uniCMS

Release 0.2.4

Giuseppe De Marco

Dec 17, 2021

INTRODUCTION

uniCMS is a Web Application Content Management System developed using **Django Framework**. The project is created by a group of passionate developers who introduces bespoke design and architecture for a next generation CMS.

Features and specs of uniCMS:

- **The default template shipped with:**
    - Compatibility and interoperability in mobile platforms
    - SEO optimized
    - Bootstrap like design and structure
    - Plugin mode and compatibility for Django applications
- Agile and adaptive design and logic (ad-hoc and easy customization)
- **OpenAPIv3** (OAS3) compliant
- Compatible with the major RDBMS engines with agile schema migrations capabilities
- **Multitenancy - create and manage multiple web applications within single platform**
- Search Engine with Query and capabilities on top of **MongoDB FullText Search**
- Extensive localization with **multiple languages**
- Ability to handle Editorial Board workflows (WiP) and permissions by contexts
- High performance thanks to its cached model based on Redis TTL
- Security by design - security by default
- Robust enterprise and scalable
- Plugin model and rich interoperability with multiple frameworks and technologies

uniCMS is designed for both end users and developers where the developers can create their own customzied web applications (CMS) without starting one from scratch and end users without any development skills can setup a professional CMS platform without difficulty.

uniCMS was created due to necessity of creation and design of a new protal for the University of Calabria. After evaluation of several options, University of Calabria having a strong in-house competitive and highly skilled technical team it was decided to opt for the development of a brand new CMS solution based on Django framework.

The entire uniCMS project code is open sourced and therefore licensed under the Apache 2.0.

For any other information please consult the Official Documentation and feel free to contribute the project or open issues.

CHAPTER ONE

INTRODUCTION

For the correct usage of uniCMS we must familiarize ourselves with the following components:

- **Web Sites**, fully qualified domain name
- **Contexts**, WebPaths like `/offices` and `/offices/employees`
- **Page templates**, with a selectable html template file through UI
- **Block templates**, elements that render things within single or multiple pages
- **Navigation Bars**, menu, footers, lists of things that can optionally fetch elements from Publications (titles, body, images . . .) to enrich its items
- **Carousels**, image sliders . . . They are the basic components of the modern web so we decided to customize them as well
- **Pages**, each webpath loads a page that's modular container of the page blocks
- **Publications**, a typical **posts**
- **Handlers**, they intercept HTTP requests providing different perspective of a standard Page Immagine the **List** and **View** resources of a News pertaining to a context (WebPath) or a way to integrate a third party Django app in uniCMS.

1.1 How to start with uniCMS

The simpler and easier way to create a web site in uniCMS consist of the following steps:

1. Select the template to be used. Refer to **Templates** section of this guide
2. Define your **blocks and Page templates** to be inherited by your Website's pages
3. **Create a WebSite** Domain name
4. Fill contents like Categories, Publications, Menus . . .
5. **Create a WebPath**, a root node like '/' or a subdirectory
6. Create a Page with as much as blocks you'd like. Dispose menus, carousels and things with regular blocks or publication contents (or part of them) using placeholder blocks.

1.2 Simple Example

The quickest way to get started with uniCMS is to run a demo platform with a few basic websites, pages and contents.

This project aims to simplify the design and implementation of a typical web portals designed for Universities/Colleges/Academic world. You'll find a simplified generalization of all entities aimed to build a common Content Management System (CMS).

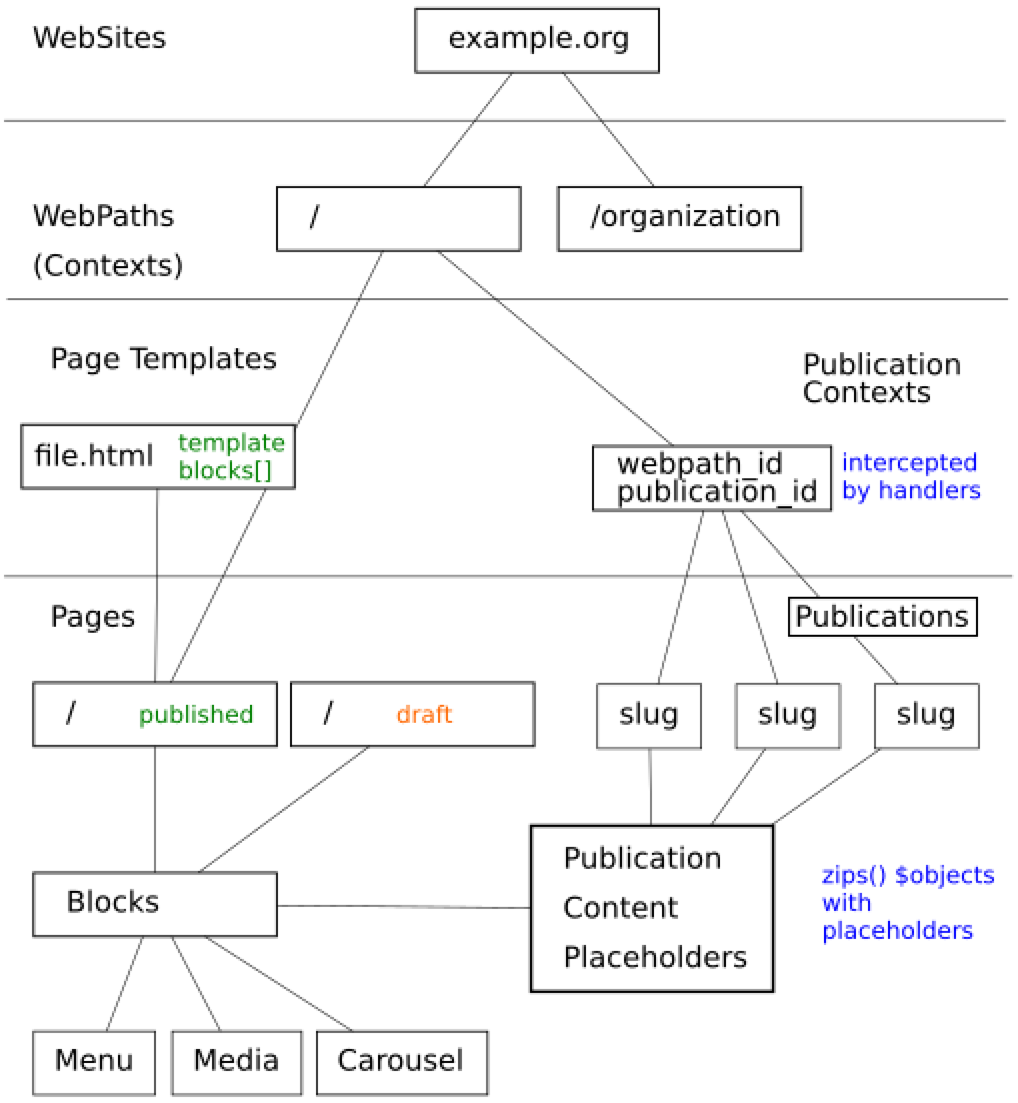
uniCMS Example Project

CHAPTER

TWO

HOW IT WORKS

This section describes which entities and relations composes uniCMS and the handling of HTTP Requests.



2.1 HTTP Requests

HTTP Requests are handled by the native Django view which is **cms.contexts.views.cms_dispatch**. It will:

1. check if a website exists
2. **check if request.get_full_path() matches one of the Handlers loaded in settings.py.**
    - If Yes -> return **handler.as_view()**
    - Else: continue
3. **check if request.get_full_path() matches a published page**
    - If Yes -> return **render(request, page.base_template.template_file, context)**
    - Else: *raise 404()*

2.2 WebPaths

A WebPath is nothing more than a path, such as **/** or **/contacts**, where the first corresponds to a home page and the second to a child path of the first. Child WebPaths are objects that have a relationship with a parent.

In uniCMS a Webpath is also called Context. Contents such as pages, menus, carousels and publications, can refer to one or more contexts at the same time.

Let's think of a templatetag of an HTML block that dynamically loads all the publications pertaining to the context where it is loaded. Moreover, a webpath can also be an alias of another one, or a third party URL. Example:

- **/about-us** could be an alias of **/contacts**
- **/polls** could be a redirect to a polling system, hosted at **https://that.polls.system.org/start**

2.3 NavigationBars and Menus

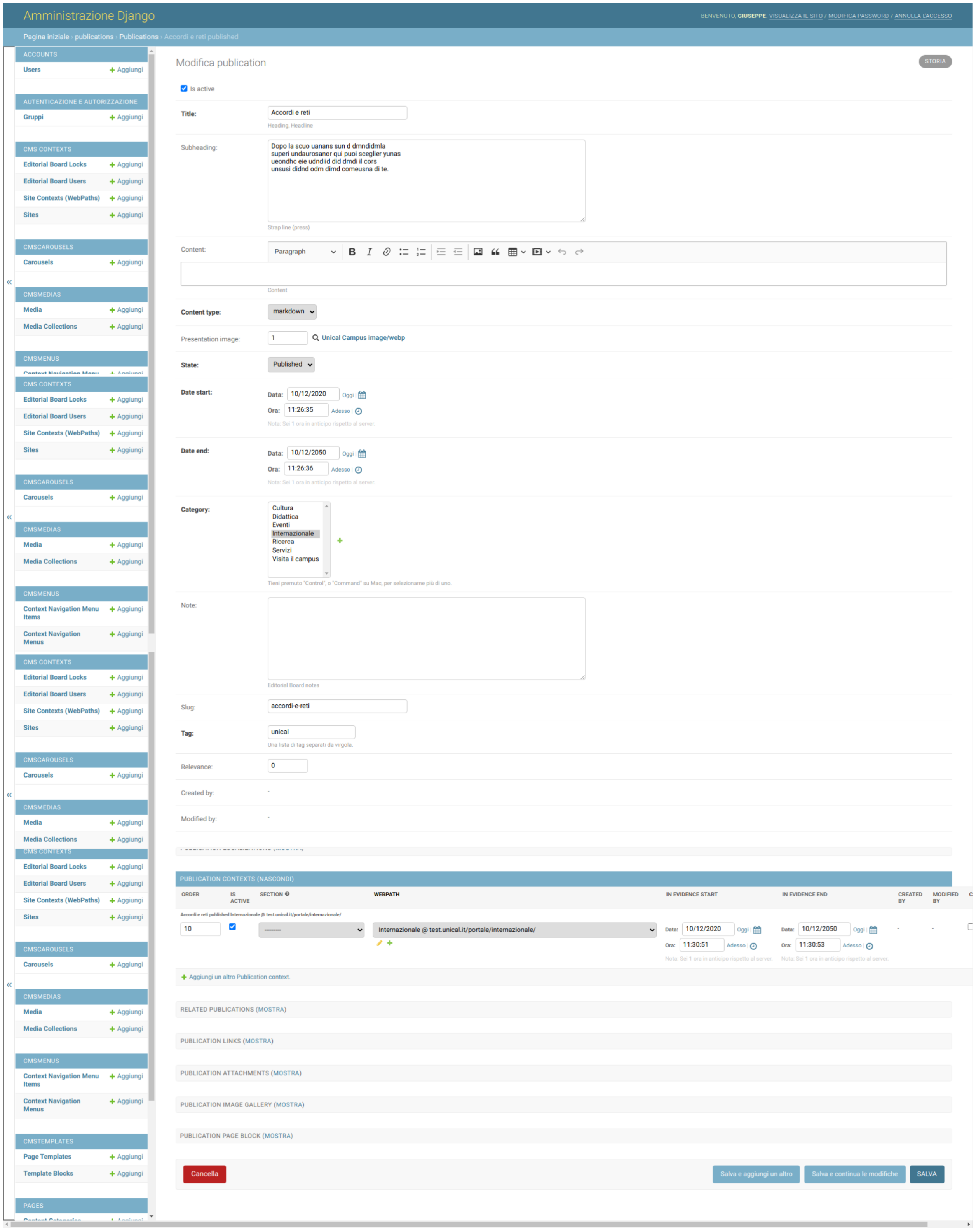
[WiP]

This section describes how to build a Menu.

- Menu object
- MenuItem objects
- How a MenuItem can inherit contents from a publication
- Render an Interactive Menu in a HTML template, reference to uniCMS's' Teamplates documentation

2.4 Publications and Handlers

Publications or Posts are something that are added daily by an Editorial Board.


Amministrazione Django
BENVENUTO, GIUSEPPE. VISUALIZZA IL SITO / MODIFICA PASSWORD / ANNULLA L'ACCESSO
Pagina iniziale › publications › Publications › Accordi e reti published
ACCOUNTS
Users + Aggiungi
AUTENTICAZIONE E AUTORIZZAZIONE
Gruppi + Aggiungi
CMS CONTEXTS
Editorial Board Locks + Aggiungi
Editorial Board Users + Aggiungi
Site Contexts (WebPaths) + Aggiungi
Sites + Aggiungi
CMSCAROUSELS
Carousels + Aggiungi
CMSMEDIAS
Media + Aggiungi
Media Collections + Aggiungi
CMSMENUS
Context Navigation Menu Items + Aggiungi
Context Navigation Menus + Aggiungi
CMSTEMPLATES
Page Templates + Aggiungi
Template Blocks + Aggiungi
PAGES
STORIA
Paragraph
Cancella
Salva e aggiungi un altro
Salva e continua le modifiche
SALVA

Modifica publication

☑ Is active

Title: Accordi e reti
Heading, Headline

Subheading:
Dopo la scuo uanans sun d dmndidmla
superi undaurosanor qui puoi sceglier yunas
ueondhc eie udndiid did dmdi il cors
unsusi didnd odm dimd comeusna di te.
Strap line (press)

Content:
Content

Content type: markdown

Presentation image: 1 Unical Campus image/webp

State: Published

Date start: Data: 10/12/2020 Oggi | Ora: 11:26:35 Adesso |
Nota: Sei 1 ora in anticipo rispetto al server.

Date end: Data: 10/12/2050 Oggi | Ora: 11:26:36 Adesso |
Nota: Sei 1 ora in anticipo rispetto al server.

Category: Cultura, Didattica, Eventi, [✓ Internazionale], Ricerca, Servizi, Visita il campus
Tieni premuto "Control", o "Command" su Mac, per selezionarne più di uno.

Note:
Editorial Board notes

Slug: accordi-e-reti

Tag: unical
Una lista di tag separati da virgola.

Relevance: 0

Created by: -

Modified by: -

PUBLICATION CONTEXTS (NASCONDI)

ORDER	IS ACTIVE	SECTION	WEBPATH	IN EVIDENCE START	IN EVIDENCE END	CREATED BY	MODIFIED BY
Accordi e reti published Internazionale @ test.unical.it/portale/internazionale/							
10	☑	---------	Internazionale @ test.unical.it/portale/internazionale/	Data: 10/12/2020 Oggi | Ora: 11:30:51 Adesso | Nota: Sei 1 ora in anticipo rispetto al server.	Data: 10/12/2050 Oggi | Ora: 11:30:53 Adesso | Nota: Sei 1 ora in anticipo rispetto al server.	-	-

+ Aggiungi un altro Publication context.

RELATED PUBLICATIONS (MOSTRA)

PUBLICATION LINKS (MOSTRA)

PUBLICATION ATTACHMENTS (MOSTRA)

PUBLICATION IMAGE GALLERY (MOSTRA)

PUBLICATION PAGE BLOCK (MOSTRA)

It would publish some news about a specific topic, as it would be similar to a simple/standard Web Blog aimed to perform some additional instructions as below:

- standard or custom template to represent a pubblication on the screen

- breadcrumbs manager that represent a human readable, interactive, webpath
- page with a list of all the posts, also filtered by category

If the concept of publication or post is clear to all audiences who have at least once published/posted in a WebBlog, a small extra effort is required to understand the fact that uniCMS enable us to:

- create a post and decide in which context (WebPath) to be published, in a single or multiple locations (Contexts)
- manage a block, called *publication_preview* for example, that represents a fancy list of all the publications that belongs to that specific webpath

Handlers will show the history of your Publications (**List**) and will let the user browse them (**View**).

2.5 Pages, Blocks and Placeholders

Pages inherit Template Pages to be used as base html template file and optionally a bunch of template blocks. Blocks can be of different type, like the basic one called HTMLBlock which is a Text Field that takes a raw html with django's template statements. This means that in a HTMLBlock we can load template tags and use Django Template filters and statements, as outlined in the Official Django Documentation.

Furthermore, there are specific blocks other than HTMLBlock with django *templatetags* as content. See Example below:

```
{% load unicms_blocks %}
<div class="row negative-mt-5 mb-3" >
    <div class="col-12 col-md-3">
        <div class="section-title-label px-3 py-1">
            <h3>Unical <span class="super-bold">world</span></h3>
        </div>
    </div>
</div>

<div class="row">
    <div class="col-12 col-lg-9">
        {% load_publications_preview template="publications_preview_v3.html" %}
    </div>
    <div class="col-12 col-lg-3">
        {% include "unical_portale_agenda.html" %}
    </div>
</div>
```

A Page Template HTML file would be splitt into several sections, each of them where a Django templatetag called **load_blocks** will fill the contents. See below Example:

```
<!-- Breadcrumbs -->
{% block breadcrumbs %}
    {% load_blocks section="breadcrumbs" %}
{% endblock breadcrumbs %}
<!-- end Breadcrumbs -->
```

Placeholders are different type of blocks. We have, for instance, **PublicationPlaceholderBlock** which is a block that will be filled by relative publication to the page it belongs to. Let's suppose to distribute 4 publication placeholders in a page and we link them to the same page. As a result we'll have each publication rendered in the Handler Block in orderly fashion and their positionings (section).

index	block type	publication
0	pub placeholder	the first ordered by “order”
1	pub placeholder	the second ordered by “order”
2	pub placeholder	the third ordered by “order”

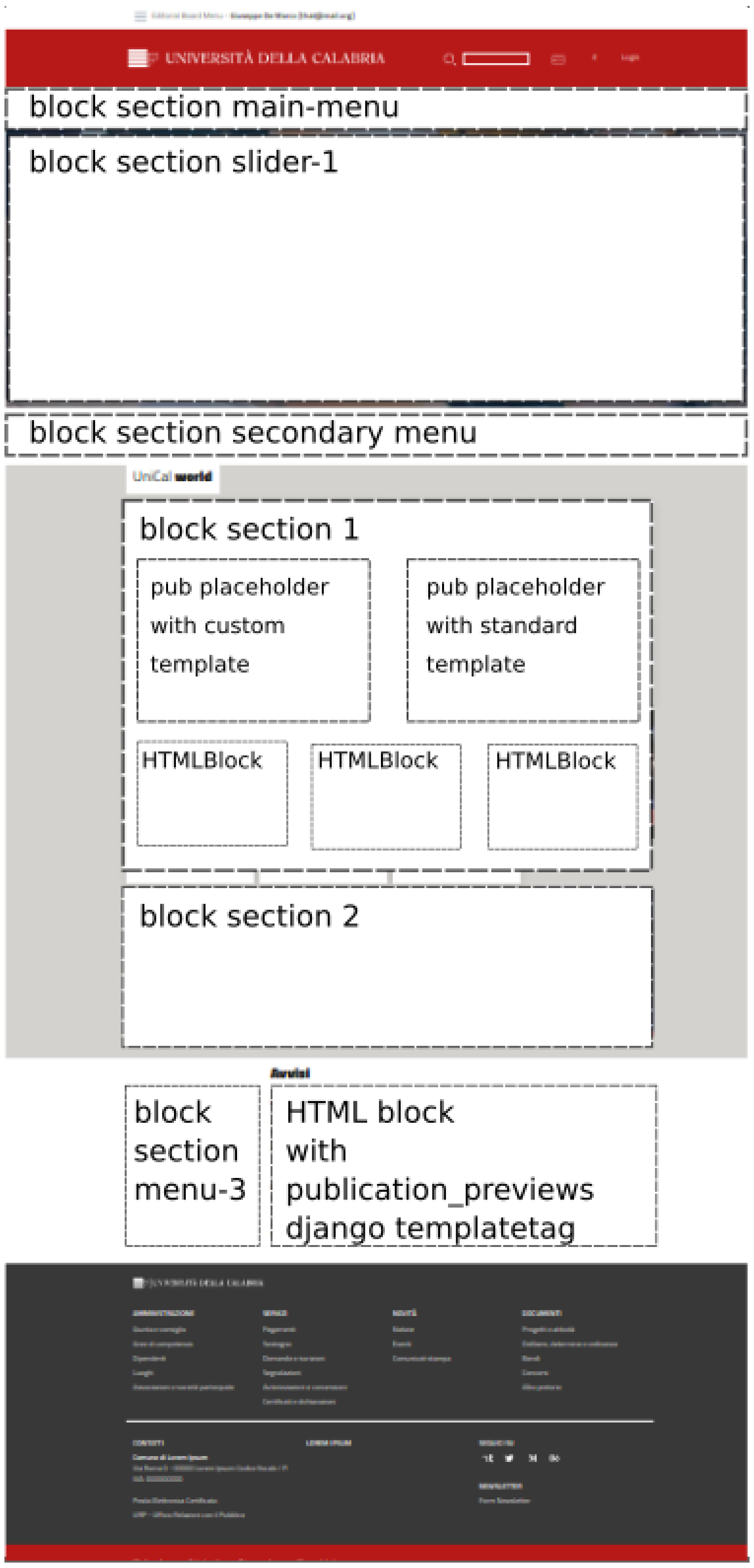
A PublicationPlaceHolder would take also a specific template to allow users to integrate their own styles, ways of representations of the contents given the publication. For example a template that takes a publication object in input will decide how and what to render: the title, subheading, main body content, related objects and so on…

The first placeholder will render the first content following the second one in sequence and so on. This model allows single page template designer to arrange placeholders without worrying about the representation of the content. The page that will inherit the uniCMS template will define which publications to import, which web links to handle and so on. Take as simple example the management of the Home Page, where each content is selectively chosen by publishers.

A page can have the following child elements:

- PAGE NAVIGATION BARS
- PAGE CAROUSELS
- PAGE BLOCKS, extends or disable those inherited from the Page Template
- PUBLICATION CONTENTS
- RELATED PAGES
- RELATED LINKS

This is a simplified page divided by sections that would show how the contents can be distribuited in a Page Template.


UNIVERSITÀ DELLA CALABRIA
block section main-menu
block section slider-1
block section secondary menu
UniCal world
block section 1
pub placeholder with custom template
pub placeholder with standard template
HTMLBlock
HTMLBlock
HTMLBlock
block section 2
Avvisi
block section menu-3
HTML block with publication_previews django templatetag

CHAPTER

THREE

SETUP

3.1 Prepare Environment and Install Requirements

```
apt install python3-pip
pip3 install virtualenv
mkdir unicms_project && cd "$_"
virtualenv -ppython3 env
source env/bin/activate

pip install unicms
```

To complete the installation make sure you have correctly loaded unicms modules to your project settings file.

```
INSTALLED_APPS = [
    'accounts',

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # templates - you can load your own templates
    'sass_processor',
    'bootstrap_italia_template',
    'unicms_template_italia', # for example
    'unicms_template_unical', # for example

    # unicms
    'nested_admin', # for admin CRUD
    'taggit', # page and publication tags
    'taggit-serializer', # taggit tags serializer

    'cms.templates',
    'cms.contexts',
    'cms.carousels',
    'cms.menus',
    'cms.medias',
    'cms.pages',
```

(continues on next page)

(continued from previous page)

```
    'cms.publications',
    'cms.api',
    'cms.search',

    # django rest
    'rest_framework' # api
    'django_filters', # api filters

    # editorial board app
    'unicms_editorial_board', # for example

]
```

3.2 Getting Started

You can start the project/examples available in uniCMS repository as follow.

```
git clone https://github.com/UniversitaDellaCalabria/uniCMS.git
cd uniCMS/example
```

Prepare Database and Preload example data

```
./manage.py migrate

# install your templates in settings.INSTALLED_APPS and then create CMS template␣
↪symbolic links
./manage.py unicms_collect_templates

# if you want to load some example datas
./manage.py loaddata ../dumps/cms.json

./manage.py createsuperuser
./manage.py runserver
```

Go to `/admin` and submit superuser credentials to start.

3.3 URLs

uniCMS URLs are fully managed with `cms.context` via admin interface. This feature enable users to load/import third-party django applications. It's important to keep in mind that the user should configure django application URLs before defining uniCMS's own URLs. Otherwise uniCMS will intercept those parameters and there is a good chance that the user will hit 404 page. The user can set the environment variable `CMS_PATH_PREFIX` to a desidered path, eg: `portale/`, to restrict uniCMS URL matching to specified root path.

Here is an example of project urls.py

```
if 'cms.contexts' in settings.INSTALLED_APPS:
    urlpatterns += path('',
                        include(('cms.contexts.urls', 'cms'),
```

(continues on next page)

(continued from previous page)

```
                                namespace="unicms"),
                        name="unicms"),

if 'cms.api' in settings.INSTALLED_APPS:
    urlpatterns += path('',
                        include(('cms.api.urls', 'cms'),
                                namespace="unicms_api"),
                        name="unicms_api"),


if 'cms.search' in settings.INSTALLED_APPS:
    urlpatterns += path('',
                        include(('cms.search.urls', 'cms_search'),
                                namespace="unicms_search"),
                        name="unicms_search"),
```

URLs that match the namespace within configuration in the `urls.py` of the the master project will be handled by uniCMS. uniCMS can match two type of resources:

1. WebPath (Context) corresponsing to a single Page (Home page and associated pages)
2. Application Handlers, a typical example would be the Pubblication List and the View resources

for the latter, uniCMS uses some reserved keywords as prefix to specific URL routings. These configurations are typically stored in settings file. See the following *Handlers* for instance.

See `cms.contexts.settings` as example. See `cms.contexts.views.cms_dispatcher` to figure how an HTTP request is intercepted and handled by uniCMS to establish if either to use a Handler or a Standard Page as response.

3.4 Settings

uniCMS by default have standard settings for its applications, in their respective settings.py file, as shown in the examples `cms/pages/settings.py` and `cms/contexts/settings.py`. Each of these parameters declared can be added to your project general (global) `settings.py` file.

uniCMS parameters are the followings.

3.4.1 Editorial Board Permissions

```
CMS_CONTEXT_PERMISSIONS = (
                           (0, _('disable permissions in context')),

                           (1, _('can translate content in their own context')),
                           (2, _('can translate content in their own context and␣
↪descendants')),

                           (3, _('can edit content created by them in their own context
↪')),
                           (4, _('can edit content in their own context')),
                           (5, _('can edit content in their own context and descendants
↪')),
```

(continues on next page)

(continued from previous page)

```
                           (6, _('can publish content in their own context')),
                           (7, _('can publish content in their own context and␣
↪descendants')),
                          )
```

3.4.2 Media

```
CMS_IMAGE_CATEGORY_SIZE = 128
CMS_IMAGE_THUMBSIZE = 128

# file validation
FILETYPE_PDF = ('application/pdf',)
FILETYPE_DATA = ('text/csv', 'application/json',
                 'application/vnd.ms-excel',
                 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',
                 'application/vnd.oasis.opendocument.spreadsheet',
                 'application/wps-office.xls',
                 )
FILETYPE_TEXT = ('text/plain',
                 'application/vnd.oasis.opendocument.text',
                 'application/msword',
                 'application/vnd.openxmlformats-officedocument.wordprocessingml.document
↪',
                )
FILETYPE_IMAGE = ('image/webp', 'image/jpeg', 'image/png',
                  'image/gif', 'image/x-ms-bmp')
FILETYPE_P7M = ('application/pkcs7-mime',)
FILETYPE_SIGNED = FILETYPE_PDF + FILETYPE_P7M
FILETYPE_ALLOWED = FILETYPE_TEXT + FILETYPE_DATA + FILETYPE_IMAGE + FILETYPE_SIGNED

# maximum permitted filename lengh in attachments, uploads
FILE_NAME_MAX_LEN = 128

FILE_MAX_SIZE = 5242880
```

3.4.3 Publications

```
# as per documentation reference or default
CMS_PUBLICATION_VIEW_PREFIX_PATH = 'contents/news/view/'
CMS_PUBLICATION_LIST_PREFIX_PATH = 'contents/news/list'

CMS_PUBLICATION_URL_LIST_REGEXP = f'^(?P<webpath>[\/a-zA-Z0-9\.\-\_]*)({CMS_PUBLICATION_
↪LIST_PREFIX_PATH})/?$'
CMS_PUBLICATION_URL_VIEW_REGEXP = f'^(?P<webpath>[\/a-zA-Z0-9\.\-\_]*)({CMS_PUBLICATION_
↪VIEW_PREFIX_PATH})(?P<slug>[a-z0-9\-]*)'
CMS_APP_REGEXP_URLPATHS = {
    'cms.handlers.PublicationViewHandler' : CMS_PUBLICATION_URL_VIEW_REGEXP,
    'cms.handlers.PublicationListHandler' : CMS_PUBLICATION_URL_LIST_REGEXP,
}
```

(continues on next page)

(continued from previous page)

```
CMS_HANDLERS_PATHS = [CMS_PUBLICATION_VIEW_PREFIX_PATH,
                      CMS_PUBLICATION_LIST_PREFIX_PATH]

# content paginator
CMS_PAGE_SIZE = 3

CMS_HOOKS = {
    'Publication': {
        'PRESAVE': [],
        'POSTSAVE': ['cms.search.hooks.publication_se_insert',],
        'PREDELETE': ['cms.search.hooks.searchengine_entry_remove',],
        'POSTDELETE': []
    },
    'Page': {
        'PRESAVE': [],
        'POSTSAVE': ['cms.search.hooks.page_se_insert',],
        'PREDELETE': ['cms.search.hooks.searchengine_entry_remove',],
        'POSTDELETE': []
    }
}
```

3.4.4 Templates

```
# see unicms-templates
CMS_TEMPLATE_BLOCK_SECTIONS =

CMS_BLOCK_TYPES = (
                   ('cms.templates.blocks.HtmlBlock', 'HTML Block'),
                   ('cms.templates.blocks.JSONBlock', 'JSON Block'),
                   ('cms.templates.blocks.CarouselPlaceholderBlock', 'Carousel␣
↪Placeholder Block'),
                   ('cms.templates.blocks.LinkPlaceholderBlock', 'Link Placeholder Block
↪'),
                   ('cms.templates.blocks.PublicationContentPlaceholderBlock',
↪'Publication Content Placeholder Block'),
)

CMS_TEMPLATES_FOLDER = f'{BASE_DIR}/templates/unicms'
CMS_BLOCK_TEMPLATES = []
CMS_PAGE_TEMPLATES = []


CMS_LINKS_LABELS = (('view', _('View')),
                    ('open', _('Open')),
                    ('read more', _('Read More')),
                    ('more', _('More')),
                    ('get in', _('Get in')),
                    ('enter', _('Enter')),
                    ('submit', _('Submit')),
```

(continues on next page)

(continued from previous page)

```
                    ('custom', _('custom'))
                  )
```

3.4.5 Redis (Cache)

uniCMS can cache HTTP responses based on relevant parameters outlined below:

```
################
# Django config
################

CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://10.0.3.89:6379/unicms",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "COMPRESSOR": "django_redis.compressors.zlib.ZlibCompressor",
            # improve resilience
            "IGNORE_EXCEPTIONS": True,
            "SOCKET_CONNECT_TIMEOUT": 2,  # seconds
            "SOCKET_TIMEOUT": 2,  # seconds
        }
    }
}
DJANGO_REDIS_LOG_IGNORED_EXCEPTIONS = True

######################
# Redis uniCMS config
######################

CMS_CACHE_ENABLED = True

CMS_CACHE_KEY_PREFIX = 'unicms_'
# in seconds
CMS_CACHE_TTL = 25
# set to 0 means infinite
CMS_CACHE_MAX_ENTRIES = 0
# request.get_raw_uri() that matches the following would be ignored by cache ...
CMS_CACHE_EXCLUDED_MATCHES =  ['/search?',]
```

3.4.6 MongoDB (Search Engine)

uniCMS default search engine is built on top of mongodb. Install and configure mongodb

```
apt install -y gnupg wget
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/4.4 main" | sudo tee /
↪etc/apt/sources.list.d/mongodb-org-4.4.list
apt update
apt install -y mongodb-org

systemctl daemon-reload
systemctl enable mongod
systemctl start mongod
```

Create your default users, using mongo CLI as follow:

```
use admin
db.createUser(
  {
    user: "admin",
    pwd: "thatpassword"
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]
  }
)

use unicms
db.createUser(
  {
    user: "unicms",
    pwd:  "thatpassword",
    roles: [{ role: "readWrite", db: "unicms" }]
  }
)

db.createUser(
  {
    user: "unicms_search",
    pwd:  "thatpassword",
    roles: [{ role: "read", db: "unicms" }]
  }
)

exit
```

Configure connection and default settings in settings.py

```
MONGO_URL = 'mongodb://10.0.3.217:27017'
MONGO_CONNECTION_PARAMS = dict(username='admin',
                               password='thatpassword',
                               connectTimeoutMS=5000,
                               socketTimeoutMS=5000,
                               serverSelectionTimeoutMS=5000)
MONGO_DB_NAME = 'unicms'
```

(continues on next page)

(continued from previous page)

```
MONGO_COLLECTION_NAME = 'search'
MODEL_TO_MONGO_MAP = {
    'cms.pages.Page': 'cms.search.models.page_to_entry',
    'cms.publications.Publication': 'cms.search.models.publication_to_entry'
}

SEARCH_ELEMENTS_IN_PAGE = 25
```

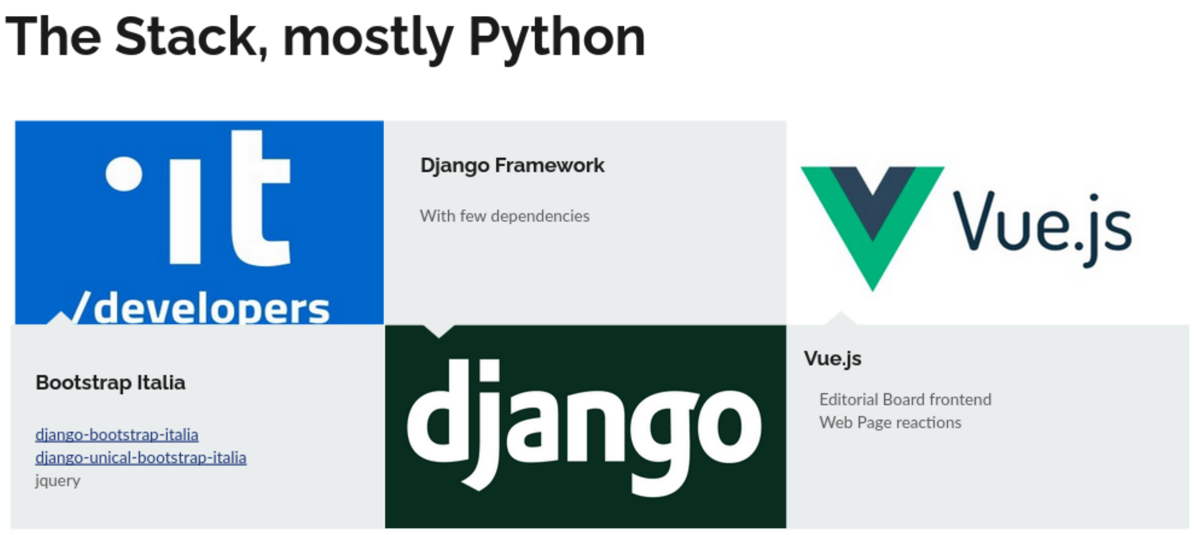
Create your fulltext indexes with the help of **cms.search** CLI. Remember that default_language is set to italian. Do the following to set your own language:

```
./manage.py cms_search_create_mongo_index -default_language english
```

CHAPTER

FOUR

TEMPLATES



The following templates are the ones currently supported:

- Bootstrap Italia Design
- Unical

CHAPTER

FIVE

SEARCH ENGINE

uniCMS is shipped with MongoDB used as search engine and the below are some of main reasons of this choice:

- Relatively small size/amount of documents stored, few kilobytes (BSON storage)
- Collections would be populated on creation/modification/deletion events by **on_$event hooks**
- Each entry is composed following a small schema, this would reduce storage usage and increase general performances at the same time

Technical specifications are available in MongoDB Official Documentation. Some usage examples are also have been posted here.

A sample document looks like (see `cms.search.models`)

```
entry = {
            "title": "Papiri, Codex, Libri. La attraverso labora lorem ipsum",
            "heading": "Itaque earum rerum hic tenetur a sapiente delectus, ut aut␣
↪reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores␣
↪repellat.",
            "content_type": "cms.publications.Publication",
            "content_id": "1",
            "image": "/media/medias/2020/test_news_1.jpg",
            "content": "<p>Sed ut perspiciatis unde omnis iste natus error sit␣
↪voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab␣
↪illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo␣
↪enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia␣
↪consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro␣
↪quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed␣
↪quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat␣
↪voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis␣
↪suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum␣
↪iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur,␣
↪vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?</p><p>&lt;h1&gt;This␣
↪HTML is escaped by default!&lt;/h1&gt;</p><p>&nbsp;</p>",
            "sites": [
                "test.unical.it"
            ],
            "urls": [
                "//test.unical.it/portale/dipartimenti/dimes/contents/news/view/unical-
↪campus-1",
                "//test.unical.it/portale/contents/news/view/unical-campus-1"
            ],
            "tags": [],
```

(continues on next page)

(continued from previous page)

```
            "categories": [
                "Didattica"
            ],
            "indexed": "2020-12-09T15:00:18.151000",
            "published": "2020-11-09T13:24:35",
            "viewed": 0,
            "relevance": 0.5714285714285714,
            "language": "italian",
            "translations": [
                {
                    "language": "english",
                    "title": "gdfg",
                    "subheading": "dfgdfgdf",
                    "content": "<p>dfgdfgdfg</p>"
                }
            ],
            "day": 9,
            "month": 11,
            "year": 2020
        },
```

5.1 Search Engine CLI

Publications and Page models comes automatically configured by some of default save_hooks such as the search engine indexers. Search Engine indexes can be rebuilt via command line interface (SE cli):

```
# show all the publications of the first November 2020
./manage.py cms_search_content_sync -type cmspublications.Publication -d 1 -y 2020 -m 11␣
↪-show

# Purge all the entries, renew and finally show them
./manage.py cms_search_content_sync -y 2020 -type cmspages.Page -purge -insert -show

# purge all the publications published in year 2020
./manage.py cms_search_content_sync -type cmspublications.Publication  -purge -y 2020

# clean up all the Publications posted in December 2020
./manage.py cms_search_content_sync -type cmspublications.Publication -m 12 -y 2020 -
↪purge -insert
```

`cms_search_content_sync` rely on `settings.MODEL_TO_MONGO_MAP` that defines which functions are involved respectively for each Model Type.

```
MODEL_TO_MONGO_MAP = {
    'cmspages.Page': 'cms.search.models.page_to_entry',
    'cmspublications.Publication': 'cms.search.models.publication_to_entry'
}
```

5.2 Search Engine Behavior

Let’s suppose we are searching the following words based on our previous entries.

The matching words:

- “my blog”
- “than reality”
- “rien la reliti”
- “my!”

Not matching words:

- ‘rien -”de plus”’
- ‘”my!”’
- ‘-nothing’

As we can see symbols like + and – represent the inclusion or exlcusion of the words. Specifying “bunch of words” will match the entire sequence.

CHAPTER

SIX

UNICMS COMPONENTS

6.1 Permissions

```
CMS_CONTEXT_PERMISSIONS = (
                            (0, _('disable permissions in context')),

                            (1, _('can translate content in their own context')),
                            (2, _('can translate content in their own context and␣
↪descendants')),

                            (3, _('can edit content created by them in their own context
↪')),
                            (4, _('can edit content in their own context')),
                            (5, _('can edit content in their own context and descendants
↪')),

                            (6, _('can publish content in their own context')),
                            (7, _('can publish content in their own context and␣
↪descendants')),
                           )
```

6.2 i18n

Menus, *Carousels*, *Publications* and *Categories* can also be localized in a single or multiple languages via Web Backend. If for instance a client browser have a Spanish localization the rendering system will render all the spanish localized block, if it is present otherwise it will switch to default language.

All the gettext values defined in our static HTML template will be handled the same way django localization does.

6.3 Page Blocks

A configurable object that would be rendered in a specified section of the page (as defined in the base template). It can take a long Text input as content, a json object or whatever given in input depending the Block Type. Examples:

- Native HTML renderer
- Customized Block element that take a json object in input for its object constructor

The following description covers some of HTML blocks. As we can see the HTML blocks in uniCMS is fully supported by Django templatetags and the native Django template context.

Load Image slider (Carousel) configured for the Page

```
{% load unicms_carousels %}
{% load_carousel section='slider' template="unical_portale_hero.html" %}

<script>
$(document).ready(function() {
  $("#my-slider").owlCarousel({
      navigation : true, // Show next and prev buttons
      loop: true,
      slideSpeed : 300,
      paginationSpeed : 400,
      autoplay: true,
      items : 1,
      itemsDesktop : false,
      itemsDesktopSmall : false,
      itemsTablet: false,
      itemsMobile : false,
      dots: false
  });
});
</script>
```

Load Publication preview in a Page The load_publications_preview templatetag is widely used. This template tag loads all of the pubblication and associated stuff to its WebPath (CMS Context) of the Page.

```
{% load unicms_blocks %}
    <div class="row negative-mt-5 mb-3" >
        <div class="col-12 col-md-3">
            <div class="section-title-label px-3 py-1">
                <h3>Unical <span class="super-bold">world</span></h3>
            </div>
        </div>
    </div>

    <div class="row">
        <div class="col-12 col-lg-9">
            {% load_publications_preview template="publications_preview_v3.html" %}
        </div>
        <div class="col-12 col-lg-3">
            {% include "unical_portale_agenda.html" %}
        </div>
    </div>
```

Youtube iframes As simple as possibile, that bunch of HTML lines.

```
<div class="row">
<div class="col-12 col-md-6">
<iframe width="100%" height="315" src="https://www.youtube.com/embed/ArpMSujC8mM"␣
↪frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media;␣
↪gyroscope; picture-in-picture" allowfullscreen></iframe>
</div>
 <div class="col-12 col-md-6">
<iframe width="100%" height="315" src="https://www.youtube.com/embed/xrjjJGqZpcU"␣
↪frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media;␣
↪gyroscope; picture-in-picture" allowfullscreen></iframe>
</div>
 </div>
```

6.4 Menu

A WebPath can have multiple Menus and Navigation bars. Menu can be fetched through a Rest API `/api/menu/<menu_id:int>` and also updated/created through the same.

Each menu item can have three types of links: raw url, page object or publication object. Each menu item can get additional contents (`inherited_contents`) from the publication. This means that a presentation url, or a subheading or whatever belonging to a publication can be made accessible during the representation of the menu items. Think about images, additional links and things that would fill up a menu entry.

6.5 Api

see `/openapi.json` and `/openapi` for OpenAPI v3 Schema.

CHAPTER
SEVEN

DEVELOPER'S

7.1 Models

The models are implemented within the following applications:

- **cms.contexts**, where websites, webpaths and EditorialBoard Users and Permissions can be defined
- **cms.templates**, where multiple page templates and blocks can be managed
- **cms.medias**, specific app for management, upload and navigation of media files.
- **cms.menus**, specific app for navigation bar management.
- **cms.carousels**, specific app for Carousel and Slider management.
- **cms.pages**, where we can create a Page linked to a Webpath.
- **cms.publications**, where Editorial boards publish contents in one or more WebPaths.
- **cms.search**, MongoDB Search Engine and management commands i.e. CLI.

The module `cms.contexts` defines the multitenancy feature. Each WebPath would have a related web pages. Each context have users (Editorial Board Editors) with single or multiple permissions (see `cms.contexts.settings.CMS_CONTEXT_PERMISSIONS`)

The modules `cms.page` and `cms.publications` defines how a Page or a Publication is built. A Page is nothing but a composition of blocks, rendered in a HTML base template. This means that a page is just container block where multiple block can be defined in different order and fashion. For every page we must define to context (webpath) belonging as well as the template that we wish to adopt to be rendered by HTML.

7.2 WebPaths

[WiP]

This section describes how WebPath works and how it can be configured.

- path value match
- child path behavior
- the role of **.get_full_path()**
- some use cases and strategies: third-party url, webpath aliases, intheritance by webpath childs

7.3 Post Pre Save Hooks

By default Pages and Publication calls pre and post save hooks. Django signals are registered in `cms.contexts.signals`. In `settings.py` we can register as many as desidered hooks within single or multiple models. Django signals will load them in each pre/post save/delete events.

```
CMS_HOOKS = {
    'Publication': {
        'PRESAVE': [],
        'POSTSAVE': ['cms.search.hooks.publication_se_insert',],
        'PREDELETE': ['cms.search.hooks.searchengine_entry_remove',],
        'POSTDELETE': []
    },
    'Page': {
        'PRESAVE': [],
        'POSTSAVE': ['cms.search.hooks.page_se_insert',],
        'PREDELETE': ['cms.search.hooks.searchengine_entry_remove',],
        'POSTDELETE': []
    },
    'Media': {
        'PRESAVE': ['cms.medias.hooks.set_file_meta',
                    'cms.medias.hooks.webp_image_optimizer'],
        'POSTSAVE': [],
        'PREDELETE': [],
        'POSTDELETE': ['cms.medias.hooks.remove_file']
    },
    'Category': {
        'PRESAVE': ['cms.medias.hooks.webp_image_optimizer'],
        'POSTSAVE': [],
        'PREDELETE': [],
        'POSTDELETE': ['cms.medias.hooks.remove_file']
    },
    'PublicationAttachment': {
        'PRESAVE': ['cms.medias.hooks.set_file_meta',],
        'POSTSAVE': [],
        'PREDELETE': [],
        'POSTDELETE': []
    }
}
```

7.4 Template tags

The HTML template and/or an HTML page block can also adopt some of the template tags that shipped with uniCMS and Django. UniCMS template context by default comes with the following two objects:

```
    'webpath': Context object (cms.contexts.models.WebPath)
    'page': Page object (cms.pages.models.Page)
```

Based on informations taken from these objects as input uniCMS adopts some additionale custom templatetags as outlined below. These templatetags will also work in Page Blocks that would take, optionally, the HTML template as parameter.

7.4.1 cms_carousels

- **load_carousel** renders in the template the first active carousel in section or the identified one, with translated items. *arguments*: context, section, template, carousel_id *example*:

```
{% load_carousel section="template-section" template="template.html" %}
{% load_carousel carousel_id="1" %}
```

7.4.2 cms_contexts

- **breadcrumbs** builds webpath breadcrumbs. If leaf, appends leaf breadcrumbs. *arguments*: webpath, template (opt, default=breadcrumbs.html), leaf (opt) *example*: `{% breadcrumbs webpath=webpath template="breadcrumbs.html" %}`
- **call** calls any object method and also pass to it whatever `**kwargs`. *arguments*: obj, method, kwargs *example*: `{% call obj=publication method="get_url_list" category_name=cat %}`
- **language_menu** builds a data dict with {url:language} pairs. If a template is present, passes it data. *arguments*: teamplate (opt) *example*:

```
{% language_menu as language_urls %}
{% for lang,url in language_urls.items %}
<li><a class="list-item" href="{{ url }}"><span>{{ lang }}</span></a></li>
{% endfor %}
```

7.4.3 cms_menus

- **load_menu** renders in the template the first active menu in section or the identified one, with translated items. *arguments*: context, section, template, menu_id *example*:

```
{% load_menu section="template-section" template="menu.html" %}
{% load_menu menu_id="1" %}
```

7.4.4 cms_pages

- **cms_categories** returns all CMS content categories. *example*: `{% cms_categories %}`
- **load_blocks** it would be configured in the base templates and defines where the blocks would be rendered. it takes `section` as argument, to query/filter only active blocks that belongs to that section. *arguments*: section (opt) *example*: `{% load_blocks section="banner" %}`
- **load_link** gets a URL as parameter and pass it to a template. *arguments*: template, url *example*: `{% load_link url="https://myvideo.it" template="iframe-video.html" %}`
- **load_page_title** returns a translated page title. *arguments*: page *example*: `{% load_page_title page=page %}`

7.4.5 cms_publication

- **load_publication** pass a single active publication to a template. *arguments*: template, publication_id *example*: `{% load_publication publication_id="1" template="publication-layout.html" %}`
- **load_publications_preview** returns all published publications in a context and passes them to a template. *arguments*: template, section (opt), number (opt, default=6), in_evidence (opt, default=False), categories_csv (opt), exclude_categories (opt, default=False), tags_csv (opt) *example*:

```
    {% load_publications_preview template="publ.html" number="3" %}
    {% load_publications_preview template="publ.html" categories="Research, Study" %}
    {% load_publications_preview template="publ.html" categories="Research" exclude_
↪categories=True %}
    {% load_publications_preview template="publ.html" tags_csv="read, sport" %}
    {% load_publications_preview template="publ.html" in_evidence=True %}
```

7.4.6 cms_templates

- **blocks_in_position** returns True if there are active blocks in passed position or its childs, else False. *arguments*: position *example*: `{% blocks_in_position position="section-1" %}`
- **supported_languages** returns settings.LANGUAGES *example*: `{% supported_languages %}`

7.5 Handlers

There are circumstances and scenarios where is necessary to create specific applications with templates and templatetags, detached from the pages that are configured within the CMS. The `cms.publications.handlers` for instance, it manages the pages for navigation of publications (List) and opening a publication (View).

In such scenario the handlers have to be registered in `settings.py` as follow:

```
CMS_PUBLICATION_VIEW_PREFIX_PATH = 'contents/news/view/'
CMS_PUBLICATION_LIST_PREFIX_PATH = 'contents/news/list'

CMS_PUBLICATION_URL_LIST_REGEXP = f'^(?P<context>[\/a-zA-Z0-9\.\-\_]*)({CMS_PUBLICATION_
↪LIST_PREFIX_PATH})/?$'
CMS_PUBLICATION_URL_VIEW_REGEXP = f'^(?P<context>[\/a-zA-Z0-9\.\-\_]*)({CMS_PUBLICATION_
↪VIEW_PREFIX_PATH})(?P<slug>[a-z0-9\-]*)'

CMS_APP_REGEXP_URLPATHS = {
    'cms.publications.handlers.PublicationViewHandler' : CMS_PUBLICATION_URL_VIEW_REGEXP,
    'cms.publications.handlers.PublicationListHandler' : CMS_PUBLICATION_URL_LIST_REGEXP,
}

CMS_HANDLERS_PATHS = [CMS_PUBLICATION_VIEW_PREFIX_PATH,
                      CMS_PUBLICATION_LIST_PREFIX_PATH]
```

The paths defined in CMS_HANDLERS_PATHS generates the list of reserved words to be considered during validation in `cms.contexts.models.WebPath`. Therefore, they create the list of reserved words that cannot be used as path value in `cms.contexts.models.WebPath`.

7.6 Middlewares

`cms.contexts.middleware.detect_language_middleware`: detects the browser user language checking both `?lang=` request arg and the web browser default language. This required to handle the Menu, Carousel and localized Publication.

`cms.contexts.middleware.show_template_blocks_sections`: toggles, for staff users, the display of block sections in pages.

`cms.contexts.middleware.show_cms_draft_mode`: toggles, for staff users, the draft view mode in pages.

7.7 Example data

If you want to dump and share your example data:

```
./manage.py dumpdata --exclude auth.permission --exclude accounts --exclude contenttypes␣
↪--exclude sessions --exclude admin --indent 2 > ../dumps/cms.json
```

CHAPTER

EIGHT

WHY ANOTHER CMS?

8.1 The Goal

We're constantly looking for limitless experiences. When we are going through the selection of CMS platforms we find different and variety of products and solutions that each are based on different functionalities, depending on what they offer and designed for. And this creates huge limitations if you are looking to customize the CMS for your specific need. Starting from our use case (in a univeristy environment) below the list of some of the reasons why another CMS.

1. **Changing the structure of pages**. In a CMS like Wordpress you cannot modify the structure of the page and you are limited to deal with its content only. To modify the structure you are forced to modify its template in use which requires PHP knowledge.
2. **Implementation of large and scalable portals/services**. Wordpress does not scale, it is designed as a personal blog therefore it does not satisfy the requirements when it comes to large volumes of traffic.
3. **Graphics customization (templates and structure)**. All platforms require highly specialized development efforts within their environment. There are (expensive) plugins that damage the users' impression of customizing graphics, but these only offer predefined assets of ordinary solutions. Even on Drupal and Joomla the designer cannot do any graphic customization without engaging with the development team.
4. **Reuse of images in different contexts**. Drupal allows it only if combined with some plugins that modify the post structure and also the publication methods.
5. **Moving content from one context to another**. In a standard CMS environment this requires access and alteration of the contents present on the database and data in general. In Drupal for example both the paths defined on the filesystem and the information within the database require changes. In UniCMS it is possible to do this simply by referencing the content to one or more webpaths.
6. **Content Inheritance**. Drupal and other similar CMS cannot inherit the contents of a parent page.
7. **High availability**. Drupal and many others does not have native support for HA.

8.2 What about other Django based CMSs?

They are brilliant but:

- they still require development skills to build a professional CMS
- some of them overkille/overwhelmed with numerous extensions and components that transforms the platform in a non-linear environment and it gest very messy
- too much outsourced integration with templates, probably hindering the most important marketplace in the industry

This of course does not indicate that uniCMS is designed for dummy users, or that it won't get oversized in time however, the plan is to place in its core all common and necessary module which will reduce the risk of it getting overwhelmed with external components such as extensions etc.

Most importantly uniCMS is Django CMS… its core strenght comes precisely from Django!

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search