Монтаж

Процесс

Каждый из следующих шагов необходимо настроить, чтобы панель инструментов отладки была полностью функциональной.

> **Предупреждение**
>
> Панель инструментов отладки в настоящее время не поддерживает асинхронные представления Django .

1. Установите пакет

Рекомендуемый способ установки панели инструментов отладки — через pip :

```
$ python -m pip install django-debug-toolbar
```

Если вы не знакомы с pip, вы также можете получить копию каталога `debug_toolbar` и добавить ее в свой путь Python.

Чтобы протестировать предстоящий выпуск, вы можете вместо этого установить версию, находящуюся в разработке, с помощью следующей команды:

```
$ python -m pip install -e git+https://github.com/jazzband/django-debug-toolbar.git#egg=django-debug-toolbar
```

Если вы обновляетесь с предыдущей версии, вам следует просмотреть журнал изменений и найти конкретные инструкции по обновлению.

2. Проверьте предварительные условия

Панель инструментов отладки требует от ядра Django двух вещей. Они уже настроены в шаблоне Django по умолчанию `startproject` , поэтому в большинстве случаев они уже настроены.

Сначала убедитесь, что `'django.contrib.staticfiles'` это указано в ваших `INSTALLED_APPS` настройках и настроено правильно :

```
INSTALLED_APPS = [
    # ...
    "django.contrib.staticfiles",
    # ...
]

STATIC_URL = "static/"
```

Во-вторых, убедитесь, что ваши `TEMPLATES` настройки содержат `DjangoTemplates` серверную часть, для параметров которой `APP_DIRS` установлены значения `True`:

```
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "APP_DIRS": True,
        # ...
    }
]
```

3. Установите приложение

Добавьте `"debug_toolbar"` в свою `INSTALLED_APPS` настройку:

```
INSTALLED_APPS = [
    # ...
    "debug_toolbar",
    # ...
]
```

4. Добавьте URL-адреса

Добавьте URL-адреса панели инструментов django-debug-toolbar в URLconf вашего проекта:

```
from django.urls import include, path

urlpatterns = [
    # ...
    path("__debug__/", include("debug_toolbar.urls")),
]
```

В этом примере используется `__debug__` префикс, но вы можете использовать любой префикс, который не конфликтует с URL-адресами вашего приложения.

5. Добавьте промежуточное ПО

Панель инструментов отладки в основном реализована в промежуточном программном обеспечении. Добавьте его в свои `MIDDLEWARE` настройки:

```
MIDDLEWARE = [
    # ...
    "debug_toolbar.middleware.DebugToolbarMiddleware",
    # ...
]
```

Предупреждение

Порядок `MIDDLEWARE` важен. Вам следует включить промежуточное программное обеспечение панели инструментов отладки в список как можно раньше. Однако он должен идти после любого другого промежуточного программного обеспечения, которое кодирует содержимое ответа, например `GZipMiddleware` .

6. Настройте внутренние IP-адреса

Панель инструментов отладки отображается только в том случае, если ваш IP-адрес указан в `INTERNAL_IPS` настройках Django. Это означает, что для локальной разработки *необходимо* добавить `"127.0.0.1"` в `INTERNAL_IPS` . Вам необходимо создать этот параметр, если он еще не существует в вашем модуле настроек:

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

Вы можете изменить логику определения необходимости отображения панели инструментов отладки с помощью параметра SHOW_TOOLBAR_CALLBACK .

Предупреждение

`INTERNAL_IPS` Если вы используете Docker , в режиме отладки вы будете правильно настроены :

```
if DEBUG:
    import socket  # only if you haven't already imported this
    hostname, _, ips = socket.gethostbyname_ex(socket.gethostname())
    INTERNAL_IPS = [ip[: ip.rfind(".")] + ".1" for ip in ips] + ["127.0.0.1", "10.0.2.2"]
```

Поиск неисправностей

На некоторых платформах команда Django `runserver` может использовать неправильные типы контента для статических ресурсов. Чтобы угадать типы контента, Django использует модуль `mimetypes` стандартной библиотеки Python, который в свою очередь использует файлы карт базовой платформы. Если вы обнаружите неправильные типы контента для определенных файлов, скорее всего, файлы карт платформы неверны или их необходимо обновить. Этого можно добиться, например, установив или обновив `mailcap` пакет в дистрибутиве Red Hat, `mime-support` в дистрибутиве Debian или отредактировав ключи `HKEY_CLASSES_ROOT` в реестре Windows.

Запрос между источниками заблокирован

Панель инструментов отладки загружает модуль JavaScript . Типичная локальная разработка с использованием Django `runserver` не пострадает. Однако если ваш сервер приложений и сервер статических файлов находятся в разных источниках, вы можете увидеть ошибки CORS в консоли разработки вашего браузера:

```
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at http://localhost/static/debug_toolbar/js/toolbar.js. (Reason: CORS header ‘Access-Control-Allow-Origin’ missing).
```

Или

```
Access to script at 'http://localhost/static/debug_toolbar/js/toolbar.js' from origin 'http://localhost:8000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
```

Чтобы решить эту проблему, настройте сервер статических файлов, добавив заголовок Access-Control-Allow-Origin с источником сервера приложений. Например, если ваш сервер приложений находится по адресу `http://example.com`, а ваши статические файлы обслуживаются NGINX, добавьте:

```
add_header Access-Control-Allow-Origin http://example.com;
```

И для Апача:

```
Header add Access-Control-Allow-Origin http://example.com
```

Каналы Django и асинхронность

Панель инструментов отладки в настоящее время не поддерживает каналы Django или асинхронные проекты. Если при использовании каналов Django возникают проблемы с загрузкой панелей, просмотрите документацию по параметру конфигурации RENDER_PANELS .

HTML-код

Если вы используете HTMX для увеличения страницы, вам нужно будет добавить в свой код следующий обработчик событий:

```
{% if debug %}
    if (typeof window.htmx !== "undefined") {
        htmx.on("htmx:afterSettle", function(detail) {
            if (
                typeof window.djdt !== "undefined"
                && detail.target instanceof HTMLBodyElement
            ) {
                djdt.show_toolbar();
            }
        });
    }
{% endif %}
```

Для использования требуется , чтобы django.template.context_processors.debug был включен в параметр настройки TEMPLATES . Конфигурация Django по умолчанию включает этот контекстный процессор. `{% if debug %}` `'context_processors'`