Contenidos a Trabajar

1. Herencia de Plantillas
   - A tener en cuenta
   - Herencia de Plantillas
2. Gestión de Archivos Estáticos (Imágenes, JS, Css, etc.)
   - Configurar Archivos Estáticos
   - Sirviendo archivos
   - Despligue

Herencia de plantillas

El lenguaje de plantillas de Django está diseñado para lograr un equilibrio entre potencia y facilidad. Está diseñado para que aquellos que están acostumbrados a trabajar con HTML se sientan cómodos. Si está expuesto a otros lenguajes de plantilla basados en texto, como Smarty o Jinja2 , debería sentirse como en casa con las plantillas de Django.

A tener en cuenta:

Si tiene experiencia en programación, o si está acostumbrado a lenguajes que mezclan código de programación directamente en HTML, querrá tener en cuenta que el sistema de plantillas de Django no es simplemente Python incrustado en HTML. Esto es así por diseño: el sistema de plantillas está destinado a expresar la presentación, no la lógica del programa.

El sistema de plantillas de Django proporciona etiquetas que funcionan de manera similar a algunas construcciones de programación (una etiqueta **if** para pruebas booleanas, una etiqueta **for** para bucles, etc.), pero estas no se ejecutan simplemente como el código Python correspondiente, y el sistema de plantillas no ejecutará expresiones arbitrarias de Python. Solo las etiquetas, los filtros y la sintaxis que se enumeran a continuación son compatibles de forma predeterminada (aunque puede agregar sus propias extensiones al idioma de la plantilla según sea necesario).

Herencia de plantillas

La parte más poderosa, y por lo tanto la más compleja, del motor de plantillas de Django es la herencia de plantillas. La herencia de plantillas le permite crear una plantilla base «esqueleto» que contiene todos loselementos comunes de su sitio y define **bloques** que las plantillas secundarias pueden anular.

Veamos la herencia de plantillas comenzando con un ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <link rel="stylesheet" href="style.css">
        <title>{% block title %}My amazing site{% endblock %}</title>
</head>

<body>
        <div id="sidebar">
                {% block sidebar %}
                <ul>
                        <li><a href="/">Home</a></li>
                        <li><a href="/blog/">Blog</a></li>
                </ul>
                {% endblock %}
        </div>

        <div id="content">
                {% block content %}{% endblock %}
        </div>
</body>
</html>
```

Esta plantilla, a la que llamaremos **base.html**, define un documento de esqueleto HTML que puede usar parauna página de dos columnas. El trabajo de las plantillas «hijas» es llenar los bloques vacíos con contenido.

En este ejemplo, la etiqueta **block** define tres bloques que las plantillas secundarias pueden completar. Todo lo que hace la etiqueta **block** es decirle al motor de plantillas que una plantilla secundaria puede anular esas partes de la plantilla.

Una plantilla secundaria podría verse así:

```
{% extends "base.html" %}
{% block title %}My amazing blog{% endblock %}
{% block content %}
{% for entry in blog_entries %}
        <h2>{{ entry.title }}</h2>
        <p>{{ entry.body }}</p>
{% endfor %}
{% endblock %}
```

La etiqueta **extends** es la clave aquí. Le dice al motor de plantillas que esta plantilla "extiende" otra plantilla. Cuando el sistema de plantillas evalúa esta plantilla, primero localiza la plantilla base, en este caso,

《base.html》.

En ese momento, el motor de plantillas notará las tres etiquetas **block** en **base.html** y reemplazará esos bloques con el contenido de la plantilla secundaria. Dependiendo del valor de **blog_entries**, la salida podríaverse como:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <link rel="stylesheet" href="style.css">
        <title>My amazing blog</title>
</head>
<body>
        <div id="sidebar">
                <ul>
                        <li><a href="/">Home</a></li>
                        <li><a href="/blog/">Blog</a></li>
                </ul>
        </div>
        <div id="content">
                <h2>Entry one</h2>
                <p>This is my first entry.</p>
```

```
            <h2>Entry two</h2>
            <p>This is my second entry.</p>
        </div>
    </body>
    </html>
```

Tenga en cuenta que dado que la plantilla secundaria no definió el bloque **sidebar**, en su lugar se usa el valorde la plantilla principal. El contenido dentro de una etiqueta **{% block %}** en una plantilla principal siempre seusa como respaldo.

Puede utilizar tantos niveles de herencia como necesite. Una forma común de usar la herencia es el siguienteenfoque de tres niveles:

- Cree una plantilla **base.html** que contenga la apariencia principal de su sitio.
- Crea una plantilla **base_SECTIONNAME.html** para cada «sección» de tu sitio. Por ejemplo, **base_news.html**, **base_sports.html**. Todas estas plantillas amplían **base.html** e incluyenestilos/diseños específicos de la sección.
- Cree plantillas individuales para cada tipo de página, como un artículo de noticias o una entrada de blog. Estas plantillas amplían la plantilla de la sección correspondiente.

Este enfoque maximiza la reutilización del código y ayuda a agregar elementos a las áreas de contenidocompartido, como la navegación en toda la sección.

Estos son algunos consejos para trabajar con la herencia:

- Si usa **{% extends %}** en una plantilla, debe ser la primera etiqueta de plantilla en esa plantilla. De lo contrario, la herencia de plantillas no funcionará.
- Más etiquetas **{% block %}** en sus plantillas base son mejores. Recuerde, las plantillas secundarias no tienen que definir todos los bloques principales, por lo que puede completar valores predeterminados razonables en una cantidad de bloques y luego definir solo los que necesita más adelante. Es mejor tener más anzuelos que menos anzuelos.
- Si se encuentra duplicando contenido en varias plantillas, probablemente signifique que debe moverse contenido a una plantilla principal.
- Si necesita obtener el contenido del bloque de la plantilla principal, la variable **{{ block.super }}** hará el truco. Esto es útil si desea agregar al

contenido de un bloque principal en lugar de anularlo por completo. Los datos insertados usando **{{ block.super }}** no se escaparán automáticamente, ya que ya se escaparon, si es necesario, en la plantilla principal.

- Al usar el mismo nombre de plantilla que hereda, **{% extends %}** se puede usar para heredar una plantilla al mismo tiempo que se anula. En combinación con **{{ block.super }}**, esta puede ser una forma eficaz de realizar pequeñas personalizaciones.
- Para mayor legibilidad, se puede dar un nombre a la etiqueta **{% endblock %}**. Por ejemplo:

```
{% block content %}
...
{% endblock content %}
```

Finalmente, tenga en cuenta que no puede definir varias etiquetas **block** con el mismo nombre en la misma plantilla. Esta limitación existe porque una etiqueta de bloque funciona en «ambas» direcciones. Es decir, una etiqueta de bloque no solo proporciona un hueco para llenar, sino que también define el contenido que llena el hueco en el *archivo principal*. Si hubiera dos etiquetas **block** con nombres similares en una plantilla, el padre de esa plantilla no sabría cuál de los contenidos de los bloques usar.

Para más información sobre platillas diríjase al a documentación oficial en: https://docs.djangoproject.com/es/3.2/ref/templates/language/

Además, en caso de querer obtener información sobre las buenas prácticas en la utilización de plantillas, lerecomendamos revisar el libro que se referencia.

Referencias: Daniel, Feldroy. Audrey, Feldroy. **Two Scoops of Django 3.x. Best practices for the Django Web Framework**. Alpha Release.

Gestión de Archivos Estáticos (Imágenes, JS, Css, Etc)

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

Aquí podrá ver cómo puede servir estos archivos estáticos.

Configurar Archivos Estáticos

1. Asegúrese de que esté **django.contrib.staticfiles** incluido en su archivo **INSTALLED_APPS**.
2. En su archivo de configuración, defina **STATIC_URL**, por ejemplo: STATIC_URL = '/static/'
3. En sus plantillas, use la etiqueta **static** de plantilla para crear la URL para la ruta relativa dada usando el archivo **STATICFILES_STORAGE**.

```
{% load static %}

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

4. Almacene sus archivos estáticos en una carpeta llamada **static** en su aplicación. **my_app/static/my_app/example.jpg** por ejemplo

Es probable que su proyecto también tenga activos estáticos que no estén vinculados a una aplicación en particular. Además de usar un directorio **static/** dentro de sus aplicaciones, puede definir una lista de directorios (**STATICFILES_DIRS**) en su archivo de configuración donde Django también buscará archivos estáticos. Por ejemplo:

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

Consulte la documentación de la configuración STATICFILES_FINDERS para obtener detalles sobre cómo staticfiles encuentra sus archivos.

Sirviendo archivos

Además de estos pasos de configuración, también deberá entregar los archivos estáticos.

Durante el desarrollo, si usa **django.contrib.staticfiles**, esto se hará automáticamente ejecutando runservercuando DEBUG esté configurado en True (ver django.contrib.staticfiles.views.serve()).

Este método es sumamente ineficaz y probablemente inseguro, por lo que no es adecuado para un despliegueen producción.

Despliegue

django.contrib.staticfiles proporciona un comando de administración de conveniencia para recopilar archivos estáticos en un solo directorio para que pueda servirlos fácilmente.

1. Establezca la configuración **STATIC_ROOT** en el directorio desde el que le gustaría servir estosarchivos, por ejemplo:

   ```
   STATIC_ROOT = "/var/www/example.com/static/"
   ```

2. Ejecute el comando **collectstatic** de administración:

   ```
   $ python manage.py collectstatic
   ```

   Esto copiará todos los archivos de sus carpetas estáticas en el directorio **STATIC_ROOT**.

3. Utilice un servidor web de su elección para servir los archivos. La implementación de archivos estáticos cubre algunas estrategias de implementación comunes para archivos estáticos.

Para más información sobre el manejo de archivos estáticos, diríjase la documentación oficial en:https://docs.djangoproject.com/en/3.2/howto/static-files/