django-audiofield Documentation

Release 0.8.2

Arezqui Belaid

Sep 27, 2017

Contents

Release 0.8.2

Date Sep 27, 2017

Contents:

CHAPTER 1

Introduction

Version 0.8.2

Release 0.8.2

Date Sep 27, 2017

Keywords django, python, audiofield, audio, wav, mp3, sox

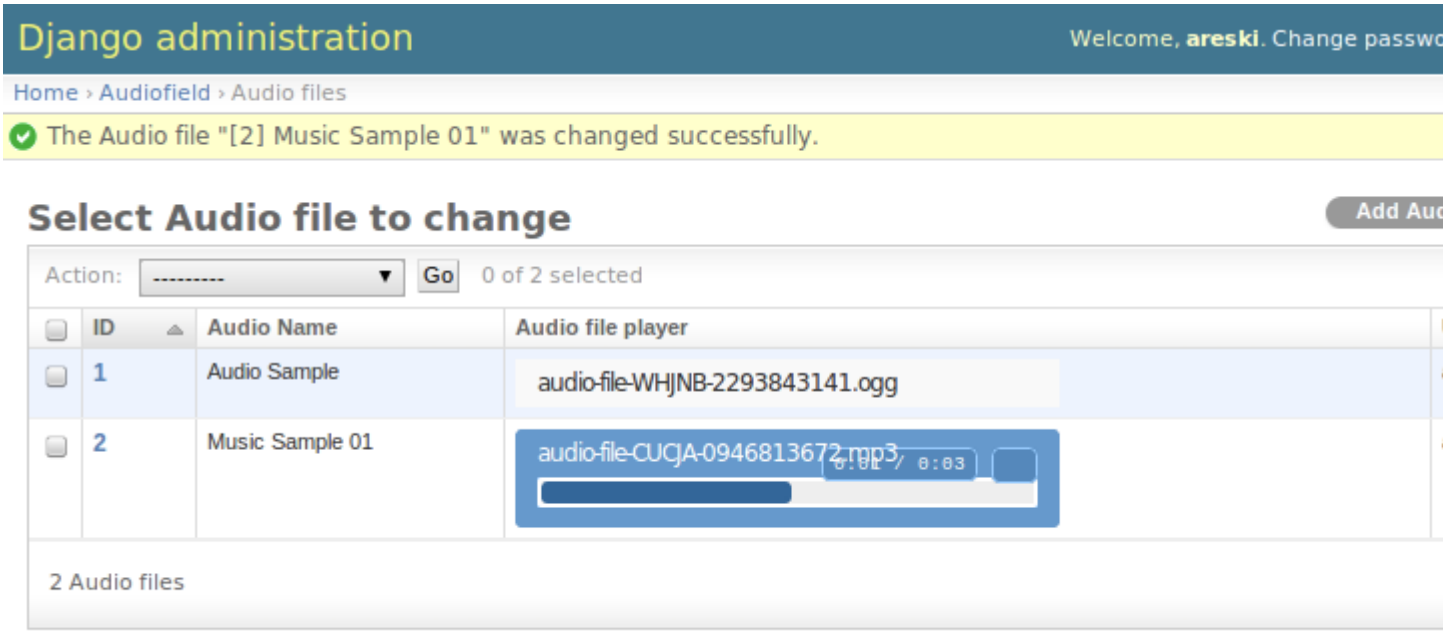
– Django-audiofield is an application written in Python, using the `Django` Framework.

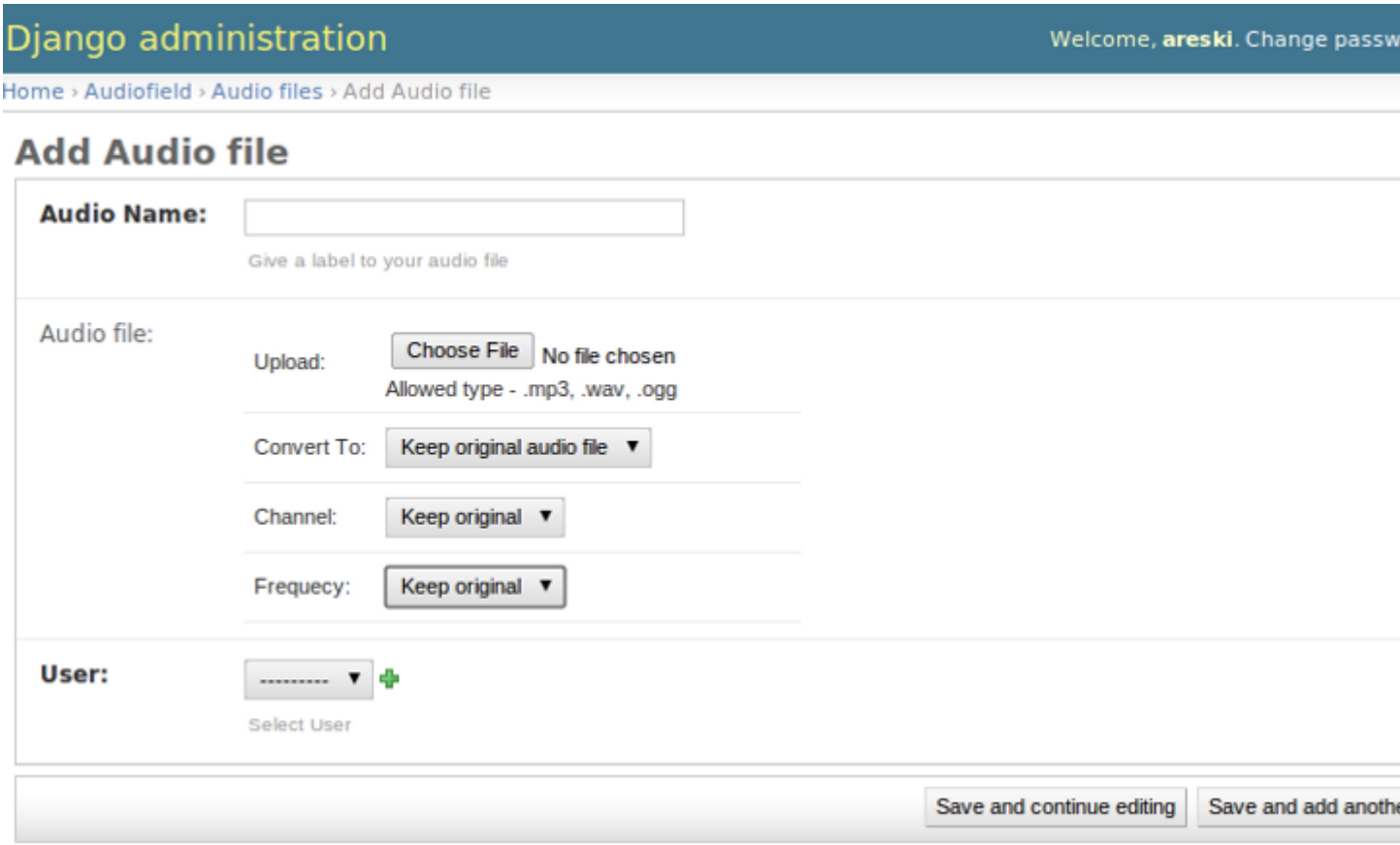
The license is MIT.

Overview

Django-audiofield is a Django application which allows audio file upload and conversion to mp3, wav and ogg format. It also makes it easy to play the audio files into your django application, for this we integrated a HTML5 and Flash audio player 'SoundManager2'

The goal of this project is to quickly manage audio files into your django project and make it easy for admins and users to listen to them.





More information about Soundmanager2 : http://www.schillmania.com/projects/soundmanager2/

Usage

Add the following lines in your models.py file:

```
from django.conf import settings
from audiofield.fields import AudioField
import os.path

# Add the audio field to your model
audio_file = AudioField(upload_to='your/upload/dir', blank=True,
                        ext_whitelist=(".mp3", ".wav", ".ogg"),
                        help_text=("Allowed type - .mp3, .wav, .ogg"))

# Add this method to your model
def audio_file_player(self):
    """audio player tag for admin"""
    if self.audio_file:
        file_url = settings.MEDIA_URL + str(self.audio_file)
        player_string = '<ul class="playlist"><li style="width:250px;">\
        <a href="%s">%s</a></li></ul>' % (file_url, os.path.basename(self.audio_file.
↪name))
        return player_string
audio_file_player.allow_tags = True
audio_file_player.short_description = _('Audio file player')
```

Add the following lines in your admin.py:

```
from your_app.models import your_model_name

# add 'audio_file_player' tag to your admin view
list_display = (..., 'audio_file_player', ...)
actions = ['custom_delete_selected']

def custom_delete_selected(self, request, queryset):
    #custom delete code
    n = queryset.count()
    for i in queryset:
        if i.audio_file:
            if os.path.exists(i.audio_file.path):
                os.remove(i.audio_file.path)
        i.delete()
    self.message_user(request, _("Successfully deleted %d audio files.") % n)
custom_delete_selected.short_description = "Delete selected items"

def get_actions(self, request):
    actions = super(AudioFileAdmin, self).get_actions(request)
    del actions['delete_selected']
    return actions
```

If you are not using the installation script, please copy following template file to your template directory:

```
cp audiofield/templates/common_audiofield.html /path/to/your/templates/directory/
```

Add the following in your template files (like admin/change_form.html, admin/change_list.html, etc... in which you are using audio field type):

```
{% block extrahead %}
{{ block.super }}
```

```
    {% include "common_audiofield.html" %}
{% endblock %}
```

Then perform following commands to create the table and collect the static files:

```
./manage.py syncdb
```

and then:

```
./manage.py collectstatic
```

Create audiofield.log file:

```
touch /var/log/audio-field.log
```

Documentation

Extensive documentation is available on 'Read the Docs': http://django-audiofield.readthedocs.org

Contributing

If you've found a bug, implemented a feature or customized the template and think it is useful then please consider contributing. Patches, pull requests or just suggestions are welcome!

Source code: http://github.com/Star2Billing/django-audiofield

If you don't like Github and Git you're welcome to send regular patches.

Bug tracker: https://github.com/Star2Billing/django-audiofield/issues

License

django-audiofield is licensed under MIT, see *MIT-LICENSE.txt*.

Credit

Django-audiofield is a Star2Billing-Sponsored Community Project, for more information visit http://www.star2billing.com or email us at info@star2billing.com

CHAPTER 2

Installation overview

Install requirements

A requirements file stores a list of dependencies to be installed for your project/application.

To get started with Django-audiofield you must have the following installed:

- python >= 2.4 (programming language)
- Apache / http server with WSGI modules
- Django Framework >= 1.3 (Python based Web framework)
- Django-uuidfield

Install dependencies

Install dependencies on Debian:

```
apt-get -y install libsox-fmt-mp3 libsox-fmt-all mpg321 ffmpeg
```

Install dependencies on Redhat/CentOS:

```
yum -y install python-setuptools libsox-fmt-mp3 libsox-fmt-all mpg321 ffmpeg
```

Install requirements

Use PIP to install the dependencies listed in the requirments file,:

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

Configuration

Add `audiofield` into INSTALLED_APPS in settings.py:

```
INSTALLED_APPS = (
    ...
    'audiofield',
    ...)
```

Add the following code to your middleware:

```
MIDDLEWARE_CLASSES = (
    ...
    'audiofield.middleware.threadlocals.ThreadLocals',
)
```

If you are going to add customer audio form on your frontend, please add following:

```
# Frontend widget values
CHANNEL_TYPE_VALUE = 0  # 0-Keep original, 1-Mono, 2-Stereo

FREQ_TYPE_VALUE = 8000  # 0-Keep original, 8000-8000Hz, 16000-16000Hz, 22050-22050Hz,
                        # 44100-44100Hz, 48000-48000Hz, 96000-96000Hz

CONVERT_TYPE_VALUE = 0 # 0-Keep original, 1-Convert to MP3, 2-Convert to WAV, 3-
↪Convert to OGG
```

Run following commands:

```
python manage.py syncdb

python manage.py collectstatic
```

CHAPTER 3

Developer Documentation

Contents:

Prerequisites

To fully understand this project, developers will need to have an advanced knowledge of:

- Django : http://www.djangoproject.com/
- Python : http://www.python.org/

Objects Description

`AudioFile`

Describe model of the AudioFile

Attributes:

- `name` - given name to the audio file / not unique .
- `audio_file` - path and filename to the audio file.
- `user` - Attach user.
- `created_date` - record created date.
- `updated_date` - record updated date.

Name of DB table: audio_file

AudioFile Forms

Forms Definition

This form aims to be used in the django admin, support all the features for convertion per default:

```
class AdminAudioFileForm(ModelForm):
    class Meta:
        model = AudioFile
        fields = ['name', 'audio_file']
```

The following form aims to be used on frontend to power simple upload of audio files without convertion:

```
class CustomerAudioFileForm(ModelForm):
    audio_file = forms.FileField(widget=CustomerAudioFileWidget)
    class Meta:
        model = AudioFile
        fields = ['name', 'audio_file']
        exclude = ('user',)
```

Forms Usage

We provide you a simple example of using the forms to list and upload audio file on the frontend.

In url.py:

```
...
(r'^$', 'frontend.views.add_audio'),
```

In view.py:

```
...
@login_required
def add_audio(request):
    template = 'frontend/add_audio.html'
    form = CustomerAudioFileForm()

    # Add audio
    if request.method == 'POST':
        form = CustomerAudioFileForm(request.POST, request.FILES)
        if form.is_valid():
            obj = form.save(commit=False)
            obj.user = User.objects.get(username=request.user)
            obj.save()
            return HttpResponseRedirect('/')

        # To retain frontend widget, if form.is_valid() == False
        form.fields['audio_file'].widget = CustomerAudioFileWidget()

    data = {
       'audio_form': form,
    }

    return render_to_response(template, data,
           context_instance=RequestContext(request))
```

This is an other example how to edit the audiofield on the frontend.

In url.py:

```
...
(r'^edit/(.+)/$', 'frontend.views.edit_audio'),
```

In view.py:

```
...
@login_required
def edit_audio(request, object_id):

    obj = AudioFile.objects.get(pk=object_id)
    form = CustomerAudioFileForm(instance=obj)

    if request.GET.get('delete'):
        # perform delete
        if obj.audio_file:
            if os.path.exists(obj.audio_file.path):
                os.remove(obj.audio_file.path)
        obj.delete()
        return HttpResponseRedirect('/')

    if request.method == 'POST':
        form = CustomerAudioFileForm(request.POST, request.FILES, instance=obj)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect('/')

    template = 'frontend/edit_audio.html'

    data = {
       'audio_form': form,
    }

    return render_to_response(template, data,
           context_instance=RequestContext(request))
```

Test Case Descriptions

How to run tests

1. Run full test suite:

```
$ python manage.py test --verbosity=2
```

2. Run AudiofileTestCase:

```
$ python manage.py test audiofield.AudiofieldAdminInterfaceTestCase --verbosity=2
```

AudiofieldAdminInterfaceTestCase

Different test-cases of the audiofield

def test_admin_index() : Test Function to check Admin index page

def test_admin_audiofield() : Test Function to check Audiofield Admin pages

CHAPTER 4

Indices and tables

- genindex
- modindex
- search