



# AltioLive Developer Training

Version 5.4

## 4. Using Events and Actions to Create a Dynamic Application

---

### *Summary*

In this course we will explain how to call Service Functions with events and actions in order to create a dynamic application.

### **Integra SP**

88 Wood Street London  
EC2V 7RS  
United Kingdom

[www.altio.com](http://www.altio.com)

tel: +44 (0) 20 8528 1045

## Contents

Introduction .....	2
Introduction to Service Functions.....	2
Introduction to Windows and Views .....	3
Introduction to Events and Actions .....	3
Events.....	3
Actions .....	3
Examples .....	4
Creating Service Functions to Add and Delete Elements in the Stock Database.....	5
Creating a NEW_STOCK service function.....	5
Creating a DELETE_STOCK service function .....	7
Creating a Data Entry window and its controls with the Designer .....	9
Creating and setting a Data Entry window .....	9
Adding Text controls to the Data Entry window.....	9
Adding a Rectangle and a Tab controls to the Data Entry window .....	11
Setting the Tab control .....	12
Setting the first panel of the Tab control.....	12
Setting the second panel of the Tab control.....	13
Adding buttons to the Data Entry window .....	14
Saving the View.....	15
Creating Actions and Using Events to trigger them .....	16
Adding and Setting two buttons to the Main Window.....	16
Adding an Event and an Action to the Add button.....	18
Adding an Action to the Cancel button on the Data Entry window .....	19
Adding an Action to the Add Stock button on the Data Entry window .....	19
Using Conditions and Masks for the Add Stock button .....	21
Making the selection of a Sector and a Market mandatory to add a stock.....	21
Making the EPIC and Company name fields mandatory by setting their control Properties.....	23
Restricting the Middle Price and Percentage Change fields to accept figure entries only.....	23
Setting a conditional action for the Delete button on the Main window .....	24
JavaScript .....	26
Functions.....	26
Further Exercise: Stocks History Graph.....	26

# Introduction

---

Previously, we defined a Service Function, called **GET\_STOCKS**, to retrieve an XML document using the Altio DB Manager. This was used as an initial data in the Designer. In this module we will explore further service functions for adding and deleting elements in the stocks data, using events triggering actions such as opening new windows.

During this module you will create and set:

- New Service Functions: **NEW\_STOCK** and **DELETE\_STOCK**,
- A new window called **Data Entry** with several controls (Text control, Tab control, button, select control, date picker,...)
- Events linked to these controls,
- And actions triggered by these events.

This module is divided in three main steps:

- Creating service functions to add and delete elements in the **Stock** database,
- Creating a **Data Entry** window and its control with the Designer,
- Creating Actions and Events to trigger them.

## Introduction to Service Functions

All requests to send and receive data from the back end application (for initial data and/or updates) are defined as service functions.

A variety of communication methods are available with service functions; HTTP, SOAP (to integrate with Web Services), Java Messaging Service, and JDBC to integrate with an SQL back-end, as well as direct calls to Java classes. This session only covers HTTP service functions.

The exact definition of service functions depends on the back-end applications they are dealing with. For service functions using HTTP, the URL for the HTTP function will require a server address, a path and a query string to identify the data required and where it can be accessed.

Several service functions are usually required to:

- Get initial data
- Send an update to data
- Delete data
- Add new data
- Get an image

Service functions are designed to provide simple integration with a back-end system and (in the case of updates) to map onto existing HTTP post requests.

In this session, the example service functions are specific to the Altio DB configuration. As most HTTP based service functions have a similar form to their definition, it is unlikely that they would be very different in other cases.

## Introduction to Windows and Views

A View defines a specific user interface presented through the AltioLive Smart Client. There can be one or more Views per AltioLive Application.

Views are normally created, modified and tested through the Designer tool, though advanced users can configure it directly in XML using standard text editors.

All of the Controls (for example: text boxes, buttons or lists) in AltioLive applications must be placed on windows. There are no restrictions on the quantity of windows which can be used in one application. A newly created window will automatically be assigned a Data Source of **`\${windowcontext}`**. This tells the window to use the context element used to open the window as the window's data element. Each window can have its own style defined in Window Style Format. For more information on Window Style, see the second module of the training: **The First Steps to build your own application**.

## Introduction to Events and Actions

### Events

An event occurs due to user interaction or the server causing something to happen on the client. Not all controls have the same events available to them, for example: command button will not have data events as no data is associated with a command button, while a list control will have data update events.

Examples of events available:

- Click
- Right-click
- Got focus
- Lost focus
- Data added
- Data updated
- Data deleted
- Data changed

Each event can trigger one or more action(s).

### Actions

Action can be conditional on certain criteria. For example, it is not possible to delete a row from a list if no row is selected; therefore it is possible to specify this as a condition prior to the action being processed.

Many actions are available including:

- Open window
- Close window

- Print
- Popup menu
- Prompt
- Refresh control
- Set focus to a control
- Set control/window properties
- Submit to server
- Save/Update/Delete local XML
- Show a URL
- Call JavaScript

## Examples

This session looks at how we control an application using Actions triggered by Events. Here are some examples:

- A user enters text in a text control and clicks an **Update** button: this is the Event which triggers one or several action(s). Here, Actions are used to take the data from the text control in the application window and then update the data behind it. The Event of the **Update** button (click) triggers an Action (updating).
- A user drags data from one list onto another: this is the Event which triggers the Action(s). Here, an Action is used in the destination list to manage the change to the list. A control receiving information (the Event) triggers an operation (the Action).

Which Events are available depends upon which type of control is selected.

# Creating Service Functions to Add and Delete Elements in the Stock Database

This practical exercise involves the creation of two service functions so that stocks may be added to the stock database, and stocks may be deleted. Using the Application Manager, we will add the new HTTP service functions and configure a datakey. Once the service functions are defined and tested then we will return to the Designer to create a data entry window to allow stock to be added to the database.

## Creating a NEW\_STOCK service function

As with all HTTP service functions, we need to be aware of the server address, path and query string. Also, when updating and adding data, we will be required to pass parameters.

When defining the service function we can map the parameters required by the back-end application to parameters on the client side.

For the **NEW\_STOCK** service function, the names on the client side of the parameter mapping will match the names of the controls in our data entry window. The names on the server side will be added as attributes to a new **STOCK** element in the database.

1. From the Studio Console, right-click on the **STOCKS** project and select the **Application Manager** tool in the context menu.
2. Since we are going to the same URL, the easiest way to create the Service Function is to copy the existing one: From the **Application View** window, expand the **Services** node and right-click on the **GET\_STOCKS** Service Function. Select **Clone** in the context menu.
3. Open the new service function, called **GET\_STOCKS1**, by double-clicking on it.
4. In the **Service Name** field, enter the new name of the service function: **NEW\_STOCK**
5. On the **Request** tab, add the following **parameter mapping** entries:

REQUEST TAB		
Name	Value	Description
APPID	\${application.id}	Substituted with the APPID from the session
ACTION	NEW	Fixed string required by Altio DB
TARGET	/STOCKS	Parent of the new STOCK element
AL_NAME	STOCK	Name of the new element
GUID	STOCK_ID	Attribute name Altio DB will use for a unique ID

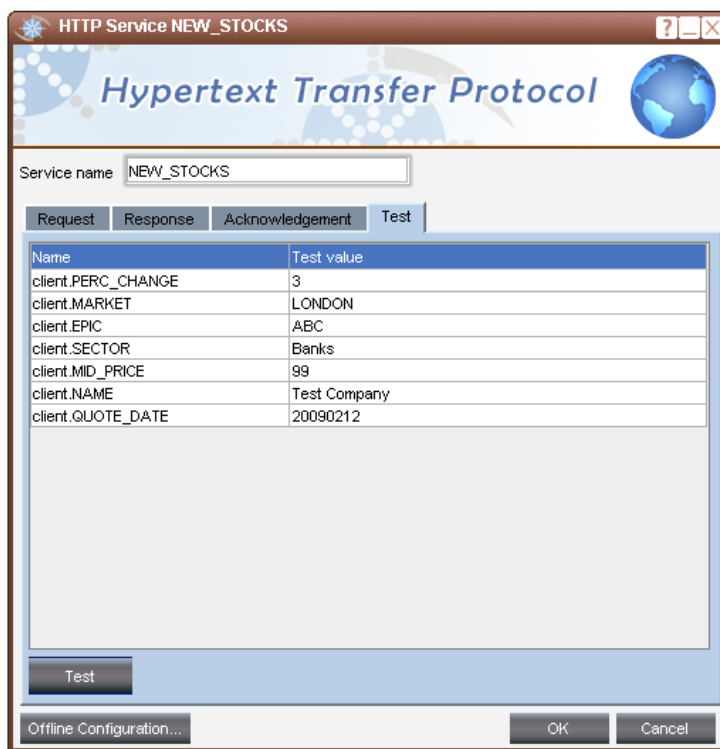
REQUEST TAB (2)		
Name	Value	Description
EPIC	\${client.EPIC}	These attributes will be mapped from the named controls on the client windows to the attribute names at the server
NAME	\${client.NAME}	
MARKET	\${client.MARKET}	
SECTOR	\${client.SECTOR}	
MID_PRICE	\${client.MID_PRICE}	
PERC_CHANGE	\${client.PERC_CHANGE}	
QUOTE_DATE	\${client.QUOTE_DATE}	

6. On the **Acknowledgement** tab, set the following parameters:

ACKNOWLEDGEMENT TAB		Description
Source	HEADER	The Altio Presentation Server searches for a header parameter named <b>X-Altio-Status</b> that specifies whether the service has failed or succeed.
On success: Source	DEFINE	Option that allows to enter a text to send when a service succeed
On success: Text	A new stock has been added to the database	This is the message that will be send to the <b>AltioLive</b> client when a service succeed
On failure: Source	DEFINE	Option that allows to enter a text to send when a service fail
On failure: Text	It was not possible to add the new stock	This is the message that will be send to the <b>AltioLive</b> client when a service fail

7. On the **Test** tab, test the service function by:

- a. adding test values in the **Test value** field. For example:



- b. pressing the **Test** button.

This should produce a result similar to the XML below (although some of the values depend on your test values):

```
<DATA AL_ID="">
  <STOCKS AL_tableLocation="dbdata/stocks.xml" AL_tableUpdated="Y" AL_ID="STOCKS">
    <STOCK EPIC="ABC" SECTOR="Banks" MARKET="LONDON" MID_PRICE="99" QUOTE_DATE="20090212"
    STOCK_ID="1006" PERC_CHANGE="3" NAME="Test Company" AL_TS="8" AL_ID="S-1006"/>
  </STOCKS>
</DATA>
```

- 8. Note down the **STOCK\_ID** of this new record you will need it further in the exercise. In the example above, it is 1006.

**Please note:**

- AltioDB has added a unique **STOCK\_ID** and the Presentation Server added a timestamp (**TS**).
- The fields in the header response:
  - X-Altio-Status: 0**
  - X-Altio-Message: Your record has been accepted.**

- 9. Press **OK** to accept the service function.

## Creating a DELETE\_STOCK service function

Now we will define a **Delete** service function with the **Application Manager**. Later, when we return to the **Designer**, we will use the service functions in our **Stocks** application.

1. In the **Application Manager**, again clone the **GET\_STOCKS** Service Function by right-clicking on it and selecting **Clone** in the context menu.
2. Open the new service function, called **GET\_STOCKS2**, by double-clicking on it.
3. In the **Service Name** field, enter the new name of the service function: **DELETE\_STOCK**
4. On the **Request** tab, add the following **parameter mapping** entries:

REQUEST TAB		
Name	Value	Description
APPID	\${application.id}	Substituted with the APPID from the session
ACTION	DELETE	Fixed string required by Altio DB
STOCK_ID	\${client.STOCK_ID}	This attribute will be mapped from the named control on the client windows to the attribute name at the server
TARGET	/STOCKS/STOCK[@STOCK_ID=\${client.STOCK_ID}]	Xpath to indicate the target of the service

5. On the **Acknowledgement** tab, set the following parameters:

ACKNOWLEDGEMENT TAB		Description
Source	HEADER	The Altio Presentation Server searches for a header parameter named <b>X-Altio-Status</b> that specifies whether the service has failed or succeed.
On success: Source	DEFINE	Option that allows to enter a text to send when a service succeed
On success: Text	The selected stock has been deleted	This is the message that will be send to the <b>AltioLive</b> client when a service succeed
On failure: Source	DEFINE	Option that allows to enter a text to send when a service fail
On failure: Text	The selected stock could not be deleted	This is the message that will be send to the <b>AltioLive</b> client when a service fail


6. Go to the **Test** tab, the Application Manager has determined that the mandatory parameter of **client.STOCK\_ID** will be supplied by the client at runtime.
7. On the **Test** tab, test the service function by:
  - a. adding a test value for **client.STOCK\_ID** in the **Test value** field. For example the one you have created, you should have noted it. (In our example, it was **1006** -see the **Test results** for the **NEW\_STOCK** service function-).
  - b. pressing the **Test** button.

This should produce a result similar to the XML below (although some of the values depend on your test values):

```
<DATA AL_ID="">
  <STOCKS AL_tableLocation="dbdata/stocks.xml" AL_tableUpdated="Y" AL_ID="STOCKS">
    <STOCK EPIC="ABC" SECTOR="Banks" MARKET="LONDON" MID_PRICE="99" QUOTE_DATE="20090212"
    STOCK_ID="1006" PERC_CHANGE="3" NAME="Test Company" AL_TS="10" AL_ACTION="DELETE" AL_ID="S-1006"/>
  </STOCKS>
</DATA>
```

**Please note:**

- the deleted record has an additional attribute of **AL\_ACTION='DELETE'**. When the client receives this, it will delete the element from its internal database.
- the fields in the header response:
  - X-Altio-Status: 0**
  - X-Altio-Message: 1 record(s) have been deleted.**



8. Click on the **Close** button to exit the **Test Results** window.
9. Press **OK** to accept the service function.
10. Save the Application by clicking on the **Save** icon .

# Creating a Data Entry window and its controls with the Designer

This section looks at creating a data entry screen which makes use of tab panels and the service functions created in the previous section.

Now that the Service Functions have been defined, we can return to the Designer and create a window to add a new stock to the database. At this point we are designing the interface only; at the next session will look at the Actions required to open windows and call the service functions.


## Creating and setting a Data Entry window

1. Add a new window to the view by doing one of the following:
  - From the main menu, select **Edit | New Window**,
  - From the **View Explorer** window, right-click on the **Windows** folder and select **New Window** in the context menu,
  - From the main menu, click on the **New Window** button .
2. From the **View Explorer** window, select the new window and display the **Properties** window by clicking on the **View Properties** icon .
3. Expand the **General** properties and set the window properties as follows:

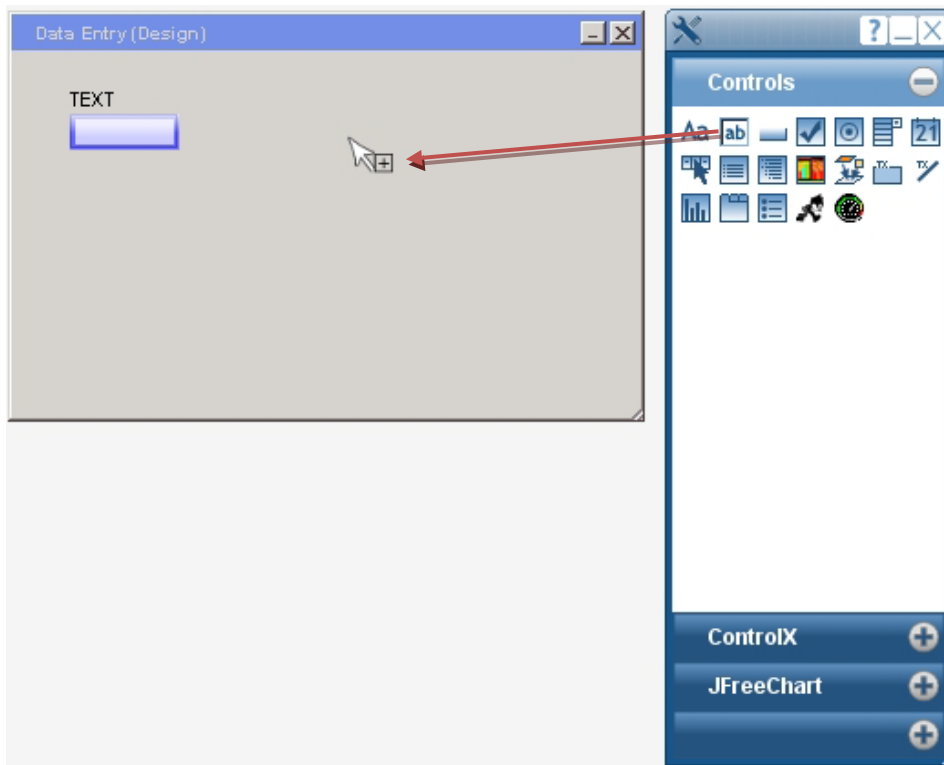
Property	Value	Comments
▼ <b>General</b>		
Name	DE_WIN	
Caption	Data Entry	
Show on startup	N	The window will not show until it is invoked
Can have multiple instance	N	It is not possible to display several DE_WIN windows.
Window style	STYLE	The style we created in the <b>First Steps...</b> module.


## Adding Text controls to the Data Entry window

We will add several editable controls to this window so that new data can be submitted to the database. Remember that the names on the controls need to correspond to the client parameters defined in the service function. These are: **EPIC**, **COMPANY**, **MARKET**, **SECTOR**, **MID\_PRICE**, **PERC\_CHANGE**, and **QUOTE\_DATE**.

1. From the **View Explorer** window, double-click on the **DE\_WIN** window to display a preview of the window.
2. If the **Control Palette** does not display, click on the **Control Palette** icon .

3. Drag and drop two Text controls  from the **Control Palette** onto the window.







4. From the **View Explorer** window, select the first Text control, called **TEXT** and display the **Properties** window by clicking on the **View Properties** icon  .
5. Expand the **General** properties and set them as follows:


Property	Value
▼ <b>General</b>	
Name	EPIC
Caption	Epic

6. From the **View Explorer** window, select the second Text control, called **TEXT1**.
7. Expand the **General** properties and set them as follows:

Property	Value
▼ <b>General</b>	
Name	NAME
Caption	Company Name

## Adding a Rectangle and a Tab controls to the Data Entry window

1. If the preview of the **Data Entry** window is not displayed, from the **View Explorer** window, double-click on the **DE\_WIN** window.
2. If the **Control Palette** is not display, click on the **Control Palette** icon .
3. Drag and drop a Tab control  from the **Control Palette** onto the window.
4. The Tab control properties do not need any changes for the moment.
5. Drag and drop a Rectangle control  from the **Control Palette** onto the window.
6. From the **View Explorer** window, select the Rectangle control, and display the **Properties** window by clicking on the **View Properties** icon .
7. Expand the **General** properties and set the **Caption** property to empty.
8. Position and resize the controls so that the window looks like the one shown below:


**Please note:** You can position the controls by using the cursor keys and use your pointer to resize them when your cursor displays as a double arrow: .



## Setting the Tab control

### Setting the first panel of the Tab control


We want to add two select controls to the Tab control to allow users to select a **MARKET** and a **SECTOR**. The easiest way is to copy the ones you have created for the **MAIN\_WIN**.

1. In the **View Explorer** window, open the **MAIN\_WIN** node to reveal the controls.
2. Right-click on **MARKET\_SELECT** and choose the **Copy Select Control** option.
3. Still in the **View Explorer** window, expand the **DE\_WIN** window to reveal the tab control.
4. Expand the Tab control, right-click on **TAB\_PANEL1** and choose **Paste Select Control**.
5. Repeat steps 1 to 4 with the **SECTOR\_SELECT** control.
6. From the **View Explorer** window, select the **MARKET\_SELECT1** under the **TAB\_PANEL1** node.
7. If the **Properties** window is not displayed, click on the **View Properties** icon .
8. From the Properties window, expand the **General** node and set the **MARKET\_SELECT1** properties as follows:

Property	Value
▼ <b>General</b>	
Name	MARKET
Caption	Market
Default Value	Please specify

9. From the **View Explorer** window, select the **SECTOR\_SELECT1** under the **TAB\_PANEL1** node.
10. From the Properties window, expand the **General** node and set the **SECTOR\_SELECT1** properties as follows:




Property	Value
▼ <b>General</b>	
Name	SECTOR
Caption	Sector
Default Value	Please specify

11. Right-click on the **Market** select control and choose **Options** in the context menu.
12. Select the **All** option and click the **Delete** button: .
13. Close the **Options list** window.
14. Repeat steps 11 and 13 for the **Sector** select control.
15. From the **View Explorer** window, select the **TAB\_PANEL1** under the **TAB** node.
16. From the **Properties** window, expand the **General** node and set the **TAB\_PANEL1** properties as follows:

Property	Value
▼ <b>General</b>	
Name	CLASSIFICATION_TAB
Caption	Classification

## Setting the second panel of the Tab control


On the second panel, called **TAB\_PANEL2**, we will add two more text controls and a date picker.

1. If the preview of the **Data Entry** window is not displayed, from the **View Explorer** window, double-click on the **DE\_WIN** window.
2. If the **Control Palette** is not displayed, click on the **Control Palette** icon .
3. Drag and drop two Text controls  from the **Control Palette** onto the second panel.
4. From the **View Explorer** window, select the first Text control called **TEXT**.
5. If the **Properties** window is not displayed, click on the **View Properties** icon .
6. From the **Properties** window, expand the **General** node and set the **TEXT** properties as follows:

Property	Value
▼ <b>General</b>	
Name	MID_PRICE
Caption	Middle price

7. From the **View Explorer** window, select the second Text control called **TEXT1**.
8. From the **Properties** window, expand the **General** node and set the **TEXT1** properties as follows:

Property	Value
▼ <b>General</b>	
Name	PERC_CHANGE
Caption	Percentage change

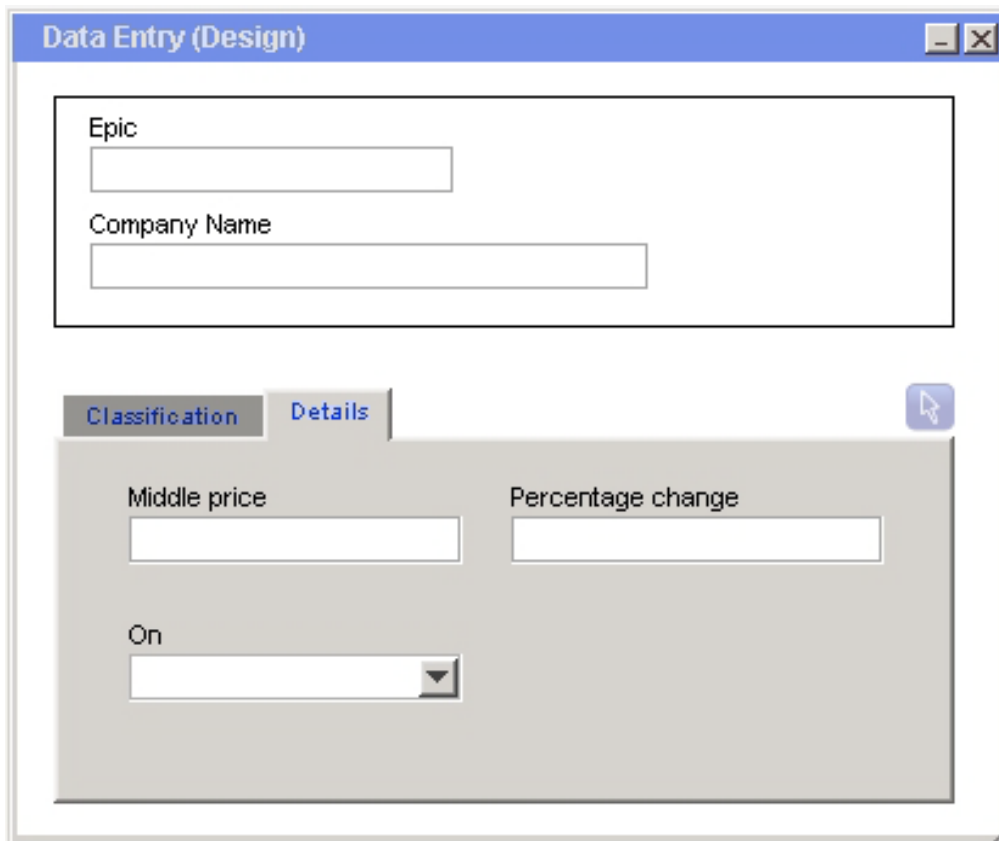
9. Drag and drop a Date Picker control  from the **Control Palette** onto the second panel.
10. From the **View Explorer** window, select the Date Picker control.
11. From the **Properties** window, expand the **General** node and set the **DATEPICKER** properties as follows:

Property	Value
▼ <b>General</b>	
Name	QUOTE_DATE
Caption	On




12. From the **View Explorer** window, select the **TAB\_PANEL2** under the **TAB** node.
13. From the **Properties** window, expand the **General** node and set the **TAB\_PANEL2** properties as follows:

Property	Value
▼ <b>General</b>	
Name	DETAILS_TAB
Caption	Details

14. Position and resize the controls so that the window looks like the one shown below:



## Adding buttons to the Data Entry window

1. If the preview of the Data Entry window does not display, from the **View Explorer** window, double-click on the **DE\_WIN** window.
2. Resize the window so that you have enough space to add two buttons below the tab control.
3. If the **Control Palette** does not display, click on the **Control Palette** icon .
4. Drag and drop two Button controls  from the **Control Palette** onto the window.
5. From the **View Explorer** window, select the **BUTTON** control.
6. If the **Properties** window is not displayed, click on the **View Properties** icon .
7. From the **Properties** window, expand the **General** node and set the **BUTTON** properties as follows:

Property	Value
▼ <b>General</b>	
Name	ADD_BUTTON
Caption	Add Stock


- From the **View Explorer** window, select the **BUTTON1** control.
- From the **Properties** window, expand the **General** node and set the **BUTTON1** properties as follows:

Property	Value
▼ General	
Name	CANCEL_BUTTON
Caption	Cancel

- Position and resize the controls so that the window looks like the one shown below:



## Saving the View

- Close the **Data Entry** Window.
- Save the View by clicking on the **Save** icon .

In the next session we will add actions to invoke this window and submit a new entry to the database.

# Creating Actions and Using Events to trigger them




In this practical exercise we will use the **Open Window**, **Close Window** and **Submit to Server** Actions. To familiarize ourselves with conditions we will check to see if various controls have been selected before each Action is processed.

We will trigger the actions from the buttons placed on the windows we created in the previous sessions.

1. If the **Designer** tool is not displaying the **Stock** view, follow this steps:
  - a. From the **Altio Live Studio**, in the **Studio Explorer** window, expand the **My projects** node.
  - b. Right-click on the **STOCKS** project and select **Designer** in the context menu.

The **Designer** displays the **Stocks** view.

## Adding and Setting two buttons to the Main Window

2. In the **View Explorer** window, expand the **Windows** folder.
3. Display a preview of the **Main** window: from the **View Explorer** window, double-click on the **MAIN\_WIN** window.
4. If the **Control Palette** does not display, click on the **Control Palette** icon .
5. Drag and drop two Button controls  from the **Control Palette** onto the window.
6. From the **View Explorer** window, select the **BUTTON** control.
7. If the Properties window is not displayed, click on the **View Properties** icon .
8. From the **Properties** window, set the first **BUTTON** properties as follows:

Property	Value
▼ <b>General</b>	
Name	ADD
Caption	Add
▼ <b>Appearance</b>	
Up image	al_stdskin/al_2_yesdragimg

9. From the **View Explorer** window, select the **BUTTON1** control.

10. From the **Properties** window, set the second **BUTTON** properties as follows:

Property	Value
<b>General</b>	
Name	DELETE
Caption	Delete
<b>Appearance</b>	
Up image	al_stdskin/al_2_nodragimg

**Please note:** We have chosen images (*al\_stdskin/al\_2\_nodragimg* and *al\_stdskin/al\_2\_yesdragimg*) from the standard Altio skin file for the exercise but you can use custom images.

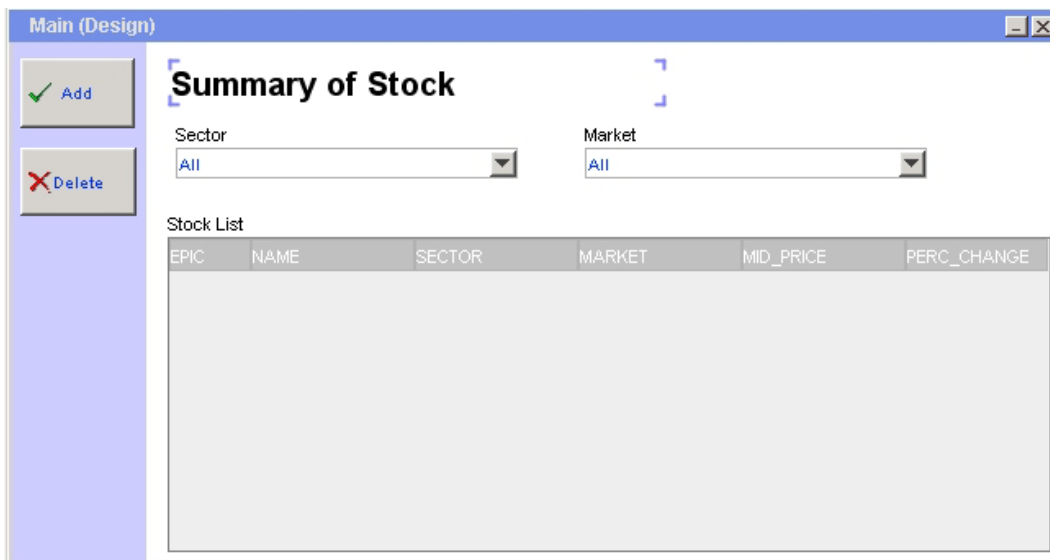
11. Drag and drop a Rectangle control  from the **Control Palette** onto the window.

12. From the **View Explorer** window, select the **RECTANGLE** control.

13. From the **Properties** window, expand the **General** node and set the **RECTANGLE** properties as follows:


Property	Value
<b>General</b>	
Name	RECTANGLE2
Caption	
Fill color	0xCCCCFF
Thickness	0
<b>Position/Size</b>	
Stretch to fill window	V

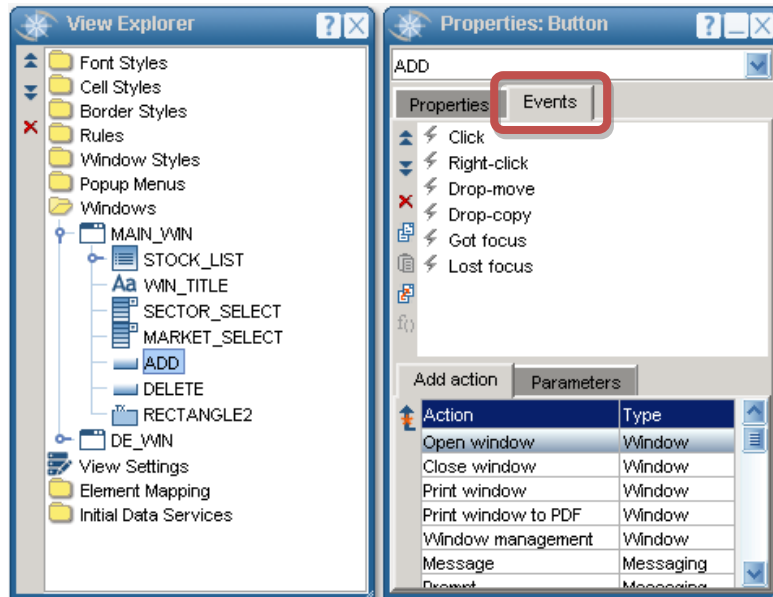
14. Position and resize the controls so that the window looks like the one shown below:




## Adding an Event and an Action to the Add button

We will now add an Action to the **Add** button to open the **Data Entry** window.

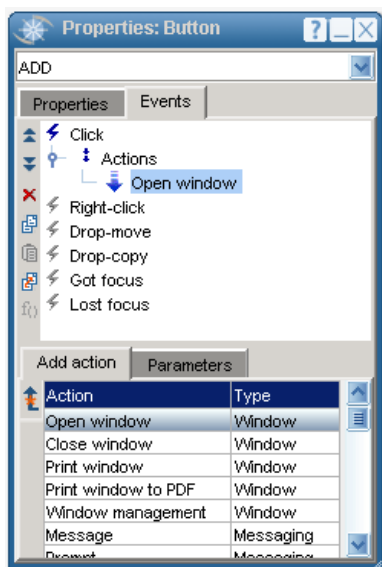
1. If the **Properties** window is not displayed, click on the **View Properties** icon .
2. From the **View Explorer** window, select the **ADD** button.
3. The properties window has two tabs. Select the **Events** tab:



The top part of the **Events** tab shows all the Events possible for this control type. The lower part shows all the possible actions.


4. We want to add a **Open Window** action for the **Click** Event, so select the **Click** Event and either:
  - double-click on the **Open window** Action or
  - Select the **Open window** Action and press the **Add Action** button .


You will see that the **Open Window** action is added to the **Click** event:



- Most actions require parameters to be defined. Switch to the **Parameters** tab and set the **Window** Parameter to **DE\_WIN** Value.



This specifies that the **Open window** action will open the **DE\_WIN** window.

- Test that the action works in test mode by clicking on the **Run** button . When you press the **Add** button, the **Data Entry** window should be invoked.

- Save the View by clicking on the **Save** icon .


Now we need to add actions to the Data Entry window.

## Adding an Action to the Cancel button on the Data Entry window

- In the **View Explorer** window, expand the **Windows** folder and the **DE\_WIN** window under it.
- From the **View Explorer** window, select the **Cancel** button.
- If the **Properties** window is not displayed, click on the **View Properties** icon .
- From the **Properties** window, click on the **Events** tab.
- Select the **Click** Event and either:
  - double-click on the **Close window** Action or
  - Select the **Close window** Action and press the **Add Action** button .


- For the **Close window** Action the only parameters are:
  - which window to close
  - whether to close all instances of a multiple-instance window.

Since the default setting is to close the window invoking the action, we don't need to specify either parameter.


- Test that the actions work in test mode by clicking on the **Run** button .
  - Press the **Add** button on the **Main** window, the **Data Entry** window should be invoked.
  - Press the **Cancel** button on the **Data Entry** window, the **Data Entry** window should close.

## Adding an Action to the Add Stock button on the Data Entry window

Now we will add an action to the **Add Stock** button to call the **NEW\_STOCK** service function.


- From the **View Explorer** window, expand the **DE\_WIN** window under the **Windows** folder.
- Select the **ADD\_BUTTON** and if the **Properties** window is not displayed, click on the **View Properties** icon .
- From the **Properties** window, click on the **Events** tab.

4. Select the **Click** Event and either:

- double-click on the **Server request** Action or
- Select the **Server request** Action and press the **Add Action** button .

5. Click on the **Parameters** tab and set them as follows:

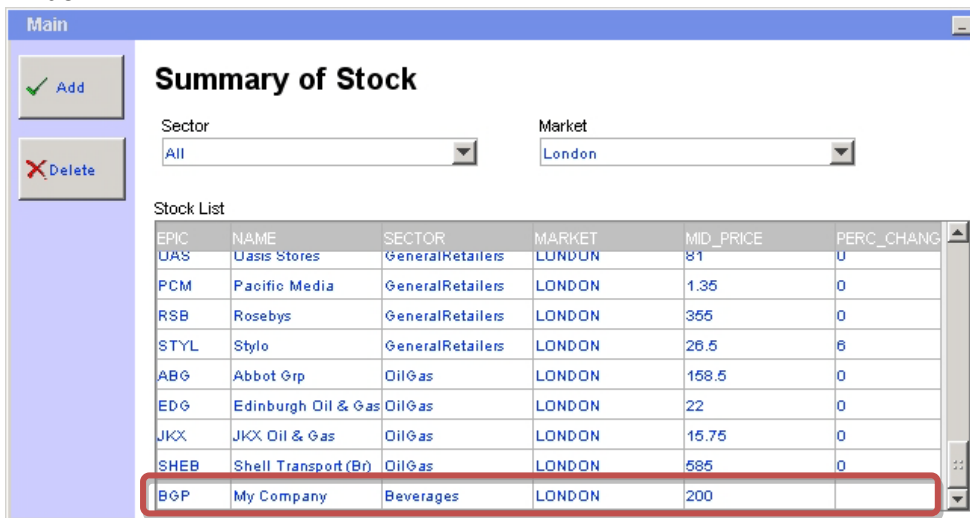
PARAMETERS TAB		
Parameter	Value	Description
Service Function	NEW_STOCK	The Service Function you have created earlier in this module
Parameter source	WINDOWDATA	Sends the value associated with each control on the window.
Window State	CLOSE	The window will be closed when the service is invoked

6. Test that the actions work in test mode by clicking on the **Run** button .

- Press the **Add** button on the **Main** window, the **Data Entry** window should be invoked.
- Enter values for all the fields in the **Data Entry** screen including both tabs. For example:



- When you press **Add Stock** you should get an acknowledgement as we defined in the service function. New stocks should appear at the bottom of the list on the **Main** window:




EPIC	NAME	SECTOR	MARKET	MID_PRICE	PERC_CHANG
UAS	Uasis Stores	GeneralRetailers	LONDON	81	0
PCM	Pacific Media	GeneralRetailers	LONDON	1.35	0
RSB	Rosebys	GeneralRetailers	LONDON	355	0
STYL	Style	GeneralRetailers	LONDON	26.5	0
ABG	Abbot Grp	OilGas	LONDON	158.5	0
EDG	Edinburgh Oil & Gas	OilGas	LONDON	22	0
JKX	JKX Oil & Gas	OilGas	LONDON	15.75	0
SHEB	Shell Transport (Br)	OilGas	LONDON	585	0
BGP	My Company	Beverages	LONDON	200	0

## Using Conditions and Masks for the Add Stock button


It is possible to click the **Add Stock** button when there is no data entered in one or all of the data entry fields. This will result in an incomplete record being added to the database and in the long term could mean that the database contains a lot of useless items of data. To prevent this situation, it is possible to put conditions on an Action so that the actions are only completed under the correct circumstances.

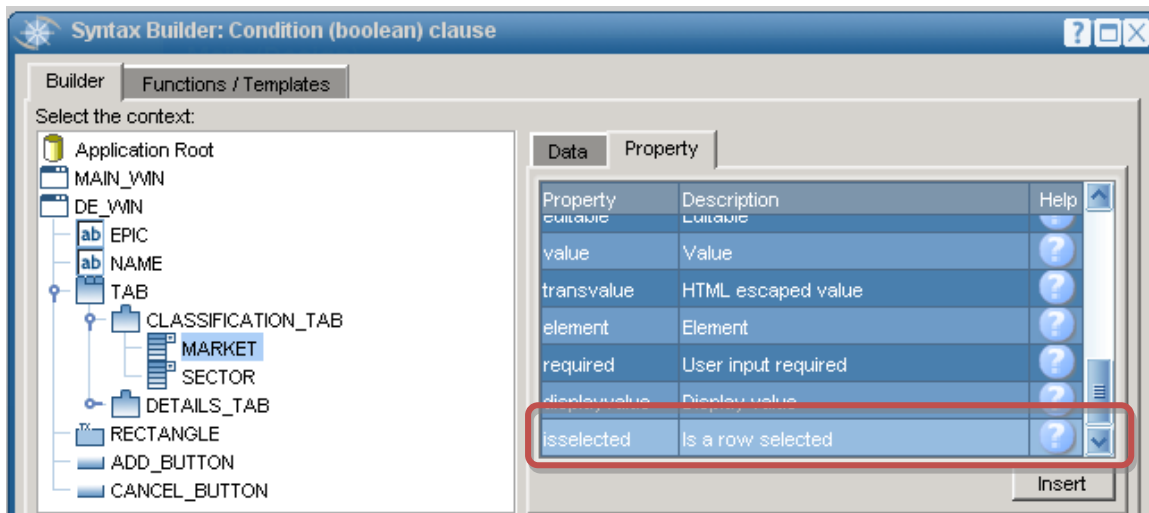
We can apply conditions to force mandatory fields and also verify the content of the field. In this example the EPIC symbol, company name, market and sector are very important. We may also want to check that the user has completed the **Middle Price** and **Percentage Change** fields and that he has entered numbers rather than words.

### Making the selection of a Sector and a Market mandatory to add a stock

1. If not already done, launch the **Designer** for the **STOCKS** view.
2. From the **View Explorer** window, expand the **Windows** folder and select the **Add Stock** button (**ADD\_BUTTON**) under the **DE\_WIN** window.
3. If the **Properties** window is not displayed, click on the **View Properties** icon .
4. From the **Properties** window, click on the **Events** tab.
5. Expand the **Click** action by double-clicking on it.
6. Select the **Actions** node to view its **Parameters**.

Notice that we can set the following parameters:

- **Description:** text entered here will replace **Actions** in the tree.
  - **Condition:** this condition must be fulfilled before any of the actions are performed. We can use the Syntax Builder to help with this.
  - **Fail Message:** will be displayed in the client if the condition fails.
  - **Window:** the window to use as the source for the actions. The Default setting is set to current window.
  - **Context:** the data context for the actions. Generally a control.
7. In the **Description** parameter, enter **Add Stock**. The node **Actions** should now display **Add Stock**.
  8. In the **Condition** field, press the ellipsis button  to show the Syntax Builder.
  9. In the **Select the context** tree, expand the **DE\_WIN** window and the **TAB** node under it.
  10. Expand the **CLASSIFICATION\_TAB** and select the **MARKET** select control.
  11. Click on the **Property** tab on the right of the tree view and scroll-down to select the **isselected** property:



12. Press the **Insert** button.  
 $\${MARKET.isselected}$  will appear in the **Statement** box.
13. Modify this so that the statement reads:  $\${MARKET.isselected}='Y'$ .
14. Press **OK** . Notice that the **Add Stock** event now has a query next to the name to indicate that a condition is set:



15. Now return to the **Condition** parameter and press the ellipsis button to show the Syntax Builder again.
16. Append a similar condition for the **SECTOR** select control, either by typing it in the statement box or by using the Syntax Builder.
17. Adjust the condition so that it reads:  $(\${MARKET.isselected}='Y')$  and  $(\${SECTOR.isselected}='Y')$
18. In the **Fail Message** parameter, enter the following text: **You must select a Market and Sector.**
19. Look at the effects of the changes in Test mode by clicking on the **Run** button :
  - a. Press the **Add** button on the **Main** window, the **Data Entry** window should be invoked.
  - b. Press the **Add Stock** button without selecting a Market or a Sector.
  - c. The error message displays:





20. Stop the test mode and save the View by clicking on the **Save** icon .

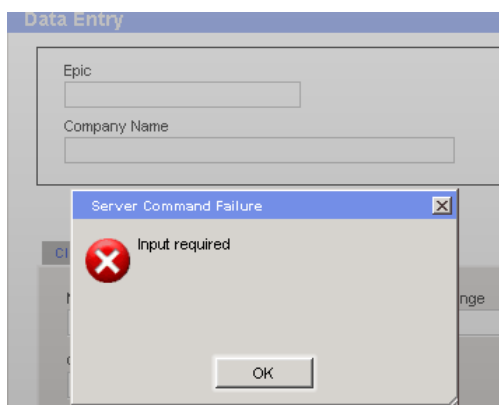
**Note about testing actions:** Where there are multiple conditions, the action will fail at the first condition that is not satisfied and subsequent conditions will be ignored. To test fully the changes, we need systematically to enter data on each control to check that all the conditions are correct.

## Making the EPIC and Company name fields mandatory by setting their control Properties

The Market and Sector select boxes had default values of **Please specify**; we were required to set conditions on the **Server request** action to make the Market and Sector selection mandatory.

For the **EPIC** and **COMPANY** fields there is another way to make them mandatory: As they have no default values, an easy way to validate user input of these fields is provided in the properties for these controls.


1. From the **View explorer** window, expand the **Windows** folder and the **DE\_WIN** window under it.
2. For each control:
  - Display the **Properties** window by selecting the control and clicking on the **View Properties** icon .
  - Expand the **Behavior** properties and set the **User input required** property to **Y**. This prevents the **Server Request** action occurring unless there is data present in this field.
3. Test the application by clicking on the **Run** button :
  - a. Press the **Add** button on the **Main** window, the **Data Entry** window should be invoked.
  - b. Do not enter values in **EPIC** or **Company Name** fields.
  - c. Try to click the **Add Stock** button.
  - d. The message **Input Required** displays and the cursor is positioned on the blank field:



4. Save the View by clicking on the **Save** icon .

## Restricting the Middle Price and Percentage Change fields to accept figure entries only


The text boxes **Middle Price** and **Percentage Change** are numerical fields and should restrict the data entry to figures rather than letters. This is achieved by applying a **Format** attribute to the control.

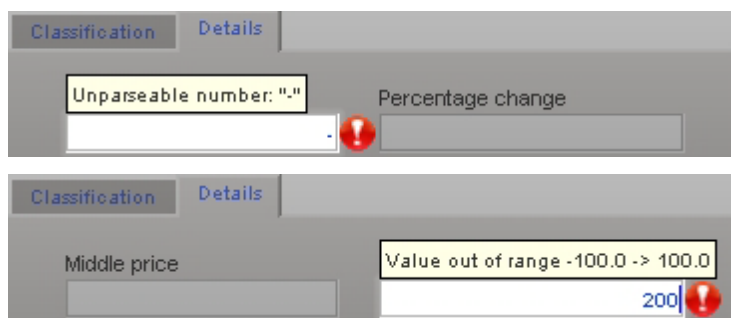
1. From the **View explorer** window, expand the **Windows** folder and the **DE\_WIN** window under it.
2. Expand the **TAB** node and the **DETAILS\_TAB** under it.
3. Display the **Properties** window by selecting the **MID\_PRICE** text control and clicking on the **View Properties** icon .

4. Expand the **Appearance** properties.
5. In the **Format** field enter **N.2**: the user can enter any positive number he wants, the only restriction is that he cannot enter more than two decimals.
6. Select **PERC\_CHANGES** text control.
7. In the **Format** field enter **I[-100,100]**: the user can enter only digits to be input for values in the range -100 to 100 inclusive.

**Please note:**

- *The values in the property drop-down are suggestions to help with syntax*
- *Validation is done on key press or when the control loses focus.*
- *See the Designer Help for a full list of the available masks.*

8. Look at the effects of the changes in Test mode by clicking on the **Run** button :
  - a. Press the **Add** button on the **Main** window, the **Data Entry** window should be invoked.
  - b. Enter values for **Epic** and **Company Name** fields to avoid error messages.
  - c. Click on **Details** tab and try different values in **Middle price** and **Percentage changes** fields to see how the validation works:





9. Save the View by clicking on the **Save** icon .

The controls also have a **Validation Mask** property and a **Validation Message**. This can be used for validating via a regular expression, such as the correct format of an email address or a telephone number. This is covered in more detail in the Designer Help and a techlet is available from the AltioLive Studio.

## Setting a conditional action for the Delete button on the Main window

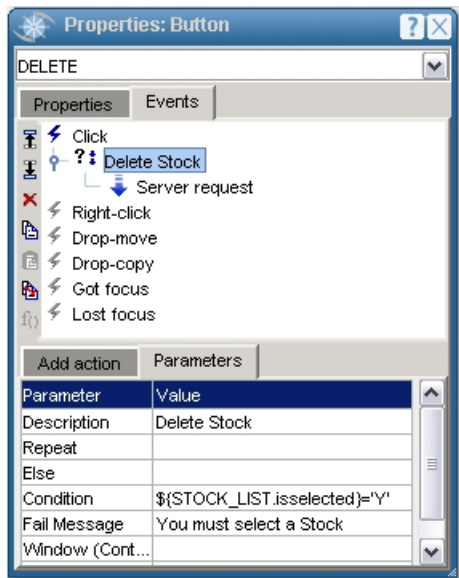
The last Service Function to use is the deletion of Stock elements from the database.



1. From the **View Explorer** window, expand the **Windows** node and the **MAIN\_WIN** under it.
2. Display the **Properties** window by selecting the **DELETE** button and clicking on the **View Properties** icon .
3. Select the **Click** Events and from the **Add action** tab either:
  - a. double-click on the **Server Request** Action or
  - b. Select the **Server Request** Action and press the **Add Action** button .

4. Set the following parameters:

PARAMETERS TAB		
Parameter	Value	Description
Service Function	DELETE_STOCK	The Service Function you have created earlier in this module
Parameter source	WINDOWDATA	Sends the value associated with each control on the window.
Window State	RESUME	The window will be closed when the service is invoked

5. Add a condition to the action so that the service function will not be run unless a row has been selected:



6. Select the **Actions** node to view its **Parameters**.
7. In the **Description** parameter, enter **Delete Stock**. The node **Actions** should now display **Delete Stock**.
8. In the **Condition** field, press the ellipsis button  to show the Syntax Builder.
9. In the **Select the context** tree, expand the **MAIN\_WIN** window and select the **STOCK\_LIST**.
10. Click on the **Property** tab on the right of the tree view and scroll-down to select the **isselected** property.
11. Press the **Insert** button.  
 $\{STOCK\_LIST.isselected\}$  will appear in the **Statement** box.
12. Modify this so that the statement reads:  $\{STOCK\_LIST.isselected\}='Y'$ .
13. In the **Fail Message** parameter, enter the following text: **You must select a Stock**.
14. Look at the effects of the changes in Test mode by clicking on the **Run** button :
  - a. Without selecting any stock, press the **Delete** button on the **Main** window.
  - b. The error message displays:



- c. Press the **OK** button.

- d. Select a **Stock** in the **Stock** list and press the **Delete** button: the stock is deleted from the list.

21. Save the View by clicking on the **Save** icon .

## JavaScript

JavaScript can be called using a **Call Javascript** Action to allow operations to be done on the client outside of the AltioLive application. The AltioLive software accepts return values from JavaScript, and therefore functions such as calculations can be done locally. The return string must be well-formed XML with a DATA element as the parent tag and the return value within another element.

For example:

```
<DATA>
<RESULT VAL="" + resultvar + "">
</DATA>
```

To allow the AltioLive applet to use the JavaScript the MAYSCRIPT attribute must be included in the APPLETTAG in the html file that invokes the Client jar.

There is a techlet on Javascript packaged with Studio.

## Functions

AltioLive allows developers to group a set of Actions in so-called "Functions" that can be accessed across a view. This provides for extensive re-usability of code and saves significant development time.

## Further Exercise: Stocks History Graph

If there is time, try the **Stocks History Graph** practical in the **Further Exercises** document. This practical uses our newly acquired knowledge of Actions to launch a History window. The History window uses a graph control to display the stock HISTORY elements of a stock selected at the main window.

**Next Step: Getting Live Data using Datapools**

AltioLive Developer Training

## **Document Information**

KEYWORDS: SERVICE FUNCTIONS, EVENTS, ACTIONS, CONDITIONAL ACTIONS

**Integra SP – Altio**

Telephone: +44 (0) 20 8528 1045

Internet: [www.altio.com](http://www.altio.com)

**Copyright © 2010 Integra SP**

Copyright in this document is vested in Integra SP. The contents of the document (wholly or in part) must not be reproduced, distributed, used or disclosed without the prior written permission of Integra SP.

Integra recognizes the trademarks or registered trademarks of any third party product or company name referenced in this document at the time of its publication.