CMSImport PRO
User manual

Version 1.1

1 Introduction

CMSImport PRO helps you import content or members from any datasource into Umbraco. The following datasources are supported by default:

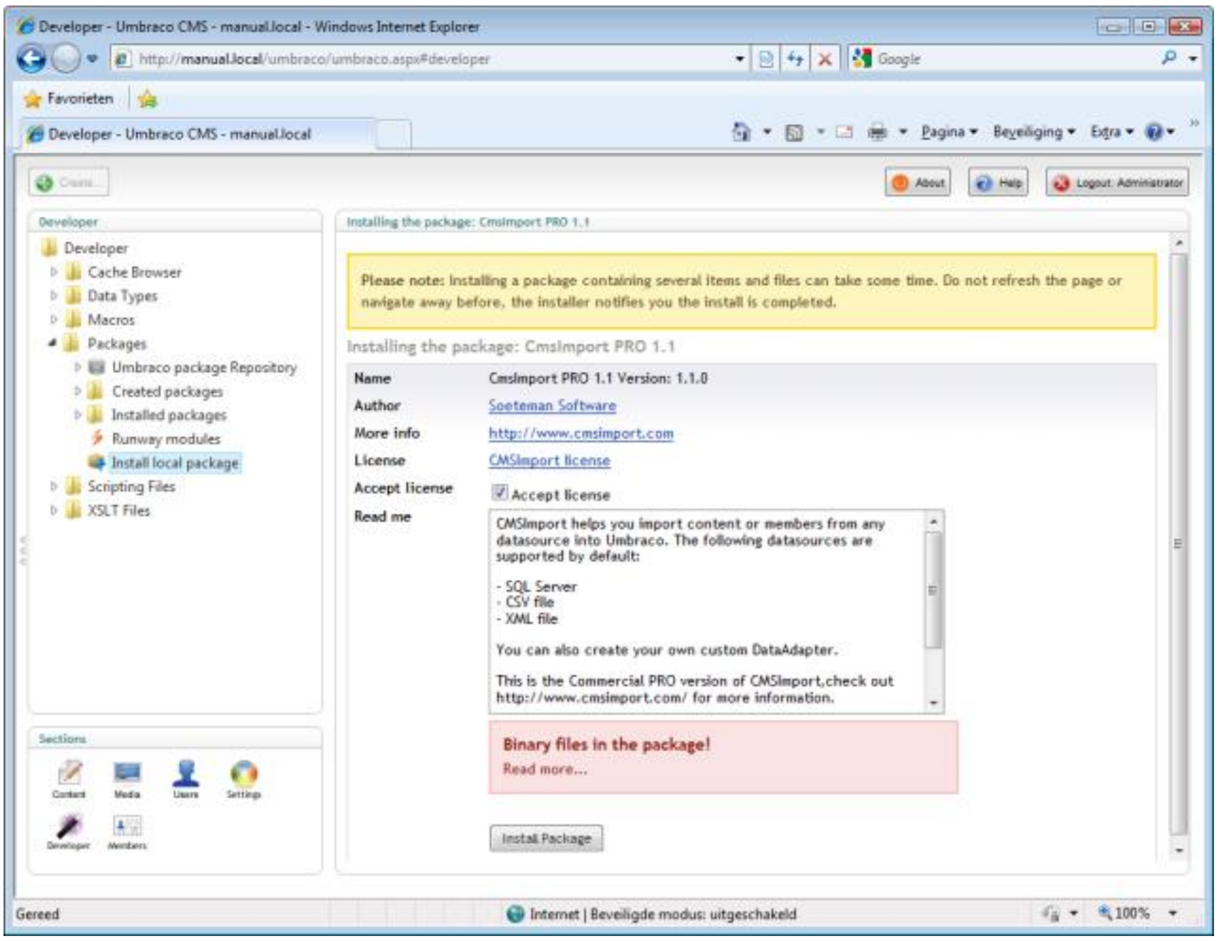
•SQL Server
•CSV file
•XML file

With CMSImport PRO it's possible to import media related to content or members also.
When media is imported references in content or member data will be updated automatically.

With CMSImport PRO it is possible to save the import steps for later use and schedule saved imports for a certain date and time.

2 Installation

Install the CMSImport PRO package using the Umbraco package installer.



Please note: Installing a package containing several items and files can take some time. Do not refresh the page or navigate away before, the installer notifies you the install is completed.

Installing the package: CmsImport PRO 1.1

Name	CmsImport PRO 1.1 Version: 1.1.0
Author	Soeteman Software
More info	http://www.cmsimport.com
License	CMSImport license
Accept license	☑ Accept license
Read me	CMSImport helps you import content or members from any datasource into Umbraco. The following datasources are supported by default: - SQL Server - CSV file - XML file You can also create your own custom DataAdapter. This is the Commercial PRO version of CMSImport,check out http://www.cmsimport.com/ for more information.

Binary files in the package!
Read more...

Make sure that the installer has modify rights on the following folders:

- /bin
- /config
- /umbraco

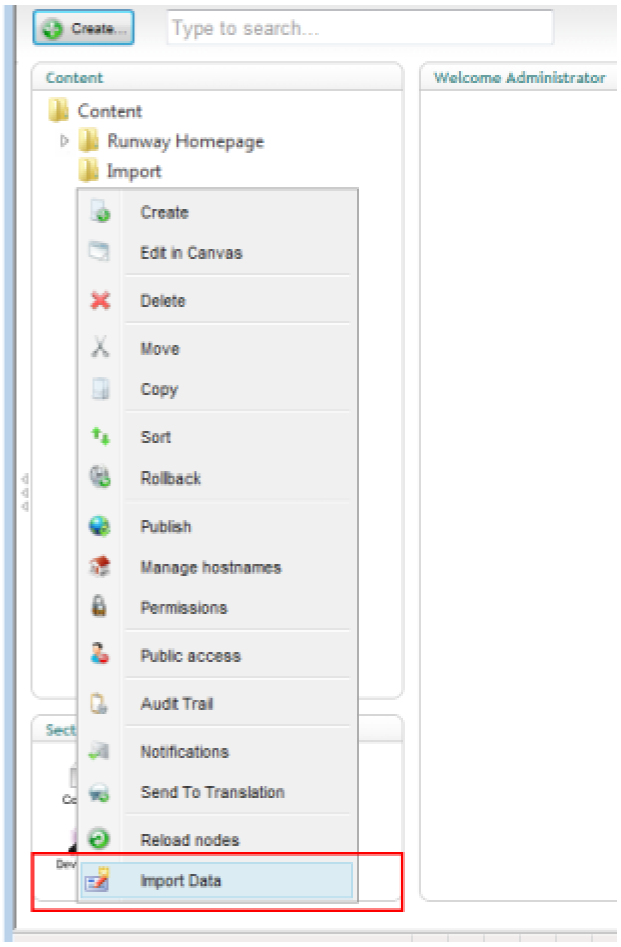
The installer also needs rights to create tables in the database.

If for some reason you can't give the installer these rights, you can install the package manually. Read the chapter manual configuration how to do this.

Once the package is installed you have an extra folder in your developer section called CMSImport. You might need a page refresh, or even logout and in again to see this folder.

3 Import Content

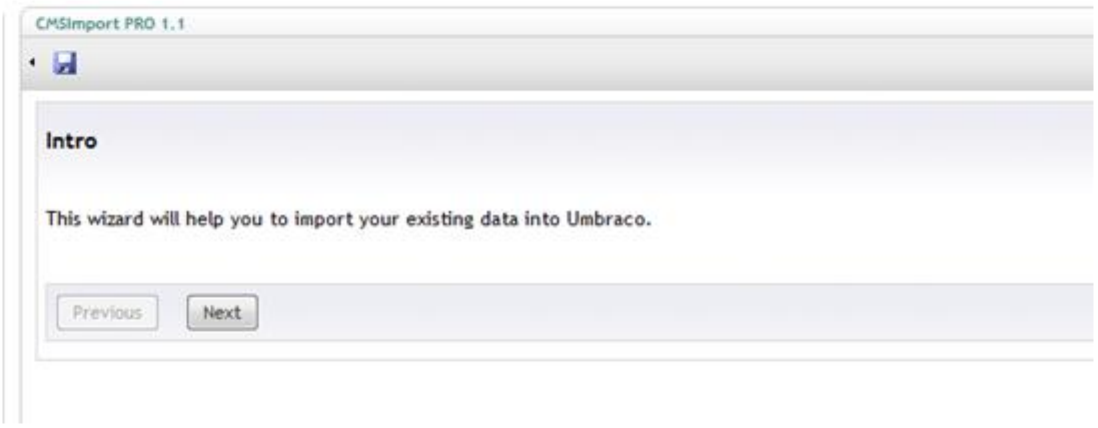
You can start CMSImport by right clicking on the content node and select the import data option (only visible for administrators).



Or you can go to the developer section, open CMSImport and select Start Import. The following wizard will open.

3.1 Introduction

When the wizard starts it will start with an introduction.

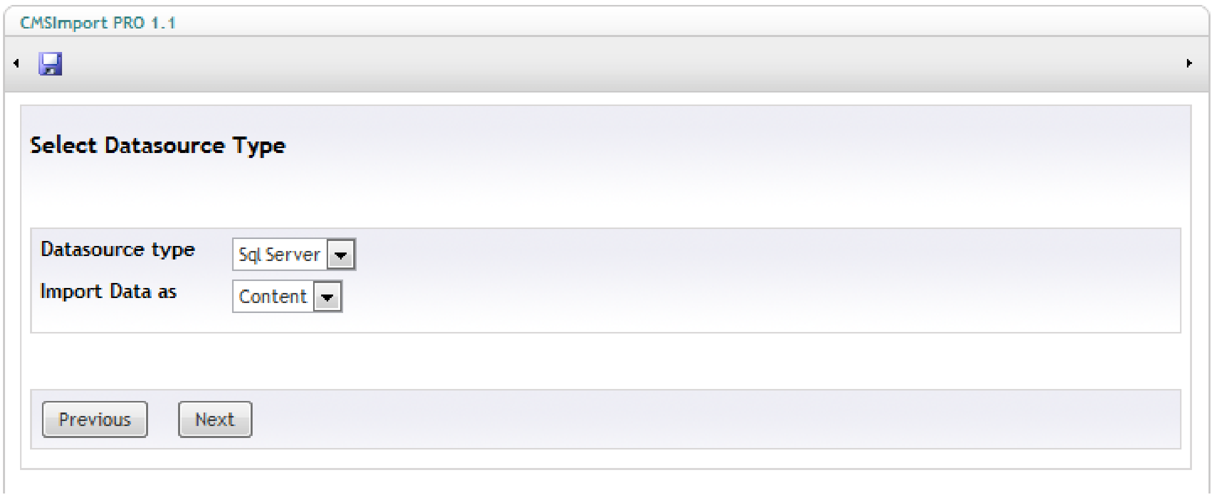


Intro

This wizard will help you to import your existing data into Umbraco.

3.2 Select Datasource Type

In this step you are asked to specify a datasource type and how the data should be imported (content/members).



Datasource type	Sql Server
Import Data as	Content

The following datasource types are supported by default:

- Sql Server
- CSV File
- XML File.

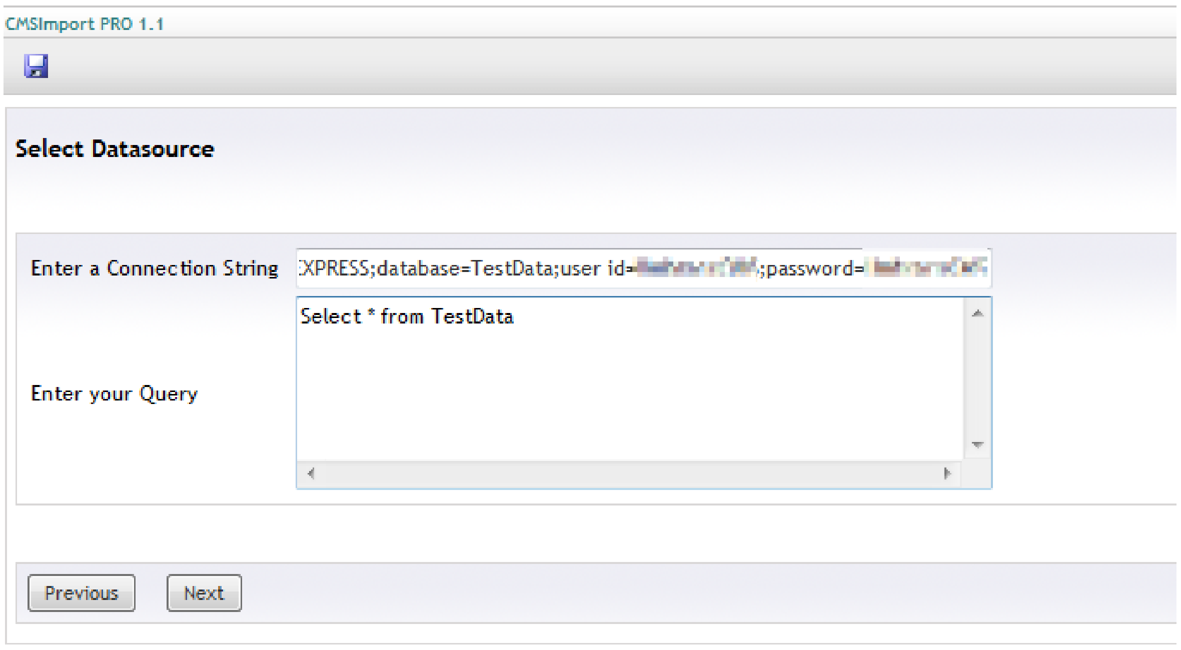
It is possible to create extra datasource types. See chapter extend CMSImport for more information.

3.3 Select datasource

In the select datasource step you need to provide the information for the selected datasource Type.

3.3.1 SQL Server

In case of SQL server you need to provide the Connection string and the query that you want to use to retrieve the data.


CMSImport PRO 1.1

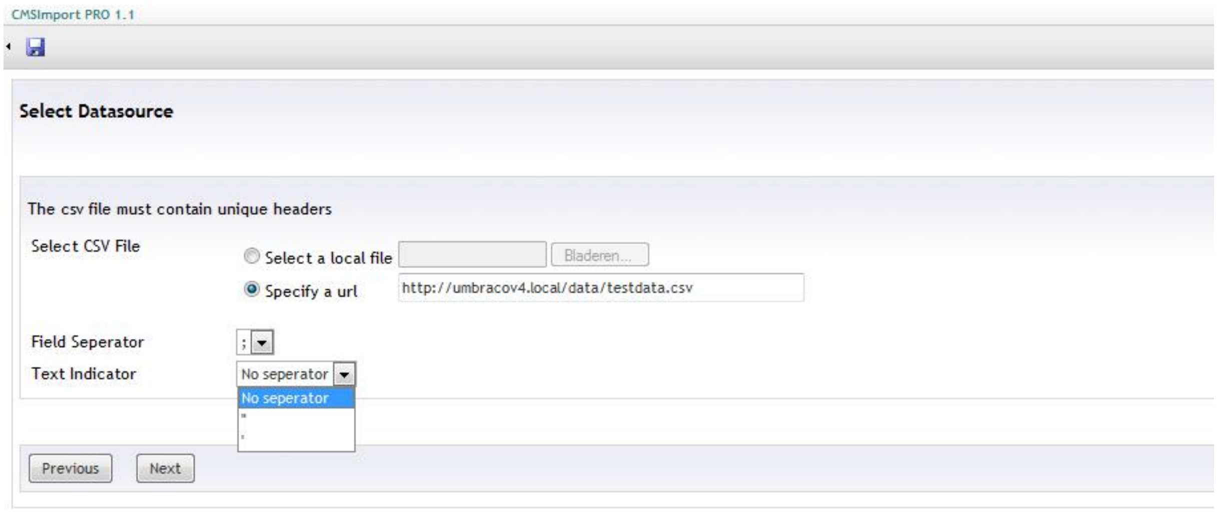
Select Datasource

Enter a Connection String: :XPRESS;database=TestData;user id=[redacted];password=[redacted]

Enter your Query: Select * from TestData

3.3.2 CSV File

In case of a CSV file you need to select a CSV File. You also need to provide the field separator and text indicator. A field separator is the character that is used to separate the columns. The text indicator is a character that indicates a text string. The default settings are based on an export from Microsoft Excel.


CMSImport PRO 1.1

Select Datasource

The csv file must contain unique headers

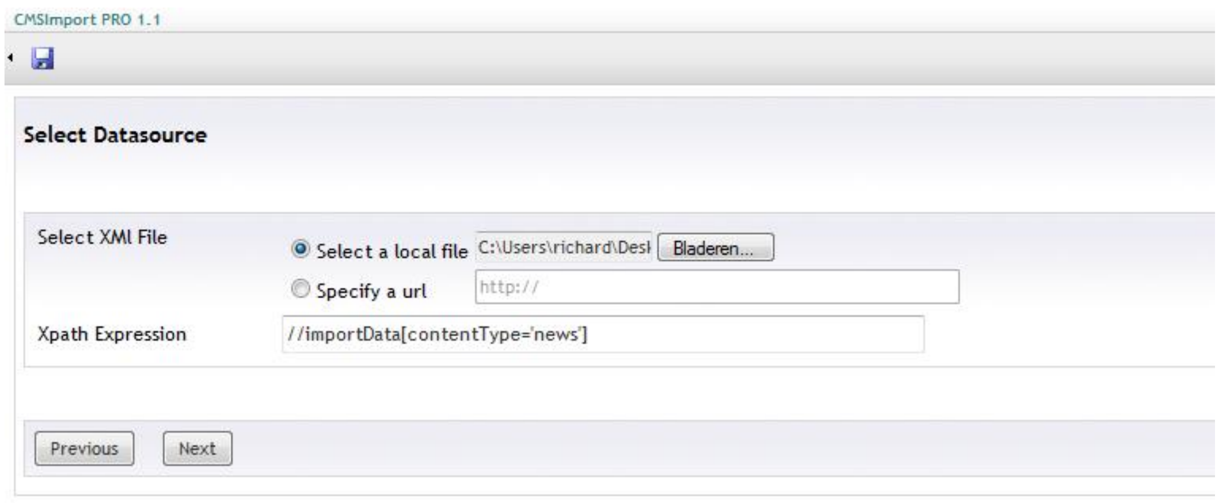
Select CSV File: ○ Select a local file [● Specify a url] http://umbracov4.local/data/testdata.csv

Field Seperator: ;

Text Indicator: No seperator

3.3.3 XML File

In case of an XML file you need to select the xml file. Optional you can specify an xpath expression.



Select XMl File

[✓ Select a local file] C:\Users\richard\Desl Bladeren...

Specify a url http://

Xpath Expression //importData[contentType='news']

When using xml you can only use elements, attributes are not supported at the moment. All elements that you want to map later must be under the first child node of the xml document.

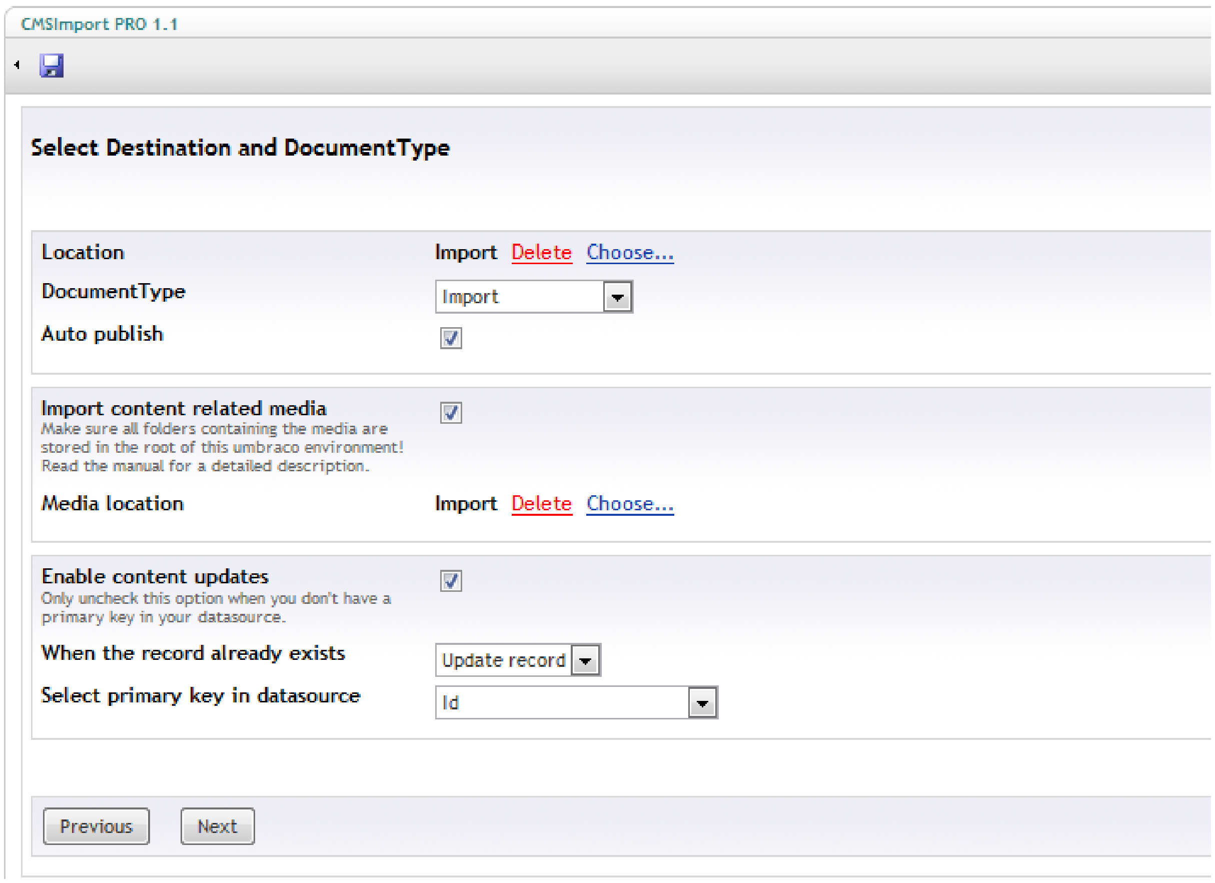
3.4 Select destination and document type

In this step you need to specify the location where you want to store the imported documents. This is prefilled if you started the import from the context menu in content. You can also specify the document type and you select the auto publish checkbox, when checked items are published automatically.

Import content related media will import any reference that CMSImport can find as a media item and update the source data with a reference to the media item. See chapter “Related media import” for more information.

The “When the item already exists” and the primary key option are needed for content updates. With the “When the item already exists” option you specify what to do when an item is already imported. Possible options are skip and overwrite. With the primary key you specify the key in the datasource. This field will be used to determine if an item is already imported.

To disable content updates uncheck the “Enable content updates” option. **Only do this when you don’t have a primary key in your datasource**. No relation between imported data and Umbraco document is stored so even when you run the import for the second time data will be imported as new records.


CMSImport PRO 1.1

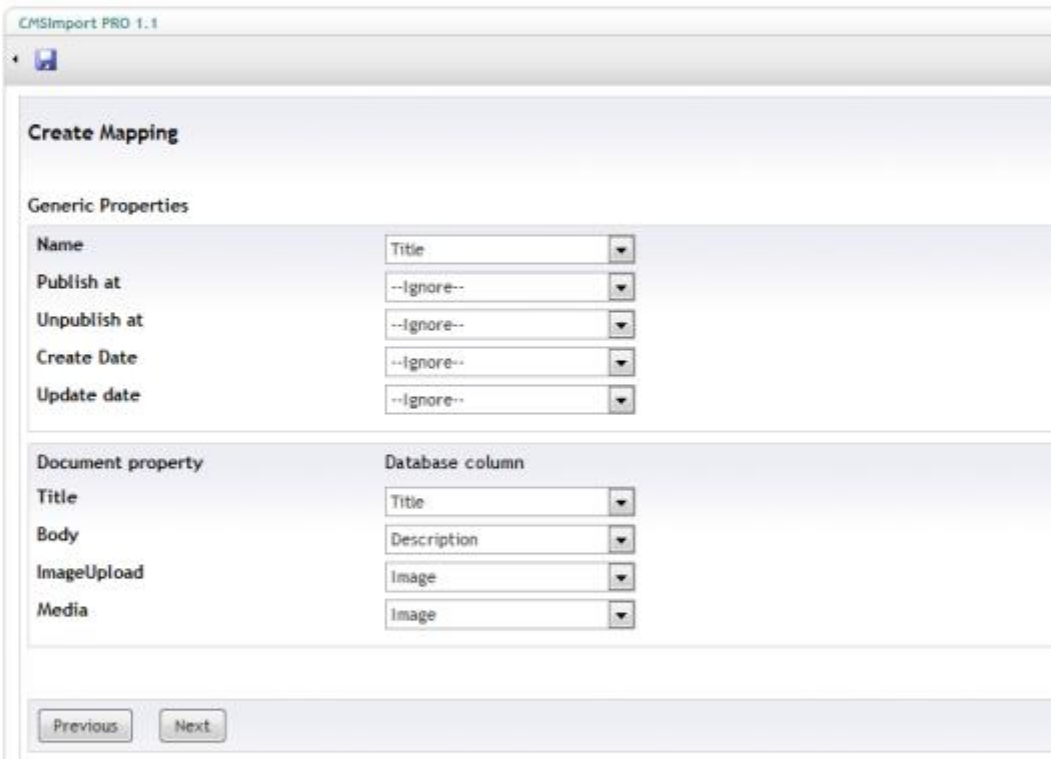
Select Destination and DocumentType

Location	Import Delete Choose...
DocumentType	Import
Auto publish	✓
Import content related media Make sure all folders containing the media are stored in the root of this umbraco environment! Read the manual for a detailed description.	✓
Media location	Import Delete Choose...
Enable content updates Only uncheck this option when you don't have a primary key in your datasource.	✓
When the record already exists	Update record
Select primary key in datasource	Id

Previous Next

3.5 Create mapping

In this step you can specify the mapping between the fields from the data source and the properties of the Umbraco document type.


CMSImport PRO 1.1

Create Mapping

Generic Properties

Name	Title
Publish at	--Ignore--
Unpublish at	--Ignore--
Create Date	--Ignore--
Update date	--Ignore--

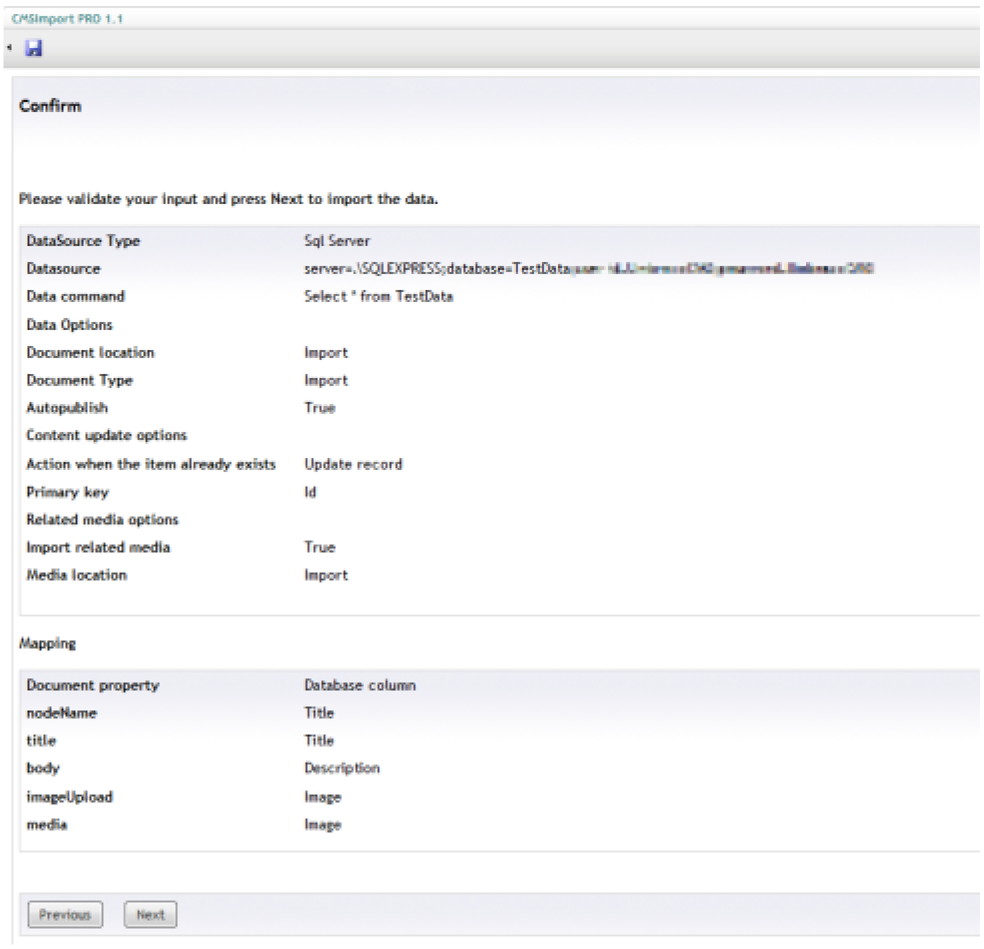
Document property	Database column
Title	Title
Body	Description
ImageUpload	Image
Media	Image

Previous Next

Quick tip:
When your fieldnames from the datasource are the same as the alias of the document property CMSImport will automatically map this field.

3.6 Confirm

In this step you can validate the selected options one more time. When you click next the import will start.



Confirm

Please validate your input and press Next to import the data.

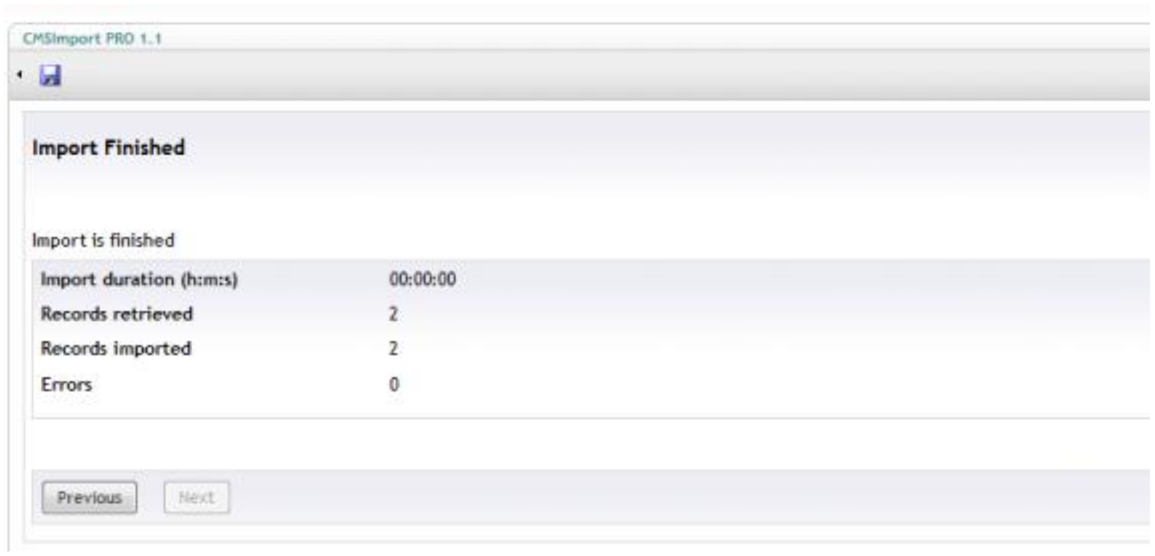
DataSource Type	Sql Server
Datasource	server=.\SQLEXPRESS;database=TestData;[redacted]
Data command	Select * from TestData
Data Options	
Document location	Import
Document Type	Import
Autopublish	True
Content update options	
Action when the item already exists	Update record
Primary key	Id
Related media options	
Import related media	True
Media location	Import

Mapping

Document property	Database column
nodeName	Title
title	Title
body	Description
imageUpload	Image
media	Image

3.7 Import

When you click next in this step the import starts. When the import is finished it will report what it did. If there were any errors it will also report the errors.


CMSImport PRO 1.1

Import Finished

Import is finished

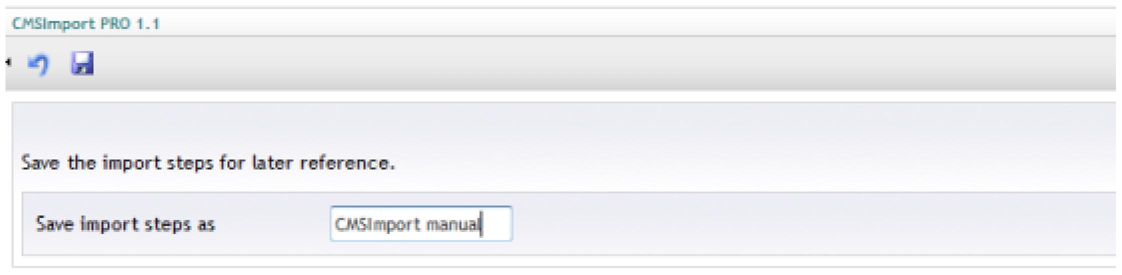
Import duration (h:m:s)	00:00:00
Records retrieved	2
Records imported	2
Errors	0

Quick tip:
In case of test imports it might be useful to install the Content Maintenance Dashboard package also to bulk publish/unpublish and delete content nodes.

You can download the package via:
http://our.umbraco.org/projects/developer-tools/content-maintenance-dashboard-package

3.8 Save Import steps

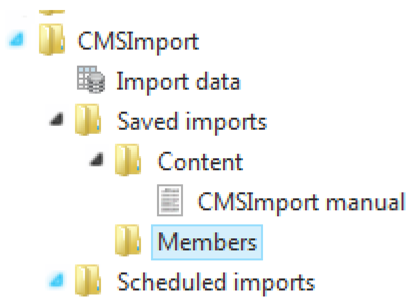
When you click save, you can specify a name and when hitting the save button again the import steps are saved for later use.


CMSImport PRO 1.1

Save the import steps for later reference.

Save import steps as: CMSImport manual

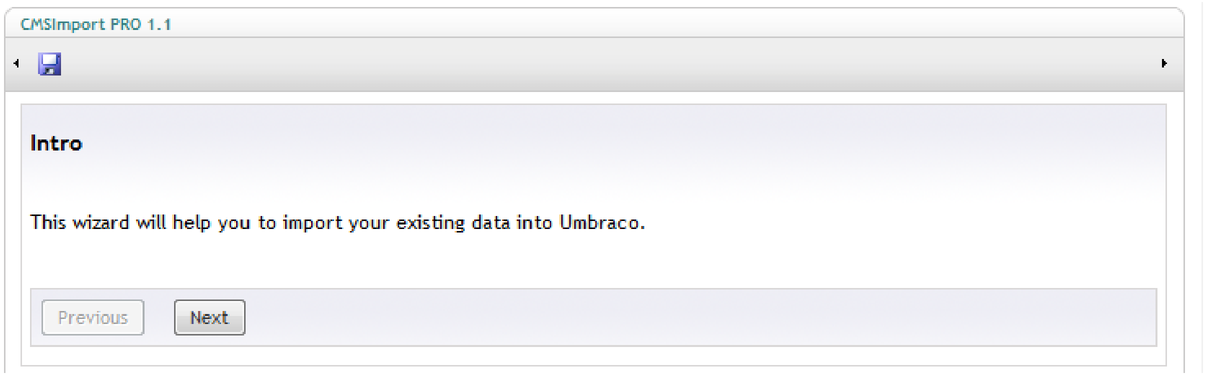
Saved imports can be found in the tree



4 Import members

4.1 Introduction

When the wizard starts it will start with an introduction



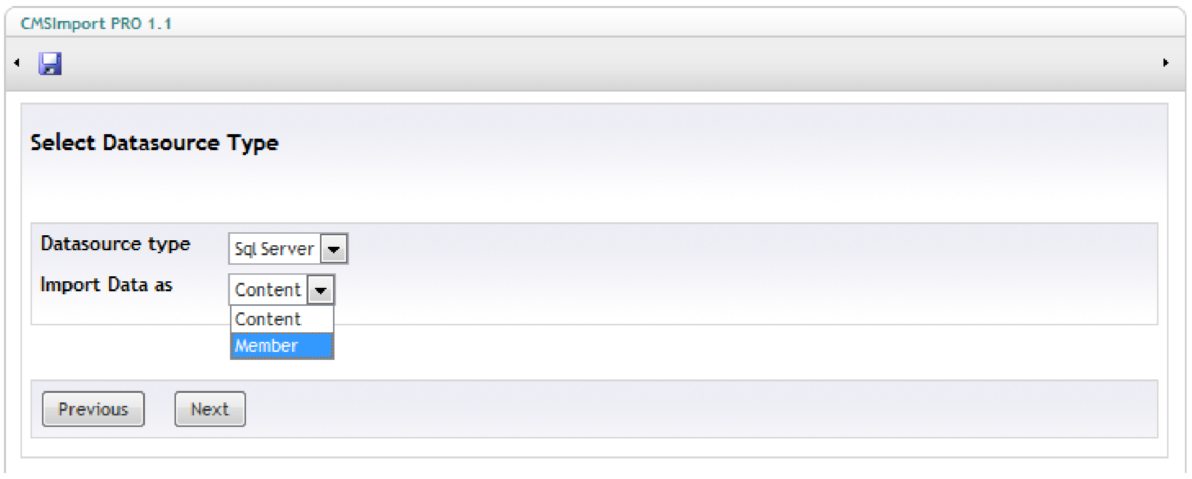
CMSImport PRO 1.1

Intro

This wizard will help you to import your existing data into Umbraco.

4.2 Select Datasource Type

In this step you are asked to specify a datasource type and how the data should be imported (content/members).



CMSImport PRO 1.1

Select Datasource Type

Datasource type: Sql Server

Import Data as: Content

The following datasource types are supported by default:

- Sql Server
- CSV File
- XML File.

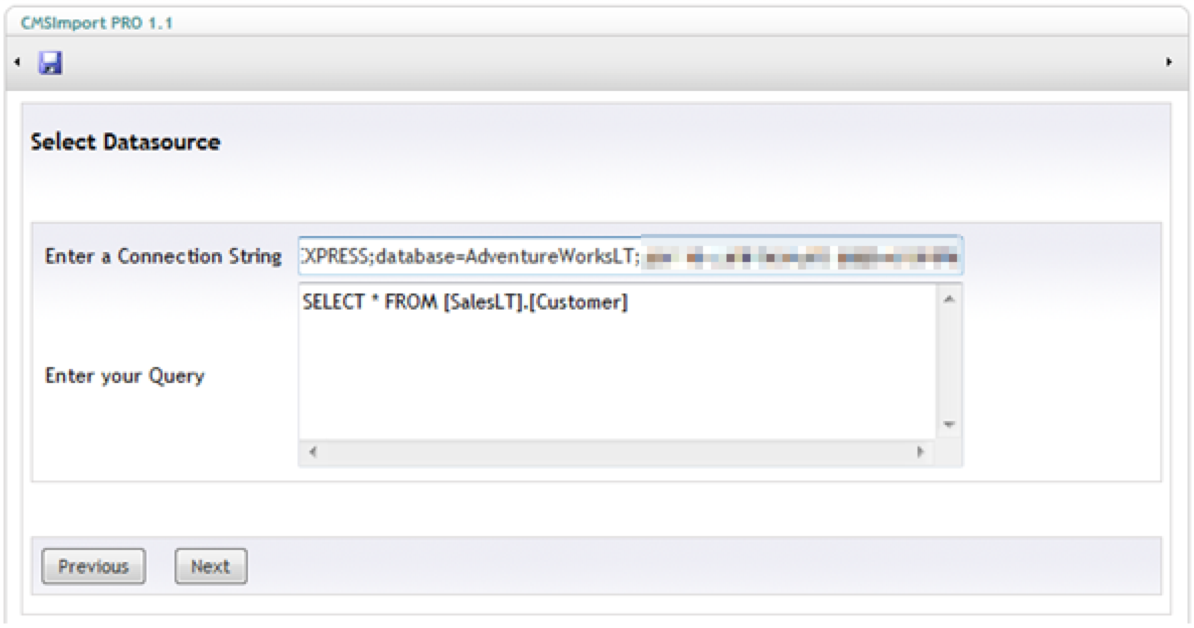
It is possible to create extra Datasource Types. See chapter extend CMSImport for more information.

4.3 Select Datasource

In the select Datasource step you need to provide the information for the selected datasource type.

4.3.1 SQL Server

In case of SQL server you need to provide the Connection string and the query that you want to use to retrieve the data.



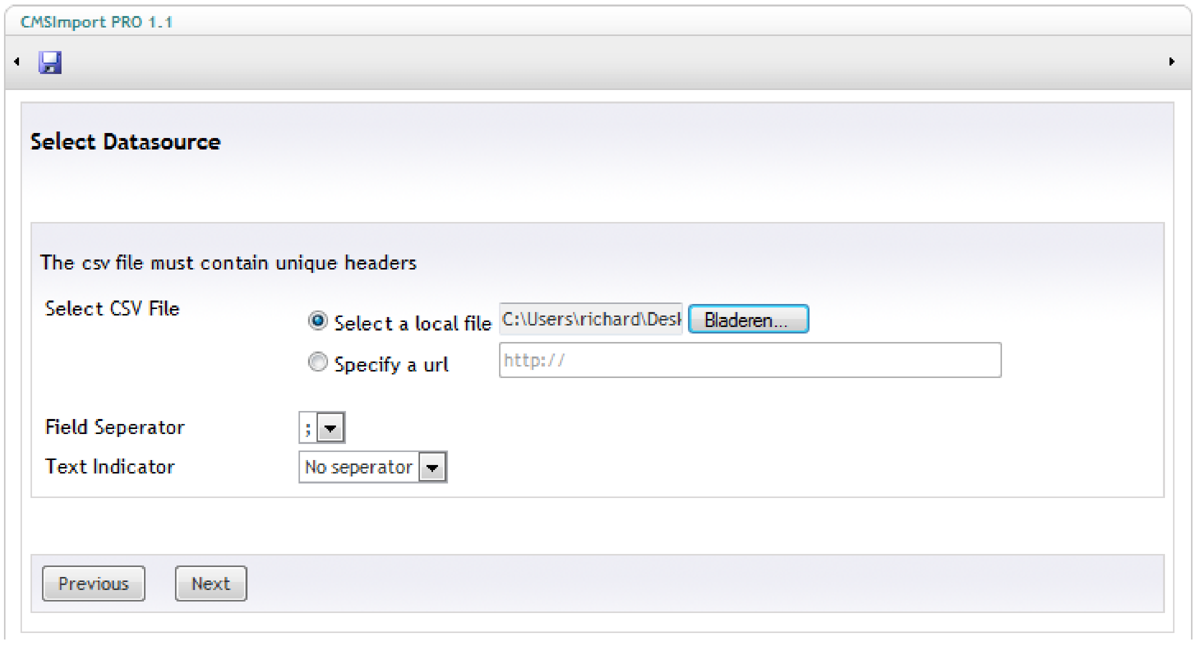
Select Datasource

Enter a Connection String: :XPRESS;database=AdventureWorksLT; [redacted]

Enter your Query: SELECT * FROM [SalesLT].[Customer]

4.3.2 CSV File

In case of a CSV file you need to select a CSV file. You also need to provide the field separator and Text Indicator. A field separator is the character that is used to separate the columns. The text indicator is a character that indicates a text string. The default settings are based on an export from Microsoft Excel.



Select Datasource

The csv file must contain unique headers

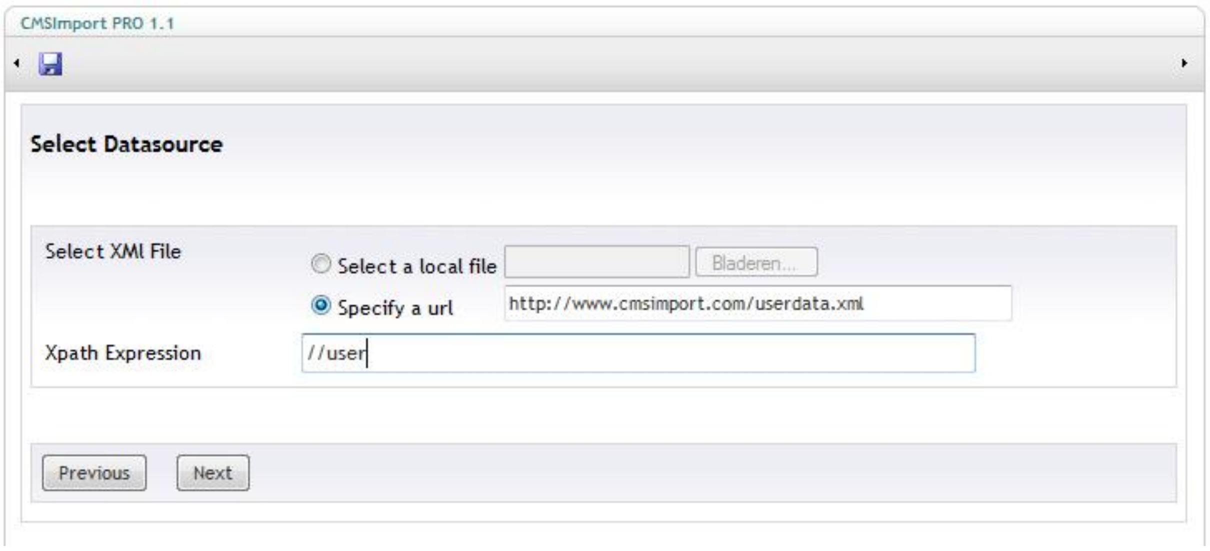
Select CSV File: [✓ Select a local file] C:\Users\richard\Desl Specify a url http://

Field Seperator: ;

Text Indicator: No seperator

4.3.3 XML File

In case of an XML file you need to select the xml File. Optional you can specify an xpath expression.


CMSImport PRO 1.1

Select Datasource

Select XMl File	Select a local file [ ] Bladeren...
	[✓ Specify a url] http://www.cmsimport.com/userdata.xml
Xpath Expression	//user

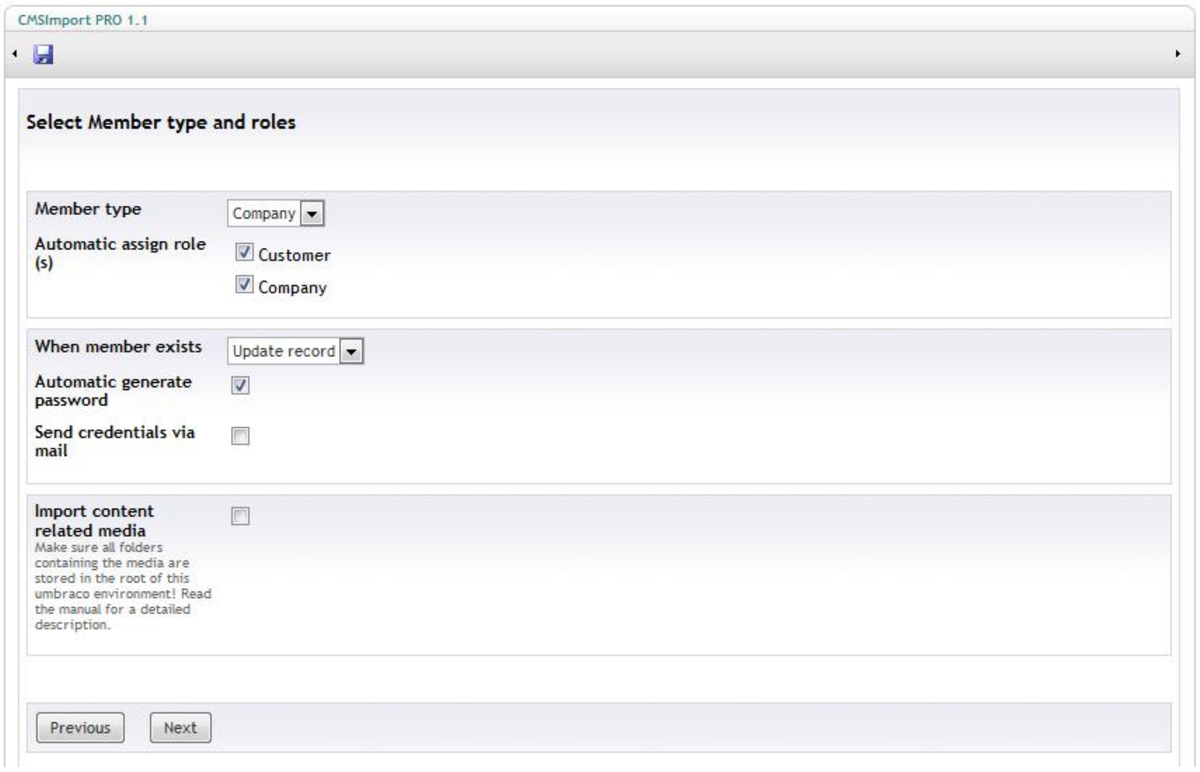
Previous Next

When using xml you can only use elements, attributes are not supported at the moment. All elements that you want to map later must be under the first child node of the xml document.

4.4 Select member type and role

In this step you can select the member type and assign one or more roles. With the "When the item already exists" option you specify what to do when an item is already imported. Possible options are skip and overwrite.

When the "Automatic generate password" option is checked a password is automatically generated for the imported member. When the "Send credentials via mail" is checked an email with login credentials is send to the imported member. You can edit the email template , check chapter settings on how to do this.


CMSImport PRO 1.1

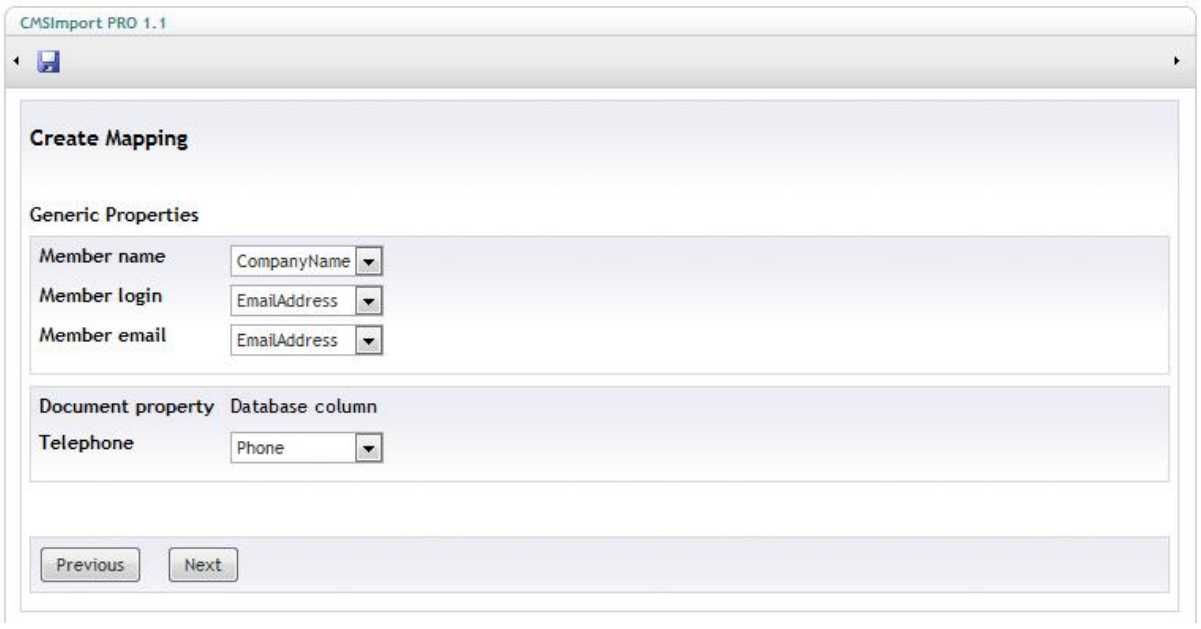
Select Member type and roles

Member type	Company
Automatic assign role (s)	✓ Customer ✓ Company
When member exists	Update record
Automatic generate password	✓
Send credentials via mail	☐
Import content related media Make sure all folders containing the media are stored in the root of this umbraco environment! Read the manual for a detailed description.	☐

Previous Next

4.5 Create Mapping

In this step you can specify the mapping between the fields from the data source and the properties of the Umbraco member type.



Create Mapping

Generic Properties

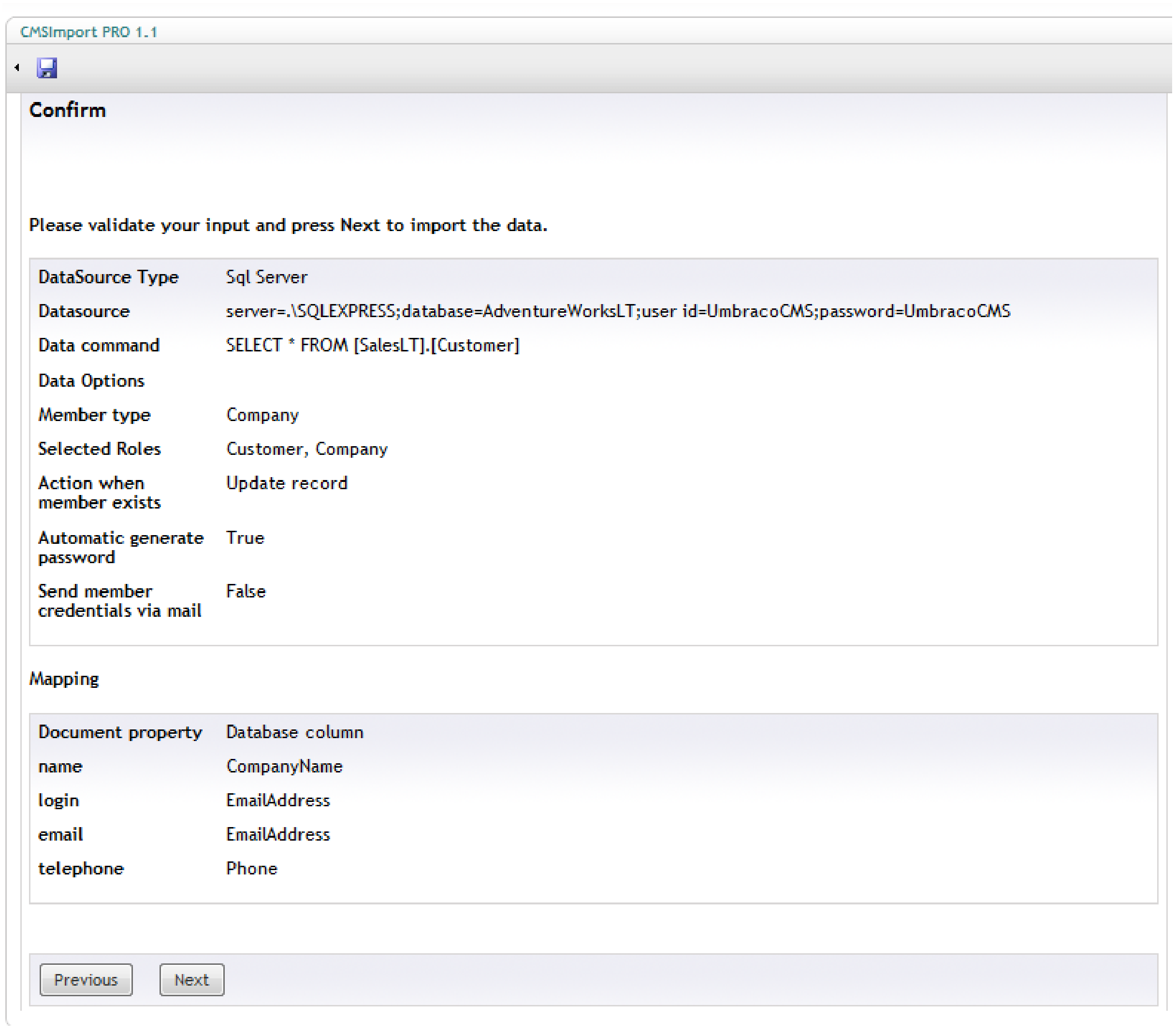
Member name	CompanyName
Member login	EmailAddress
Member email	EmailAddress

Document property	Database column
Telephone	Phone

Quick tip:
When your fieldnames from the datasource are the same as the alias of the document property CMSImport will automatically map this field.

4.6 Confirm

In this step you can validate the selected options one more time. When you click next the import will start.



Confirm

Please validate your input and press Next to import the data.

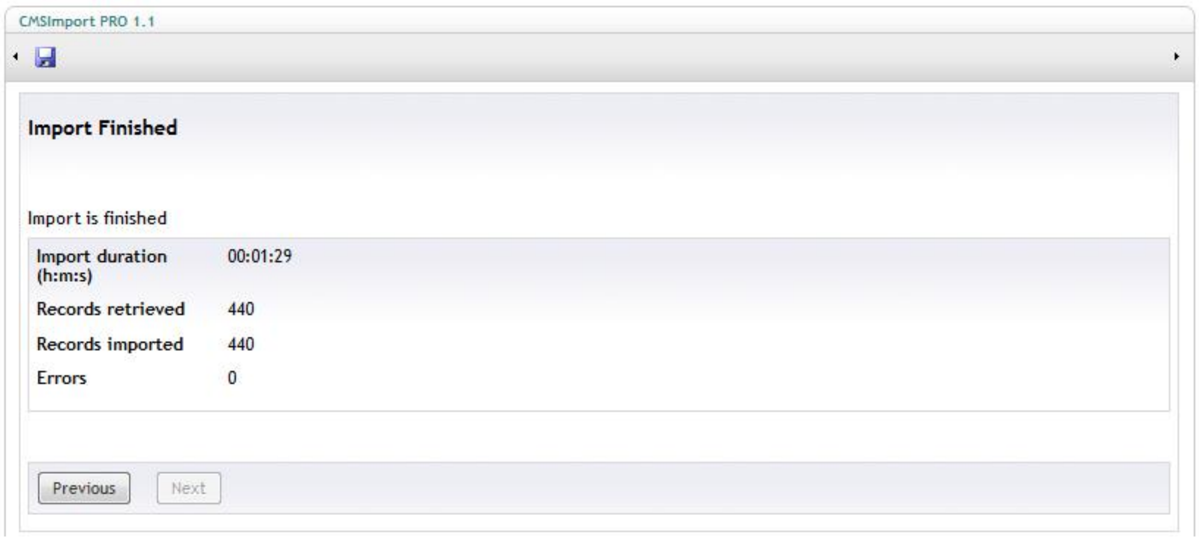
DataSource Type	Sql Server
Datasource	server=.\SQLEXPRESS;database=AdventureWorksLT;user id=UmbracoCMS;password=UmbracoCMS
Data command	SELECT * FROM [SalesLT].[Customer]
Data Options	
Member type	Company
Selected Roles	Customer, Company
Action when member exists	Update record
Automatic generate password	True
Send member credentials via mail	False

Mapping

Document property	Database column
name	CompanyName
login	EmailAddress
email	EmailAddress
telephone	Phone

4.7 Import

When you click next in this step the import starts. When the import is finished it will report what it did. If there were any errors it will also report the errors.



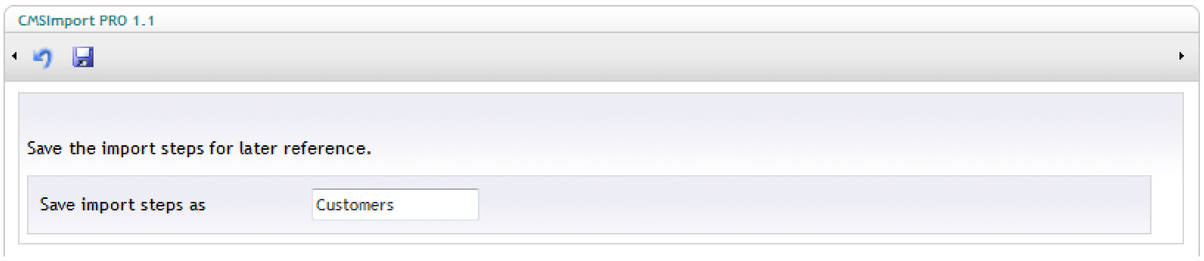
Import Finished

Import is finished

Import duration (h:m:s)	00:01:29
Records retrieved	440
Records imported	440
Errors	0

4.8 Save Import steps

When you click save, you can specify a name and when hitting the save button again the import steps are saved for later use.



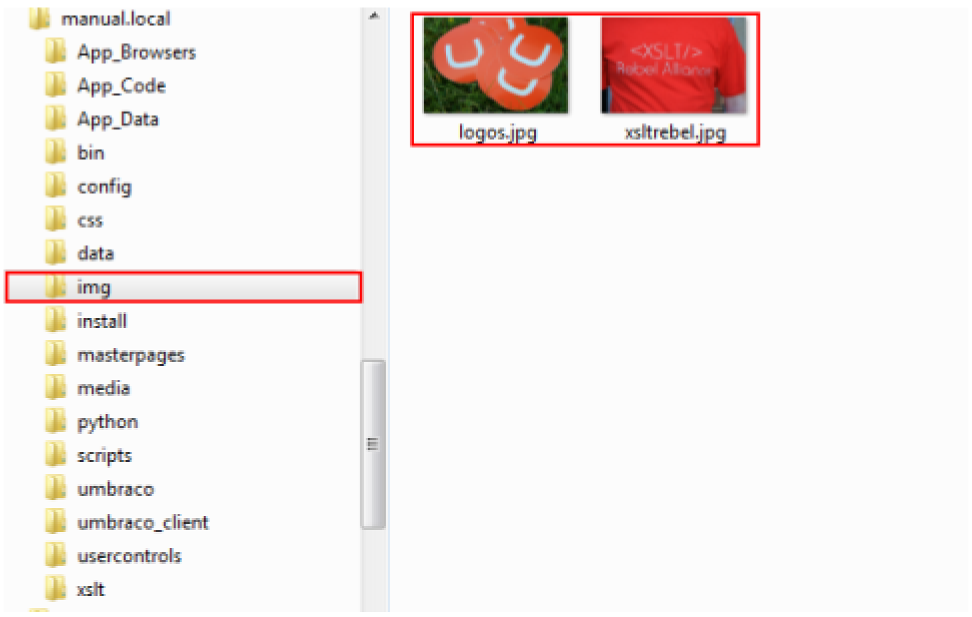
Save the import steps for later reference.

Save import steps as: Customers

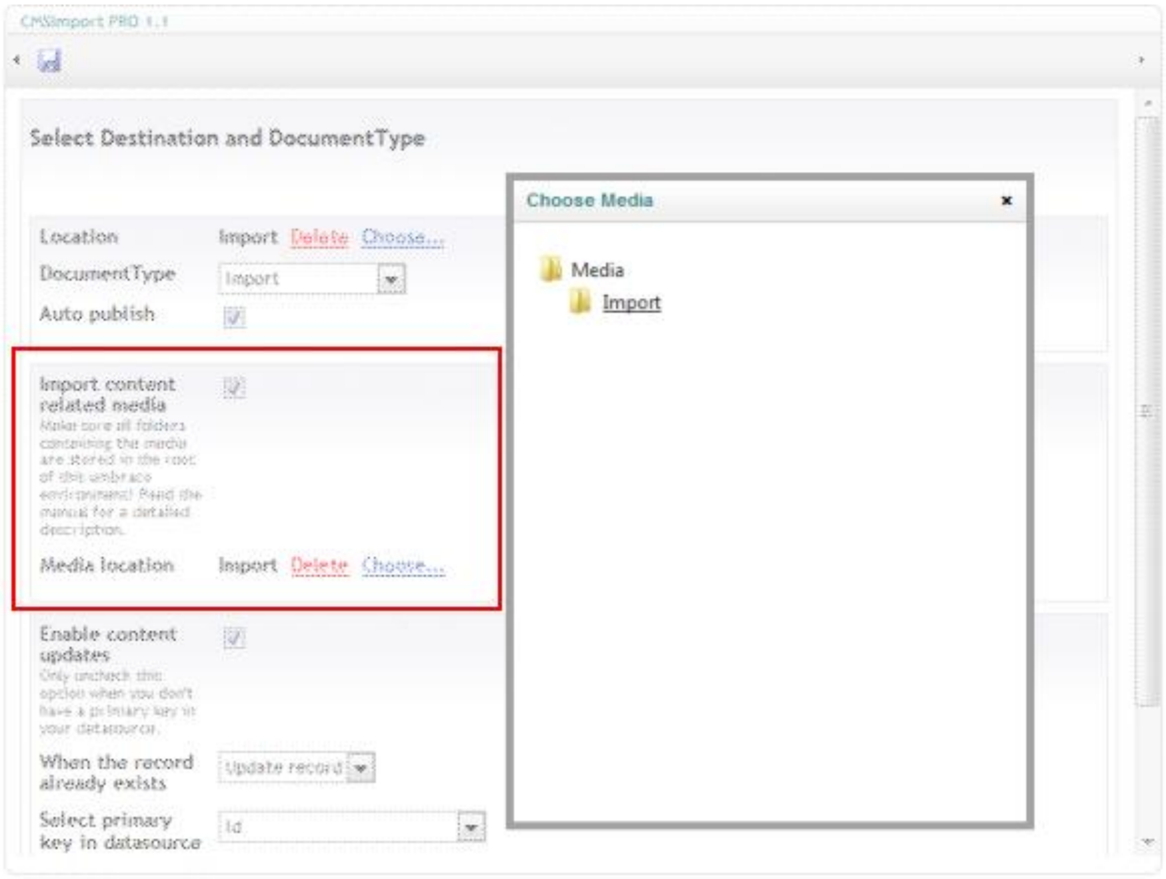
5 Related media import

CMSImport can import media also. This isn't a separate import process but integrated in content or media import. When CMSImport finds a reference to a relative path it will try to get the item and convert it to a media item, or store it in the media folder in case of an upload field.
The only required thing is that the original media folder is copied to the root of your Umbraco folder.

In the example below the img folder of the original site containing two images is stored in the Umbraco root.



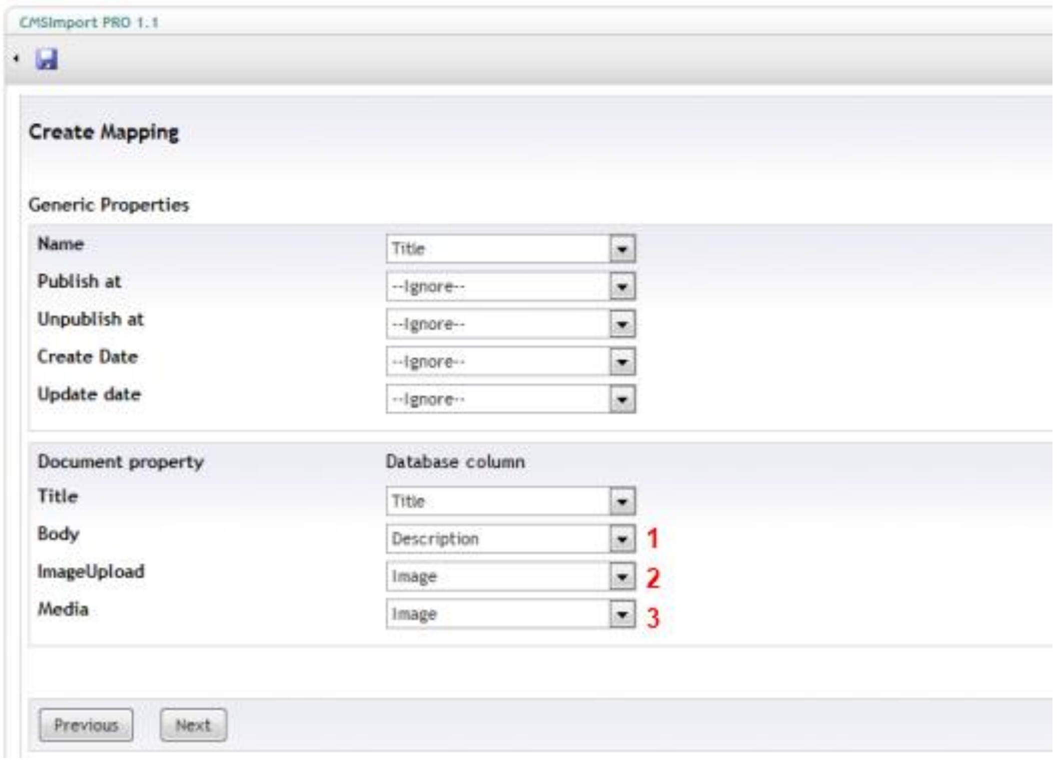
In the Import wizard you can specify that you want to import media items also. Check the option "Import content related media". If you want to map media against media pickers and/or the TinyMCE editor, you need to specify a media folder also. CMSImport will keep the imported folder structure.



Select Destination and DocumentType

Location	Import Delete Choose...
DocumentType	Import
Auto publish	✓
Import content related media Make sure all folders containing the media are stored in the root of the umbraco environment! Read the manual for a detailed description.	✓
Media location	Import Delete Choose...
Enable content updates Only uncheck this option when you don't have a primary key in your datasource.	✓
When the record already exists	Update record
Select primary key in datasource	Id

Then in the next step you can create the mapping like you would normally do.



Create Mapping

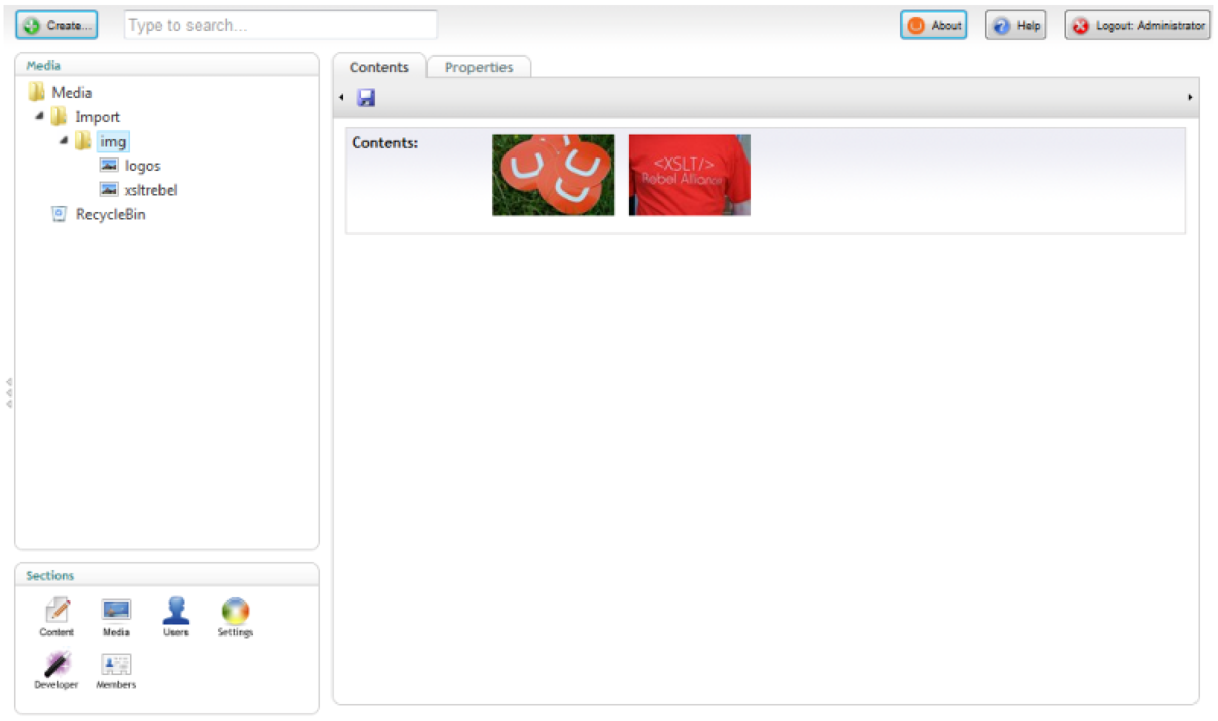
Generic Properties

Name	Title
Publish at	--Ignore--
Unpublish at	--Ignore--
Create Date	--Ignore--
Update date	--Ignore--

Document property	Database column
Title	Title
Body	Description
ImageUpload	Image
Media	Image

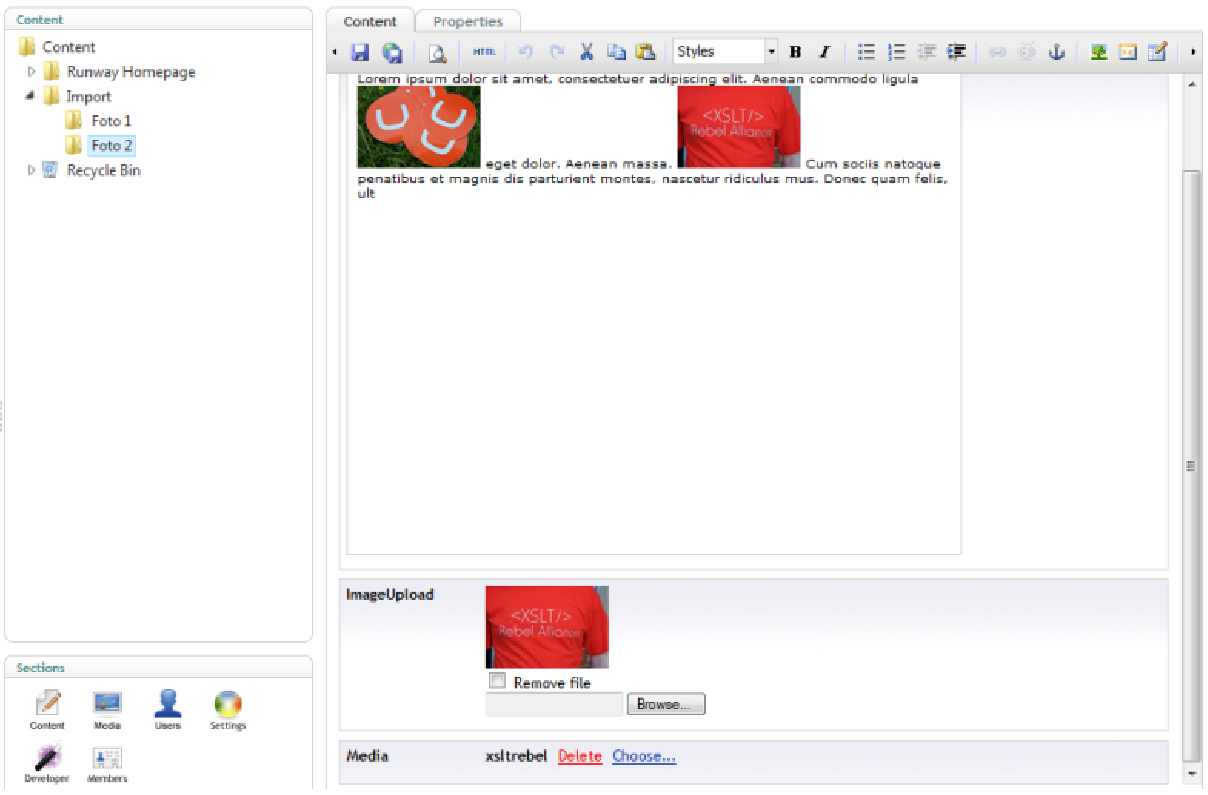
1. When a reference to an image is found in the content, CMSImport will create a media Item and update the image source to the new Media item. [1]
2. An image (could also be a file) reference is mapped against an upload field. CMSImport will store the image in the Umbraco Media folder and update the reference in the Upload field
3. An image (could also be a file) reference is mapped against a media picker. CMSImport will create a media item and store the Id of the media item .

When the import process is finished you'll see that the media items are imported



And when you open an imported item in the content section you'll see that all the references are updated to the imported media items

[1] Currently only images are supported in Rich Text. Version 1.2 will have support for files also. Media Pickers and Upload fields will work with files also.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ult

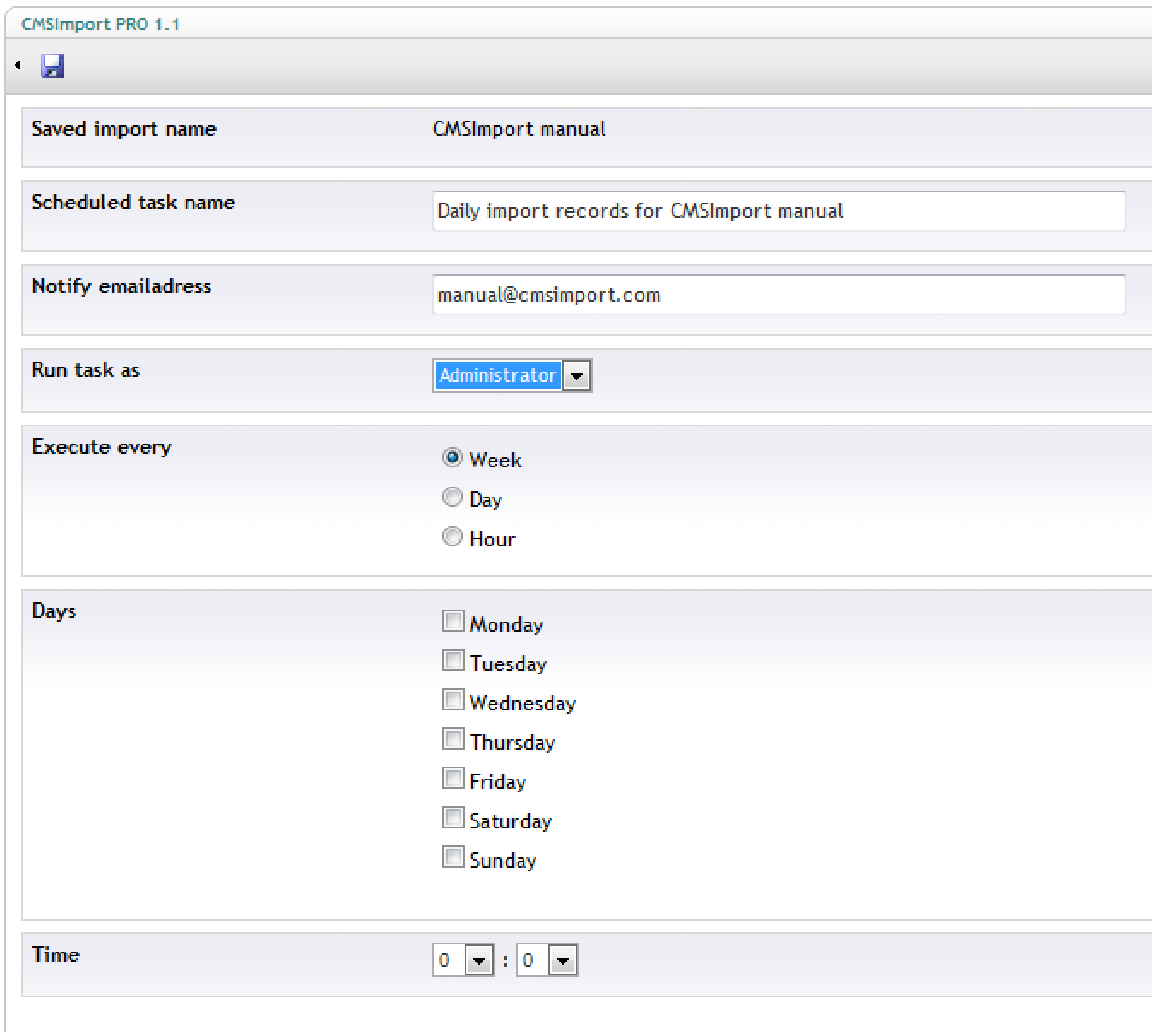
Currently this import process will work for the following datatypes:

- Upload field
- Media Picker
- TinyMCE (Images only)

In future releases CMSImport will support custom datatypes and File import for the TinyMCE datatype also.

6 Schedule Imports

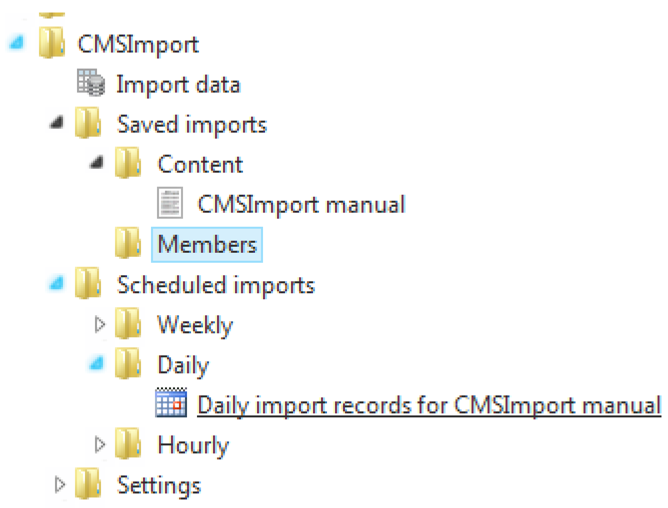
With CMSImport it's possible to run imports for on a certain day/time. When you right click on a saved import item (content/member) and click schedule the following screen will show up.



Saved import name	CMSImport manual
Scheduled task name	Daily import records for CMSImport manual
Notify emailadress	manual@cmsimport.com
Run task as	Administrator
Execute every	● Week ○ Day ○ Hour
Days	☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday ☐ Sunday
Time	0 : 0

You can schedule imports to run every week on certain days/time, every day on a certain time or every hour.

You can specify an email address to receive a notification when the import is finished and you can specify a user that will be assigned as creator for imported documents/members. When you click save the item will run on the selected day/time.



When the import is finished you will receive a notification email on the specified email address.

7 Settings

Here you can modify the following settings

7.1 Login credential settings

These settings will be used when an email with login credentials is send to an imported member. You can specify:

- The from email address
- The email subject
- The email body

In the email body you can use the following placeholders that will be replaced with real values when sending the email:

Snippet	Description
[#loginname]	The loginname of the imported member
[#password]	The password (NOT ENCRYPTED) of the imported member
[#email]	The Email address of the imported member
[#{property alias}]	This will replace the property alias with the imported value

7.2 Scheduler result settings

These settings will be used when an email is send to inform a user that a scheduled task is finished. You can specify:

- The from email address
- The email subject
- The email body

In the email body you can use the following placeholders that will be replaced with real values when sending the email:

Snippet	Description
[#Taskname]	The name of the scheduled task
[#Duration]	The duration of the import process
[#RecordCount]	The amount of records in the datasource
[#RecordsAdded]	The amount of records added during the import process
[#RecordsUpdated]	The amount of records updated during the import process
[#RecorsdSkippedCount]	The amount of records skipped during the import process
[#ErrorCount]	The amount of errors during the import process
[#Errors]	The error descriptions that occurred during the import process

8 Extend CMSImport

Although you don't need to, there are several ways to extend CMSImport. This chapter describes how you can make use of these extension points in your code. All samples can be downloaded from http://www.cmsimport.com/download/cmsimportsamplepackage.zip and a sample VS2010 solution can be downloaded from http://www.cmsimport.com/download/CMSImportVs2010Samples.zip . To use the samples you need to install the AdventureWorks Lite database which can be downloaded from the CodePlex site http://msftdbprodsamples.codeplex.com/ and use CMSImport PRO(although some samples might work in the free version)

8.1 Setting up Visual Studio

When you want to create an extension for CMSImport you can create a new Class Library and add a reference to the Assembly **CMSImport.Extensions**. When you need to use Umbraco functionality, or want to use CMSImport events also you can create references to the assemblies Umbraco, interfaces, cms and businesslogic.

8.2 What's in CMSImport.extensions

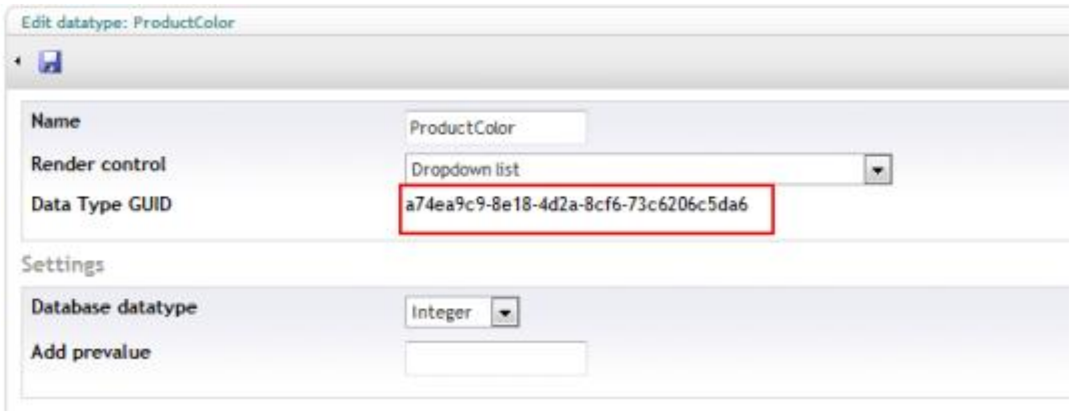
8.2.1 DataAdapter

A DataAdapter is used as a generic interface to talk to datasources. You can use a DataAdapter by using ImportDataAdapter as a base class. Checkout the samples for a full implementation of a DataAdapter.

8.2.2 FieldAdapter

A FieldAdapter could be used to convert original data in a datasource to a datatype specific format during import. CMSImport uses this already to make sure values like "true/false" will be converted to a Boolean value which could be mapped against a true/false datatype without causing an error and it uses FieldAdapters to import media.

A FieldAdapter is basically an implementation of the IFieldAdapter interface, which contains the DataAdapterId property and the Parse method. The DataAdapterID must match with the DataAdapterControl . This will ensure that the parse method gets called when data is imported for the Color property.


Edit datatype: ProductColor

Name	ProductColor
Render control	Dropdown list
Data Type GUID	a74ea9c9-8e18-4d2a-8cf6-73c6206c5da6

Settings

Database datatype	Integer
Add prevalue	

This Parse method accepts the following parameters:

- Value.	The original value which we can manipulate
- Document property.	Gives us information about the document type
- FieldAdapterOptions	Bunch of general setting mostly related to Media Import

8.2.3 Events

There are several events that you can use for both content and member import. You can hook up events in the same way you hook up events for other Umbraco functionality by deriving from ApplicationBase and hook up the event in the constructor.

```
public class Sample :umbraco.BusinessLogic.ApplicationBase
{
    public Sample()
    {
       //Hook up event
       MemberImport.RecordImporting +=
       new MemberImport.RecordImportingEventHandler (MemberImport_RecordImporting);
    }

    void MemberImport_RecordImporting(object sender, RecordImportingEventArgs e)
    {
        //Logic here
    }
}
```

8.2.3.1 RecordImporting

The recordImporting event gets hit before a record got imported, the sender object contains the document or member being imported. RecordImportingEventArgs contains the following information:

- DataAdapter
- DataKeyName (Primary key column)
- DataKeyValue (Primary key value)
- Items (collection of items that contains the original data)

You can cancel the import for this record by setting the cancel property to true.

8.2.3.2 RecordImported

The RecordImported event gets hit after a record got imported, the sender object contains the imported document or member. RecordImportedEventArgs contains the following information:

- DataAdapter
- DataKeyName (Primary key column)
- DataKeyValue (Primary key value)
- ImportAction (ImportAsNew, update or skip record)
- Items (collection of items that contains the original data)

8.2.3.3 Importing

The Importing event gets hit before the import starts, ImportEventArgs contains the alias of the DataAdapter.

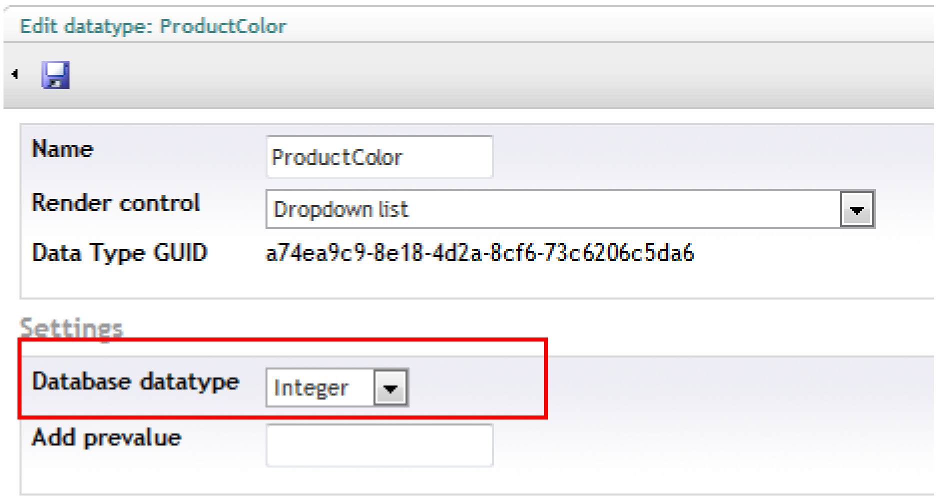
8.2.3.4 Imported

The Imported event gets hit after the import has finished, ImportEventArgs contains the alias of the DataAdapter.

8.3 Samples

When you've downloaded and installed the samples you can use two extra document types in your Umbraco installation, ProductCategory and Product.

Before you start make sure the ProductColor datatype uses the integer Database datatype.



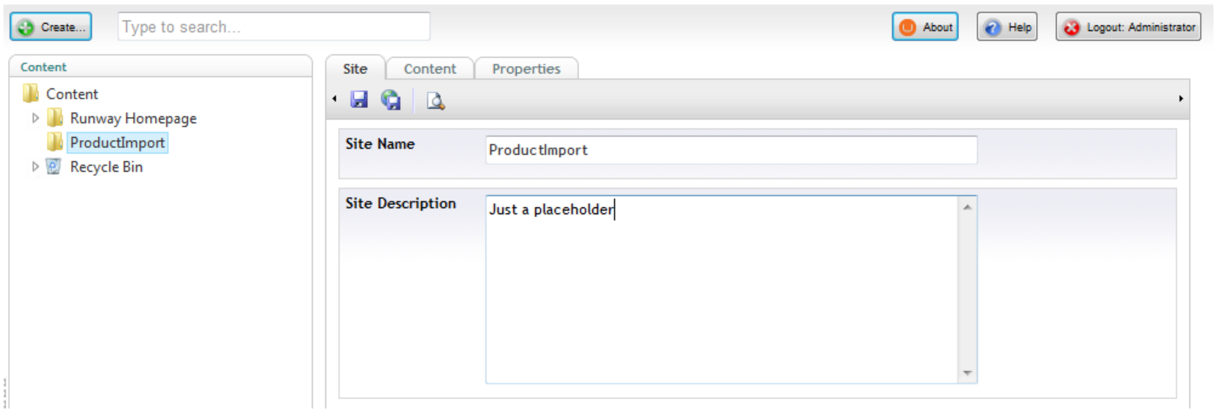
Edit datatype: ProductColor

Name	ProductColor
Render control	Dropdown list
Data Type GUID	a74ea9c9-8e18-4d2a-8cf6-73c6206c5da6

Settings

Database datatype	Integer
Add prevalue	

All samples will use the above mentioned document types. Only thing you need to do is to create a root folder and use that folder as import location for all the samples.



Site Name	ProductImport
Site Description	Just a placeholder

8.4 Create a DataAdapter for Product Categories

A DataAdapter could be used to communicate with a DataSource. Basically it lets you set some properties and returns data to CMSImport which can then be imported. In the Sample project we are using two DataAdapters

1. ProductCategoryAdapter, responsible for importing all product category data
2. ProductDataAdapter, responsible for importing all product data.

8.4.1 Create the class

To create a custom DataAdapter you need to create a class that derives from **CMSImport.Extensions.DataAdapter.ImportDataAdapter**

```
public class AdventureWorksLTProductCategeoriesDataAdapter : ImportDataAdapter
{
.....
}
```

8.4.2 Specify an alias

The alias you specify by overriding the Alias property will be shown in the pulldownlist where the user can pick a DataAdapter. This alias must be unique.

```
public override string Alias
{
        get { return "AdventureWorks ProductCategories"; }
}
```

8.4.3 Add UI

The ImportAdapter class itself derives from the Webcontrol Class, so we can directly add UI Controls to this class.

```
private Panel _contentPanel = new Panel();
private TextBox _datasourceTextBox = new TextBox();
private Literal _selectDataSourceLiteral = new Literal();

protected override void OnInit(EventArgs e)
{
        base.OnInit(e);
        _contentPanel.ID = "CategorycontentPanel";
        _contentPanel.CssClass = "propertypane";

        //Labels
        _selectDataSourceLiteral.ID = "SelectDataSourceLiteral";
        _selectDataSourceLiteral.Text = "Enter the connection string";

        //TextBox
        _datasourceTextBox.ID = "datasource";
        _datasourceTextBox.CssClass = "umbEditorTextField";
        _datasourceTextBox.Text = DataSource;

        //Create Layout
        _contentPanel.Controls.Add(_selectDataSourceLiteral);
        _contentPanel.Controls.Add(new LiteralControl("&nbsp;&nbsp;"));
        _contentPanel.Controls.Add(_datasourceTextBox);

        Controls.Add(_contentPanel);
}
```

8.4.4 Update DataAdapter values

When the user clicks next you need to update the DataAdapter properties, in this case DataSource and DataCommand.

```
public override void UpdateAdapter()
{
        //Uses the SqlConnection from the TextBox as the Datasource
        DataSource = _datasourceTextBox.Text;
        //We are creating a custom DataAdapter for a productCategories, don't bother
        //the user with sql
        DataCommand = "Select * from SalesLT.ProductCategory order by
        ParentProductCategoryID, Name";
}
```

8.4.5 Validate

When the user clicks next the validate method is called **after the DataAdapter values are updated.** Return true in this method when the DataSource is valid, false when not valid

```
public override bool Validate()
{
        bool result = true;
        try
             {
                 using (SqlConnection sqlConnection = new SqlConnection(DataSource))
                 {
                     sqlConnection.Open();
                     SqlCommand sqlCommand = new SqlCommand(DataCommand,
                     sqlConnection);
                     sqlCommand.ExecuteReader();
                 }
             }
         catch (Exception ex)
             {
                 //Cannot validate against the Datasource;
                 result = false;
                 //Set the ValidationErrorMessage with the correct error.
                 ValidationErrorMessage = string.Format("Error validating the Data
source: {0}", ex.Message);
             }
             return result;
         }
```

8.4.6 GetData

The GetData method gets called by CMSImport during the import process and it will return the data from DataSource as an IDataReader. In the example below you'll see the GetData Method that we are using in the AdventureWorks ProductCategory DataAdapter

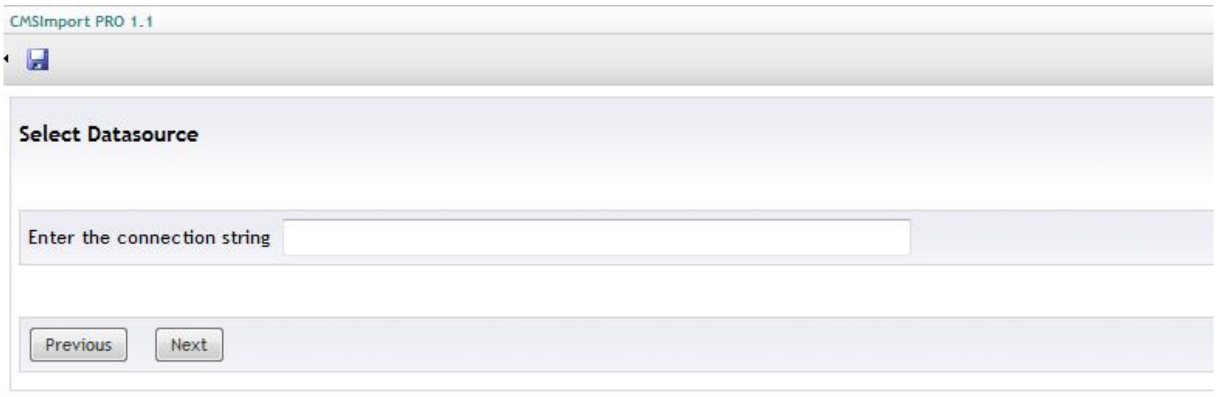
```
public override IDataReader GetData()
{
        SqlConnection sqlConnection = new SqlConnection(DataSource);
        sqlConnection.Open();
        SqlCommand sqlCommand = new SqlCommand(DataCommand, sqlConnection);
        return sqlCommand.ExecuteReader();
}
```

CMSImport will dispose the Reader once it's finished with the import.

If you need to convert XML to IDataReader you can use the helper method XmlToDataReader which takes an xml file/url and xpath Expression. Currently this only works on elements attributes will be ignored.

8.4.7 End result

When we will build the dll and drop it into the bin folder of the Umbraco install we will see the “AdventureWorks ProductCategory” in the pulldown list of possible datasources and when we select the AdventureWorks ProductCategory DataAdapter we will see the following screen:



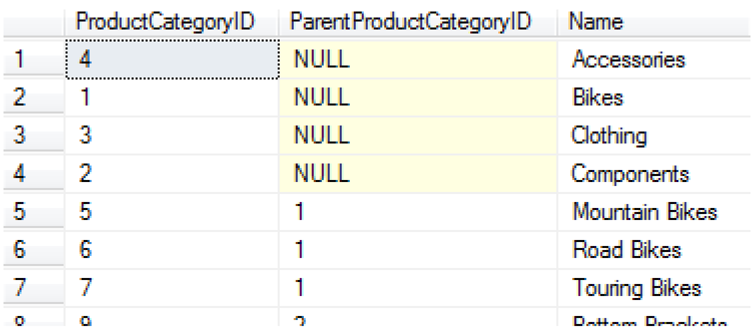
We can provide a Connection string and click next to continue the import process the normal way.

8.5 Create a DataAdapter for Products.

Since we will be importing both Categories and Products we want to have a Custom DataAdapter for products also. Do this by copying the ProductCategories DataAdapter and name it AdventureWorksProducts, rename the alias to “AdventureWorks Products” and replace the query in the UpdateAdapter method to “SELECT * FROM SalesLT.Product order by Name”

8.6 Using events to maintain structure

Now that we have our DataAdapters ready we can use them to import the data. But when we take a closer look at the ProductCategory data we see that it uses the ParentProductCategoryID column to maintain the structure.

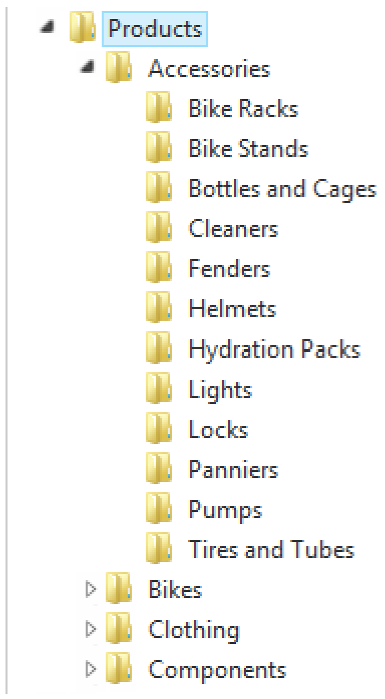


	ProductCategoryID	ParentProductCategoryID	Name
1	4	NULL	Accessories
2	1	NULL	Bikes
3	3	NULL	Clothing
4	2	NULL	Components
5	5	1	Mountain Bikes
6	6	1	Road Bikes
7	7	1	Touring Bikes
8	8	2	Bottom Brackets

It would be nice to keep the structure during import. Events make that possible. Once a record is imported the RecordImported event is fired. This will pass the document as sender and ImportedEventArguments. The RecordImportedEventArguments contains a hashtable called Items. This hastable contains a readonly reference to the orginal data. So even we did not map the ParentProductCatagoryID column, we can still get the parent ID. This comes handy because CMSImport adds relations between the original imported data and Umbraco documents. Internally it uses this to determine if a record needs to be imported ad new or updated. You can retrieve this information yourself by using the helper method **ContentRelation.GetRelatedDocument**. This method accepts the following parameters:

- DataAdapter alias.
- Primary key column
- The primary key

For product categories data we can use the current DataAdapter alias and Primary key. We only need the original primary key which is ParentProductCategoryID. Based on this information the parent document will be returned and we can move the imported document to its parent. As you can see in the image below



Below you'll find the complete source code for this event.

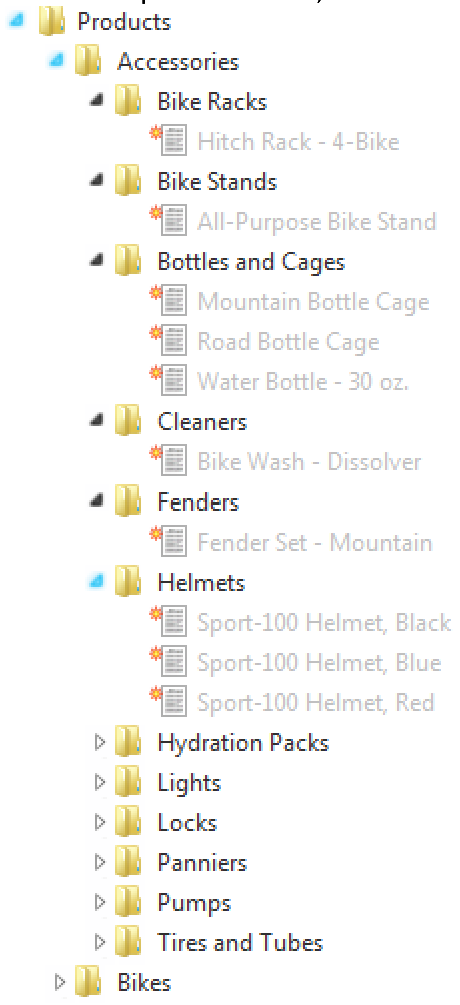
```
public class MaintainProductCategoryStructure : ApplicationBase
{
        /// <summary>
        /// Hook up the RecordImporting event so it gets hit during import
        /// </summary>
        public MaintainProductCategoryStructure()
        {
                ContentImport.RecordImported += new
                ContentImport.RecordImportedEventHandler(ContentImport_RecordImported);
        }

        void ContentImport_RecordImported(object sender, RecordImportedEventArgs e)
         {
                Document doc = sender as Document;
                //Only execute when the document and parent not is null and the
                dataAdapter is AdventureWorks ProductCategories
                if (doc != null && e.DataAdapter == "AdventureWorks ProductCategories" &&
                e.Items["ParentProductCategoryID"] != null)
                {
                        //Use the Content
                        Document target =
                        ContentRelation.GetRelatedDocument(e.DataAdapter, e.DataKeyName,
                        e.Items["ParentProductCategoryID"]);
                        if (target != null)
```

```
                    {
                            doc.Move(target.Id);
                    }
               }
          }
     }
```

We can use the same mechanism to import products into their related Product Category. The following snippet will retrieve the ProductCategory document . See that it's using the ProductCategories dataTypeAlias and ProductCategoryID to determine the original data.

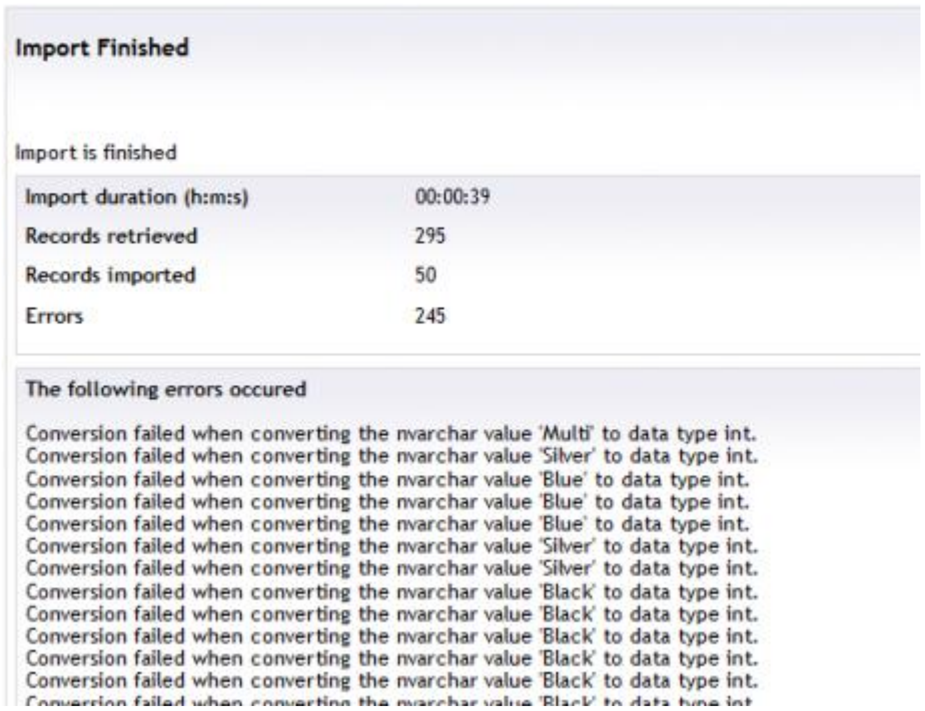
```
Document productCategory = ContentRelation.GetRelatedDocument("AdventureWorks
ProductCategories", "ProductCategoryID", e.Items["ProductCategoryID"]);
```

When the import is finished, the end result would look like the image below.


Products
Accessories
Bike Racks
Hitch Rack - 4-Bike
Bike Stands
All-Purpose Bike Stand
Bottles and Cages
Mountain Bottle Cage
Road Bottle Cage
Water Bottle - 30 oz.
Cleaners
Bike Wash - Dissolver
Fenders
Fender Set - Mountain
Helmets
Sport-100 Helmet, Black
Sport-100 Helmet, Blue
Sport-100 Helmet, Red
Hydration Packs
Lights
Locks
Panniers
Pumps
Tires and Tubes
Bikes

8.7 Using a FieldAdapter to map text values against a Dropdownlist DataType

As you might have seen our Product DocumentType contains a property ProductColor which is based on the ProductColor Dropdownlist. When we map the color column form the AdventureWorks Products table directly against the ProductColor document property the import will fail because it can't convert text to integer values.


Import Finished

Import is finished

Import duration (h:m:s)	00:00:39
Records retrieved	295
Records imported	50
Errors	245

The following errors occured

```
Conversion failed when converting the nvarchar value 'Multi' to data type int.
Conversion failed when converting the nvarchar value 'Silver' to data type int.
Conversion failed when converting the nvarchar value 'Blue' to data type int.
Conversion failed when converting the nvarchar value 'Blue' to data type int.
Conversion failed when converting the nvarchar value 'Blue' to data type int.
Conversion failed when converting the nvarchar value 'Silver' to data type int.
Conversion failed when converting the nvarchar value 'Silver' to data type int.
Conversion failed when converting the nvarchar value 'Black' to data type int.
Conversion failed when converting the nvarchar value 'Black' to data type int.
Conversion failed when converting the nvarchar value 'Black' to data type int.
Conversion failed when converting the nvarchar value 'Black' to data type int.
Conversion failed when converting the nvarchar value 'Black' to data type int.
Conversion failed when converting the nvarchar value 'Black' to data type int.
```

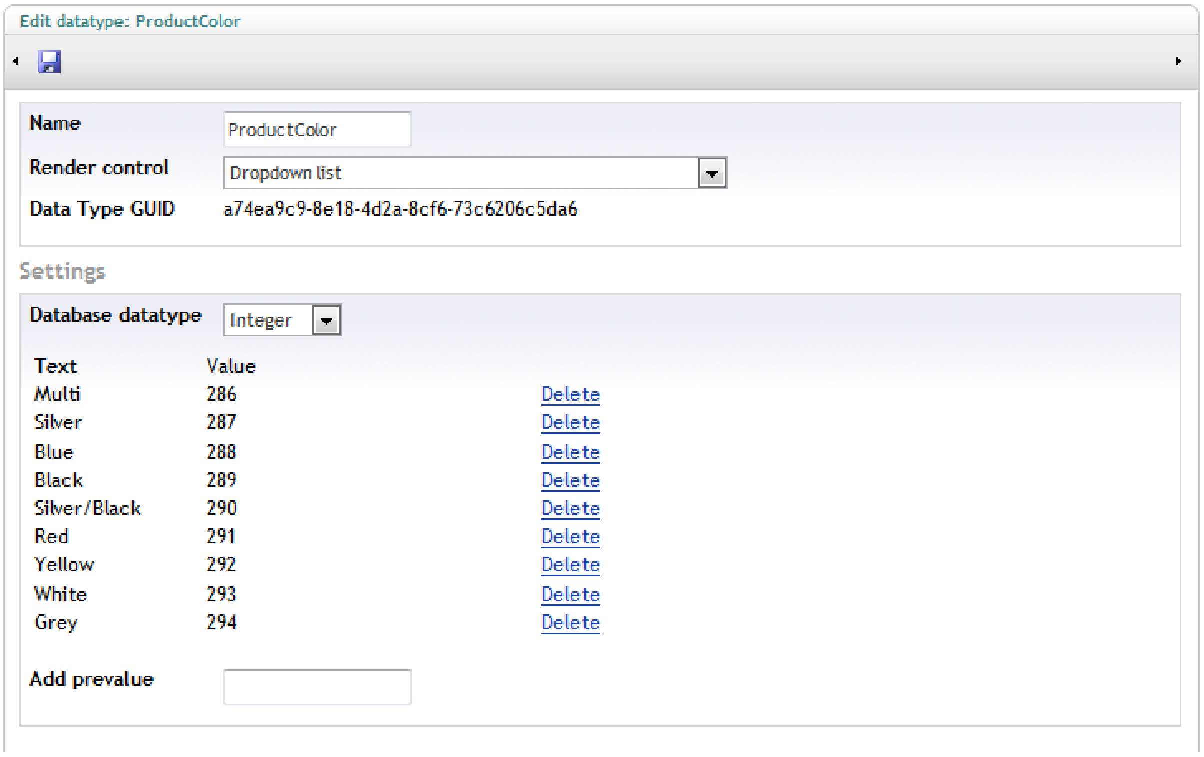
This can be solved using a FieldAdapter

For our implementation we want to retrieve the Id of the prevalue and return that id instead of the text value. Additional it would also be nice if we can create the prevalue when it doesn't exists so we don't need to create the prevalues first.

Below you find the FieldAdapter. The DataTypeId returns the Guid of the Dropdown Datatype. The parse method first checks if the datatype is the ProductColor datatype and when it is the ProductColor datatype it will get (or create) the integer id of the prevalue.

```
    /// </summary>
    public class AutoAddProductColorToList : IFieldAdapter
    {
        /// <summary>
        /// Gets the data type id.
        /// </summary>
        /// <value>The data type id.</value>
        public Guid DataTypeId
        {
            get { return new Guid("a74ea9c9-8e18-4d2a-8cf6-73c6206c5da6 "); }
        }

        public object Parse(object value, umbraco.cms.businesslogic.property.Property
property, FieldAdapterOptions fieldAdapterOptions)
        {
            if (property.PropertyType.DataTypeDefinition.Text == "ProductColor")
            {
                DataTypeDefinition dt = property.PropertyType.DataTypeDefinition;
                value = GetOrCreatePrevalue(dt.Id, value.ToString());
            }
            return value;
        }
    }
}
```

Once the import has finished using this FieldAdapter you'll see that all options are added to the ProductColor datatype.



Edit datatype: ProductColor

Name	ProductColor
Render control	Dropdown list
Data Type GUID	a74ea9c9-8e18-4d2a-8cf6-73c6206c5da6

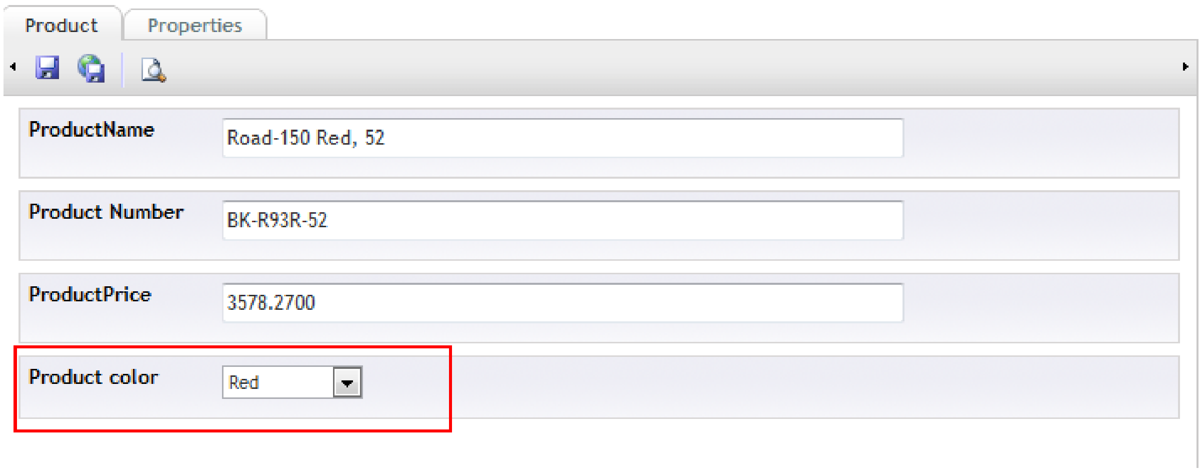
Settings

Database datatype: Integer

Text	Value	
Multi	286	Delete
Silver	287	Delete
Blue	288	Delete
Black	289	Delete
Silver/Black	290	Delete
Red	291	Delete
Yellow	292	Delete
White	293	Delete
Grey	294	Delete

Add prevalue

And you'll see that these Prevalues are mapped correctly against the document



ProductName	Road-150 Red, 52
Product Number	BK-R93R-52
ProductPrice	3578.2700
Product color	Red

9 Manual Installation/Configuration

If you renamed the Umbraco folder or for some reason can't give the installer sufficient rights to create tables in the database , or the sufficient rights to modify the following folders /bin, /config, /umbraco you need to install CMSImport PRO Manually.

9.1 Manual installation of files

- Open de folder in the zip file.
- Copy all **.dll** files to the /bin folder of your Umbraco installation.
- Open the /umbraco/plugins/ folder.
- Create the folder **CMSImport**.
- Create the following folders in the **/umbraco/plugins/CMSImport** folder.
  - Config
  - Handlers
  - Pages
  - Usercontrols
- In the **/umbraco/plugins/CMSImport /Usercontrols** folder create the folder **ImportSteps**
- In the **/umbraco/plugins/CMSImport /Usercontrols/ImportSteps** folder create the folder **ContentImport**
- In the **/umbraco/plugins/CMSImport /Usercontrols/ImportSteps** folder create the folder **MemberImport**
- Copy all .config and .license files from the zip file to the folder **/umbraco/plugins/CMSImport/Config/**
- Copy all .ashx files from the zip file to the folder **/umbraco/plugins/CMSImport/Handlers/**
- Copy all .aspx files from the zip file to the folder **/umbraco/plugins/CMSImport/Pages/**
- Copy the files **CMSImport.ascx** and **CMSImportInstaller.ascx** from the zip file to the folder **/umbraco/plugins/CMSImport/Usercontrols/**
- Copy the files **ConfirmSelectedOptions.ascx**, **Importing.ascx, Intro.ascx, MapProperties.ascx, SelectDataSource.ascx, SelectDataSourceType.ascx** from the zip file to the folder **/umbraco/plugins/CMSImport/Usercontrols/ImportSteps/**
- Copy **SelectUmbracoTypeAndLocation.ascx** from the zip file to the folder **/umbraco/plugins/CMSImport/Usercontrols/ImportSteps/ContentImport**
- Copy **SelectMembertype.ascx** from the zip file to the folder **/umbraco/plugins/CMSImport/Usercontrols/ImportSteps/MemberImport**
- Copy all the png files from the zip file to the folder **/umbraco/Images/**

9.2 Manual configuration of Database

Run the following script to install the database tables

```
CREATE TABLE [dbo].[CMSImportState](
          [Id] [int] IDENTITY(1,1) NOT NULL,
          [UniqueIdentifier] [uniqueidentifier] NOT NULL,
          [Name] [nvarchar](250) NOT NULL,
          [ImportType] [nvarchar](250) NOT NULL,
          [ImportState] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_CMSImportState] PRIMARY KEY CLUSTERED
(
          [Id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[CMSImportScheduledTask](
          [ScheduleId] [int] IDENTITY(1,1) NOT NULL,
          [ScheduleGUID] [uniqueidentifier] NOT NULL,
          [ImportStateGUID] [uniqueidentifier] NOT NULL,
	[ScheduledTaskName] [nvarchar](50) NOT NULL,
	[NotifyEmailAddress] [nvarchar](250) NOT NULL,
	[ExecuteEvery] [nvarchar](50) NOT NULL,
	[ExecuteDays] [nvarchar](50) NOT NULL,
	[ExecuteHour] [int] NOT NULL,
	[ExecuteMinute] [int] NOT NULL,
 CONSTRAINT [PK_CMSImportScheduledTask] PRIMARY KEY CLUSTERED 
(
	[ScheduleId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]


CREATE TABLE [dbo].[CMSImportRelation](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[UmbracoID] [int] NOT NULL,
	[UmbracoParentId] [int] NOT NULL,
	[DataSourceKey] [nvarchar](250) NOT NULL,
 CONSTRAINT [PK_CMSImportRelation] PRIMARY KEY CLUSTERED 
(
	[Id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[CMSImportMediaRelation](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[UmbracoMediaId] [int] NOT NULL,
	[SourceUrl] [nvarchar](500) NOT NULL,
 CONSTRAINT [PK_CMSImportMediaRelation] PRIMARY KEY CLUSTERED 
(
	[Id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

INSERT [umbracoAppTree] ([treeSilent], [treeInitialize], [treeSortOrder], [appAlias],
[treeAlias], [treeTitle], [treeIconClosed], [treeIconOpen], [treeHandlerAssembly],
[treeHandlerType], [action]) VALUES (0, 0, 16, N'developer', N'CMSImportSettings',
N'Settings', N'folder.gif', N'folder_o.gif', N'CMSImportLibrary',
N'ApplicationTree.ImportSettingsTree', N'')
INSERT [umbracoAppTree] ([treeSilent], [treeInitialize], [treeSortOrder], [appAlias],
[treeAlias], [treeTitle], [treeIconClosed], [treeIconOpen], [treeHandlerAssembly],
[treeHandlerType], [action]) VALUES (0, 1, 10, N'developer', N'CMSImportWizard', N'CMSImport',
N'folder.gif', N'folder_o.gif', N'CMSImportProLibrary', N'ApplicationTree.CMSImportProTree',
N'')
INSERT [umbracoAppTree] ([treeSilent], [treeInitialize], [treeSortOrder], [appAlias],
[treeAlias], [treeTitle], [treeIconClosed], [treeIconOpen], [treeHandlerAssembly],
[treeHandlerType], [action]) VALUES (0, 0, 11, N'developer', N'CMSImportWizardSaveImportTree',
N'Saved Imports', N'folder.gif', N'folder_o.gif', N'CMSImportProLibrary',
N'ApplicationTree.SavedImportTree', N'')
INSERT [umbracoAppTree] ([treeSilent], [treeInitialize], [treeSortOrder], [appAlias],
[treeAlias], [treeTitle], [treeIconClosed], [treeIconOpen], [treeHandlerAssembly],
[treeHandlerType], [action]) VALUES (0, 0, 15, N'developer',
N'CMSImportWizardScheduleImportTree', N'Scheduled Imports', N'folder.gif', N'folder_o.gif',
N'CMSImportProLibrary', N'ApplicationTree.ScheduledTaskTree', N'')
```

9.3 Manual configuration of the language files

When the install failed due insufficient rights of the installer. It’s better to assign sufficient rights to the /umbraco/config/lang/ folder (and all the xml files in that folder). Then start CMSImport PRO again. CMSImport Pro will determine that the language file is not updated and will automatically update the language files again.

If for some reason this isn’t the case you can modify the files manually. To do this open the necessary language files. For example if you use the English language in Umbraco open en.xml, if you use the Dutch language in Umbraco open nl.xml etc.

Replace

```
<area alias="actions">
```

With

```
<area alias="actions">
<key alias="DeleteCMSImportAction" version="4.0">Delete</key>
<key alias="ExecuteCMSImportWizardAction" version="4.0">Execute</key>
<key alias="ScheduleCMSImportWizardAction" version="4.0">Schedule</key>
<key alias="StartCMSImportWizardAction" version="4.0">Import Data</key>
```

9.4 Manual configuration of the scheduled task handler

Open the /config/umbracoSettings.config file.

Replace

```
</scheduledTasks>
```

With

```
<task log="true" alias="CMSImportScheduler" interval="60" url="[url to
mysite/umbraco/]/plugins/CMSImport/Handlers/ScheduleTaskHandler.ashx" />
</scheduledTasks>
```

And replace [url to mysite/umbraco/] with the url of your site including the umbraco path. For example: http://cmsimport.com/umbraco//plugins/CMSImport/Handlers/ScheduleTaskHandler.ashx

When the scheduled task handler is configured correctly and you browse to that url using your favorite browser it should report “CMSImport scheduler”

10 Troubleshooting

10.1 I don't see the CMSImport package in my developer section

Make sure you have sufficient rights to install the package. See chapter 2, otherwise perform a manual installation see chapter94.

10.2 I don't see my column names when importing from a CSV file.

Make sure that your csv file contains column names

10.3 I get weird column names when importing from a CSV file.

Make sure that you set the correct csv options to display the CSV file. For example, choose **;** as the delimiter and **"** as a string indicator.

10.4 I can't import attributes when importing from an xml file.

Currently only elements are supported.

10.5 Email is not send when importing a member

Make sure you have configured your smtp server in your web.config file. Also check the UmbracoLog table for SMTP errors.

10.6 I get an Invalid License exception.

Make sure you've bought the correct license for the (sub)domain , or an enterprise license. Contact support@soetemansoftware.nl for help.