Web Forms for Marketers 2.1

Reference Guide

A Reference Book for Administrators and Developers

Table of Contents

Chapter 1 Introduction 3
Chapter 2 Using the Module 5
2.1 Storing Web Forms 6
2.2 Creating a New Form 7
2.3 Restricting Placeholders Shown in the Placeholder List 8
2.4 Form Field Types 9
2.4.1 List Items 14
Using XPath Query 15
Using Sitecore Query 15
Using Fast Query 15
Localizing List Items 15
2.5 Validations 16
2.6 Submit Actions 18
2.6.1 Form Verification 18
Creating a New Form Verification Action 18
Changing the Form Verification Error Message 19
2.6.2 Save Actions 19
2.6.3 Success 21
2.6.4 Auditing information 21
2.7 Reports 23
2.7.1 Supported Databases 24
2.7.2 Data Request Timeouts 24
2.7.3 Summary 24
2.8 Configuring a User's Access to the Module 25
2.8.1 The Security Roles for Web Forms 25
2.9 Events and the Session Trail 28
2.10 Multi-site Implementation 30
2.11 Multi-server Environment 31
2.11.1 Deploying the Module on the Content Delivery Server 31
2.12 Multiple Sitecore Instances 33
2.13 Running Web Forms in Live Mode 34
Chapter 3 Web Forms Developer's Notes 35
3.1 How to Extend/Override Standard Functionality 36
3.2 How to Add a New Field Type 38
3.3 How to Create a Save Action 43
3.4 How to Use CSS Themes 45
3.5 How to Configure CSS Styles 46
3.6 How to Load Items from the Content Tree to a List Control 48
3.7 How to Export to ASCX 49
3.8 How to Add an ASCX Control to the Page 50
3.9 How to Configure a Data Provider 51
3.10 How to Send SMS/MMS Using a Custom Processor 52
3.11 How to Implement "Required" Checkbox Field 53

Chapter 1

Introduction

This document is designed for Sitecore administrators and contains information about how to set up the module. For more detailed end users instructions, see the Web Form User Guide.

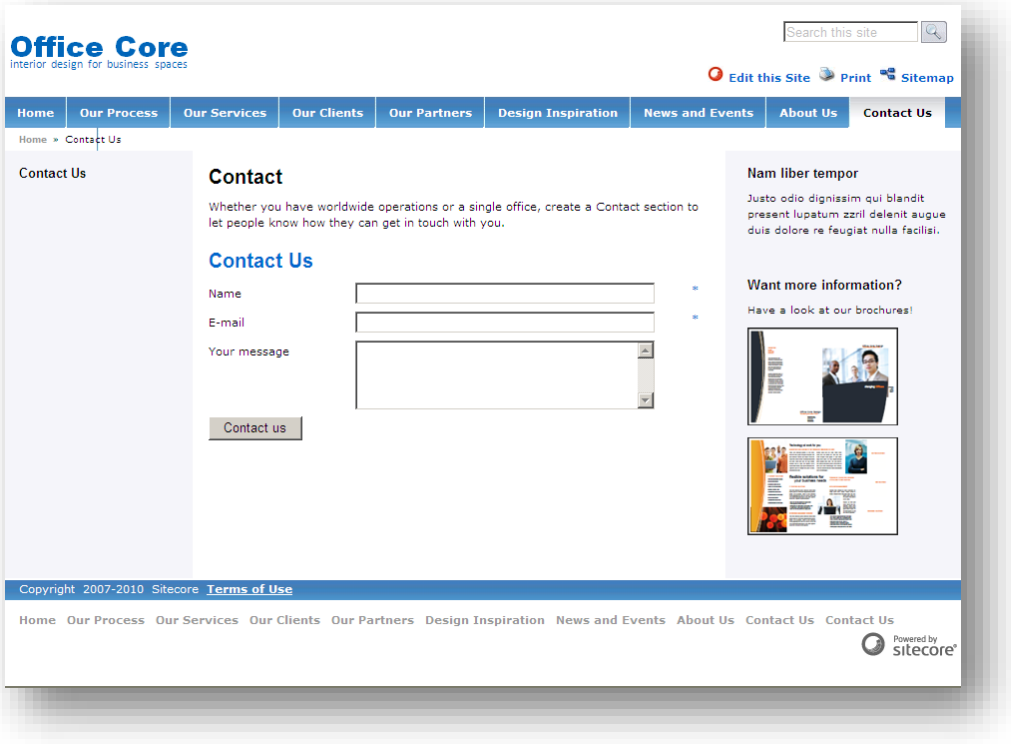
Just as most forms on a Web site are simple and straightforward, the process of creating and managing them should also be simple and straightforward.

The Web Forms for Marketers module is designed to let you create simple forms in a blink of an eye and in a user-friendly manner. The forms that you can create with this module are WCAG 2.0 and XHTML 1.0 compliant. The Web Forms for Marketers module provides users with Web analytics and reporting capabilities. It also records and reports all the information that Web site visitors enter in forms regardless of whether they successfully submit the form or not. The module is fully integrated with the Online Marketing Suite for Sitecore CMS 6.

The module can be configured so that forms have only a few adjustable parameters thereby making the user interface as simple as possible. The basic options cover the needs of the average content editor — creating basic input fields, such as, text boxes and check boxes, creating basic actions, such as, save to a database, send an e-mail, and creating basic validators, such as, RequiredField validator, Email address validator, and so on.

You can also develop more complex forms that are based on a form generated by this module. For example, you can convert a form into a sublayout (`.ascx` file), and then edit this sublayout with development tools like Visual Studio.

Here is an example of a form created with this module:



Chapter 2

Using the Module

This chapter describes the functionality of the module.

This chapter contains the following sections:

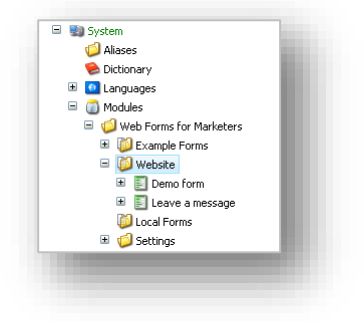
- Storing Web Forms
- Creating a New Form
- Restricting Placeholders Shown in the Placeholder List
- Form Field Types
- Validations
- Submit Actions
- Reports
- Configuring a User's Access to the Module
- Events and the Session Trail
- Multi-site Implementation
- Multiple Sitecore Instances
- Running Web Forms in Live Mode

2.1 Storing Web Forms

The structure of the forms is defined in the Sitecore CMS content tree. User input that is entered in the forms is saved in the module's own database called `Sitecore_WebForms`. This database is located in the `website/data` folder. In the previous version of this module, this database was called `Sitecore.WebForms`.

The Web Forms for Marketers module has its own configuration file called `forms.config`. This file is located in the `website/app_config/include` folder.

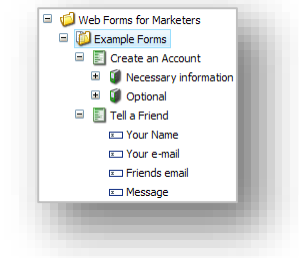
The items that make up forms are stored in appropriate folders under `/sitecore/System/Modules/Web Forms for Marketers`.



You specify where new forms are stored in the `web.config` file —in the `formsRoot` attribute of the `/sitecore/sites/site` node. If the `formsRoot` attribute is not defined for a site, new forms are created in the `/sitecore/System/Modules/Web Forms for Marketers/Local Forms` folder.

The form folders are based on the `/sitecore/Templates/Web Forms for Marketers/Forms Folder` template.

Each form is based on the `/sitecore/Templates/Web Forms for Marketers/Form` template and can contain any number of sections or fields.



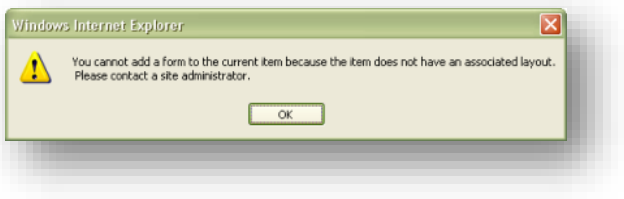
Every form contains a submit button that you can associate actions with. Actions are executed on the server.

The list of available actions is stored under the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Actions` folder.

2.2 Creating a New Form

Users can use the **Page Editor** and the **Content Editor** to create new forms.

When a user clicks the **Insert Form** button, the wizard first checks whether the current item has a layout assigned to it. If this condition is not met, the wizard displays the following message:



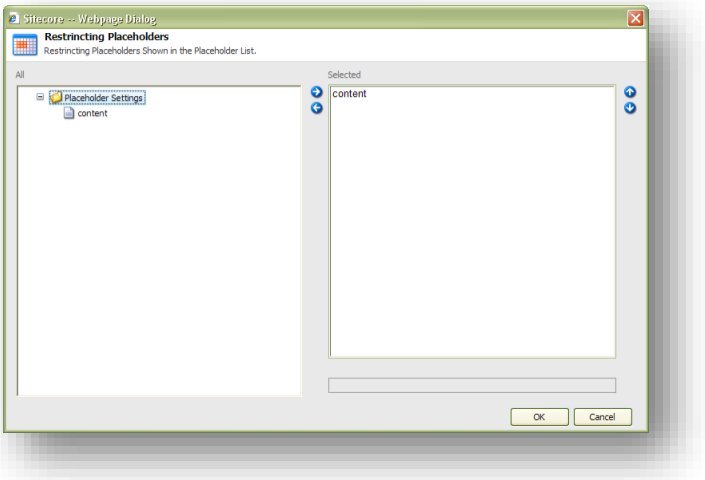
Forms are renderings and should therefore be attached to layouts. For more information about layouts in the Web Forms for Marketers module, see the *Restricting Placeholders Shown in the Placeholder List* section.

Important
If you create a web form that will be used as a "Form to Copy" and define its **Display Name** field value, all the copies will have the same **Display Name** value.

2.3 Restricting Placeholders Shown in the Placeholder List

The **Insert a New Form** wizard only allows you to add forms to placeholders that have “Placeholder Settings” items. A user that add a new form must have write access to an item where a form is added to see placeholders in the “Placeholder list”. This allows developers and Web site administrators to define which placeholders may contain a form.

The **Restricting Placeholders** wizard helps to restrict the list of placeholders shown in the **Placeholder list** of the **Insert a New Form** wizard. To run the **Restricting Placeholders** wizard click *Sitecore*, *All Applications*, *Web Forms for Marketers*, *Form Reports, Restricting Placeholders.* This wizard is also displayed when the Web Forms for Marketers module installation is completed.

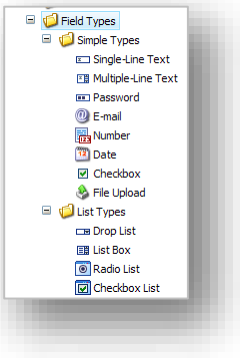


The **Selected** field lists the placeholders where users can add a new form. To add a placeholder from the **All** list to the **Selected** list, select a placeholder and click . When you click **OK** all the changes are saved.

To add a placeholder to the **All** list in the **Restricting Placeholders** dialog box, you must create a new placeholder under the `Sitecore/Layout/Placeholder Settings` item. If you already have a placeholder, you would like to use, create a placeholder with the same name under the *Placeholder Settings* item.

2.4 Form Field Types

The Web Forms for Marketers module contains a number of field types that you can use to build your forms. All the form field types are stored under the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types` item and are based on the `/sitecore/Templates/Web Forms for Marketers/Field Type` template.



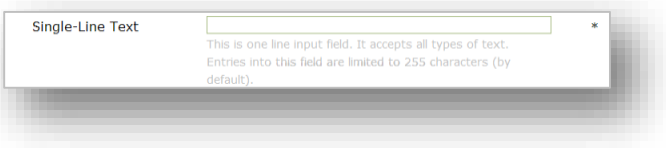
The *Field Type* template contains the following fields:

- **Required** — the field which defines whether the *required value* validator is applied to this field type.
- **Validation** — the list of validators that should be applied to the value entered in this field.
- **User Control** — the field defines a reference to an ASCX control.
- **Deny Tag** — the field defines whether information entered to this field type is saved to the Tag of the *Analytics* database.

The Web Forms for Marketers module contains the following field types:

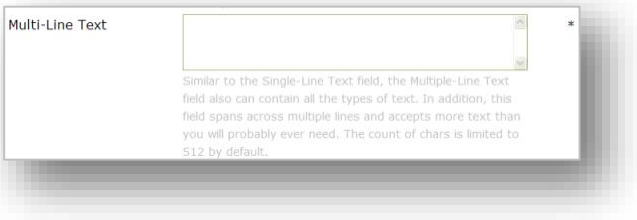
Single-Line Text

Use this field type to enter one line of text. The length of the field is limited to 255 characters by default.



Multi-Line Text

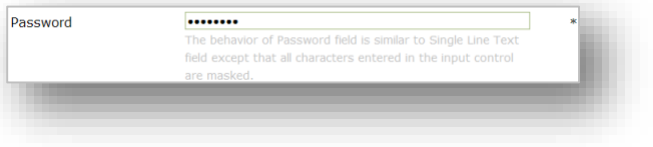
Use this field type to enter multiple lines of text. The number of characters is limited to 512 by default.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/Multiple-Line Text` item contains settings for this field type.

Password

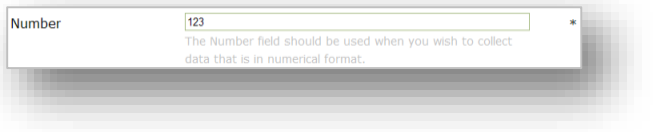
Use this field type to enter a password. All the characters you enter in a **Password** field are masked. The **Password** field is a text field.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/Password` item contains the settings for this field type.

Number

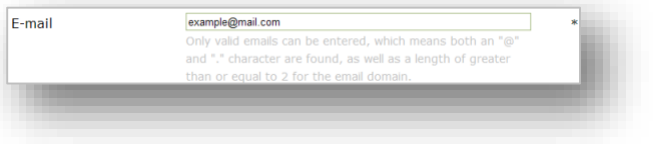
Use this field type to enter numerical data.



The /`sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/Number` item contains the settings for this field type.

E-mail

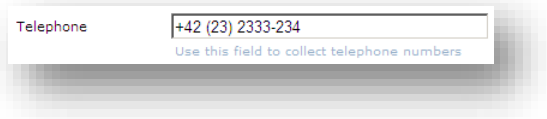
Use this field type to enter e-mail addresses. The "@" and "." characters are validated, as well as the length of the e-mail server domain.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/E-mail` item contains the settings for this field type.

Telephone

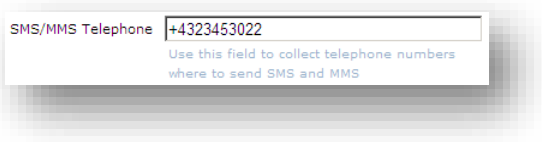
Use this field type to enter telephone numbers. This is a number field which also allows the user to enter the following characters: "+", "-", " (", ")" and spaces.



SMS/MMS Telephone

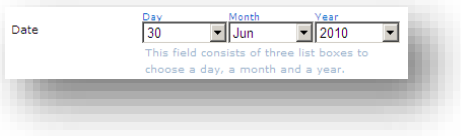
Use this field type to enter telephone numbers that you can send SMSs and MMSs to. This field allows you to enter numbers and the "+" character. The data entered can only contain the "+" character as the

first symbol. This field type is used with the **Send SMS** and **Send MMS** save actions.



Date

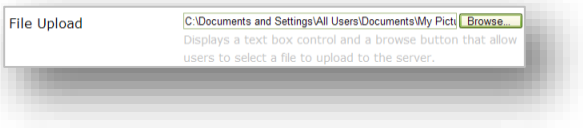
Use this field type to enter dates.



The /`sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/Date` item contains the settings for this field type.

File Upload

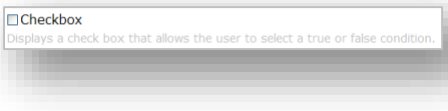
Use this field type to display a text box and a browse button that you can use to select a file that you want to upload to the server.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/File Upload` item contains the settings for this field type. This field works with the master database.

Checkbox

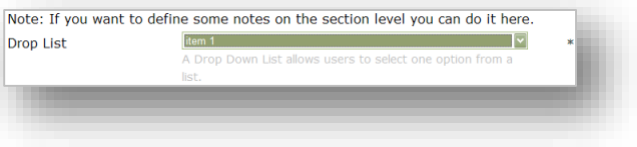
Use this field type to display a check box that allows you to select a true or false condition.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Simple Types/Checkbox` item contains the settings for this field type.

Drop List

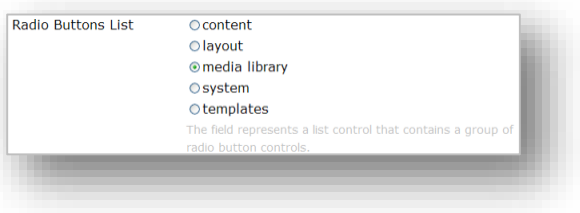
Use this field type to select an option from a list.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/List Types/Drop List` item contains the settings for this field type.

Radio Button List

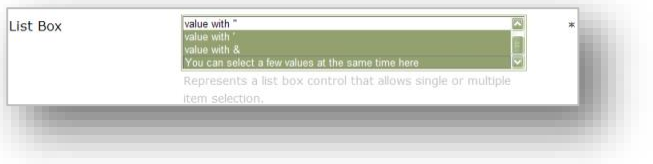
Use this field type to display a group of options.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/List Types/Radio List` item contains the settings for this field type.

List Box

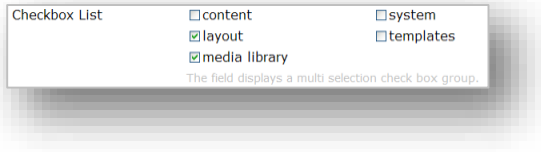
Use this field type to display a list box that allows you to select one or more items.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/List Types/List Box` item contains the settings for this field type.

Checkbox List

Use this field type to display a group of check boxes. You can select one or more check boxes in the list.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/List Types/Checkbox List` item contains the settings for this field type.

Section

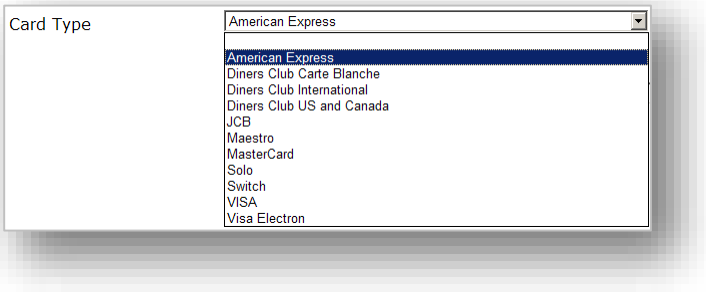
A **Section** is a special field type that you can use as a container for other fields.

Credit Card

This field type contains two fields: one which allows the user to select a credit card type, and another which allows users to enter a credit card number.

This field type validates the credit card number with the possible number ranges and combinations allowed by the different types of credit card. You can specify that validation should be based on the Luhn formula or one of the following credit card types: American Express, Diners Club, Carte Blanche, Diners Club International, Diners Club US and Canada, JCB, Maestro, MasterCard, Solo, Switch, Visa, Visa

Electron.



The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Complex/Credit Card` item contains the settings for this field type.

Password Confirmation

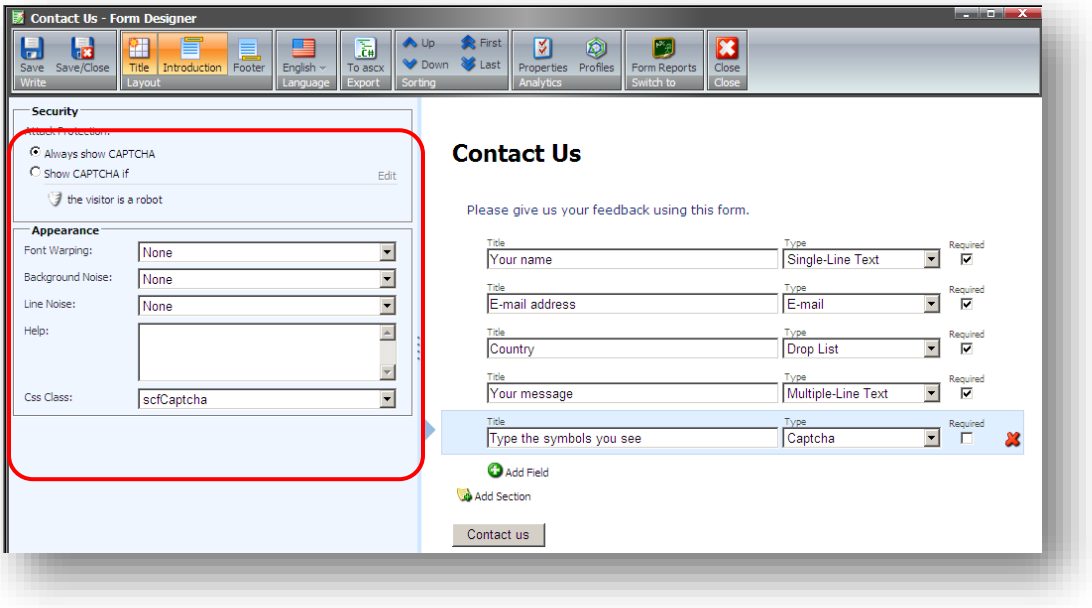
This field type contains two fields: a **Password** field and a **Confirm Password** field. these are used to create a password. Every character entered in these fields is masked.

The `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Complex/Password Confirmation` item contains the settings for this field type.

Captcha

This field type contains two fields: an image field and a text confirmation field. The user should enter the text from the image field in the text field. This can be used to prevent robots registering on Web sites.

In the properties of the form field, you can set the amount of line noise, background noise, and font warp that should be used in the Ccaptcha field:



Users configure the Captcha field using three options:

- The visitor is a robot
- A suspicious visitor is detected
  This option requires the users to enter the thresholds of times and minutes
- Suspicious form activity detected
  This option requires the users to enter the thresholds of times and minutes

You can configure the range of the values that the users can enter in the last two options.

A suspicious visitor is detected

In the `forms.config` file, change the value of the *WFM.SessionThreshold* parameter. By default, the value is `2/1-100/60` — from 2 times in 1 minute to 100 times in 60 minutes.

Suspicious form activity detected

In the `forms.config` file, change the value of the *WFM.ServerThreshold* parameter. By default, the value is `2/1-100/60` — from 2 times in 1 minute to 100 times in 60 minutes.

2.4.1 List Items

List field types include the following field types:

- Drop List
- List Box
- Radio List

- Checkbox List

You can specify the items displayed in a list by:

- Manually entering names
- Selecting Sitecore items
- Using XPath query
- Using Sitecore query
- Using fast query

The first two methods are used mainly by marketers. For more information about manually entering item names and selecting Sitecore items see the Web Forms for Marketers User Guide.

The last three methods can be used to find specific Sitecore items and use them as list field values.

List field types use the value-text concept. This means that list fields' values contain the text that is displayed to the user and the value that is stored in the database. The **Text** field of the list items is usually used to localize forms or to display user-friendly text.

Using XPath Query

This method is used to select Sitecore items with the help of XPath queries. Selected Sitecore items are used as list field values. For more information about XPath Query, visit this Web page.

Using Sitecore Query

This method is used to select Sitecore items with the help of Sitecore queries. Selected Sitecore items are used as list field values. For more information about Sitecore queries, visit the Sitecore Developers Network.

Using Fast Query

The **Using Fast query** method is used to select Sitecore items with the help of Fast queries. Selected Sitecore items are used as list field values. For more information about Fast Query, visit the Sitecore Developers Network.

Localizing List Items

You can translate predefined values of the list items.

To localize list items that are specified using query methods:

1. Translate the same field (for example the **Display Name** field) for all Sitecore items that you will use as list items.
2. In the **Form Designer**, start editing the form and switch to the new language.
3. In the **List Items** Wizard, select the field that you translated as a **Text** value.
   The **Preview** is a language specific, so you will see the translated items immediately.

2.5 Validations

The module contains predefined validations that allow users to add basic validations to fields, as well custom validations. These are explained in the Web Forms for Marketers User Guide.

The module also contains some built-in validations that are used for some of the form field types provided by default in the module.

The default validations are located under the `sitecore/System/modules/Web Forms for Marketers/settings/validation` item.

The default validations are:

Validation	Description
Count chars	Checks the number of symbols in a string. You can set the minimum and maximum number of symbols.
Date	Checks whether or not the value entered is a date.
E-mail	Checks whether or not the value entered uses the format of an e-mail address.
Number	Checks whether or not the values entered are numbers (negative numbers and integers are allowed).
Number range	Checks whether or not the values entered are within a specified range of numbers.
Regex pattern	Checks whether or not the values entered conform to a rule you specify.
Credit card	Checks the validity of a credit card number based on the type of credit card.
Password-Confirmation	Compares the values entered in the **Password** and **Confirmation** fields
Captcha	Compares the text displayed on an image with the value entered by the user

The module installs two validator templates:

- `/sitecore/Templates/Web Forms for Marketers/Validators/BaseValidator`.
- /`sitecore/Templates/Web Forms for Marketers/Validators/Regular Expression Validator`.

The *Regular Expression Validator* template inherits all the fields from the *BaseValidator* template and contains one more field: *Validation Expression*. Validations based on the *Regular Expression Validator* template use validation expressions. Validations based on the *BaseValidator* template use validations defined in classes.

A validator item contains the following fields:

Field	Description
Class	The full name of the class which handles the validation.

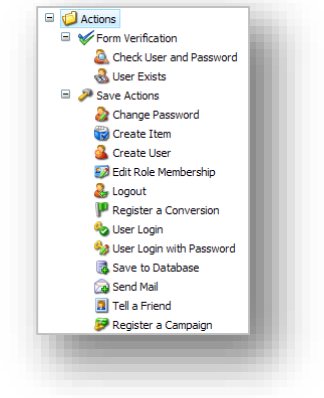
Field	Description
Assembly	The name of the assembly that contains the class.
Error Message	The text for the error message displayed in a *ValidationSummary* control when validation fails. **Note**: If your website is multilingual, add required language versions to the validator item and translate this field.
Text	The text displayed in the validation control when validation fails. **Note**: If your website is multilingual, add required language versions to the validator item and translate this field.
Static Display	The display behavior of the error message in a validation control: • *None* — the validation message is never displayed inline. • *Static* — space for the validation message is allocated in the page layout. • *Dynamic* — space for the validation message is dynamically added to the page if validation fails.
Enable Script Validation	Whether or not client-side validation is enabled.
Parameters	The additional parameters for the validation control.
Inner Control	Indicates the place where the validation control is added.
Validation Expression	Sets the regular expression assigned to be the validation criteria. (Only for validation items that use the *Regular Expression Validator*.)

2.6 Submit Actions

When a Web site visitor clicks Submit, three types of actions are performed:

- Form Verification
- Save Actions
- Success

All the actions are stored under the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Actions` item.



Note:
If you upgrade from the previous version of the Web Forms for Marketers Module, your existing actions are moved to the **Save Actions** folder.

You can configure the following fields of an action:

- **Class** — the full name of the class that processes the form data on the server side.
- **Assembly** — the name of the assembly that contains the class referred to.
- **Parameters** — the parameters for the action irrespective of the parameters of the form. These are the global parameters, common to all the forms. For example, in the *Send Mail* save action, this field is used to specify SMTP server to send e-mails.
- **Editor** — provides the application that you use to change the parameters of the action.

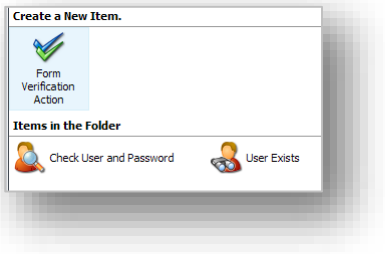
2.6.1 Form Verification

Form verification verifies the values that have been entered in one or more form fields. If a form verification fails, the visitor is returned to the form and an error message is displayed. No other subsequent form verifications or save actions are performed.

Creating a New Form Verification Action

To create a new form verification action:

4. In the **Content Editor**, select the `sitecore/content/system/modules/web forms for marketers/settings/actions/form verification` item.
5. Create a new form verification action item and add the appropriate parameters.



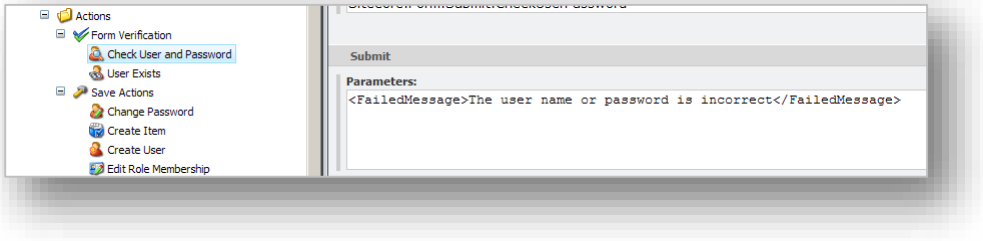
Creating a new form verification action is similar to the creating new save action. For more information about creating a save action, see the *How to Create a Save Action* section.

Changing the Form Verification Error Message

You can set default form verification error message and localized one. For more information about how to set the localized error message, see the Web Forms for Marketers User Guide.

To set default error message:

1. In the **Content Editor**, select the form verification action that you want to edit.
2. In the **Parameters** field, enter an error message in the `<FailedMessage>` tag.



2.6.2 Save Actions

Save actions can be assigned to a form and are executed when a visitor clicks **Submit** on a form. There are 18 save actions in the Web Forms for Marketers module, by default.

A save action is stored as a Sitecore item and executes an action implemented in a .net class. This class is specified in the *Class* field of the save action item. Save actions items located under the `sitecore/content/system/modules/web forms for marketers/settings/actions/save actions` item.

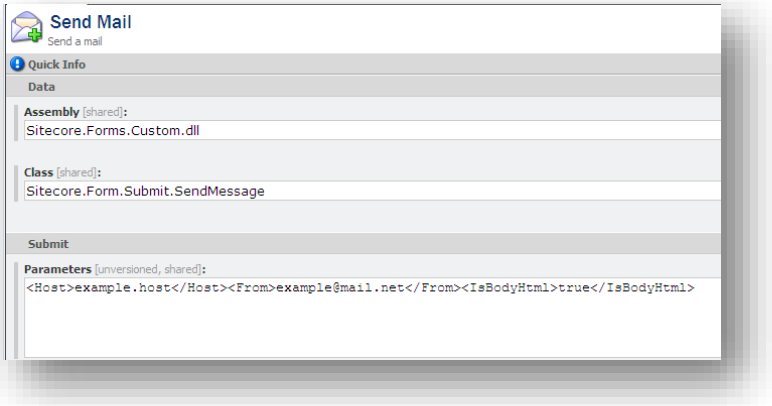
You can create a custom save action. For more information about creating a save action, see the *How to Create a Save Action* section.

There are three save actions that send notifications to a user: *Send Mail*, *Send SMS* and *Send MMS*. All these save actions use default SMTP server settings specified in the `web.config` file. You can specify different SMTP server settings for each save action.

Send Mail

To use different SMTP settings for this action, in the **Content Editor**, navigate to *Sitecore/System/Modules/Web Forms for Marketers/Settings/Actions/Save Actions/Send Mail*, in the **Parameters** field, set the appropriate values.

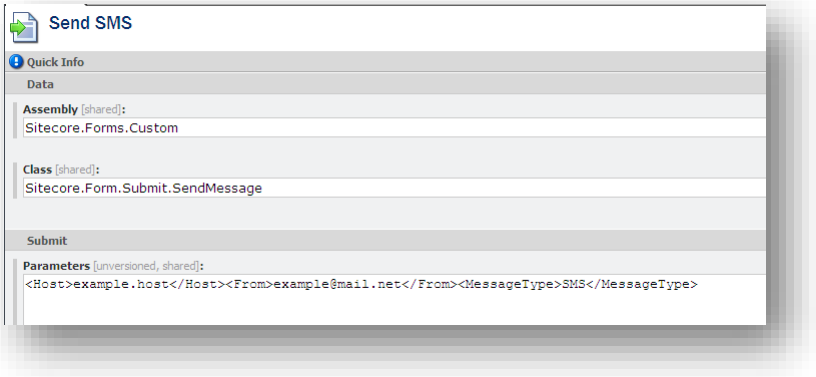
For example, to use *Host* and *From* parameters that are different from one defined in the `web.config` file, enter the following code into the **Parameters** field:



Send SMS

To use different SMTP settings for this action, in the **Content Editor**, navigate to *Sitecore/System/Modules/Web Forms for Marketers/Settings/Actions/Save Actions/Send SMS*, in the **Parameters** field, set the appropriate values.

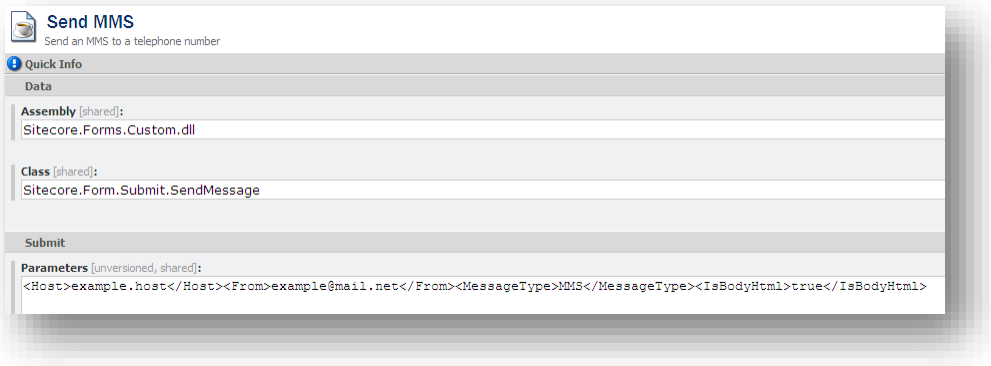
For example, to use `Host` and `From` parameters that are different from one defined in the `web.config` file, enter the following code into the **Parameters** field:



Send MMS

To use different SMTP settings for this action, in the **Content Editor**, navigate to *Sitecore/System/Modules/Web Forms for Marketers/Settings/Actions/Save Actions/Send MMS*, in the **Parameters** field, set the appropriate values.

For example, to use `Host` and `From` parameters that are different from one defined in the `web.config` file, enter the following code into the **Parameters** field:



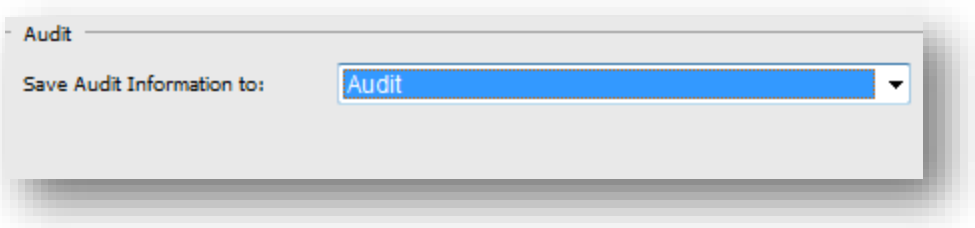
2.6.3 Success

This action allows you to select either a Sitecore item or a message which is presented to a Web site visitor after they successfully submit a form. This is the final action performed in a form submission. The pipeline attached to this is called `successAction` pipeline.

2.6.4 Auditing information

Some default save actions create or edit users or roles in Sitecore's security model. These are referred to as *security actions*. These include the *Create User, Edit Role Membership, User Login*, and *User Login with Password* save actions. As all of these can affect user information, the ability to register audit information in the user profile to record what actions have been performed is useful.

The security actions all have a **Save Audit Information to:** dropdown list which list the field in the user profile to which audit information can be written.

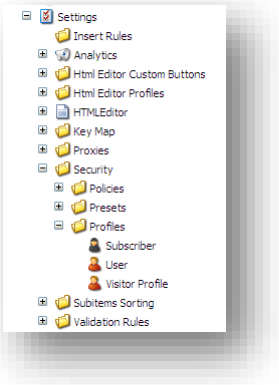


By default all rich text, html, text, memo, multi-line text, and single-line text fields can be used to register audit information. The field types in which audit information can be registered can be configured using the `WFM.AuditAllowedTypes` setting in the `forms.config` file:

```
<setting name="WFM.AuditAllowedTypes" value="|Rich Text|html|text|Multi-Line Text|Single-Line Text|memo|" />
```

All user profiles are items in the Core database under the */sitecore/system/Settings/Security/Profiles*

folder:



The *Visitor profile* is used by default. Each form item has a reference to this user profile in the *User Profile* field.

2.7 Reports

You can see information how Web site visitors interact with forms in the various reports that come with the Web Form module.

The Web Forms for Marketers module supports the creation of the form dropout reports which contain information about users who did not successfully submit forms. The functionality is achieved using AJAX features by tracking each individual field and recording user input in the *Analytics* database.

System events are recorded in the session trail. For more information about system events, see the *Events and the Session Trail* section.

To open the form reports:

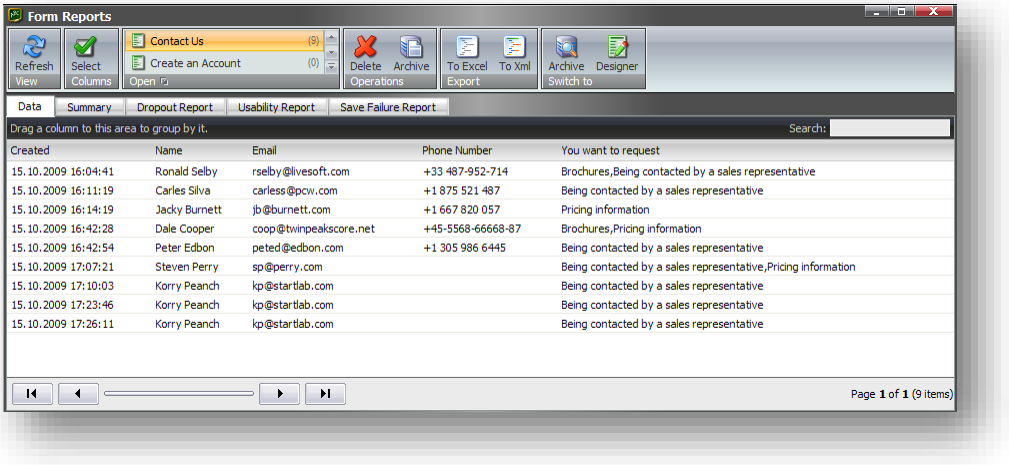
1. Log in to the Sitecore desktop
2. Click *Sitecore*, *All Applications*, *Web Forms for Marketers*, *Form Reports*
3. Select the form for which you want to view reports.

Alternatively, in the **Content Editor**, select the form under the `/sitecore/System/Modules/Web Forms for Marketers/Website` item and click **Form Reports**.

The following reports are available:

- Data
- Summary
- Dropout Report
- Usability Report
- Save Failure Report

The form reports are displayed on the different tabs in the **Form Reports** window:



All the reports display the last 200 records by default. To change this number, open the report in the Web Reports Designer and change the value in the appropriate SQL query.

2.7.1 Supported Databases

The Web Forms for Marketers module reports support the following database servers:

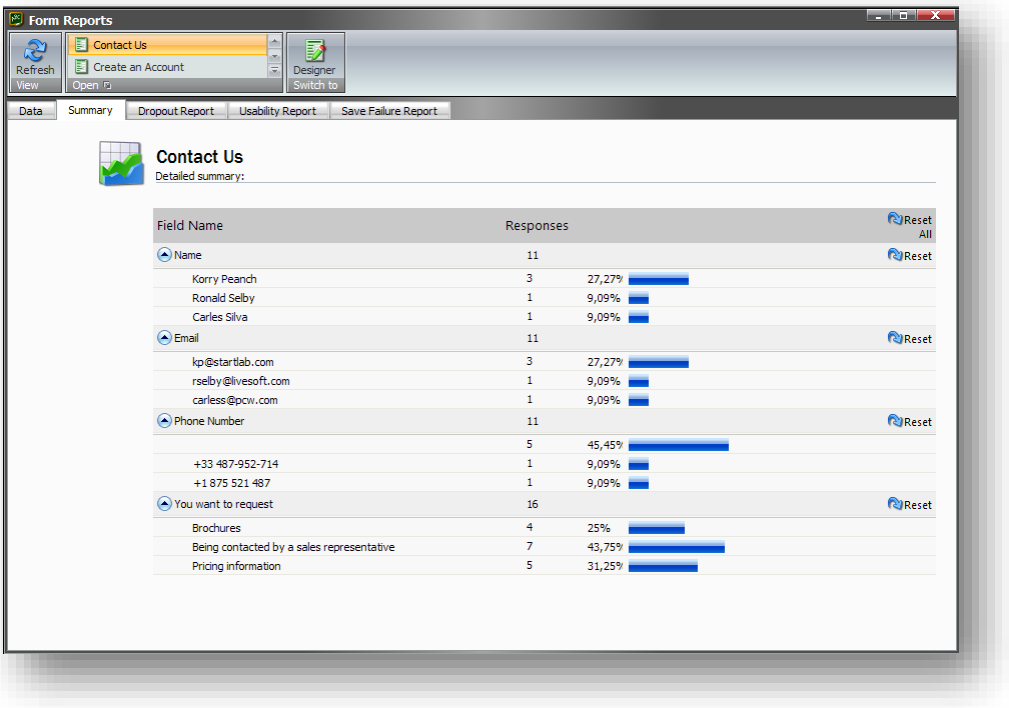
Report	Supported Database
Data	MSSQL, SQLite
Summary	MSSQL, SQLite
Dropout Report and sub-reports	MSSQL, Oracle
Usability Report and sub-reports	MSSQL, Oracle
Save Failures Report and sub-reports	MSSQL, Oracle

2.7.2 Data Request Timeouts

By default the Data report request timeout is 180 seconds. You can modify this value by changing the `WFM.CommandTimeout` parameter in the `\WebSite\App_Config\Include\forms.config` file.

2.7.3 Summary

Only three records are displayed in this report by default.



To change the number of records displayed in this report, in the `\WebSite\App_Config\Include\forms.config` file, change the `WFM.MostPopularApplicantCount` parameter.

If the value of any field exceeds 80% then the blue stripe turns green by default. You can modify this value by changing the `WFM.RelevantScale` parameter in the `\WebSite\App_Config\Include\forms.config` file.

2.8 Configuring a User's Access to the Module

To configure the access that a user has to the features in the module, an administrator can assign the following roles to the user:

- Sitecore Client Form Author
- Sitecore Client Developing
- Analytics Maintaining
- Analytics Reporting
- Sitecore Marketer Form Author
- Sitecore Client Securing

To give the user minimum access rights to the module, assign the *Sitecore Client Form Author* role to the user.

To give the user access to all the features in the module, assign the following roles to the user:

- Sitecore Marketer Form Author
- Sitecore Client Developing
- Sitecore Client Securing

2.8.1 The Security Roles for Web Forms

Here is a short description of the security roles that have been created for the Web Forms for Marketers module:

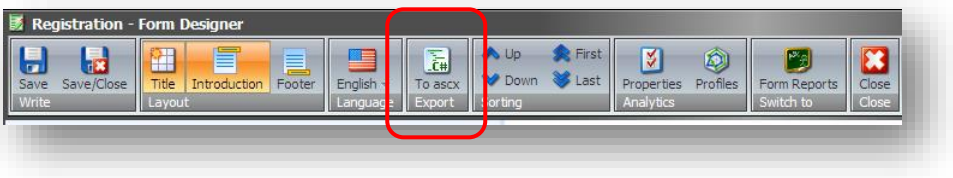
Sitecore Client Form Author

The *Sitecore Client Form Author* role gives the user access to the minimum features of the Web Forms for Marketers module. All the other roles expand the user's access rights. This role allows the user to:

- Insert a new form.
- Edit an existing form.
- View the *Data* report.
- View the *Summary* report.

Sitecore Client Developing

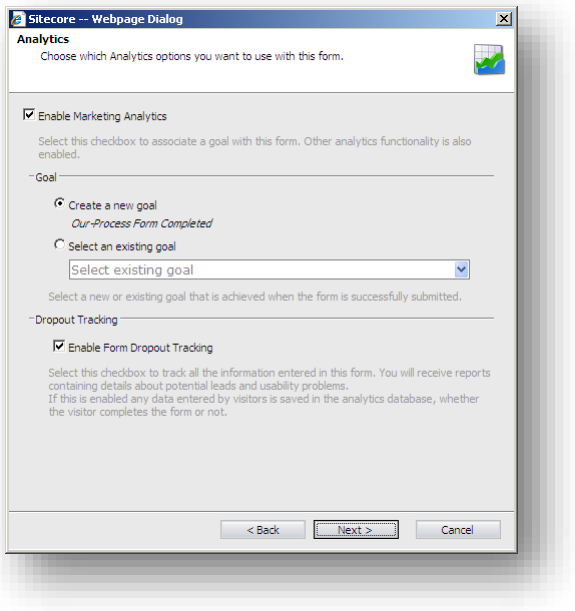
The *Sitecore Client Developing* role allows the user to use the **Export to ASCX** button in the **Form Designer**:



Analytics Maintaining

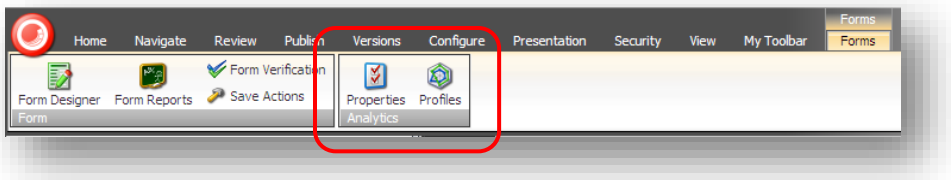
The **Analytics Maintaining** role allows the user to:

- Use the **Analytics** page in the **Create a New Form** wizard:

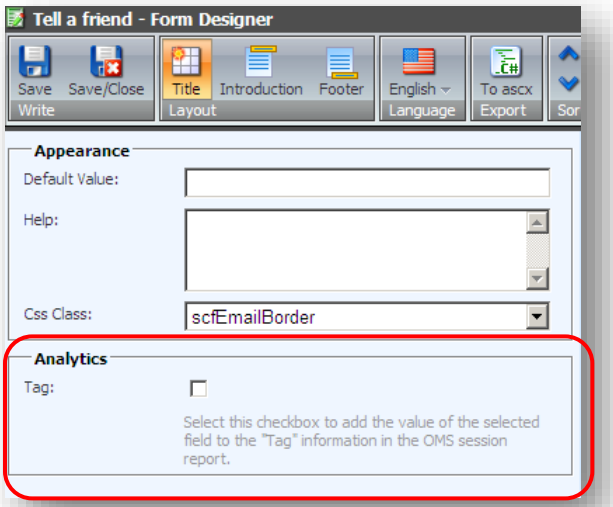


- Use the **Analytics** section on the **Form Designer** ribbon:

- Use the **Analytics** section on the **Content Editor** ribbon:



- Use tags:



Analytics Reporting

The *Analytics Reporting* role allows the user to:

- View the Dropout report.
- View the Usability report.
- View the Save Failures report.

Sitecore Marketer Form Author

The *Sitecore Marketer Form Author* role inherits access rights from the following roles:

- Sitecore Client Form Author
- Analytics Maintaining
- Analytics Reporting

Members of the *Sitecore Marketer Form Author* role have all the rights assigned to these roles.

Sitecore Client Securing

The *Sitecore Client Securing* role allows the user to:

- Edit the *Create User* save action.
- Edit the *Edit Role Membership* save action.
- Edit the *Change Password* save action.

2.9 Events and the Session Trail

The session trail is an OMS feature which records all the activities a user has performed on a Web site, including what pages they have visited and when.

The session trail reports also list the events that are triggered when a visitor fills in a form. Those events are standard OMS page events that can be used for development purposes. The reports list both field events and form events.

For all the events, the **Datakey** field in the Analytics database equals the Form GUID, the exceptions are: Form check action error, Form submit, and Form conversion events.

Several field events can occur in the same field during the same attempt to submit the form, as users change the information in the field.

The events that are listed in the form reports are:

Field Completed

This event is triggered when a field on a form is completed and tabbed or clicked out of. This is done with AJAX.

Field Not Completed

This event is triggered when validation of a required field fails because the field has not been filled in by the visitor.

Field Out of Boundary

This event is triggered when validation of a field fails because the value entered falls outside the boundaries defined for the field. This event is used for the Min and Max Length of Text and Password fields, as well as for Date and Number fields.

Form Check Action Error

This event is triggered when a Check Action fails. This is not a failure, but an event. If a Check Action fails, the visitor is returned to the form. No other Check Actions or save actions are initiated.

Form Save Action Failure

This event is triggered when a save action fails. This is a failure and not an event — the *IsFailure* property is enabled).

Form Submit

This event is triggered when a visitor clicks the **Submit** button or presses ENTER. This indicates an attempt to submit the form.

Invalid Field Syntax

This event is triggered when validation of a field fails because the data entered in the field fails a particular syntax check. The event is used for the following field validations: Regular Expressions in Text and Password types, and Email fields.

Submit Success

This event is triggered when a Submit action does not return an error. The event is written in along with the Form Conversion event and is primarily used to facilitate the SQL statements required for the *Form Dropout*, *Form Usability*, and *Form Failures* reports.

Form Conversion

The event is triggered when a form is successfully submitted and is triggered after the Form Submit event. It also indicates that the goal associated with the form has been successfully completed.

2.10 Multi-site Implementation

The Web Forms for Marketers module supports multi-site environments. This means that administrators can define different form locations and settings for different Web sites. This is done in the `formsRoot` attribute in the definition of the site in the `.config` files.

The value of this attribute is a Sitecore path which defines:

- The folder that stores the forms on the current site.
- The appearance and color settings for forms on the current site.
- The access rights.

The `formsRoot` parameter must contain either an item path or the ID of the target item. The target item must be based on the `/sitecore/Templates/Web Forms for Marketers/Forms Folder` template.

For example, this is how the `formsRoot` parameter is defined in `web.config` file:

```
<sites>
 <site
     name="samplesite"
     virtualFolder="/"
     physicalFolder="/"
     rootPath="/sitecore/content"
     startItem="/forms" database="web" domain="extranet"
     formsRoot="/sitecore/System/modules/Web Forms for Marketers/local forms"
     ...
```

This can be defined in the `forms.config` file using the ID:

```
 <site name="website">
   <patch:attribute name="formsRoot">
     {F1F7AAB6-C8CE-422F-A214-F610C109FA63}
   </patch:attribute>
 </site>
</sites>
```

We recommend that you define the `formsRoot` parameter in the `/App_Config/forms.config` file. This approach ensures that there are no duplicated values.

When the `formsRoot` attribute is not defined for a Web site, new forms are stored in the `/sitecore/System/Modules/Web Forms for Marketers/Local Forms` folder.

2.11 Multi-server Environment

You can scale the Web Forms for Marketers module by configuring a multi-server environment. In such a configuration, the module partially delegates execution of save actions from the content delivery server to the master one.

To configure a multi-server environment:

4. Install the module on the master server. For more information about installing the module, go to the SDN website.
5. Deploy the module on all content delivery servers.

2.11.1 Deploying the Module on the Content Delivery Server

When the module is installed on the master server, deploy it on all content delivery servers.

To deploy the module on the content delivery server:

6. Publish the Master database to all the content delivery servers. For more information about publishing the Master database, see the Scaling Guide*, 3.1 Configuring a Remote Publishing Target*.
7. Deploy the module files to the content delivery servers.

   If content delivery servers have not the client interface, copy the following files and folders from the content management server to all the content delivery ones:

   - Bin\sitecore.forms.core.dll
   - Bin\Sitecore.Forms.Custom.dll
   - Bin\MSCaptcha.dll
   - Bin\Sitecore.Captcha.dll
   - Bin\system.data.sqlite.dll
   - App_Config\Include\forms.config
   - App_Config\Include\Captcha.config
   - Sitecore modules\shell\Web Forms for Marketers\Themes*.*
   - Sitecore modules\web\Web Forms for Marketers\scripts*.*
   - Sitecore modules\web\Web Forms for Marketers\Tracking.aspx
   - Sitecore modules\web\Web Forms for Marketers\control*.ascx
   - Sitecore modules\web\Web Forms for Marketers\UI\UserControl*.ascx

   If content delivery servers have the client interface, install the Web Forms for Marketers package using the Installation Wizard on the content delivery servers.
8. Add the following connection string to the `connectionstrings.config` file in all content delivery servers:

```
          <add name="remoteWfmService"
connectionString="url=http://[masterserver]/sitecore%20modules/shell/Web%20Forms%20for%20Marketer
s/Staging/WfmService.asmx;user=[domain\username];password=[password];timeout=60000" />
```

   Where

`[masterserver]` – IP or the host name of the master server.

`[domain\username]` – Sitecore user (full name). The necessary access rights depend on the items that the form is using.

`[password]` – user's password.

Note

After you added the *remoteWfmService* connection string to the `connectionstrings.config` file in all content delivery servers, all save actions with the cleared **Client Action** check box are executed on the content management server instead on the current server. All uploaded files are uploaded to the content management server too. For more information about the **Client Action** check box, see the section ***Error! Reference source not found.***.

2.12 Multiple Sitecore Instances

The Web Forms for Marketers module supports using multiple Sitecore instances. If you have Sitecore CMS 6.2 or earlier and are using staging or Sitecore CMS 6.3 and using multiple instances, then the following configuration is required.

Add a new connection string that points on your master server to the `connectionstrings.config` file on the slave.

```
          <add name="remoteWfmService"
connectionString="url=http://localhost/sitecore%20modules/shell/Web%20Forms%20for%20Marketers/Staging/WfmService.asmx;user=[domain\username];password=[password];timeout=60000" />
```

Where

`[domain\username]` = a Sitecore domain and a Sitecore user name. The access rights required depend on the items the form is using.

`[password]` = a user's password.

2.13 Running Web Forms in Live Mode

Sitecore supports running a Web site directly from the master database, referred to as — “running in live mode”. Running in live mode eliminates the need to publish content and is similar to viewing a site in the Preview client. A Web site that is configured to run in live mode acts exactly like a normal Web site. Live mode respects all publishing restrictions and workflows in the same way that a default Web site supports these features.

To run the Web Forms for Marketers module in Live Mode, you must edit the `web.config` file:

1. In the `web.config` file, find the relevant `<site>` section and change `database="web"` to `database="master"`.
2. Find the `<modules_website>` section and change `database="web"` to `database="master"`.

The `<sites>` section of your `web.config` file should look like this:

```
<sites>
...
      <site name="modules_website"...database="master".../>

      <site name="website"...database="master".../>
...
</sites>
```

Chapter 3

Web Forms Developer's Notes

This chapter contains information for developers on how to customize the Web Forms for Marketers module.

This chapter contains the following sections:

- How to Extend/Override Standard Functionality
- How to Add a New Field Type
- How to Create a Save Action
- How to Use CSS Themes
- How to Configure CSS Styles
- How to Load Items from the Content Tree to a List Control
- How to Export to ASCX
- How to Add an ASCX Control to the Page
- How to Configure a Data Provider
- How to Send SMS/MMS Using a Custom Processor

3.1 How to Extend/Override Standard Functionality

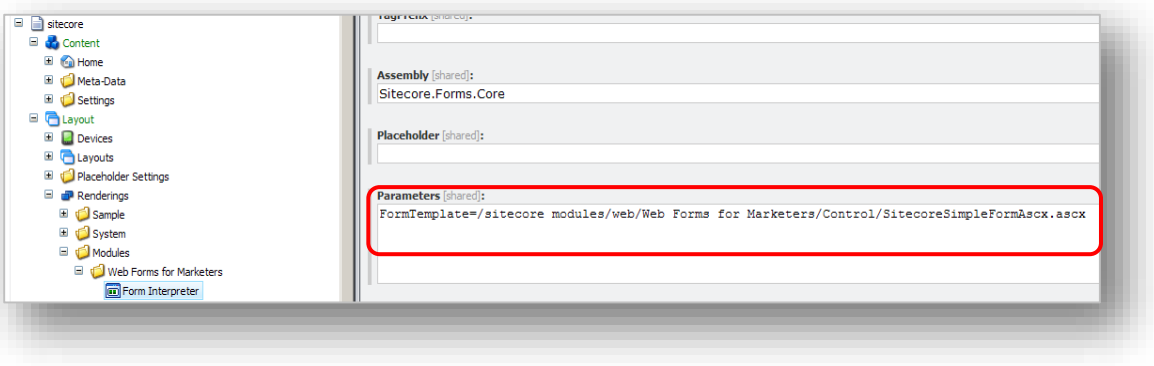
The Web Forms for Marketers module allows you to extend existing functionality.

Layout

The layout of the form is defined in the `sitecore\modules\Web\Web Forms for Marketers\Control SitecoreSimpleFormAscx.ascx` file. This means that you can modify the form structure. You can add new controls (for example, a **Cancel** button) to the form or change the placement of an existing control (for example, change the position of the panel showing error messages).

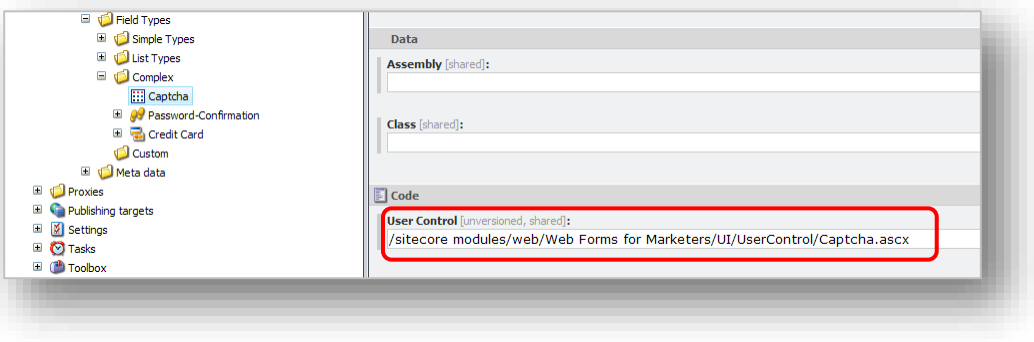
Template

You can change the default form template in the **Parameters** field, in the `/sitecore/layout/Renderings/Modules/Web Forms for Marketers/Form Interpreter` item.



Field Controls

The Web Forms for Marketers module supports `ascx` field controls. You should type the path to an `.ascx` control to use it in the form.



You can change the appearance of the following `.ascx` field controls:

- Password-Confirmation
- Credit-Card
- Captcha

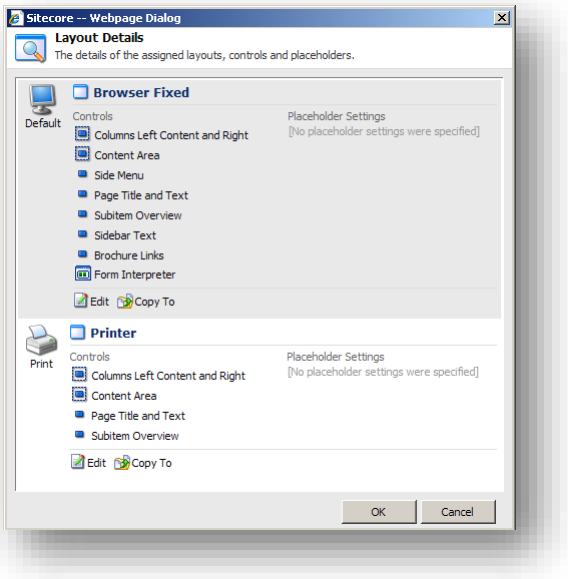
These are located in the `sitecore\modules\Web\Web Forms for Marketers\UI` folder.

Form Behavior

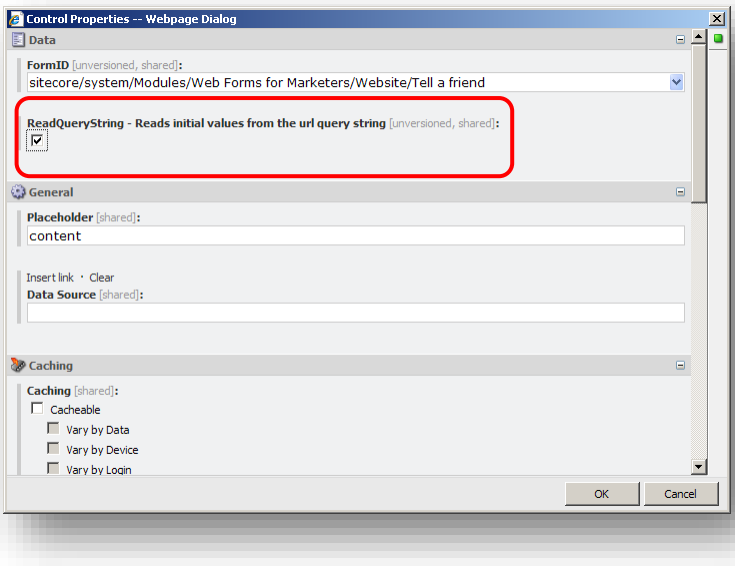
The *Form Interpreter* rendering contains some settings that influence the behavior of the form. The `ReadQueryString` parameter defines whether or not the initial values for this form are taken from the URL query string.

To change the `ReadQueryString` parameter:

1. Select the item and in the **Presentation** tab, click **Details** and the **Layout Details** dialog box appears.



2. In the **Layout Details** dialog box, click **Form Interpreter**.
3. In the **Control Properties** window, select the **ReadQueryString** check box.



Use the following format of the URL query string:

```
Field name1=field value1&Field Name2=field value2
```

3.2 How to Add a New Field Type

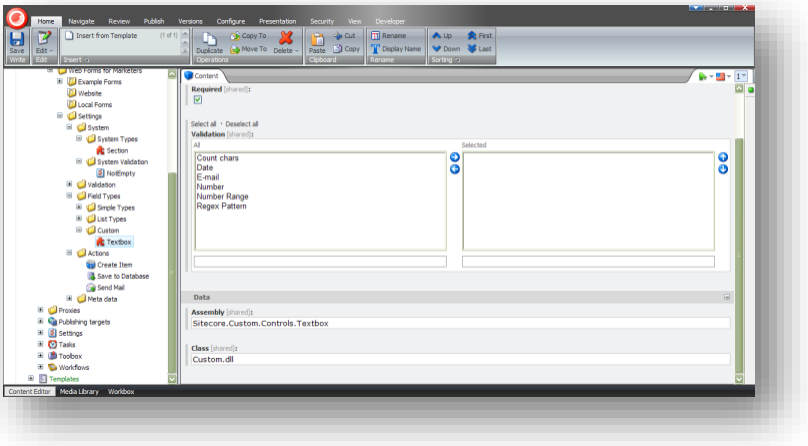
You can extend the list of field types by creating a class that renders the field and registers it as Sitecore content.

After you use `/sitecore/Templates/Web Forms for Marketers/Field Type` template in the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Custom` folder to create a new item, you should complete the associated fields.

For example, in the `Custom.dll` assembly, create a new control called `Sitecore.Custom.Controls.Textbox`:

```
namespace Sitecore.Custom.Controls
{
   public class Textbox
   {
…
   }
}
```

Create a new item in the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Field Types/Custom` folder and set its **Class** and **Assembly** fields.



Select the **Required** check box if you want your control to use the `Required Value` validator. In the Validation section, select any other validations that you require.

If your control must return a value to the server, the control should implement the `IResult` interface:

```
namespace Sitecore.Form.Web.UI.Controls
{
   public interface IResult
   {
      ControlResult Result { get; }

      string ControlName { get; set; }

      string FieldID { get; set; }
   }
}
```

- `Result` — returns the value of the field in the object of the `ControlResult` class. The constructor of the `ControlResult` has the following signature:

```
public ControlResult(string fieldName, object value, string parameters)
```

where

- `fieldName` — the name of the field, defined only for the customization.
- `value` — the value of the field.
- `parameters` — additional parameters.

If the returned value is not a string, you must set an adapter for it. You can do this by using the `Adapter` attribute before the class definition.

```
namespace Sitecore.Form.Web.UI.Controls
{
   [Adapter(typeof (DateAdapter))]
   [ValidationProperty("Value")]
   public class DateSelector : ValidateControl, IHasTitle
```

The type of this attribute must inherit the `Adapter` abstract class.

```
using System.Collections.Generic;

namespace Sitecore.Form.Core.Client.Submit
{
   public abstract class Adapter
   {
      // Convert object value to string
      public virtual string AdaptResult(object value)
      {
         return value.ToString();
      }

      // Convert value from database to friendly value. Using in the Form Data Viewer
      public virtual string AdaptToFrendlyValue(string value)
      {
         return value;
      }

      // Convert list value from database to friendly value. Using in the Form Data
Viewer
      public virtual IEnumerable<string> AdaptToFrendlyListValues(string value)
      {
         return new List<string>(new[]{value});
      }
   }
}
```

- `ControlName` — the name of the item that corresponds to this field is moved here.
- `FieldID` — initialized by the id of the item that corresponds to this field.

You can also use the `Sitecore.Form.Web.UI.Controls.BaseControl` abstract class that implements the `ControlName` and `FieldID` properties of the `IResult` interfaces.

The control must implement the `IHasTitle` interface if you want the control to be iniatilized with the title that you define in the **Form Designer**.

```
namespace Sitecore.Form.Web.UI.Controls
{
   public interface IHasTitle
```

```
        {
            string Title { set; }
        }
    }
```

If you would like properties from your control to be available in the **Form Designer**, you must use the following special attributes:

- `VisualProperty` means that the property is displayed in the **Form Designer**.

  Constructors:

```
public VisualPropertyAttribute(string displayName)

public VisualPropertyAttribute(string displayName, int sortorder)
```

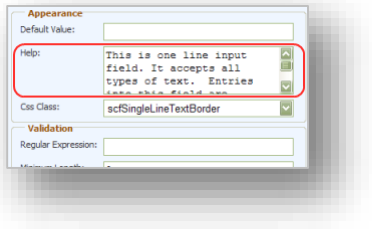
  where

- `displayName` — defines the name of property in the designer.
- `sortorder` — sets the position of the sorting in the designer.
- `VisualCategory` — sets a category name.
- `VisualFieldType` — defines the type of the input control for the property. If the attribute is not specified the property has the EditField input type.
- `DefaultValue` — sets an initial value in the designer.
- `Localize` — defines whether the property supports multiple languages or not.

For example, the definition of the following property:

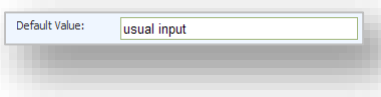
```
[VisualProperty("Help:", 500)]
[VisualCategory("Appearance")]
[VisualFieldType(typeof(TextAreaField)), Localize]
public string Information
{
 ...
}
```

is shown:

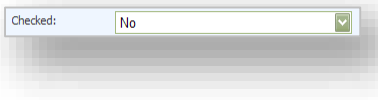


The `VisualFieldType` attribute gives the following values to the different field types:

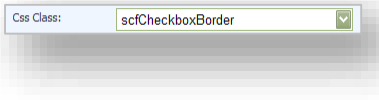
- An EditField looks like usual a input control:



- A BooleanField always contains 2 values: Yes and No.

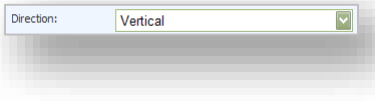


- A CssClassField allows you to choose one of the available css classes.

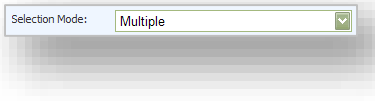


- A DateFormatField contains the permitted date formats.

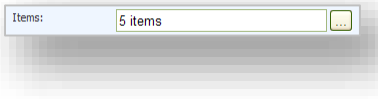
- A DirectionField always contains 2 values: Vertical and Horizontal.



- A SelectionModeField contains 2 values: Single and Multiple.

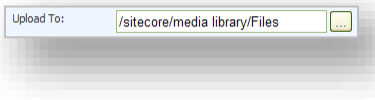


- A ListField allows you to input items for list controls.

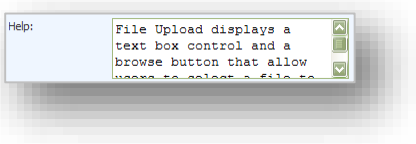


- A SelectDateField allows you to select a date

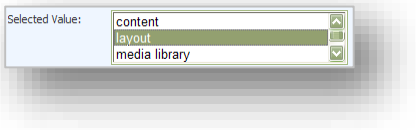
- A SelectDirectoryField allows you to select a path in the content tree



- A TextAreaField looks like a big input control



- A MultipleSelectedValueField works with a ListField field and allows you to set selected values for list items



- An EmptyChoiceField works with a ListField field and allows you to add an empty choice to list items

3.3 How to Create a Save Action

To create a save action, for instance, a login action:

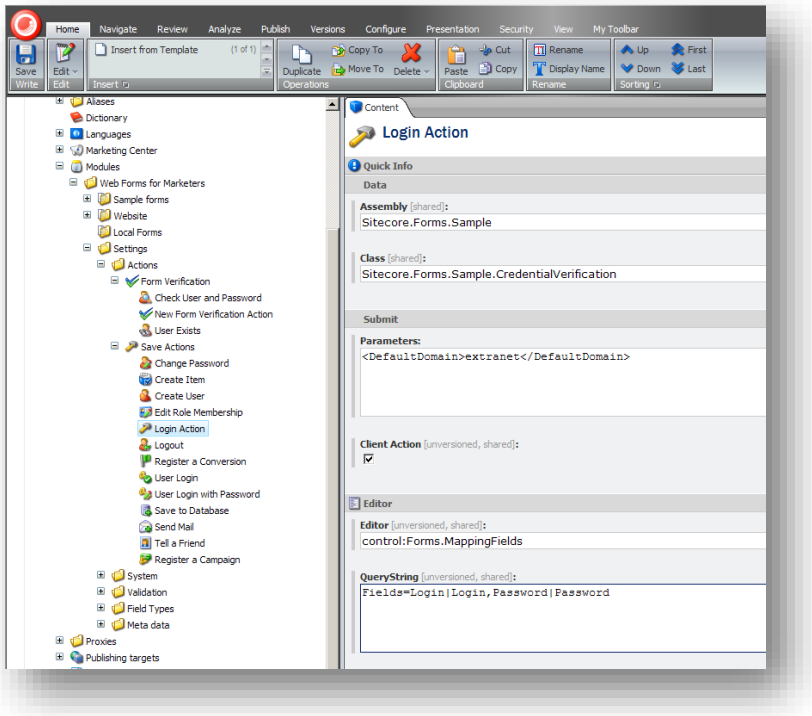
1. Create a new project, for example `Sitecore.Forms.Sample`.
2. Add a new reference to the `Sitecore.Forms.Core` assembly.
3. Create a new class that inherits the `Sitecore.Form.Submit.ISaveAction` interface.

   Action Code:

```
using Sitecore.Data;
using Sitecore.Diagnostics;
using Sitecore.Form.Core.Client.Data.Submit;
using Sitecore.Form.Core.Controls.Data;
using Sitecore.Form.Submit;
using Sitecore.Security.Authentication;

namespace Sitecore.Forms.Sample
{
   /// <summary>
   /// Login action
   /// </summary>
   public class LoginAction : ISaveAction
   {
      #region Methods

      /// <summary>
      /// Initializes a new instance of the <see cref="LoginAction"/> class.
      /// </summary>
      public LoginAction()
      {
         this.DefaultDomain = "sitecore";
      }

      /// <summary>
      /// Executes the login action.
      /// </summary>
      /// <param name="formid">The form id.</param>
      /// <param name="fields">The fields of the form.</param>
      /// <param name="data">The custom data.</param>
      public void Execute(ID formid, AdaptedResultList fields, params object[] data)
      {
         AdaptedControlResult login = fields.GetEntry(this.Login, "Login");
         AdaptedControlResult password = fields.GetEntry(this.Password, "Password");

         Assert.ArgumentNotNull(login, "You should point the login field.");
         Assert.ArgumentNotNull(password, "You should point the password field.");

         string userName = login.Value;
         Assert.ArgumentNotNullOrEmpty(userName, "Login can't be empty");

         if (!userName.Contains("\\"))
         {
            userName = string.Join("\\", new[] { this.DefaultDomain, userName });
         }

         AuthenticationManager.Login(userName, password.Value, true);
      }

      #endregion

      #region Properties

      /// <summary>
      /// Gets or sets the default domain.
      /// </summary>
```

```
            /// <value>The default domain.</value>
            public string DefaultDomain { get; set; }

            /// <summary>
            /// Gets or sets the login field.
            /// </summary>
            public string Login { get; set; }

            /// <summary>
            /// Gets or sets the password field.
            /// </summary>
            public string Password { get; set; }

            #endregion

        }
    }
```

4. In the **Content Editor**, create a new item under `/sitecore/System/Modules/Web Forms for Marketers/Settings/Actions/Save Actions` and specify the following parameters:



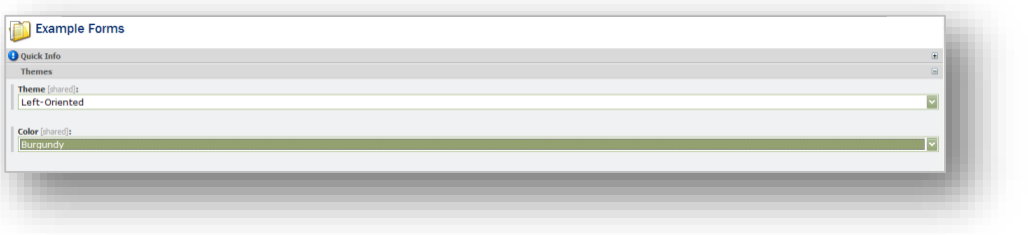
   - **Assembly** — the appropriate assembly.
   - **Class** — the created class.
   - **Parameters** — define the action's parameters — these are common to all forms.
   - **Client Action** — is only used in the staging environment. If this check box is clear, this save action is transferred from the Slave to the Master server and performed there. If this check box is selected, the save action is performed on the Slave server.
   - **Editor** — specify the control that is presented to the Sitecore user when they edit the parameters for this action in the **Form Designer**. In this example, these are Login and Password.
   - **QueryString** — additional settings for the editor.

3.4 How to Use CSS Themes

The module contains a few CSS themes that you can apply to forms. To change a theme, go to the forms folder. This item is based on the `/sitecore/Templates/Web Forms for Marketers/Forms Folder` template.

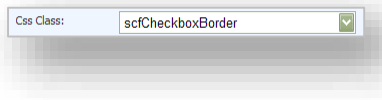
The *Forms Folder* item contains the following fields:

- **Theme** — sets the appearance theme that is used for all the forms that are stored under this item.
- **Color** — sets the color theme that is used for all the forms that are stored under this item.



If you want to extend the list of available themes, you must register the new theme. You must create a new item under the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Meta data/Themes` folder or the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Meta data/Colors` folder. The name of this item must coincide with the name of the file where you defined the CSS styles. This file must be in the `sitecore modules/web forms for marketers/themes or sitecore modules/web forms for marketers/themes/colors` folder.

You can also change the CSS class at field level. In the **Form Designer**, you can use `Css Class` property that is available for every field.



To extend or add a new CSS classes to the list:

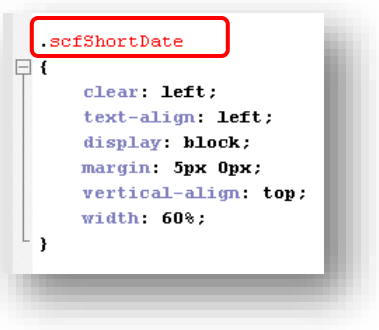
1. Add the definition of the CSS class to the `sitecore/System/modules/web forms for marketers/themes/Custom.css` class file.
2. Under the `/sitecore/System/Modules/Web Forms for Marketers/Settings/Meta data/Css Classes` folder, create a new item based on the `/sitecore/Templates/Web Forms for Marketers/Meta Data/Extended List Item` template.
3. In the **Value** field of the new item, add the name of your CSS class.

3.5 How to Configure CSS Styles

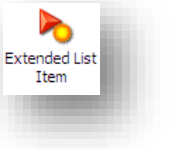
The module contains a few CSS styles that you can apply to form fields. Every field type has a default CCS style that is applied to it. You can customize CSS styles.

To add a new CSS style:

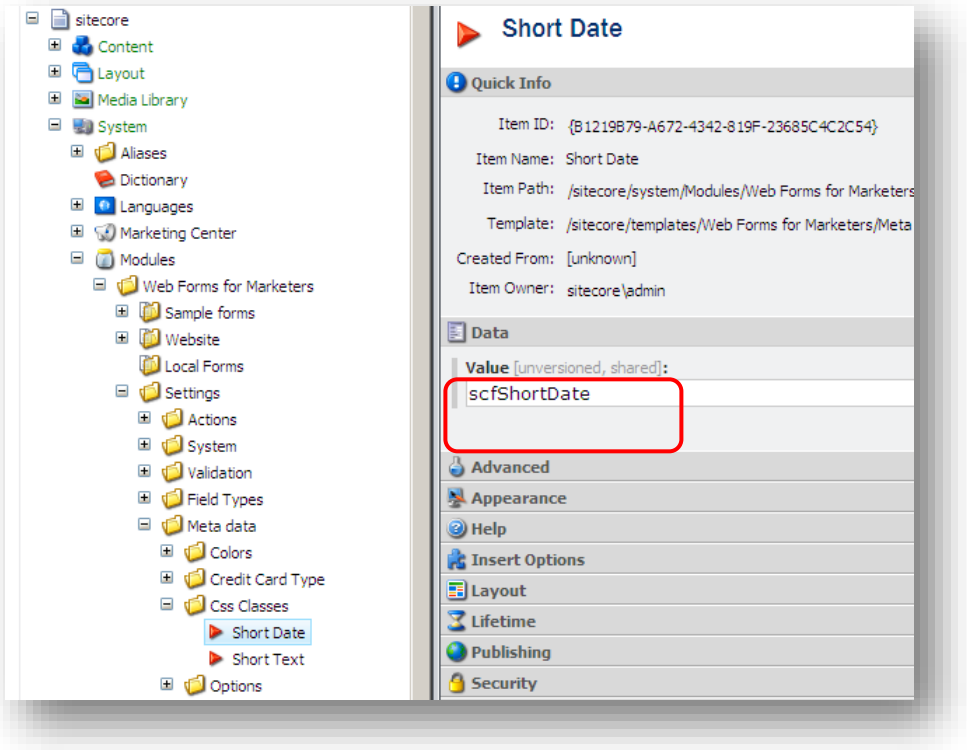
1. In the `custom.css` file, located under the `Website\sitecore modules\Shell\Web Forms for Marketers\Themes\` folder, define a new CSS style.



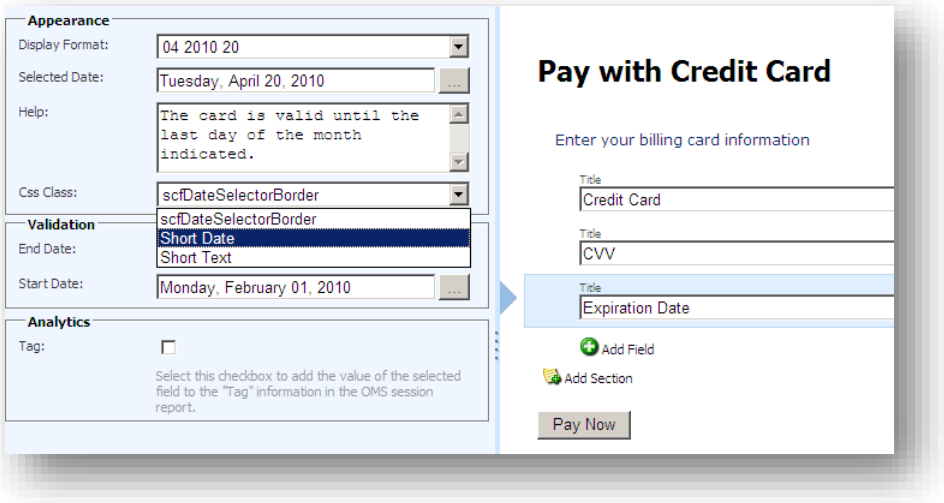
2. In the **Content Editor**, navigate to the `Sitecore/System/Modules/Web Forms for Marketers/Settings/Meta data/CSS Classes` folder.
3. Create a new **Extended List item**.



4. In the new item, in the value field, enter the name of the CSS style.

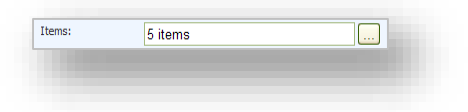


Your custom CSS style is added to the system. You can apply it to any form field in the **Form Designer**:

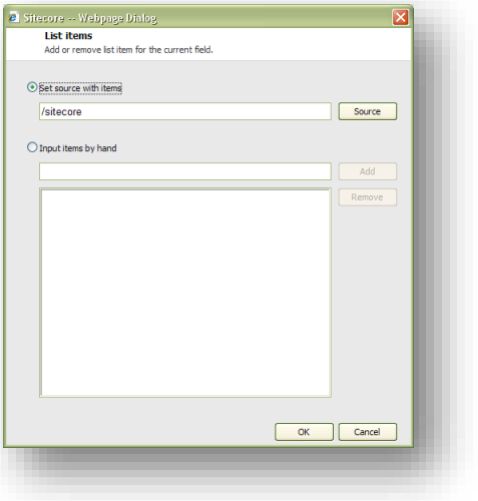


3.6 How to Load Items from the Content Tree to a List Control

Any list control used by the Web Forms for Marketers module can read its items from the content tree. To do this, set the source from which the control will take its items. Use the *Items* property in the **Form Designer** for this:



Click the browse button …to open the **List Items** dialog box that you use to edit the list items:



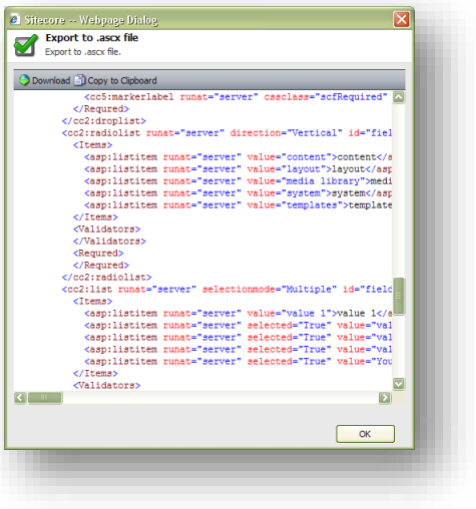
The control can read either *source* or *predefined list* items.

To select the source, click **Source**.

To select a **predefined list**, select the **Input items by hand** option**,** and enter the name of the list item and then click **Add**.

3.7 How to Export to ASCX

In the **Form Designer**, click **Export to ascx** to convert the form to an ascx control. The code is displayed in the **Export to ascx file** dialog box.



Click **Download** to store the code as an `.ascx` file.

3.8 How to Add an ASCX Control to the Page

1. In the layouts folder, create a new `default.aspx` page.

```
        <%@ Page Language="C#" AutoEventWireup="true" %>
        <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

        <html xmlns="http://www.w3.org/1999/xhtml" >
        <head runat="server">
            <title>Untitled Page</title>
        </head>
        <body>
            <form id="form1" runat="server">
            </form>
        </body>
        </html>
```

2. Click **Export to ascx** to export the form you need to the layouts\ folder. Use the `forms.ascx` format for the name of the file.
3. Add the following changes to the `default.aspx` page:

```
        <%@ Page Language="C#" AutoEventWireup="true"  %>
        <%@ Register Src="form.ascx" TagName="SimpleForm" TagPrefix="uc1" %>
        <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

        <html xmlns="http://www.w3.org/1999/xhtml" >
        <head runat="server">
            <title>Untitled Page</title>
        </head>
        <body>
            <form id="form1" runat="server">
            <div>
                <uc1:SimpleForm ID="WebUserControl1" runat="server" />
            </div>
            </form>
        </body>
        </html>
```

In the browser, type http://localhost/layouts/default.aspx.

After you make any changes to the `.ascx` file, the module does not verify that the form works correctly.

3.9 How to Configure a Data Provider

The module supports MSSQL and SQLite databases.

SQLite configuration

If your solution uses SQLite databases, no additional configurations are needed.

MSSQL configuration

If you use MSSQL databases, you must configure a connection between the module and the databases to make the form reports work correctly.

The MSSQL databases are located under the `[site root]\data` folder:

- `sitecore.webforms.mdf` – MSSQL database;
- `sitecore.webforms.bak` – backup file of the empty MSSQL database;
- `sitecore.webforms_log.ldf` – MSSQL log file.

In the `forms.config` file, select a data provider. Use the `type` attribute in the `formsDataProvider` tag.

Among the possible values are:

- `Sitecore.Forms.Data.DataProviders.WFMDataProvider,Sitecore.Forms.Core` — the provider for the MSSQL databases.

```
            <formsDataProvider
type="Sitecore.Forms.Data.DataProviders.WFMDataProvider,Sitecore.Forms.Core">
              <param desc="connection string">Database=(database);Data Source=(server);user
id=(user);password=(password);Connect Timeout=30</param>
            </formsDataProvider>
```

- `Sitecore.Forms.Data.DataProviders.SQLite.SQLiteWFMDataProvider,Sitecore.Forms.Core` — the provider for the SQLite database.

```
            <formsDataProvider
type="Sitecore.Forms.Data.DataProviders.SQLite.SQLiteWFMDataProvider,Sitecore.Forms.Core">
              <param desc="connection string">Data
Source=/data/sitecore_webforms.db;version=3;BinaryGUID=true</param>
            </formsDataProvider>
```

3.10 How to Send SMS/MMS Using a Custom Processor

The *Send MMS* and the *Send SMS* save actions delegate sending messages to MMS/SMS gateways through a SMPT server.

You can use the `processMessage` pipeline to change this behavior and send messages, for example, through a third party paid Web service.

Your custom processor can look as follows:

```
namespace Sitecore.Form.Core.Pipelines.ProcessMessage
{
  using System.IO;
  using System.Net;

  public class SendSMSorMMS
  {
    public void Process(ProcessMessageArgs args)
    {
      if (args.MessageType == MessageType.MMS || args.MessageType == MessageType.SMS)
      {
        WebClient wc = new WebClient();
        wc.Credentials = (NetworkCredential)args.Credentials;

        wc.QueryString.Add("sendto", args.Recipient);
        wc.QueryString.Add("message", args.Mail.ToString());
        if (!string.IsNullOrEmpty(args.From))
        {
          wc.QueryString.Add("from", args.From);
        }
        using (Stream responseStream = wc.OpenRead("https://3rdparty.smsormms.com/"))
        {
          using (StreamReader responseReader = new StreamReader(responseStream))
          {
            responseReader.ReadToEnd();
            responseReader.Close();
            responseStream.Close();
          }
        }
      }
    }
  }
}
```

The new processor must be registered in the `forms.config` file:

```
            <processMessage>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="ExpandLinks"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="ExpandTokens"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="AddHostToItemLink"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="AddHostToMediaItem"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="AddAttachments"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="BuildToFromRecipient"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.SendSMSorMMS,
MyAssembly"/>
              <processor type="Sitecore.Form.Core.Pipelines.ProcessMessage.ProcessMessage,
Sitecore.Forms.Core" method="SendEmail"/>
            </processMessage>
```

3.11 How to Implement “Required” Checkbox Field

In the Web Forms for Marketers module, the Checkbox field does not support the “required” validation rule. To make the Checkbox field “required”, follow one of the methods below.

Use the CheckboxList field

1. Add the Checkbox List field to a form.
2. Add only one item to the list.
3. Mark this Checkbox List as required.

Create a custom validator for the CheckboxList field

1. Create a class that is inherited from the System.Web.UI.WebControls.BaseValidator class. See a sample code:

```
        class CheckboxValidation : BaseValidator
            {
                protected CheckBox ctrToValidate;
                protected CheckBox CheckBoxToValidate
                {
                    get
                    {
                        if (ctrToValidate == null)
                        {
                            ctrToValidate = base.FindControl(ControlToValidate) as CheckBox;
                        }
                        return ctrToValidate;
                    }
                }
                protected override bool ControlPropertiesValid()
                {
                    if (base.ControlToValidate == null || base.ControlToValidate.Length == 0)
                    {
                        throw new HttpException(string.Format("The ControlToValidate property of
'{0}' cannot be blank.", this.ID));
                    }
                    if (this.CheckBoxToValidate == null)
                    {
                        throw new HttpException(string.Format("The CheckBoxValidator can only
validate controls of type CheckBox."));
                    }
                    return true;
                }
                protected override bool EvaluateIsValid()
                {
        this.ErrorMessage = string.Format(this.ErrorMessage, "{0}", CheckBoxToValidate.Text);
                    //Validate whether checkbox is checked
                    return this.CheckBoxToValidate.Checked == true;
                }
            }
```

2. Create an item under the *sitecore/system/modules/web forms for marketers/settings/validation* folder. The item must be based on the BaseValidator template.
3. In the **Assembly** and **Class** fields, enter the appropriate values of the custom assembly.
4. In the **Error Message** field, enter this string: “*The {0} checkbox must be checked*”
5. In the **Text** field, enter appropriate message. If this field is blank its value is the same as the **Error Message** one.

6. Duplicate the */sitecore/system/modules/web forms for marketers/settings/field types/simple types/checkbox* item and rename it to **CheckboxRequired,** for instance.
   Note: In the **CheckboxRequired** item, do not select the “Required” checkbox.
7. In *the /sitecore/system/modules/web forms for marketers/settings/field types/simple types/CheckboxRequired* item, in the **Validation** field, select your custom validator.