How to put Django app into maintenance mode

Putting a Django app into maintenance mode is important when updating, fixing bugs, or making significant changes to your app. Here are the details on how to do this.

Putting a Django app into maintenance mode is important when updating, fixing bugs, or making significant changes to your app. Here are the details on how to do this.

By temporarily restricting user access and displaying a maintenance page, you can deliver important messages, ensure a smooth update process, and prevent potential conflicts or data loss.

Whether you're currently a programmer or a system administrator, understanding how to implement maintenance mode in Django will allow you to maintain a user-friendly and reliable application.

How to use the Django-Maintenace-Mode . package

Thanks to support from the vast community, Django offers a wide range of packages that can greatly enhance your development workflow, allowing you to work faster and more efficiently. These packages reduce the burden of having to repeat tasks, ensuring a smoother experience for developers.

One of the packages provided by Django is the **django-maintenance-mode** package that you can use to put your Django app into maintenance mode. **The django-maintenance-mode** package works by displaying a page for the HTTP 503 status code. You can use **django-maintenance-mode** in your application by following these steps:

Step 1: Install django-maintenance-mode in virtual environment

1. In the project's virtual environment, install this package with Python's pip package manager. Run the command below in the command line interface (CLI):

```
pip install django-maintenance-mode
```

2. After installing this package, add **maintenace_mode** to the list of **INSTALLED_APPS** in the **settings.py** file :

```
INSTALLED_APPS = [ # m?t s? app khác, 'maintenance_mode', ]
```

3. Next, add middleware for **django-maintenance-mode** to **MIDDLEWARE** in **settings.py** file :

```
MIDDLEWARE = [ # M?t s? middleware khác c?
a django, 'maintenance_mode.middleware.MaintenanceModeMiddleware', ]
```

Step 2: Create an HTML template to show maintenance mode notifications

For the **django-maintenance-mode** package that shows a 503 error page, it looks for a **503.html template file in the templates** directory . To set it up, do the following:

1. Create a folder named templates in the root folder.

2. Open the newly created **templates** folder and create a file named **503.html** .

3. In the **settings file, select the TEMPLATES** setting and configure the **DIRS** list within it as follows:

```
'DIRS': [BASE_DIR/'templates'],
```

4. Open the file **503.html** and write the HMTL code to display an error message to the user. Here is sample code that you can use:

```
 503 Service Unavailable
```

503 Service Unavailable

Oops! We are currently working on some updates. We apologize for the inconvenience and appreciate your patience.

Please visit the website later or contact our support team

```
 Contact support
```

Step 3: Turn on maintenance mode and restart the server

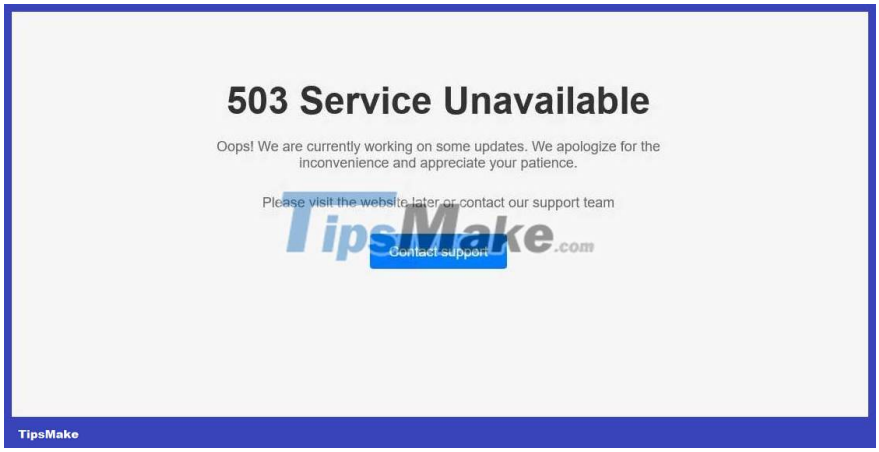
In the settings.py file, add this code to enable maintenance mode:

```
MAINTENANCE_MODE = True
```

Restart the development server by running the following command into the CLI:

```
python manage.py runserver
```

When navigating to the web, you will see the created maintenance page.



How to bypass admin page in Django maintenance mode

To allow the admin site to continue working even when in maintenance mode, **django-maintenance-mode** provides a setting named **MAINTENANCE_MODE_IGNORE_ADMIN_SITE** . You would add this setting to the **settings.py** file and set it to **True** :

```
MAINTENANCE_MODE_IGNORE_ADMIN_SITE = True
```

The default value of the above setting is **False** , so the admin page here will be affected by maintenance mode if you don't set it to **True** .

How to dismiss a specific function-based viewport in Django maintenance mode

The django-maintenence-mode package provides a decorator to prevent a window or certain page, such as About - from going into maintenance mode. To do this, first import the decorator into the **views.py** module .

```
from maintenance_mode.decorators import force_maintenance_mode_off
```

After importing decorator, add it to the viewer like so:

```
@force_maintenance_mode_off def view_name(request): # th?c hi?n logic c?a s
? xem # không bao gi? quay l?i ph?n h?i 503 response
```

After implementing the appropriate decorator, the URL of the particular view window will allow the user to access it.

How to ignore a class viewer in Django maintenance mode

Ignoring a class-based viewer is similar to omitting a function-based viewer. However, the best method here is to do it in the **urls.py** file .

First, you need to enter **decoratorforce_maintenance_mode_off** in your app's **urls.py file.** Then you need to include it in the URL path. For example:

```
from maintenance_mode.decorators import force_maintenance_mode_off from .views i
? quay l?i ph?n h?
i 503 path('', force_maintenance_mode_off(YourView.as_view()), name='my_view'),
```

Make sure you also enter the necessary data such as **the path** and the class-based viewer.

How to enable maintenance mode for a specific function-based viewer

1. To enable maintenance mode for a single view window, first disable maintenance mode in the **settings.py** file as follows:

```
MAINTENANCE_MODE = False
```

2. Next, in **views.py** , you should enter decorator **force_maintenance_mode_on** and add it to the view window:

```
from maintenance_mode.decorators import force_maintenance_mode_on @force_mainten
```

How to enable maintenance mode for a specific class-based viewer

First, you need to disable maintenance mode in the **settings.py** file :

```
MAINTENANCE_MODE = False
```

Next, in **urls.py** , you should import decorator and add it to the requested URL path:

```
from maintenance_mode.decorators import force_maintenance_mode_on from .views im
```

How to use different template names for Django maintenance mode

By default, the **django-maintenance-mode** package looks for **templates/503.html** . You can decide to override it in the **settings.py** file .

Assuming you have a separate folder for error handling in your app, you'll want to include the **503.html** template in this folder. Therefore, your template will be in **templates/errors/503.html** .

The default settings for this configuration are:

```
MAINTENANCE_MODE_TEMPLATE = "503.html"
```

To override it, you should add another path that points to the error page. For example:

```
MAINTENANCE_MODE_TEMPLATE = "errors/503.html"
```

You can also change the filename if you want, and everything will be fine if you add the necessary configurations.

Upgrading the maintenance mode in the app can make things easier for you and your users. By temporarily disabling access to all or part of your app during updates or maintenance, you can minimize interruptions and errors that arise from concurrent user interactions.