django-avatar Documentation

Release 2.0

django-avatar developers

November 06, 2015

Contents

Django-avatar is a reusable application for handling user avatars. It has the ability to default to Gravatar if no avatar is found for a certain user. Django-avatar automatically generates thumbnails and stores them to your default file storage backend for retrieval later.

CHAPTER 1

Installation

If you have pip installed, you can simply run the following command to install django-avatar:

```
pip install django-avatar
```

Included with this application is a file named `setup.py`. It's possible to use this file to install this application to your system, by invoking the following command:

```
python setup.py install
```

Once that's done, you should be able to begin using django-avatar at will.

CHAPTER 2

Usage

To integrate `django-avatar` with your site, there are relatively few things that are required. A minimal integration can work like this:

1. List this application in the `INSTALLED_APPS` portion of your settings file. Your settings file will look something like:

```
INSTALLED_APPS = (
    # ...
    'avatar',
)
```

2. Update your database:

```
python manage.py syncdb
```

3. Add the avatar urls to the end of your root urlconf. Your urlconf will look something like:

```
urlpatterns = patterns('',
    # ...
    (r'^avatar/', include('avatar.urls')),
)
```

4. Somewhere in your template navigation scheme, link to the change avatar page:

```
<a href="{% url 'avatar_change' %}">Change your avatar</a>
```

5. Wherever you want to display an avatar for a user, first load the avatar template tags:

```
{% load avatar_tags %}
```

   Then, use the `avatar` tag to display an avatar of a default size:

```
{% avatar user %}
```

   Or specify a size (in pixels) explicitly:

```
{% avatar user 65 %}
```

CHAPTER 3

Template tags and filter

To begin using these template tags, you must first load the tags into the template rendering system:

{% load avatar_tags %}

{% avatar_url user [size in pixels] %} Renders the URL of the avatar for the given user. User can be either a `django.contrib.auth.models.User` object instance or a username.

{% avatar user [size in pixels] %} Renders an HTML `img` tag for the given user for the specified size. User can be either a `django.contrib.auth.models.User` object instance or a username.

{% render_avatar avatar [size in pixels] %} Given an actual `avatar.models.Avatar` object instance, renders an HTML `img` tag to represent that avatar at the requested size.

{{ request.user|has_avatar }} Given a user object returns a boolean if the user has an avatar.

CHAPTER 4

Global Settings

There are a number of settings available to easily customize the avatars that appear on the site. Listed below are those settings:

AVATAR_AUTO_GENERATE_SIZES An iterable of integers representing the sizes of avatars to generate on upload. This can save rendering time later on if you pre-generate the resized versions. Defaults to `(80,)`

AVATAR_RESIZE_METHOD The method to use when resizing images, based on the options available in PIL. Defaults to `Image.ANTIALIAS`.

AVATAR_STORAGE_DIR The directory under `MEDIA_ROOT` to store the images. If using a non-filesystem storage device, this will simply be appended to the beginning of the file name.

AVATAR_USERID_AS_USERDIRNAME By default, `User.username` will be use as directory name under `AVATAR_STORAGE_DIR`. If set to ``True, `User.id` wil be use instead of `User.username`. Usefull if user can change his username into app to avoid duplicate content. Defaults to `False`.

AVATAR_GRAVATAR_BACKUP A boolean determining whether to default to the Gravatar service if no `Avatar` instance is found in the system for the given user. Defaults to True.

AVATAR_DEFAULT_URL The default URL to default to if `AVATAR_GRAVATAR_BACKUP` is set to False and there is no `Avatar` instance found in the system for the given user.

AVATAR_GRAVATAR_FIELD The name of the user's field containing the gravatar email. Defaults to `email`. If you put, for example, `gravatar`, django-avatar will get the user's gravatar in `user.gravatar`.

AVATAR_ALLOWED_MIMETYPES Limit allowed avatar image uploads by their actual content payload and what image codecs we wish to support. This limits website user content site attack vectors against image codec buffer overflow and similar bugs. You must have python-imaging library installed. Suggested safe setting: `("image/png", "image/gif", "image/jpeg")`. When enabled you'll get the following error on the form upload *File content is invalid. Detected: image/tiff Allowed content types are: image/png, image/gif, image/jpg*.

AVATAR_MAX_SIZE File size limit for avatar upload. Default is `1024 * 1024` (1mb).

CHAPTER 5

Management Commands

This application does include two management command: `rebuild_avatars` and `migrate_avatars`.

rebuild_avatars It takes no arguments and, when run, re-renders all of the thumbnails for all of the avatars for the pixel sizes specified in the `AUTO_GENERATE_AVATAR_SIZES` setting.

migrate_avatars It takes no arguments and, when run, check all avatar userdirname folders. If folder doesn't correspond with actual userdirname pattern (affect with options like `AVATAR_USERID_AS_USERDIRNAME` or `AVATAR_HASH_USERDIRNAMES`), it move avatar files to the right place, call `rebuild_avatars` and delete useless folders.