Sitecore Forms Module

User Guide

A guide for end users

1 Contents

2 Synopsis

Forms are an editor, enabling you to create Web forms with drag-n-drop. The forms you create can be fully customized to meet any requirement you might have.

You have full control over the location and design of your controls.

2.1 Concept

When you open the forms editor, you initially start with a blank form. On this form you drag the desired controls from the library. Controls include text fields; drop down boxes, check boxes, buttons and so on.

Each control (and the form itself) has events that are fired whenever an action occurs; when page is loaded, when button is pressed, when control is selected and so forth.
On each event, you can attach one or more functions that execute a piece of code when that event occurs.

The number and type of events vary from control to control.

The API for the functions is open, which means that you can develop your own functions (applies to the Pro version only). See the Developer Manual on how to develop functions.

2.1.1 Concept example: Create a mail form

If I was to create a mail form, I could add a text field and a button. In the text field, the user can enter some text that should be sent to me. On the buttons OnClick() event (an event that is fired when the user clicks the button), I could attach a SendEmail function. This function would be given the text field as parameter.
The SendEmail function would then send me the text entered by the user as an email.

3 Starting With Forms

When forms are installed, a new development tool is found:

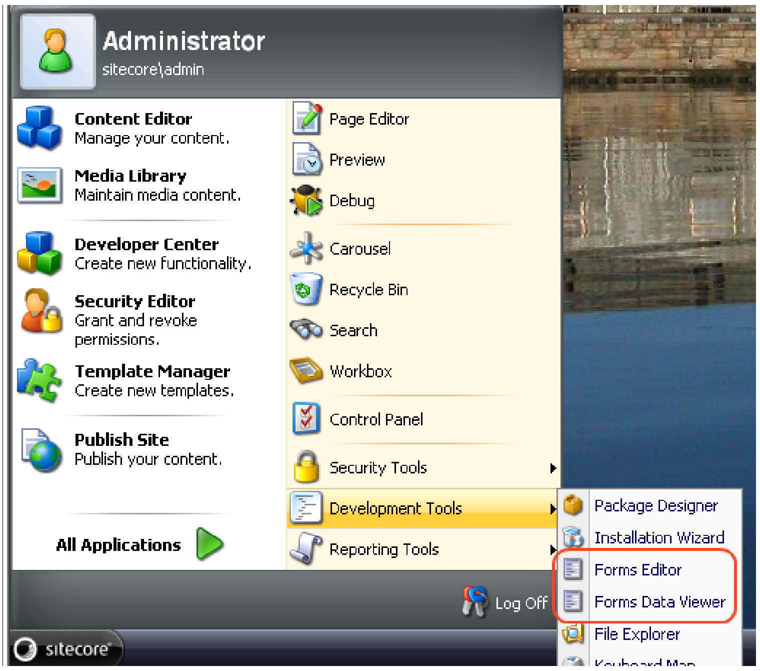


Figure 1 Forms Editor

Clicking the Forms Editor opens an item browser:

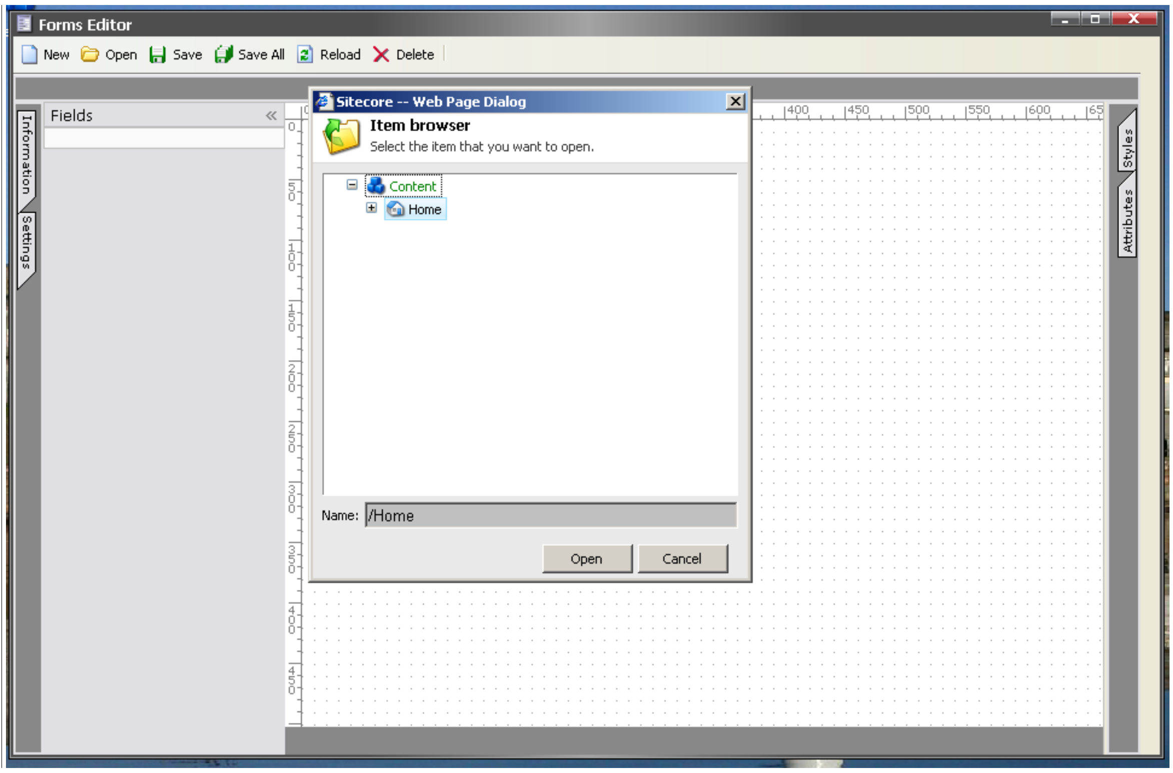


Figure 2 Items Browser

Use this to select the node where you wish to edit a form.

If a form already exists, this form will be opened. Otherwise a new form will be created for you.

The form is created as a sublayout. This sublayout will automatically be placed in on the main layout:

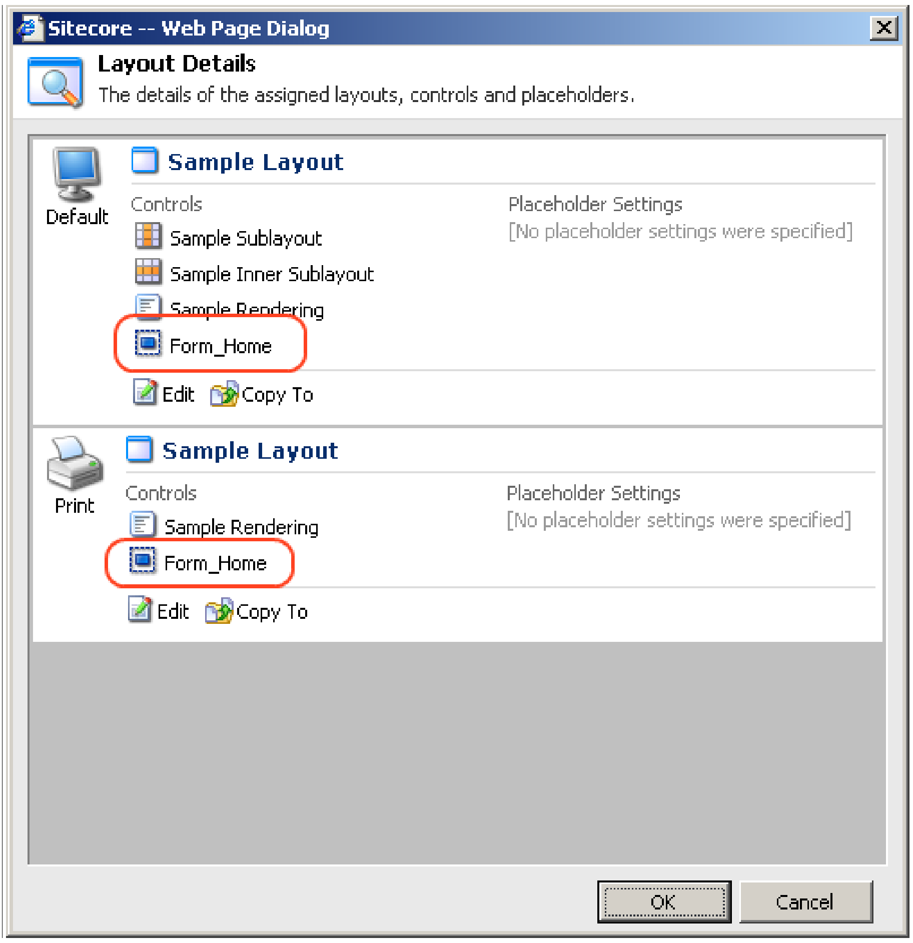
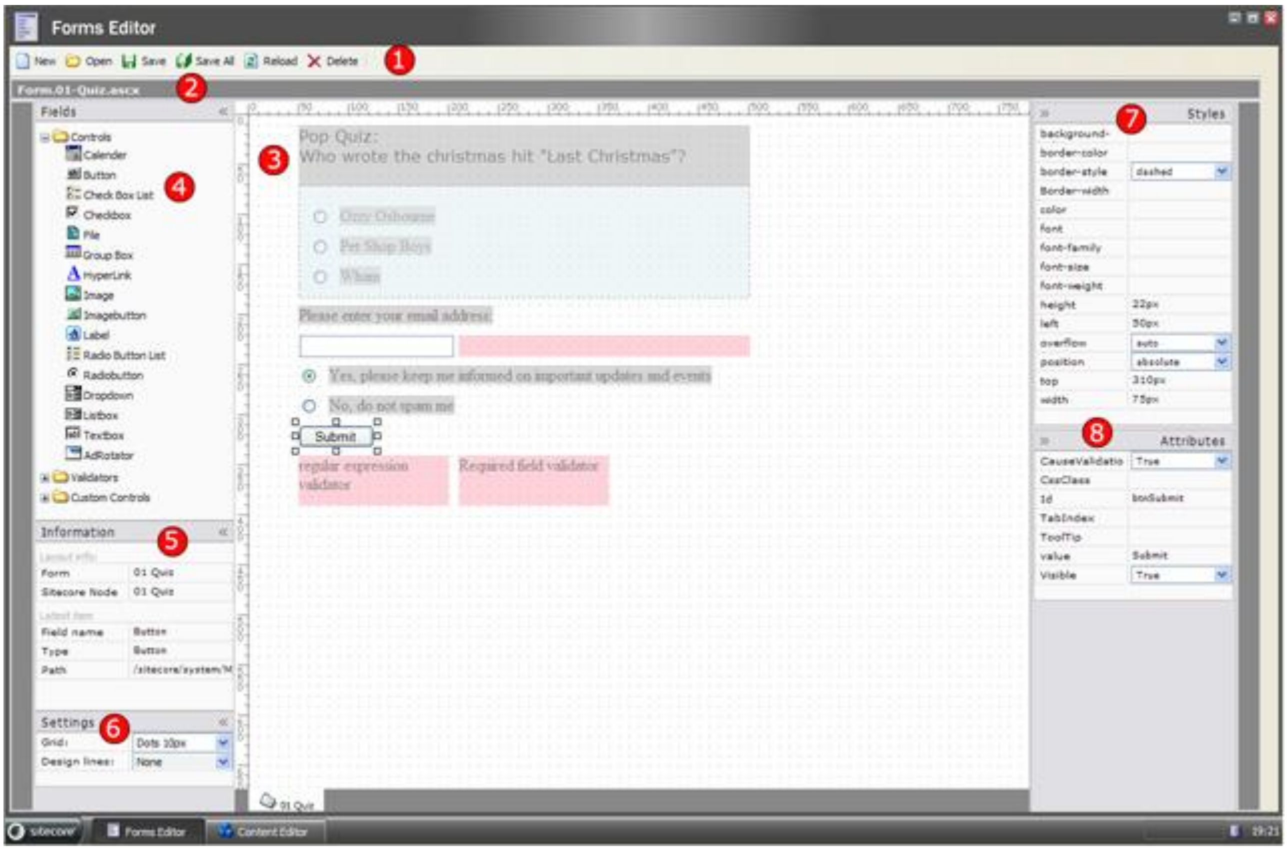


Figure 3 Forms sublayout added to default layout

4 The Forms Editor

The Forms Editor is the heart of the Forms Module. It is in this editor you define the layout, properties and functionality of your form.



Information

Form	01 Quiz
Sitecore Node	01 Quiz
Field name	Button
Type	Button
Path	/sitecore/system/M

Settings

Grid:	Dots 10px
Design lines:	None

Styles

background-	
border-color	
border-style	dashed
Border-width	
color	
font	
font-family	
font-size	
font-weight	
height	22px
left	30px
overflow	auto
position	absolute
top	310px
width	75px

Attributes

CauseValidatio	True
CssClass	
Id	btnSubmit
TabIndex	
ToolTip	
value	Submit
Visible	True

Figure 4 Forms Editor

The editor consists of several panels:

#	Panel	Function
1	Functions	General functionality
2	Caption	Displays the name of the current form.
3	Form	The actual form. You define the form design by dragging controls, validators etc. into this panel.
4	Fields	Tree view of all available fields, validators, etc.
5	Information	Displays general information about the current form
6	Settings	Setup of grid lines and design lines on the form
7	Styles	Information about the available styles settings for the selected control on the form
8	Attributes	Information about the available attributes for the selected control on the form

4.1 Functions

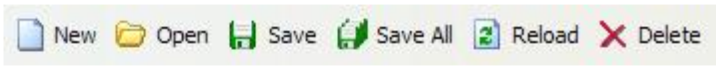


Figure 5 Functions Panel

- New: Creates a sub form on your form. Read about sub forms in chapter 9, Using SubForms.
- Save: Saves your form
- Save All: Saves all forms open
- Reload: Clears the cache and reloads the form
- Delete: Deletes the current form

4.2 Caption

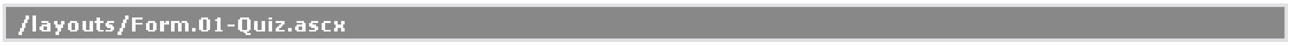

/layouts/Form.01-Quiz.ascx

Figure 6 Caption

The caption shows the path and name of the current form being edited.

4.3 Form

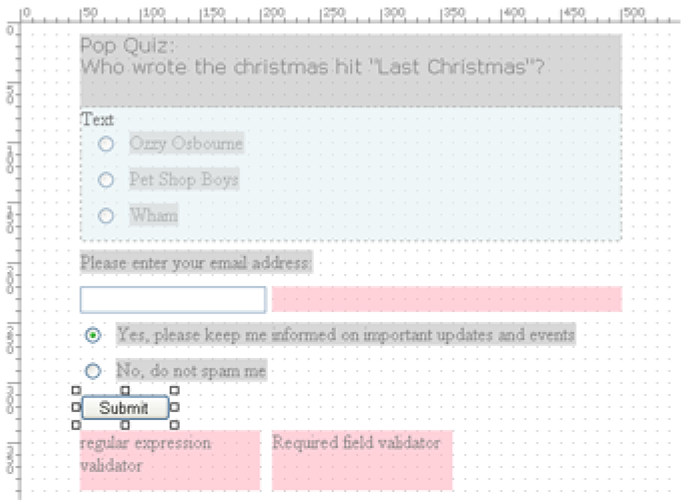

Pop Quiz:
Who wrote the christmas hit "Last Christmas"?
Text
Ozzy Osbourne
Pet Shop Boys
Wham
Please enter your email address:
Yes, please keep me informed on important updates and events
No, do not spam me
Submit
regular expression validator
Required field validator

Figure 7 Form

The form represents the actual form in Sitecore. You drag controls onto this panel. The grid size can be controlled in the "Settings" panel, see 4.6, Settings.

When you have dragged a control onto the form, you can edit its properties and attributes by clicking the control. In Figure 7 Form, I have selected the "Submit" button.

You can drag controls by holding down the left mouse button on a control and drag the control to the desired position.

You can change the control size by holding the left mouse button on one of the square markers and drag the marker to the desired size. You can see whether you have selected the correct marker by looking at the mouse cursor that changes depending on what marker you have selected.

If a control has a text property, you can click once again on the selected control, and you will be allowed to edit the text. Please note that this is not a double-click action. You need to wait until the control is properly selected by the system before you click again:

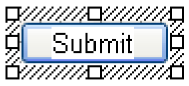


Figure 8 Button control about to have its text property edited

If a control has events, you can double click the selected control to open the function manager, where you attach functions to events.

4.4 Fields

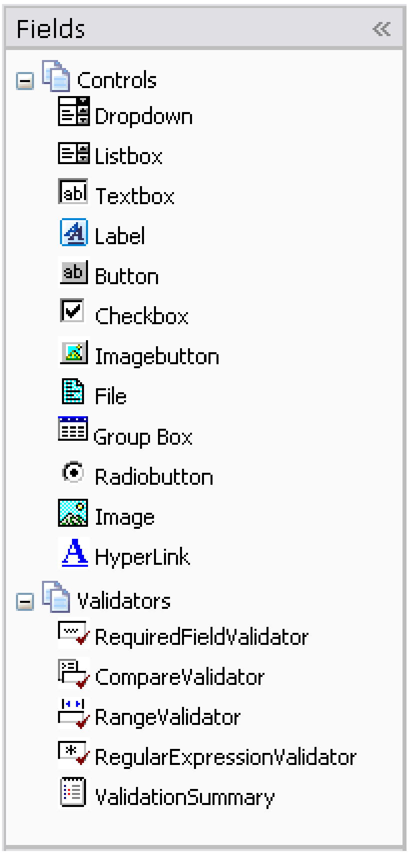
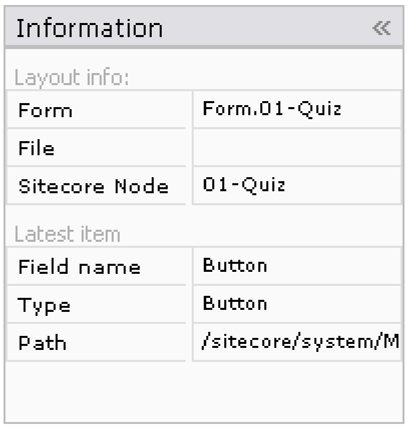


Figure 9 Fields

The fields panel holds all controls, validators, etc. The number of available controls varies from installation to installation, as it is possible to develop new controls, as well as it is possible to hide controls.

You use the fields by selecting the ICON of the control and dragging the control onto the form.

4.5 Information



Information

Layout info:

Form	Form.01-Quiz
File	
Sitecore Node	01-Quiz

Latest item

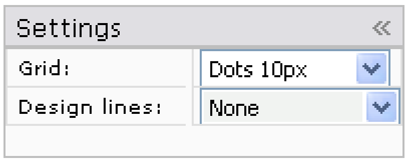
Field name	Button
Type	Button
Path	/sitecore/system/M

Figure 10 Information

The information panel displays information about the current form:

- Form: Name of the form, and equivalent to the name of the sublayout which holds the form.
- Sitecore Node: The name of the Sitecore node which holds the form you are editing.
- Field name: The name of the control you have selected last.
- Type: The type of the control you have selected last.
- Path: The Sitecore path to the definition of the control you have selected last.

4.6 Settings



Settings

Grid:	Dots 10px
Design lines:	None

Figure 11 Settings

Use the settings panel to setup what grid and design lines you would like to have on your form.

Grid lines are lines that your form snaps to.

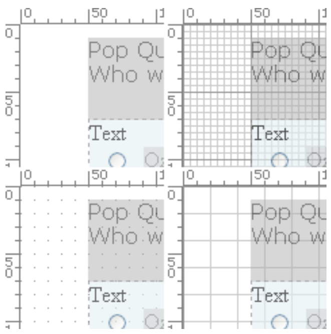
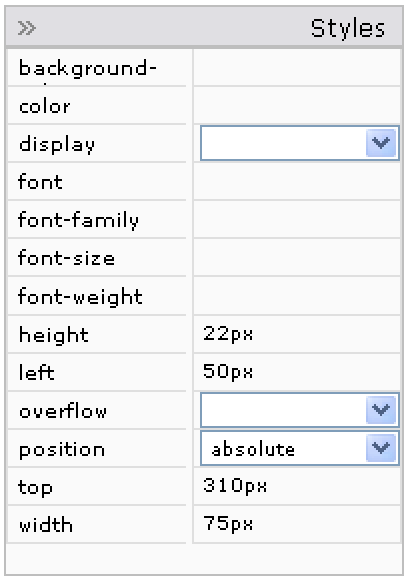


Figure 12 Different grid lines

Read more about design lines in chapter 7, Using Design Lines.

4.7 Styles


Styles

background-color	
display	
font	
font-family	
font-size	
font-weight	
height	22px
left	50px
overflow	
position	absolute
top	310px
width	75px

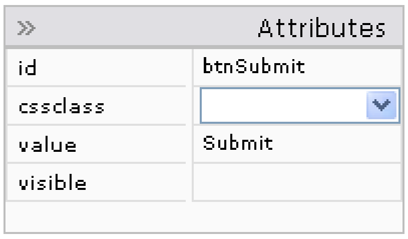
Figure 13 Styles

The Styles panel represents the CSS style settings for the current selected control.
A style setting controls the visuals for a control.
The number of styles available differs from control to control.
Position the mouse over any style name to get the full name as popup.

Please note that the changes will only take effect after you leave the field, for example positioning the cursor in another style field or clicking on a control on the form.

The number of available attributes can be modified. See chapter 8.2, Manipulating Attributes and Styles on Controls.

4.8 Attributes


Attributes

id	btnSubmit
cssclass	
value	Submit
visible	

Figure 14 Attributes

The Attributes panel represents the available attributes for the current selected control.
An attribute controls the behavior of a control.
The number of attributes available differs from control to control.

Position the mouse over any attribute name to get the full name as popup.

Please note that the changes will only take effect after you leave the field, for example positioning the cursor in another style field or clicking on a control on the form.

The number of available attributes can be modified. See chapter 8.2, Manipulating Attributes and Styles on Controls.

5 Assigning Functions to Events

Double click a control to open the function manager:

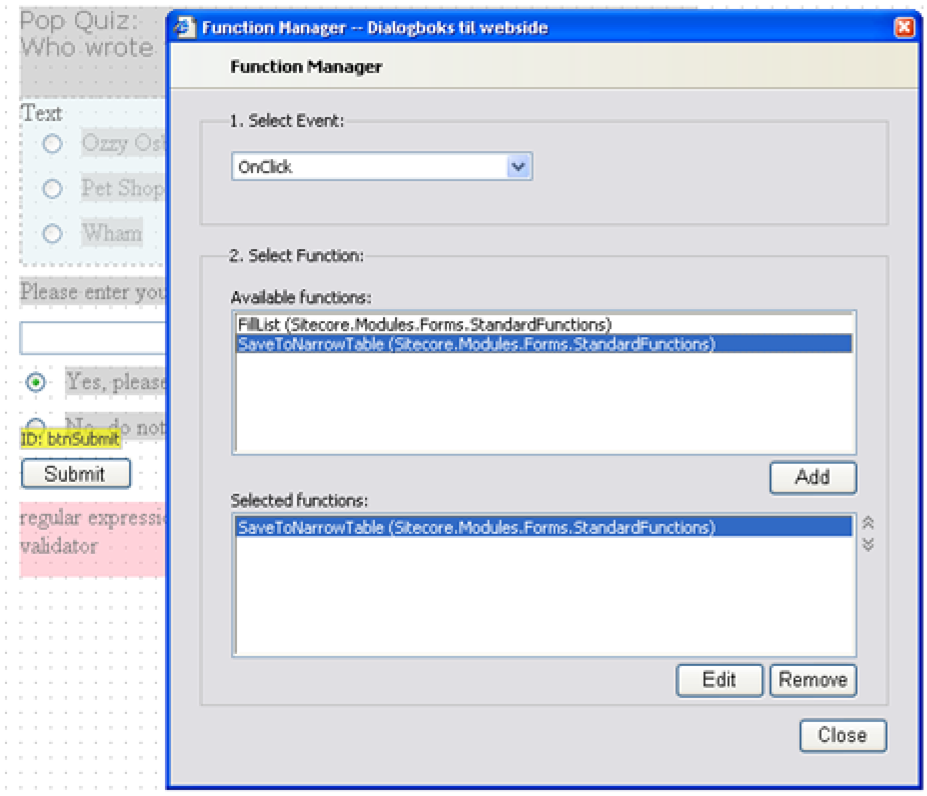
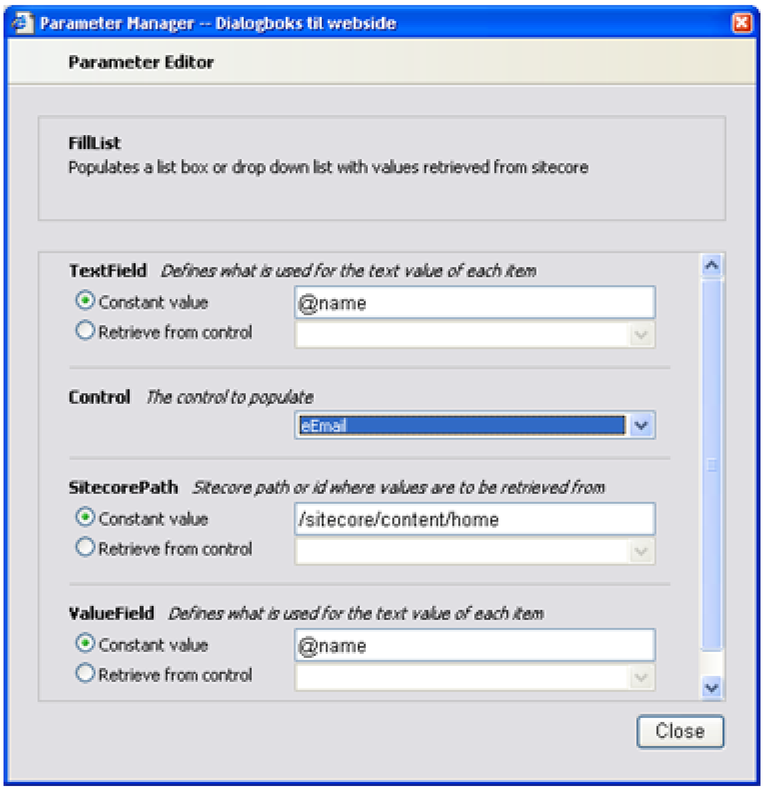


Figure 15 Function Manager

The Function Manager allows you to attach one or more functions to all the events that the selected control has.

Select the event you wish to assign functions to and use the “Add” button to assign the functions. Functions will be executed in the sequence they are represented in the list. Use the arrow buttons to change the sequence.

To edit the parameters for the function, select the function from the “Selected functions” list and press “Edit”. The Parameter Editor window will open:



Parameter Manager -- Dialogboks til webside

Parameter Editor

FillList
Populates a list box or drop down list with values retrieved from sitecore

TextField *Defines what is used for the text value of each item*
(●) Constant value: @name
( ) Retrieve from control

Control *The control to populate*
eEmail

SitecorePath *Sitecore path or id where values are to be retrieved from*
(●) Constant value: /sitecore/content/home
( ) Retrieve from control

ValueField *Defines what is used for the text value of each item*
(●) Constant value: @name
( ) Retrieve from control

Figure 16 Parameter Editor

All available parameters for the selected function will appear in the list in the middle of the window.

For some parameters you can assign either a constant value (a text string), or you can select a control to get the value from. For other parameters, only a control is a valid parameter.

Check the "Constant value" box to use a constant value. Check the "Retrieve from control" to get the value from a control.

Take this situation as an example: I have a list that needs to be filled with data from the Sitecore tree. I would add a listbox on my form, and on the OnClick() event, I would add the FillList function. The function takes several parameters as input: The Sitecore path, the Sitecore node, the control to populate and son on.

When the user opens the web form, the list box is populates with the nodes from Sitecore.

6 Controls

The following controls are available by default:

	Name	Description
	Calendar	A calendar control
	Button	A normal button
	CheckBoxList	A list of checkboxes. The values of the checked buttons are pipe (|) separated.
	Checkbox	A normal checkbox
	File	A file selection text box. It is a text box and a "browse" button
	Group box	A group box. Used to group controls together
	Hyperlink	A hyperlink label
	Image	A normal image
	ImageButton	An image that has the same events as a button
	Label	A normal label
	RadioButtonList	A list of radio buttons. Only one button can be pressed at any time
	RadioButton	A normal radio button. Use the "Groupname" attribute to group radio buttons
	Dropdown	A normal dropdown box
	Listbox	A list of strings. The values of the selected items are pipe (|) separated.
	Textbox	A normal text box
	AdRotator	An AdRotator

6.1 FieldValidators

A field validator is a control that is attached to a control. It is capable of validating that the input of the field is correct.

Please be aware that if you do not have client side validation enabled, all standard functions will fire even if a validator fails.

The client side validation is performed by the WebUIValidation.js file that is supplied by Microsoft.NET and is by default located at:

/WINDOWS/Microsoft.NET/Framework/v1.1.4322/ASP.NETClientFiles/WebUIValidation.js

Client side validation is usually enabled by default.
To make sure that you have enabled client side validation you can:

- Run the following command: aspnet_regiis –c
- Or copy the ASP.NETClientFiles folder to the root of the web site.

Please contact your web site administrator if you do not know whether client side validation is enabled or not.

Field validators are not visible unless the entered input is invalid.

The following field validators are available:

	Name	Description
	Required	Validates that the value of the control is not empty
	Compare	Compares the contents of 2 controls and acts upon the "operator" attribute. The following operators are available: • Equal • GreaterThan • GreaterThanEqual • LessThan • LessThanEqual • NotEqual • DataTypeCheck Also use the "type" attribute to define the correct type for the comparison.
	Range	Validates that the value of the control is within a certain range.
	RegularExpression	The most flexible validator. Validates the input of a control with a regular expression. For example, to validate that the control contains a valid email address, use this expression: (\w[-._\w]*\w@\w[-._\w]*\w\.\w{2,3})
	Summary	Use this control to summarize all validations

6.1.1 How to use a Field Validator

To use a field validator, select the validator you desire, and drop it on your form:

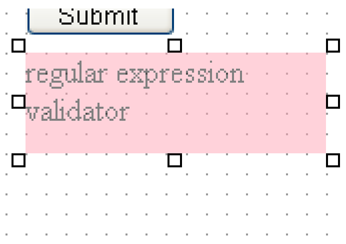
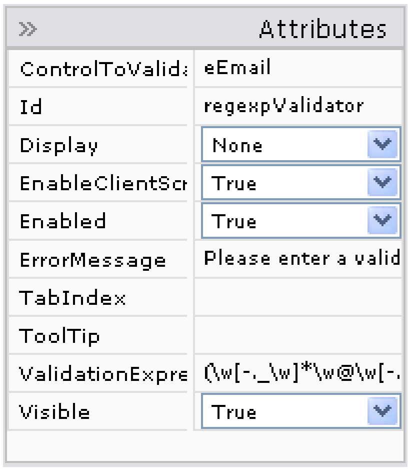


Figure 17 Validator dropped on a form

Use the ControlToValidate attribute to assign the validator to a control:



ControlToValid:	eEmail
Id	regexpValidator
Display	None
EnableClientScr	True
Enabled	True
ErrorMessage	Please enter a valid
TabIndex	
ToolTip	
ValidationExpre	(\w[-._\w]*\w@\w[-.
Visible	True

Figure 18 Validator Attributes

In Figure 18, the validator validates the value of eEmail.

The ErrorMessage attribute defines what text to display when an error occurs. If the ErrorMessage attribute is empty, the validator displays the text written in the text field.

The validator in Figure 17, would display "regular expression validator", if I did not use a SummaryValidator. When using a SummaryValidator, make sure that the "Display" attribute is set to "None".

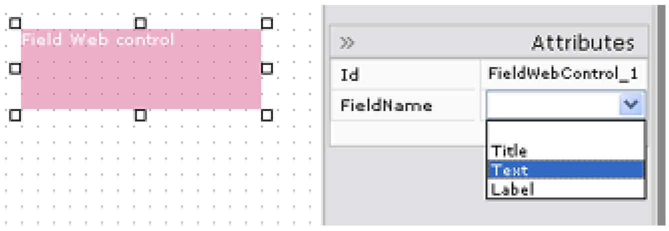
6.2 Custom controls

Custom controls are non standard controls.

	Name	Description
	PageControl	A page control used with sub forms. See chapter 9, Using SubForms.
	Xsl Rendering	A control used to render a XSL (default sitecore rendering)
	Field Web Control	A control used to render the contents of a Sitecore field. Especially good for multilizing labels.

6.2.1 Using a Field Web Control

A field web control is a custom control that is able of rendering the contents of a Sitecore field.

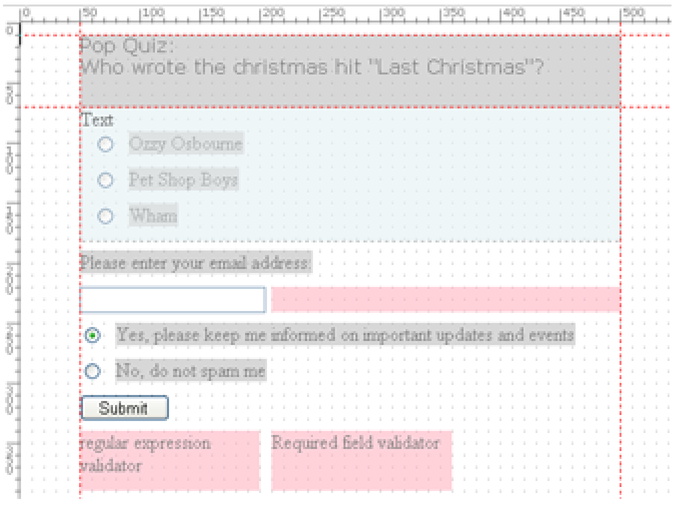


Id	FieldWebControl_1
FieldName	

Figure 19 Field Web Control

The FieldName attribute defines all available fields for the current node. If you assign the FieldName to a field, the Field Web Control will print the contents of that field in the current language.

7 Using Design Lines


Pop Quiz:
Who wrote the christmas hit "Last Christmas"?

Text
- Ozzy Osbourne
- Pet Shop Boys
- Wham

Please enter your email address:

- [✓ Yes, please keep me informed on important updates and events]
- No, do not spam me

Submit

regular expression validator

Required field validator

Figure 20 Design Lines

Design lines are guiding horizontal and vertical lines that can be enabled in the editor. The lines can help you place the controls according to your website design rules.

The guidelines give you visual aid when positioning your controls. They can help you keeping the same left margin, the same width of all controls, placing the header the same place and so on.

Please note that your controls will not snap to the design lines. They are merely meant to be visual guidelines.

The administrator of the web site can define the design lines.

8 Forms Settings

This chapter applies to website administrators or the person responsible of setting up the forms module.

All Forms settings are located under /system/modules/forms.

8.1 Defining Design Lines

You define design lines under /system/modules/forms/editor settings/design lines:

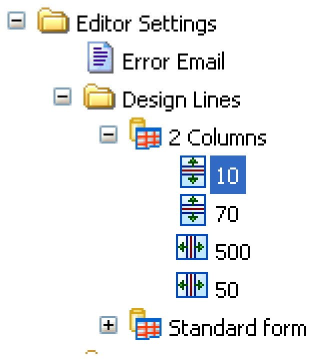


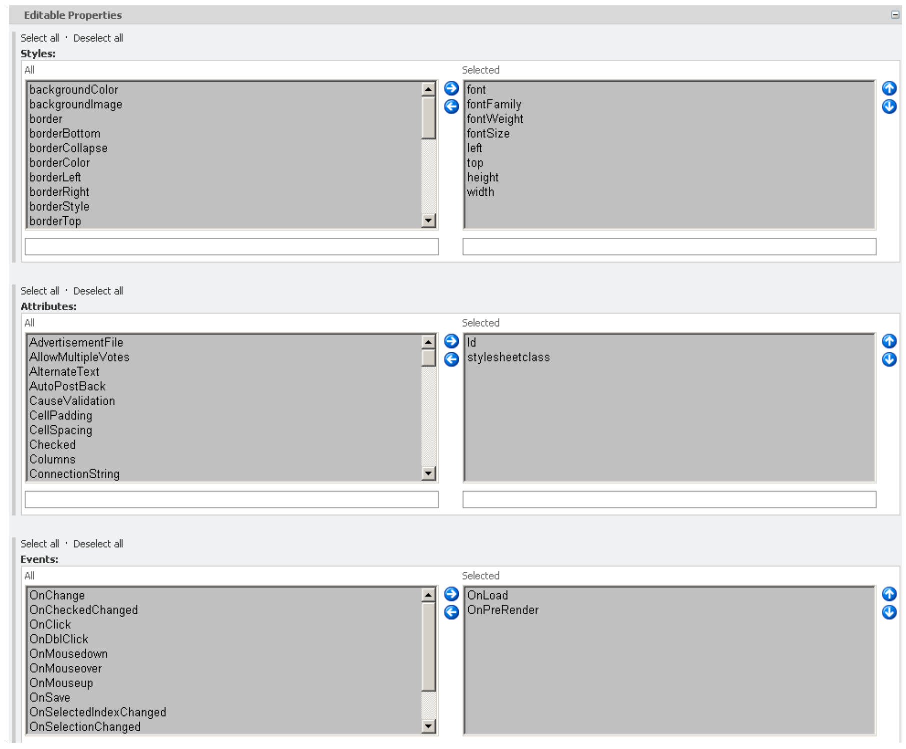
Figure 21 Design Lines

Add new design line settings under "Design Lines". Add the horizontal and vertical lines under your design line setting.

8.2 Manipulating Attributes and Styles on Controls

The standard package includes the most common style and attributes settings for each control. But if you wish, you can add or remove any attribute or setting you may wish.

All controls are defined under "/system/modules/forms/field definitions". Choose a control and go to the "editable properties" page to manipulate the attributes and styles:



Editable Properties

Select all · Deselect all
Styles:

All:
backgroundColor
backgroundImage
border
borderBottom
borderCollapse
borderColor
borderLeft
borderRight
borderStyle
borderTop

Selected:
font
fontFamily
fontWeight
fontSize
left
top
height
width

Select all · Deselect all
Attributes:

All:
AdvertisementFile
AllowMultipleVotes
AlternateText
AutoPostBack
CauseValidation
CellPadding
CellSpacing
Checked
Columns
ConnectionString

Selected:
Id
stylesheetclass

Select all · Deselect all
Events:

All:
OnChange
OnCheckedChanged
OnClick
OnDblClick
OnMousedown
OnMouseover
OnMouseup
OnSave
OnSelectedIndexChanged
OnSelectionChanged

Selected:
OnLoad
OnPreRender

Figure 22 Styles and Attributes

Styles and Attributes are defined in "/system/modules/forms/field styles" and "/system/modules/forms/field attributes":

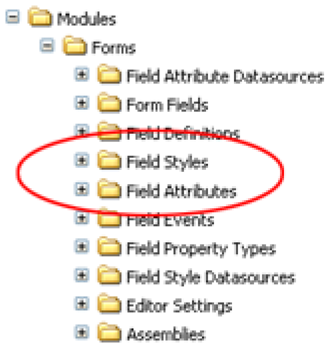


Figure 23 Styles and Attributes definitions

Please do not remove the "Id" attribute. Also, do not change events unless you know exactly what you are doing.

9 Using SubForms

In forms it is possible to create sub forms, i.e. multiple forms on the same page.
If you add a sub form, this form will be drawn together with the main form (the first form on the page).
The intention of sub forms is to create wizard style forms, where you can browse through the pages in the wizard.

Each sub form is completely independent of other sub forms. You cannot see the controls of other sub forms. If sub forms need to exchange data, you can use the SessionSave function.

9.1 Concept

To properly create a wizard style form, you must add a **PageControl** to the main form. The page control has a PlaceholderKey attribute that must match the placeholder of the forms that the PageControl will display.

You design the wizard as follows:

On the first form, drop a PageControl. Set the PlaceholderKey to “subforms” (just an example, you can give any name you like).

Create a new sub form by clicking ‘New’ in the toolbar. Use the Caption (see chapter 4.2, Caption) to change the placeholder of the sub form to “subforms”.

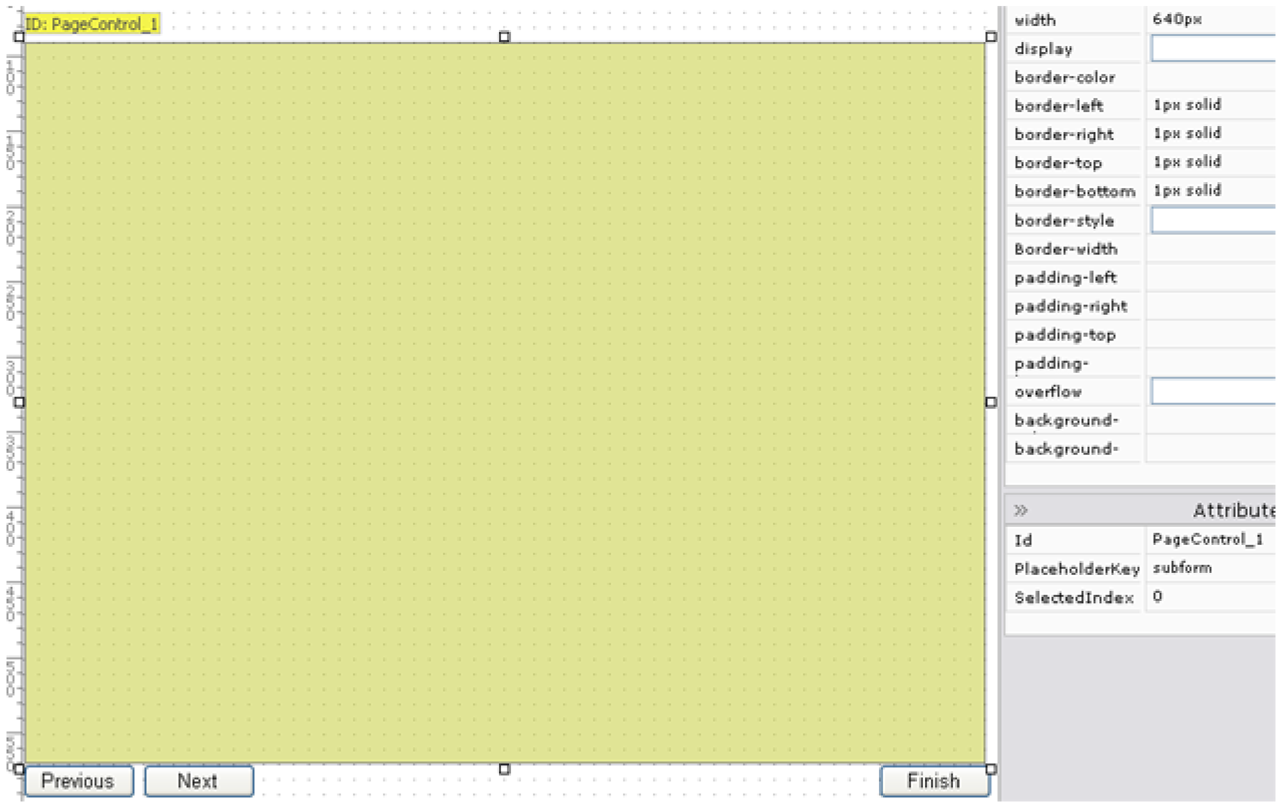
Add as many sub forms as you need, and remember to change the placeholder to “subforms”.

The PageControl will display the sub forms one by one, defined by the SelectedIndex attribute, where 0 is the first sub form and so on.

Use the function PageControlChangePage to switch between the sub forms.

9.2 Example: Create a 3 page Wizard

In this example I have created a main form, and added a PageControl to the main form. The placeholderkey is named “subform”. I have also added 3 button, “Previous”, “Next” and “Finish”:



width	640px
display	
border-color	
border-left	1px solid
border-right	1px solid
border-top	1px solid
border-bottom	1px solid
border-style	
Border-width	
padding-left	
padding-right	
padding-top	
padding-	
overflow	
background-	
background-	

» Attribute

Id	PageControl_1
PlaceholderKey	subform
SelectedIndex	0

Figure 24 My PageControl

The “Previous” button calls the PageControlChangePage with the “Previous” parameter, and the “Next” button calls the PageControlChangePage with the “Next” parameter. The “Finish” button calls the “SendMail” function.

Furthermore, I have added 3 subforms, and given each the placeholder id “subform”. The main form and the sub forms are presented in the tab control at the bottom:

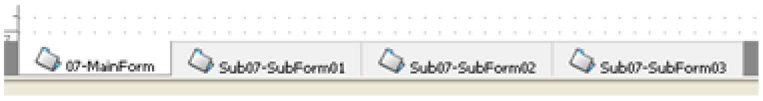
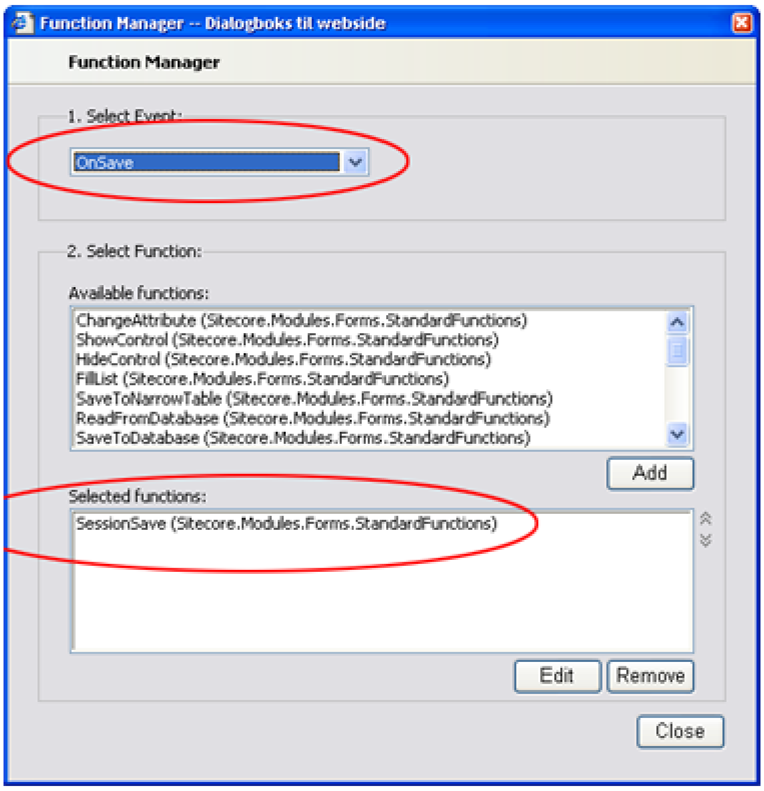


Figure 25 Tab Control

To rearrange the sequence of the forms, drag the ICON of the sub form to the desired destination.

On the first 2 sub forms, I have added the SessionSave function to the OnSave event:


Function Manager -- Dialogboks til webside

Function Manager

1. Select Event:

OnSave

2. Select Function:

Available functions:

ChangeAttribute (Sitecore.Modules.Forms.StandardFunctions)
ShowControl (Sitecore.Modules.Forms.StandardFunctions)
HideControl (Sitecore.Modules.Forms.StandardFunctions)
FillList (Sitecore.Modules.Forms.StandardFunctions)
SaveToNarrowTable (Sitecore.Modules.Forms.StandardFunctions)
ReadFromDatabase (Sitecore.Modules.Forms.StandardFunctions)
SaveToDatabase (Sitecore.Modules.Forms.StandardFunctions)

Selected functions:

SessionSave (Sitecore.Modules.Forms.StandardFunctions)

Figure 26 SessionSave attached to OnSave

The SessionSave function saves the values of all controls in session variables. This data can later be retrieved by SessionRead() or by a special syntax in SendMail() and SaveToDatabase().

The last sub form is a “thank you for entering data” form, that does nothing but telling the user that the wizard is finished, and ensuring that the OnSave() event is called on the second sub form.

9.2.1 Retrieving the session variables with SendMail

If you use SendMail() or SaveToDatebase() functions, you can retrieve the session variables by adding a “session:” in front of the control ID.

Take this example: In the first subform, I have a field called Field1. This field is saved to a session variable called Field1. To retrieve the value, write the following in the mail body or header field:

{session:Field1}

This string will be replaced with the value of the session variable. The syntax is case sensitive, so be careful that you write “session” in all small caps, and the session variable correct.

10 Advanced Functions

10.1 Difference between display=none and visible=false

A lot of controls has a style setting called “display” which can be set to “none”, “block” and “inline”, and a attribute called “Visible” that can be true or false.

It may look as if they are doing the same thing: hiding or displaying the control, but behind the scenes the result is much different.

If you set the display=none style, the control is actually on the form, but it is not displayed.
If you set the visible attribute to false, the control is removed from the form, and you cannot access the control at runtime.

10.2 How to automatically size a dropdown control

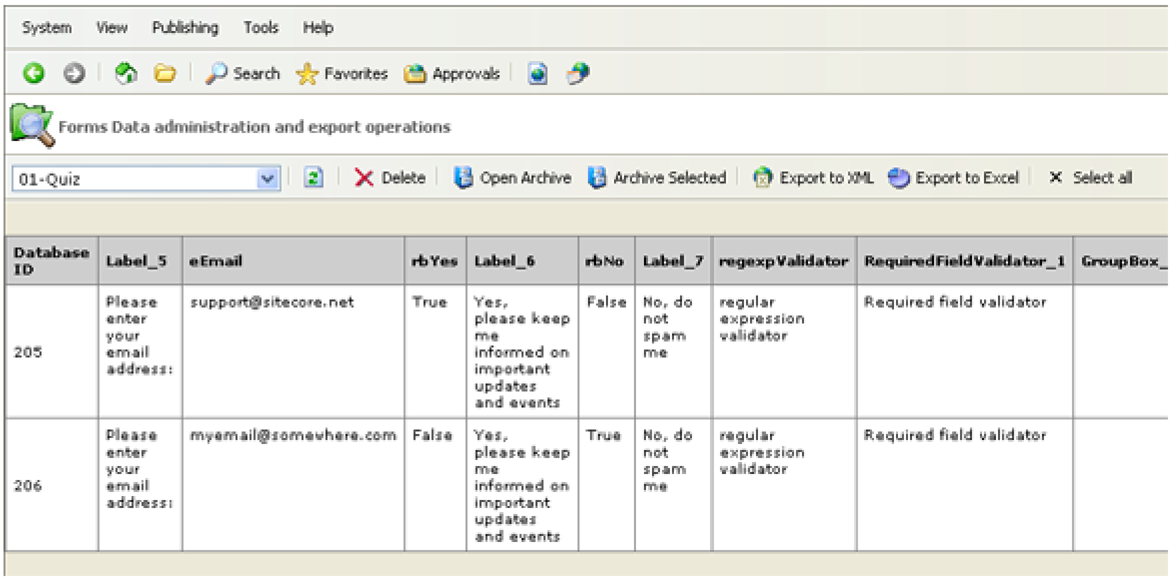
If you wish to have a dropdown control automatically size to the text inside the dropdown, then you should not set the width style.

If you set the width style on the drop down box, it will always have that size.

11 Using the Forms Data Viewer

When using the SaveToNarrowTable function, all data are saved to a value-list table. These data can be viewed with the standard Forms Data Viewer.

Select the viewer from Tools|Forms Data Viewer:

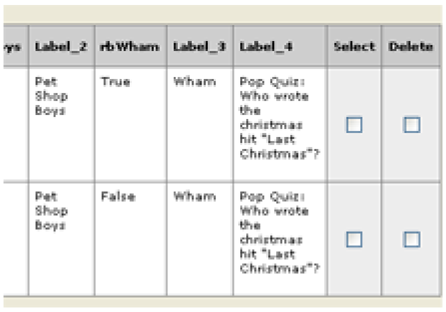


Database ID	Label_5	eEmail	rbYes	Label_6	rbNo	Label_7	regexpValidator	RequiredFieldValidator_1	GroupBox_
205	Please enter your email address:	support@sitecore.net	True	Yes, please keep me informed on important updates and events	False	No, do not spam me	regular expression validator	Required field validator	
206	Please enter your email address:	myemail@somewhere.com	False	Yes, please keep me informed on important updates and events	True	No, do not spam me	regular expression validator	Required field validator	

Figure 27 Forms Data Viewer

The data is stored by form. Use the drop down box at the top to select which form you wish to see data for.

All data entered through time is displayed chronologically, with the oldest at the top. Each value for each control on the form is displayed in their own column. At the far right of each column is a select and delete check box:



ys	Label_2	rbWham	Label_3	Label_4	Select	Delete
	Pet Shop Boys	True	Wham	Pop Quiz: Who wrote the christmas hit "Last Christmas"?	☐	☐
	Pet Shop Boys	False	Wham	Pop Quiz: Who wrote the christmas hit "Last Christmas"?	☐	☐

Figure 28 Select and Delete column

To delete data, check the "Delete" box and press the "Delete" button.
To archive data, check the "Select" box and press the "Archive Selected" button.
To export data to XML check the "Select" box and press the "Export to XML" button.
To export data to Excel check the "Select" box and press the "Export to Excel" button.
To view archived data, press the "Open Archive" button. You have the same options for archived data as for normal data.

12 Example: Create a Mail form

In this example, I will create a complete form where users of the website can order brochures. The order is sent to the administrator of the brochures in my fictive company.

My fictive company is a computer accessories manufacturer. On their website you can order brochures on their different products. The form looks like this:

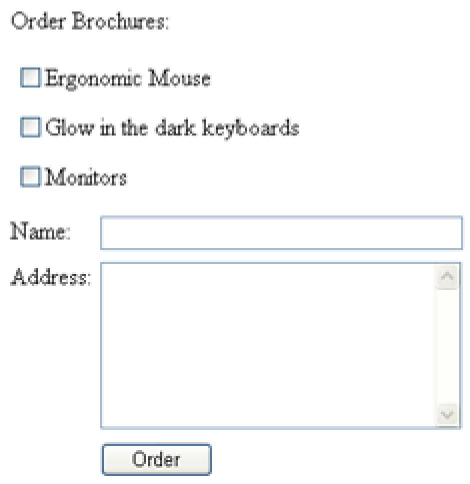


Figure 29 Order Brochures

For this for to work, I need the following items:

- A page in Sitecore that will show the form
- A node in Sitecore where I can write the text that should go into the mail
- A list of nodes with the names of the brochures that can be ordered

12.1 Create the nodes

In this example I have compacted my nodes a bit. Usually you would store the different nodes elsewhere, but for the sake of explaining, I have made the following tree:

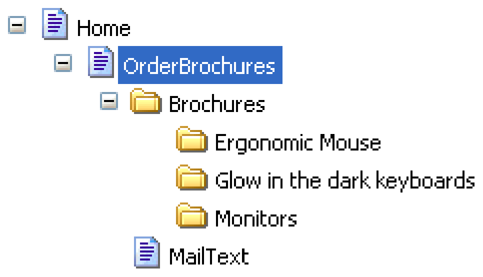


Figure 30 Mail form nodes

The "OrderBrochures" is the node that will show the form.
The "Brochures" node with underlying nodes is my list of brochures that I later use to fill a checkboxlist. Usually you wouldn't put such master data under a page node, but in a "global" section on your website.
Finally, the "MailText" node contains the text in the mail. This node would usually also be located under a "global" section, so it wouldn't be confused with a page on the website.

12.2 Create the form

I select the “OrderBrochures” node, and press the “Forms Editor” button. I also name my form “OrderBrochures”.

Remember, that when you are finished with your form, you must add a layout for your form. If you don’t do that, the form will not be displayed. See chapter 3, Starting With Forms for more information.

On the form, I add a checkboxlist, 3 edit boxes and a button, along with the appropriate labels:

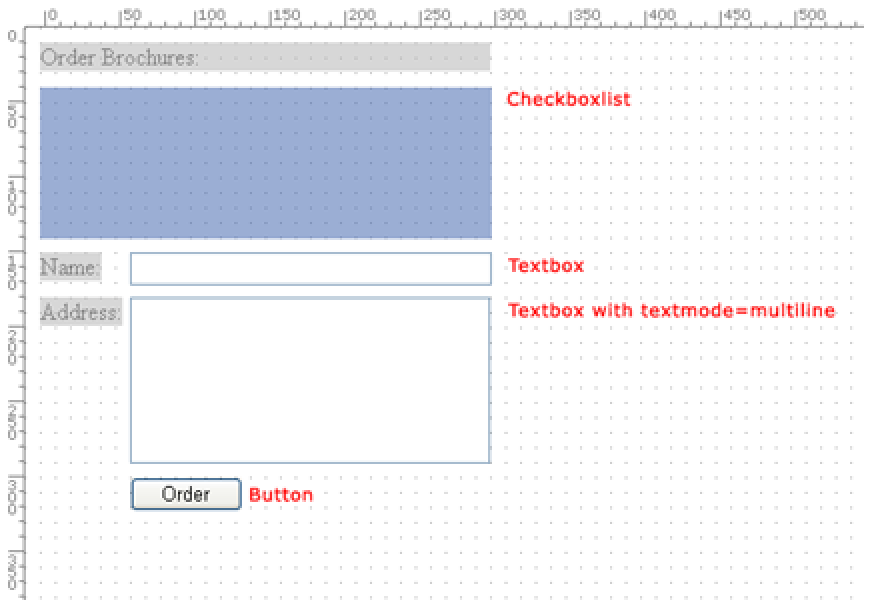
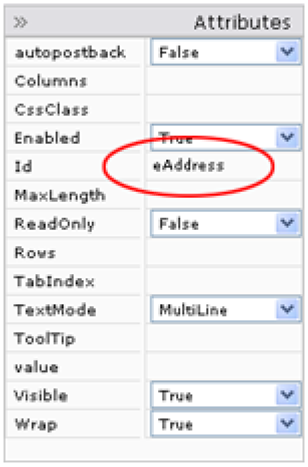


Figure 31 Mail form

It is important that you give your controls a good ID. If you don’t do that, it will be more than difficult later in the process to find the right control.



»	Attributes
autopostback	False
Columns	
CssClass	
Enabled	True
Id	eAddress
MaxLength	
ReadOnly	False
Rows	
TabIndex	
TextMode	MultiLine
ToolTip	
value	
Visible	True
Wrap	True

Figure 32 the address field is named "eAddress"

12.3 Assigning functions to events

Right now, the form doesn't do anything. I must assign some functions to the events. First I double click the checkboxlist.
When the form is loaded, I wish to have the checkboxlist filled with the brochures.

I therefore select the OnLoad() event, and choose the "FillList" function from the list of available functions.

I give the "FillList" function the following values:

- Textfield: "@name" (the checkboxlist will display the name of the node)
- Control: "cblBrochures" (I fill the checkboxlist control with the nodes)
- SitecorePath: "/sitecore/content/home/orderbrochures/brochures" (my data comes from the children of that node)
- ValueField: "@name" (the checkboxlist will have the name of the nodes as value)

Now, on to the button: When the button is pressed, I would like the form to send me a mail.

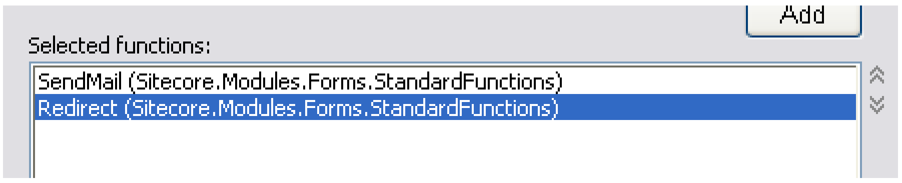
I double click the button, choose the OnClick() event, and assign the SendMail function.

I give the "SendMail" function the following values:

- BodyField: "Text" (the field where the body of the mail is written is named Text)
- From: "me@company.com" (the from email address)
- To: "brochuremanager@company.com" (the email address of the receiver)
- Subject: "A brochure has been ordered" (the title of the mail)
- BodySitecorepath: "/sitecore/content/home/orderbrochures/mailtext" (the node where the body of the mail is written)

Also, when the brochure has been ordered, I wish to redirect to a "Thank you for ordering" page. I do not have such a page in this example, so I redirect to the front page.

Below the "SendMail", I add the "Redirect" function:



Selected functions:
SendMail (Sitecore.Modules.Forms.StandardFunctions)
Redirect (Sitecore.Modules.Forms.StandardFunctions)

Figure 33 2 Functions assigned

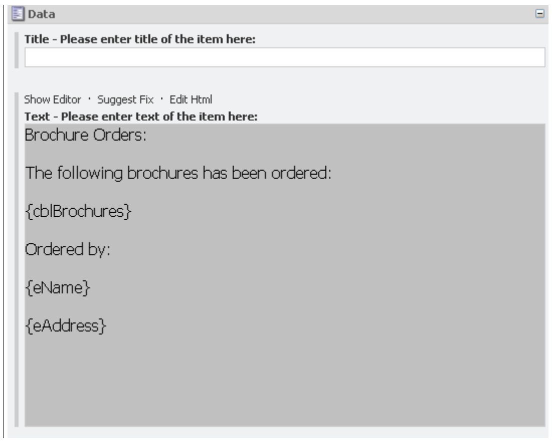
The "Redirect" is given the following values:

- EndResponse: "True" (execution of the current page should terminate)
- Url: "/" (I redirect to the front page. In real life, the Url would probably be more like /company/thankyou.aspx)

12.4 Write the mail text

Now it is time to write the body of the mail message. Now I am glad that I gave my controls some good IDs, because I need to remember them.

I open the "MailText" node, select the "Text" field and write the following text:



Title - Please enter title of the item here:

Text - Please enter text of the item here:

Brochure Orders:

The following brochures has been ordered:

{cblBrochures}

Ordered by:

{eName}

{eAddress}

Figure 34 Mail Text

At runtime, when the web site user presses the "Order" button, the text in the curly brackets will be replaced with the text of those controls with the same ID.

The {cblBrochures} text is replaced with the values selected in the cblBrochures checklistbox. The {eName} is replaced with the value in the eName text box.

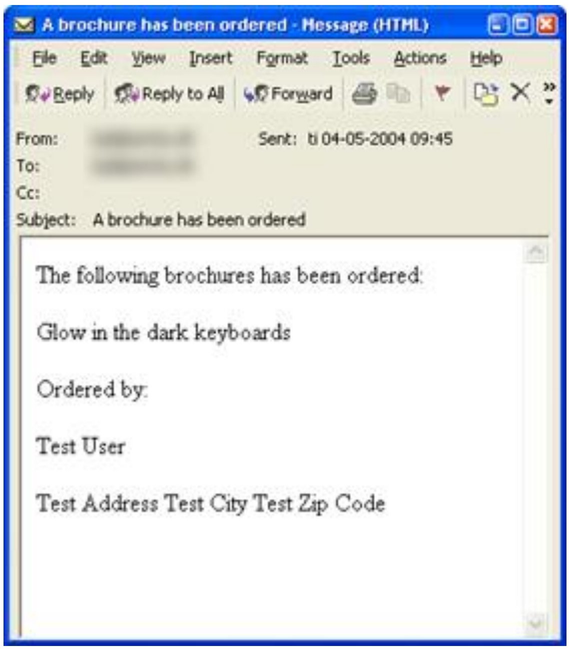
12.5 Time to test the form

It is now time to see if the form actually works. Press "F9" to publish the web site, and open the web page.

Does your page display nothing? You probably forgot to assign a layout to the node. Remember, that when you are finished with your form, you must add a layout for your form (unless you have defined a layout on the template. In that case the layout is automatically added). If you don't do that, the form will not be displayed. See chapter 3, Starting With Forms for more information.

I try to enter some test data. I order the "Glow in the dark keyboards" brochure, and enters some "Test User, Test address…".

The following mail is sent to me:


A brochure has been ordered - Message (HTML)
File Edit View Insert Format Tools Actions Help
Reply Reply to All Forward

From: [redacted] Sent: ti 04-05-2004 09:45
To: [redacted]
Cc:
Subject: A brochure has been ordered

The following brochures has been ordered:

Glow in the dark keyboards

Ordered by:

Test User

Test Address Test City Test Zip Code

Figure 35 Mail is sent to me