EBook Gratis

APRENDIZAJE
django-forms

Free unaffiliated eBook created from
Stack Overflow contributors.

#django-forms

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: django-forms

It is an unofficial and free django-forms ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official django-forms.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con las formas django

Observaciones

Esta sección proporciona una descripción general de qué es django-forms y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de django-formas y vincular a los temas relacionados. Dado que la Documentación para django-forms es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar django-forms.

Lea Empezando con las formas django en línea: https://riptutorial.com/es/django-forms/topic/8924/empezando-con-las-formas-django

Capítulo 2: Django Formularios incorporados

Introducción

Django se envía con varias vistas que requieren formularios. Estas formas son, naturalmente, incorporadas. Un buen ejemplo son los formularios incorporados de autenticación .

Este tema pretende presentar documentación sobre cómo trabajar con estos formularios.

Examples

Añadir clases CSS personalizadas

Los formularios incorporados son excelentes, pero a veces es necesario personalizarlos, agregar nuevos campos o simplemente cambiar los atributos de CSS.

Este ejemplo es aplicable a varios casos de uso, pero aquí se presenta con respecto a PasswordChangeForm y su uso en un sitio web de Bootstrap .

La solución es crear otro formulario que inicie `PasswordChangeForm` actualizar el Widget :

```
class PasswordChangeCustomForm(PasswordChangeForm):
    def __init__(self, user, *args, **kwargs):
        super(PasswordChangeCustomForm, self).__init__(user,*args, **kwargs)
        for field in self.fields:
            self.fields[field].widget.attrs['class'] = 'form-control'
```

Si solo pretendes cambiar ciertos campos puedes hacerlo:

```
class PasswordChangeCustomForm(PasswordChangeForm):
    def __init__(self, user, *args, **kwargs):
        super(PasswordChangeCustomForm, self).__init__(user, *args, **kwargs)
        self.fields['old_password'].widget.attrs.update({'class': 'form-control'})
        self.fields['new_password1'].widget.attrs.update({'class': 'form-control'})
        self.fields['new_password2'].widget.attrs.update({'class': 'form-control'})
```

Nota: todos los formularios de bootstrap requieren el `form-control` clase para mantener el aspecto y la apariencia del sitio web.

Lea Django Formularios incorporados en línea: https://riptutorial.com/es/django-forms/topic/8927/django-formularios-incorporados

Capítulo 3: Pruebas

Introducción

Una de las características principales de Django son las pruebas unitarias.

Este tema pretende presentar una documentación completa sobre cómo probar los formularios.

Examples

Prueba simple

```
from django.test import TestCase
from myapp.forms import MyForm

class MyAppTests(TestCase):
    def test_forms(self):
        form_data = {'field1': 'fieldvalue1'}
        form = MyForm(data=form_data)
        self.assertTrue(form.is_valid())
```

Lea Pruebas en línea: https://riptutorial.com/es/django-forms/topic/8928/pruebas

Capítulo 4: Usando el formulario modelo

Introducción

Django ModelForm permite la creación de una clase de formulario a partir de un modelo de Django.

Examples

Usando el formulario modelo de Django con la vista basada en la clase de Django.

Django Model Form con Django Class Based view es una forma clásica de crear páginas para realizar / actualizar operaciones en la aplicación django rápidamente. Dentro del formulario podemos poner métodos para ejecutar tareas. Es una forma más limpia de poner las tareas en forma en lugar de poner en vistas / modelos.

Para dar un ejemplo utilizando el formulario de modelo de Django, primero debemos definir nuestro modelo.

```
class MyModel(models.Model):
   name = models.CharField(
          verbose_name = 'Name',
          max_length = 255)
```

Ahora hagamos un formulario usando este modelo:

```
class MyModelForm(forms.ModelForm):

    class Meta:
        model = MyModel
        fields = '__all__'
```

Permite agregar un método para imprimir hola mundo en él.

```
class MyModelForm(forms.ModelForm):
    class Meta:
        model = MyModel
        fields = '__all__'

    def print_hello_world(self):
         print('Hello World')
```

Vamos a hacer una plantilla para mostrar el formulario:

```
<form method="post" action="">
    {% csrf_token %}
<ul>
```

```
    {{ form.as_p }}
</ul>
<input type="submit" value="Submit Form"/>
</form>
```

Ahora usaremos este formulario en tres vistas diferentes que crearán y actualizarán las tareas respectivamente.

```
from django.views.generic.edit import CreateView, UpdateView
from myapp.models import MyModel

class MyModelCreate(CreateView):
    model = MyModel
    fields = ['name']
    form_class = MyModelForm
    template_name = 'my_template.html'

    def form_valid(self, form):
        # This method is called when valid form data has been POSTed.
        # It should return an HttpResponse.
        form.print_hello_world() # This method will print hello world in console
        return super(MyModelCreate, self).form_valid(form)


class MyModelUpdate(UpdateView):
    model = MyModel
    fields = ['name']
    form_class = MyModelForm
    template_name = 'my_template.html'
```

Ahora vamos a crear una URL para acceder a esas vistas.

```
from django.conf.urls import url
from myapp.views import MyModelCreate, MyModelUpdate

urlpatterns = [
    # ...
    url(r'mymodel/add/$', MyModelCreate.as_view(), name='author-add'),
    url(r'mymodel/(?P<pk>[0-9]+)/$', MyModelUpdate.as_view(), name='author-update')
]
```

Está bien, nuestro trabajo ha sido hecho. Podemos acceder a url: `localhost:8000/mymodel/add` para crear una entrada en el modelo. También acceda a `localhost:8000/mymodel/1` para actualizar esa entrada.

Haciendo campos no editables

Django `1.9` agregó el atributo Field.disabled :

> El argumento booleano deshabilitado, cuando se establece en Verdadero, deshabilita un campo de formulario usando el atributo HTML deshabilitado para que los usuarios no puedan editarlo. Incluso si un usuario manipula el valor del campo enviado al servidor, se ignorará a favor del valor de los datos iniciales del formulario.

Y así solo necesitas hacer:

```
MyChangeForm(ModelForm):

    def __init__(self, *args, **kwargs):
        super(MyChangeForm, self).__init__(*args, **kwargs)
        self.fields['<field_to_disable>'].disabled = True
```

Y creando la forma que necesitas:

```
MyChangeForm(initial={'<field_to_disable>': "something"})
```

Antes de la versión `1.9` tenías que:

```
class MyChangeForm(ModelForm):
    def __init__(self, *args, **kwargs):
        super(ItemForm, self).__init__(*args, **kwargs)
        instance = getattr(self, 'instance', None)
    if instance and instance.id:
        self.fields['<field_to_disable>'].required = False
        self.fields['<field_to_disable>'].widget.attrs['disabled'] = True

def clean_<field_to_disable>(self):
    # As shown in the above answer.
    instance = getattr(self, 'instance', None)
    if instance:
        return instance.<field_to_disable>
    else:
        return self.cleaned_data.get('<field_to_disable>', None)
```

Y creando la forma que necesitas:

```
MyChangeForm(instance=MyChange.objects.get_or_create(<field_to_disable>="something"))
```

Este ejemplo se basó en esta pregunta .

Lea Usando el formulario modelo en línea: https://riptutorial.com/es/django-forms/topic/8926/usando-el-formulario-modelo

Creditos

S. No	Capítulos	Contributors
1	Empezando con las formas django	Community
2	Django Formularios incorporados	NBajanca
3	Pruebas	NBajanca
4	Usando el formulario modelo	NBajanca, ruddra