Edit on GitHub

# Documents ¶

Documents are the CMS part of Pimcore and are the way to go for managing unstructured contents using pages, content snippets and navigations.

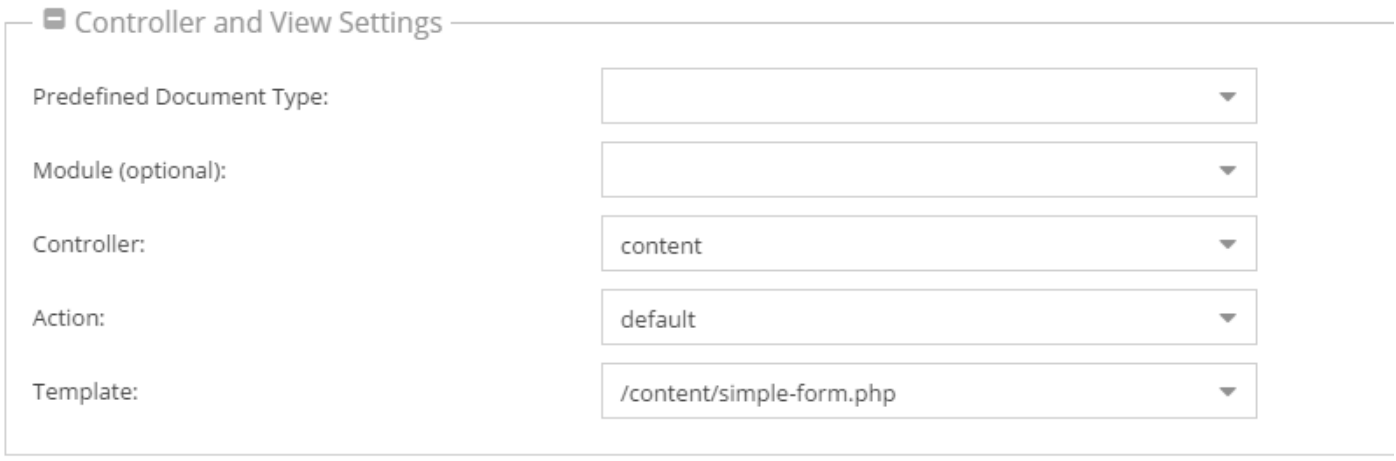
## Document Types ¶

Pimcore offers different types of documents and each of them offers functionality specific for the intended use-case.

| Type | Description |
|---|---|
| Page | Represents a typical web-page, the path in the tree is equal to the address in the browser. |
| Snippet | Makes it easier to extract often used contents into reusable containers. Can be embedded in pages or nested into other snippets. |
| Link | A simple web-link to be used in navigation. |
| Email | A document like the page, but with special functionality for transactional emails. |
| Newsletter (Document_Types/Newsletter_Documents.html) | Like email, but offering additional newsletter functionality. |
| Hardlink | Create links to other document structures and reuse them within a different structure / context. (see Hard link (https://en.wikipedia.org/wiki/Hard_link)) |
| Folder | Just like the folders you know from your local filesystem. |
| PrintPage (Document_Types/Print_Documents.html) | Like pages, but specialized for print (PDF preview, rendering options, ...) |
| PrintContainer (Document_Types/Print_Documents.html) | Organizing print pages in chapters and render them all together. |

## Document Configuration ¶

Many documents types are tight to the MVC (../MVC/index.html) and therefore they need a underlying controller/action and a template. They are directly specified in the document settings in the admin interface:




Not all of them are necessary, the table below shows which configurations are possible:

| Type | Controller | Action | Template | Description |
|---|---|---|---|---|
| 1 | X | X | | The specified controller/action is executed. If the action returns a response object, it is used for rendering. If no response is returned, the controller implements TemplateControllerInterface and `$this->setViewAutoRender($event->getRequest(), true, 'php');` is called, the template is auto-discovered based on controller and action name like `<BUNDLENAME>:<CONTROLLERNAME>:<ACTIONNAME>.html.php` . |
| 2 | X | X | X | Same as above but the template specified is rendered and not the auto-discovered template (only if action does not return a response). |
| 3 | | | X | Renders the template with the default controller/action, this is practical if there is only templating stuff. |

Optionally you can specify a module to each of the above combinations, this is useful if you want to use controllers/actions or templates out of plugins which are simply another Symfony bundle. The default module (when empty) is AppBundle.

Pimcore is shipped with a default controller containing a default action, which is called when only a template is given to the document.

You can set a default module/bundle, controller and action in the symfony configuration:

```
pimcore:
    routing:
        defaults:
            bundle: AppBundle
            controller: Default
            action: default
```

# Properties ¶

Properties (../Tools_and_Features/Properties.html) are very powerful in combination with documents. Below, you can find some examples where properties can be very useful for the use with documents.

1. **Navigation** - If you build the navigation based on the document-tree, sometimes you need special settings for the frontend, like separators or highlightings.
2. **Header Images** - Often there are header images on a website, if you don't want to define it for every page, you can use properties with inheritance. Then you can define a default one at the root document, and overrule this on a deeper level in the tree structure.
3. **Sidebars** - You can easily manage visibility of sidebars in specific documents.
4. **SEO** - It's also possible to use properties for SEO. It's very painful to define a nice title and description for every page on your site, with properties this is not necessary (inheritance).
5. **Protected Areas** - Closed user groups
6. Change the appearance of the website depending on the properties (eg. micro-sites, nested sites)
7. Mark them for some automated exports (PDF, RPC's, ...)

As you can see there are really useful cases for properties, feel free to use them for whatever they seem to be useful.

# A Few Facts ¶

- Documents follow the MVC pattern; therefore, Pimcore requires that there is at least one controller with an action and a template file.
- Pimcore comes with a DefaultController containing a defaultAction and a template file.
- Because of the MVC architecture, there is a clear separation between your data, the functionality and the templates.
- The normal way to create templates for Pimcore is to use pure PHP. There's no new template syntax to learn - just code what you want - feel free!
- Although the templates are written in PHP, there is a clear separation as mentioned before.

# Create Your First Document ¶

Working with documents is described in detail in our Create a First Project (../Getting_Started/Create_a_First_Project.html) manual.

# Document Topics ¶

- Creating editable templates (Editables/index.html)
- Navigation (Navigation.html)
- Inheritance (Inheritance.html)
- Working with the PHP API (Working_with_PHP_API.html)
- Web-to-Print (Document_Types/Print_Documents.html)
- Predefined Document Types (Predefined_Document_Types.html)