webcore Documentation

Release 1

Maxime Haineault

March 24, 2015

Contents

Webcore is a collection of django apps, utilities and libraries that were originally created and maintained internally at Motion Média. We are now proud to open source it for the django community.

The open source version is still in its infancy (not production ready!), but we hope it will flourish and gain maturity.

CHAPTER 1

Apps

- _google_analytics: http://code.google.com/p/django-google-analytics/ (tweaked)
- _HTML5boilerplate: http://html5boilerplate.com/ (tweaked)
- _colors: http://code.google.com/p/django-colors/

CHAPTER 2

Libraries & plugins

- _jQuery: http://jquery.com/
- _jQuery UI: http://jqueryui.com/
- _jQuery mobile: http://jquerymobile.com/
- _960 Grid system: http://960.gs/
- _modernizr: http://www.modernizr.com/
- _CSS3 PIE: http://css3pie.com/
- _jQuery.colorbox: http://colorpowered.com/colorbox/
- _dd_belatedpng.js: http://www.dillerdesign.com/experiment/DD_belatedPNG/
- _excanvas.js: http://excanvas.sourceforge.net/
- _jquery.mousewheel.js
- _jquery.cookie.js
- _jquery.slugify.js

CHAPTER 3

Installation & Setup

3.1 Installation

How to install webcore

CHAPTER 4

webcore

4.1 Generic URLs

Webcore provide a set of basic and reusable URLs, you can include them all in your project like this:

```
from django.conf.urls.defaults import *
from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/', include(admin.site.urls)),
    (r'',        include('yourproject.urls')),
    (r'',        include('webcore.urls')),
)
```

Note: Only webcore.urls.sitemap isn't included by default.

Alternatively you can cherry pick which URLs you want to include as documented below.

4.1.1 Available URLs

webcore.urls.robots

Example:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'', include('yourproject.urls')),
    (r'', include('webcore.urls.robots')),
)
```

This will serve the webcore/templates/robots.txt which contains only this:

```
# www.robotstxt.org/
# www.google.com/support/webmasters/bin/answer.py?hl=en&answer=156449
```

User-agent: * User-agent: * Disallow: /admin/

You can override this template by creating a robots.txt file in your project templates folder to set your own rules.

webcore.urls.sitemap

Example:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'', include('yourproject.urls')),
    (r'', include('webcore.urls.sitemap')),
)
```

This will serve the following dummy site-map file which you should override using yourproject/templates/sitemap.xml.:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset
    xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
    http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">
    <url>
      <loc>http://www.example.com/</loc>
      <priority>1.00</priority>
    </url>
</urlset>
```

webcore.urls.favicon

Lots of browsers have an annoying tendency to request /favicon.ico even if the HTML doesn't specify it explicitly. This URL will redirect those requests to {{ MEDIA_URL }}img/favicon.ico:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'', include('yourproject.urls')),
    (r'', include('webcore.urls.favicon')),
)
```

webcore.urls.ifdev

This URL will serve media file with the dev server if *DEV* = *True* in the settings.py file.:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'', include('yourproject.urls')),
    (r'', include('webcore.urls.favicon')),
)
```

To better understand what's happening, here's the code of ifdev.py:

```
if settings.DEV:
    urlpatterns = patterns('',
        (r'^media/(.*)$', 'django.views.static.serve', {
            'document_root': settings.MEDIA_ROOT,
            'show_indexes': True}),
    )
```

```
else:
    urlpatterns = patterns('',)
```

4.2 Templates

Webcore provide some useful reusable generic templates. Most of them only needs to be styled with CSS, others are a good starting point to make your own.

4.2.1 HTML5boilerplate

For the moment this is the only base template available. This base template is a HTML5boilerplate template which contains (hopefully) all the necessary blocks to be extended.

First it's important to understand the blocks name spacing. The three most important namespaces are *site*, *project*, *app*.

Here's an hierarchical map of the block namespaces:

```
- site: generic stuff
    - project: project global stuff
        - app: application specific stuff
```

Extending base

The application templates should extend a base.html template that sits at project level. For portability convenience it should not extends the base.html provided by webcore directly.

For example, you project's base.html could look like this:

```
{% extends "html5boilerplate/base.html" %}

{% block project.headstyles %}
<link rel="stylesheet" href="{{ STATIC_URL }}libs/960/css/960.css" />
<link rel="stylesheet" href="{{ STATIC_URL }}libs/960/css/text.css" />
{% endblock%}

{% block project.title %}webcore{% endblock %}

{% block project.head %}
<h1>Webcore</h1>
{% block app.head %}{% endblock %}
{% endblock %}

{% block project.body %}
<div class="{% block container.class %}container_12{% endblock %}">
    {# It's strongly suggested to use content as main content block #}
    {# since most reusable apps use this block. #}
    {% block content %}{% endblock %}
</div>

{% block project.footer %}
{% block app.footer %}{% endblock %}
<div id="footer">
    &copy; Copyright {% now "Y" %} Motion Média, all rights reserved.
```

```
</div>
{% endblock %}

{% endblock %}
```

Site blocks

The site blocks are top level blocks that usually wraps project and app blocks. Those block serves as "defaults" that can be extended.

Block	Description
site.head	Wraps the content of <*head*>
site.title	Wraps the <*title*> element
site.meta	Wraps the description, keywords, project & app metas
site.headstyles	Wraps the project and app style blocks
site.headscripts	Wraps the project and app script blocks
site.body	Wraps the project head, body, footer blocks
site.styles	Wraps the project and app style blocks
site.scripts	Wraps the project and app script blocks. Also contains some other JS functionalities

Project blocks

The project blocks provide blocks that should be used at project level in django.

Block	Description
project.title	Project title are appended after app title
project.description	Project's description
project.keywords	Project's keywords
project.meta	Project title are appended before app meta
project.favicon	Wraps the favicon element
project.headstyles	Block to add styles to the head of document
project.headscripts	Block to add scripts to the head of document
project.head	Project's head
project.body	Project's body
project.footer	Project's footer

Other blocks

Some other blocks are provided for more edgy use cases

Block	Description
doctype	wraps the doctype, by default <*!DOCTYPE html*>
html.class	You can add classes to the <*html*> tag with this
html.extra	You can add attributes to the <*html*> tag with this

4.2.2 Pagination

A generic pagination template. You can paginate any list views simply by including the template like this:

```
{% include "pagination.html" %}
```

This template outputs something like this:

```
<ol class="clearfix">
    <li><a title="Previous page" href="../1/">Previous »</a></li>
    <li class="active"><span>2</span></li>
    <li><a href="../3">3</a></li>
    <li><a href="../4">4</a></li>
    <li><a title="Next page" href="../5/">Next »</a></li>
</ol>
```

The pagination should be wrapped in a container of you choice (like a div) to allow more flexible styling.

Here's a sample style:

```
.pagination {
    border-bottom: 1px solid #ccc;
}

.pagination ol {
    margin: 0;
    padding: 0;
    background: #eee;
}

.pagination li {
    margin: 0;
    padding: 0;
    list-style: none;
    float: left;
}

.pagination a,
.pagination span {
    display: block;
    padding: 3px 6px;
    text-decoration: none;
    background: #f4f4f4;
}

.pagination span {
    background: #f4f4f4;
}

.pagination .pagination-active a {
    background: #def;
}

.pagination .pagination-prev a {}
.pagination .pagination-next a {}

.pagination.bottom {
    border-top: 1px solid #ccc;
}
```

4.3 Template tags

Webcore ships with a number of apps which provide templatetags to make a developer's life easier.

4.3.1 webcore.design.libs

Webcore design libs provide shortcuts to load packaged CSS/JS libraries for convenience.

To use it simple load the template tags (you must add *'webcore.design.libs'* to your *settings.py* file):

```
{% load libs_tags %}
```

This will load the following tags:

- css
- csslib
- js
- jslib

css & csslib

Example:

```
{% block project.headstyles %}

{# Load libraries by their names #}
{% csslib "960 960_text 960_grid" %}

{# Load a file in media/ #}
{% css "css/site.css" %}

{# Load a file in static/ #}
{% css "somepath/widget.css" "static" %}
{% endblock%}
```

js & jslib

Example:

```
{% block project.scripts %}

{# Load libraries by their names #}
{% jslib "jqueryui live" %}

{# Load a file in media/ #}
{% js "mysite/poney.js" %}

{# Load a file in static/ #}
{% js "mysite/poney.js" "static" %}
{% endblock%}
```

CHAPTER 5

webcore.utils

5.1 Utils: debug

Currently there is only one debug utility named "brake" shipped in webcore.

5.1.1 Brake

Brake is a thin wrapper around IPython's interactive shell. To use it you must first install iPython, on Debian based distribution it's quite simple:

```
sudo apt-get install ipython
```

Here's an usage example:

```
from webcore.utils.debug import brake

from django.http import Http404

def detail(request, poll_id):
    try:
        p = Poll.objects.get(pk=poll_id)
        # If no exception is triggered, the interactive shell will
        # suspend the interpreter's execution here and you will be
        # able to play with the "p" variable interactively in console.
        brake()()
    except Poll.DoesNotExist:
        raise Http404
    return render_to_response('polls/detail.html', {'poll': p})
```

Warning: If IPython isn't available on your system this will fail silently.

This is the actual code from debug.py:

```
try:
    from IPython.Shell import IPShellEmbed
    def brake():
        return IPShellEmbed()
except:
    def brake():
        return lambda x: x
```

5.2 Utils: Google Analytics

If you want to use Google Analytics on your site, simply add *webcore.utils.google_analytics* in your *settings.INSTALLED_APPS*.

Then in your project's *base.html* template you can insert it like this:

```
{% extends "html5boilerplate/base.html" %}
{% load analytics %}

{% block project.scripts %}
{% analytics "UA-xxxxxx-x" %}
{% endblock %}
```

For the django functionality I use the django-google-analytics <http://code.google.com/p/django-google-analytics/> project.

However the JS was a bit out of date so the tracker code was updated with the asynchroneous tracker <http://code.google.com/apis/analytics/docs/tracking/asyncTracking.html>.

django-google-analytics also support multiple code tracking to handle multi-site setups. From the project page:

1. Add the google_analytics application to your INSTALLED_APPS section of your settings.py. This mode requires that you be using the Django sites framework too, so make sure you have that set up as well.
2. Add GOOGLE_ANALYTICS_MODEL = True to your settings.py Run a ./manage.py syncdb to add the database tables Go to your project's admin page (usually /admin/) and click into a site objects
3. You'll now see a new field under the normal site information called "Analytics Code". In this box you put your unique analytics code for your project's domain. It looks like UA-xxxxxx-x and save the site.
4. In your base template (usually a base.html) insert this tag at the very top: {% load analytics %}
5. In the same template, insert the following code right before the closing body tag: {% analytics %}

5.3 Utils: storage

Some nice storage things that are either gone or missing from django ..

5.3.1 file_cleanup

File cleanup callback used to emulate the old delete behavior using signals. Initially django deleted linked files when an object containing a File/ImageField was deleted.

Here's an usage example:

```
from django.db.models.signals import post_delete
from webcore.utils.storage import file_cleanup

post_delete.connect(file_cleanup, sender=MyModel, dispatch_uid="mymodel.file_cleanup")
```

5.3.2 ASCIISafeFileSystemStorage

Same as FileSystemStorage, but converts unicode characters in file name to ASCII characters before saving the file. This is mostly useful for the non-English world.

Usage (settings.py):

```
DEFAULT_FILE_STORAGE = 'webcore.utils.storage.ASCIISafeFileSystemStorage'
```

CHAPTER 6

webcore.design

6.1 Design: colors

Django Colors offers a set of filters to manipulate colors.

Project page: http://code.google.com/p/django-colors/

6.1.1 Filters

Color manipulation

Template filters to manipulate colors:

Filter	Description
color	Returns the hexadecimal value of a named color (ex: black -> 000000)
lightness	Set lightness to x, accept hexadecimal or hsv tuple as value
hue	Set hue to x, accept hexadecimal or hsv tuple as value
opposite	Returns the opposite (complementary) color on the HSV color space
saturation	Set saturation to x, accept hexadecimal or hsv tuple as value

Color spaces conversions & utils

Template filters to convert colors:

Filter	Description
hex_to_rgb	Returns the opposite color on the HSV color space
hex_to_hsv	Returns the HSV value of a hexadecimal color
hsv_to_hex	Returns the hexadecimal value of a HSV color
expand_hex	Expands shorthand hexadecimal code, ex: c30 -> cc3300
short_hex	Return shorthand hexadecimal code, ex: cc3300 -> c30

6.1.2 Real world examples

Saturation example

In this example we take an hexadecimal string and set its saturation to *10*:

```
{% load colors %}

h1 {
    background: #{{ hexcode|saturation:"10" }}
}
```

Chaining example

It's possible to chain transformations, in the next example we set the saturation to *50* and set the lightness to *40* on the result:

```
{% load colors %}

h1 {
    background: #{{ hexcode|saturation:"50"|lightness:"40" }};
}
```

Using RGB values in CSS

Example:

```
{% load colors %}

h1 {
    background: {{ hexcode|hex_to_rgb:"rgb(%s, %s, %s)" }};
}
```

Dynamically generated gradients

Here's a more useful example:

```
{% load colors %}

#my_div {
    /* non-css3 */
    background: #{{ color }};
    /* IE */
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#{{ color }}', endColorstr='#{
    /* webkit */
    background: -webkit-gradient(radial, left top, left bottom, from(#{{ color }}), to(#{{ color|ligh
    /* firefox 3.6+ */
    background: -moz-linear-gradient(top,  #{{ color }},  #{{ color|lightness:"30" }});
}
```

6.2 Design: libs

Webcore comes packed with the lots of commonly used CSS/JS libraries.

Tip: You can use libs templatetags to load any library in your template using its keyword.

Example:

```
{% load libs_tags %}
{% csslib "960 960_text" %}
```

For more informations you can refer to the *Template tags* documentation.

6.2.1 CSS libs

keyword	lib
960_12_col	960 grid (12 cols)
960_12_col_rtl	960 grid (12 cols, right to left)
960_16_col	960 grid (16 cols)
960_16_col_rtl	960 grid (16 cols, right to left)
960_24_col	960 grid (24 cols)
960_24_col_rtl	960 grid (24 cols, right to left)
960	960 grid
960_rtl	960 grid (right to left)
960_grid	shows grid background
960_reset	960 grid reset
960_reset_rtl	960 grid reset (right to left)
960_text	960 grid text
960_text_rtl	960 grid text (right to left)
colorbox_theme1	colorbox theme #1
colorbox_theme2	colorbox theme #2
colorbox_theme3	colorbox theme #3
colorbox_theme4	colorbox theme #4
colorbox_theme5	colorbox theme #5
ui_lightness	jQuery UI lightness theme
ui_darkness	jQuery UI darkness theme

6.2.2 JS libs

keyword	lib
csspie	CSS3 Pie <http://css3pie.com/>
colorbox	jQuery Colorbox <http://colorpowered.com/colorbox/>
colorbox_src	jQuery Colorbox (uncompressed)
jquery	jQuery <http://jquery.com/>
jquerymobile	jQuery Mobile <http://jquerymobile.com/>
jqueryui	jQuery UI <http://jqueryui.com/>
live	Live.js <http://livejs.com/>
modernizr	Modernizr <http://www.modernizr.com/>

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*