EXT: Formhandler Extbase

Extension Key: formhandler_extbase

Language: en

Version: 0.0.1

Keywords:

Table of Contents

Introduction

What does it do?

The formhandler_extbase extension gives you the possibility to run Extbase controllers in your preprocessors, interceptors or finishers. Additionally it allows you to use Extbase validators for validating the submitted form values.

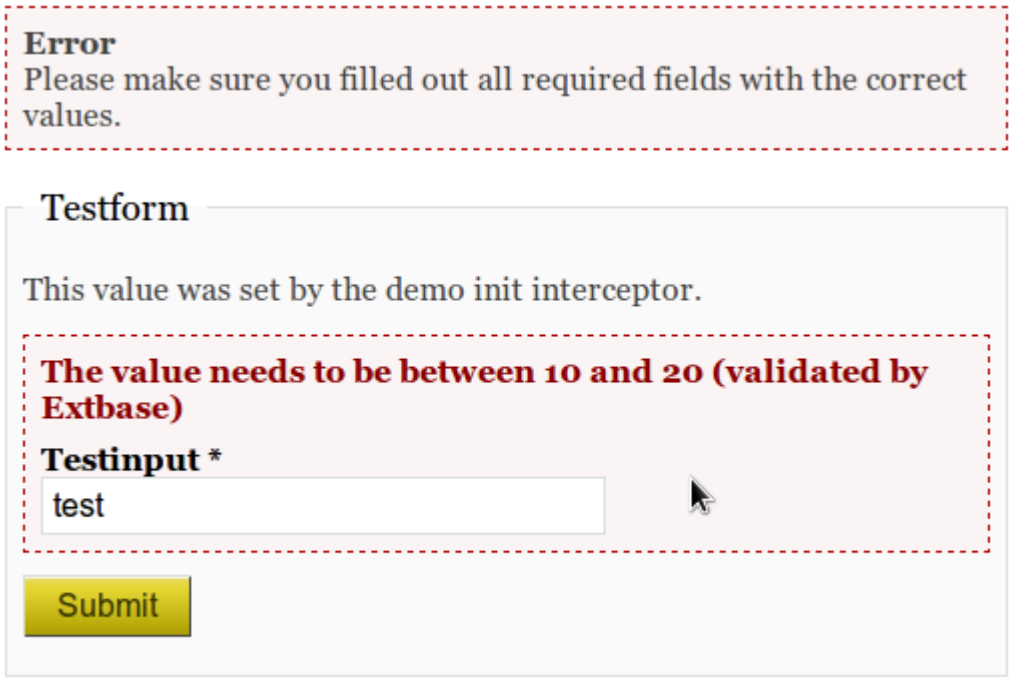
Features and targets of this extention

- give developers an easy to use possibility to use their Extbase functions in Formhanlder
- 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 simple demo form that is included in the extension:



Users manual

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

Now you are already good to go. Here is an example how to use an Extbase interceptor in your form:

```
initInterceptors.1 {

      class = Tx_FormhandlerExtbase_Formhandler_ExtbaseInterceptor
      config {
            extensionName = FormhandlerExtbase
            pluginName = DemoInterceptorInit
      }
}
```

You need to set at least the `extensionName` and the `pluginName` in the configuration.

At the moment is is not possible to influence the called controller or the called action. The first controller action that is configured for the given plugin will be executed.

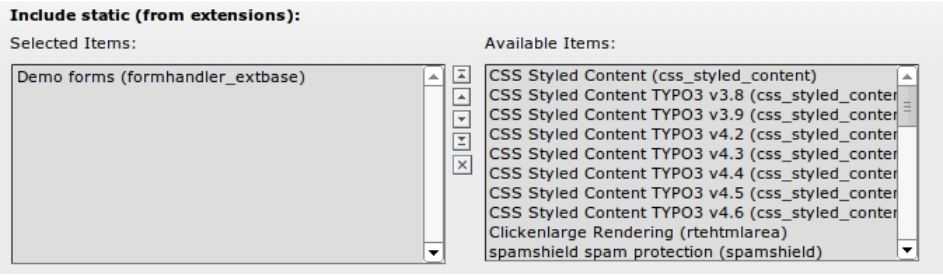
This is an example plugin configuration in your `ext_localconf.php` file:

```
Tx_Extbase_Utility_Extension::configurePlugin(
      $_EXTKEY,
      'DemoInterceptorInit',
      array('Demo' => 'interceptorInit'),
      array('Demo' => 'interceptorInit')
);
```

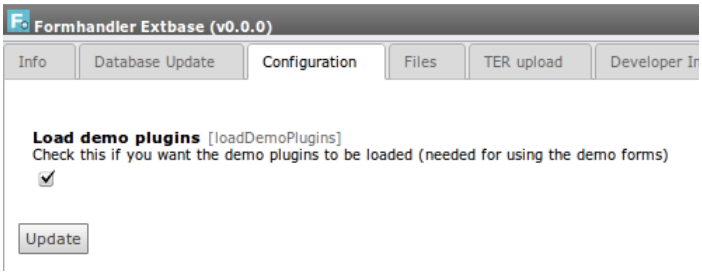
When you now use the `DemoInterceptorInit` plugin the `interceptorInit` action will be called in the `DemoController`.

Using the demo forms

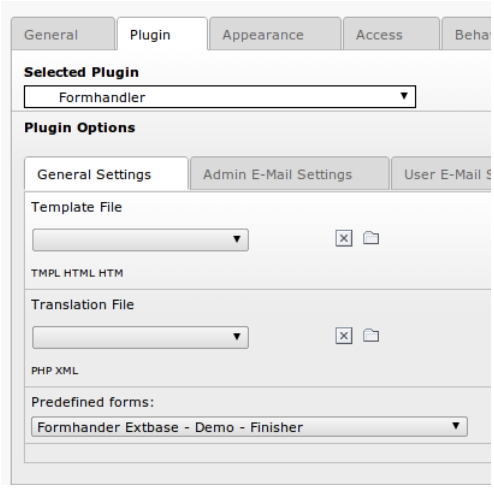
The extension comes with some demo forms. When you want to have a look at them you need to add the static TypoScript configuration for the demo forms:



Additionally you need to enable the demo plugin configuration in the extension settings:



Now you can use the demo forms in the formhandler plugin settings:



You can find the files for the demo forms in the extension folder in these directories:

Files	Directory
HTML template and language file	`Resources/Private/Forms/Demo`
TypoScript configuration for demo forms	`Configuration/TypoScriptDemo`
Configuration for the demo Extbase plugins	`ext_localconf.php`

Administration

The main administration task is to create create formhandler templates and to use the Extbase classes in the TypoScript configuration.

Edit configuration

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

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.

Available classes

You can uses these classes in your form configurations. The sections columns tells you, in which section of the formhandler configuration you can use them.

Class:	Can be used in section:
Tx_FormhandlerExtbase_Formhandler_ExtbaseFinisher	finishers
Tx_FormhandlerExtbase_Formhandler_ExtbaseInterceptor	initInterceptors, saveInterceptors
Tx_FormhandlerExtbase_Formhandler_ExtbasePreProcessor	preProcessors
Tx_FormhandlerExtbase_Formhandler_ExtbaseValidator	validators

Required settings

These settings are required in all classes (except the validator):

Property:	Data type:	Description:	Example:
extensionName	string	This settings tells Extbase in wich extension it should search for the plugin that should be executed.	FormhandlerExtbase
pluginName	string	This setting telle Extbase wich plugin should be called (needs to be configured in `ext_localconf.php`)	DemoPreProcessor

All settings are directly passed on the the Extbase framework which means you can access them in the settings array in your controllers.

Validator settings

The settings for extensionNames and pluginName are set to “FormhandlerExtbase” and “Validator” by default. This will run the validateAction in the ValidationController. This action will use Extbase validators to validate the user input.

You can of course use your own controller for validation, but it might be easier to simply write your own Validator.

To setup validation you can use these settings:

Property:	Data type:	Description:	Default:
field	string	Required! Any erros that will be returned by the configured validator will be attached to this field. If you do not set a `value`, the field name you set here will also be used as the key for getting user input from the GET/POST array of formhandler.	
validator	string	Required! The name of the validator that should be used for validating the input value. You can use the short names for the default Extbase validators (e.g. `NumberRange`) or the full name for your own validators (e.g. `Tx_FormhandlerExtbase_Demo_RequiredNumberValidator`)	

Property:	Data type:	Description:	Default:

validatorOptions	array	Array containing the options that will be passed to the validator, e.g.:
		
		`validatorOptions {`
		`  minimum = 10`
		`  maximum = 20`
		`}`	

value	string /stdWrap	By default the value that is validated is read from GET/POST data array of formhandler. The name configured in `field` is used as array key. With this setting you can overwrite this value with something else.	

Developers

This extension can be used for developers, who develop Extbase based extensions and want to use some of their logic in a formhandler form.

Writing a controller for formhandler_extbase

A controller for formhandler_extbase is not very different from any other controller you develop for Extbase. But there are some things to consider.

Recommended base class

When you write a controller for formhandler_extbase it is recommended to use the `Tx_FormhandlerExtbase_Controller_AbstractActionController` class as your base class.

The first advantage by using this base class is, that you have direct access to the `Tx_FormhandlerExtbase_Mvc_FormhandlerData` object (in `$this->formhandlerData`) wich gives you access to the current GP array of formhandler and lets you update GP values.

The second advantage is, that you can use `$this->disableViewResolution()` for preventing your controller from rendering any output or an error that no template was found for the current action.

Returning and non returning controllers

For every class in formhandler you can set a `return` setting. When this setting is `TRUE` formhandler will stop all further processing and will ouput the content the class returned.

When you write a controller, that manipulates the GP array of formhandler but does not return any content you do not need to use the `return` setting. You simple update the GP value using `$this->formhandlerData->setGpValue()` and use `$this->disableViewResolution()` to prevent the controller from rendering any content.

When you write a controller that should return content, you need to set the `return` setting to `1` in you formhandler class config. You can then use a fluid template for rendering content or simply return a string with the content you want formhandler to display.

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.