Django-Tailwind

Configurações iniciais

Instalação

Passo a Passo da Instalação

Instale o pacote django-tailwind via pip:

```
python -m pip install django-tailwind
```

Adicione 'tailwind' ao INSTALLED_APPS no settings.py:

```
INSTALLED_APPS = [
  # outras aplicações do Django
  'tailwind',
]
```

Crie uma aplicação Django compatível com Tailwind CSS, gosto de chamá-la de theme:

```
python manage.py tailwind init
```

Adicione sua nova aplicação 'theme' ao INSTALLED_APPS no settings.py:

```
INSTALLED_APPS = [
  # outras aplicações do Django
  'tailwind',
  'theme'
]
```

Registre a aplicação 'theme' gerada adicionando a seguinte linha ao arquivo settings.py:

```
TAILWIND_APP_NAME = 'theme'
```

Verifique se a lista INTERNAL_IPS está presente no arquivo settings.py e contém o endereço IP 127.0.0.1:

```
INTERNAL_IPS = [
    "127.0.0.1",
]
```

Instale as dependências do Tailwind CSS, executando o seguinte comando:

```
python manage.py tailwind install
```

Se você não estiver usando o base.html que vem com o Django Tailwind, adicione {% tailwind_css %} ao base.html:

```
{% load static tailwind_tags %}
...
<head>
   ...
   {% tailwind_css %}
   ...
</head>
```

Adicione e configure também o django_browser_reload, que cuida das atualizações automáticas de página e CSS no modo de desenvolvimento. Adicione-o ao INSTALLED_APPS no settings.py:

```
INSTALLED_APPS = [
  # outras aplicações do Django
  'tailwind',
  'theme',
  'django_browser_reload'
]
```

Mantendo-se no settings.py, adicione o middleware:

```
MIDDLEWARE = [
  # ...
  "django_browser_reload.middleware.BrowserReloadMiddlewar
```

```
e",
  # ...
]
```

O middleware deve ser listado após qualquer middleware que codifique a resposta, como o GZipMiddleware do Django. O middleware insere automaticamente a tag de script necessária em respostas HTML antes de </body> quando DEBUG for True.

Inclua a URL django_browser_reload no seu arquivo urls.py raiz:

```
from django.urls import include, path
urlpatterns = [
    ...,
    path("__reload__/", include("django_browser_reload.url
s")),
]
```

Finalmente, você deve ser capaz de usar as classes Tailwind CSS no HTML. Inicie o servidor de desenvolvimento executando o seguinte comando no seu terminal:

```
python manage.py tailwind start
```

Configurações Opcionais

Configuração de Regras de Conteúdo (anteriormente Purge)

A seção de conteúdo do arquivo tailwind.config.js é onde você configura os caminhos para todos os seus modelos HTML, componentes JavaScript e qualquer outro arquivo de origem que contenha nomes de classe Tailwind.

Dependendo da estrutura do seu projeto, você pode precisar configurar as regras de conteúdo em tailwind.config.js. Este arquivo está na pasta static_src do aplicativo de tema criado por python manage.py tailwind init {APP_NAME}.

Por exemplo, o arquivo theme/static_src/tailwind.config.js pode se parecer com isso:

```
module.exports = {
    content: [
        // Modelos dentro do aplicativo de tema (por exemplo, base.html)
        '../templates/**/*.html',
        // Modelos em outros aplicativos
        '../../templates/**/*.html',
        // Ignorar arquivos em node_modules
        '!../../**/node_modules',
        // Incluir arquivos JavaScript que podem conter classes Tailwind CSS
        '../../**/*.js',
        // Incluir arquivos Python que podem conter classes Tailwind CSS
        '../../**/*.py'
    ],
    ...
}
```

Observe que você pode precisar ajustar esses caminhos para se adequar à sua estrutura de projeto específica. É crucial garantir que todos os arquivos HTML (ou arquivos que contenham conteúdo HTML, como arquivos .

Considerações Finais

- **Desenvolvimento Ativo**:
  Durante o desenvolvimento, é importante manter o comando `python manage.py tailwind start` em execução para que o Tailwind CSS recompile automaticamente sempre que você fizer alterações nos arquivos CSS ou nos templates Django que utilizam classes do Tailwind.
- **Build para Produção**:
  Quando você estiver pronto para a produção, você pode construir o CSS otimizado usando:

  ```
  python manage.py tailwind build
  ```

Quando você executa `python manage.py tailwind build`, ele realiza várias tarefas:

1. **Compilação do CSS**:

- O Tailwind CSS pega todos os seus arquivos fonte e gera um único arquivo CSS que contém apenas as classes utilizadas no seu projeto. Isso é feito através da análise dos seus templates Django, arquivos JavaScript, e quaisquer outros arquivos que possam conter classes do Tailwind.

2. **Minificação**:
    - O arquivo CSS resultante é minificado. Minificação é o processo de remover todos os espaços em branco desnecessários, quebras de linha, e outras informações não essenciais do arquivo CSS para reduzir o tamanho do arquivo.
3. **PurgeCSS**:
    - O PurgeCSS é uma ferramenta que remove todas as classes CSS que não estão sendo utilizadas no seu projeto. Isso é extremamente útil para reduzir o tamanho do arquivo CSS, já que o Tailwind CSS, por padrão, gera um arquivo muito grande contendo todas as classes possíveis.