

Django-Debug-Toolbar 的安装与使用

Django-Debug-Toolbar —— 一款调试 Django 项目的神器，在项目开发阶段，我们可以使用它进行辅助调试和优化。配置了它之后，我们就可以很方便的查看到如下表所示的项目运行信息，这些信息对调试项目和优化Web应用性能起到至关重要的作用。

项目	说明
History	查看请求接口的历史记录
Versions	Django的版本
Time	显示视图耗费的时间
Settings	配置文件中设置的值
Headers	HTTP请求头和响应头的信息
Request	和请求相关的各种变量及其信息
SQL	向数据库发送的SQL语句及其执行时间
Static files	静态文件加载情况
Templates	模板的相关信息
Cache	缓存的使用情况
Signals	Django内置的信号信息
Logging	被记录的日志信息

安装与配置

1. 安装

在使用 Django-Debug-Toolbar 之前，我们首先要安装它的第三方库 `django-debug-toolbar`，可以在 Vagrant 虚拟机中使用 `pip` 命令安装：

```
1 | pip install django-debug-toolbar
```

也可以在 PyCharm 中安装，具体的安装方法可以参考《开发环境搭建与项目初始化教程》中的“接入 Django Rest Framework”部分。

2. 配置

a. 配置 settings.py

• INSTALLED_APPS

在 twitter/settings.py 文件中的 INSTALLED_APPS 加入 debug_toolbar :

```
1 INSTALLED_APPS = [  
2     # .....  
3     'debug_toolbar',  
4 ]
```

修改如下图所示:

<pre>INSTALLED_APPS = ['django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', # third party 'rest_framework', # project apps 'accounts',]</pre>	<pre>33 34 35 36 37 38 39 40 41 42 43 44 45 46</pre>	<pre>33 34 35 36 37 38 39 40 41 42 43 44 45 46</pre>	<pre>INSTALLED_APPS = ['django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', # third party 'rest_framework', 'debug_toolbar', # project apps 'accounts',]</pre>
---	--	--	---

• MIDDLEWARE

在 MIDDLEWARE 加入 debug_toolbar.middleware.DebugToolbarMiddleware :

```
1 MIDDLEWARE = [  
2     # include the debug toolbar middleware as early as possible  
3     # but must come after any other middlewares that encodes the response  
4     # content  
5     # such as 'GZipMiddleware'  
6     'debug_toolbar.middleware.DebugToolbarMiddleware',  
7     # .....  
8 ]
```

修改如下图所示:

<pre>MIDDLEWARE = ['django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware',]</pre>	<pre>51 52 53 54 55 56 57 58 59 60</pre>	<pre>53 54 55 56 57 58 59 60 61 62</pre>	<pre>MIDDLEWARE = ['debug_toolbar.middleware.DebugToolbarMiddleware', 'django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware',]</pre>
---	--	--	--

• INTERNAL_IPS

新增 INTERNAL_IPS 并加入 IP 地址, 在此之前我们要先获取 IP 地址。首先我们要先修改

login_status 接口方法

<pre>def login_status(self, request): """ 查看用户当前的登录状态和具体信息 """ data = {'has_logged_in': request.user.is_authenticated} if request.user.is_authenticated: data['user'] = UserSerializer(request.user).data return Response(data)</pre>	<pre>93 94 95 96 97 98 99 100 101 102 103</pre>	<pre>def login_status(self, request): """ 查看用户当前的登录状态和具体信息 """ data = { 'has_logged_in': request.user.is_authenticated, 'ip': request.META['REMOTE_ADDR'] } if request.user.is_authenticated: data['user'] = UserSerializer(request.user).data return Response(data)</pre>
---	---	--

然后重启服务, 访问 localhost:/api/accounts/login_status/ 获取 IP 地址

Login status

查看用户当前的登录状态和具体信息

```
GET /api/accounts/login_status/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "has_logged_in": false,
  "ip": "10.0.2.2"
}
```

接着在 `settings.py` 文件中添加如下代码：

```
1 INTERNAL_IPS = ['10.0.2.2']
```

b. 配置 urls.py

在 `twitter/urls.py` 文件中，添加如下代码：

```
1 from django.conf import settings
2
3 # .....
4
5 if settings.DEBUG:
6     import debug_toolbar
7     urlpatterns.append(
8         path('__debug__', include(debug_toolbar.urls))
9     )
```

修改如下图所示：

<pre>from django.contrib import admin from django.urls import include, path from rest_framework import routers from accounts.api import views router = routers.DefaultRouter() router.register(r'api/users', views.UserViewSet) router.register(r'api/accounts', views.AccountViewSet, basename='accounts') urlpatterns = [path('admin/', admin.site.urls), path('', include(router.urls)), path('api-auth/', include('rest_framework.urls', namespace='rest_framework'))]</pre>	<pre>16 16 from django.contrib import admin 17 17 from django.urls import include, path 18 18 from rest_framework import routers 19 19 from accounts.api import views 20 20 from django.conf import settings 21 21 22 22 router = routers.DefaultRouter() 23 23 router.register(r'api/users', views.UserViewSet) 24 24 router.register(r'api/accounts', views.AccountViewSet, basename='accounts') 25 25 26 26 urlpatterns = [27 27 path('admin/', admin.site.urls), 28 28 path('', include(router.urls)), 29 29 path('api-auth/', include('rest_framework.urls', namespace='rest_framework')) 30 30] 31 31 32 32 33 33 if settings.DEBUG: 34 34 import debug_toolbar 35 35 urlpatterns.append(36 36 path('__debug__', include(debug_toolbar.urls)) 37 37)</pre>
--	--

安装和配置完毕后，让我们在再浏览器中输入 `localhost`，可以看到如下图所示界面：

Django REST framework

Api Root

Api Root

The default basic root view for DefaultRouter

GET /

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "api/users": "http://localhost/api/users/"
}
```

Hide » Log in

History ☒

/

Versions ☒

Django 3.1.3

Time ☒

CPU: 195.78ms (195.78ms)

Settings ☒

Headers ☒

Request ☒

APIRootView

SQL ☒

0 queries in 0.00ms

Static files ☒

10 files used

Templates ☒

rest_framework/api.html