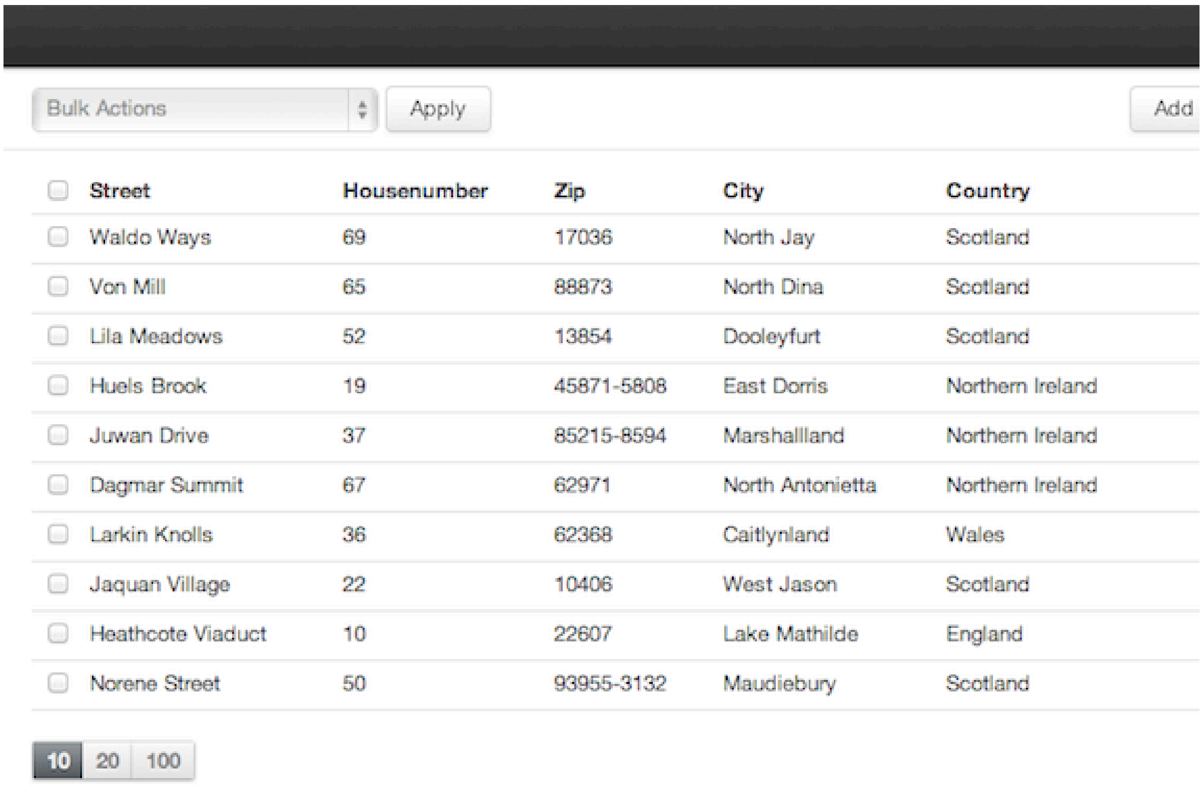
Automatic Administration Interface

Abstract Automatic Administration Interface for FLOW3Marc Neuhaus mneuhaus@famelo.com2010 Marc Neuhaus

Chapter 1: Introduction

This Packages tries to provide a Base to have basic CRUD Operations with a good User Interface and Experience with the least effort and code as Possible, because i believe, that, the DRY principle should be Used for the GUI of your application as much as Possible as Well. In Principle you only need to specify one Tag called @Admin\Annotations\Active in your Model's class and this Admin package will take care of the rest. Every other Option is optional to optimize the Experience of the GUI. For Maximum Flexibility and Versatility a System of Fallback Mechanisms is in place to override the Default Template.



	Street	Housenumber	Zip	City	Country
☐	Waldo Ways	69	17036	North Jay	Scotland
☐	Von Mill	65	88873	North Dina	Scotland
☐	Lila Meadows	52	13854	Dooleyfurt	Scotland
☐	Huels Brook	19	45871-5808	East Dorris	Northern Ireland
☐	Juwan Drive	37	85215-8594	Marshallland	Northern Ireland
☐	Dagmar Summit	67	62971	North Antonietta	Northern Ireland
☐	Larkin Knolls	36	62368	Caitlynland	Wales
☐	Jaquan Village	22	10406	West Jason	Scotland
☐	Heathcote Viaduct	10	22607	Lake Mathilde	England
☐	Norene Street	50	93955-3132	Maudiebury	Scotland

1 Features

- Create and Update Items
  - Appropriate Widgets for Various Data Types
    - String, Integer, Float -> Textfield
    - Boolean -> True/False Selector
    - Single Relation -> Selectbox
    - Multiple Selection -> Special Selectbox with multiple Selection Support
    - DateTime -> Textfield with DatePicker enhanced
    - Ressource -> Upload Input

- Delete Items
- List View
  - Basic List in tabular form
  - Simple Filter Mechanism using a Selectbox
  - Bulk Actions
    - Delete included
- Single View
  - Dummy View to show an Models Data on a seperate Page
- Security
  - User and Roles Management
  - Finely grained Rights Management

Chapter 2: User Manual

1 Basic Usage

There are 2 Ways to Configure the Admin Interface:

1. Settings.yaml
2. Class Reflections inside the Models

Note The Settings.yaml overrules the Class Reflections in order to make it Possible to change the Behaviour of 3rd Party Packages without messing with external Code.

1.1 Settings.yaml

```
PHPCR:
            ░


  Beings:
    F3\Blog\Domain\Model\Post:
      autoadmin: true
      properties:
        content:
          widget: TextArea
```

This Example Activates the Post model of the Blog Example (autoadmin:true) and Changes the Widget for the Content Property from a simple Textfield to a Textarea

1.2 Class Reflections

```
/**
 * A blog post
 * ...
 * @Admin\Annotations\Active
 */
class Post {
    /**
     * @var string
     * @Admin\Annotations\Widget TextArea
     */
    protected $content;
}
```

This Example Does the exact same thing as the Settings.yaml Example but this time inside the Post.php file with the Tag @Admin\Annotations\Active and @Admin\Annotations\Widget TextArea

Chapter 3: Configuration

There are Basically 2 Levels of Configurations

1. Class/Model wide Configuration
2. Property Configuration

1 Model Configuration

1.1 Autoadmin

Enable the Admin Interface for a Model

Class Reflection

```
/**
 * A blog post
 * ...
 * @Admin\Annotations\Active
 */
```

YAML

```
F3\Blog\Domain\Model\Post:
  autoadmin: true
```

1.2 Group

Specifiy a Group in which the Model will be Listed in the Menu. By Default the Models will be Sorted in Categories based on the Package name.

Class Reflection

```
/**
 * A blog post
 * ...
 * @Admin\Annotations\Active
 * @Admin\Annotations\Group Settings
 */
```

YAML

```
F3\Blog\Domain\Model\Post:
  autoadmin: true
  group: Settings
```

Result

Cannot display image: /Users/mneuhaus/Sites/flow3/Packages/Application/Admin/Documentation/Manual/DocBook/en/img/group.png (No such file or directory)

1.3 Set

By Default all Properties will be in a General Fieldset called General in the Order in which they are listed in the Models class. You can override this by specifiying specific Set

Class Reflection

```
/**
 * A blog post
 * ...
 * @Admin\Annotations\Active
 * @Admin\Annotations\Set Main (title,content)
 * @Admin\Annotations\Set Extended Informations (linkTitle,date,author,image)
 */
```

YAML

```
F3\Blog\Domain\Model\Post:
  autoadmin: true
    set:
      - Main (title,content)
      - Extended Informations (linkTitle,date,author,image)
```

Result

Cannot display image:
/Users/mneuhaus/Sites/flow3/Packages/Application/Admin/Documentation/Manual/DocBook/en/img/set.png
(No such file or directory)

2 Property Configuration

2.1 Label

With this Tag you can set the Label for a Property which is by Default a CamelCased version of the propertyname

Class Reflection

```
/**
 * @var string
 * @Admin\Annotations\Label Post Title
 */
protected $title;
```

YAML

```
F3\Blog\Domain\Model\Post:
  properties:
    title:
      label: Post Title
```

Result

Cannot display image:
/Users/mneuhaus/Sites/flow3/Packages/Application/Admin/Documentation/Manual/DocBook/en/img/label.png
(No such file or directory)

2.2 Widget

Instead of the automatically Assigned Widget you can use this Tag to specify a specific Widget.

Class Reflection

```
/**
 * @var string
 * @Admin\Annotations\Widget TextArea
 */
protected $content;
```

YAML

```
F3\Blog\Domain\Model\Post:
  properties:
    content:
      widget: TextArea
```

Result

Cannot display image:
/Users/mneuhaus/Sites/flow3/Packages/Application/Admin/Documentation/Manual/DocBook/en/img/widget-content-textare
(No such file or directory)

2.3 Ignore

There are a number of Properties which have no use to be Administrated through a GUI. With the ignore Tag you can control the Visibility of the Property to the Admin Interface.

2.3.1 Ignore the Property Completely

Class Reflection

```
/**
 * @var string
 * @Admin\Annotations\Ignore
 */
protected $id;
```

YAML

```
F3\Blog\Domain\Model\Post:
  properties:
    id:
      ignore: true
```

2.3.2 Ignore the Property in specific Views

In some cases you may just want to Ignore a Property in the List View because it just clutters the View

Class Reflection

```
/**
 * @var string
 * @Admin\Annotations\Ignore list,view
 */
protected $content;
```

YAML

```
F3\Blog\Domain\Model\Post:
  properties:
    content:
      ignore: list,view
```

This will Ignore the Property Content in the List and Single View

2.4 Infotext

For more Information about a Property aside from the Title you can provide an Infotext that will be shown beside/below the Input Widgets in the Create and Update Views

Class Reflection

```
/**
 * @var string
 * @Admin\Annotations\InfoText Please tell us who you are
 */
protexted $author;
```

YAML

```
F3\Blog\Domain\Model\Post:
  properties:
    author:
      infotext: Please tell us who you are
```

Result

Cannot display image:
/Users/mneuhaus/Sites/flow3/Packages/Application/Admin/Documentation/Manual/DocBook/en/img/infotext-author.pn
(No such file or directory)

2.5 Class

Adds one or more Classes to the Input Widget

Class Reflection

```
/**
 * @string
 * @Admin\Annotations\Class content
 */
protected $content;
```

YAML

```
F3\Blog\Domain\Modle\Post:
  properties:
    content:
      class: content
```

2.6 Validate

Please Check the FLOW3 Documentation for the Validation rule:

http://flow3.typo3.org/documentation/tutorials/getting-started/gettingstarted.validation/

2.7 Title

You can Specify any Property that can be Converted to a String as a Title to be used as a simple String Repesentation which is for example used in the Single and Multiple Relation Widget to Identify an Model Item

Class Reflection

```
/**
 * @var string
 * @title
 */
protected $title;
```

YAML

```
F3\Blog\Domain\Model\Post:
  properties:
    title:
      title: true
```

Note By Default the Admin Interface tries to Resolve an appropriate Title for the Model automatically. To do this the Following things will be tried:

1. Does the Model Provide a __toString() Method
2. Does one or more @title Tags exist
3. Are there Properties Tagged as @identity which can be converted to String
4. Is there a Property with the name "title" or "name"

First thing that matches will be used in the Order specified

2.8 Filter

By Tagging a Property with this Tag the Admin Interface will try to provide a Selectbox to Filter the List View by the Possible Values of this Property

Class Reflection

```
/**
 * @var string
 * @filter
 */
protected $author;
```

YAML

```
F3\Blog\Domain\Model\Blog:
  properties:
    author:
      filter: true
```

2.9 OptionsProvider

The OptionsProvider takes care of the Preperation to create an Optionslist for the SingleRelation and MultipleRelation Widget.

```
/**
 * @var \SplObjectStorage<\F3\Admin\Security\Policy>
 * @Admin\Annotations\OptionsProvider \F3\Admin\OptionsProvider\PolicyOptionsProvider
 **/
protected $grant;
```

Chapter 4: Templates

Since the Admin Interface would be not of much use if you could only use the Templates that the Admin Package ships with by default there is a Fallback System in place to automatically choose the most specific Template possible to Render the Admin Interface. The Fallbacks are Configured in the Settings.yaml and can be customized if needed.

1 Views

The Admin Interface will check for the existence of each of this Fallbacks until a Template is found and then Render accordingly:

```
- resource://@package/Private/Templates/@being/@action/@variant.html
- resource://@package/Private/Templates/Admin/@action/@variant.html
- resource://@package/Private/Templates/@being/@action.html
- resource://@package/Private/Templates/Admin/@action.html
- resource://Admin/Private/Templates/Standard/@action/@variant.html
- resource://Admin/Private/Templates/Standard/@action.html
```

Note

These Variables are available

Placeholder	Replacement
@package	Name of the Package which Contains the Model to be Rendered
@being	Short Name of the Model (F3\Blog\Domain\Model\Post -> Post)
@action	Action to Render (List,Create,Update,Confirm,View)

@variant	Variant of the Action (Tabular, Block)

2 Partials

Partials are Subparts which can be Reused in more than one View (Form, Table, Toolbar,...)

```
- resource://@package/Private/Partials/@being/@action/@partial.html
- resource://@package/Private/Partials/@being/@partial.html
- resource://@package/Private/Partials/@action/@partial.html
- resource://@package/Private/Partials/@partial.html
- resource://Admin/Private/Partials/@action/@partial.html
- resource://Admin/Private/Partials/@partial.html
```

Note *These Variables are available*

Placeholder	Replacement
@package	Name of the Package which Contains the Model to be Rendered
@being	Short Name of the Model (F3\Blog\Domain\Model\Post -> Post)
@partial	Name of the Partial (Form, Table, Toolbar,...)

3 Widgets

```
- resource://@package/Private/Partials/@being/Widgets/@partial.html
- resource://@package/Private/Partials/Widgets/@partial.html
- resource://Admin/Private/Partials/Widgets/@partial.html
```

Note *These Variables are available*

Placeholder	Replacement
@package	Name of the Package which Contains the Model to be Rendered
@being	Short Name of the Model (F3\Blog\Domain\Model\Post -> Post)
@partial	Name of the Partial (TextField, Boolean, DateTime,...)

Chapter 5: Widgets

1 Available Properties in a Widget Template

Inside the Widget Partial there is one essential Variable available called {property} This Variable Provides the

Property	
{property.adapter}	Classname of the Current Adapter
{property.widget}	Name of the Widget
{property.value}	Unprocessed Value of the Property, this might be almost Anything depending on the Data in the Object. Handle this with care, because it might

	cause Rendering errors. In most cases you should simply use the {property.string} option to get an String representation of the Value
{property.infotext}	Informational Text for the Property
{property.label}	label for the Property
{property.string}	String representation of the Propertyvalue
{property.inputName}	Appropriate name for an input name for the Property including prefix (item[propertyname]
{property.type}	DataType of the Property
{property.name	Name of the Property

2 Replace the Default Widget for a Specific Data Type

Widgets are assigned to DataTypes by a Fallback System configured in the Settings.yaml

```
PHPCR:
  Widgets:
    Mapping:
      # Pattern (Text or Regex)          Widget

      string:                            Textfield
      readonly:                          TextfieldReadonly
      integer:                           Spinner
      float:                             Textfield
      boolean:                           Boolean
      \F3\FLOW3\Resource\Resource:       Upload
      \DateTime:                         DateTime
      ^\\*F3\\[A-Za-z]+\\Domain\\Model\\[A-Za-z]+$: SingleRelation
      ^\\*\SplObjectStorage\<\\*F3\\[A-Za-z]+\\Domain\\Model\\[A-za-z]+>$: MultipleRelati
      ^\\*\SplObjectStorage\<\\*F3\\[A-Za-z]+\\Security\\[A-za-z]+>$: MultipleRelation
```

On The Left side you have your Classes or Names of the DataTypes and on the Right Side is the repsonsible Widget to use. You can override any of these Widgets in your Production/Development Settings.yaml. Aside from Classes or DataType Names you can specify an Regular Expression to Match more Complex things, like in this Case Entity Models.

Chapter 6: Bulk Actions

Bulk Actions are available in the List View to Change more than one Item at a time. The Admin Interface provides an Action to Delete multiple Items in the List View.

1 Creating a Simple Bulk Action

2 Best Practises

Chapter 7: Options Providers

An Options Provider creates the List of Options for the SingleRelation and MultipleRelation Widgets. Currently there is just the Default implementation which creates the Options Using the ID and String Representation of the Object. But for Example the PolicyOptionsProvider ensures that there are all needed Options as Policy available when the Roles Object is loaded

Chapter 8: Object Converters

Object Converters take care of the Conversion of a Value from String and to String

Chapter 9: Adapters

The Admin Interface is designed quite Modular to enable the Addition of other Storage Engines beside PHPCR. This Chapter gives an Overview of what needs to be Implemented to Create a new Adapter