bobtemplates.plone Documentation

Release 3.0

Plone Community

Mar 11, 2022

Contents

Description

Overview of bobtemplates.plone features, compatibility and installation.

CHAPTER 1

Templates provided by bobtemplates.plone

Description

Overview and documentation of templates provided by bobtemplates.plone.

1.1 Add-on template

Description

Creating an add-on package and extending it with sub-template's.

1.1.1 Behavior sub-template

Description

Adding a custom Behavior to an existing add-on package.

With this `sub-template`, you can add a Behavior to a Plone package.

First create a Plone add-on package:

```
mrbob -O collective.todos bobtemplates.plone:addon
```

then change into the created folder `collective.todos` and create your Behavior:

```
mrbob bobtemplates.plone:behavior
```

It will ask you about the name of your Behavior class.

You will find the created behavior in the behaviors folder. You need to add the concrete code to your Behavior.

1.1.2 Content Type sub-template

Description

Adding a Dexterity Content Type to an existing add-on package.

With this sub-template, you can add a Dexterity Content Type to a Plone package.

First create a Plone add-on package:

```
plonecli create addon ./collective.todolist
```

or without the `plonecli`:

```
mrbob bobtemplates.plone:addon -O collective.todolist
```

then change into the created folder `collective.todolist` and create your first Content Type:

```
plonecli add content_type
```

or without the `plonecli`

```
mrbob bobtemplates.plone:content_type
```

It will ask you about the name of your Content Type and will use this name also for classes, interfaces. You also have the choice of using the XML supermodel or the zope.schema to define the models. The default base class is Container, but you can choose also Item. By default the template will create a class for every content type, but you can decide to use the generic Dexterity classes if you don't need your own content type classes.

Example

```
$ cd collective.todolist
```

Add a container Content Type

```
$ plonecli add content_type

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.

Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.
```

(continues on next page)

(continued from previous page)

```
RUN: git status --porcelain --ignore-submodules
Git state is clean.

--> Content type name (Allowed: _ a-z A-Z and whitespace) [Todo Task]: Todo List

--> Content type description: A todo list

--> Use XML Model [y]:

--> Dexterity base class (Container/Item) [Container]:

--> Should the content type globally addable? [n]: y

--> Should we filter content types to be added to this container? [y]:

--> Create a content type class [y]:

--> Activate default behaviors? [y]:


Should we run?:
git add .
git commit -m "Add content_type: Todo List"
in: /home/maik/develop/src/bobtemplates.plone/tmp/collective.todolist
[y]/n:
RUN: git add .
RUN: git commit -m "Add content_type: Todo List"
[master 5cb2b99] "Add content_type: Todo List"
11 files changed, 329 insertions(+), 1 deletion(-)
create mode 100644 src/collective/todolist/content/__init__.py
create mode 100644 src/collective/todolist/content/todo_list.py
create mode 100644 src/collective/todolist/content/todo_list.xml
create mode 100644 src/collective/todolist/profiles/default/types.xml
create mode 100644 src/collective/todolist/profiles/default/types/Todo_List.xml
create mode 100644 src/collective/todolist/tests/robot/test_ct_todo_list.robot
create mode 100644 src/collective/todolist/tests/test_ct_todo_list.py

Generated file structure at /home/maik/develop/src/bobtemplates.plone/tmp/collective.
↪todolist
```

Add an item Content Type

```
$ plonecli add content_type

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.

Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.
```

(continues on next page)

(continued from previous page)

```
RUN: git status --porcelain --ignore-submodules
Git state is clean.

--> Content type name (Allowed: _ a-z A-Z and whitespace) [Todo Task]: Todo List Item

--> Content type description: A todo list item

--> Use XML Model [y]:

--> Dexterity base class (Container/Item) [Container]: Item

--> Should the content type globally addable? [n]:

--> Parent container name [my_parent_container_type]: Todo List

--> Create a content type class [y]:

--> Activate default behaviors? [y]: n


('profile-plone.app.dexterity:default already in metadata.xml, skip adding!',)
Should we run?:
git add .
git commit -m "Add content_type: Todo List Item"
in: /home/maik/develop/src/bobtemplates.plone/tmp/collective.todolist
[y]/n:
RUN: git add .
RUN: git commit -m "Add content_type: Todo List Item"
[master 5226adf] "Add content_type: Todo List Item"
10 files changed, 310 insertions(+), 1 deletion(-)
create mode 100644 src/collective/todolist/content/todo_list_item.py
create mode 100644 src/collective/todolist/content/todo_list_item.xml
create mode 100644 src/collective/todolist/profiles/default/types.xml.example
create mode 100644 src/collective/todolist/profiles/default/types/Todo_List_Item.xml
create mode 100644 src/collective/todolist/tests/robot/test_ct_todo_list_item.robot
create mode 100644 src/collective/todolist/tests/test_ct_todo_list_item.py

Generated file structure at /home/maik/develop/src/bobtemplates.plone/tmp/collective.
↪todolist
```

1.1.3 Indexer sub-template

Description

Adding an indexer to an existing add-on package.

With this sub-template, you can add an indexer to a Plone add-on package, to define what and how an index get's filled.

First create a Plone add-on package:

```
mrbob -O collective.todo bobtemplates.plone:addon
```

then change into the created folder `collective.todo` and create your first View:

```
mrbob bobtemplates.plone:indexer
```

It will ask you for the indexer name and then creates a zcml config and a python file in the indexers folder for it. The Pyhon file contains a indexer method which you customize to your needs.

Example

```
$ cd collective.todo
```

Add a indexer

```
$ mrbob bobtemplates.plone:indexer

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.

Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.

RUN: git status --porcelain --ignore-submodules
Git state is clean.

--> Indexer name [my_custom_index]: funky_title

>>> reading Plone version from bobtemplate.cfg

<include package=".indexers" />
already in /home/maik/develop/src/collective.todo/src/collective/todo/configure.zcml,␣
↪skip adding!
Should we run?:
git add .
git commit -m "Add indexer: funky_title"
in: /home/maik/develop/src/collective.todo
[y]/n:
RUN: git add .
RUN: git commit -m "Add indexer: funky_title"
[master 2f4e8f9] "Add indexer: funky_title"
4 files changed, 53 insertions(+)
create mode 100644 src/collective/todo/indexers/funky_title.py
create mode 100644 src/collective/todo/indexers/funky_title.zcml
create mode 100644 src/collective/todo/tests/test_indexer_funky_title.py
```

1.1.4 Portlet sub-template

Description

Adding a Portlet to an existing add-on package.

With this sub-template, you can add a Portlet to a Plone add-on package.

First create a Plone add-on package:

```
mrbob -O collective.todo bobtemplates.plone:addon
```

then change into the created folder `collective.todo` and create your first Portlet:

```
mrbob bobtemplates.plone:portlet
```

Portlets are used for wide variety of tasks in Plone and you can read about it in Portlets reference manual for in-depth information. It will just ask for the name of the portlet and the same name is used to create file structures and as well as the portlet name. This will create a basic portlet which ask for city name and country and it will fetch weather data of that city using Yahoo Weather API.

Example

```
$ cd collective.todo
```

Add a Portlet

```
$ mrbob bobtemplates.plone:portlet

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.

Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.



RUN: git status --porcelain --ignore-submodules
Git state is clean.

--> Portlet name to display for the portlet [Weather]: My Weather Portlet

>>> reading Plone version from bobtemplate.cfg

Should we run?:
git add .
git commit -m "Add portlet: My Weather Portlet"
in: /Users/akshay/plone/collective.todo
[y]/n:
RUN: git add .
RUN: git commit -m "Add portlet: My Weather Portlet"
[master ea9d848] "Add portlet: My Weather Portlet"
5 files changed, 177 insertions(+)
create mode 100644 src/collective/todo/portlets/my_weather_portlet.pt
create mode 100644 src/collective/todo/portlets/my_weather_portlet.py
create mode 100644 src/collective/todo/tests/test_my_weather_portlet.py

Generated file structure at /Users/akshay/plone/collective.todo
```

1.1.5 Rest-API-Service sub-template

Description

Adding a restapi_service to an existing add-on package.

With this sub-template, you can add a restapi_service to a Plone package.

First create a Plone add-on package:

```
mrbob -O collective.todos bobtemplates.plone:addon
```

then change into the created folder `collective.todos` and create your restapi_service:

```
mrbob bobtemplates.plone:restapi_service
```

It will ask you about the name of your Rest-API service class and name.

You will find the created restapi_service in the services folder inside the api folder.

1.1.6 Subscriber sub-template

Description

Adding a subscriber to an existing add-on package.

With this sub-template, you can add an event subscriber (handler) to a Plone add-on package.

First create a Plone add-on package:

```
mrbob -O collective.todo bobtemplates.plone:addon
```

then change into the created folder `collective.todo` and create your first View:

```
mrbob bobtemplates.plone:subscriber
```

It will ask you for the subscriber handler file name and then creates this file in the subscribers folder. This file contains a handler method which you customize to your needs.

Example

```
$ cd collective.todo
```

Add a subscriber

```
$ mrbob bobtemplates.plone:subscriber

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.
```

(continues on next page)

(continued from previous page)

```
Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.



RUN: git status --porcelain --ignore-submodules
Git state is clean.

--> Subscriber handler file name (without extension) [obj_modified_do_something]: obj_
↪mod_clear_cache

>>> reading Plone version from bobtemplate.cfg

rename example zcml file
Should we run?:
git add .
git commit -m "Add subscriber: obj_mod_clear_cache"
in: /home/maik/develop/src/collective.todo
[y]/n:
RUN: git add .
RUN: git commit -m "Add subscriber: obj_mod_clear_cache"
[master 53d7e16] "Add subscriber: obj_mod_clear_cache"
5 files changed, 47 insertions(+)
create mode 100644 src/collective/todo/subscribers/__init__.py
create mode 100644 src/collective/todo/subscribers/configure.zcml
create mode 100644 src/collective/todo/subscribers/obj_mod_clear_cache.py
create mode 100644 src/collective/todo/tests/test_subscriber_obj_mod_clear_cache.py
```

1.1.7 Svelte app sub-template

Description

Adding a Svelte app to an existing add-on package.

With this sub-template, you can add modern Svelte app's and/or custom elements ak *web components* to a Plone package.

First create a Plone add-on package:

```
mrbob -O collective.svelteapps bobtemplates.plone:addon
```

then change into the created folder `collective.svelteapps`:

```
mrbob bobtemplates.plone:svelte_app
```

It will ask you about the name of your app and if you want Svelte to generate custom element instead of a native Svelte app. This will generate the structure of your app and also register the bundles inside the Plone package.

Mainly it creates a folder for the Svelte src in ./svelte_src/<my_svelte_app>. Inside this folder you have to run:

```
npm install
```

and then to build the app bundles from the src:

```
npm run build
```

This will build you bundles and put them into the output folder, which is inside of your Plone package in svelte_apps/<my_svelte_app>. From here it will be loaded into Plone by the resource registry.

For development you can also run:

```
npm run dev
```

This will watch the src folder and update the bundles after every change. In the Plone HTML filter control panel, you have to make sure that your custom element (tag) is in the list of valid tags. If you are not using custom element support, you can also change the target selector in the main.js to select based on CSS id's or classes and use a div tag with that class to activate the app.

To activate the app in Plone, you can put either the custom element for example my-custom-element inside any html, even with the TinyMCE source code mode.

```
<h1>Heading in your Plone page</h1>
<my-custom-element></my-custom-element>
<p>Some more static content of the Plone page.</p>
```

or in case of a nativ Svelte app, which uses a CSS class "my-custom-element" to activate the app, you can include it like this:

```
<h1>Heading in your Plone page</h1>
<div class="my-custom-element"></div>
<p>Some more static content of the Plone page.</p>
```

Necessary settings in Plone

If you are using custom-elements there some settings, to make so that you can use theme in TinyMCE.

In the HTML filter, you want to add the name of your custom element, for example svelte-card to the list of allowed tags.

We also added the slot attribute to the custom attrbitures list, so that we can use it in our custom elements.

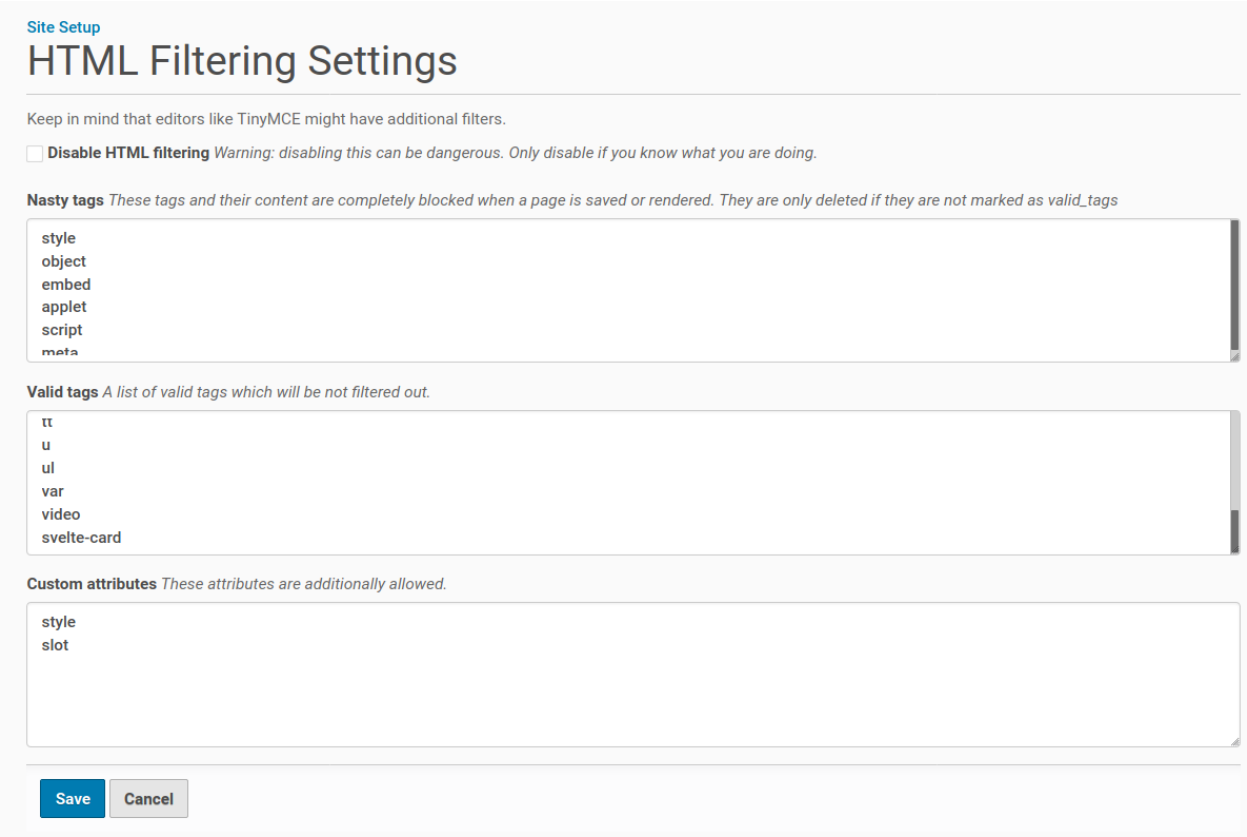
```
<h1>Heading in your Plone page</h1>

<svelte-card>
  <span slot="head">Svelte Card</span>
  <p slot="content">Svelte is nice! This custom element has no dependencies, is very␣
↪small and works in any modern Browser and most JS Frameworks.</p>
</svelte-card>

<p>Some more static content of the Plone page.</p>
```

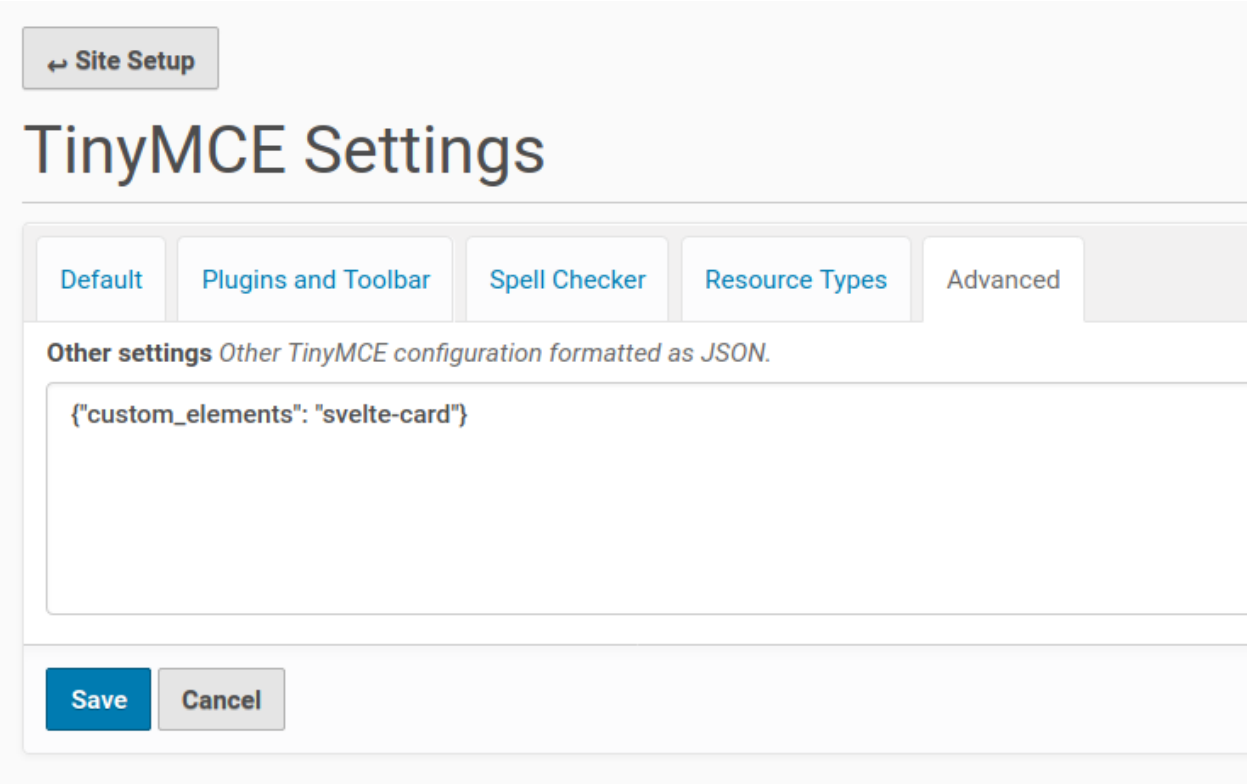
The code above needs us to add some settings to TinyMCE.

You could do this also via registry settings in you profiles:

```
<?xml version='1.0' encoding='UTF-8'?>
<registry>
<records interface="Products.CMFPlone.interfaces.controlpanel.ITinyMCESchema" prefix=
↪"plone">
    <value key="other_settings" purge="false">{"custom_elements": "svelte-card"}</
↪value>
```

(continues on next page)


Site Setup
HTML Filtering Settings
Keep in mind that editors like TinyMCE might have additional filters.
Disable HTML filtering Warning: disabling this can be dangerous. Only disable if you know what you are doing.
Nasty tags These tags and their content are completely blocked when a page is saved or rendered. They are only deleted if they are not marked as valid_tags
style
object
embed
applet
script
meta
Valid tags A list of valid tags which will be not filtered out.
tt
u
ul
var
video
svelte-card
Custom attributes These attributes are additionally allowed.
style
slot
Save
Cancel


Site Setup
TinyMCE Settings
Default
Plugins and Toolbar
Spell Checker
Resource Types
Advanced
Other settings Other TinyMCE configuration formatted as JSON.
{"custom_elements": "svelte-card"}
Save
Cancel

(continued from previous page)

```
</records>
</registry>
```

Optimizing

By default all apps are enabled and loaded by default. This makes it easy for you to start, but can lead to too much resources loaded, even if they don't needed at the moment. To optimize this, you can change the *enabled* setting in the registry: profiles/default/registry/rreg-<your_svelte_app_name>.xml. If you set the *enabled* setting to False, Plone will not load the resources files globally. Instead you can include them by your self where you need it.

For example, you could have a Plone BrowserView which renders the app and also enables the resources just for this view. For more infos on how to enable resources on demand, have a look in the docs here: https://docs.plone.org/adapt-and-extend/theming/resourceregistry.html#controlling-resource-and-bundle-rendering

Another way to include them is to simply add them in a page template with the fill-slot command and inject it into the header.

1.1.8 Theme sub-template

Description

Adding a theme to an existing add-on package.

With this sub-template, you can add a Plone theme to a Plone package. This theme template is useful to integrate existing themes or mockups from Designers into Plone. It doesn't come with any bacelonata resources nor does it make any assumptions how you build your static file like LESS, SCSS or JavaScript. You have to build your own setup for this. We do this because, many themes come with a different set of tools, so just use the tooling of the theme or build your own if there isn't any.

First create a Plone add-on package:

```
mrbob -O plonetheme.blacksea bobtemplates.plone:addon
```

then change into the created folder `plonetheme.blacksea` and your theme:

```
mrbob bobtemplates.plone:theme
```

It will ask you about the name of your theme and will generate the structure of your theme and also register it inside the Plone package.

The only thing you might want to add to your setup.py manually are the two following packages, which are to be added to the `install_requires`:

- collective.themesitesetup
- collective.themefragments

These packages are optional but recommended to have support for configuration and additional PageTemplate thru theme packages. If you want to deploy your theme later as a ZIP-Files, be aware that these packages should be installed on the server as well.

Themes which are supporting these additional functionality are called `Extended Themes`.

1.1.9 Theme Barceloneta sub-template

Description

Adding a barceloneta theme to an existing add-on package.

With this sub-template, you can add a Plone theme to a Plone package.

First create a Plone add-on package:

```
mrbob -O plonetheme.blacksea bobtemplates.plone:addon
```

then change into the created folder `plonetheme.blacksea` and your theme:

```
mrbob bobtemplates.plone:theme_barceloneta
```

It will ask you about the name of your theme and will generate the structure of your theme and also register it inside the Plone package.

The only thing you might want to add to your setup.py manually are the two following packages, which are to be added to the `install_requires`:

- collective.themesitesetup
- collective.themefragments

These packages are optional but recommended to have support for configuration and additional PageTemplate thru theme packages. If you want to deploy your theme later as a ZIP-Files, be aware that these packages should be installed on the server as well.

Themes which are supporting these additional functionality are called `Extended Themes`.

1.1.10 View sub-template

Description

Adding a View to an existing add-on package.

With this sub-template, you can add a View to a Plone add-on package.

First create a Plone add-on package:

```
mrbob -O collective.todo bobtemplates.plone:addon
```

then change into the created folder `collective.todo` and create your first View:

```
mrbob bobtemplates.plone:view
```

It will ask if you need Python class and template file, you can have both or at least one to have a working view. Based on the input it will ask about class name, template name or both. By default it will suggest you to use class name as view name (the part of url) but you can also change it. You can see your newly created view by using the url that you used for view name on `IFolderish` interface.

Example

```
$ cd collective.todo
```

Add a View

```
$ mrbob bobtemplates.plone:view

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.

Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.



RUN: git status --porcelain --ignore-submodules
Git state is clean.

--> Should the view have a Python class? [y]: y

--> Python class name [MyView]: DemoView

--> View name (part of the URL) [demo-view]: demo-view

--> Should the View have a template file? [y]:

--> Template name (without extension) [demo_view]: demo

>>> reading Plone version from bobtemplate.cfg

Should we run?:
git add .
git commit -m "Add view: demo-view"
in: /Users/akshay/plone/collective.todo
[y]/n: y
RUN: git add .
RUN: git commit -m "Add view: demo-view"
[master 64d8a8b] "Add view: demo-view"
4 files changed, 93 insertions(+)
create mode 100644 src/collective/todo/tests/test_view_demo_view.py
create mode 100644 src/collective/todo/views/demo.pt
create mode 100644 src/collective/todo/views/demo_view.py

Generated file structure at /Users/akshay/plone/collective.todo
```

1.1.11 Viewlet sub-template

Description

Adding a Viewlet to an existing add-on package.

With this sub-template, you can add a Viewlet to a Plone add-on package.

First create a Plone add-on package:

```
mrbob -O collective.todo bobtemplates.plone:addon
```

then change into the created folder `collective.todo` and create your first Viewlet:

```
mrbob bobtemplates.plone:viewlet
```

It will ask if you need Python class and template file, you can have both or at least one to have a working viewlet. Based on the input it will ask about class name, template name or both. By default it will suggest you to use class name as viewlet name but you can also change it. This will create a viewlet registered to `IAboveContentTitle` viewlet manager and on `IDocument` interface.

Example

```
$ cd collective.todo
```

Add a Viewlet

1.1.12 Vocabulary sub-template

Description

Adding a vocabulary to an existing add-on package.

With this sub-template, you can add a dynamic Vocabulary to a Plone package.

First create a Plone add-on package:

```
mrbob -O collective.todos bobtemplates.plone:addon
```

then change into the created folder `collective.todos` and create your Vocabulary:

```
mrbob bobtemplates.plone:vocabulary
```

It will ask you about the name of your Vocabulary class.

You will find the created Vocabulary in the vocabularies folder. You need to change the concrete code to generate your Vocabulary terms. You will also find your Vocabulary in the Dexterity schema editor in your Browser. Your Vocabulary is registered in the configure.zcml, there you can find also the name of the Vocabulary under which you can get it in Python code.

With this template you can create a basic Plone package.

```
mrbob -O collective.todos bobtemplates.plone:addon
```

This will create a Python package for you, which you can extend manually or by using other sub-templates like `theme` or `content_type` from bobtemplates.plone.

1.1.13 Example

```
$ mrbob bobtemplates.plone:addon -O collective.todolist

Welcome to mr.bob interactive mode. Before we generate directory structure, some␣
↪questions need to be answered.

Answer with a question mark to display help.
Values in square brackets at the end of the questions show the default value if there␣
↪is no answer.


--> Package description [An add-on for Plone]: A todo list add-on for Plone

--> Do you want me to initialze a GIT repository in your new package? (y/n) [y]:


RUN: git init
Leeres Git-Repository in /home/maik/develop/src/bobtemplates.plone/tmp/collective.
↪todolist/.git/ initialisiert

Should we run?:
git add .
git commit -m "Create addon: collective.todolist"
in: /home/maik/develop/src/bobtemplates.plone/tmp/collective.todolist
[y]/n:
RUN: git add .
RUN: git commit -m "Create addon: collective.todolist"
[master (Basis-Commit) 04b6727] "Create addon: collective.todolist"
48 files changed, 1381 insertions(+)
create mode 100644 .coveragerc
create mode 100644 .editorconfig
create mode 100644 .gitattributes
create mode 100644 .gitignore
create mode 100644 .gitlab-ci.yml
create mode 100644 .travis.yml
create mode 100644 CHANGES.rst
create mode 100644 CONTRIBUTORS.rst
create mode 100644 DEVELOP.rst
create mode 100644 LICENSE.GPL
create mode 100644 LICENSE.rst
create mode 100644 MANIFEST.in
create mode 100644 README.rst
create mode 100644 bobtemplate.cfg
create mode 100644 buildout.cfg
create mode 100644 docs/index.rst
create mode 100644 requirements.txt
create mode 100644 setup.cfg
create mode 100644 setup.py
create mode 100644 src/collective/__init__.py
create mode 100644 src/collective/todolist/__init__.py
create mode 100644 src/collective/todolist/browser/__init__.py
create mode 100644 src/collective/todolist/browser/configure.zcml
create mode 100644 src/collective/todolist/browser/overrides/.gitkeep
create mode 100644 src/collective/todolist/browser/static/.gitkeep
create mode 100644 src/collective/todolist/configure.zcml
create mode 100644 src/collective/todolist/interfaces.py
```

(continues on next page)

(continued from previous page)

```
create mode 100644 src/collective/todolist/locales/README.rst
create mode 100644 src/collective/todolist/locales/__init__.py
create mode 100644 src/collective/todolist/locales/collective.todolist.pot
create mode 100644 src/collective/todolist/locales/en/LC_MESSAGES/collective.todolist.
↪po
create mode 100644 src/collective/todolist/locales/update.py
create mode 100755 src/collective/todolist/locales/update.sh
create mode 100644 src/collective/todolist/permissions.zcml
create mode 100644 src/collective/todolist/profiles/default/browserlayer.xml
create mode 100644 src/collective/todolist/profiles/default/catalog.xml
create mode 100644 src/collective/todolist/profiles/default/metadata.xml
create mode 100644 src/collective/todolist/profiles/default/registry.xml
create mode 100644 src/collective/todolist/profiles/default/rolemap.xml
create mode 100644 src/collective/todolist/profiles/uninstall/browserlayer.xml
create mode 100644 src/collective/todolist/setuphandlers.py
create mode 100644 src/collective/todolist/testing.py
create mode 100644 src/collective/todolist/tests/__init__.py
create mode 100644 src/collective/todolist/tests/robot/test_example.robot
create mode 100644 src/collective/todolist/tests/test_robot.py
create mode 100644 src/collective/todolist/tests/test_setup.py
create mode 100644 src/collective/todolist/upgrades.py
create mode 100644 src/collective/todolist/upgrades.zcml

Generated file structure at /home/maik/develop/src/bobtemplates.plone/tmp/collective.
↪todolist
```

1.2 Buildout template

Description

Creating a Plone Buildout.

With this template you can create a basic Plone Buildout, which often used as a project Buildout. The project Buildout then can contain all project related packages directly or as separate repositories.

```
mrbob bobtemplates.plone:Buildout -O my_project
```

This will ask you for a Plone version and will create a Buildout folder for you. The Buildout configuration is using the buildout.plonetest configurations provided by the Plone community for developing and testing Plone. For more information about the buildout.plonetest configurations see here: https://github.com/collective/buildout.plonetest

```
$ tree -L 2 my_project/
my_project/
├── bobtemplate.cfg
├── buildout.cfg
├── requirements.txt
└── src
```

To initialize the Buildout, change into the Buildout folder and use the following commands:

```
virtualenv .
./bin/pip install -r requirements.txt
./bin/buildout
```

1.3 Theme Package template

Description

Creating a Plone package with a Barceloneta theme.

With this `template`, you can create a full theme package at once. This will create a Plone package with a Barceloneta theme including a full Grunt setup and a copy of all Barceloneta resources.

If you like it a bit more modular or you don't want to use the Barceloneta theme as a start, you might want to use the *theme sub-template* instead!

Note: The theme_package template is deprecated, please use *theme_barceloneta sub-template* inside an existing add-on package.

```
mrbob -O plonetheme.blacksea bobtemplates.plone:theme_package
```

then change into the created folder `plonetheme.blacksea` initialize it as followed:

```
virtualenv .
./bin/pip install -r requirements.txt
./bin/buildout
```

After that you can start the Plone instance and activate your theme.

CHAPTER 2

GIT-Support

Since 3.2.x bobtemplates.plone has integrated GIT support and can check for clean repository state, initialize a GIT repository for a new created package and commit all changes a template is making.

For your convenience, you can set some defaults in your mrbob user configuration

```
cat ~/.mrbob

[mr.bob]
verbose = False

[variables]
author.name = Maik Derstappen
author.email = md@derico.de
author.github.user = MrTango
plone.version = 5.1-latest
package.git.init = True
package.git.autocommit = True
package.git.disabled = False
```

This will allow you to define if you want to allways git initialize a created package and do a commit after every sub-templates was rendered, without asking all the time. You can even disable the git support completely if you need to.

2.1 Example configurations

2.1.1 Default

If you don't set any default values, the default answers of the questions are as follows.

You will be questioned for `git init` and `git auto commit`, by default they are set to YES. But you can always switch them to No.

2.1.2 Fix settings

If you always work the same, you can set these answers in your .mrbob file and you will not be ask all the time.

```
package.git.init: True
package.git.autocommit: True
```

2.1.3 Disable git init

In case you are don't want an extra git repository for every package you create, but use a parent project based repository, you can disable the git init, by setting `package.git.init` to `False`.

```
package.git.init = False
```

2.1.4 Disable git support completely

To disable the GIT support completely, set `package.git.disable` to `True`. But it's hightly recommended to let it active, because sub-templates can silendly override files. **So be warned!**

```
package.git.disabled = True
```

CHAPTER 3

Upgrade existing Plone packages

3.1 bobtemplate.cfg

To upgrade an existing Plone package, to work with the new sub-templates and also the plonecli. You simply need to add a bobtemplate.cfg with some content like this:

```
[main]
version = 5.2
template = plone_addon
python = python3.7
```

The version is used to make a difference in some templates, either for Plone 5 or Plone 4 packages. You can always create an addon package with the plonecli and copy over the bobtemplate.cfg in your old package.

3.2 other files

You should have at leased a requirements.txt and a buildout.cfg file. But it's recommended to add the following files from a generated addon package into your existing package:

- requirements.txt
- constraints.txt
- constraints_plone52.txt
- constraints_plone51.txt
- test_plone52.cfg
- test_plone51.cfg
- tox.ini

3.3 Folder structure

Since this package expects a specific folder structure, you should compare it to your existing structure and adjust it, where needed.

3.4 Upgrade steps

The upgrade_step sub-template uses an upgrades folder to create upgrade steps in it. Therefor you should adjust existing upgrade step configuration to it. You can have a look at Products.EasyNewsletter where this was done.

CHAPTER 4

Developing bobtemplates.plone templates

4.1 Setup dev environment

In the package folder create a virtualenv and install the package:

```
virtualenv --clear .
./bin/python setup.py develop
```

4.2 Intro

We can have standalone templates and sub-templates for bobtemplates.plone. By convention we will have a python module and a template folder for every template and use some generic functions from the base module.

All templates are living inside the `bobtemplates/plone` folder, in their own template folder. All module files are placed inside the `bobtemplates/plone` folder and are referenced from hook commands in the `.mrbob.ini` file in the template folders. They get called by different `mrbob hooks`, like `pre_render`, `post_render` or `pre_question hook` and so on.

4.3 Standalone templates

Standalone templates are normal templates for mrbob, which are meant to live standalone and are not depending on any other template.

Examples are the `buildout` and `addon` templates. For details see the documentation of the mrbob package.

4.4 Sub-templates

Sub-templates are templates which are living inside an existing package created by a standalone template like the `addon` template. These templates extend the existing standalone template structure by new features like a `theme` or a `content_type`.

Sub-templates are searching first for the `setup.py` and a `bobtemplate.cfg` file inside the package and configure all needed parameters with this information. See also *Upgrade existing Plone packages* to see how to upgrade an existing package to be compatible with sub-templates. Every sub-template should define a `pre_renderer` and a `post_renderer` hook in their `.mrbob.ini` which points to a method in their sub-templates module.

```
[template]
pre_render = bobtemplates.plone.<YOURTEMPLATE_MODULE>:pre_renderer
post_render = bobtemplates.plone.<YOURTEMPLATE_MODULE>:post_renderer
```

```
from bobtemplates.plone.base import base_prepare_renderer


def pre_renderer(configurator):
    configurator = base_prepare_renderer(configurator)
    configurator.variables['template_id'] = 'content_type'


def post_renderer(configurator):
    """
    """
```

As you can see, by convention we define a template_id here. We also call the base_prepare_renderer method from the base module first to setup some generic variables. The `pre_renderer` hook is a good place to set template specific variables. If you need you can override or extend the variables, like we do in the `content_type` module with the `target_folder`.

```
configurator.target_directory = configurator.variables['package_folder']
```

The `post_renderer` method is a good place to update configuration files, like we do for example in the `theme` and `content_type` sub-templates.

You can also print some useful advice for the developer here, as we do in the `vocabulary` sub-template for example.

4.5 Template Registration

Even though you can use bobtemplates without registration, you should register the template to allow plonecli and future mrbob versions to query for it. The registration is done by adding a Python entry point into the `setup.py` of `bobtemplates.plone` and by adding a short method to the `bobregistry.py` file. You can of course create your own custom package, analog to bobtemplates.plone and register your templates plonecli the same way. This could be used for example for your agency or client specific code structures. If you need help by creating such custom bobtemplates and plonecli integration's, give us a sign on Gitter: https://gitter.im/plone/plonecli.

Let's look first on the entry point:

```
entry_points={
    'mrbob_templates': [
        'plone_addon = bobtemplates.plone.bobregistry:plone_addon',
        'plone_content_type = bobtemplates.plone.bobregistry:plone_content_type',
```

(continues on next page)

(continued from previous page)

```
        'plone_vocabulary = bobtemplates.plone.bobregistry:plone_vocabulary',
    ],
```

This registers every template globally for mrbob and tools like plonecli. The first part is the global template name and the second part points to a method in the bobregistry module. This method gives back some details for the template.

```
def plone_vocabulary():
    reg = RegEntry()
    reg.template = 'bobtemplates.plone:vocabulary'
    reg.plonecli_alias = 'vocabulary'
    reg.depend_on = 'plone_addon'
    return reg
```

The method defines the following things:

- `template`: the mrbob template to use
- `plonecli_alias`: a short name alias which will be used by plonecli
- `depend_on`: an optional global parent template

We use here globally unique template names which have the `plone_` prefix. That is because other `bobtemplate` packages might register templates too and we want to avoid name clashes.

4.6 Testing

All templates and sub-templates should have tests for the structure they provide.

These tests will give developers a good starting point to write tests for their own code. Also these tests will be called by Tox and on Travis to make sure that all the structures created by `bobtemplates.plone` are working and tested.

We run tests for both all the templates with every combination and inside the generated packages.

For example tests could be run only on `addon`. Alternately, for a package with Dexterity content types, tests could be run first for the `add-on` template, then inside the package created by the `content_type` sub-template.

The tests are running after all templates for a case are applied.

To run all tests locally, just run `tox` without any parameter. You can also run individual tests for a specific environment. To get a list of all environments run `tox -l`.

```
$ tox -l
py37-lint
py27-lint
docs
py27-packagetests
py37-packagetests
py27-skeletontests-Plone43-template-addon
py27-skeletontests-Plone51-template-addon
py27-skeletontests-Plone52-template-addon
py37-skeletontests-Plone52-template-addon
py27-skeletontests-Plone43-template-addon_content_type
py27-skeletontests-Plone51-template-addon_content_type
py27-skeletontests-Plone52-template-addon_content_type
py37-skeletontests-Plone52-template-addon_content_type
py27-skeletontests-Plone43-template-addon_view
py27-skeletontests-Plone51-template-addon_view
```

(continues on next page)

(continued from previous page)

```
py27-skeletontests-Plone52-template-addon_view
py37-skeletontests-Plone52-template-addon_view
py27-skeletontests-Plone43-template-addon_viewlet
py27-skeletontests-Plone51-template-addon_viewlet
py27-skeletontests-Plone52-template-addon_viewlet
py37-skeletontests-Plone52-template-addon_viewlet
py27-skeletontests-Plone43-template-addon_portlet
py27-skeletontests-Plone51-template-addon_portlet
py27-skeletontests-Plone52-template-addon_portlet
py37-skeletontests-Plone52-template-addon_portlet
py27-skeletontests-Plone43-template-addon_theme
py27-skeletontests-Plone51-template-addon_theme
py27-skeletontests-Plone52-template-addon_theme
py37-skeletontests-Plone52-template-addon_theme
py27-skeletontests-Plone51-template-addon_theme_barceoneta
py27-skeletontests-Plone52-template-addon_theme_barceoneta
py37-skeletontests-Plone52-template-addon_theme_barceoneta
py27-skeletontests-Plone43-template-addon_vocabulary
py27-skeletontests-Plone51-template-addon_vocabulary
py27-skeletontests-Plone52-template-addon_vocabulary
py37-skeletontests-Plone52-template-addon_vocabulary
py27-skeletontests-Plone43-template-addon_behavior
py27-skeletontests-Plone51-template-addon_behavior
py27-skeletontests-Plone52-template-addon_behavior
py37-skeletontests-Plone52-template-addon_behavior
py27-skeletontests-Plone43-template-addon_restapi_service
py27-skeletontests-Plone51-template-addon_restapi_service
py27-skeletontests-Plone52-template-addon_restapi_service
py37-skeletontests-Plone52-template-addon_restapi_service
py27-skeletontests-Plone43-template-theme_package
py27-skeletontests-Plone51-template-theme_package
coverage-report
```

You can run just one of them:

```
tox -e py27-skeletontests-Plone52-template-addon
```

or call all of the same template but for different Plone versions:

```
tox -e py27-skeletontests-Plone43-template-addon_content_type,py27-skeletontests-
↪Plone51-template-add-on_content_type,py27-skeletontests-Plone52-template-add-on_
↪content_type
```

Note: There is no empty space between the list elements!

4.6.1 Running a specific test

The actual tests are written with the pytest module, therefor you can always run them with pytest directly.

To run a specific pytest with Tox, you can pass additional arguments to pytest, buy putting them after the `--` parameter.

```
$ tox -e py36-packagetests -- -k test_set_global_vars
```

Increase verbosity of Tox/Pytest

```
tox -e py37-packagetests -vv -- -s
```

Package tests

Package tests are for testing the code of bobtemplates.plone it self. These code is used to generate and update the structures of the generated packages.

You can find these test in the `package-test` folder. This is a good place to test everything related to the generation process.

Skeleton tests

Skeleton tests are for testing, that the generated packages are actually work. We generate the packages, with different combinations of sub-templates, build and run the tests inside.

The tests are defined in the directory `skeleton-tests` and are called by `tox` as defined in `tox.ini`.

If you add new test cases (files), make sure that they are in the `tox.ini` and also included int the Travis matrix, see below!

Skeleton tests it self are using pytest too, but the tests inside the generated packages are Zope tests running by zc.testrunner. Starting from version 4.x, packages generated by bobtemplates.plone are containing also a tox setup by them self. This allows you to easily test your package against multiple Python and Plone versions.

4.7 Generating Travis matrix from tox.ini

```
$ python tox2travis.py
matrix:
    include:
        - env: TOXENV=py37-lint
        python: "3.7"
        - env: TOXENV=py27-lint
        - env: TOXENV=docs
        - env: TOXENV=py27-packagetests
        - env: TOXENV=py37-packagetests
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon
        - env: TOXENV=py27-skeletontests-Plone51-template-addon
        - env: TOXENV=py27-skeletontests-Plone52-template-addon
        - env: TOXENV=py37-skeletontests-Plone52-template-addon
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_content_type
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_content_type
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_content_type
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_content_type
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_view
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_view
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_view
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_view
        python: "3.7"
```

(continues on next page)

(continued from previous page)

```
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_viewlet
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_viewlet
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_viewlet
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_viewlet
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_portlet
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_portlet
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_portlet
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_portlet
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_theme
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_theme
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_theme
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_theme
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_theme_barceoneta
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_theme_barceoneta
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_theme_barceoneta
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_vocabulary
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_vocabulary
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_vocabulary
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_vocabulary
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_behavior
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_behavior
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_behavior
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_behavior
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-addon_restapi_service
        - env: TOXENV=py27-skeletontests-Plone51-template-addon_restapi_service
        - env: TOXENV=py27-skeletontests-Plone52-template-addon_restapi_service
        - env: TOXENV=py37-skeletontests-Plone52-template-addon_restapi_service
        python: "3.7"
        - env: TOXENV=py27-skeletontests-Plone43-template-theme_package
        - env: TOXENV=py27-skeletontests-Plone51-template-theme_package
        - env: TOXENV=coverage-report
```

replace the current matrix in `.travis.yml` with the result.

CHAPTER 5

Introduction

`bobtemplates.plone` provides a mr.bob template to generate packages for Plone projects.

To create a package like `collective.myaddon`

```
pip install bobtemplates.plone
mrbob -O collective.myaddon bobtemplates.plone:addon
```

You can also create a package with nested name space

```
mrbob -O collective.foo.myaddon bobtemplates.plone:addon
```

Note: With the plonecli, we have a nice commandline client for bobtemplates.plone. We highly recommend to use the plonecli, because it adds auto completion and some nice helpers to bobtemplates.plone.

CHAPTER 6

Features

packages created with `bobtemplates.plone` use the current best-practices when creating an add-on and does all of boilerplate for you.

6.1 Provided templates

- addon
- behavior
- content_type
- indexer
- portlet
- restapi_service
- subsriber
- svelte_app
- theme
- theme_barceloneta
- view
- viewlet
- vocabulary
- buildout
- theme_package [deprecated] >> Please use the theme_barceloneta sub-template!

Note: For the full list of supported templates/subtemplates, you can use: `plonecli -l`

CHAPTER 7

Compatibility

Add-on's created with `bobtemplates.plone` are tested to work in Plone 4.3.x and Plone 5. They should also work with older versions but that was not tested. It should work on Linux, Mac and Windows.

CHAPTER 8

Installation

8.1 Installation global for a user (recommended)

To have `bobtemplates.plone` and the plonecli always avilable, it's recommended to install it globally for your user.

```
pip install plonecli --user
```

This will install the `plonecli` and `bobtemplates.plone`. If you only want `bobtemplates.plone`, you install only this package as follow:

```
pip install bobtemplates.plone --user
```

8.2 Installation in a Virtualenv

You can also install `bobtemplates.plone` in a Virtualenv.

```
pip install bobtemplates.plone
```

With `pip 6.0` or newer `mr.bob` will automatically be installed as a dependency. If you still use a older version of pip you need install `mr.bob` before `bobtemplates.plone`.

```
pip install mr.bob
```

8.3 Use plonecli and bobtemplates.plone

You can use the bobtemplates now with the `plonecli`:

```
plonecli create addon src/collective.foo
cd src/collective.foo
plonecli add content_type
plonecli build test serve
```

```
mrbob bobtemplates.plone:addon -O collective.foo
cd src/collective.foo
mrbob bobtemplates.plone:content_type
virtualenv .
./bin/pip install -r requirements.txt
./bin/buildout
./bin/test
./bin/instance fg
```

8.3.1 Changing the default Python and Plone versions

By default you will build a virtualenv with Python2.7 and a buildout Plone 5.2. You can change this by customizing the buildout.cfg to extend one of the other test file, like test_plone43.cfg. Also you can change the requirements.txt to point to another constraints file like constraints_plone43.txt.

8.4 Additional information on plonecli and mrbob

See plonecli and mr.bob documentation for further information.

8.5 Installing and using it in a buildout

```
[buildout]
parts += mrbob

[mrbob]
recipe = zc.recipe.egg
eggs =
    mr.bob
    bobtemplates.plone
```

This creates a mrbob-executable in your bin-directory. Call it from the `src`-directory of your Plone project like this.

```
../bin/mrbob -O collective.foo bobtemplates:addon
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search