# Django

Documentação

## Gerenciando arquivos estáticos (por exemplo, imagens, JavaScript, CSS)

Os sites geralmente precisam fornecer arquivos adicionais, como imagens, JavaScript ou CSS. No Django, nos referimos a esses arquivos como “arquivos estáticos”. Django fornece **`django.contrib.staticfiles`**ajuda para gerenciá-los.

Esta página descreve como você pode servir esses arquivos estáticos.

### Configurando arquivos estáticos

1. Certifique-se de que **`django.contrib.staticfiles`**está incluído no seu **`INSTALLED_APPS`**.
2. Em seu arquivo de configurações, defina **`STATIC_URL`**, por exemplo:

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

3. Em seus modelos, use a **`static`**tag de modelo para construir a URL para o caminho relativo fornecido usando o configurado **`STATICFILES_STORAGE`**.

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

4. Armazene seus arquivos estáticos em uma pasta chamada **`static`**em seu aplicativo. Por exemplo **`my_app/static/my_app/example.jpg`**.

**Servindo os arquivos**

Além dessas etapas de configuração, você também precisará servir os arquivos estáticos.

Durante o desenvolvimento, se você usar **`django.contrib.staticfiles`**, isso será feito automaticamente **`runserver`**quando **`DEBUG`**for definido como **`True`**(consulte **`django.contrib.staticfiles.views.serve()`**).

Este método é **extremamente ineficiente** e provavelmente **inseguro** , portanto, não é **adequado para produção** .

Consulte Implementando arquivos estáticos para estratégias adequadas para servir arquivos estáticos em ambientes de produção.

Your project will probably also have static assets that aren’t tied to a particular app. In addition to using a **`static/`** directory inside your apps, you can define a list of directories (**`STATICFILES_DIRS`**) in your settings file where Django will also look for static files. For example:

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
    '/var/www/static/',
]
```

See the documentation for the **`STATICFILES_FINDERS`** setting for details on how **`staticfiles`** finds your files.

**Static file namespacing**

Now we *might* be able to get away with putting our static files directly in **my_app/static/** (rather than creating another **my_app** subdirectory), but it would actually be a bad idea. Django will use the first static file it finds whose name matches, and if you had a static file with the same name in a *different* application, Django would be unable to distinguish between them. We need to be able to point Django at the right one, and the easiest way to ensure this is by *namespacing* them. That is, by putting those static files inside *another* directory named for the application itself.

## Serving static files during development

If you use **django.contrib.staticfiles** as explained above, **runserver** will do this automatically when **DEBUG** is set to **True**. If you don't have **django.contrib.staticfiles** in **INSTALLED_APPS**, you can still manually serve static files using the **django.views.static.serve()** view.

This is not suitable for production use! For some common deployment strategies, see Deploying static files.

For example, if your **STATIC_URL** is defined as **/static/**, you can do this by adding the following snippet to your urls.py:

```
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)
```

**Note**

This helper function works only in debug mode and only if the given prefix is local (e.g. **/static/**) and not a URL (e.g. **http://static.example.com/**).

Also this helper function only serves the actual **STATIC_ROOT** folder; it doesn't perform static files discovery like **django.contrib.staticfiles**.

## Serving files uploaded by a user during development

During development, you can serve user-uploaded media files from **MEDIA_ROOT** using the **django.views.static.serve()** view.

This is not suitable for production use! For some common deployment strategies, see Deploying static files.

For example, if your **MEDIA_URL** is defined as **/media/**, you can do this by adding the following snippet to your urls.py:

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

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

**Note**

This helper function works only in debug mode and only if the given prefix is local (e.g. **/media/**) and not a URL (e.g. **http://media.example.com/**).



## Testing

When running tests that use actual HTTP requests instead of the built-in testing client (i.e. when using the built-in **LiveServerTestCase**) the static assets need to be served along the rest of the content so the test environment reproduces the real one as faithfully as possible, but **LiveServerTestCase** has only very basic static file-serving functionality: It doesn't know about the finders feature of the **staticfiles** application and assumes the static content has already been collected under **STATIC_ROOT**.

Because of this, **staticfiles** ships its own **django.contrib.staticfiles.testing.StaticLiveServerTestCase**, a subclass of the built-in one that has the ability to transparently serve all the assets during execution of these tests in a way very similar to what we get at development time with **DEBUG = True**, i.e. without having to collect them using **collectstatic** first.

## Deployment

**django.contrib.staticfiles** provides a convenience management command for gathering static files in a single directory so you can serve them easily.

1. Set the **STATIC_ROOT** setting to the directory from which you'd like to serve these files, for example:

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

2. Run the **collectstatic** management command:

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

   This will copy all files from your static folders into the **STATIC_ROOT** directory.

3. Use a web server of your choice to serve the files. Deploying static files covers some common deployment strategies for static files.

## Learn more

This document has covered the basics and some common usage patterns. For complete details on all the settings, commands, template tags, and other pieces included in **django.contrib.staticfiles**, see the staticfiles reference.

**Overriding templates**

**Deploying static files**

Contribute to Django

Submit a Bug

Report a Security Issue

---

**Follow Us**

GitHub

Twitter

News RSS

Django Users Mailing List

---

**Support Us**

Sponsor Django

Official merchandise store

Amazon Smile

Benevity Workplace Giving Program