamp considerations. See the documentation for more detailed coverage of these topics.

Sample Database

We'll use the Eclipse BIRT sample database that allows you to learn the database connector using a well-known database before moving on to your own real-world data sources. You can download the sample database from the following link: http://www.frevvo.com/bucket/tutorial/dbconnector/birt-database-2_0_1.zip. MySQL and Microsoft Access databases are included with the download. The ER-Diagram shows the table structure and relationships.

To install the sample databases, see the Installing the Sample Database instructions.

Example Forms

You can download the DBConnectorTutorial_app.zip containing all of the example forms used in this tutorial. This download is a Live Forms application zip file that you can upload into your user account. See Application Home Page documentation for more information on uploading this application zip file.

Database Connector Configuration

If you have not installed the database connector please visit Connecting to your Database for instructions.

To use the database connector, you must first configure it so that it can find your database and connect to it. Here are examples for MySQL and MS Access.

```
<queryset name="BIRT" dateFormat="yyyy-MM-dd" xmlDateFormat="yyyy-MM-dd"
xmlTimeStampFormat="MM/dd/yyyy">
        <resource-def>
            <url>jdbc:mysql://localhost/ClassicModels</url>
            <driver>com.mysql.jdbc.Driver</driver>
            <user>root</user>
            <password></password>
        </resource-def>
        <query name="allCustomers">
            <retrieve>
                <!-- maps to HTTP GET -->
                <statement> SELECT customerNumber,customerName from Customers order by
customerName </statement>
            </retrieve>
        </query>
        <query name="customerByNumber">
            <retrieve>
                <!-- maps to HTTP GET -->
                <statement> SELECT * from Customers where customerNumber={cnum}
</statement>
            </retrieve>
        </query>
 </queryset>
```

```
<queryset name="BIRT" dateFormat="yyyy-MM-dd" xmlDateFormat="yyyy-MM-dd"
xmlTimeStampFormat="MM/dd/yyyy">
        <resource-def>
            <url>jdbc:odbc:ClassicModels</url>
            <driver>sun.jdbc.odbc.JdbcOdbcDriver</driver>
            <user/>
            <password/>
        </resource-def>
        ...
    </queryset>
```

Change the port number to the port where your database connector is running. For example if you installed database.war into the Live Forms server's tomcat/webapps directory and Live Forms is running port 8082, then you must change the <url> from localhost to localhost:8082.

First we define a queryset and give it a name - BIRT in this case. The database connector can simultaneously connect to multiple databases; each must be defined in its own queryset. The queryset consists of a resource definition and any number of named queries. The resource definition is shown above for MySQL. You will need to modify it as required: change localhost to the name of the machine running MySQL, specify the MySQL driver, change the database user name and password.

Next we define a simple query named 'allCustomers'. Queries can contain four SQL statements (create, retrieve, update, delete). In this example, our query contains just one SQL statement to get the list of customers.

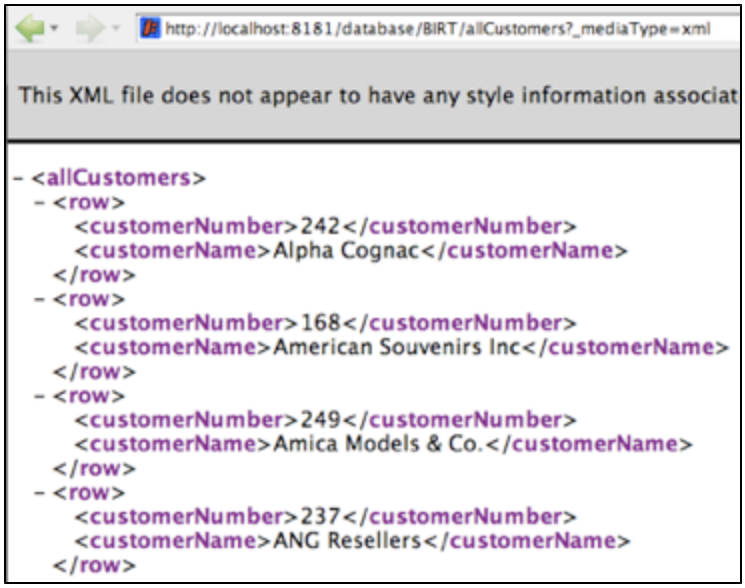
Mapping HTTP requests to SQL queries

The database connector maps between HTTP requests issued from Live Forms and SQL queries. It is important to understand how HTTP URLs map to SQL queries in the database connector. The above query returns a resultset that contains data as shown below:

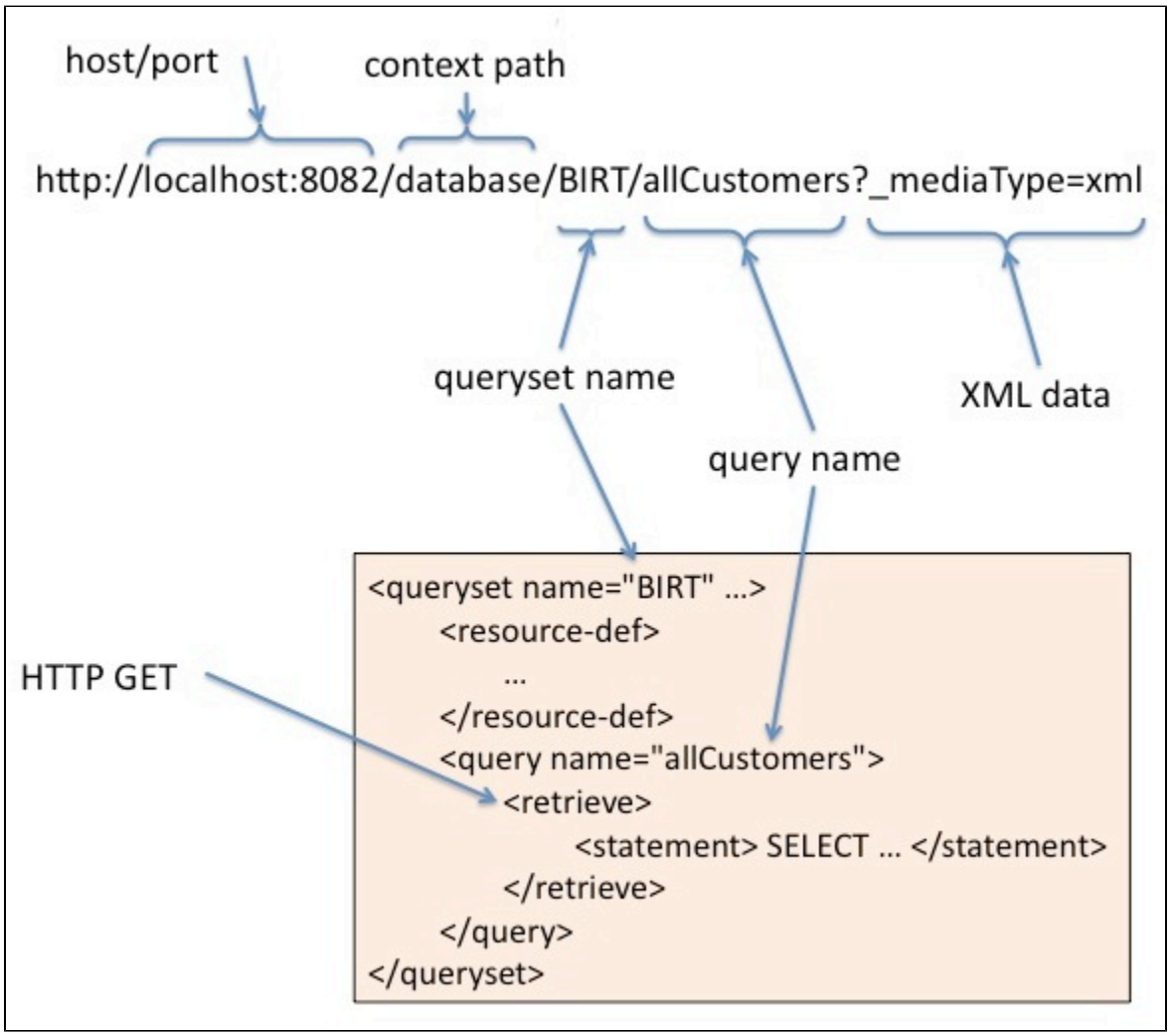
Query

※customerNumber	customerName
242	Alpha Cognac
168	American Souvenirs Inc
249	Amica Models & Co.
237	ANG Resellers
276	Anna's Decorations, Ltd
465	Anton Designs, Ltd.
206	Asian Shopping Network, Co
348	Asian Treasures, Inc.
103	Atelier graphique
471	Australian Collectables, Ltd
114	Australian Collectors, Co.
333	Australian Gift Network, Co
256	Auto Associés & Cie.
406	Auto Canal+ Petit
198	Auto-Moto Classics Inc.
187	AV Stores, Co.
121	Baane Mini Imports

Live Forms forms work with XML or JSON data. You can test the above query using the URL: **http://localhost:8082/database/BIRT/allCustomers**. This should return an XML document with customerNumber and customerName for each customer as shown below. The DB connector has converted the SQL data into a format that is usable by Live Forms.



Let's take a closer look at this URL.



- The 'localhost:8082' is obvious and points to the database connector inside a servlet container (e.g. Tomcat) on localhost and listening on port 8082.
- 'database' refers to the context path for the database connector web application running inside Tomcat.
- 'BIRT' refers to the name of the queryset defined above.
- 'allCustomers' is the name of the query.
- Since we are issuing an HTTP GET from the browser, this maps to the <retrieve> SQL.

- _mediaType=xml tells the connector to return an XML document. This is the default; if you leave out a _mediaType, the connector will return XML.
  When the HTTP GET is issued, the connector executes the <retrieve> SQL for the 'allCustomers' query in the 'BIRT' queryset using the resource definition to connect to the appropriate database. The data in the resultset is converted to XML and returned.

The connector is also capable of returning JSON data. Try the URL: **http://localhost:8082/database/BIRT/allCustomers?_mediaType=json,** th e connector will return the same data as JSON. Live Forms can use JSON data in business rules for dynamic behavior.

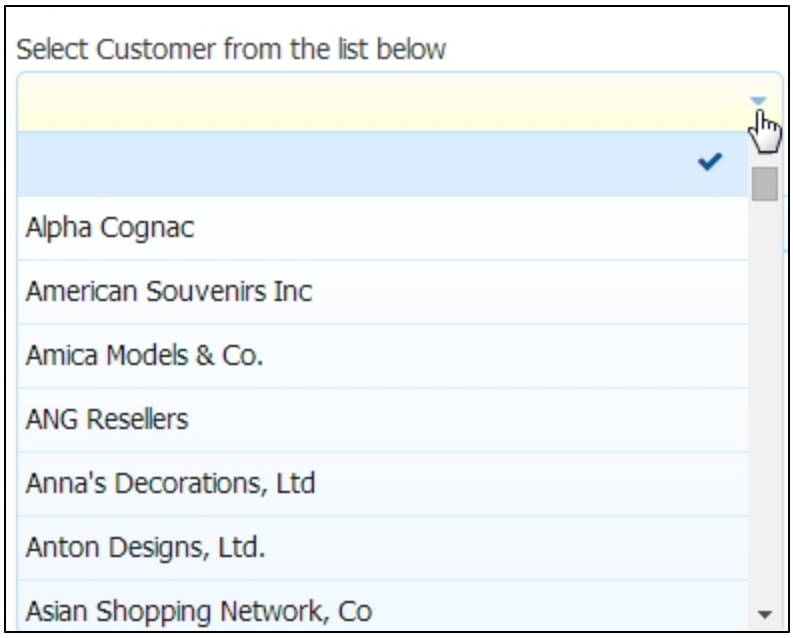

```
{"resultSet":[
{"customerName":"Alpha Cognac","customerNumber":"242"},
{"customerName":"American Souvenirs Inc","customerNumber":"168"},
{"customerName":"Amica Models & Co.","customerNumber":"249"},
{"customerName":"ANG Resellers","customerNumber":"237"},
{"customerName":"Anna's Decorations, Ltd","customerNumber":"276"},
{"customerName":"Anton Designs, Ltd.","customerNumber":"465"},
{"customerName":"Asian Shopping Network, Co","customerNumber":"206"},
{"customerName":"Asian Treasures, Inc.","customerNumber":"348"},
{"customerName":"Atelier graphique","customerNumber":"103"},
```


We will use these features in the examples below.

> It is very helpful to test your queries as shown above by entering the query URL directly into your web browser and verify that the data returned to your browser as a web page is as you expect. Note however that browsers often cache web pages. If you edit your configuration.xml SQL query and reload/refresh the URL in your browser you may NOT get the updated results due to your browser's caching. Avoid this caching issue by always opening a new browser tab to retest an updated query.

Example 1: View customer list

Our first example is to create a form with a dropdown control that is dynamically populated with the above list of customers from the database. When the user selects a customer by name, the value of the dropdown control should be set to the customer number.



The steps are as follows:

Define a query

We will use the query 'allCustomers' that we defined above.

Create a form

1. Create a new form. Accept the defaults by clicking the Finish button. The Form Designer will appear.
2. Drag a Dropdown control from the Palette and drop it into the form.
3. In the properties pane on the left, change the Label as desired.
4. In the properties pane, change the Name to sc.

Create a business rule

We will use a business rule to retrieve the list of customers from the database as JSON and dynamically populate this drop down control.

1. Click the
   
   icon in the toolbar at the top of the Form Designer.
2. Click
   
   to create a new rule and the
   
   Edit icon to open the rule.
3. Set the Name to 'Show customers'.
4. In the Rule text area: copy and paste the following:

```
/*member customerName, customerNumber, resultSet */
var x;

if (form.load)
{
    eval ('x=' + http.get('http://localhost:8082/database/BIRT/allCustomers'));
    var opts= [];
    for (var i=0; i < x.resultSet.length; i++) {
        if (x.resultSet[i]) {
            opts[i] = x.resultSet[i].customerNumber + '=' +
x.resultSet[i].customerName;
        }
    }
    sc.options = opts;
}
```

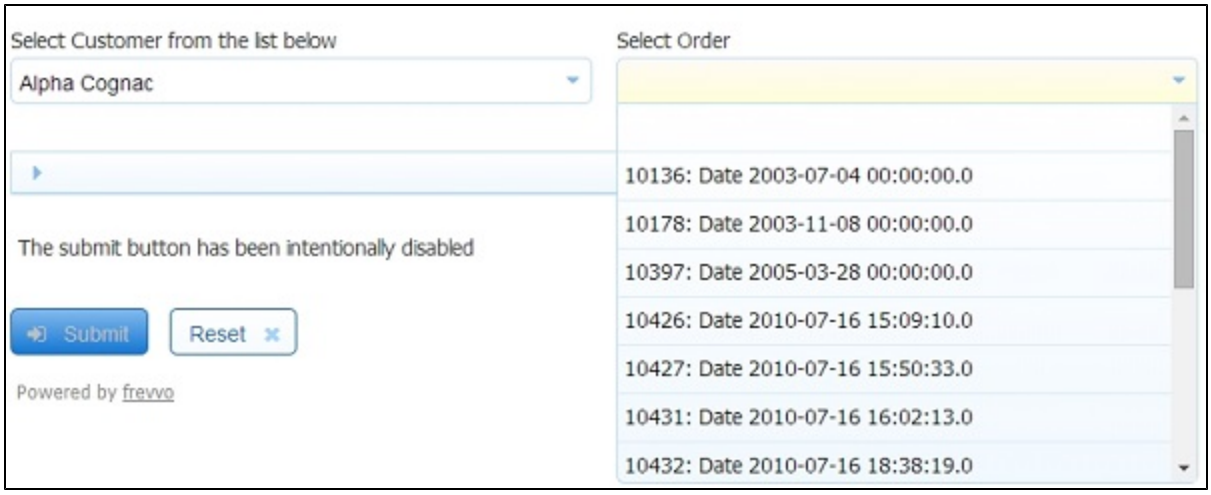
Let's analyze this rule.

1. if (form.load) - this implies that the rule will execute when the form first loads.
2. eval ('x=' + http.get('http://localhost:8082/database/BIRT/allCustomers')) - this causes an HTTP GET to be issued to the indicated URL. "When invoked in the context of a rule, the media type is automatically set to JSON". As we saw above, the database connector will return the list of customers as a JSON object.
3. The rest of the rule is just JavaScript. We create an array of options in the format value=label (e.g. 242=Alpha Cognac) from the data returned in the JSON.
4. Finally, we set the options of the dropdown to the array of options using the name we chose earlier (sc).

When the form is loaded, the drop down will be populated with the list of customers as shown in the image above.

Example 2: View Orders for a customer

Next, we will add a drop down that displays the list of orders for a particular customer. When the user selects a customer name from the first drop down, we will display the list of orders for that customer from the database. This is essentially repeating the steps above except that the database query is triggered by a user action (selecting a customer from the drop down) and depends on the selected customer.



Define a query

Check the configuration file for the following query. If it is not there, addit. Note that,**this query requires a parameter indicated by the {cnum}**.

```
<query name="ordersByCustomer">
  <retrieve>
    <!--maps to HTTP GET -->
    <statement>
      SELECT orderNumber, orderDate, status, customerNumber FROM Orders
      WHERE customerNumber={cnum} ORDER BY orderDate
    </statement>
  </retrieve>
</query>
```

Modify the form

1. Make a copy of the form from Example 1 and open it for editing by clicking the Edit button
2. Drag a Dropdown control from the Palette and drop it into the form.
3. Set the Label as desired.
4. Set the Name to 'so'.

Create a business rule

1. Click the
   
   icon in the toolbar at the top of the Form Designer.
2. Click
   
   to create a new rule and the
   
   Edit icon to open the rule.
3. Set the Name to 'Show orders'.
4. In the Rule text area: copy and paste the following:

```
/*member orderDate, orderNumber, resultSet */
var x;

if (sc.value.length > 0) {
  eval ('x=' + http.get('http://localhost:8082/database/BIRT/ordersByCustomer?cnum=' +
sc.value));
  var opts= [];
  for (var i=0; i < x.resultSet.length; i++) {
    if (x.resultSet[i]) {
      opts[i] = x.resultSet[i].orderNumber + '=' + x.resultSet[i].orderNumber + ':
Date ' + x.resultSet[i].orderDate;
    }
  }
  so.options = opts;
}
```

Let's analyze this rule.

1. if (sc.value.length > 0) - this implies that the rule will execute when the select customer drop down has a value and the value changes.
2. eval ('x=' + http.get('http://localhost:8082/database/BIRT/ordersByCustomer?cnum=' + sc.value)) - this causes an HTTP GET. We pass the selected customer number as the value of cnum to the query.
3. The rest of the rule is essentially identical to the previous rule. For the label we concatenate order number and order date.

When a customer is selected from the first drop down, the orders drop down will populate with orders for that customer. Select a different customer and the orders will update dynamically.

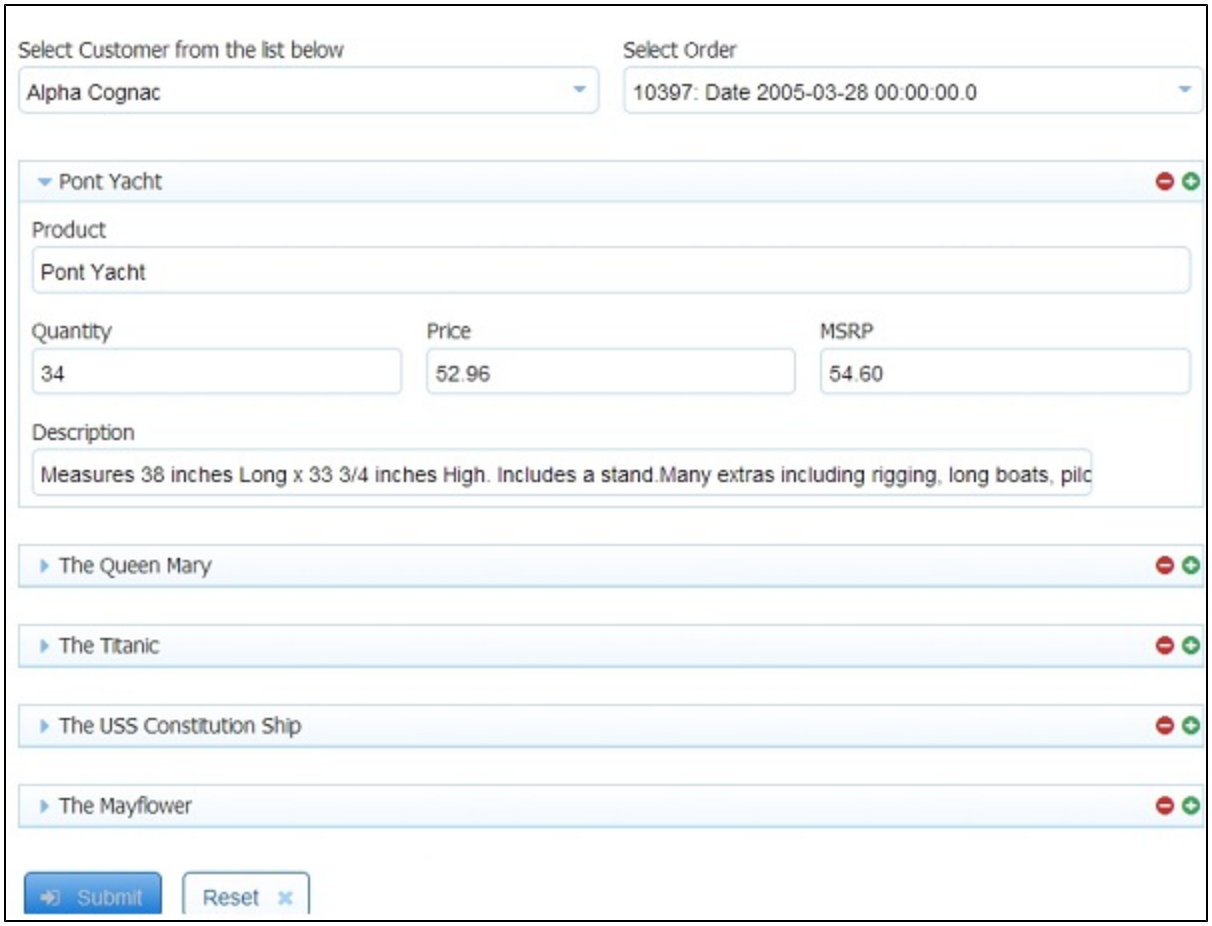
Example 3: View Details for an Order

Now, we will display order details for any given order. For example, for order number 10397 above, the order details resultset is shown below.

Query

product	quantity	price	MSRP	description
Pont Yacht	34	52.96	54.6	Measures 38 inches Long x 33 3/4 inches
The Queen Mary	36	80.44	99.31	Exact replica. Wood and Metal. Many extras
The Titanic	48	86.15	100.17	Completed model measures 19 1/2 inches
The USS Constitution Ship	22	62.88	72.28	All wood with canvas sails. Measures 31 1/2"
The Mayflower	32	69.29	86.61	Measures 31 1/2 inches Long x 25 1/2

Since order details is a complex structure (product name, quantity, price etc.) we will use Live Forms's built-in XML features to generate controls and populate them from the database. When an order is selected, the order line items will be displayed as shown below:



Define a query

Check the configuration file for the following query. If it is not there, add it. The query requires a parameter indicated by {onum}.

```
<query name="orderDetailsByOrder">
  <retrieve>
    <!--maps to HTTP GET -->
    <statement> SELECT p.productName as product, o.quantityOrdered as quantity,
o.priceEach as price,
                   p.productDescription as description, p.MSRP FROM OrderDetails o,
Products p WHERE
                   o.productCode=p.productCode and o.orderNumber={onum} ORDER by
o.orderLineNumber
    </statement>
  </retrieve>
</query>
```

Generate XML schema

The database connector can generate an XML schema from any query's <retrieve> operation. This schema can then be used to generate a Live Forms form.

1. In a browser go to the URL: **http://localhost:8082/database/BIRT/orderDetailsByOrder/schema?onum=10100**. Save the resulting XML schema file to disk. Note that we need to pass in a known order number so that the query can successfully execute.
2. Upload the schema to Live Forms. See Data Sources for instructions.

Modify the form

See Data Sources for help on the steps below.

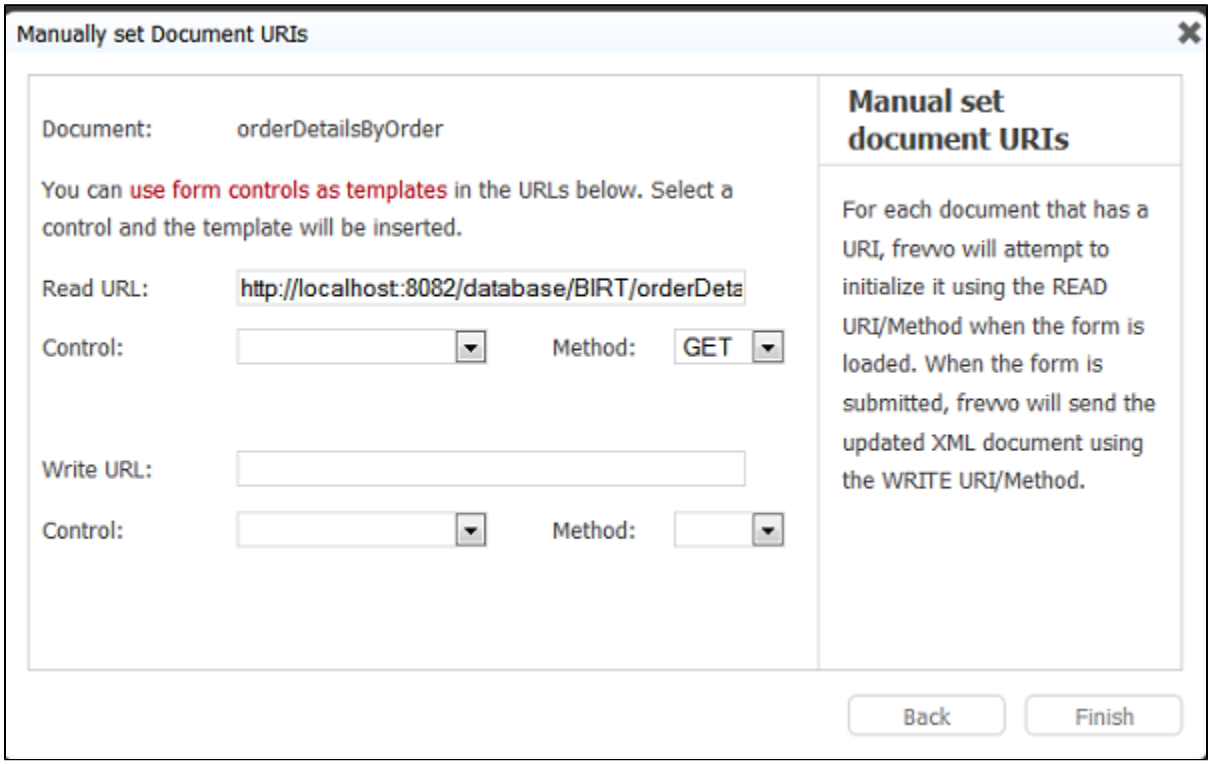
1. Make a copy of the form from Example 2 above and open it for editing.
2. In the properties panel on the left, open the Data Sources pane by clicking on it.
3. Click 'New from XSD'.
4. You should see your schema in the resulting dialog box. Expand it and add the 'orderDetailsByOrder' element to the form by clicking the + icon next to it.
5. Close the dialog box by clicking X in the top right corner.
6. In the Data Sources pane, click the + icon next to the orderDetailsByOrder data source. Live Forms will generate controls in the form.
7. You can drag and drop to re-arrange as you wish.
8. Change the label of the Section named 'row' to the string '{product}'. We'll explain this in more detail below.
9. Collapse this Section.

This step generates controls that are bound to the orderDetailsByOrder data source by Live Forms. You can render the controls as you wish but L ive Forms will always generate an XML document conforming to the XML schema above.

Link to the database

Now, we connect the generated controls to the database query defined above. See Doc URl Wizards for help on the steps below.

1. Click on the Doc Action button in the toolbar at the top of the form.
2. In the dialog box that appears click on 'Manually set document URIs'.
3. A new dialog box will appear. Click the Next button.
4. You should see the Document name change to 'orderDetailsByOrder', which is the data source we are interested in.
5. Set the URL to: **http://localhost:8082/database/BIRT/orderDetailsByOrder?onum={so}**.
6. Set the Read method to GET.
7. Click the Finish button.



This step indicates to Live Forms that the 'orderDetailsByOrder' document is linked to the above URL via the READ method GET. Live Forms will issue an HTTP GET request to the above URL and if an XML document conforming to the schema above is returned, Live Forms will automatically initialize controls in the form using that document.

The URL itself is a dynamic URL and has a parameter indicated by {so}. Remember that we have a control in the form named 'so' (defined in this step. When the value of the control named 'so' changes (which will happen when the user selects a particular order), Live Forms will automatically issue an HTTP GET to the above URL using the new order number and will dynamically update the order details including creating new line items, removing old ones etc. without refreshing the form as shown in the picture below.


print
Select Customer from the list below
Alpha Cognac
Select Order
10136: Date 2003-07-04 00:00:00.0
10178: Date 2003-11-08 00:00:00.0
10397: Date 2005-03-28 00:00:00.0
Select order number 10397
http://localhost:8082/database/BIRT/orderDetailsByOrder?onum={so}
frevvo resolves URL
http://localhost:8082/database/BIRT/orderDetailsByOrder?onum=10397
DB Connector returns XML which automatically updates form
Select Customer from the list below
Alpha Cognac
Select Order
10397: Date 2005-03-28 00:00:00.0
Pont Yacht
Product
Pont Yacht
Quantity
34
Price
52.96
MSRP
54.60
Description
Measures 38 inches Long x 33 3/4 inches High. Includes a stand.Many extras including rigging, long boats, pilc
The Queen Mary
The Titanic
The USS Constitution Ship
The Mayflower
Submit
Reset

Example 4: Create a new order

Now, let's look at submitting form data to the database. We will create a form using which a customer can place a new order. The order is then saved to the database. Note that we have simplified the form for demonstration purposes.

Define queries

We will define two queries. The first one creates an order. In addition to the usual <retrieve> operation, this query also has a <create> operation. Check the configuration file for this query. If it is not there, add it.

```
<query name="createOrder">
            <retrieve>
                <statement>
                    SELECT orderNumber as onum, customerNumber as cnum from Orders
                    WHERE orderNumber=10100</statement>
            </retrieve>
            <create>
                <statement>INSERT into Orders (orderNumber,orderDate, requiredDate,
status,customerNumber)
                    VALUES ({onum}, Now(), Now(), 'In Process', {cnum})
                </statement>
            </create>
        </query>
```

Let's understand this query. We specify a <retrieve> operation (SELECT statement) in order to generate an XML schema/form controls. When the form is submitted, Live Forms generates an XML document for those controls. The XML document is used to execute the <create> operation, which INSERTs the data. We have simply hard-coded a known order number for the example but you can use any query that generates a result set with the desired columns.

Note that we'll just insert the current date as the order date and required date for this example. You can easily customize the query to add a date of your choice. If you want the date to come from the form, you must add it to the SELECT statement so that a control is generated in the form. Dates/Times tend to be database-specific and we'll look at an example later.

The second query is simply used to generate an order number. Check the configuration file for this query. If it is not there, add it.

```
<query name="getOrderNumber">
            <retrieve>
                <statement>SELECT max(orderNumber) + 1 as onum FROM Orders</statement>
            </retrieve>
        </query>
```

Once again, this is an oversimplification for tutorial purposes. In practice, order numbers might be generated by the outer application rather than the database.

Generate XML schema

- In your browser go to the URL: **http://localhost:8082/database/BIRT/createOrder/schema**. Save the resulting XML schema file to disk.
- Edit the file. Due to a current limitation in the database connector, the generated schema needs to be edited. Remove the maxOccurs="unbounded" attribute from the row element declaration. Replace:

```
<xsd:element maxOccurs="unbounded" name="row">
```

with

```
<xsd:element name="row">
```

- Upload the schema to Live Forms. See Data Sources for instructions.

Create the form

See Data Sources for help on the steps below.

1. Make a copy of the form from Example 1 and open it for editing.
2. In the properties panel on the left, open the Data Sources pane by clicking on it.
3. Click 'New from XSD'.
4. You should see your schema in the resulting dialog box. Expand it and add the 'createOrder' element to the form by clicking the + icon next to it.
5. Close the dialog box by clicking X in the top right corner.
6. In the Data Sources pane, expand the createOrder data source by clicking on the + icon to the left.
7. Click the + icon next to the row element. Live Forms will generate controls in the form.
8. You can drag and drop to re-arrange as you wish.
9. Rename the Section named Row to Order Info.

This step generates controls that are bound to the createOrder data source by Live Forms. You can render the controls as you wish but Live Forms will always generate an XML document conforming to the XML schema above.

Create a business rule

1. Click the
   
   icon in the toolbar at the top of the Form Designer.
2. Click
   
   to create a new rule and the
   
   Edit icon to open the rule.
3. Set the Name to 'Copy customer number and generate an order number'.
4. In the Rule text area: copy and paste the following:

```
/*member onum, resultSet */
var x;
if (sc.value.length > 0) {
  cnum.value = sc.value;
  eval ('x=' + http.get('http://localhost:8082/database/BIRT/getOrderNumber'));
  onum.value = x.resultSet[0].onum;
}
```

Let's analyze this rule.

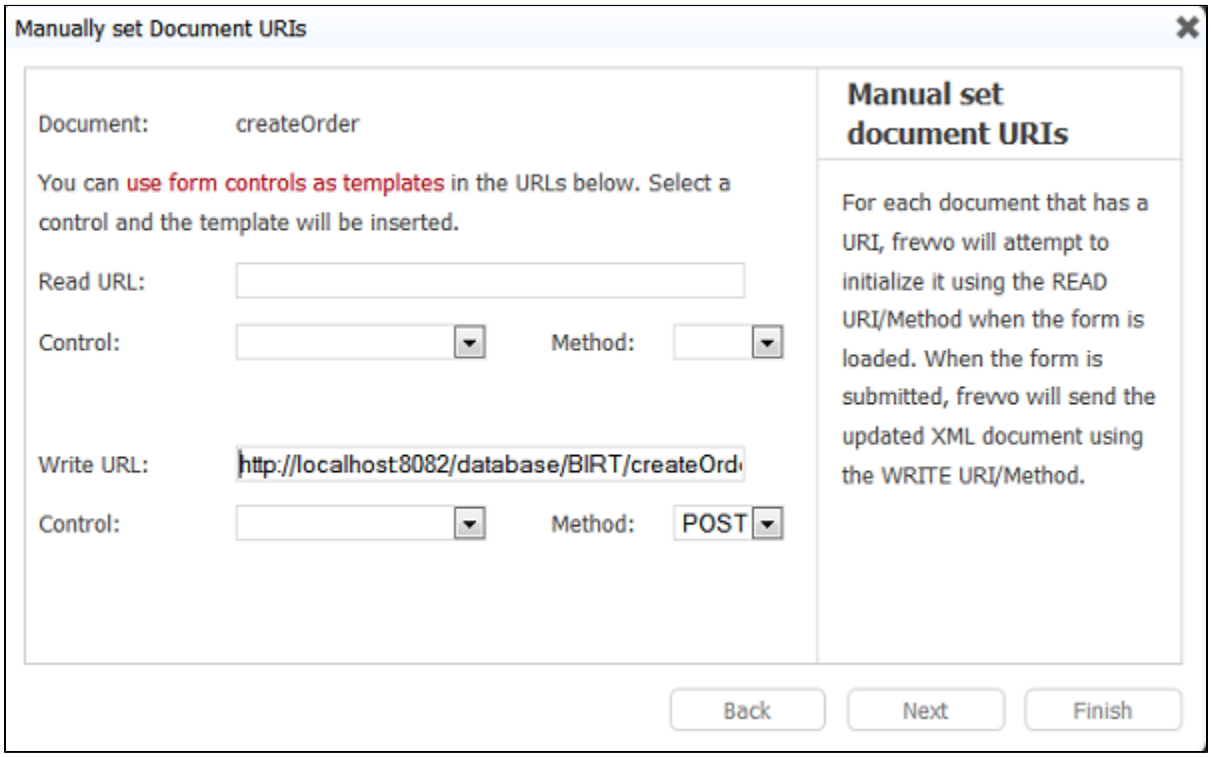
1. if (sc.value.length > 0) - this implies that the rule will execute when the select customer drop down has a value and the value changes.
2. cnum.value = sc.value; simply copies the customer number into the cnum control.
3. eval ('x=' + http.get('http://localhost:8082/database/BIRT/getOrderNumber')) - this gets an order number HTTP GET.
4. onum.value = x.resultSet[0].onum; copies the order number from the result set into the onum control.

Link to the database

Now, we connect the generated controls to the database query defined above. See Doc URI Wizards for help on the steps below.

1. Click on the Doc Action button in the toolbar at the top of the form.
2. In the dialog box that appears click on 'Manually set document URIs'.

3. A new dialog box will appear. Click the Next button.
4. You should see the Document name change to 'createOrder', which is the data source we are interested in.
5. Leave the Read method empty.
6. Set the Write URL to: **http://localhost:8082/database/BIRT/createOrder**.
7. Set the Write method to POST.
8. Click the Finish button.



This step indicates to Live Forms that the 'createOrder' document is linked to the above URL via the WRITE method POST. When the form is submitted, Live Forms will issue an HTTP POST request to the above URL sending the createOrder XML document in the payload. The database connector will map this to the <create> operation of the createOrder query that we defined above, will use the data in the XML document to resolve the query (replace {cnum} and {onum} with values from the XML) and perform the INSERT.

Example 5: Add Order Details

Now, we'll add line items to the order.

Define queries

Once again, we will require two queries. The first one fetches product codes and descriptions from the database so that the user can select a product for purchase. Check the configuration file for this query. If it is not there, add it.

```
<query name="allProducts">
            <retrieve>
                <!--maps to HTTP GET -->
                <statement> SELECT productCode, productName from Products order by
productName </statement>
            </retrieve>
        </query>
```

The second query creates the order detail line item records in the database. Check the configuration file for this query. If it is not there, add it.

```
<query name="createOrderDetail">
            <retrieve>
                <statement>SELECT orderNumber as
onum,productCode,quantityOrdered,priceEach,orderLineNumber
                    from OrderDetails where orderNumber=10100
                </statement>
            </retrieve>
            <create>
                <statement>INSERT into OrderDetails
(orderNumber,productCode,quantityOrdered,priceEach,orderLineNumber)
                    VALUES ({onum},
'{productCode}',{quantityOrdered},{priceEach},{orderLineNumber})
                </statement>
            </create>
        </query>
```

The steps are very similar to the earlier examples.

Generate XML schema

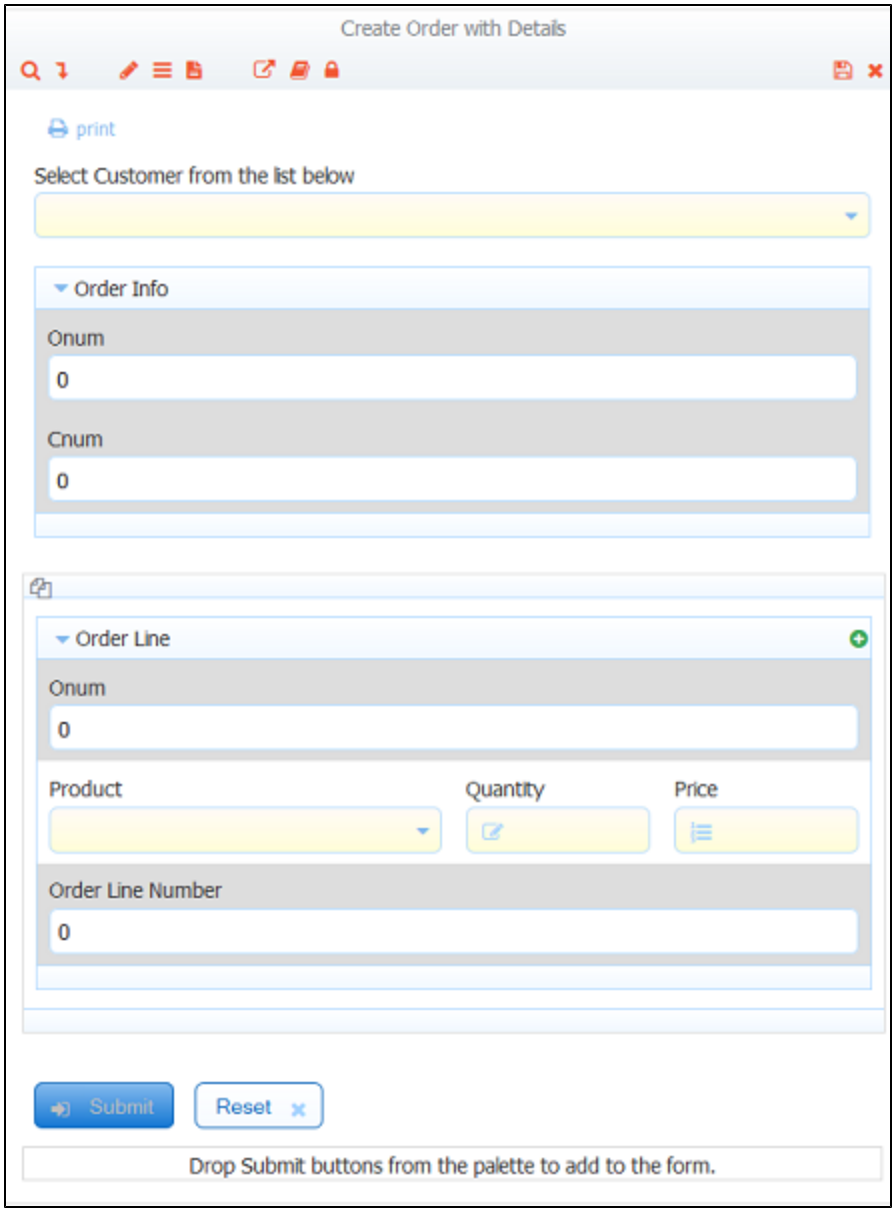
- In your browser go to the URL: **http://localhost:8082/database/BIRT/createOrderDetail/schema**. Save the resulting XML schema file to disk.
- Upload the schema to Live Forms. See Data Sources for instructions.

Modify the form

See Data Sources for help on the steps below.

1. Make a copy of the form from Example 4 above and open it for editing.
2. We can hide the Order Info section since it does not contain information entered by the user. Select the section by clicking on its header and uncheck the Visible checkbox in the properties pane. This section will be grayed out when you click on another control on the canvas.
3. In the properties panel on the left, open the Data Sources pane by clicking on it.
4. Click 'New from XSD'.
5. You should see your schema in the resulting dialog box. Expand it and add the 'createOrderDetail' element to the form by clicking the + icon next to it.
6. Close the dialog box by clicking X in the top right corner.
7. In the Data Sources pane, expand the createOrderDetail data source by clicking on the + icon to the left.
8. Click the + icon next to the row element. Live Forms will generate controls in the form. Note that Live Forms automatically generates a repeating element since a single Order can contain any number of line items.
9. You can drag and drop to re-arrange as you wish.
10. Change the label and the name the Section named Row to Order Line.
11. Name the repeat control OrderLines. Take note of the "s" in the name.
12. Select the Product Code input control. In the properties panel:
    a. Set the name to pc.
    b. Select Dropdown from the Display As property list.
13. Select the Onum control inside the Order Line section and in the properties pane set the name to olonum. Uncheck the Visible property.
14. Select the Order Line Number control inside the Order Line section and in the properties pane set the name to oln. Uncheck the Visible property.
15. Set default values in the Onum and Cnum controls inside the Order Info section and in the Onum and Order Line Number controls inside the Order Line section. You can set any values since we will be overriding them in a rule. We need to set a default value otherwise the form cannot be submitted.

This step generates controls that are bound to the createOrderDetail data source by Live Forms. You can render the controls as you wish but Live Forms will always generate an XML document conforming to the XML schema above. Here's what our form looks like:



Create/Update business rules

- Click the
  
  icon in the toolbar at the top of the Form Designer.
- Edit the Show Customers rule we created in Example 4. Change the name to Show Customers/Products.
- In the Rule text area: copy and paste the following:

```
/*member customerName, customerNumber, productCode, productName, resultSet */
var x;
if (form.load)
{
  eval ('x=' + http.get('http://localhost:8082/database/BIRT/allCustomers'));
  var opts= [];
  for (var i=0; i < x.resultSet.length; i++) {
    if (x.resultSet[i]) {
      opts[i] = x.resultSet[i].customerNumber + '=' + x.resultSet[i].customerName;
    }
  }
  sc.options = opts;
  // Products.
  eval ('x=' + http.get('http://localhost:8082/database/BIRT/allProducts'));
  opts= [];
  for (var i=0; i < x.resultSet.length; i++) {
    if (x.resultSet[i]) {
      opts[i] = x.resultSet[i].productCode + '=' + x.resultSet[i].productName;
    }
  }
  pc[0].options = opts;
}
```

This rule will fire on form load and initialize the customer drop down as before. In addition, we initialize the products drop down in each order line (there could be multiple) with the list of products from the database.

- Edit the 'Copy customer number and generate an order number' rule we created in Example 4.
- In the Rule text area: copy and paste the following:

```
/*member onum, resultSet */
var x;
if (form.unload) {
  cnum.value = sc.value;
  eval ('x=' + http.get('http://localhost:8082/database/BIRT/getOrderNumber'));
  onum.value = x.resultSet[0].onum;
  for (var i = 0; i < olonum.value.length; i++) {
    olonum[i].value = x.resultSet[0].onum;
    oln[i].value = i+1;
  }
}
```

This rule will fire when the form unloads (just after the user clicks Submit) and set the values of the hidden fields. We make sure that the Order has a valid customer number and order number, the order number is copied into each order detail element and the order line number is correctly set.

- Click + to create a new rule and the Edit button to open the rule.
- Set the Name to 'Order Line Item Added'.

1. In the Rule text area: copy and paste the following:

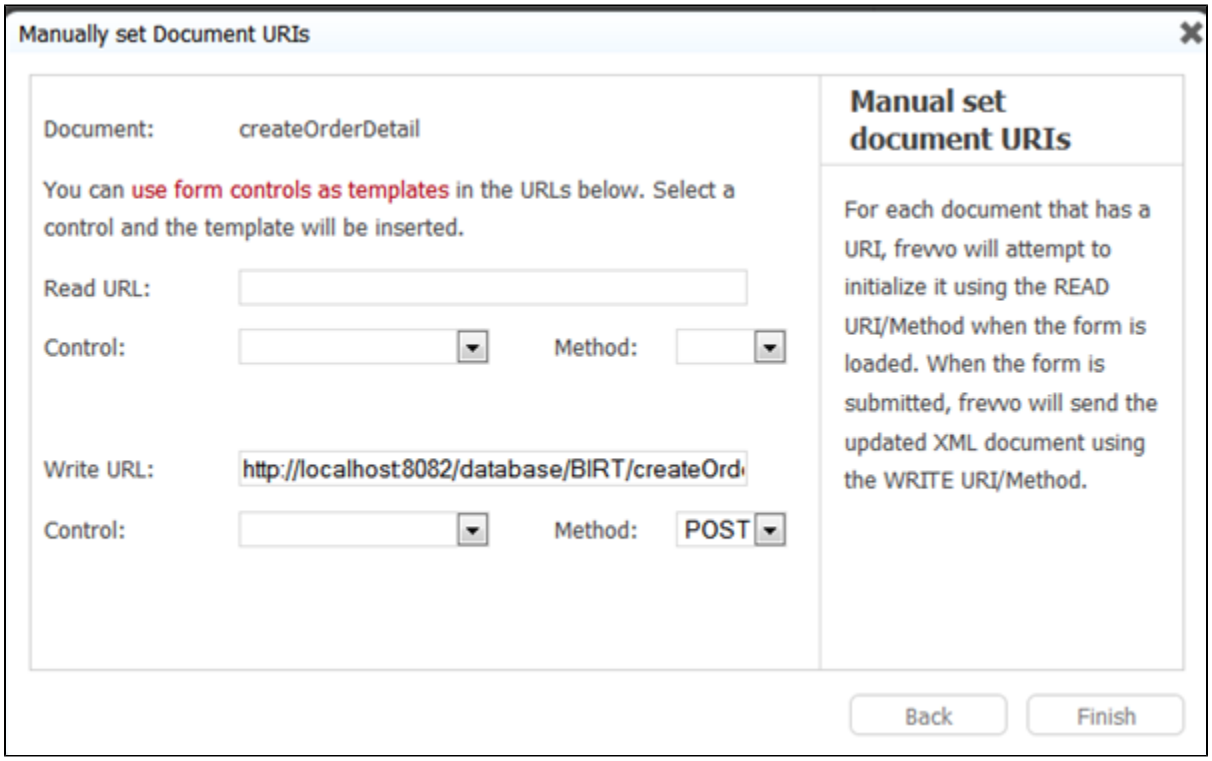
```
if (OrderLines.itemAdded) {
  var ix = OrderLines.itemIndex;
  pc[ix].options = pc[0].options;
  oln[ix].value = 0;
  olonum[ix].value = 0;
}
```

This rule will fire when the user adds a new Order Line repeat item. We copy the product list into the newly added Order Line and set default values in the hidden fields.

Link to the database

Finally, we connect the createOrderDetails document to the database query defined above. See Doc URI Wizards for help on the steps below.

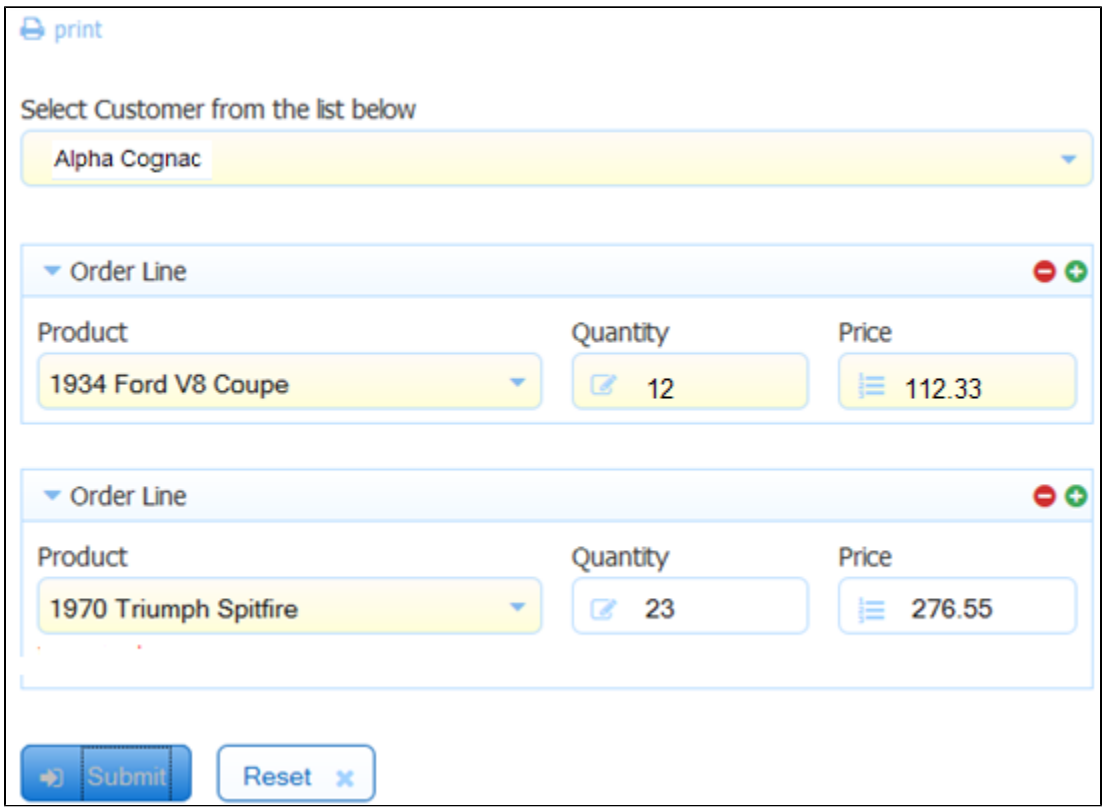
1. Click on the Doc Action button in the toolbar at the top of the form.
2. In the dialog box that appears click on 'Manually set document URIs'.
3. A new dialog box will appear. Navigate to the desired document (createOrderDetail) by clicking the Next button twice.
4. Set the URL to: **http://localhost:8082/database/BIRT/createOrderDetail**.
5. Leave the Read method empty.
6. Set the Write method to POST.
7. Click the Finish button.



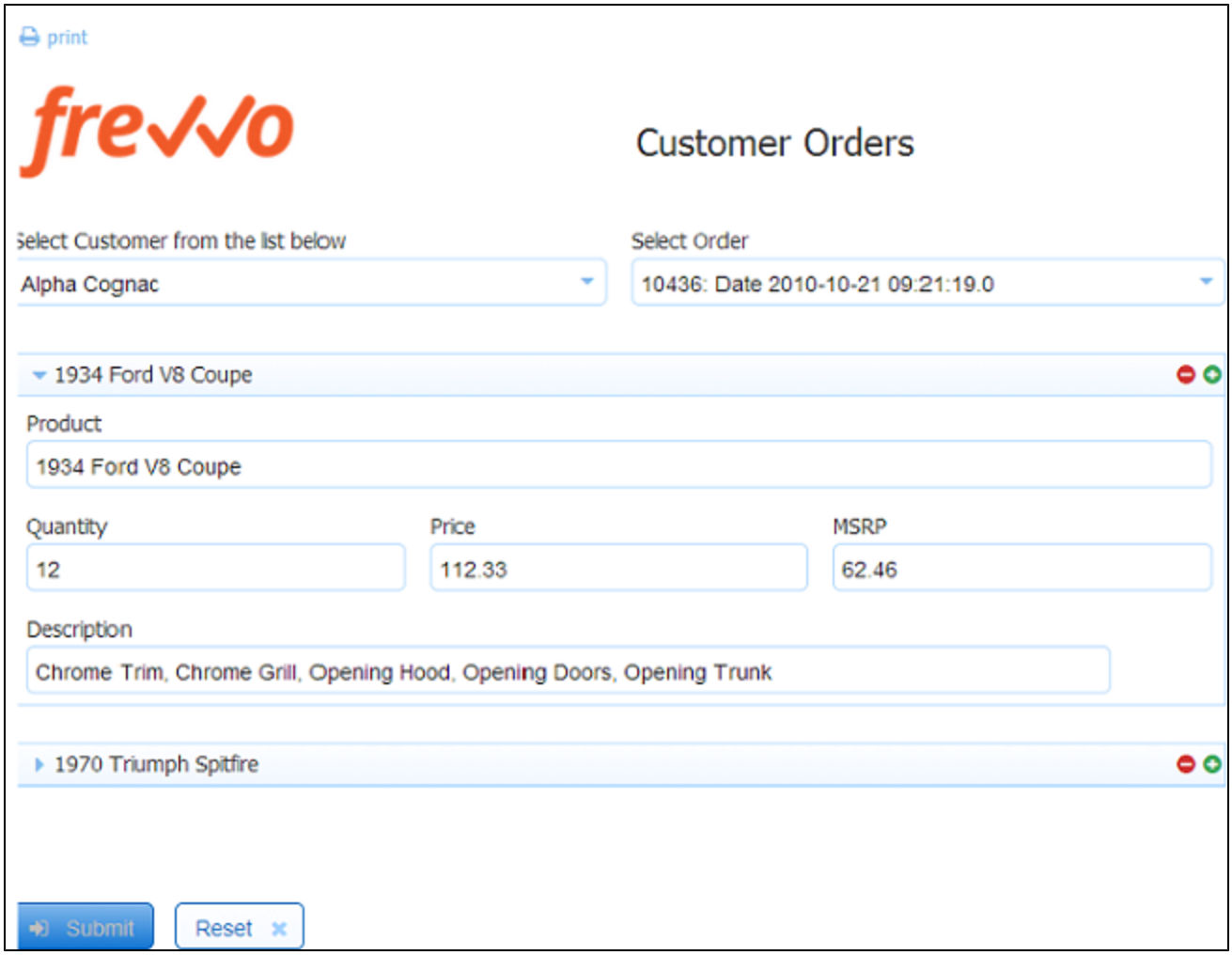
When the form is submitted, Live Forms will issue an HTTP POST request to the above URL sending the createOrderDetail XML document in the payload. The database connector will map this to the <create> operation of the createOrderDetail query that we defined above, will use the data in the XML document to resolve the query. Since there are multiple line items, the query will be executed once for each Order line item thereby inserting the multiple line items for an Order.

Test the example

- Click the Test button.
- Select a customer from the drop down.
- Enter at least one line item.
- Submit the form.



- Verify that the order was successfully inserted.
- Use the form from Example 3 by clicking the Test button.
- Select the customer.
- You should see your order. Select the order.
- You should see Line Item details appear below.



Example 6: Update existing records

In this example, we will update the credit limit for an existing customer.

Define query

As always we have to first define the query. This query has an <update> operation in addition to the usual <retrieve> operation. We retrieve the customerName in the retrieve operation purely for display purposes. Check the configuration file for this query. If it is not there, add it.

```
<query name="customerCreditLimit">
            <retrieve>
                <statement>
                    SELECT customerNumber as cnum,customerName,creditLimit FROM
Customers where customerNumber={cnum}
                </statement>
            </retrieve>
            <update>
                <statement>
                    UPDATE Customers SET creditLimit = {creditLimit} WHERE
customerNumber={cnum}
                </statement>
            </update>
        </query>
```

Generate XML schema

- In your browser go to the URL: **http://localhost:8082/database/BIRT/customerCreditLimit/schema?cnum=242**. Save the resulting XML schema file to disk.
- Edit the file. Due to a current limitation in the database connector, the generated schema needs to be edited. Remove the maxOccurs="unbounded" attribute from the row element declaration. Replace:

```
<xsd:element maxOccurs="unbounded" name="row">
```

with

```
<xsd:element name="row">
```

- Upload the schema to Live Forms. See Data Sources for instructions.

Create the form

See Data Sources for help on the steps below.

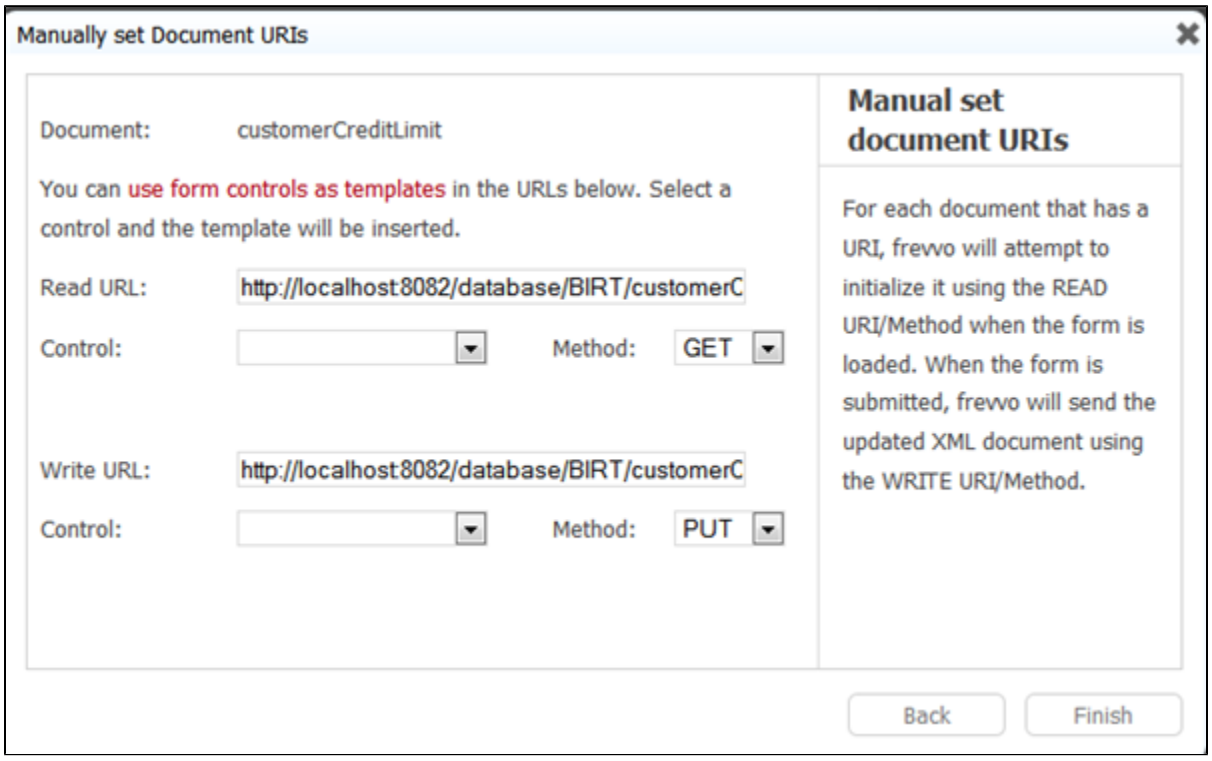
1. Make a copy of the form from Example 1 and open it for editing.
2. In the properties panel on the left, open the Data Sources pane by clicking on it.
3. Click 'New from XSD'.
4. You should see your schema in the resulting dialog box. Expand it and add the 'customerCreditLimit' element to the form by clicking the + icon next to it.
5. Close the dialog box by clicking X in the top right corner.
6. In the Data Sources pane, expand the customerCreditLimit data source by clicking on the + icon to the left and expand the row element.
7. Click the + icon next to the cnum and creditLimit elements. Live Forms will generate controls in the form.
8. You can drag and drop to re-arrange as you wish.
9. Hide the cnum element by unchecking the Visible checkbox in the properties pane.

Link to the database

Connect the customerCreditLimit document to the database query defined above. See Doc URI Wizards for help on the steps below.

1. Click on the Doc Action button in the toolbar at the top of the form.
2. In the dialog box that appears click on 'Manually set document URIs'.
3. A new dialog box will appear. Navigate to the desired document (customerCreditLimit) by clicking the Next button.
4. Set the Read and Write URLs to: **http://localhost:8082/database/BIRT/customerCreditLimit?cnum={sc}**.
5. Set the Read method to GET.

6. Set the Write method to PUT.
7. Click the Finish button.



This step indicates to Live Forms that the 'customerCreditLimit' document is linked to the above URL via the READ method GET and the WRITE METHOD PUT. Live Forms will issue an HTTP GET request to the above URL and if an XML document conforming to the schema above is returned, Live Forms will automatically initialize controls in the form using that document. The URL itself is a dynamic URL and has a parameter indicated by {sc}. When the value of the control named 'sc' changes (which will happen when the user selects a customer), Live Forms will automatically issue an HTTP GET to the above URL using the new customer number and will dynamically update the values of the credit Limit and cnum controls without refreshing the form.

When the form is submitted, Live Forms will issue an HTTP PUT request to the above URL sending the customerCreditLimit XML document in the payload. The database connector will map this to the <update> operation of the customerCreditLimit query that we defined above, will use the data in the XML document to resolve the query (replace {creditLimit} and {cnum} in the query with the actual values from the XML document) and perform the UPDATE.

```
<query name="customerByNumber“>
            <retrieve>
                  <!-- maps to HTTP GET -->
                  <statement>
    SELECT * from Customers where customerNumber={cnum}
                  </statement>
            </retrieve>
</query>
```