
Sitecore Forms Extensions Documentation

Release 1.4.3

Bart Verdonck

Feb 28, 2021

1	Custom Fields	3
1.1	Recaptcha	3
1.2	File Upload	4
1.3	Raw HTML	5
1.4	Hidden Field	6
2	Send Mail	7
2.1	Create an automated message in Email Experience Manager	7
2.2	How to use the Send Mail action	11
2.3	Attachments	15
3	Form Bindings (Prefilling)	17
3.1	How to configure a field	17
3.2	Supported Form Elements	18
3.3	Example	19
4	Value Provider Conditions	23
4.1	How to create a set of rules	23
4.2	How to use a set of rules	25
5	In place “Thank you” message	27
5.1	How	27
5.2	Result	29
6	Installation and Setup	31
6.1	Download	31
6.2	Install	31
6.3	Configuration	32
7	Custom Bindings	39
7.1	Add a custom binding source	39
7.2	Configure preferred email, address and phonenumber	41
8	Configure Date Timespan Validator	43

[Click here](#) to read the documentation on Sitecore Forms Extensions 2.1 for Sitecore 9.1.

Welcome, happy to see your are (considering) using the Sitecore Forms Extensions module to give your Sitecore 9 forms a feature boost!

With this module you will be able to prefill forms, get a file upload field, get better integration with EXM, a Google Recaptcha, additional validators and much more.

These pages serve as the documentation for the module. They are splitted into 2 sections:

- **Content Editor Documentation** Here you will find how to use the module once installed and configured.
- **Developer Documentation** This part contains installation instructions and info on how to add customizations on the module that fit your solution.

If you are missing stuff in this documentation. Feel free to send me a private message on Sitecore Slack (@bverdonck) or send me a tweet on twitter (@_onelittlespark).

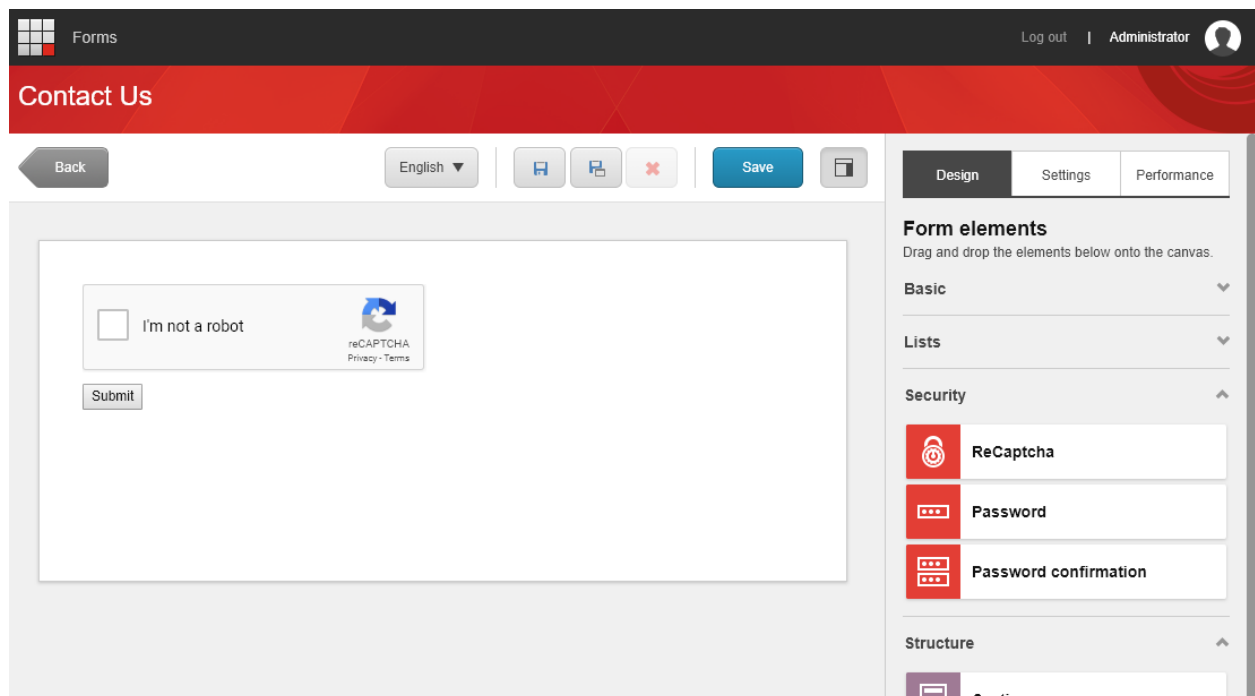
Found a bug? Please reported it on [Github](#)

Have fun with the module!

The Sitecore Forms Extensions module provides a number of additional fields to use on your forms.

1.1 Recaptcha

To use the recaptcha component, just drag it onto your form. There are no configurations to be made.



1.2 File Upload

Allow your users to upload files through your form by using the File Upload field.

To use the component drag it onto the form.

1.2.1 Set the allowed files

Optionally, if you wish to restrict the file upload to certain file types, you can enable the ContentType Validator.

When enabled, you need to fill in the Allowed Content Types field and add the content types you wish to allow. Add multiple content types on separate rows.

The screenshot displays the Sitecore Forms configuration interface for a 'File Upload' field. The interface includes a top navigation bar with 'Forms', 'Log out', and 'Administrator' links. The main area shows a 'Contact Us' form with a 'File Upload' field. The configuration panel on the right is titled 'File Upload' and contains the following settings:

- Label:** A text input field with the value 'Label'.
- Field name:** A text input field with the value 'File Upload'.
- Allowed Content Types:** A text area with a red box around it containing the values 'image/jpeg' and 'image/png'.
- Maximum file size (bytes):** A text input field.
- Validation:** A section with a red box around the 'ContentType Validator' checkbox, which is checked. The 'FileSize Validator' checkbox is unchecked.
- Field importance:** Radio buttons for 'Optional' (selected) and 'Mandatory'.
- Buttons:** 'Apply' and 'Cancel' buttons at the bottom.

1.2.2 Limit file size

Optionally, you can limit the size of the files that are allowed to upload.

With the file size validator you can limit the maximum allowed file size of the uploads. The value is in bytes and is validated client and server side.

Contact Us

Back English Save

File Upload

Label
Label

Field name
* File Upload

Allowed Content Types

Maximum file size (bytes)
50000

Validation

Field validation
☐ ContentType Validator
☒ FileSize Validator

Field importance
☒ Optional ☐ Mandatory

Apply Cancel

1.3 Raw HTML

With the Raw HTML component you can add unescaped content onto your form. This allows to add small custom javascripts or some custom tailored html snippets on your form.

Draw the Raw HTML component on the form. Select the component and add html in the HTML field.

The screenshot shows the 'New form' editor in Sitecore Forms. The main canvas displays a light blue rectangular field with the text 'Hello!'. The right-hand sidebar is open to the 'RawHtml' field configuration. The 'Details' section is expanded, showing the 'Html' property set to `<h1>Hello!</h1>`, which is highlighted with a red rectangle. Below this, the 'Field name' is set to 'RawHtml'. The 'Styling' section is also visible, with the 'CSS class' property empty. At the bottom of the sidebar are 'Apply' and 'Cancel' buttons.

1.4 Hidden Field

The hidden fields renders an `<input type="hidden" value="" />` in your form.

This can be usefull when you want to send additional data to the client browser to be processed by some javascript.

An example of this could be to send additional data on the google datalayer.

To use, drag the component on the form.

The screenshot shows the 'New form' editor with a 'Hidden' field added to the canvas. The field is a light blue rectangle containing the text 'Hidden Field: Hidden'. The right-hand sidebar is open to the 'Hidden' field configuration. The 'Field name' is set to 'Hidden'. The 'Bindings' section is expanded, showing the 'Source' property set to a dropdown menu. The 'Prefill value' checkbox is checked, with a note below it: 'The field will be prefilled with the value in the bin...'. The 'Store value' checkbox is unchecked, with a note below it: 'The value filled in by the user is stored inside the...'. At the bottom of the sidebar are 'Apply' and 'Cancel' buttons.

Give the field a name, and select a prefill binding to fill in the value of the field.

More info no prefilling can be found on [Form Bindings \(Prefilling\)](#)

With the “Send Email” submit action, a mail can be send out after the submission of the form.

The action works in collaboration with Email Experience Manager (EXM), the build in email client of Sitecore.

To use the “Send Email” action, you should first create an automated message in the EXM module.

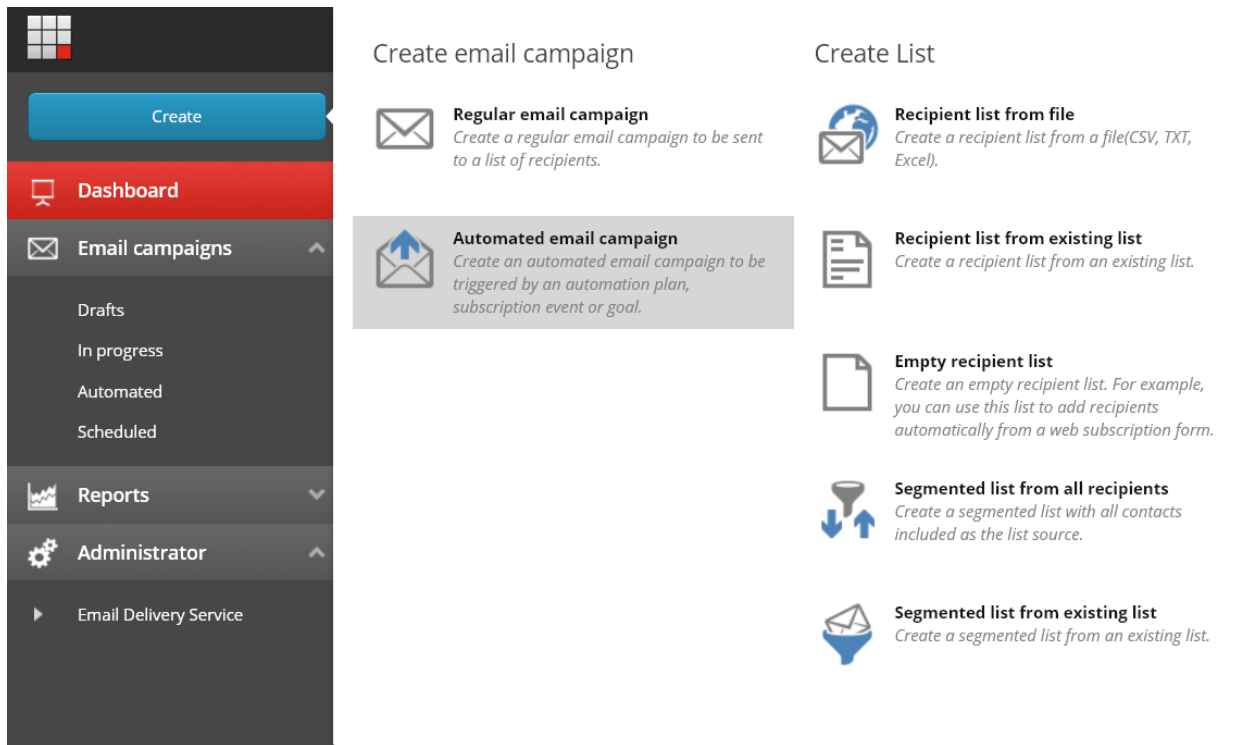
2.1 Create an automated message in Email Experience Manager

The “Send Email” submit action works in collaboration with Email Experience Manager (EXM).

Learn here, how to setup an email to be used for the send mail action.

2.1.1 Creating a new automated campaign

First, go to Email Experience Manager and create a new automated message.



Take any of the available templates. In the best case, some custom tailored templates are provided by the developers for your site.

Automated email campaign
✕

Import

HTML file

Templates

Sample Newsletter

Two-column message

One-column message

Simple HTML message

Plain text message

Existing page

Name:

Create

Cancel

Fill in the general info like “from” name and email.

GENERAL
MESSAGE
REVIEW
DELIVERY

General information

Name:

Description:

Location:

Campaign group:

Sender details

From name:

From email: ☒

Reply to:

Email campaign info

Type	Automated
Status	Draft
Included recipients	0
Excluded recipients	0
Globally excluded	0
Total recipients	0
End activation	
Active end time	
Attachments (0)	View
Previewed for recipient	None

Use the template to fill in the body of your email message.

GENERAL
MESSAGE
REVIEW
DELIVERY

Details

Subject

Body

Hi,

The form on the website was submitted.

The value for firstname was: \$form_firstName\$

The entire form looked like:

\$formFields\$

Finally, we need to activate the message in the delivery tab. Only activated message will be available to use on the “Send Mail” action.

The screenshot shows the 'DELIVERY' tab in the Sitecore Forms configuration interface. The 'Delivery options' section includes three main settings: 'Schedule delivery' with radio buttons for 'Activate message now' (selected) and 'Activate message at scheduled time'; 'Notification' with a checkbox for 'Notify these email addresses when the message delivery is complete.' and an associated text input field; and 'Personalization' with a checkbox for 'Select to render the personalized content, for example rule-based personalization, in this message for each recipient. This does not affect tokens that are always rendered for each recipient, or how opens and clicks are tracked.' A 'More' link is visible next to the personalization checkbox. At the bottom right, there is an 'Activate message' button.

GENERAL > MESSAGE > REVIEW > DELIVERY

Delivery options

Schedule delivery:

- ☒ Activate message now
- ☐ Activate message at scheduled time

Notification:

☐ Notify these email addresses when the message delivery is complete.

For more than one email use comma as separator.

Personalization:

☐ Select to render the personalized content, for example rule-based personalization, in this message for each recipient. This does not affect tokens that are always rendered for each recipient, or how opens and clicks are tracked.

[More](#)

Activate message

2.1.2 Using form values in the email

To print the form data entered by the visitor in the email, you must use tokens. These tokens will be replaced with the actual values when someone submits the form and an email gets sent out.

The form fields can be referenced individual, field per field, or you can use one token that will print all values from the form in one token.

Using individual tokens

In the forms module, each field you add should be given a name.

Use this name to construct the token.

Token format:

\$form_fieldname\$ where fieldname should be replaced by the name of the field.

So in this case this would be **\$form_firstName\$**

Print all field values with one token

Use the token **\$formFields\$** to print all fields with their value in the email.

2.2 How to use the Send Mail action

Create a basic form with Sitecore Forms make sure to add a submit button.

The screenshot shows the Sitecore Forms editor interface. On the left, a preview of a 'Contact Us' form is displayed with fields for First Name, Last Name, Email, Phone, and a text area for 'Your Question'. A 'Submit' button is at the bottom. On the right, the configuration panel for the 'Submit button' is open. It includes a 'Field name' dropdown set to 'Submit Button', a 'Navigation step' dropdown set to 'Submit', and a 'Styling' section. The 'Submit actions' section is currently empty, showing 'There are no actions to display'. At the bottom of the panel are 'Apply' and 'Cancel' buttons.

On the submit button, go to submit actions and add the send email to fixed address action.

This screenshot shows the same 'Contact Us' form preview on the left. On the right, the configuration panel for the 'Submit button' is shown with the 'Submit actions' section expanded. A list of actions is visible, including 'Trigger Goal', 'Send Email', 'Trigger Campaign Activity', 'Update Contact', 'Trigger Outcome', 'Send Email Campaign Message', and 'Redirect to Page'. The 'Send Email' action is highlighted, indicating it has been selected for configuration.

Now, we need to pick the automated message created in EXM. This message will be send upon submission of the form. Next, we need to choose who will be the recipients of the mail.

2.2.1 Send to current contact

Pick mailto “current xDB contact’s email address”.

Just like the sitecore build in ‘send email campaign message’ submit action, this will send the exm mail to the current identified contact.

However, using the send email action from the Sitecore Forms Extensions module will make the forms field values available as tokens for the composed email.

Note that, when you use the action, you must make sure that the current visitor is identified and has an email address on his contact data. In order to this, you could use the *Form Bindings (Prefilling)* feature on an email field in the form.

Select email campaign message

Select the email campaign message that should be sent when the button is clicked.

Email Campaign

Message

My Form

Select your automated email campaign, created in EXM

Send To

Mail to

Current xDB contact's email address

OK

Cancel

2.2.2 Send to fixed email address

Choose mailto “Fixed Email Address”

This options is most usefull when you want to email the backoffice after a form was submitted.

Fill in the email addresses the mail needs to be send to.

You can specify multiple addresses by using a semicolon between the addresses. Make sure to leave no spaces.

Select email campaign message

Select the email campaign message that should be sent when the button is clicked.

Email Campaign

Message

My Form

Select your automated email campaign, created in EXM

Send To

Mail to

Fixed Email Address

Settings

Email

bart.verdonck@gmail.com;jos@reference.be

Use a semicolon to specify multiple addresses

OK

Cancel

2.2.3 Send to form field value

Use this option to email the visitor that has filled in the form.

Choose mailto “Value from field in form”

Choose one of the fields of the form. This field must contain an email address. It can be a free email field for the visitor to enter it’s email address or it could also be a dropdown with choices that have an email as value. (For example if you want to mail to a different department based on a chosen topic).

Select email campaign message

Select the email campaign message that should be sent when the button is clicked.

Email Campaign

Message

My Form

Select your automated email campaign, created in EXM

Send To

Mail to

Value from field in this form

Settings

Email Field

Email

☐ Update current contact

If selected, the current session will be identified on this email address. If not, a separate system contact will be retrieved or created.

OK

Cancel

With this option you can also check the “update current contact” checkbox.

When you check this box, then upon submission, the current session will be updated with the value from the chosen field. This session will then become identified instead of anonymous. (If the session was already identified, the email will just be updated in the profile.

When this box is not checked, we will lookup if there is a contact with the provided email address. If we find a contact, we will use this contact, if not, a new contact will be created.

2.3 Attachments

When using *File Upload* fields in your form, you can choose to add them as an attachment of the email.

Form Bindings (Prefilling)

With the form bindings, you can prefill fields in your form based on xDB contact data. The form bindings also support to store the data entered in the form on your xDB contact.

Note: For security reasons it might be prudent to limit prefilling to forms behind a login. Storing info on the profile is less sensitive and can be used on forms without login. More info on <https://www.campaignion.org/blog/all-you-have-know-about-pre-filling-forms>

3.1 How to configure a field

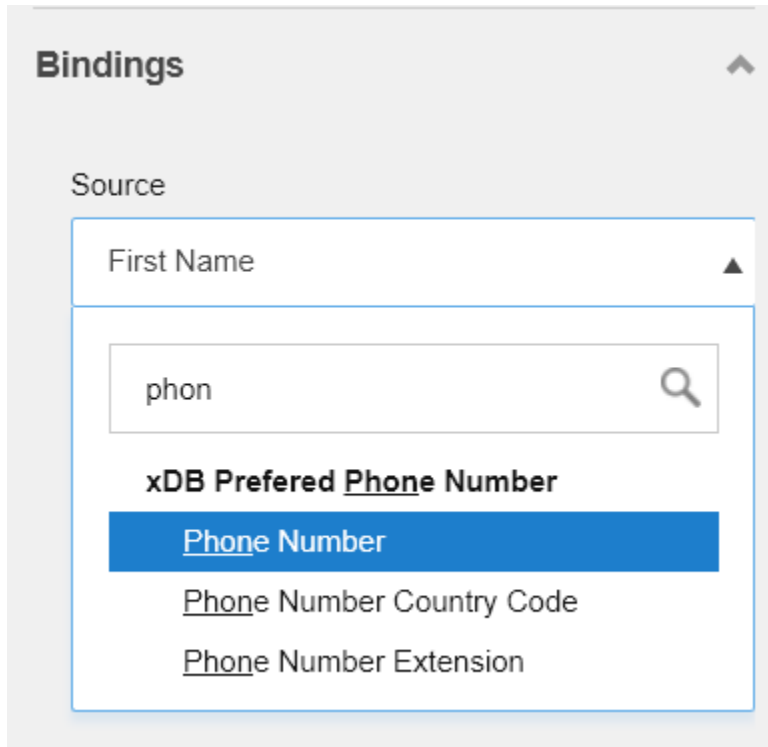
In the forms editor, select the field you wish to prefill.
Check the “Bindings” section in the panel on the right.

The screenshot displays the Campaignion Forms Editor interface. At the top, a red header bar contains the text "Test Form Bart". Below this, a toolbar includes a "Back" button, a language dropdown set to "English", and icons for undo, redo, delete, save, and a form preview. The main workspace shows a form with several fields: "First Name" (highlighted in blue), "Email", "Remarks" (a text area), "Telephone", a checkbox for "Do Not Market", and a "Birth Date" field with a date picker. A "Verstuur" (Send) button is at the bottom of the form. On the right side, a configuration panel for the "Single-line text" field is open. It has tabs for "General" and "Performance". The "Details" section is expanded, showing the "Bindings" configuration. Under "Source", "First Name" is selected. The "Prefill value" checkbox is checked, with a note: "The field will be prefilled with the value in the bin...". The "Store value" checkbox is also checked, with a note: "The value filled in by the user is stored inside the...". At the bottom of the panel are "Apply" and "Cancel" buttons. A small console at the bottom left shows "avscriptvoid(0)".

3.1.1 Source

Mandatory

In the source field, you must select what source you want to link the field to. The prefill value of the form will be fetched from this source.



The screenshot shows a configuration panel titled "Bindings" with a "Source" dropdown menu. The dropdown is open, displaying a search bar with the text "phon" and a magnifying glass icon. Below the search bar, a list of options is shown: "xDB Preferred Phone Number", "Phone Number" (highlighted in blue), "Phone Number Country Code", and "Phone Number Extension".

3.1.2 Prefill Value

Optional

Check the “prefill value” checkbox in order to prefill this field with the selected source.

3.1.3 Store Value

Optional

Check the “store value” checkbox in order to save the value entered by the visitor in the form to the selected source.

3.2 Supported Form Elements

This functionality is available for form elements of the type:

- Single-Line Text
- Multi-line Text
- Number
- Email

- Hidden
- Telephone
- Checkbox
- Date

3.3 Example

3.3.1 Build Form

Let's create a new form with 3 fields:

FirstName We will prefill this from the xDB profile and store it back upon form submission

The screenshot displays the Sitecore Forms Builder interface. On the left, a form canvas shows a single-line text input field labeled 'FirstName'. Above the canvas is a toolbar with buttons for 'Back', 'English' (with a dropdown arrow), a save icon, a delete icon, a 'Save' button, and a preview icon. On the right, a configuration panel for the 'Single-line text' field is open. It has two tabs: 'General' (selected) and 'Performance'. Under the 'General' tab, there are sections for 'Details', 'Bindings', and 'Validation'. The 'Bindings' section is expanded, showing a 'Source' dropdown set to 'First Name'. Below this, there are two checked checkboxes: 'Prefill value' and 'Store value'. Descriptive text for these options reads: 'The field will be prefilled with the value in the bind...' and 'The value filled in by the user is stored inside the p...'. At the bottom of the configuration panel are 'Apply' and 'Cancel' buttons.

Email We will prefill this from the xDB profile and store it back upon form submission

New form

Back

English ▼
📄
📄
✖
Save
📄

First Name

Email

Email

Maximum text length

255

Bindings

Source

Email ▼

☒ Prefill value

The field will be prefilled with the value in the bind...

☒ Store value

The value filled in by the user is stored inside the p...

Validation

Apply
Cancel

javascript:void(0)

Birthday We will only store the value, but not prefill it

First Name

Email

dd/mm/yyyy
Birth Day

End date

Bindings

Source

Birthdate ▼

☐ Prefill value

The field will be prefilled with the value in the bind...

☒ Store value

The value filled in by the user is stored inside the p...

Finally we will add a submit button with no submit actions attached. (just for the example, it is perfectly possible to add submit actions)

The screenshot shows the Sitecore Forms builder interface. On the left, a form is being built with three input fields: 'First Name', 'Email', and 'Birth Day' (with a date mask 'dd/mm/yyyy'). Below these is a blue 'Submit' button. On the right, the 'Styling' panel is visible, showing a 'CSS class' input field and a 'Submit actions' section with icons for delete, edit, up, down, and add. Below the icons, it says 'There are no actions to display.'

Finish by putting the form on a page and test it.

3.3.2 Testing

Let's visit the form now.

Considering a new first time visitor, the form will be empty, since there is no data on his xDB profile yet.

The screenshot shows the rendered form on a page. It consists of three input fields: 'First Name', 'Email', and 'Birth Date' (with a date mask 'dd/mm/yyyy'). Below these is a green circular 'Submit' button.

The visitor fills in the form and submit's. The data entered will now be stored in the selected xDB fields.

The screenshot shows the rendered form on a page after submission. The input fields now contain the data entered by the visitor: 'First Name' is 'Jane', 'Email' is 'jane@doe.com', and 'Birth Date' is '10/10/1984'. The green circular 'Submit' button is still present.

The visitors revisits the form (or any other form configured with data binding), the fields first name and email will be prefilled with the data from the xDB profile.

First Name Jane
Email jane@doe.com
Birth Date dd/mm/yyyy
<input type="submit" value="Submit"/>

Value Provider Conditions

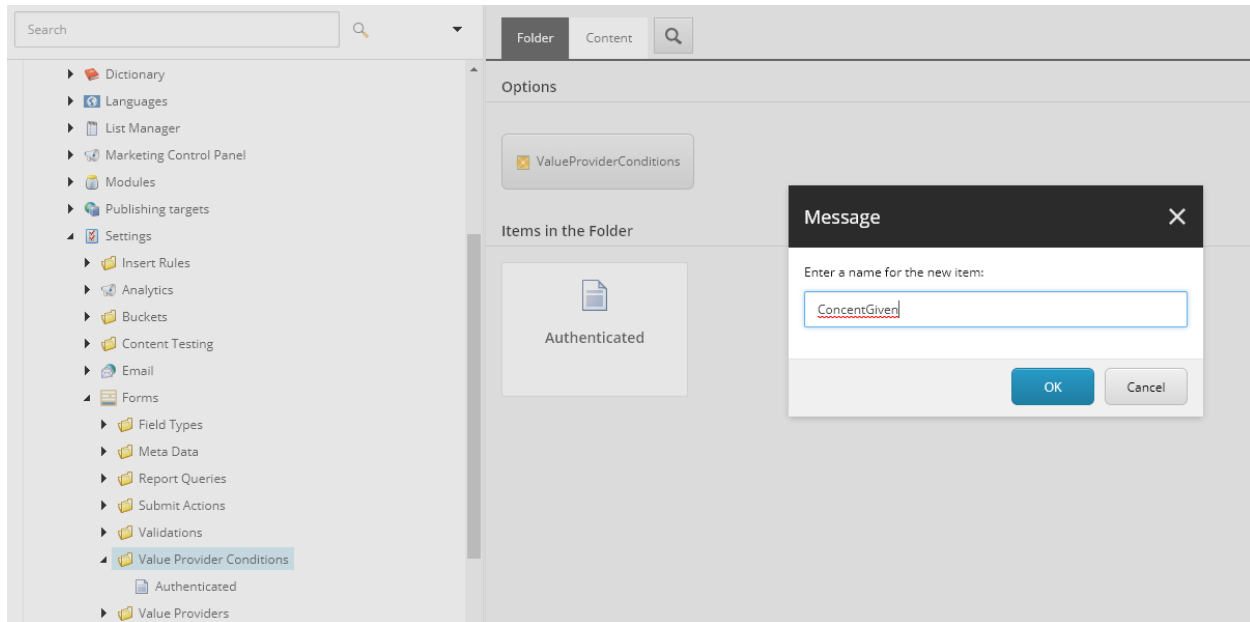
When a visitor browses to your site, we might already have a lot of information on him based on his xDB tracking cookie. When prefilling of forms is enabled by use of the fieldbindings, we can prefill the form fields from xDB. This might not always be the desired behaviour, there may be cases where you only want to prefill when a user is logged in, or has given consent to prefill the form.

With the Value Provider Conditions, you can build rules that must be met, before prefilling is applied.

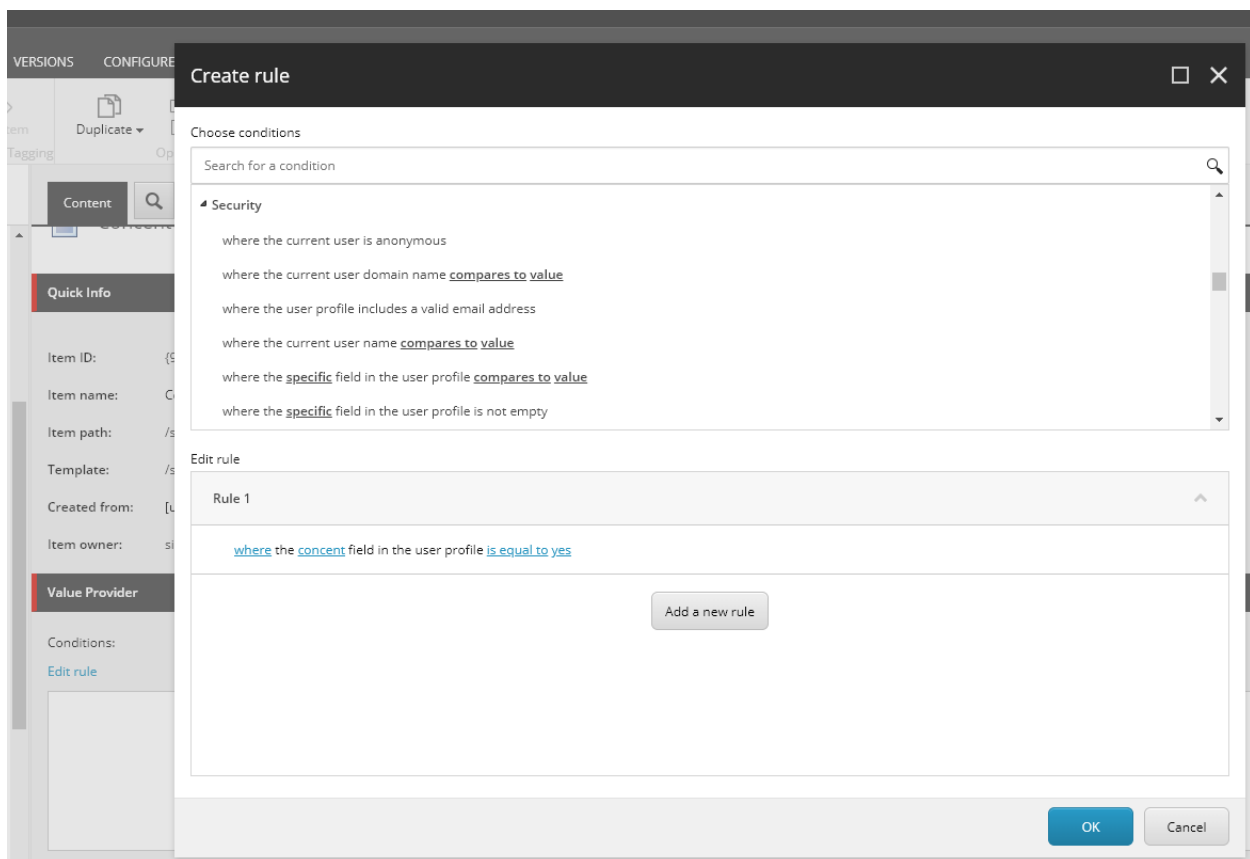
4.1 How to create a set of rules

In the content explorer, navigate to `/sitecore/system/Settings/Forms/Value Provider Conditions`

Create a new `ValueProviderConditions` item.



Next, click on “Edit Rule” and build the conditional rule you desire.

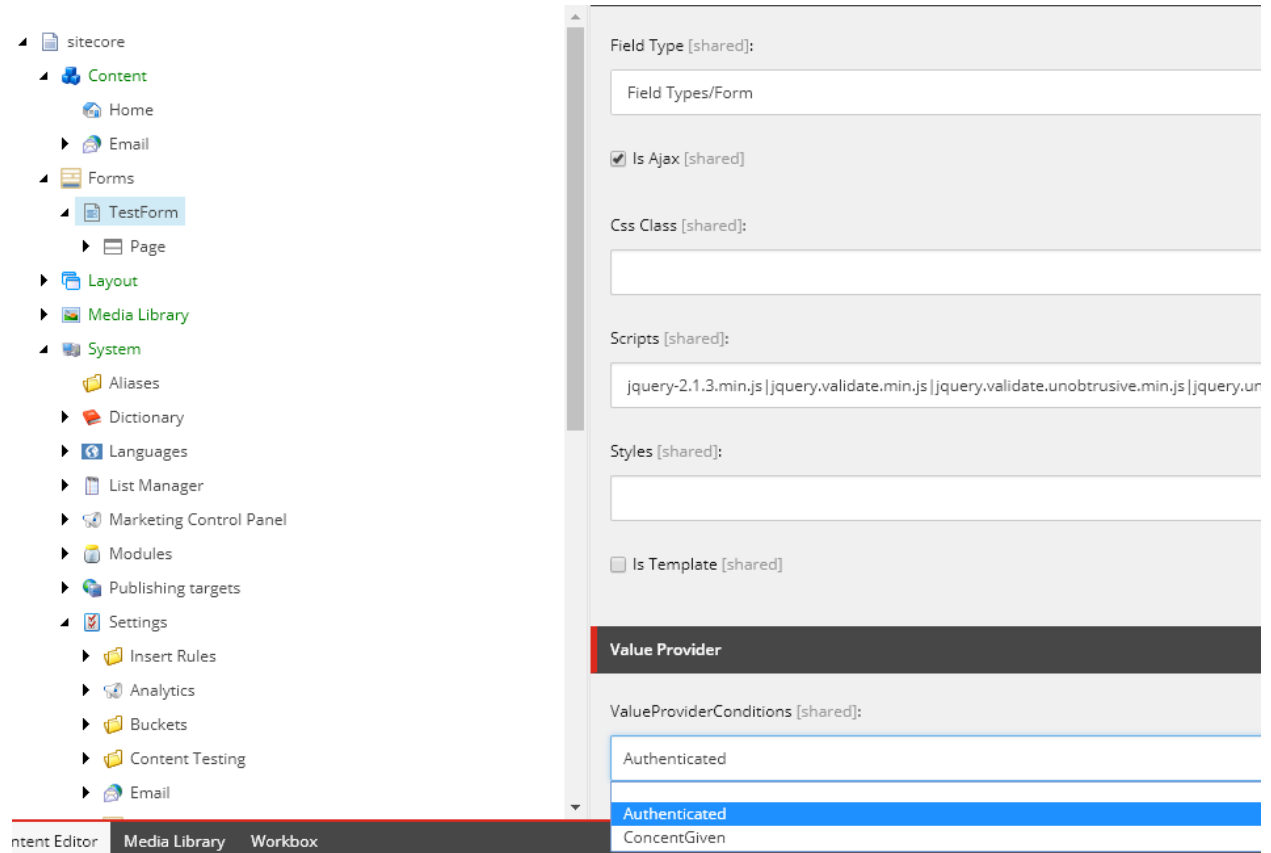


Save and publish the rule.

4.2 How to use a set of rules

In the content explorer, navigate to your form. (Sitecore → Forms → Your Form)

In the Value Provider section, choose your conditions.



Save and publish the form.

In place “Thank you” message

The ShowFormPage custom submit action makes it possible to show a thank you message after submitting the form without the need of creating a dedicated separate thank you page.

With this action, the form is replaced inline via ajax, there is no other page loaded.

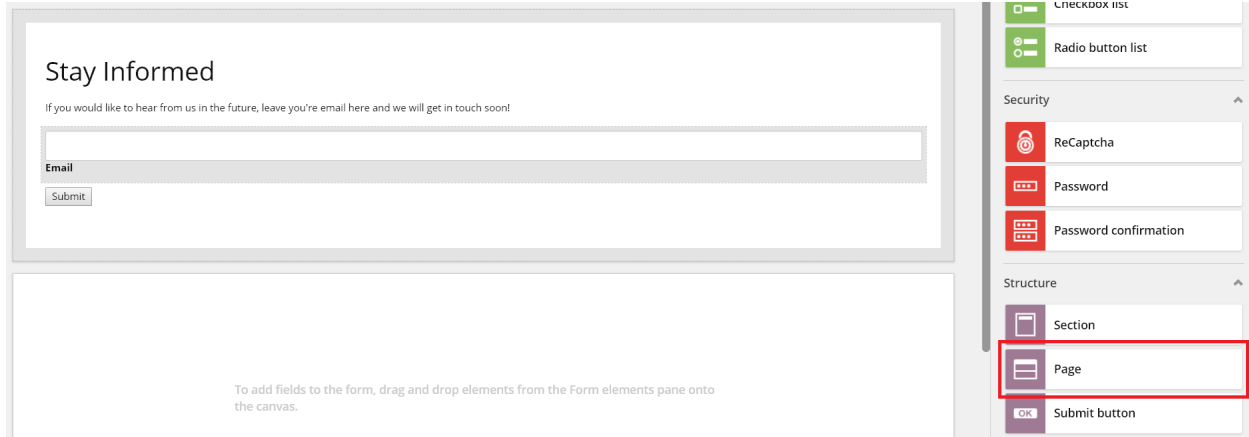
5.1 How

Let’s create a simple form to let users subscribe to a kind of newsletter.

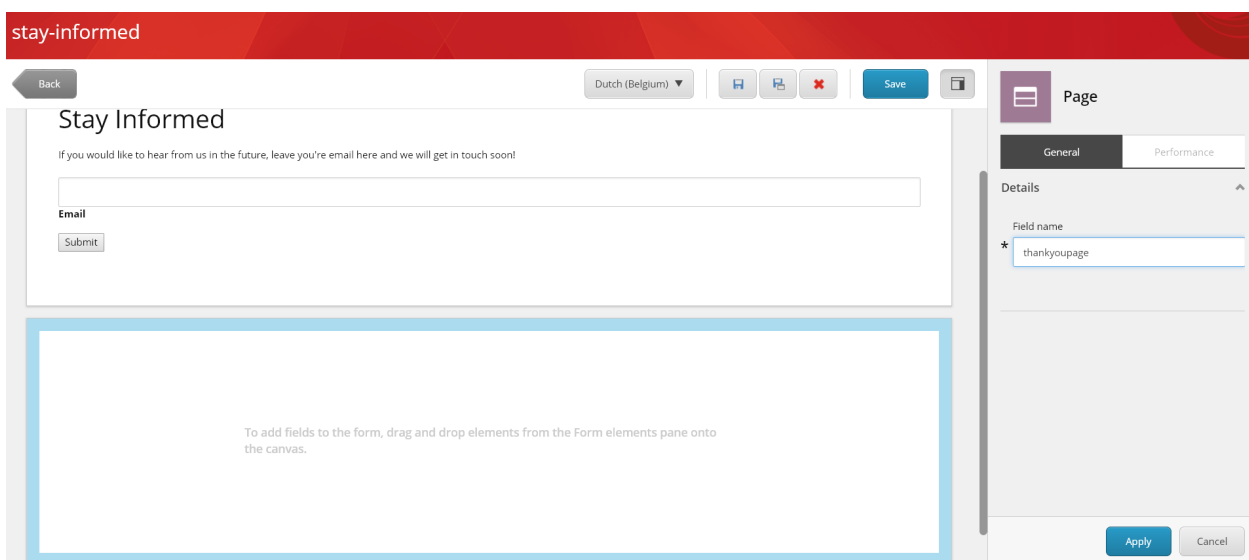
The screenshot displays the 'Forms' application interface. At the top, there's a header with 'Forms' on the left, 'Log out | Administrator' on the right, and a red banner with 'stay-informed'. Below the banner, a 'Back' button is on the left, and a toolbar with 'English', 'Save', and other icons is on the right. The main area shows a form titled 'Stay Informed' with the text 'If you would like to hear from us in the future, leave you're email here and we will get in touch soon!'. The form has an 'Email' input field and a 'Submit' button. To the right of the form is a configuration panel for the 'Submit button'. It includes a 'Submit' dropdown, a 'Styling' section with a 'CSS class' input, and a 'Submit actions' section with icons for delete, edit, up, down, and add. At the bottom of the panel is a 'Save Data' button. The 'Apply' and 'Cancel' buttons are at the very bottom of the configuration panel.

The form contains a single email field and a submit button, with the submit action save data.

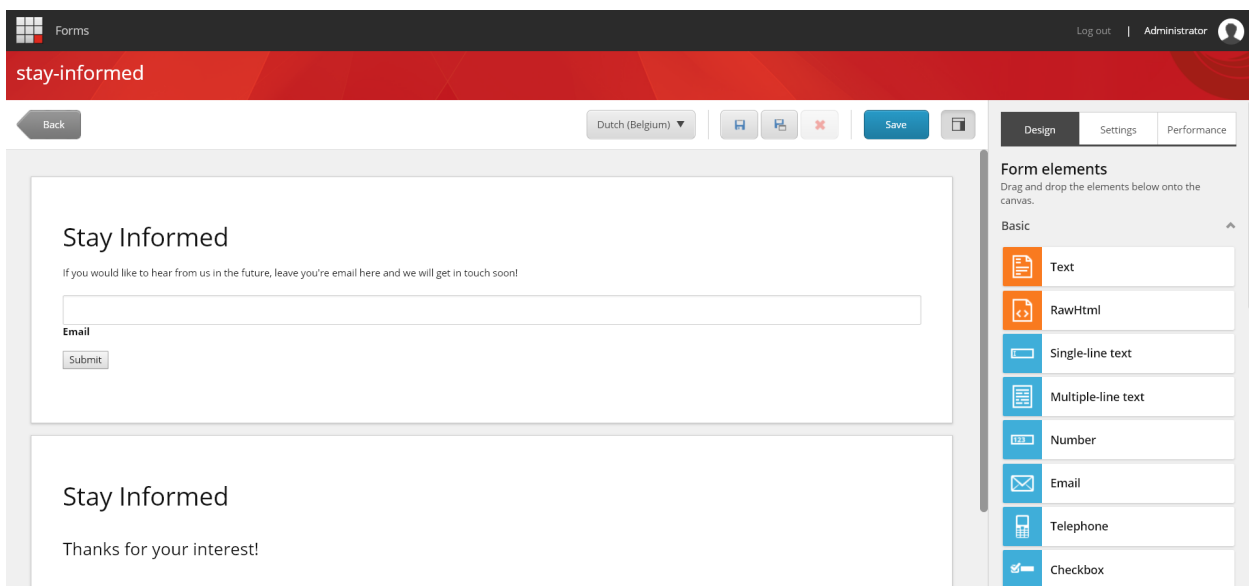
Now, let’s add a second page in the form (just like you would do for a multipage form).



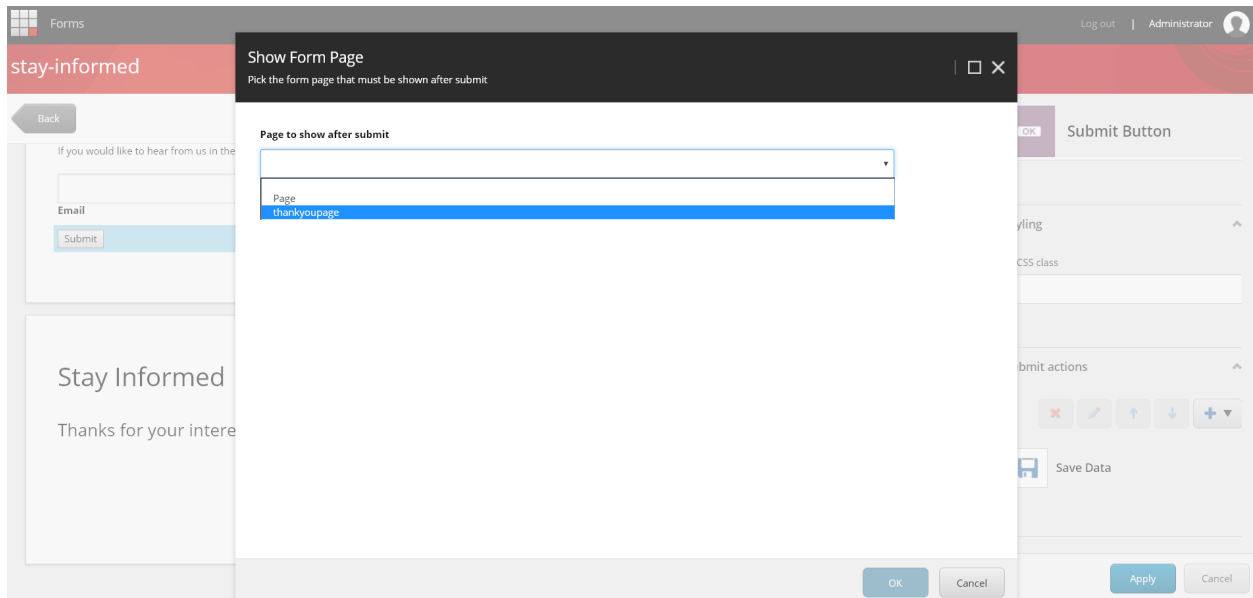
Give the page a name.



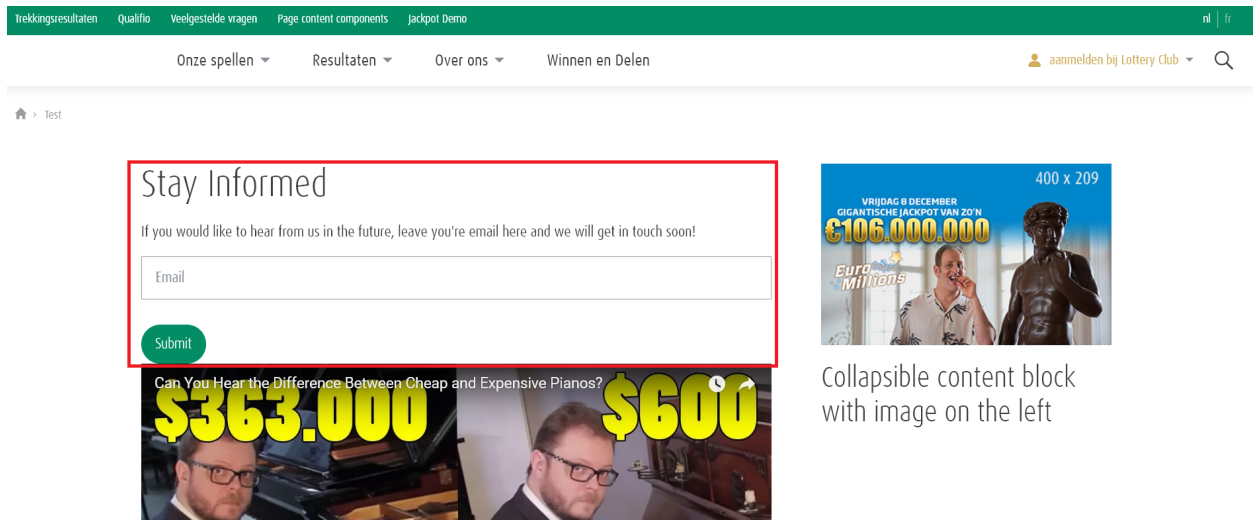
Add some components on the thankoupage.



On the submit button: add the submit action “show form page” and select the thank you page.



5.2 Result



After submit:

TrekkingsresultatenQualifioVeelgestelde vragenPage content componentsJackpot Demo

nl | fr

Onze spellen ▾Resultaten ▾Over ons ▾Winnen en Delen

aanmelden bij Lottery Club ▾

Q

Test

Stay Informed

Thanks for your interest!

Can You Hear the Difference Between Cheap and Expensive Pianos?

\$363.000

\$600

VRIJDAG 8 DECEMBER
GIGANTISCHE JACKPOT VAN 20'N
€106.000.000
Euro Millions

400 x 209

Collapsible content block
with image on the left

6.1 Download

The latest version can be found in the download section on [Github](#).

There are 3 distribution packages available:

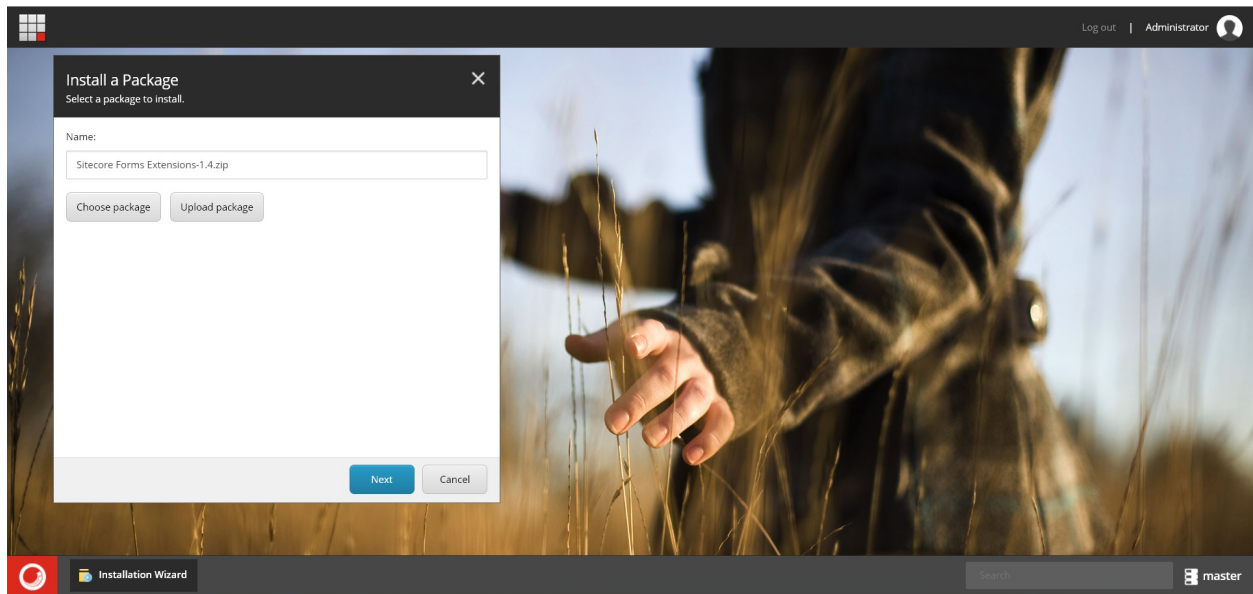
Sitecore Forms Extensions-<version>.zip Sitecore Package to install via Sitecore's installation wizard. This is the most common option.

Sitecore Forms Extensions-<version>.scwpd.zip Use this package for initial install on Azure PaaS via ARM templates.

Sitecore Forms Extensions-<version>-nodb.scwpd.zip Use this package for redeployment via ARM on Azure PaaS, this package will only install DLL's and file, sitecore items are excluded from this package.

6.2 Install

Just install the package in Sitecore via installation wizard.



6.3 Configuration

Most functionalities of the library work out of the box. However, the fileupload and captcha features require some additional configuration.

6.3.1 File Upload

The file upload requires to set a location to store the uploaded files. A file upload storage provider must be added in configuration.

There are currently 2 store locations available:

Configure Azure Blob Storage

Setup Storage Account in Azure

First, in the case you don't have an Azure Blob Storage in your solution already, you will need to create one.

To do so, go into azure portal. Click New and choose Storage Account.

The screenshot shows the Microsoft Azure portal interface for creating a new storage account. The left-hand navigation pane lists various services, with 'Storage accounts' selected. The main content area is divided into two sections: 'Storage accounts' (showing a list of existing accounts) and 'Create storage account' (the active wizard). The wizard form includes the following fields and options:

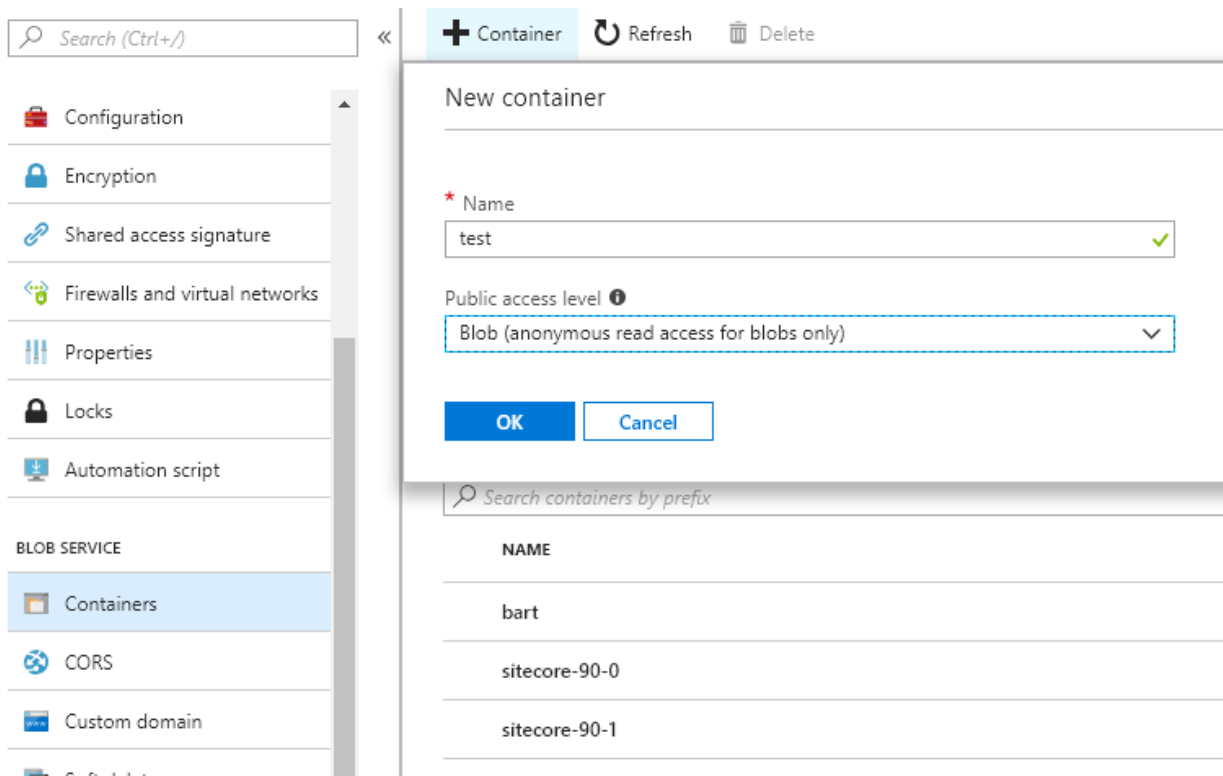
- Name:** `olsstoragedev` (with a green checkmark indicating it's valid).
- Deployment model:** `Resource manager` (selected) and `Classic`.
- Account kind:** `Storage (general purpose v1)` (selected).
- Performance:** `Standard` (selected) and `Premium`.
- Replication:** `Read-access geo-redundant storage (RA-...)` (selected).
- Secure transfer required:** `Disabled` (selected) and `Enabled`.
- Subscription:** A dropdown menu showing a selected subscription.
- Resource group:** `OLS-DEV-STORAGE` (selected). Radio buttons for `Create new` and `Use existing` are present.
- Location:** `West Europe` (selected).
- Virtual networks (Preview):** `Configure virtual networks` with `Disabled` (selected) and `Enabled` options.

At the bottom of the wizard, there is a `Pin to dashboard` checkbox and a `Create` button. A link for `Automation options` is also visible.

Next, create a storage container.

Choose public access level blob if you want easy access to the uploaded files by link. (All files will be renamed with a random GUID for security.)

Note: This is not required for the module to work



Add config file to your solution

Next, we will tell the module how to connect to the storage account.

Create a file *Feature.FormsExtensions.AzureStorageProvider.config* and add it in your website folder under *App_Config/Environment*

```
<?xml version="1.0"?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/" xmlns:role="http://
↪www.sitecore.net/xmlconfig/role/">
  <sitecore>
    <formExtensions>
      <fileUploadStorageProvider type="Feature.FormsExtensions.Business.FileUpload.
↪AzureBlobStorageFileUploadStorageProvider, Feature.FormsExtensions">
        <connectionString></connectionString>
        <blobContainer></blobContainer>
        <folder>formextentions/{formName}/{fieldName}/{language}</folder>
      </fileUploadStorageProvider>
    </formExtensions>
  </sitecore>
</configuration>
```

Go to your storage account on Azure, browse to Access Keys, and copy either one of the connectionstrings. Put this connectionstring in the config.

Enter the name you have chosen for your container in the blobcontainer part of the configuration.

Finally, you will notice the folder attribute in the configuration. You can leave this empty or put in any desired folder structure that should be followed to store your files in.

All forms will follow the same config for upload of the files. You cannot create individual storages for individual forms. This is a design choice, so that content editors do not have to worry about where to store files.

However, the folder structure does support 3 variables that can be used:

- {formName} This handle will get replaced by the name of the form.
- {fieldName} Each form-upload field can be named in the forms-editor. This handle will be replaced by that name.
- {language} If you want to separate the content by language, use this handle.

Note that none of these handles are required. They are all optional, including the order.

Local Blob Storage

Please note that when using this configuration on a loadbalanced CD, you must make sure, the local filepath is shared between the instances.

Add config file to your solution

Create a file *Feature.FormsExtensions.LocalStorageProvider.config* and add it in your website folder under App_Config/Environment

```
<?xml version="1.0"?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/" xmlns:role="http://
↪www.sitecore.net/xmlconfig/role/">
  <sitecore>
    <formExtensions>
      <fileUploadStorageProvider type="Feature.FormsExtensions.Business.FileUpload.
↪FileSystemFileUploadStorageProvider, Feature.FormsExtensions">
        <rootStoragePath>c:\temp\</rootStoragePath>
        <fileDownloadUrlBase>https://myfile.com/{0}</fileDownloadUrlBase>
        <folder>formextensions/{formName}/{fieldName}/{language}</folder>
      </fileUploadStorageProvider>
    </formExtensions>
  </sitecore>
</configuration>
```

The fileDownloadUrlBase is use to build a url that will host the uploaded images. Note that the module won't serve the images. It is up to you to provide a download service for the images. If you don't want to create this, consider using *Azure Blob Storage Provider*. It is not mandatory that a download service is provided for the module to work.

Make sure the application has enough rights to create files in the rootStoragePath you configured.

All forms will follow the same config for upload of the files. You cannot create individual storages for individual forms. This is a design choice, so that content editors do not have to worry about where to store files.

However, the folder structure does support 3 variables that can be used:

- {formName} This handle will get replaced by the name of the form.
- {fieldName} Each form-upload field can be named in the forms-editor. This handle will be replaced by that name.
- {language} If you want to separate the content by language, use this handle.

Note that none of these handles are required. They are all optional, including the order.

6.3.2 Google Captcha

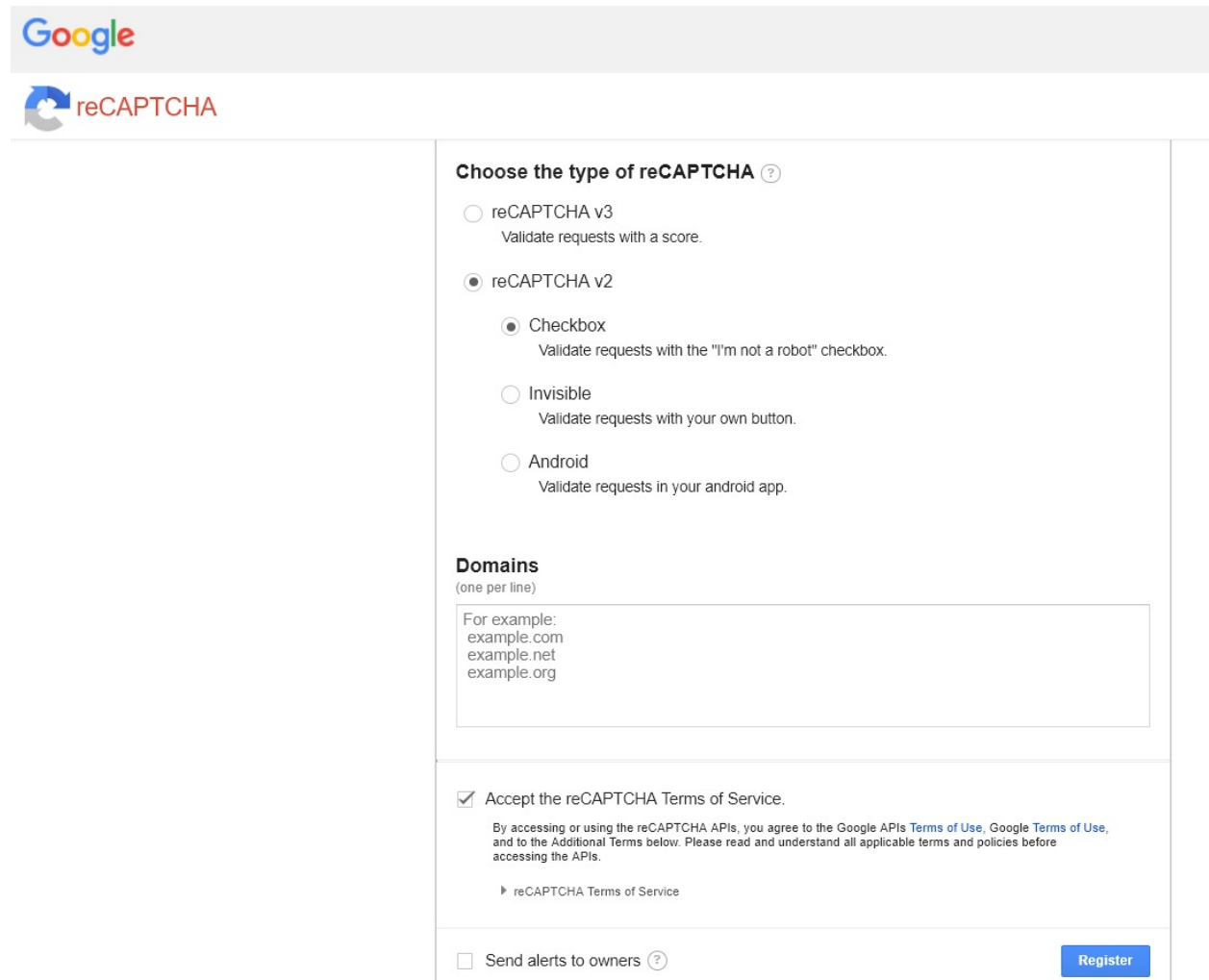
For Google Captcha to work, we need to set the public and private key in the configuration.

Configure Google Captcha v2

Setup Google Account

First we need to configure our account in Google Recaptcha Portal.

<https://www.google.com/recaptcha/admin#list>



The screenshot shows the Google reCAPTCHA Admin Portal configuration page. At the top, there is a Google logo and the reCAPTCHA logo. The main content area is titled "Choose the type of reCAPTCHA" with a help icon. It contains four radio button options: "reCAPTCHA v3" (Validate requests with a score.), "reCAPTCHA v2" (selected), "Checkbox" (Validate requests with the "I'm not a robot" checkbox.), "Invisible" (Validate requests with your own button.), and "Android" (Validate requests in your android app.). Below this is a "Domains" section with a text input field containing "example.com", "example.net", and "example.org". There is a checkbox for "Accept the reCAPTCHA Terms of Service." with a link to the terms of service. At the bottom, there is a checkbox for "Send alerts to owners" and a "Register" button.

Add Config File to your Solution

Create a file *Feature.FormsExtensions.GoogleRecaptcha.config* and add it in your website folder under App_Config/Environment

```
<?xml version="1.0"?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/" xmlns:role="http://
www.sitecore.net/xmlconfig/role/">
```

(continues on next page)

(continued from previous page)

```
<sitecore role:require="Standalone or ContentManagement or DedicatedDispatch">
  <settings>
    <setting name="GoogleCaptchaPublicKey" value="6Lfik1cUAAAAA03osc-yX6ZfZhckPm_
↪4GIAKGdKh" />
    <setting name="GoogleCaptchaPrivateKey" value="6Lfik1cUAAAAAInwnC6-
↪jlgbcGeZGp701AOXaSHL" />
  </settings>
</sitecore>
</configuration>
```

The setting “GoogleCaptchaPublicKey” should contain the site key.

The setting “GoogleCaptchaPrivateKey” should contain the secret key.

① Adding reCAPTCHA to your site

Keys

Site key

Use this in the HTML code your site serves to users.

6Lfik1cUAAAAA03osc-yX6ZfZhckPm_4GIAKGdKh

Secret key

Use this for communication between your site and Google. Be sure to keep it a secret.

6Lfik1cUAAAAAInwnC6-jlgbcGeZGp701AOXaSHL

Step 1: Client side integration

Step 2: Server side integration

Key Settings

Delete Key

Label

(reCAPTCHA v2)

For example, example.com: Comments page

Domains

(one per line)

scformsext.local.reference.be
scformsexttest.local.reference.be
scformsext-demo.local.reference.be
scformsext-demo-dev.local.reference.be
scformsext-demo-tm.local.reference.be

☒ Accept the reCAPTCHA Terms of Service.

By accessing or using the reCAPTCHA APIs, you agree to the Google APIs [Terms of Use](#), Google [Terms of Use](#), and to the Additional Terms below. Please read and understand all applicable terms and policies before accessing the APIs.

[reCAPTCHA Terms of Service](#)

☐ Send alerts to owners

Register

6.3. Configuration

37

Extend the *Form Bindings (Prefilling)* with your own custom xDB facets or even with an entirely different source like an ERP system.

7.1 Add a custom binding source

7.1.1 Create BindingHandler

To add a custom binding source, you should write a `IBindingHandler` interface.

```
namespace Feature.FormsExtensions.Business.FieldBindings
{
    public interface IBindingHandler
    {
        IBindingHandlerResult GetBindingValue();
        void StoreBindingValue(object newValue);
    }
}
```

The `GetBindingValue` should return a `IBindingHandlerResult`.

An example implementation could be like:

```
namespace Feature.FormsExtensions.Business.FieldBindings
{
    public class DemoBindingHandler : IBindingHandler
    {
        public IBindingHandlerResult GetBindingValue()
        {
            var fullName = Sitecore.Context.User.Profile.FullName;
            if (string.IsNullOrEmpty(fullName))
            {
                return new NoBindingValueFoundResult();
            }
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

    }
    return new BindingValueFoundResult(fullName);
}

public void StoreBindingValue(object newValue)
{
    if (newValue is string fullName)
    {
        Sitecore.Context.User.Profile.FullName = fullName;
    }
}
}

```

The storebindingvalue is only called when the newValue is not null.

7.1.2 Register BindingHandler

To register the bindinghandler(s) you have created, you must create a processor.

```

public class DemoBindingHandlerLoader : MvcPipelineProcessor
{
    <LoadFieldBindingHandlersArgs>
    {
        public override void Process(LoadFieldBindingHandlersArgs args)
        {
            var tokenKey = new FieldBindingTokenKey("My Custom Handlers", "x.y.z.customhandler
            <Custom Handler>");
            args.FieldBindingHandlers.Add(tokenKey, new DemoBindingHandler());
        }
    }
}

```

The tokenkey consists of 3 parameters:

- Category: this will group the handlers in the sitecore forms user interface
- Id: a unique id for your handler (this can be anything)
- Name: the name of the handler that will be shown to the user

Once forms are created with your custom handler, you should not change the id anymore. The category and name can be safely changed as they are not stored on the form components.

7.1.3 Add the BindingHandlerLoader to the loader pipeline

Create a config file to add your loader to the forms.loadFieldBindingHandlers pipeline.

```

<configuration>
  <sitecore>
    <pipelines>
      <forms.loadFieldBindingHandlers>
        <processor type="mypackage.DemoBindingHandlerLoader , mydll" resolve="true" />
      </forms.loadFieldBindingHandlers>
    </pipelines>
  </sitecore>
</configuration>

```

7.2 Configure preferred email, address and phonenumber

The module comes with a set of databinding handlers to support xDB. The email, address and phonenumber facet on the contact profile contain a lists. There is always one preferred entry in the list.

The build in bindings always store and load from the preferred email, address or phonenumber.

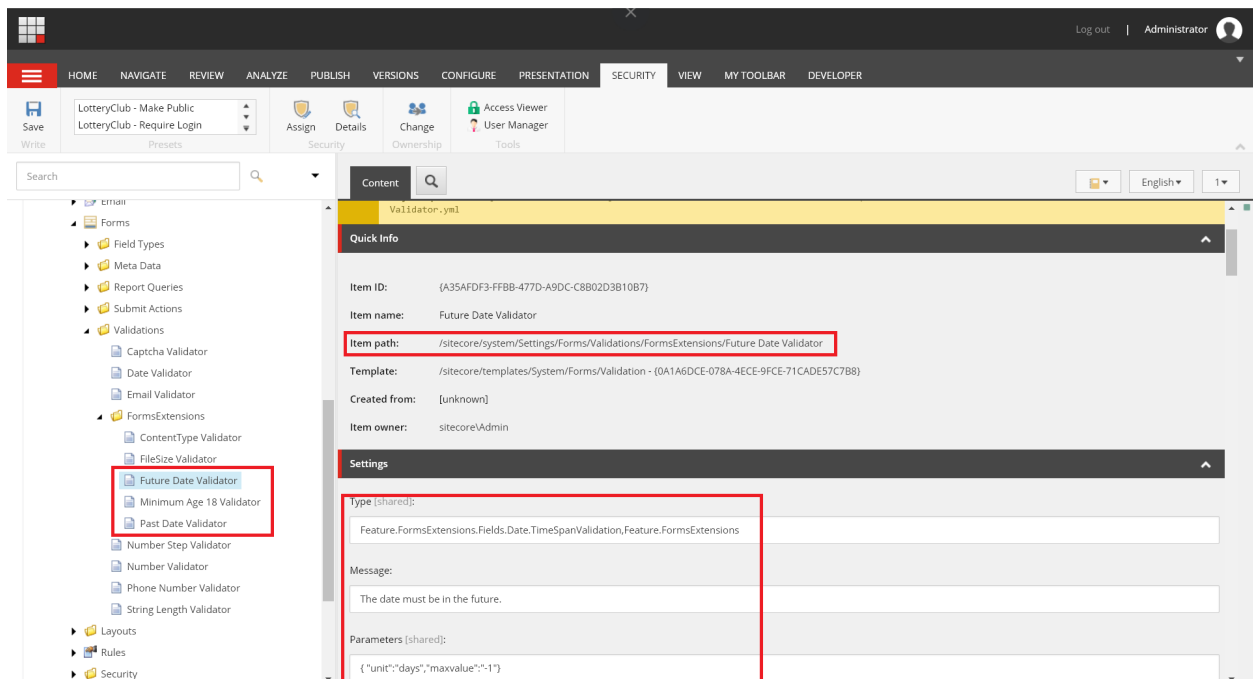
If the facet does not yet exist, it has to create the facet and set the preferred email, address or phonenumber. The key that is used for this is stored in a sitecore setting. These settings can be overridden to fit your projects needs.

```
<?xml version="1.0"?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/" xmlns:role="http://
↪www.sitecore.net/xmlconfig/role/">
  <sitecore>
    <settings>
      <setting name="XDbPreferredAddress" value="address" />
      <setting name="XDbPreferredPhoneNumber" value="phone" />
      <setting name="XDbPreferredEmailAddress" value="email" />
    </settings>
  </sitecore>
</configuration>
```


Configure Date Timespan Validator

The Date Timespan Validator is a new custom validator released in version 1.5 of the Sitecore Forms Extensions module.

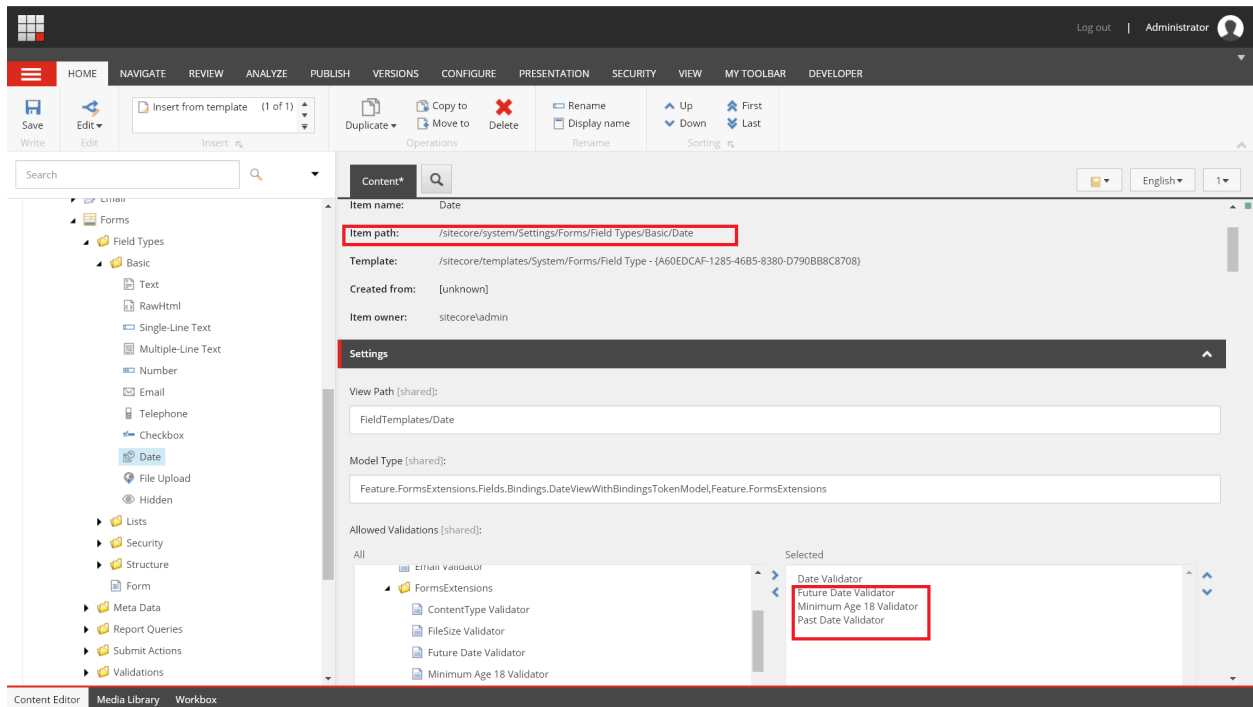
The module comes with 3 preconfigured timespan validators, but you can easily add you own.



The validators are defined in `/sitecore/system/settings/forms/validations/formsextensions`.

We provide a future date, past date and minimum age 18 validator with the package. Feel free to add you own timerange validator.

The validators are added on the date field located at `/sitecore/system/settings/forms/field types/basic/date`



Once configured, the validators become available in the forms editor on a date field.

