EXT: Formhandler Subscription

Extension Key: formhandler_subscription

Language: en

Version: 0.0.1

Keywords:

Table of Contents

Introduction

What does it do?

The formhandler_subscription extension provides additional utilities (Finishers, PreProcessors) for the formhandler extension. With these utilites you can build a complete (newsletter) subscription system that includes the subscription itself and updating or deleting a subscription.

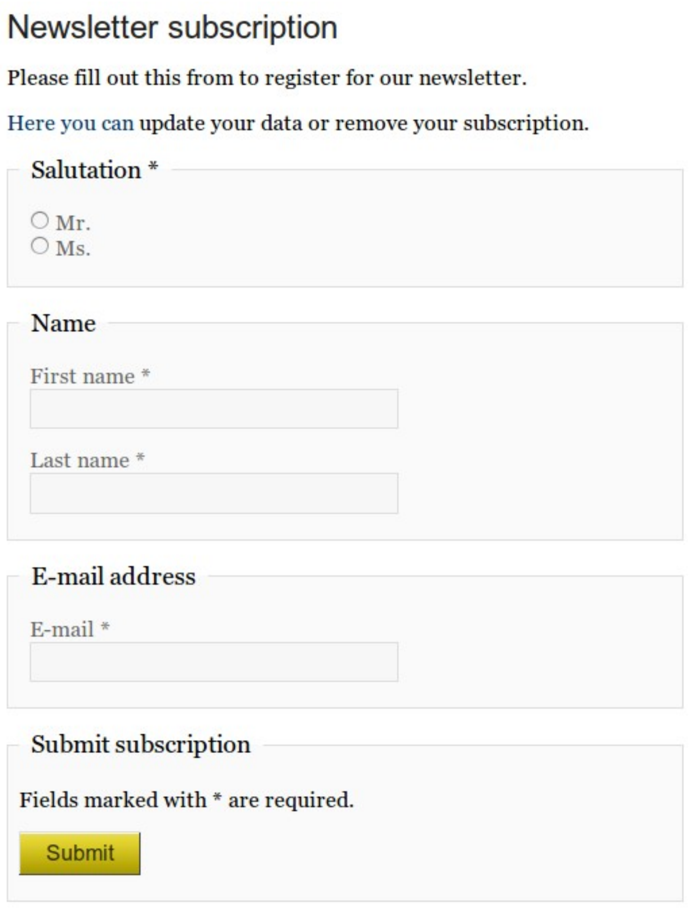
Features and targets of this extention

- provide a good set of default templates and texts (HTML markup of the templates is based on http://www.yaml.de, especially the forms are based on the YAML Form Construction Kit)
- be flexible (since formhandler is really flexible this target is not too hard to archive)
- be stealthy: do not let users know if an email address is already registered
- provide good documentation
- keep it simple: since formhandler already provides a huge feature set we should be able to keep this extension lightweight

To use this extension you will need a general understanding how the formhandler extension works, so if you do not know it have a look at http://www.typo3-formhandler.com/.

Screenshots

This is the default registration form that is included in the extension:



Newsletter subscription

Please fill out this from to register for our newsletter.

Here you can update your data or remove your subscription.

Salutation *

Mr.
Ms.

Name

First name *

Last name *

E-mail address

E-mail *

Submit subscription

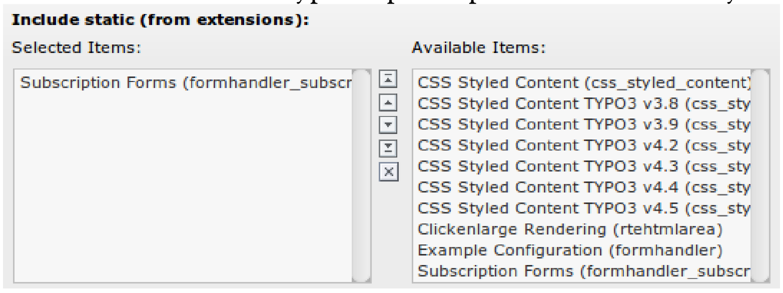
Fields marked with * are required.

Submit

Users manual

The first step, to get this extension up and running is importing it in the extension manager and creating the required database tables. Since it is based on the formhandler extension you will need to import and install it as a requirement.

The second step is to include the static TypoScript setup of the extension in your template:



Include static (from extensions):

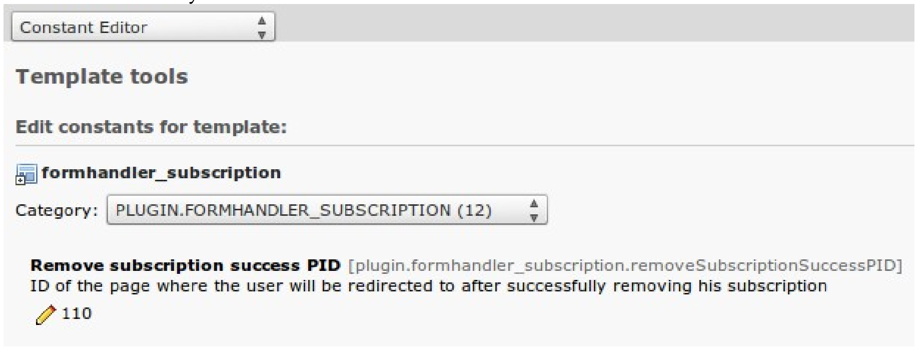
Selected Items:
Subscription Forms (formhandler_subscr

Available Items:
CSS Styled Content (css_styled_content)
CSS Styled Content TYPO3 v3.8 (css_sty
CSS Styled Content TYPO3 v3.9 (css_sty
CSS Styled Content TYPO3 v4.2 (css_sty
CSS Styled Content TYPO3 v4.3 (css_sty
CSS Styled Content TYPO3 v4.4 (css_sty
CSS Styled Content TYPO3 v4.5 (css_sty
Clickenlarge Rendering (rtehtmlarea)
Example Configuration (formhandler)
Subscription Forms (formhandler_subscr

To create the page structure that is needed for the registration process you can import one of the t3d files that you can find in `Resources/T3D/pagestructure-XX.t3d`. At the moment there are only two languages available, english (en) and german (de).

Now you need to create a sysfolder where you would like to store your subscriber records. You can use any table you like. By default the tt_address table is used. If you want to use tt_address records you will need to install the **tt_address** extension.

The last step is setting up the required constants for the extension. If you use the constant editor you should get all information you need.



Constant Editor

Template tools

Edit constants for template:

formhandler_subscription

Category: PLUGIN.FORMHANDLER_SUBSCRIPTION (12)

Remove subscription success PID [plugin.formhandler_subscription.removeSubscriptionSuccessPID]
ID of the page where the user will be redirected to after successfully removing his subscription
110

The most important settings are the IDs of the different pages you created by importing the t3d file:

Constant title	Page title (English version)
Request subscription success PID	Newsletter subcription successful
Confirm subscription PID	Newsletter subcription confirmed
Request update success PID	Request successful
Update subscription form PID	Update subscription
Update subscription success PID	Update successful
Remove subscription success PID	Removal successful
Auth code invalid PID	Invalid URL

That's it. You're good to go :)

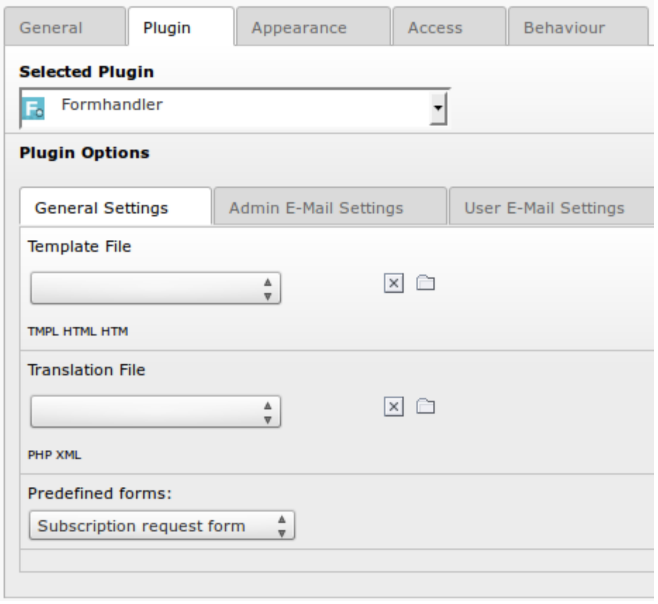
Hint: by default this extension uses the TYPO3 mailing engine, so please make sure your TYPO3 installation is able to send e-mails. You can check this in the TYPO3 Install Tool.

Administration

The administration of this extension basically consists of creating or editing pages and content elements, modifying TypoScript settings and overwriting language labels.

Available Formhander configurations

This is a short overview of the available predefined Formhander configurations:



Selected Plugin
Formhandler

Plugin Options

Template File
TMPL HTML HTM

Translation File
PHP XML

Predefined forms: Subscription request form

Formhandler configuration	Function
Subscription request form	This is the form that new subscribers can use to sign up for the newsletter. After submitting it, they will receive an email that contains a link that can be used to confirm their subscription.
Subscription confirmation	This configuration does not contain a form but only checks for a submitted auth code generated by the “Subscription request form” or the “Subscription update request” configurations. If the submitted auth code is valid, no further action is taken. If it is invalid the user will be redirected to the page configured in the “Auth code invalid PID” constant.
Subscription update request	This form only contains one input field for an e-mail address. If it is submitted, the user will get an e-mail, depending on his subscription state. If he is a new subscriber, he will get an error message. If he is an unconfirmed subscriber, he will get a link for confirming his subscription. If he is a confirmed subscriber, he will get a link for updating his data or unsubscribing from the newsletter.
Subscription update form	This form is only accessible if the user submitted a valid auth code. If the auth code is valid the form will be loaded with the current data of the user. If the auth code is invalid the user will be redirected to the page configured in the “Auth code invalid PID” constant.
Subscription removal	This form only consists of a checkbox. If it is checked and the user submits it his data will be deleted from the database and he will be redirected to the page configured in the “Remove subscription success PID” constant. This form is also only accessible, if a valid auth code was submitted.

Example configuration for direct_mail categories

There are also example configurations available that let the user choose the direct_mail categories. The example adds two checkboxes to the example forms where the user can select if he wants to receive the newsletter in German or English. For this to work you will need to create two direct_mail category records:

- Category “English” with UID 1
- Category “German” with UID 2

You will need to use these form configurations instead the ones described above to let the users choose the direct_mail categories:

Formhandler configuration	Function
Subscription request form - direct_mail categories	This is the same form as described above but it uses a template that contains to checkboxes for the newsletter languages German and English. It also contains a configuration for the UpdateMmData finisher to update the related direct_mail categories.
Subscription update form - direct_mail categories	This is the same form as described above but it uses a template that contains to checkboxes for the newsletter languages German and English. It also contains a configuration for the UpdateMmData finisher to update the related direct_mail categories.
Subscription removal - direct_mail categories	This is the same form as described above but when it is submitted all direct_mail relation records will be removed from the database.

Edit configuration

The extension is fully configured via TypoScript. The configuration options that are available can be found on the Formhandler homepage and in the Configuration section of this document.

Edit language labels

To overwrite or add language labels you will need two things:

1. Create a language file in the TYPO3 Locallang XML format
2. Add this Locallang file to the configuration of formhandler

This is a short example of a Locallang file:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<T3locallang>
      <data type="array">
            <languageKey index="default" type="array">
                  <label index="legend_email">E-mail address</label>
            </languageKey>
      </data>
</T3locallang>
```

You can find the Locallang file that is used for the default configuration in `Resources/Language/Global.xml`.

You need to add you own file (let's assume you put it in `fileadmin/formhandler_subscription_lang.xml`) to the configuration of the form(s) where you want add or overwrite labels:

```
plugin.Tx_Formhandler.settings.predef {
      formhandler_subscription_request_subscription {
            langFile.2 = fileadmin/formhandler_subscription_lang.xm
      }
}
```

If you do not want to use the default Locallang file that comes with this extension, you can use `langFile.1` instead of `langFile.2`. Then the default lang file will not be loaded any more.

This is a short overview of the available TypoScript configuration keys of the different forms:

Formhandler configuration	Configuration key
Subscription request form	`formhandler_subscription_request_subscription`
Subscription confirmation	`formhandler_subscription_confirm_subscription`

Formhandler configuration	Configuration key
Subscription update request	`formhandler_subscription_request_update`
Subscription update form	`formhandler_subscription_update_subscription`
Subscription removal	`formhandler_subscription_remove_subscription`

Configuration

Most of the configuration is done with the default Formhandler components. But there are some additional classes that come with this extension. These will be documented here.

Global options

These options are set in the global section of the form configuration and are not connected to a finisher, pre-processor etc.

Property:	Data type:	Description:	Default:
authCodeDBExpiryTime	string	This setting defines the time when an auth code gets invalid. This string will be parsed through strtotime(). Be aware that an exception will be thrown if a parsing error occurs.	1 day ago
authCodeDBAutoDeleteExpired	string	Before an auth code is read from the database all expired auth codes will be deleted by default. If you do not want this you can set this to 0.	1

Finisher_GenerateAuthCodeDB

This finisher is based on the `Finisher_GenerateAuthCode` of the Formhandler extension. The big difference is: the generated auth code is totally random (not only based on the referenced record data), it is stored in the database (in the `tx_formhandler_subscription_authcodes` table) and it is only valid for one day by default. Additionally the generated code can be used for two purposes: unhiding a record or accessing a form.

These are the settings that are relevant to the finisher:

Property:	Data type:	Description:	Default:
table	string	Required! The table where the referenced record is stored (e.g. tt_address).	
uidField	string	The database field that contains the unique identifier of the referenced record.	uid
action	string	The action that can be triggered when using the auth code. At the moment, two actions are available: • enableRecord • accessForm For more information about what these two actions do have a look at the documentation of `PreProcessor_ValidateAuthCodeDB`.	enableRecord
hiddenField	string	The database field that marks a record as hidden. This is normally configured in: `$GLOBALS['TCA']['tablename']['ctrl']['enablecolumns']['hidden']`	The field configured in the TCA in the `enablecolumns` section in the key `hidden`, if this is not set it defaults to 'hidden'.

Finisher_InvalidateAuthCodeDB

This finisher is very basic. It simply invalidates the submitted auth code by removing it from the database, the session, and the GP array. Additionally, all auth codes referencing the same database record will be removed from the database.

Important! Please make sure that you call `PreProcessor_ValidateAuthCodeDB` before using this finisher. Additionally, the generated auth code has to be generated with the `accessForm` action. Otherwise an exception will be thrown if no auth code was submitted or the submitted auth code was invalid.

Finisher_RemoveAuthCodeRecord

This finisher deletes the referenced record from the database. It has only one configuration option.

Property:	Data type:	Description:	Default:
markAsDeleted	int	If this is set to 1 the referenced record will only be marked as deleted and it will not be removed from the database. The default is to really delete the record so that the user data is protected.	0

Important! Like for the `Finisher_InvalidateAuthCodeDB` you will have to use the `PreProcessor_ValidateAuthCodeDB` and you need to generate the auth code with the `accessForm` action.

Finisher_Subscribe

This finisher does two things.

1. it checks, if the submitted e-mail address (or whatever you want to use as a key, but e-mail probably makes the most sense) already is a subscriber or not, and if the subscriber is confirmed or not
2. According to the result of the check more finishers are called (I call them sub-finishers)

And this is how you can configure it:

Property:	Data type:	Description:	Default:
subscribersTable	string	Required! This is the table where the subscribers are stored (e.g. tt_address)	
setTemplateSuffix	int	If this is true the template suffix will be modified. • for new subscribers it will be set to `_NEW_SUBSCRIBER` • for unconfirmed subscribers it will be set to `_UNCONFIRMED_SUBSCRIBER` • for confirmed subscribers it will be set to `_CONFIRMED_SUBSCRIBER`. Since this is quite useful in most cases (e.g. you want to send different e-mails to the user for these different cases) the default is 1.	1
checkExistenceSelect	->select	This select configuration is used to check if a subscriber exists in the database or not. There is one additional configuration option that is not available in the default select. This option is `showHidden`. If this is true, also hidden records will be retrieved, otherwise only non-hidden records are loaded. See http://wiki.typo3.org/TSref/select	

Property:	Data type:	Description:	Default:
checkConfirmedSelect	->select	This select configuration is used to check if a subscriber is confirmed.	
finishersNewSubscriber	->finishers	These finishers are called, if the subscriber does not yet exist in the database. Please be aware: If you have a finisher that has the option `return` set to 1 you will also need to set `return` to 1 for the subscribe finisher. See http://www.typo3-formhandler.com/documentation/finisher/	
finishersExistingUnconfirmedSubscriber	->finishers	These finishers are called, if the subscriber exists in the database, but has not yet confirmed his subscription.	
finishersExistingConfirmedSubscriber	->finishers	These finishers are called, if the subscriber exists in the database and has confirmed his subscription.	

Finisher_UpdateMmData

This finisher does two things.

This finisher can be used to update data in a many-to-many relationship in a MM table.

There is one limitation: The data you want to insert into the table must not be submitted within an array. So if your prefix for your form fields is myform, the name of your select field or your checkboxes needs to be something like myform[myfield][]. It is not possible to introduce another array like myform[myarray][myfield][].

Property:	Data type:	Description:	Default:
table	string /stdWrap	Required! This is the table where the mm records are stored (e.g. `sys_dmail_ttaddress_category_mm`)	
localUid	int /stdWrap	Required! This is normally the UID of the record that is currently updated. You might want to use the stdWrap data property and set it to `GP:myformprefix|uid`.	
allowedForeignUids	string /stdWrap	Required if `deleteAll` is not set to TRUE! For security reasons you must specify the allowed foreign UIDs that the user can submit. You can disable this check by setting `disableForeignUidCheck` to TRUE.	
foreignUidsField	string /stdWrap	Required if you do not want to remove all records! This is the name of the array field that contains the related records. In the example it needs to be set to `myfield`.	
uidLocalField	string /stdWrap	With this option you can overwrite the name of the database field that contains the local UIDs.	uid_local
uidForeignField	string /stdWrap	With this option you can overwrite the name of the database field that contains the foreign UIDs.	uid_foreign
disableForeignUidCheck	int /stdWrap	Handle with care! If you set this to 1 the finisher will accept all foreign UIDs the user has submitted. This might be a security risk if you use the finisher to set groups that handle access rights.	0
deleteAll	int /stdWrap	Set this to 1 if you want to use the finisher to remove all related records.	0

Finisher_ValidateAuthCodeUID

This finisher checks, if the uid that was submitted by the form matches the uid of the record that was referenced by the currently available auth code.

This can be used to make sure that the person filling out the update form only can update his or her own record but not the record of other people by manipulating a hidden field in the form.

You have to make sure, that the uid that should be validated is submitted in a (hidden) form field. The name of the field either has to have the same name as the uid field that was used to create the auth code (see setting `uidField` of the `Finisher_GenerateAuthCodeDB`) or you need to read the UID value by using the `compareUid` setting.

Important! Like for the Finisher_InvalidateAuthCodeDB you will have to use the `PreProcessor_ValidateAuthCodeDB` and you need to generate the auth code with the `accessForm` action.

Property:	Data type:	Description:	Default:
compareUid	int /stdWrap	This option can be used to specify the UID that will be compared with the the auth code UID. If nothing is specified the finisher will try to read the UID from the GET/POST variable that has the same name as the database field that contains the UID (usually this field is called "uid").	

View_AuthCodeMail

This view is based on Tx_Formhandler_View_Mail. It works basically exactly like it's parent. But there is one little difference. The auth code stored URL stored in the marker `###value_authCodeUrl###` is parsed by htmlspecialchars(). This is quite unhandy when you want to send plain text e-mails (which this extension does by default).

So there is a new marker available that can be used in the plain text mails that has no encoded html characters. This marker is called `###value_authCodeUrlPlain###`.

PreProcessor_ValidateAuthCodeDB

This pre-processor works similar to the Tx_Formhandler_PreProcessor_ValidateAuthCode, but of course there are some differences and enhancements.

First of all, it validates the submitted auth code against the auth codes stored in the database (in the `tx_formhandler_subscription_authcodes` table). Additionally, only auth codes that are not older than one day (this threshold can be configured in the global configuration option `authCodeDBExpiryTime`) are considered to be valid. Last but not least: auth codes that are invalid already are removed from the database. This can be switched off by the global configuration option `authCodeDBAutoDeleteExpired`.

Like is parent class two pages can be configured: one where the user will be redirected to, if the auth code is valid (`redirectPage`), and one where the user will be redirected to if the auth code is invalid (`errorRedirectPage`).

If the auth code is correct the configured action will be executed. After that, the user will be redirected if the pre-processor was configured to do so.

There are two possible actions at the moment: enableRecord and accessForm:

Action "enableRecord"

If this setting is used as action, the referenced record will be unhidden when the PreProcessor_ValidateAuthCodeDB is called and the auth code is submitted. After unhiding the record the auth code will be invalidated immediately.

Action “accessForm”

If this setting is used, PreProcessor_ValidateAuthCodeDB, is called and the auth code is submitted these actions will be triggered:

- the auth code is stored in a session variable so you can link to a different pages without worrying about appending the auth code as a GET parameter
- the auth code data is stored in the GP array (accessible as `$this->gp`)
- additinally, the data of the referenced record can be merged in the GP array if this is configured with `mergeRecordDataToGP`

This information is available in the GP array, after validating the auth code:

- **authCode**: the submitted auth code
- **authCodeData**: the auth code data that was stored in the database, this includes
  - `tstamp`: the time when the auth code was created or when is was used the last time successfully with the accessForm action
  - `reference_table`: the table where the referenced record is stored
  - `reference_table_uid_field`: the uid field of the table where the referenced record is stored
  - `reference_table_uid`: the uid of the referenced record
  - `reference_table_hidden_field`: the hidden field of the referenced record
  - `action`: the action that was authorized by this auth code
  - `auth_code`: the auth code itsself
- **authCodeRecord**: the data of the referenced record that was read from the database

Configuration

These configuration options are available for the `PreProcessor_ValidateAuthCodeDB`.

Property:	Data type:	Description:	Default:
authCodeIsOptional	int	If this is true the auth code will only be validated if the user has submitted one. If no auth code was submitted the user can access the form but no auth code action is executed. The default behaviour is that if no auth code is submitted the user will either get an error message or he will be redirected to the `errorRedirectPage`.	0
mergeRecordDataToGP	int	If this is true the data of the record that is referenced by the auth code will be merged into the GP array accessible as `$this->gp`. This means you can access the data of the referenced record with the normal `###value_[fieldname]###` markers.	0
doNotInvalidateAuthCode	int	By default the auth code is invalidated when the `enableRecord` action is used. This can be prevented by setting this to 1.	0

Information to the additional configuration options can be found in the Formhandler documentation. Options supported by this pre-processor include:

- redirectPage
- errorRedirectPage
- correctRedirectUrl
- additionalParams

Known problems

Please be aware that this is a very young and **not** heavily tested extension. The there might be problems not yet detected. Testing and reporting problems is highly appreciated.

To-Do list

You and find the roadmap in the forge project of this extension.

ChangeLog

You and find the change log in the forge project of this extension.