ACRcms Documentation

Release 0.3.7

Axant

Aug 22, 2017

Contents

ACRCms is an open source Python web content management system based on Turbogears and libacr.

ACRCms is designed to let non technical people create and manage their web pages and thanks to its plugin system and libacr it is also easily extensible and embeddable in any web site using the Turbogears framework.

CHAPTER 1

Getting Started

Let's start using ACRCms

Introduction

Creating a website with ACRCms is easy and painless. In this tutorial, you will learn how to:

- *Log in*
- *Insert your first content*
- *Edit a content*
- *Add a cool image*
- *Insert a community driven comments thread*
- *Create a new page*
- *Add a fully functional search bar*
- *Breadcrumbs*
- *Contact Form*

Log In

Visit the url `yourwebpage/login`, a login mask will appear. After the installation the default credentials are:

User: manager

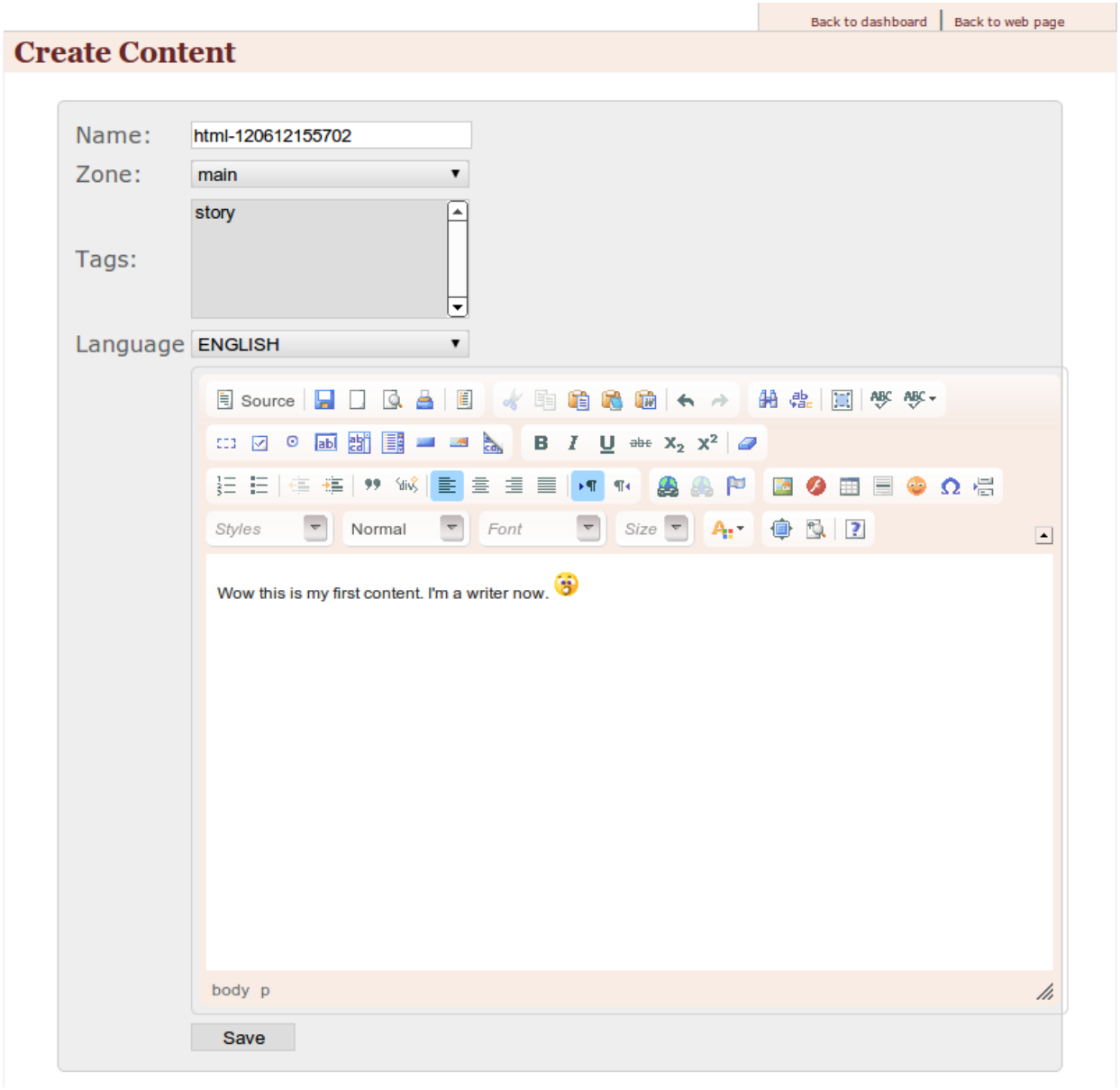
Password: managepass

When you are logged in, at top of the page appears the topbar and over every content appears an edit bar. If the edit bars bother you, you can disable them clicking on TOGGLE EDIT BARS.

Insert your first content

The easiest way to add a custom content to the page is the html slice. Click on **ADD SLICE** in the top bar, and then on **HTML**.

The html edit box will appear. For now the only field that interests us is the rick WYSIWYG editor, let's write something interesting:



save it and take a look to the web page. The content is there. Simple, isn't it?

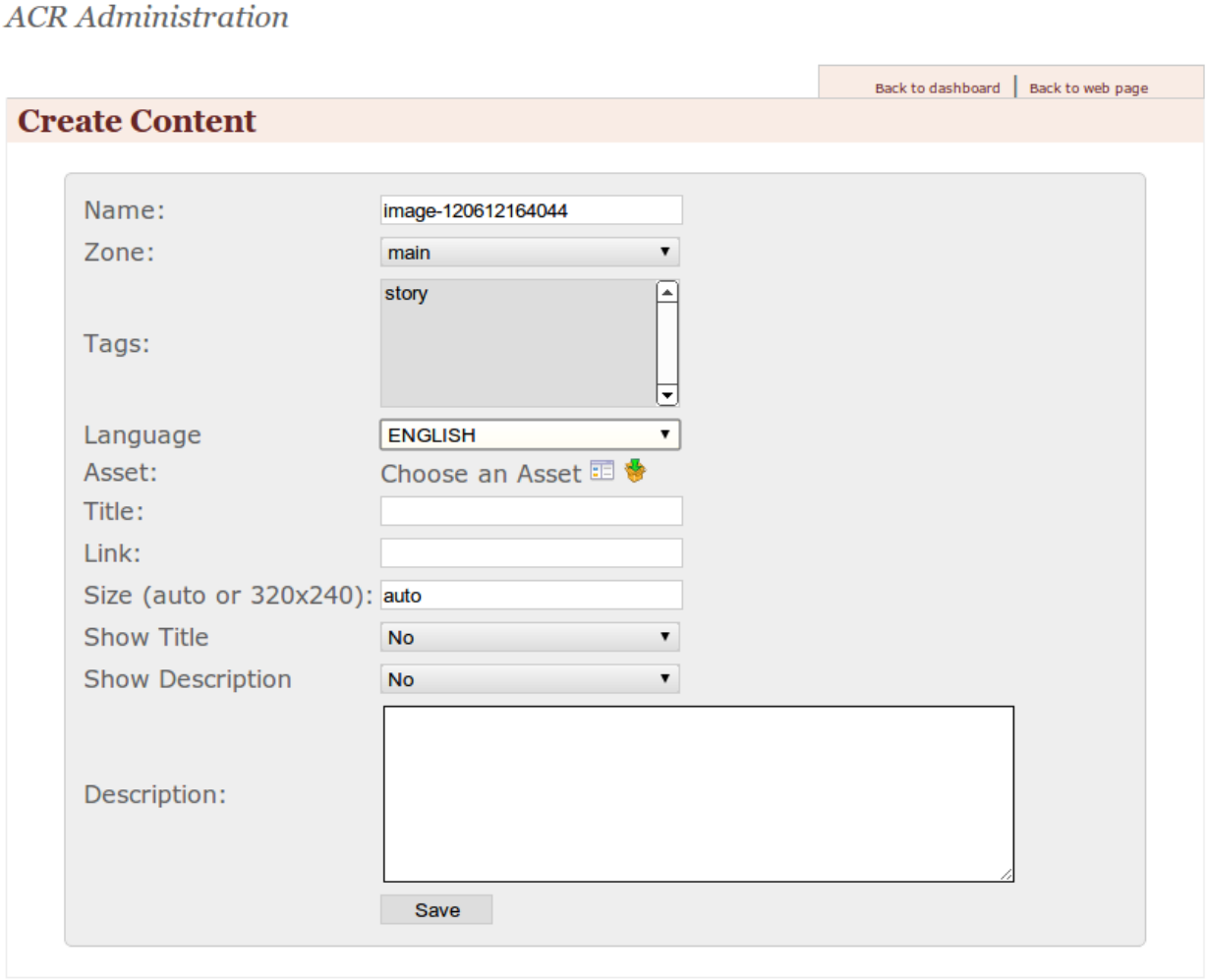
Edit a content

Editing a content is as simple as inserting a new one, just click the edit button in the edit bar over the content you want to modify. As you can see, is now appeared the versions box. What the hell is that? Well, it's one of the most powerful features of ACRCms: every time you edit a content, the previous version remains stored in the database and you can

roll it back whenever you want. Ok, so, click on edit in the versions and translations box, edit a content and then save it. And, then try to click on revert on a previous version. It's magic™.

Add a cool image

Add an image slice (**ADD SLICE -> IMAGE**). The box that appears is the ACRCms asset chooser.



the usage is pretty simple, just select an already uploaded asset from the assets browser, or upload a new image, return to the asset chooser and select it.

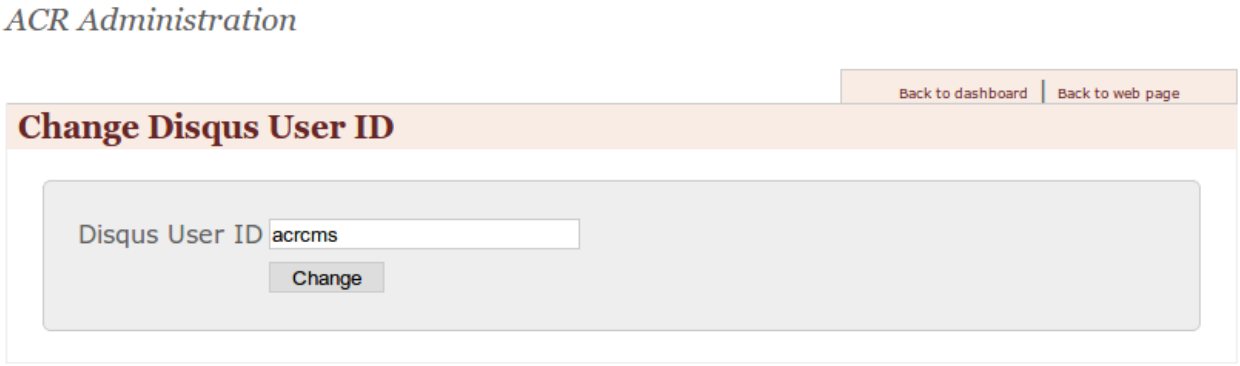
Fill the title field, insert a web link if want to open a new web page on image click, insert a specific dimension (*auto* will preserve the actual image size) and provide a description (facoltative).

Save the slice, and your image is displayed on the bottom of the page.

Insert a community driven comments thread

Ready to face the community? So you can add a comment thread, provided by the disqus plugin. A preliminary step is required only the first time you add a comment thread: register yourself to disqus.com, from the disqus dashboard, add a website, fill the url of your ACRCms website and add the site name. Return back to ACRCms, click on ADMIN in the topbar, and

then click on Disqus icon (general setting panel). Insert the website name in the User ID field and click change.



Now return back to your site page, click on **ADD SLICE -> DISQUS THREAD**, don't change any field and click save. Ok you have your comments thread.

Create a new page

When your site has grown up, you need a new page to accomodate your contents. It's a trivial task: click on **ADMIN -> CREATE PAGE**.

On the page create box, you have to select the page parent. Just remember a couple of things:

1. if you select `----------` as parent, the page will be on the root of the website
2. if you select `Default page for global layout` as parent, the page will be hidden
3. *Any other choice* as parent, the page will be a child of the page selected

For this tutorial, we are going to select `------------`, insert test as title and url and save. You will be redirected to the newly created page and a new menu entry for that page is automagically added on the top menu.

Add a fully functional search bar

Almost any content-centric website needs a search bar. In ACRCms is simple to add one, click on **ADD SLICE -> SEARCH**, save and the search bar will be placed in your page.

Add a breadcrumb bar

Using Genshi slices, you can create contents with more logic. We are going to create a breadcrumb bar using a Genshi slice:

click on **ADD SLICE -> GENSHI**, insert this content:

```
<?python
nodes = []
curpage = page
while curpage:
   if curpage.slices:
       nodes.append(curpage)
   curpage = curpage.parent
nodes = list(reversed(nodes))
?>
```

```
<div class="breadcrumb">
<span class="breadcrumb_home">
<a href="/">HOME</a> &gt;
</span>
<span py:for="bcpage in nodes[:-1]">
<a href="${bcpage.url}">${Markup(bcpage.i18n_title)}</a> &gt;
</span>
<span>${Markup(page.i18n_title)}</span>
</div>
```

Save it and you have a breadcrumb bar. For details please read *Genshi Slice Reference*.

Add a contact form

Add a new form slice **ADD SLICE -> FORM**, fill the destination email address and the email subject, then in the fields text box insert fields description:

```
Name=text
Content=textarea
Sex=[Male, Female]
```

click submit and you will have a form like this:

CHAPTER 2

User Reference

Contents

In ACR every piece of content is called **SLICE**.

To add a content, click on **ADD SLICE** in the top bar. You will see the list of available slices. The available slices depends on installed plugins. Clicking on a slice type, the create page will appear. Mind that the new slice will be created on the page where you were when you clicked on the slice type.

Common Slice Fields

Every slice type edit page has these standard fields:

- `Name` the name of the slice, for your reference
- `Zone` the page zone where the content will appear
- `Tags` the content tags of the content
- `Language` the language of the content, usefull for internationalize of your website.

Standard Slices

Html

HTML slice contains only a rich badass WYSIWYG editor, where you can freely insert any kind of content. The image insert uses ACRcms's assets manager to store and retrieve your contents. Please refer to the CKEditor user guide for the complete how to.

Genshi

GENSHI slice permits you to add Genshi Template Directives, such as conditions, loops, etc... to your content.

Consider this snippet of Genshi code:

```
<div py:for="i in range(7)">
  this is cool :)
</div>
```

it will return something like this:

```
this is cool :)
this is cool :)
this is cool :)
this is cool :)
this is cool :)
this is cool :)
this is cool :)
```

ACRCms injects some useful variable in Genshi slice:

```
acr.slices_with_tag(tag): returns all the slices with the given tag
acr.page_from_urllist([urls]): given the url hierarchy, returns the page object
acr.preview_slice(page, slice): renders the preview of a slice
acr.render_slice(page, slice): renders a slice
acr.url(url, **params): returns the project relative encoded url
acr.request : the current WebOb request
acr.depot : a key-value storage that can be used to store or retrieve data

page.url: the url of the current page
page.i18n_title: the internationalized title of the current page
page.ancestors: iterable of the current page ancestors
page.children: iterable of the current page children

slice.tags: iterable of the tags associated with the current slice
slice.name: the name of the current slice
```

You can also access all the *helpers* using the variable `h`.

Menu

Autogenerated menu, you only have to select if you want an horizontal or a vertical menu, and the root of the menu.

Link

An external html link.

Ajax

Ajax loaded content.

Rss

rss aggregator, add the rss feed uri and eventually a tag filter and it magically works.

Comments thread

Deprecated, left only for retrocompatibility. Please use *Disqus thread*.

Twitter rss

What the hell, Twitter supports only Atom feeds :)

File

Insert a downloadable file using the asset manager. Choose an asset using the assets box, or upload a new one. For further reference, please read ACRcms's assets manager.

Form

An automatic submission form creator. The syntax is pretty simple, `fieldname=fieldtype`, one for row. Available field types:

- `text` single line text field
- `textarea` multiline text area
- `[]` select field, comma separated values inside square brackets

A little example:

```
Name=text
Content=textarea
Sex=[Male, Female]
```

will be rendered as:

Image

Insert an image using the asset manager. Choose an image using the assets box, or upload a new one. For further reference, please read ACRcms's assets manager.

You can also set the image size, an external link and the image title.

Search

Inserts a simple search bar. It's possible to restrict the search on selected tags.

Video

Insert a video using the asset manager. Choose a video file using the assets box, or upload a new one. For further reference, please read ACRcms's assets manager.

You can also set the video size and the video title.

The video will be automatically converted to a web suitable format.

Script

With this slice type you can insert custom JavaScript code in your page to customize your page behavior.

Tag cloud

Tag cloud that aggregates the tags of your slices.

Slicegroup Admin

Adds a bar with two links to add or edit slice group contents.

Disqus thread

Adds a discussion thread to your page using the disqus comment system. You must configure the plugin with your disqus User ID before using it. Refer to Disqus Config for more details.

Map

Google Maps Plugin, you have only to fill the address field and configure the zoom level and the map size.

Blog Articles

The blog articles slice aggregates all the post of a blog category. You must have configured at least one blog category, refer to Blog Config.

Photo Album

A complete photogallery plugin, with categories and thumbnails generator. The use is simple, just add the slice, then select the album and it's done. To add and manage albums refer to Photo Config.

Custom Content

ACR permits to declare new slice types through the `User Defined Views` in the `Advanced` section of the administration dashboard. User defined views permit to bind a set of fields to a genshi script to render them.

Creating Custom Content

The `Add User Definied View` form will ask for:

- `Name` the name of the slice type, will be used by the ADD menu to make possible to create slices of that type.
- `Fields Definition` fields of the content, refer to *field types* for syntax.
- `Code` the content genshi template code which will be used when rendering the slice. Refer to *Genshi content* for more details
- `Preview Template` a genshi template code which will be used when rendering a preview of the slice.

Field Types

The fields of a custom content are defined through the `.ini` syntax, refer to python `ConfigParser` module documentation for more details on the syntax.

Each entry inside the `[fields]` section will define a field, the name of entry will be the field name and the value will be the field type. Valid field types are: `text`, `textarea`, `html`, `file`, `select`. Every field can be defined with `name = type` except for the `select` type which requires a list of valid options in the form `name = select option1, option2, option3`

EXAMPLE:

```
[fields]
name=text
price=text
photo=file
description=textarea
```

The Depot

ACR provides a key-value storage that makes possible to store and retrieve data in collections without the need to create and manage database tables. The `Depot` also provides a way to lookup data, keep in mind that while storing and retrieving data is quite fast, lookup is a costly operation that requires fetching and checking all the available data for the filters you are applying.

The Depot will be available inside *Genshi* and *User Defined Views* as `acr.depot`.

Store/Retrieve/Delete

Basic Depot functions include:

- **`acr.depot.create(collection_name, data)` -> collection_name is the name of the collection and data a** dictionary of strings to store, returns a `Result` object with data and object_id of the newly created object.
- **`acr.depot.get(collection_name, object_id)` -> given a collection and an object_id will return the `Result`** object for the currently stored entry or `None`.
- **`acr.depot.update(collection_name, object_id, data)` -> Updates an existing entry with the given `objec`** setting its fields to every property specified in the `data` dictionary. Properties not specified in the new data dictionary will be kept to the previous value. The update call returns a `Result` object representing the state of the entry before update or `None`.

- **`acr.depot.delete(collection_name, object_id)` -> Deletes an entry from the specified collection, if the actio** is successful a `Result` object with the entry recently deleted is returned.

EXAMPLE:

```
created = acr.depot.create('depot_test', {'path':acr.request.path, 'num':random.
↪randint(1, 10)})
updated = acr.depot.update('depot_test', created.object_id, {'num':created.data['num
↪']+1})
after_update = acr.depot.get('depot_test', updated.object_id)
acr.depot.delete('depot_test', after_update.object_id)
```

Lookup Data

The `Depot` provides the `lookup` method to search data not by object_id. A call to lookup will return a `ResultSet` which provides the `first()`, `all()` and `count()` methods. `ResultSet` objects are also iterable.

Lookup function is defined as `lookup(collection_name, filters)` where filters is a dictionary of data which the entries will be looked for.

EXAMPLE:

```
for i in range(100):
    acr.depot.create('depot_test', {'num':random.randint(1, 10), 'time':time.time()})

first_entry_with_four = acr.depot.lookup('depot_test', {'num':'4'}).first()
print first_entry_with_four.data['time']

for entry in acr.depot.lookup('depot_test', {'num':'8'}):
    print entry.object_id, entry.data['time']
```

Performing Genshi Scripts

ACR provides a way to perform small python scripts and generate pages on the fly using the `genshi` template engine. This can be used to implement small web applications, for more complex functions using plugins is suggested.

To create a script which can later be performed you must go to **Unbound Entities** and press the **Create Genshi Script** button. This will create a new unbound slice (a slice without a page) which can then be performed using `/acr/perform/{slice_name}` url. *Slices bound to a page cannot be performed.*

Refer to *Genshi* for more details about the objects available inside a genshi script.

EXAMPLE:

```
<?python
import random
filter = {}

if acr.request.GET.get('create'):
    created = acr.depot.create('depot_test', {'path':acr.request.path, 'num':random.
↪randint(1, 10)})
elif acr.request.GET.get('delete'):
    acr.depot.delete('depot_test', acr.request.GET['delete'])
elif acr.request.GET.get('search'):
    filter = {'num':acr.request.GET['search']}
?>
```

```
<a href="?create=1">Create</a><br/>
<br/>
<py:for each="entry in acr.depot.lookup('depot_test', filter)">
   <a href="?search=${entry.data['num']}">search</a> ${entry.object_id} - ${entry.
↪data.items()} <a href="?delete=${entry.object_id}">delete</a><br/>
</py:for>
```

CHAPTER 3

Developer Reference

ACRCms installation

Preliminary steps

System Packages Installation

For Ubuntu/Debian systems:

```
$ sudo apt-get install build-essential python-dev python-setuptools python-virtualenv
```

For Fedora systems:

```
$ su -c 'yum install gcc sqlite-devel python-virtualenv'
```

Standard installation

The standard installation pattern provides a full ACRCms instance

Environment setup

Install the environment:

```
$ virtualenv --no-site-packages acrenv
$ cd acrenv/
$ source bin/activate
(acrenv)$ easy_install -i http://tg.gy/222 tg.devtools
(acrenv)$ hg clone ssh://hg@bitbucket.org/axant/acrcms
(acrenv)$ cd acrcms/acr_cms/
(acrenv)$ python setup.py develop
(acrenv)$ paster setup-app development.ini
```

Serve the contents:

```
(acrenv)$ paster serve development.ini --reload
```

Enable libacr functionalities in any TG project

Instead of creating a brand new project, you can add the cms funcionalities provided by libacr in any TurboGears2 project.

Install libacr:

```
(yourenv)$ easy_install libacr
```

Enable libacr in your project.

Add this lines to `yourproject/lib/helpers.py`:

```
import libacr
from libacr.helpers import *

icons = {}
icons.update(libacr.helpers.icons)
```

Add this lines to `yourproject/lib/base.py`:

```
from libacr.lib import *

#[in the __call__ method]
full_acr_js.inject()
acr_css.inject()
```

Add this lines to `yourproject/config/app_cfg.py`:

```
try:
    from tgext.pluggable import plug
    base_config.acr_pluggable_enabled = True
except ImportError:
    base_config.acr_pluggable_enabled = False

if base_config.acr_pluggable_enabled:
    try:
        plug(base_config, 'photos')
    except ImportError:
        pass

    try:
        plug(base_config, 'smallpress')
    except ImportError:
        pass
```

Add this line to `yourproject/model/__init__.py`:

```
from libacr.model import init_acr_model
Content, ContentData, Tag, Page, Slice, View, Comment = init_acr_model(DBSession,␣
↪DeclarativeBase, User, Group)
```

Add this line to `yourproject/controllers/root.py`:

```
from libacr.controllers.cms import AcrRootController

#[in the RootController class]
acr = AcrRootController()
```

Deploy ACRCms on Red Hat's OpenShift PaaS

Get up and running with a ACRCms instance on OpenShift using acrcms-openshift-quickstart. It automatically handles creating a Python virtualenv, populating a MySQL database, and deploying the cms to the cloud.

Features

- Completely free, thanks to Red Hat's OpenShift Express
- MySQL database automatically setup
- Dynamic database configuration at runtime. No passwords stored in your configs.
- Automatic deployment upon git push
- No need to think about servers, let alone apache/mod_wsgi configuration

How To

- Create an account at http://openshift.redhat.com/
- Add a namespace to your account:

```
rhc domain create -n <yournamespace> -l your@email.com
```

- Deploy the cms:

```
rhc app create -a ACRCms -t python-2.6 -l your@email.com
rhc app cartridge add -a ACRCms -c mysql-5.1 -l your@email.com
cd ACRCms
git remote add upstream -m master git@bitbucket.org:simock85/acrcms-openshift-
↪quickstart.git
git pull -s recursive -X theirs upstream master
git push
```

Monitoring your logs

```
rhc-tail-files -a ACRCms -l your@email.com
```

Helpers

preview_slice(page, slice):
Renders the preview of the slice.

draw_slice(page, slicename):
Renders the slice selected by slicename.

draw_section(page, sect):
Renders a page section, selected by section name.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search