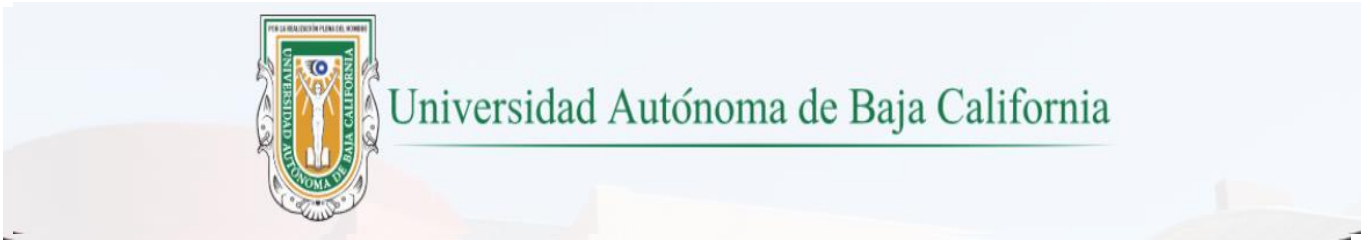

PLANTILLAS DJANGO

PYTHON

LOPEZ HERNANDEZ
ROGELIO

372

Al ser un marco web, Django necesita una forma conveniente de generar HTML dinámicamente. El enfoque más común se basa en las plantillas. Una plantilla contiene las partes estáticas de la salida HTML deseada, así como también una sintaxis especial que describe cómo se insertará el contenido dinámico. Para ver un ejemplo práctico de cómo crear páginas HTML con plantillas, consulte el Tutorial 3 .

Un proyecto de Django se puede configurar con uno o varios motores de plantillas (o incluso cero si no usa plantillas). Django incluye backends integrados para su propio sistema de plantillas, llamado creativamente el lenguaje de plantillas de Django (DTL), y para la popular alternativa Jinja2 . Backends para otros idiomas de la plantilla puede estar disponible de terceros.

Django define una API estándar para cargar y renderizar plantillas independientemente del backend. La carga consiste en encontrar la plantilla para un identificador dado y preprocesarla, generalmente compilándola en una representación en memoria. Representación significa interpolar la plantilla con datos de contexto y devolver la cadena resultante.

El lenguaje de plantillas de Django es el propio sistema de plantillas de Django. Hasta que Django 1.8 era la única opción integrada disponible. Es una buena biblioteca de plantillas a pesar de que es bastante crítica y tiene algunas idiosincrasias. Si no tiene una razón urgente para elegir otro backend, debe usar la DTL, especialmente si está escribiendo una aplicación conectable y tiene la intención de distribuir plantillas. Las aplicaciones de contribución de Django que incluyen plantillas, como django.contrib.admin , usan la DTL.

Por razones históricas, tanto el soporte genérico para los motores de plantillas como la implementación del lenguaje de plantillas de Django viven en el **django.template** espacio de nombres.

Configuración

Los motores de plantillas están configurados con la **TEMPLATES** configuración. Es una lista de configuraciones, una para cada motor. El valor predeterminado está vacío. El **settings.py** generado por el **startproject** comando define un valor más útil:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            # ... some options here ...
        },
    },
]
```

BACKEND es una ruta de Python punteada a una clase de motor de plantilla que implementa la API de backend de plantilla de Django. Los backends incorporados son **django.template.backends.django.DjangoTemplates**y**django.template.backends.jinja2.Jinja2**.

Como la mayoría de los motores cargan plantillas de archivos, la configuración de nivel superior para cada motor contiene dos configuraciones comunes:

- **DIRS** define una lista de directorios donde el motor debe buscar archivos de origen de plantilla, en orden de búsqueda.
- **APP_DIRS** Indica si el motor debe buscar plantillas dentro de las aplicaciones instaladas. Cada backend define un nombre convencional para el subdirectorio dentro de las aplicaciones donde se deben almacenar sus plantillas.

Si bien es poco común, es posible configurar varias instancias del mismo backend con diferentes opciones. En ese caso debes definir un único **NAME**para cada motor.

OPTIONS contiene configuraciones específicas del backend.

Uso

El **django.template.loader**módulo define dos funciones para cargar plantillas.

get_template(***template_name*** **,** ***utilizando*** **=** ***Ninguno*** **) [fuente]**

> Esta función carga la plantilla con el nombre dado y devuelve un **Template**objeto.
>
> El tipo exacto del valor de retorno depende del backend que cargó la plantilla. Cada backend tiene su propia **Template**clase.
>
> **get_template()**Prueba cada motor de plantillas en orden hasta que uno tenga éxito. Si no se puede encontrar la plantilla, se eleva**TemplateDoesNotExist**. Si se encuentra la plantilla pero contiene una sintaxis no válida, se eleva **TemplateSyntaxError**.
>
> La forma en que se buscan y se cargan las plantillas depende del backend y la configuración de cada motor.
>
> Si desea restringir la búsqueda a un motor de plantilla en particular, pase el motor **NAME**en el **using**argumento.

select_template(***template_name_list*** **,** ***utilizando*** **=** ***Ninguno*** **) [fuente]**

> **select_template()**es como **get_template()**, excepto que toma una lista de nombres de plantillas. Intenta cada nombre en orden y devuelve la primera plantilla que existe.

Si falla la carga de una plantilla **django.template**, pueden surgir las siguientes dos excepciones, definidas en :

excepción__**TemplateDoesNotExist (*msg* , *intento* = *Ninguno* , *backend* = *Ninguno* , *cadena* = *Ninguno*) [fuente]**

Esta excepción se produce cuando no se puede encontrar una plantilla. Acepta los siguientes argumentos opcionales para rellenar la plantilla postmortem en la página de depuración:

backend

La instancia de backend de plantilla desde la que se originó la excepción.

tried

Una lista de fuentes que se probaron al encontrar la plantilla. Esto se formatea como una lista de tuplas que contiene , donde es un objeto similar al origen y es una cadena con la razón por la que no se encontró la plantilla.**(origin, status)originstatus**

chain

Una lista de **TemplateDoesNotExist** excepciones intermedias generadas al intentar cargar una plantilla. Esto es usado por funciones, como por ejemplo **get_template()**, que intentan cargar una plantilla dada desde múltiples motores.

***excepción*TemplateSyntaxError (*msg*) [fuente]**

Esta excepción se produce cuando se encuentra una plantilla pero contiene errores.

Template

Los objetos devueltos por **get_template()**y **select_template()** deben proporcionar un **render()**método con la siguiente firma:

Template.render(*contexto* = *Ninguno* , *solicitud* = *Ninguno*) ¶

Representa esta plantilla con un contexto dado.

Si **context**se proporciona, debe ser un **dict**. Si no se proporciona, el motor renderizará la plantilla con un contexto vacío.

Si **request**se proporciona, debe ser un **HttpRequest**. Luego, el motor debe hacerlo, así como el token CSRF, disponible en la plantilla. Cómo se logra esto depende de cada backend.

Bibliografía

https://docs.djangoproject.com/es/2.1/intro/plantillas/