Introduction to Ingeniux Forms Builder

90 minute Course
CMSFB-V6
P.0-20080901

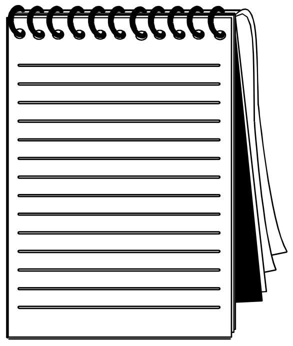
INGENIUX

Table of Contents

Course Objectives

In this course, you will:

- Install Forms Builder.
- Build a form page.
- Create a form handler page.



Introducing Ingeniux Forms Builder

The Ingeniux CMS has an application module called Forms Builder. This tool is actually a set of files and code that allow the creation of forms and form processors through the Ingeniux CMS.

There is no unique user interface or wizard used by Forms Builder. Instead, its files are placed on the server and configured to recognize forms created through the CMS.

Forms pages, components, and actions are created within the CMS in the same general way that any new page or component is created. The schemas for these form-related items contain form-specific elements that must be completed in a certain way to ensure proper functioning.

This document provides the information necessary to set up, configure, and use the Ingeniux Forms Builder application module.

➢ Acquiring Ingeniux Forms Builder

The Forms Builder application module is provided to customers by Ingeniux per contractual agreements; it is not built into the application by default.

☞ Contact your account manager at Ingeniux if your contract specifies Forms Builder to acquire the application files.

☞ Forms Builder is not currently a supported application.

Installing Forms Builder

The Forms Builder application module consists of several files that must be placed within the site on the design time server. The steps for installing the application include:

1. Install ComExecute for sending email.
2. Copy files to the appropriate XML subdirectories
3. Revise existing XSL stylesheets to display forms
4. Verify existing schemas include a component element

☞ The newest versions of Forms Builder require Ingeniux CMS 5.x or later.

➢ Installing ComExecute for Sending Mail

The ComExecute file is used to send email. It must be installed before a form can perform a send mail action.

The ComExecute file must correspond with the version of the software currently installed. If the Ingeniux CMS software is upgraded, the ComExecute file must also be upgraded to match.

☞ This file must be run each time a site is upgraded, as well as whenever a site is moved to a new server.

➲ To Install ComExecute:

1. Locate the ComExecute installer in the Forms Builder application files provided by Ingeniux.
2. Run the installer on the design time server.

☞ This requires file system access to the design time server.

➢ Copying Files into the XML Directory

Within the Forms Builder application module files there are several files that should be copied to the site's XML directory on the design time server. There are four subdirectories into which these files belong:

- Documents

- Prebuilt
- Schemas
- Stylesheets

➲ To Copy Files into the XML Directory:

1. Locate the four directories provided in the Forms Builder application files provided by Ingeniux.
2. Copy these files into the corresponding directories in the XML directory for the site on the design time server.

☞ This requires file system access to the design time server.

➢ Revising XSL Stylesheets to Display Forms

Before a page can display any content entered through the CMS, the corresponding stylesheet must be told to display the content. The following actions are taken to attach Forms Builder stylesheets to default.xsl, or whatever root stylesheet is used by any page type that will contain a form. If a single root stylesheet does not exist, these actions must be taken on a stylesheet by stylesheet basis.

Adding Included Stylesheets

Once the stylesheets included with Forms Builder are copied to the stylesheets directory, the stylesheets specific to the site must be told to point to those Forms Builder-specific stylesheets.

Add the following to the include section, typically found at the beginning of the default stylesheet.

- `<xsl:include href=”include-FormBuilder.xsl” />`

  This controls display of the rendered form.

- `<xsl:include href=”include-serializeXML.xsl” />`

  This converts XML to a stream.

- `<xsl:include href=”include-tokenizer.xsl” />`

  This converts a stream to XML.

☞ This requires file system access to the design time server.

Updating PageType Match

For each page type schema that should display a form, the corresponding stylesheet must indicate where the form appears in the context of that page.

For example, if the form will be used on pages using the Detail schema, below the title on these pages, the stylesheet must be updated so that the Detail schema applies the form template after the title.

Usually, this is done through a single, default stylesheet. If custom stylesheets were created for each page type schema, these individual stylesheets must be updated individually.

➲ To Update PageType Match:

1. In the stylesheet, locate the position where the form is to display for a page type.
2. Add
   `<xsl:apply-templates select=”FormBuilder_Form” />`
3. Repeat for each page type that will need to display a form.

☞ This requires file system access to the design time server.

➢ Updating Schemas to Include a Form Component Element

Before a user can choose to include a form on a page through the Ingeniux CMS, the page must contain a component element for the form. Typically, the schema will include code that reads as:

Declaration:

<ElementType content="eltOnly" name="xpowercomponent_Form" />

Instance:

<element type="xpowercomponent_Form" label="Form" CompTypes="FormBuilder_Form" />

Building Forms Pages

Once Forms Builder has been installed and configured on the design time server, forms can be created within the Ingeniux CMS. Whereas configuration occurred on the file system, forms creation occurs through the Ingeniux software interface.

➢ Creating a Form Component

A *form component* contains information about the form. Its role is similar to that of the <*form*> tag in HTML.

➲ To Create a Form Component:

1. Log into the Ingeniux CMS, if necessary.
2. Choose **New**, **Component**.
3. Enter a name for the form component.
4. Select FormBuilder-Form Component and choose **OK**.
5. Enter the form name, with no spaces.
6. Enter a description of the form, if desired.
7. Select the desired method.
8. Enter the location of the desired action.
9. Fill in any additional elements as desired.
10. Save the component.

Form Component Element Reference

Element	Description
Form Name	Defines the form; cannot contain spaces.
Description	Describes the form; can contain spaces.
Method	Determines the method used by the form when passing the information collected in the form. GET: collected values show up in URL, as in a Google search. This method is inappropriate for sensitive data. POST: no collected values appear in URL; values are stored in HTTP body of the request.
Action	Identifies the location of the form processor. Use an absolute reference to point to a form processor living outside the CMS. Use a relative reference to point to a 3rd party application uploaded to the XML\PreBuilt directory on the design time server. Use the xID.xml to point to a form processor page created within the Ingeniux CMS. ☞Refer to instructions below for creating a forms processor within the CMS.
Window	Determines where the resulting response page appears. _parent allows target page to take over the main window, if the form itself is in an iframe _blank displays response in a new window _self displays response in the existing window
AJAX Request	Uses javascript so that the page does not refresh. This option requires that the browser allow javascript. If javascript is not enabled, it becomes a normal form post as a fallback.
Form Appearance	Controls positions of the labels. Width of form represents percentage of the form container. Width of label column can be indicated in percent or pixels. This can be overridden at the field level.
Form Input Fields Navigation	Points to the location of the input elements for the form. Typically this points to the children for the form component. In the case of reusable input fields, it can be pointed to another location using the Start Page value. This element must be set to 2 levels deep to ensure that multiple input options are included when present.

➢ Creating Input Components

Input components represent the individual fields that display in a form. There are three ways in which input fields are used, although they all use the same form input component. Create the input components as children of the form component in which they will be included.

Creating Input Components for Text Input

For those fields that will contain text, an input name and label are needed.

➲ To Create an Input Component for Text:

1. Log into the Ingeniux CMS, if necessary.
2. Choose **New**, **Component**.
3. Enter a name for the input component.
4. Select FormBuilder-Input Component and choose **OK**.
5. Enter the input name, with no spaces.
6. Enter a label.
7. Choose **Text** as input type.
8. Save the component.

Creating Input Components for Button Input

For those fields that will serve as buttons, such as Submit or Reset, different options are chosen when filling out the input component.

➲ To Create an Input Component for a Button:

1. Choose **New**, **Component**.
2. Enter a name for the input component.
3. Select FormBuilder-Input Component and choose **OK**.
4. Skip the input name element.
5. Skip the label element.
6. Choose **Submit**, **Reset**, **Image**, or **Button** as input type.
7. Enter a value for text or image to display on the button.
8. Save the component.

Creating Input Components for Inputs with Fixed Choices

Fields that have a fixed set of responses, such as drop-down lists, checkboxes, and radio buttons, also use the input component. Once an input component is created, *choice components* are created beneath it that provide the list of options to the form.

➲ To Create an Input Component for Use with Choices:

1. Choose **New**, **Component**.
2. Enter a name for the input component.
3. Select FormBuilder-Input Component and choose **OK**.
4. Enter the input name, with no spaces.
5. Enter a label.
6. Choose **Select**, **Multi-Select**, or **Radio** as input type.
7. Save the component.

Once the input component is created, each choice is created beneath it.

➲ To Create a Choice Component:

1. Right-click on the input component in the site tree.
2. Choose **New**, **Component**.
3. Enter a name for the choice component.
4. Select FormBuilder-Radio/Choice Component and choose **OK**.
5. Complete the elements.
6. Save the component.

➢ Configuring Input Components

Certain input fields may have special requirements that go beyond the basic name, label, and type. Each input component can be further configured to control field validation and its look and feel.

Setting Input Component Validation

Validation checks the value entered into an input field by a site visitor to see that the entered value meets certain expectations. If not, a standard error message displays.

Any combination of the following validation settings may be used on an input field.

- Required

  This toggle indicates whether the user must include a value for the field.

- Regular Expression (reg-expression)

  This type of validation uses the public standard for regular expressions. It requires additional information in the form of an expression be entered.

  For example, `\\d{3}-\\d{3}-\\d{4}` validates for a phone number, while `.{1.}@-{1}.\\{2,5}` validates for an email address.

- Range

  This type of validation sets an inclusive range of values in which the entered value must fall. It requires additional information in the form of values separated by commas.

- Compare

  This type of validation confirms against another element on the form. It requires additional information in the form of the name of the other input component to compare against.

Controlling Input Field Look and Feel

There is often a need for an input field to stand apart from the rest of the fields on the form. For example, required fields are often set apart to help cue the site visitor to fill them out.

Formatting can be assigned to an input component by entering an existing CSS class or by specifying exact CSS (for example, `width:200 px; border: 2 px solid gray;`).

☞ It is recommended that formatting changes be handled with a pre-existing CSS class to represent the cause for the exceptional formatting, rather than handling formats at the individual field level.

➢ Exploring Advanced Input Settings

Additional things to consider when working with input components include:

- Multi-column settings only apply to radio fields.
- Don’t overwrite style because CSS is the better solution
- Ignore Start a New Line element
- Visibility Dependency is not implemented, but leave this element in to capitalize on future functionality
- Preset Form Field Value: can pull from any part of the page’s expanded data using xPath query in curly braces. For example, `{/*/IGX.Info/COOKIES/UserName}`.
- Advanced settings include the option to Inherit Previous Form. This is used for creating a series of forms that mimic the behavior of a wizard. When used, the POST or GET inherits values and passes them down.
- Divider components can be used to break up a form into sections or to include additional text in the form. They can also be used to create a heading for a series of related checkboxes.

➢ Associating Form Components with Specific Pages

Once a form component and its related input and choice components have been created, the form can be used on pages within the site.

☞ Form components can only be added to pages whose schemas contain a component element and whose stylesheets have been updated to display the value of the component element. Refer to the instructions for installing and configuring Forms Builder for more information on this prerequisite.

➲ To Associate a New Form with an Existing Page:

1. Check in the desired form component and its corresponding input and choice components.
2. Mark for publish the desired form component and its corresponding input and choice components.
3. Navigate to the page on which the form is desired.
4. Pick the newly created form component for inclusion in the form component element on the page.
5. Preview the page to verify everything displays as expected.

Creating a Form Handler to Process Form Actions

Forms created using Ingeniux can be set up with actions that are external to the Ingeniux environment. It is also possible to create form handling actions within the Ingeniux CMS.

The process for creating a custom action involves:

1. Create a processor page schema.
2. Create SMTP and/or dbQuery components to process the data submitted via the form.
3. Add the SMTP and/or dbQuery components to the processor page.
4. Add the new processor page as the action on the form.

➢ Creating a Processor Page Schema

There are two scenarios for a processor page. In the first case, the processor page will jump to a result page if the form processing is successful. In the second case, AJAX is used, and therefore only one page is needed. However this requires JSON format. And if javascript is disabled, the first scenario occurs. Therefore, it is a good idea to have both a processor page that is not seen by the site visitor, as well as a results page that will display after the form is processed.

A form processor page schema should include a group on the page to hold multiple processor components.

☞ Currently the processor page will continue to submit the page upon refresh. The workaround is to add a redirect element to the processor page stylesheet that reads `<meta http-equiv=”refresh” content=0;url>`.

➢ Creating an SMTP Emailer Component

SMTP Emailer components are used for lead emails (sent to one address, submitted by many) and response emails (sent to many, from one address).

☞ Use of the SMTP Emailer requires proper installation of ComExecute. Refer to the instructions above.

➲ To Create an SMTP Emailer Component:

1. Choose **New**, **Component**.

2. Enter a name for the component.
3. Select SMTP Emailer Component and choose **OK**.
4. Enter the SMTP server name.
5. Verify the correct port is selected.
6. Enter the Sender or Receiver email address.
7. Complete the body of the email.
8. Specify an attachment, if desired.
9. Save the component.

Element	Description
SMTP Server	Use SMTP server provided by network administrator.
SMTP Port	Use SMTP port provided by network administrator.
Sender Email	For lead emails: use generic email or substitute in email entered in form. For POST, `<Form>`*`inputname`*`</Form>` For GET, `<QueryString>`*`fieldname`*`</QueryString>` or `<Cookie>name</Cookie>` (can pull in info when someone is logged in) or `<ServerVariable>`*`name`*`</ServerVariable>` (can pull in browser, location, etc.) For response emails: use specific email.
Receiver Email	For lead emails: use specific email. For response emails: use email entered in the form, as referenced above.
Body	Body of the email is plain text, and can contain spaces and hard returns. If a variable is pulled in from the form, use the following syntax: `<Form>`*`inputname`*`</Form>`
Attachment	Contains the actual path from the root of the runtime server to the desired attachment. Multiple attachments are possible, separated by a semicolon. Attachments on the runtime server should live outside the site directory. Do not use UNC paths for security reasons. DO NOT use mapped drives.

➢ Creating a dbQuery Component

DbQuery components are used to pull or submit data to a database.

➲ To Create a dbQuery Component:

1. Choose **New**, **Component**.
2. Enter a name for the component.
3. Select dbQuery Component and choose **OK**.
4. Enter the DSN as provided by the database administrator.
5. Enter the SQL query string specific to the desired database.
6. Save the component.

Element	Description
DSN	Use DSN as provided by database administrator. Refer to http://www.connectionstrings.com for more on standard ODBC connection strings.
SQL Query String	Use stored procedures as much as possible to prevent SQL injection attacks.

➢ Putting Processor Pages into Action

Once SMTP Emailer or dbQuery components exist, they can be associated with a processor page. The processor page is then referenced from the form component.

☞ A processor page schema needs to be in place before these steps can occur. Refer to information above on creating processor page schemas.

➲ To Put Processor Pages into Action:

1. Check in the desired SMTP Emailer and/or dbQuery components.
2. Mark for publish the desired SMTP Emailer and/or dbQuery components.
3. Navigate to a form processor page created using the processor page schema.
4. Pick the newly created SMTP Emailer and/or dbQuery components for inclusion in the list of processors.
5. Complete any other desired elements on the processor page.
6. Check in the processor page and mark it for publish.
7. Note the xID for the processor page.
8. Navigate to the form component that will use the processor.

9. Check out the form and add the xID.xml for the processor page as value for the action element.
10. Check the form in.