diy-django Documentation

Release 1.0.0

Collin Anderson

May 13, 2015

Contents

https://github.com/collinanderson/diy-django

Contents:

CHAPTER 1

diy-django email auth

1.1 Django Username Email Auth User Model

Create an email auth user model like this:

```
from django.contrib.auth.models import AbstractBaseUser, PermissionsMixin
from django.db import models


class EmailUserManager(models.Manager):
    def get_by_natural_key(self, email):
        return self.get(email=email)


class EmailUser(AbstractBaseUser, PermissionsMixin):
    USERNAME_FIELD = 'email'
    email = models.EmailField(unique=True)
    is_staff = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    objects = EmailUserManager()

    def get_short_name(self):
        return self.email
```

1.2 Installing the Email Username Auth Model

In settings.py, set:

```
AUTH_USER_MODEL = 'myapp.EmailUser'
```

CHAPTER 2

diy-django ajax file upload progress widget

2.1 The Field, Widget, and Javascript

Create an UploadField and UploadWidget like this:

```
from django.forms import fields, widgets
from django.utils.safestring import mark_safe


class UploadWidget(widgets.TextInput):

    def render(self, name, value, attrs=None):
        attrs = attrs or {}
        attrs['class'] = 'vTextField'
        input = super(UploadWidget, self).render(name, value, attrs=attrs)
        value = ('<a href="%s" target="_blank">view</a>' % value.url) if value and getattr(value, 'u
        onchange = r'''
f = this.files[0]
this.value = ''
var xhr = new XMLHttpRequest()
xhr.open('POST', '/simple_upload/' + f.name, true)
xhr.setRequestHeader('X-CSRFToken', this.form.csrfmiddlewaretoken.value)
xhr.upload.status = this.parentNode.firstChild.nextElementSibling // <span>
xhr.upload.status.innerHTML = 'pending...'
xhr.upload.onprogress = function(e){ this.status.innerHTML = Math.round(100 * e.loaded / e.total) +
xhr.send(f)
xhr.onload = function(){
  this.upload.status.innerHTML = '<a href=\'/media/' + encodeURI(this.responseText) + '\' target=_bla
  this.upload.status.previousElementSibling.value = this.responseText // <input type=hidden>
}'''
        assert '"' not in onchange
        return mark_safe('<p>%s<span>%s</span><br><input type="file" onchange="%s" value="upload"></p


class UploadField(fields.CharField):
    widget = UploadWidget
```

2.2 The backend view to receive the file upload

In views.py:

```
from django.contrib.admin.views.decorators import staff_member_required
from django.core.files.base import ContentFile
from django.core.files.storage import default_storage
from django.http import HttpResponse


@staff_member_required
def simple_upload(request, name):
    return HttpResponse(default_storage.save(name, ContentFile(request.body)), 'text/plain')
```

In urls.py:

```
from . import views
urlpatterns = [
    url(r'^simple_upload/(.+)$', simple_upload),
]
```

2.3 Using the FileUpload field in the admin

In admin.py:

```
class MyAdmin(admin.ModelAdmin):
    formfield_overrides = {models.FileField: {'form_class': UploadField}}
```

CHAPTER 3

diy-django markdown

Install using pip:

```
pip install markdown
```

3.1 Using markdown with django models

Set up your models like this:

```
from django.utils.safestring import mark_safe
import markdown


class MyModel(models.Model):
    content = models.TextField()

    def content_markdown(self):
        return mark_safe(markdown.markdown(self.content))
```

In your template, render the markdown like so:

```
{{ my_object.content_markdown }}
```

3.2 Using markdown with django template tags

An alternative is to use a markdown templatetag:

```
from django import template
register = template.Library()

@register.filter
def markdown(value):
    import markdown
    return markdown.markdown(value)
markdown.is_safe = True
```

This can be used in a template like so:

```
{{ my_object.content|markdown }}
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search