
python™
django

Contenido

- Templates
- Estáticos
- Formularios - Parte 1

Templates

Al ser un framework web, Django necesita una forma conveniente de generar HTML de forma dinámica. El enfoque más común se basa en plantillas. Una plantilla contiene las partes estáticas de la salida HTML deseada, así como una sintaxis especial que describe cómo se insertará el contenido dinámico.

Un proyecto de Django se puede configurar con uno o varios motores de plantilla. Django envía backends integrados para su propio sistema de plantillas, llamado creativamente el lenguaje de plantillas Django (DTL).

Templates

Al ser un framework web, Django necesita una forma conveniente de generar HTML de forma dinámica. El enfoque más común se basa en plantillas. Una plantilla contiene las partes estáticas de la salida HTML deseada, así como una sintaxis especial que describe cómo se insertará el contenido dinámico.

Un proyecto de Django se puede configurar con uno o varios motores de plantilla. Django envía backends integrados para su propio sistema de plantillas, llamado creativamente el lenguaje de plantillas Django (DTL).

Lenguaje de template de Django (Variables)

Las variables están son contenidas por {{y}} así:

```
My first name is {{ first_name }}. My last name is {{ last_name }}.
```

Con un contexto de {'first_name': 'John', 'last_name': 'Doe'}, esta plantilla se representa en:

```
My first name is John. My last name is Doe.
```

Uso de diccionarios, objetos o listas

```
{{ my_dict.key }}
{{ my_object.attribute }}
{{ my_list.0 }}
```

Lenguaje de template de Django (Tags)

Las etiquetas están contenidos por {% y %} así:

```
{% csrf_token %}
```

Los tag aceptan argumentos

```
{% cycle 'odd' 'even' %}
```

Algunos requieren inicio y fin

```
{% if user.is_authenticated %}Hello, {{ user.username }}.{% endif %}
```

Lenguaje de template de Django (Filtros)

Los filtros transforman los valores de las variables y los argumentos de los tags.

```
{{ django|title }}
```

Con un contexto de {'django': 'the web framework for perfectionists with deadlines'}, esta plantilla se renderiza a:

```
The Web Framework For Perfectionists With Deadlines
```

Algunos filtros toman un argumento:

```
{{ my_date|date:"Y-m-d" }}
```

Lenguaje de template de Django (Comentarios)

Se ven así

```
{# this won't be rendered #}
```

Se puede usar el tag comment para incluir múltiples líneas

```
{% comment %}
this won't be rendered
this won't be rendered
{% endcomment %}
```

Estáticos

Los sitios web generalmente necesitan entregar archivos adicionales como imágenes, JavaScript o CSS. En Django, nos referimos a estos archivos como "archivos estáticos". Django proporciona ***django.contrib.staticfiles*** para ayudarlo a administrarlos.

Configuración en settings.py

```
STATIC_URL = 'static/'
```

```
STATICFILES_DIRS = [
    BASE_DIR / "static",
    '/var/www/static/',
]
```

Uso en templates

```
{% load static %}
<img src="{% static 'my_app/example.jpg' %}" alt="My image">
```

Estáticos

Para desarrollar se recomienda usar el mismo motor de Django agregando estáticos en url

```
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    # ... the rest of your URLconf goes here ...
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

En Producción:

En producción se recomienda usar el servidor web para servir los estáticos, no usar django para servir estos archivos, una herramienta util es collecstatic para organizarlos fácilmente.

Formularios

Django cuenta con un módulo completo que nos permite crear formulario, validar su contenido, conectar estos formularios con bases de datos, etc. (Documentación)

Ejemplo de formulario básico:

```
from django import forms


class CommentForm(forms.Form):
    name = forms.CharField(initial='class')
    url = forms.URLField()
    comment = forms.CharField()
```

Formularios - Guardado

```python
from django import forms

from .models import Comment


class CommentForm(forms.Form):
    name = forms.CharField(initial='class')
    url = forms.URLField()
    comment = forms.CharField()

    def save(self, commit=True):
        instance = Comment(self.name, self.url, self.comment)
        if commit:
            instance.save()
        return instance
```

Formularios - Validaciones

Los formularios en base a los campos que se le incorporan realizan validaciones al llamar al método *is_valid()* del formulario el cual se puede personalizar, así como también la validación de cada campo por medio del método *clean_<nombre del campo>()*. (Documentación)

```python
from django import forms


class CommentForm(forms.Form):
    name = forms.CharField(initial='class')
    url = forms.URLField()
    comment = forms.CharField()

    def clean_name(self):
        data = self.cleaned_data["name"]
        if data == "":
            raise ValidationError("El nombre está vacío.")
        return data
```

Formularios - Formularios basados en Modelos

Dentro del modulo de formularios, existe el *ModelForm*, Esta clase permite crear formularios basados en modelos, que van a validar los datos según el modelo, guardar en el, etc. (Documentación)

```
from django.forms import ModelForm
from myapp.models import Article


class ArticleForm(ModelForm):
     class Meta:
         model = Article
         fields = ['pub_date', 'headline', 'content', 'reporter']
         exclude = [update_date, 'insert_date']
```

Formularios - Widgets

Los el modulo formularios permite trabajar con distintos tipos de widgets que van a ser los componentes que van a ver los usuarios para interactuar con el formulario (Documentación).

```
from django import forms
from django.forms import ModelForm, Textarea
from .models import Author

YEAR_CHOICES = ['1980', '1981', '1982']
COLORS_CHOICES = [('blue', 'Blue'),  ('green', 'Green'),
                          ('black', 'Black')]

class AuthorForm(ModelForm):
    year = forms.DateField(widget=forms.SelectDateWidget(years=YEAR_CHOICES))
    favorite_colors = forms.MultipleChoiceField(required=False, widget=forms.CheckboxSelectMultiple,
                                                choices=COLORS_CHOICES,)
    class Meta:
        model = Author
        fields = ('name', 'title', 'birth_date')
        widgets = {'name': Textarea(attrs={'cols': 80, 'rows': 20}),}
```


django