EXT: Justimmo REST API

Extension Key: justimmo
Language: en
Version: 1.0.2
Keywords: forEditors, forAdmins, forIntermediates, forAdvanced
Copyright 2012, Thomas Juhnke, <tommy@van-tomas.de>

This document is published under the Open Content License
available from http://www.opencontent.org/opl.shtml

The content of this document is related to TYPO3
- a GNU/GPL CMS/Framework available from www.typo3.org

Table of Contents

EXT: Justimmo REST API....1
Introduction....3
What does it do?....3
Screenshots....3
Users manual....5
TypoScript setup....5
FAQ....6
Administration....7
Server requirements....7
Configuration....7
realurl / Speaking URLs....8
Configuration....10
Reference....10
Tutorial....13
Tutorial 1 – Adding lightbox capabilties to the detail view....13
Known problems....17
Extension of filter criteria....17
Speed/performance....17
To-Do list....18
ChangeLog....19

Introduction

What does it do?

This plugin provides an interface to the justimmo.at REST API for querying realty data. It ships with 3 plugins for different search requirements (realty number, quick & detail search) and a list & detail view for the search results. You need to acquire an API authentication in order to use this extension.

Screenshots

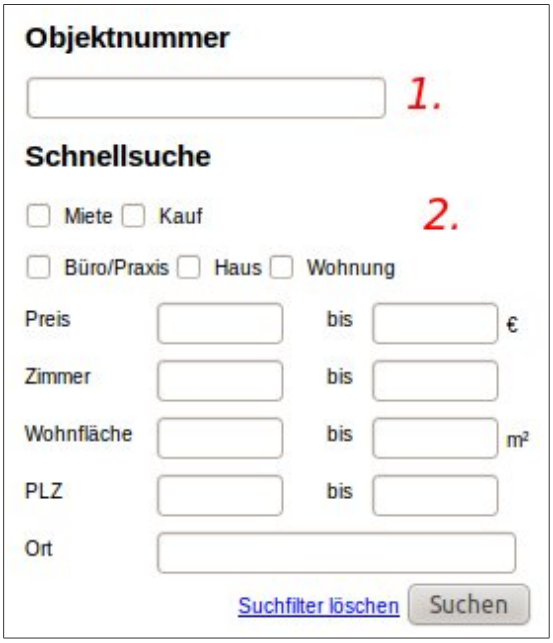

Fig. 1: realty number & quick search

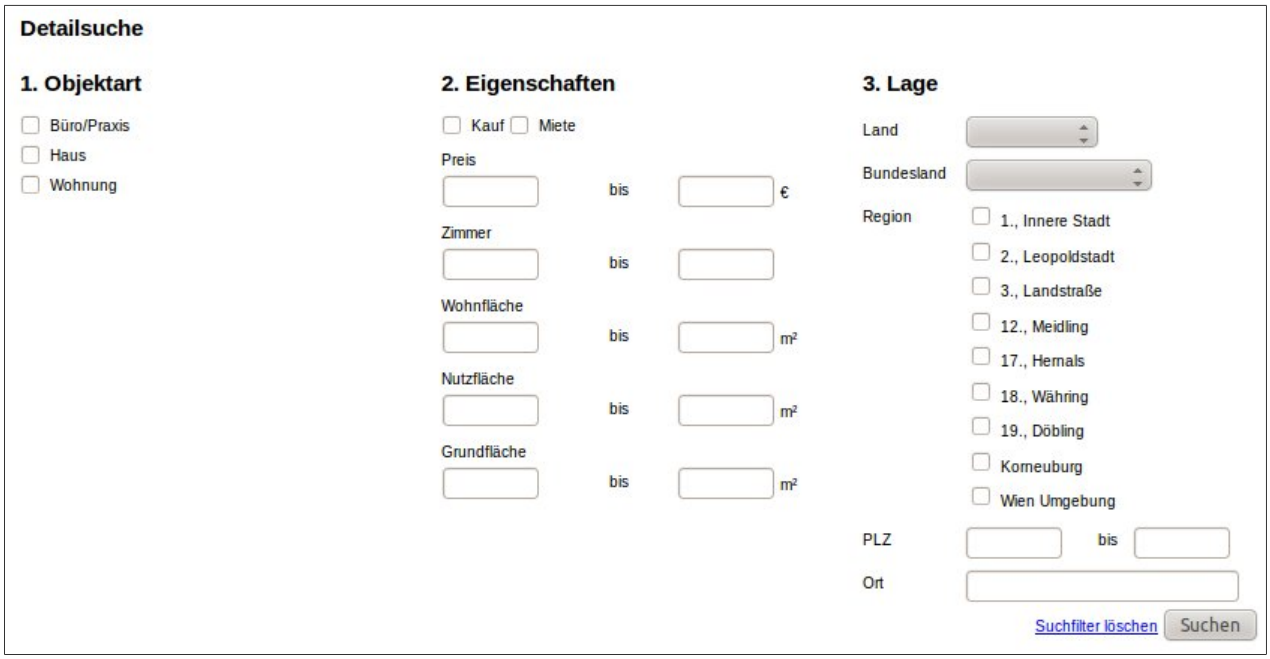

Fig. 2: detail search

Fig. 3: search/filter results view

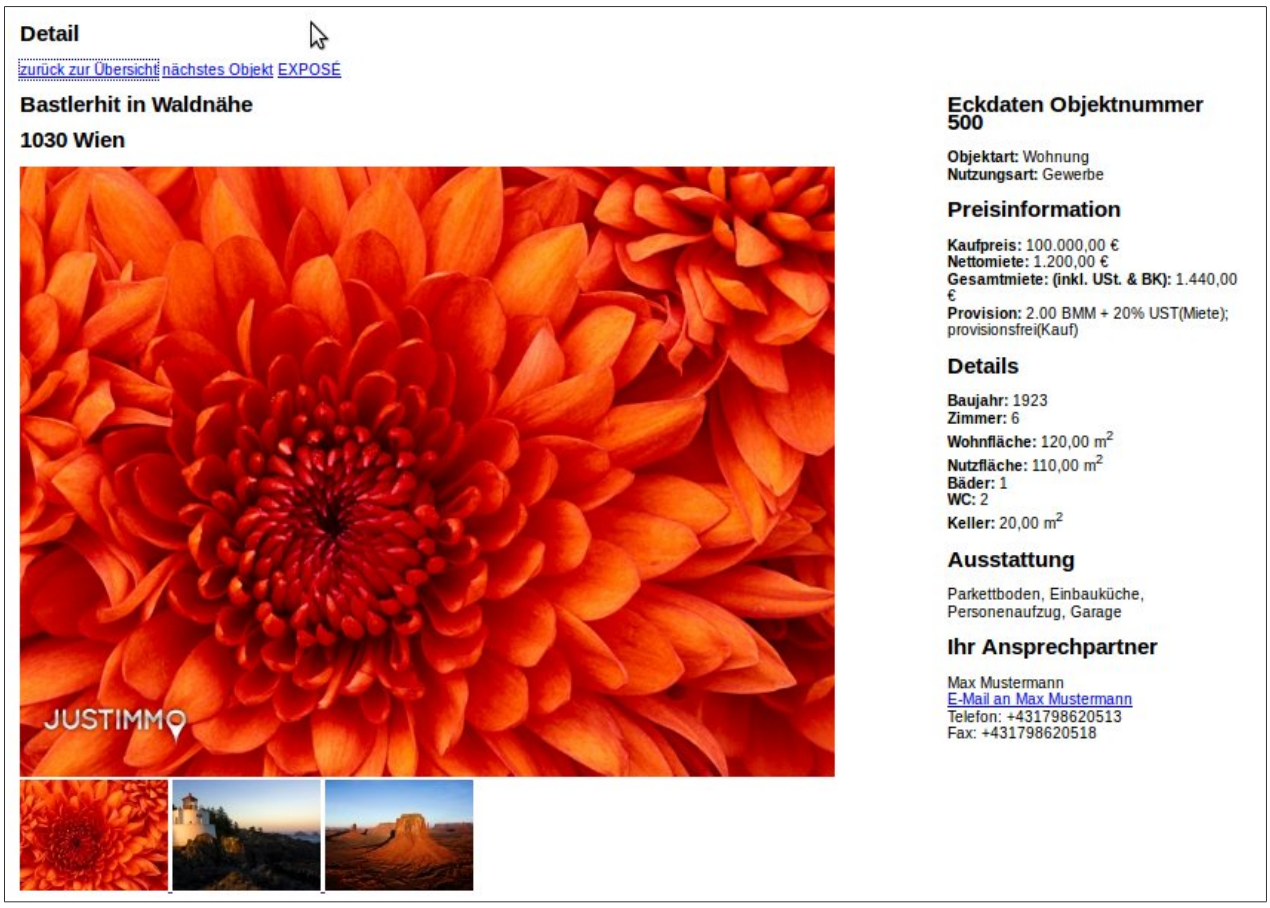

Fig. 4: realty detail view

Users manual

Download & install the extension with the extension manager. After installation you must integrate the shipped plugins into different parts of your website. You must not create any data records or anything, but ask your administrator to setup the extension properly by specifying the API authentication parameters username & password in the TypoScript setup in an extension TypoScript template.

At first you should sit down and think about what search plugins do you need & where to integrate them. For example the realty number & quick search plugins are designed to fit perfectly into a tight sidebar or a footer.

You must create the page structure for the search results plugin and the detail view plugin. After this, it's also possible to create dedicated pages for “direct link” searches. But therefore, you have to meet with your TYPO3 integrator and define which filter parameters have to be set for these pages. The configuration for the “direct link” search pages must happen in dedicated TypoScript extension templates which are bound to each of this pages. A search results view plugin must be inserted into this pages to show the resulting list of realty objects.

In the following screenshot you see an example page structure setup:

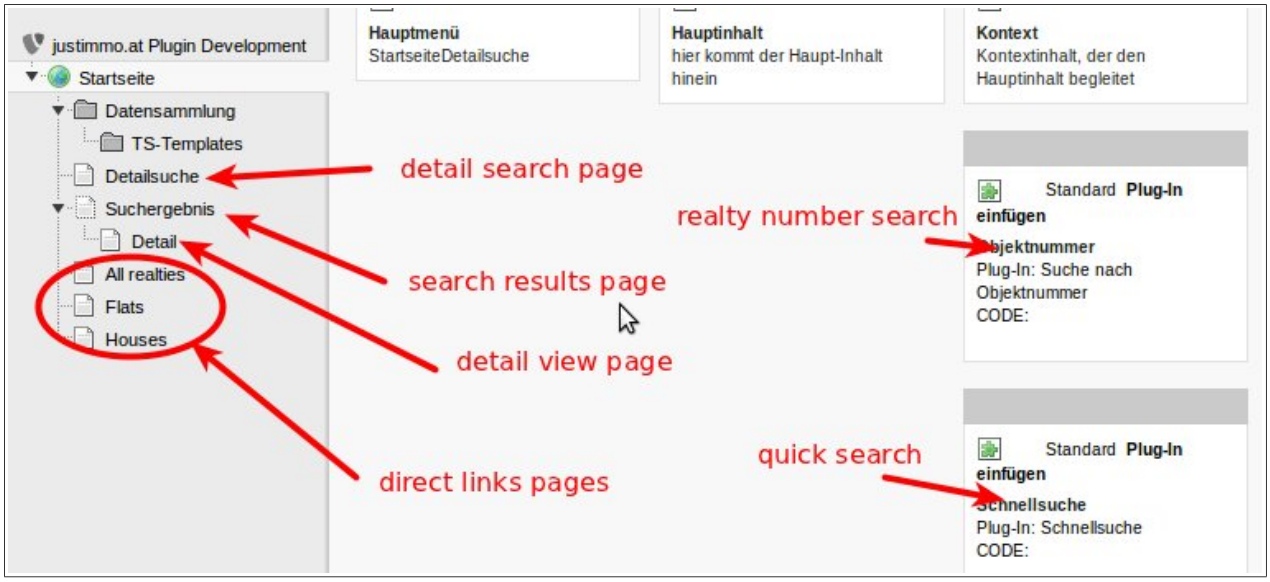

Fig. 5: example page structure setup

TypoScript setup

After the basic content element/plugin insertion setup the administrator/integrator of the TYPO3 website must create an extension TypoScript template and configure the justimmo extension.

It is important to define the following constants with the help of the constant editor:

```
plugin.tx_justimmo.settings.api.username = [API_USERNAME]
plugin.tx_justimmo.settings.api.password = [API_PASSWORD]
plugin.tx_justimmo.settings.searchResultsPid = 4
plugin.tx_justimmo.settings.realtyDetailPid = 6
```

As you see, you have to define the API username & password but also the page IDs of the search results (4, Fig. 5: “Suchergebnis”) and realty detail view (6, Fig. 5: “Detail”) pages.

After that, he has to create TypoScript templates for all of the direct links pages and define the filter defaults in the setup key

```
plugin.tx_justimmo.settings.realtyRepository.defaults.filter
```

FAQ

Q: Is it possible to define the kind of filters which are visible in the quick/detail search forms?

A: The configuration is done directly in the filter template. See the administration section for further information.

Q: Which kind of filters are available at all for the search forms?

A: Please see the justimmo.at API documentation.

Administration

First, you need to include the static extension template to your template root as usual for such kind of extensions. Furthermore you have to ensure, that a backend user is able to insert the different plugins of the justimmo extension. On the other side, you can keep the whole page/plugin structure hidden for the user but then you have to define where the plugins and which of them must be integrated into the website.

The other tasks you have to perform are

1. TypoScript setup/configuration of the extension
2. adjustments of the extension HTML/Fluid templates to match the filter ability requirements

The latter task requires some Fluid knowledge as all templates are built with this modern TYPO3 template technique. Futhermore, you need to adjust the TypoScript configuration a little further. Please see the Configuration chapter for more information.

Server requirements

The extension makes usage of PHP's *curl_** functions, so it is necessary that you've installed the curl library and PHP modules on the server side.

Configuration

The basic setup can be performed with the constants editor. Open it now and switch to the category “PLUGIN.TX_JUSTIMMO” to see what options are available.

As you can see in the following screenshot, the most important settings are already set. Namely: “API username”, “API password”, “Realty detail PID” & “Search results PID”.

The setting “Path to template root (FE)” is interesting if you want to adjust the plugin templates according to your needs or the requirements of the project. Change the path value to the path your new template resides (e.g. fileadmin/templates/domain.tld/ext/justimmo/Private/Templates/). Ensure you copy all of the controller related subdirectories from the extension's template directory to this new directory.

Delete the cache and check if the extension still works. If everything is OK, start modifying the desired templates.

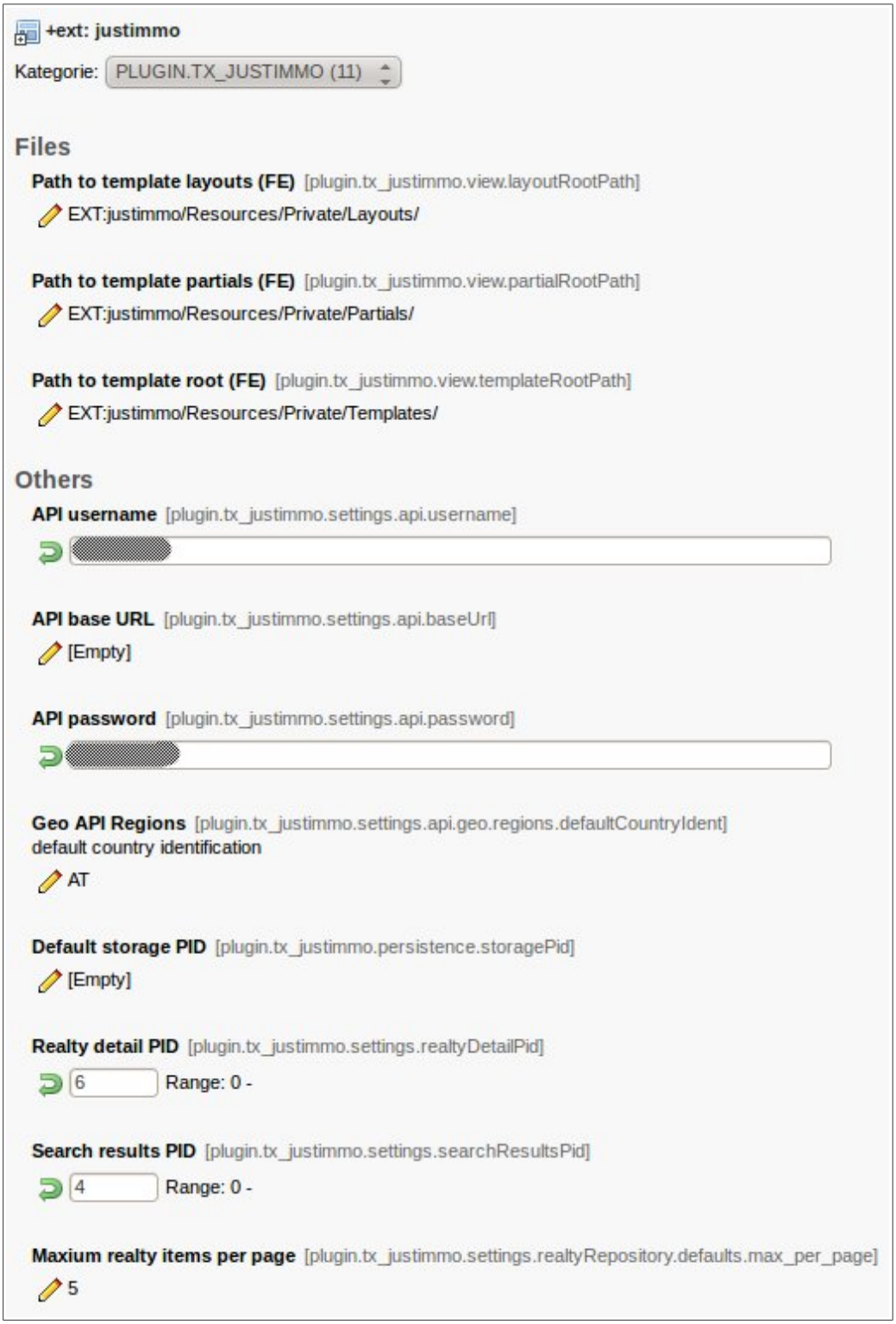

Fig. 6: constant editor settings

realurl / Speaking URLs

This extension ships with a realurl auto configuration, ready for simple drop in to your existing site.

As this extension was developed mainly for the german market, the URL path segments are defined in german language. If you develop a german website, simply activate the automatic configuration of realurl and you're done.

If you're integrating a site for another target language, please study the auto configuration and adapt it to your needs. The default configuration can be found in the file EXT:justimmo/Classes/Utility/RealurlAutoconfiguration.php. Please read the realurl manual for further information about the individual configuration keys.

Configuration

The whole configuration is done in TypoScript templates. No Page/User-TSConfig nor extension configuration.

Reference

Constants: template settings

Please ensure, that you copy the whole subdirectories and all of the files of the adjusted view property path to the new directory!

Property:	Data type:	Description:	Default:
layoutRootPath	string	layoutRootPath of the Fluid templates	EXT:justimmo/Resources/Private/Layouts/
partialRootPath	string	partialRootPath of the Fluid templates	EXT:justimmo/Resources/Private/Partials/
templateRootPath	string	templateRootPath of the Fluid templates	EXT:justimmo/Resources/Private/Templates/

[tsref:plugin.tx_justimmo.view]

Constants: common settings

Property:	Data type:	Description:	Default:
realtyDetailPid	int+	page ID where the realty detail view plugin is integrated **This is a required setting!**	
searchResultsPid	int+	page ID where the realty detail view plugin is integrated **This is a required setting!**	
pager.maxBefore	int+	amount of pages shown in pager before the current page	3
pager.maxAfter	int+	amount of pages shown in pager after the current page	3
pageTitle.enable	boolean	flags if the realty detail view plugin should set the page title	1
pageTitle.mode	keyword	select how to set the page title before – realty object title is set before current page title after – realty object title is set after current page title override – realty object title overrides the current page title	before
pageTitle.usePageRenderer	boolean	flags if the page renderer of TYPO3 should be used to set the title. **Note**: this is known to be not working currently, but is set for future improvements of the extension.	0

[tsref:plugin.tx_justimmo.settings]

Constants: API settings

Property:	Data type:	Description:	Default:
username	string	API authentification username **This is a required setting!**	
password	string	API authentification password **This is a required setting!**	
geo.regions.defaultCountryIdent	string/int+	the default country identification ISO2 or ID for the initial region list in the search views.	AT

[tsref:plugin.tx_justimmo.settings.api]

Constants: realty repository settings

Property:	Data type:	Description:	Default:
defaults.max_per_page	int+	defines the maximum realty objects per page in search results	5
resetFilterOnInit	boolean	set this setting to 1/TRUE if you need to reset the filter on specific pages (e.g. “direct links” pages)	0

[tsref:plugin.tx_justimmo.settings.realtyRepository]

TypoScript: realty filter defaults

These settings are needed if you plan to create direct link search pages. Integrate the justimmo search results list view plugin on that page and set up your default filters in a TypoScript template on that page.

Property:	Data type:	Description:	Default:
[valid filter criteria]	scalar, mixed	Please read the justimmo.at API reference for the available filter criteria. Bear in mind, that TypoScript is a configuration *array*, so you're able to setup an array filter criteria by simply opening a new configuration level.	

[tsref:plugin.tx_justimmo.settings.realtyRepository.defaults.filter]

Example

```
plugin.tx_justimmo.settings.realtyRepository.defaults.filter {
      // rentable objects
      miete = 1
      // objektart_id is an array criteria, new configuration level needed;
      // key is simply the array index, should start at 1 or in increments of 10
      objektart_id {
            // flats/appartements
            1 = 2
            // houses
            2 = 3
      }
}
```

TypoScript: style defaults

The extension comes with some default styles for the different plugins. Here's a list of keys which are defined by default. Feel free to override or delete them if necessary.

Property:	Data type:	Description:	Default:
tx_justimmo_bootstrap	fileResource	basic styles of Twitter's bootstrap for grid elements.	EXT:justimmo/Resources/Public/Css/bootstrap.min.css
tx_justimmo_default	fileResource	default styles for all plugins	EXT:justimmo/Resources/Public/Css/default.css
tx_justimmo_list	fileResource	default styles for search results list view plugin	EXT:justimmo/Resources/Public/Css/list.css
tx_justimmo_detail	fileResource	default styles for detail view plugin	EXT:justimmo/Resources/Public/Css/detail.css
tx_justimmo_search_quick	fileResource	default styles for quick search plugin	EXT:justimmo/Resources/Public/Css/search_quick.css

Property:	Data type:	Description:	Default:
tx_justimmo_search_detail	fileResource	default styles for detail search plugin	EXT:justimmo/Resources/Public/Css/search_detail.css

[tsref:page.includeCSS]

HTML/Fluid templating: define filter criteria

To stick with the KISS principle I've decided to keep the filter criteria configuration in the template, where the fields must be defined anyway.

Scenario 1

Let's say you want to extend the realty object type ("*objektart_id*") filter criteria which is probably one of the most changed filter criteria for querying the justimmo API. Open your favourite IDE or editor and open the file EXT:justimmo/Resources/Private/Templates/Search/Detail.html.

Search for *objekartId* and you will find some predefined filter criteria for this property. Now, let's say there is an object type with the ID 1. So the simple approach is to insert a new Fluid view helper to generate the checkbox:

```
<f:form.checkbox property="objektartId" value="1" id="detail_objektart_id_1" />
```

You see: it's that simple to add a new object type filter criteria to the template! There are a two basic rules, which will be explained here:

1. Every property must be converted from *underscored_lower_case* to *lowerCamelCase*
2. Write the property value into the *value* attribute of the view helper.

Scenario 2

Now, some time has passed and the justimmo API gets a new filter criteria. In this case, the extension of the template is not that easy. Because we're in an extbase context with the whole extension, the steps are a bit more complex:

1. Update the domain model object *Filter*: add the property, setter & getter methods
2. Update the domain model object validator *FilterValidator*: read the isValid() method and add the filter property validation accordingly.
3. After these steps, you should clear the cache to update the reflection cache of extbase
4. Add the new filter criteria to one of the forms as described in scenario 1

Tutorial

Tutorial 1 – Adding lightbox capabilties to the detail view

This tutorial should help you understand the basics regarding Fluid template adjustments of this extension. You need a basic understanding and some knowledge in how to install TYPO3 extensions and how to work with the Fluid templating system.

Note: wherever pointed to “example.org” or “DOMAIN.TLD” in the following guideline, please replace this with your actual domain name or any other string/identifier.

Step 1: think about and install necessary extensions

At first, you should sit down and think about what extension from TER you want to integrate into your running TYPO3 environment. Please check and double-check the installed extensions, maybe you already downloaded and installed one? In this case read the manual of this extension and study its TypoScript setup what kind of adjustments it does to enable lightbox functionalty in the frontend.

In this tutorial, I'd like to cover one of my favourite extensions: sk_fancybox. It works with the famous jQuery framework and integrates well into TYPO3 by adding some nifty TypoScript setup for changing the rendering of image content objects with activated “click enlarge”.

Step 2: download and install sk_fancybox

Download and install sk_fancybox with the extension manager. During the installation you can set some system-wide settings. For example, you need to set “dontIncludeJquery” if you already installed jQuery to your TYPO3 powered website.

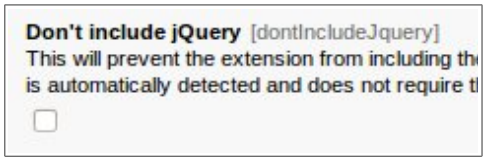

Fig. 7: check this checkbox to disable jQuery

Also, you have the possibility to exclude fancybox itself from the page rendering, if you uploaded your own (probably modified version) of this great JavaScript plugin. In this case the extension only adds the necessary TypoScript setup and fancybox configuration.

Fig. 8: check this checkbox to disable fancybox.js inclusion

After that process, you need to add the static extension TypoScript setup to your root TS template. Go to “Web → Template” and select your root page. Switch to “Info/Edit” and click on “edit the whole template record”. Switch to the third tab and select the static extension template from the list on the right. It gets added automatically to the left list.

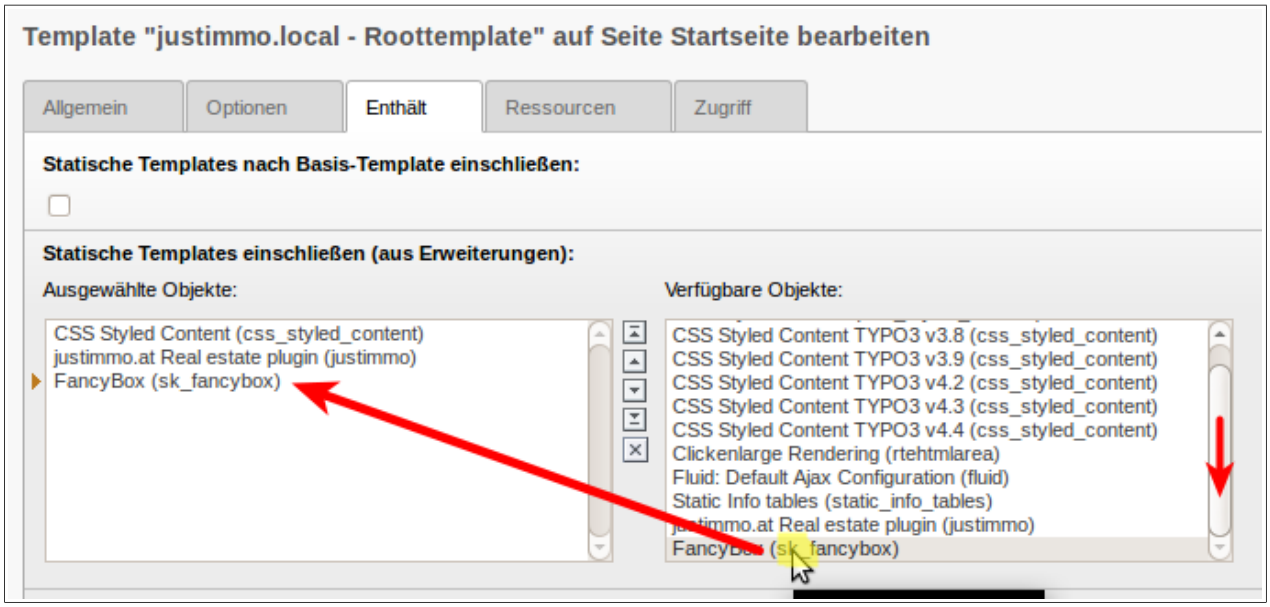

Fig. 9: how to add static extension template of sk_fancybox

Save the root TypoScript template by clicking the little disk icon on the upper right corner of the editing pane. The lightbox extension setup is complete now. Let's proceed with the adjustments on the ext:justimmo templates. Please read the extension manual of sk_fancybox if you want to know more or need help regarding this extension.

Step 3: ext:justimmo template adjustments

To have a clean HTML/Fluid template structure, I prefer to create all website specific templates under the directory path fileadmin/templates/DOMAIN.TLD/. Create this folder, if it doesn't exist yet and also create the following subfolders in it: ext/justimmo/Partials/. The resulting directory tree should look like the following:

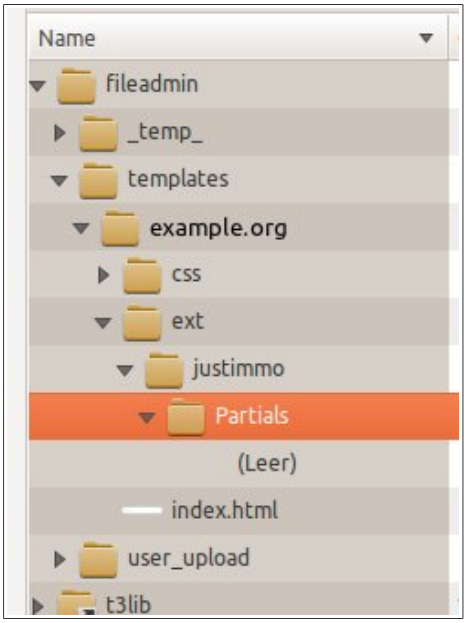

Fig. 10: example template directory structure

Now copy the default contents of the extension HTML/Fluid folders and files from typo3conf/ext/justimmo/Resources/Private/Partials/ to your template path fileadmin/templates/DOMAIN.TLD/ext/justimmo/Partials/.

The result should look like this:

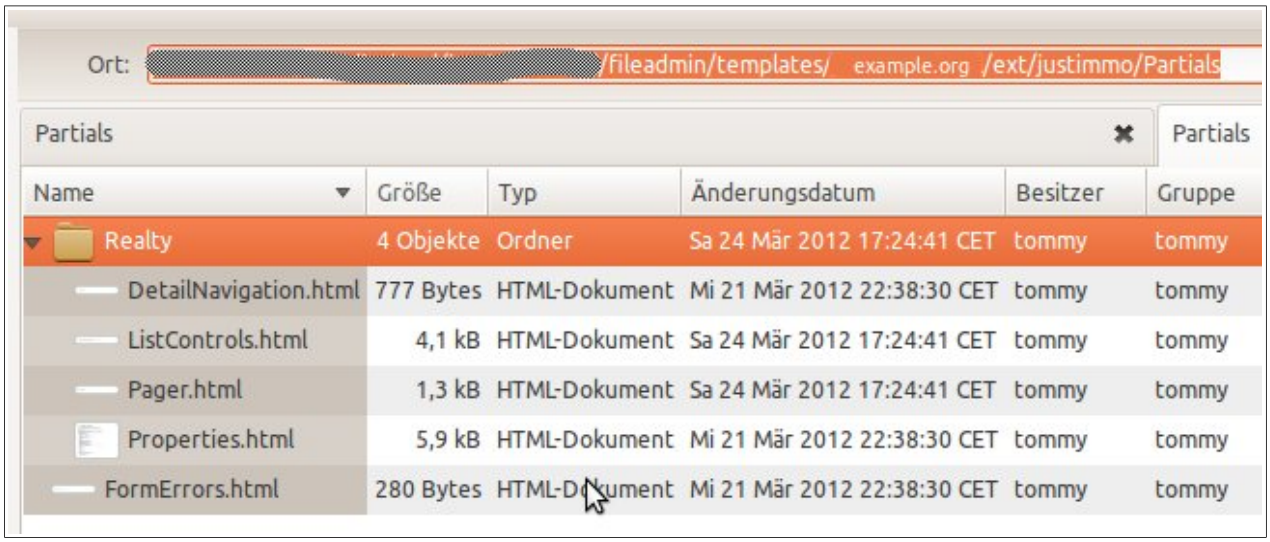

Fig. 11: copied templates in your local template path

You may ask: “Why do I have to copy all that stuff? I just want to adjust the detail view for a realty object!”, but as this documentation states: the template path of an extbase/Fluid extension can only be adjusted on the whole. So if you want to adjust one template, you need to copy all templates of the extension, because otherwise the templating system wouldn't find the original template files from the extension directory. The same is true for partial or layout adjustments of an extbase/Fluid based TYPO3 extension.

Step 4: get your hands dirty: the lightbox adjustments

Now open up your favourite IDE or text editor and load the file fileadmin/templates/DOMAIN.TLD/ext/justimmo/Partials/Realty/Properties.html.

On line 10, wrap the image into the following anchor tag:

```
<a href="{realty.anhaenge.0.daten.pfad}" class="fancybox"
rel="fancybox{realty.id}">[original markup goes here]</a>
```

As you can see, we adding a <a>-tag pointing it's href-attribute to the image src, adding class=”fancybox” and rel=”fancybox{realty.id}”. The rel-attribute is able to group the images, so be sure to add a proper identificator (the realty id in this case) to the end of the attribute value.

On line 15, there is already a link to the original image source. All we need to do is adjust the <a>-tag by adding the classname “fancybox” and also setting the rel-attribute. The line has to look like this:

```
<a class="tx_justimmo_detail_image_thumbnail_item fancybox" rel="fancybox{realty.id}"
href="{anhang.daten.pfad}">
```

That's it! We're done with editing the detail view template. Now make ext:justimmo know where to fetch the new template.

Step 5: final polish – TypoScript setup & clear your caches

Open your root page TypoScript template and add the following line to the constants field:

```
plugin.tx_justimmo.view.partialRootPath =
fileadmin/templates/DOMAIN.TLD/ext/justimmo/Partials/
```

Clear all caches by clicking the yellow flash icon and selecting the first submenu item in the upper right corner of the TYPO3 backend pane.

Refresh the browser and view the result by clicking one of the images in the detail view:

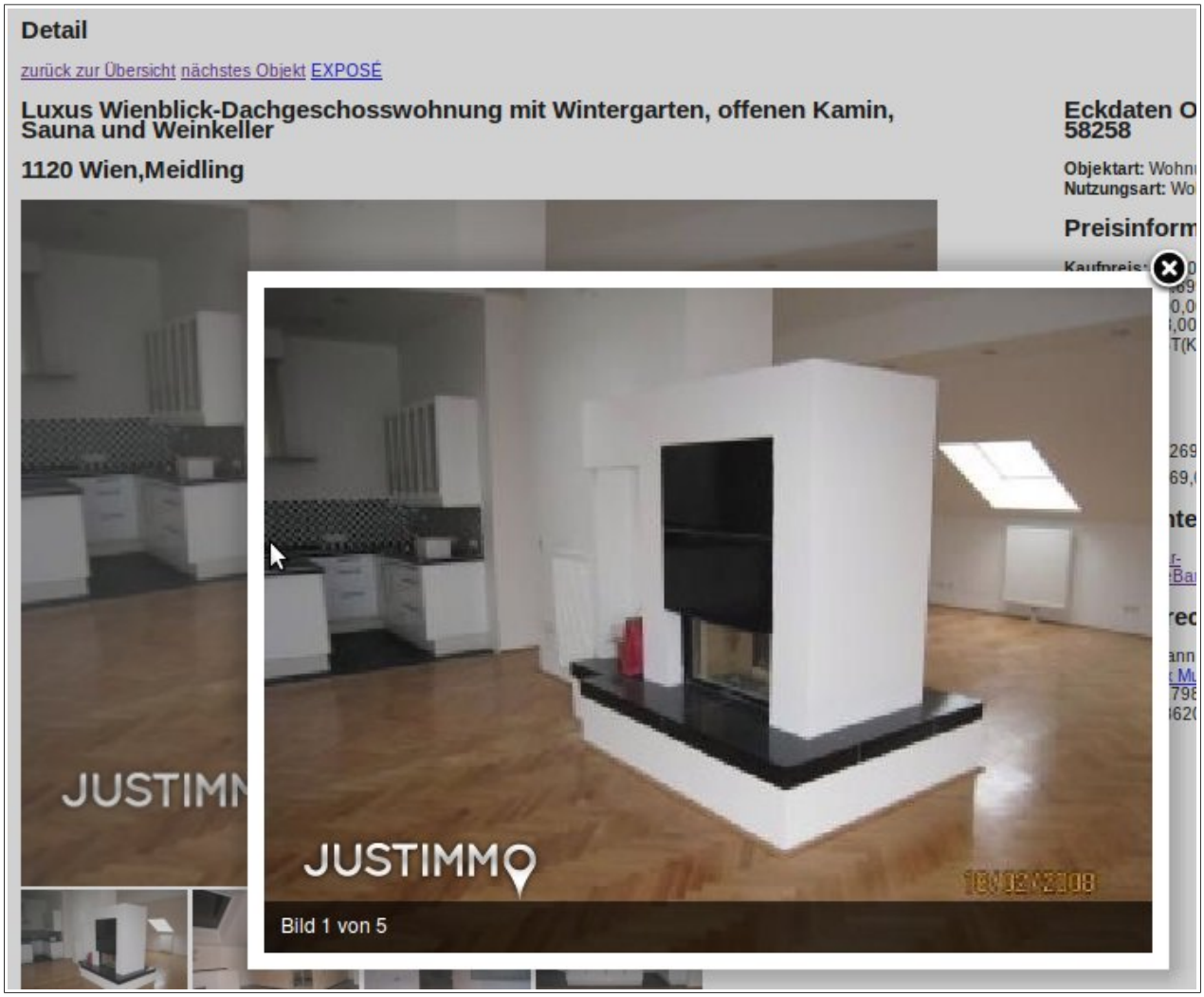

Fig. 12: the final result - realty object images in a lightbox

Known problems

Extension of filter criteria

Filter extension is a bit more complicated as this requires the involvement of an software developer who has to extend the corresponding domain model objects and validators. It would be great to improve that in the future, but this is a thing for the todo list.

Speed/performance

As commonly known, the extbase framework is not one of the best performing ones. Please consider to enable some kind of caching/compiling software like APC, PHP accelerator or eAccelerator – to name a few. See this list for other PHP accelerators.

To-Do list

- improve extension of filter properties
- improve persistence layer/SimpleXMLElement mapping
- flexform configuration for plugins
- improve translation
- improve justimmo geo viewhelpers
- unit testing

ChangeLog

Version:	Changes:
1.0.1	adding resetFilterOnInit setting to allow filter resets on “direct links” pages
1.0.0	first release