django-embed-video Documentation

Release 1.2.0-stable

Juda Kaleta

Aug 29, 2019

Contents

Django app for easy embeding YouTube and Vimeo videos and music from SoundCloud.

Repository is located on GitHub: https://github.com/jazzband/django-embed-video

CHAPTER 1

Installation & Setup

1.1 Installation

The simpliest way is to use pip to install package:

```
pip install django-embed-video
```

If you want latest version, you may use Git. It is fresh, but unstable.

```
pip install git+https://github.com/jazzband/django-embed-video
```

1.2 Setup

Add `embed_video` to `INSTALLED_APPS` in your Django settings.

```
INSTALLED_APPS = (
    ...

    'embed_video',
)
```

To detect HTTP/S you must use `request` context processor:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'django.core.context_processors.request',
)
```

CHAPTER 2

Examples

2.1 Template examples

First you have to load the `embed_video_tags` template tags in your template:

```
{% load embed_video_tags %}
```

Embedding of video:

```
{# you can just embed #}
{% video item.video 'small' %}

{# or use variables (with embedding, too) #}
{% video item.video as my_video %}
    URL: {{ my_video.url }}
    Thumbnail: {{ my_video.thumbnail }}
    Backend: {{ my_video.backend }}
    {% video my_video 'small' %}
{% endvideo %}
```

Default sizes are `tiny` (420x315), `small` (480x360), `medium` (640x480), `large` (960x720) and `huge` (1280x960). You can set your own size:

```
{% video my_video '800x600' %}
```

This usage has been added in version 0.7.

And use relative percentual size:

```
{% video my_video '100% x 50%' %}
```

It is possible to set backend options via parameters in template tag. It is useful for example to enforce HTTPS protocol or set different query appended to url.

```
{% video my_video query="rel=0&wmode=transparent" is_secure=True as my_video %}
    {{ my_video.url }} {# always with https #}
{% endvideo %}
```

Tip: We recommend to use sorl-thumbnail to change thumbnail size.

Tip: To speed up your pages, consider template fragment caching.

Tip: You can overwrite default template of embed code located in `templates/embed_video/embed_code.html` or set own file for custom backend (*template_name*).

`template_name` has been added in version 0.9.

2.2 Model examples

Using the `EmbedVideoField` provides you validation of URLs.

```
from django.db import models
from embed_video.fields import EmbedVideoField

class Item(models.Model):
    video = EmbedVideoField()  # same like models.URLField()
```

2.3 Admin mixin examples

Use `AdminVideoMixin` in `admin.py`.

```
from django.contrib import admin
from embed_video.admin import AdminVideoMixin
from .models import MyModel

class MyModelAdmin(AdminVideoMixin, admin.ModelAdmin):
    pass

admin.site.register(MyModel, MyModelAdmin)
```

2.4 Custom backends

If you have specific needs and default backends don't suits you, you can write your custom backend.

`my_project/my_app/backends.py`:

```
from embed_video.backends import VideoBackend

class CustomBackend(VideoBackend):
```

(continues on next page)

(continued from previous page)

```
    re_detect = re.compile(r'http://myvideo\.com/[0-9]+')
    re_code = re.compile(r'http://myvideo\.com/(?P<code>[0-9]+)')

    allow_https = False
    pattern_url = '{protocol}://play.myvideo.com/c/{code}/'
    pattern_thumbnail_url = '{protocol}://thumb.myvideo.com/c/{code}/'

    template_name = 'embed_video/custombackend_embed_code.html'  # added in v0.9
```

You can also overwrite *VideoBackend* methods, if using regular expressions isn't well enough.

`my_project/my_project/settings.py`:

```
EMBED_VIDEO_BACKENDS = (
    'embed_video.backends.YoutubeBackend',
    'embed_video.backends.VimeoBackend',
    'embed_video.backends.SoundCloudBackend',
    'my_app.backends.CustomBackend',
)
```

2.5 Low level API examples

You can get instance of *VideoBackend* in your python code thanks to *detect_backend()*:

```
from embed_video.backends import detect_backend

my_video = detect_backend('http://www.youtube.com/watch?v=H4tAOexHdR4')
```

CHAPTER 3

Example project

For easy start with using django-embed-video, you can take a look at example project. It is located in example_project directory in root of repository.

3.1 Running example project

1. Install requirements:

   ```
   pip install -r requirements.txt
   ```

2. Create database:

   ```
   python manage.py migrate --run-syncdb --noinput
   ```

   Or, for older versions of Django:

   ```
   python manage.py syncdb --noinput
   ```

3. Run testing server:

   ```
   python manage.py runserver
   ```

4. Take a look at http://localhost:8000 . You can log in to administration with username `admin` and password `admin`.

3.2 Testing HTTPS

To test HTTPS on development server, follow this instructions.

CHAPTER 4

Development

4.1 Contributing

I will be really pleased if you will provide patch to this Django app. Feel free to change whatever, but keep PEP8 rules and Zen.

It is a good habit to cover your patches with *tests*.

Repository is hosted on Github: https://github.com/jazzband/django-embed-video

4.2 Testing

4.2.1 Requirements

The library needs `Django` and `requests` and `nose`, `mock`, `south` and `testfixtures` libraries to run tests. They will be installed automatically when running tests via setup.py

4.2.2 Running tests

Run tests with this command:

```
python setup.py nosetests
```

Be sure to run it before each commit and fix broken tests.

Run tests with coverage:

```
python setup.py nosetests --with-coverage --cover-package=embed_video
```

4.3 Changelog

4.3.1 Release 1.2.0 (October 04, 2018)

- hotfix docs directive (#94)
- update docs (#92)
- use tests_require and setup_requires for nose testing requirements (#91)
- add renderer kwarg to Widget render method to support Python 2.1 and later (#88)
- enable default HTTPS support for YouTube, VimeoBackend, SoundCloudBackend (#86)
- added syntax highlight in README.rst (#81)
- updating requests >=2.19

4.3.2 Release 1.1.2 (April 27, 2017)

- fix pypi

4.3.3 Release 1.1.1 (March 24, 2017)

- updates for Django 1.10 and 1.11 (#73)
- update requirements for installation of the example project (#72)
- use secure connection to query soundcloud endpoint (#68)

4.3.4 Release 1.1.0 (Jan 19, 2016)

- added support fort Django 1.9 (#52)
- if possible YouTube thumbnails are returned in better resolution (#43)

4.3.5 Release 1.0.0 (May 01, 2015)

Backward incompatible changes:

- filter *embed_video_tags.embed* has been removed
- changed behaviour of extra params in video tag (#34, #36)

Backward compatible changes:

- added support for Django 1.7 and Django 1.8
- added support for Vimeo channels (#47)
- fix resizing of SoundCloud iframe (#41)

4.3.6 Release 0.11 (July 26, 2014)

- add support for YouTube mobile urls (#27)
- fix passing parameters in calling request library (#28)
- fix validation of urls (#31)

4.3.7 Release 0.10 (May 24, 2014)

- `video` tag accepts kwargs (#20)
- `video` tag will not crash anymore with `None` passed as url (#24)

4.3.8 Release 0.9 (Apr. 04, 2014)

- Add `VideoBackend.template_name` and rendering embed code from file.
- Allow relative sizes in template tag (#19).
- Fix handling invalid urls of SoundCloud. (#21).
- Catch `VideoDoesntExistException` and `UnknownBackendException` in template tags and admin widget.
- Add base exception `EmbedVideoException`.

4.3.9 Release 0.8 (Feb. 22, 2014)

- Add `EMBED_VIDEO_TIMEOUT` to settings.
- Fix renderering template tag if no url is provided (#18)
- If `EMBED_VIDEO_TIMEOUT` timeout is reached in templates, no exception is raised, error is just logged.
- Fix default size in template tag. (See more…)

4.3.10 Release 0.7 (Dec. 21, 2013)

- Support for sites running on HTTPS
- `embed` filter is deprecated and replaced by `video` filter.
- caching for whole backends was removed and replaced by caching properties
- minor improvements on example project (fixtures, urls)

4.3.11 Release 0.6 (Oct. 04, 2013)

- Ability to overwrite embed code of backend
- Caching backends properties
- PyPy compatibility
- Admin video mixin and video widget

4.3.12 Release 0.5 (Sep. 03, 2013)

- Added Vimeo thumbnails support
- Added caching of results
- Added example project
- Fixed template tag embed
- Fixed raising UnknownIdException in YouTube detecting.

4.3.13 Release 0.4 (Aug. 22, 2013)

- Documentation was rewrited and moved to http://django-embed-video.rtfd.org/ .
- Custom backends (http://django-embed-video.rtfd.org/en/latest/examples.html#custom-backends).
- Improved YouTube and Vimeo regex.
- Support for Python 3.
- Renamed `base` to `backends`.

4.3.14 Release 0.3 (Aug. 20, 2013)

- Security fix: faked urls are treated as invalid. See this page for more details.
- Fixes:
  - allow of empty video field.
  - requirements in setup.py
- Added simplier way to embed video in one-line template tag:

```
{{ 'http://www.youtube.com/watch?v=guXyvo2FfLs'|embed:'large' }}
```

- `backend` variable in `video` template tag.

  Usage:

```
{% video item.video as my_video %}
    Backend: {{ my_video.backend }}
{% endvideo %}
```

4.3.15 Release 0.2 (June 25, 2013)

- Support of SoundCloud

4.3.16 Release 0.1 (June 1, 2013)

- Initial release

CHAPTER 5

Websites using django-embed-video

- Tchorici.cz
- Tiempoturco.com

Are you using django-embed-video? Send pull request!

CHAPTER 6

Library API

6.1 API reference

6.1.1 Admin

class `embed_video.admin.`**`AdminVideoMixin`**

Mixin using *`AdminVideoWidget`* for fields with *`EmbedVideoField`*.

Usage:

```
from django.contrib import admin
from embed_video.admin import AdminVideoMixin
from .models import MyModel

class MyModelAdmin(AdminVideoMixin, admin.ModelAdmin):
    pass

admin.site.register(MyModel, MyModelAdmin)
```

`formfield_for_dbfield`(*db_field*, ***kwargs*)

class `embed_video.admin.`**`AdminVideoWidget`**(*attrs=None*)

Widget for video input in administration. If empty it works just like `django.forms.TextInput`. Otherwise it renders embedded video together with input field.

Todo: Django 1.6 provides better parent for this widget - `django.forms.URLInput`.

`render`(*name*, *value=”*, *attrs=None*, *size=(420, 315)*, *renderer=None*)

6.1.2 Backends

exception embed_video.backends.**EmbedVideoException**
Parental class for all embed_video exceptions

class embed_video.backends.**SoundCloudBackend**(*url*)
Backend for SoundCloud URLs.

get_code()
Returns video code matched from given url by `re_code`.

Return type str

get_embed_code(*width*, *height*)
Returns embed code rendered from template `template_name`.

Return type str

get_info()

Return type dict

get_thumbnail_url()
Returns thumbnail URL folded from `pattern_thumbnail_url` and parsed code.

Return type str

get_url()
Returns URL folded from `pattern_url` and parsed code.

height

Return type str

is_secure = True
Decides if secured protocol (HTTPS) is used.

Type bool

width

Return type str

exception embed_video.backends.**UnknownBackendException**
Exception thrown if video backend is not recognized.

exception embed_video.backends.**UnknownIdException**
Exception thrown if backend is detected, but video ID cannot be parsed.

class embed_video.backends.**VideoBackend**(*url*)
Base class used as parental class for backends.

Backend variables:

url	URL of video.
code	Code of video.
thumbnail	URL of video thumbnail.
query	String transformed to QueryDict appended to url.
info	Additional information about video.
is_secure	Decides if secured protocol (HTTPS) is used.
protocol	Protocol used to generate URL.

Continued on next page

Table 1 – continued from previous page

`template_name`	Name of embed code template used by `get_embed_code()`.

```
class MyBackend(VideoBackend):
    ...
```

allow_https = True
Sets if HTTPS version allowed for specific backend.

Type bool

code
Code of video.

default_query = ''
Default query string or *QueryDict* appended to url

Type str

get_code()
Returns video code matched from given url by `re_code`.

Return type str

get_embed_code(*width*, *height*)
Returns embed code rendered from template `template_name`.

Return type str

get_info()

Return type dict

get_thumbnail_url()
Returns thumbnail URL folded from `pattern_thumbnail_url` and parsed code.

Return type str

get_url()
Returns URL folded from `pattern_url` and parsed code.

info
Additional information about video. Not implemented in all backends.

is_secure = False
Decides if secured protocol (HTTPS) is used.

Type bool

classmethod is_valid(*url*)
Class method to control if passed url is valid for current backend. By default it is done by `re_detect` regex.

pattern_thumbnail_url = None
Pattern in which the code is inserted to get thumbnail url.

Example: `http://static.myvideo.com/thumbs/%s`

Type str

pattern_url = None
Pattern in which the code is inserted.

Example: `http://myvideo.com?code=%s`

Type str

protocol
Protocol used to generate URL.

query
String transformed to QueryDict appended to url.

re_code = None
Compiled regex (`re.compile()`) to search code in URL.

Example: `re.compile(r'myvideo\.com/\?code=(?P<code>\w+)')`

re_detect = None
Compilede regec (`re.compile()`) to detect, if input URL is valid for current backend.

Example: `re.compile(r'^http://myvideo\.com/.*')`

set_options(*options*)

template_name = 'embed_video/embed_code.html'
Name of embed code template used by *get_embed_code()*.

Passed template variables: `{{ backend }}` (instance of VideoBackend), `{{ width }}`, `{{ height }}`

Type str

thumbnail
URL of video thumbnail.

url
URL of video.

exception embed_video.backends.**VideoDoesntExistException**
Exception thrown if video doesn't exist

class embed_video.backends.**VimeoBackend**(*url*)
Backend for Vimeo URLs.

get_info()

Return type dict

get_thumbnail_url()
Returns thumbnail URL folded from `pattern_thumbnail_url` and parsed code.

Return type str

is_secure = True
Decides if secured protocol (HTTPS) is used.

Type bool

class embed_video.backends.**YoutubeBackend**(*url*)
Backend for YouTube URLs.

get_code()
Returns video code matched from given url by `re_code`.

Return type str

get_thumbnail_url()
Returns thumbnail URL folded from `pattern_thumbnail_url` and parsed code.

Return type str

is_secure = True
Decides if secured protocol (HTTPS) is used.

Type bool

`embed_video.backends.`**detect_backend**(*url*)
Detect the right backend for given URL.

Goes over backends in `settings.EMBED_VIDEO_BACKENDS`, calls *`is_valid()`* and returns backend instance.

Parameters **url** (*str*) – URL which is passed to *is_valid* methods of VideoBackends.

Returns Returns recognized VideoBackend

Return type *VideoBackend*

6.1.3 Fields

class `embed_video.fields.`**EmbedVideoField**(*verbose_name=None*, *name=None*, ***kwargs*)
Model field for embedded video. Descendant of `django.db.models.URLField`.

class `embed_video.fields.`**EmbedVideoFormField**(**args*, ***kwargs*)
Form field for embeded video. Descendant of `django.forms.URLField`

6.1.4 Settings

EMBED_VIDEO_BACKENDS

List of backends to use.

Default:

```
EMBED_VIDEO_BACKENDS = (
    'embed_video.backends.YoutubeBackend',
    'embed_video.backends.VimeoBackend',
    'embed_video.backends.SoundCloudBackend',
)
```

EMBED_VIDEO_TIMEOUT

Sets timeout for `GET` requests to remote servers.

Default: `10`

EMBED_VIDEO_YOUTUBE_DEFAULT_QUERY

Sets default *`query`* appended to YouTube url. Can be string or `QueryDict` instance.

Default: `"wmode=opaque"`

6.1.5 Template tags

You have to load template tag library first.

```
{% load embed_video_tags %}
```

class embed_video.templatetags.embed_video_tags.**VideoNode**(*parser*, *token*)

Template tag `video`. It gives access to all `VideoBackend` variables.

Usage (shortcut):

```
{% video URL [SIZE] [key1=value1, key2=value2...] %}
```

Or as a block:

```
{% video URL [SIZE] [key1=value1, key2=value2...] as VAR %}
    ...
{% endvideo %}
```

Examples:

```
{% video item.video %}
{% video item.video "large" %}
{% video item.video "340x200" %}
{% video item.video "100% x 300" query="rel=0&wmode=opaque" %}

{% video item.video is_secure=True as my_video %}
    URL: {{ my_video.url }}
    Thumbnail: {{ my_video.thumbnail }}
    Backend: {{ my_video.backend }}
{% endvideo %}
```

classmethod embed(*url*, *size*, *context=None*, ***options*)

Direct render of embed video.

Parameters

- **url** (*str*) – URL to embed video
- **size** (*str*) – Size of rendered block
- **context** (*django.template.RequestContext | None*) – Django template RequestContext

static get_backend(*backend_or_url*, *context=None*, ***options*)

Returns instance of VideoBackend. If context is passed to the method and request is secure, than the is_secure mark is set to backend.

A string or VideoBackend instance can be passed to the method.

Parameters

- **backend** – Given instance inherited from VideoBackend or url
- **context** (*django.template.RequestContext | None*) – Django template RequestContext

Return type *VideoBackend*

classmethod get_size(*value*)

Predefined sizes:

size	width	height
tiny	420	315
small	480	360
medium	640	480
large	960	720
huge	1280	960

You can also use custom size - in format `WIDTHxHEIGHT` (eg. `500x400`).

Returns Returns tuple with (width, height) values.

Return type tuple[int, int]

render (*context*)

Returns generated HTML.

Parameters **context** (*django.template.RequestContext*) – Django template RequestContext

Returns Rendered HTML with embed video.

Return type django.utils.safestring.SafeText | str

render_block (*context*, *backend*)

Parameters

- **context** (*django.template.RequestContext*) – Django template RequestContext
- **backend** (`VideoBackend`) – Given instance inherited from VideoBackend

Return type django.utils.safestring.SafeText

resolve_options (*context*)

Parameters **context** (*django.template.RequestContext*) – Django template RequestContext

6.1.6 Utils

`embed_video.utils.`**import_by_path** (*dotted_path*, *error_prefix=''*)

Import a dotted module path and return the attribute/class designated by the last name in the path. Raise ImproperlyConfigured if something goes wrong.

Warning: Deprecated since version Django: 1.6

Function `django.utils.module_loading.import_by_path()` has been added in Django 1.6.

Parameters **dotted_path** (*str*) – Path to imported attribute or class

Returns imported attribute or class

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index