

Debug 工具安装与 Bug Fix 练习

授课人 令狐东邪

版权声明

九章的所有课程均受法律保护，不允许录像与传播录像
一经发现，将被追究法律责任和赔偿经济损失

本章需要做什么

安装与配置
Django-Debug-Toolbar

A

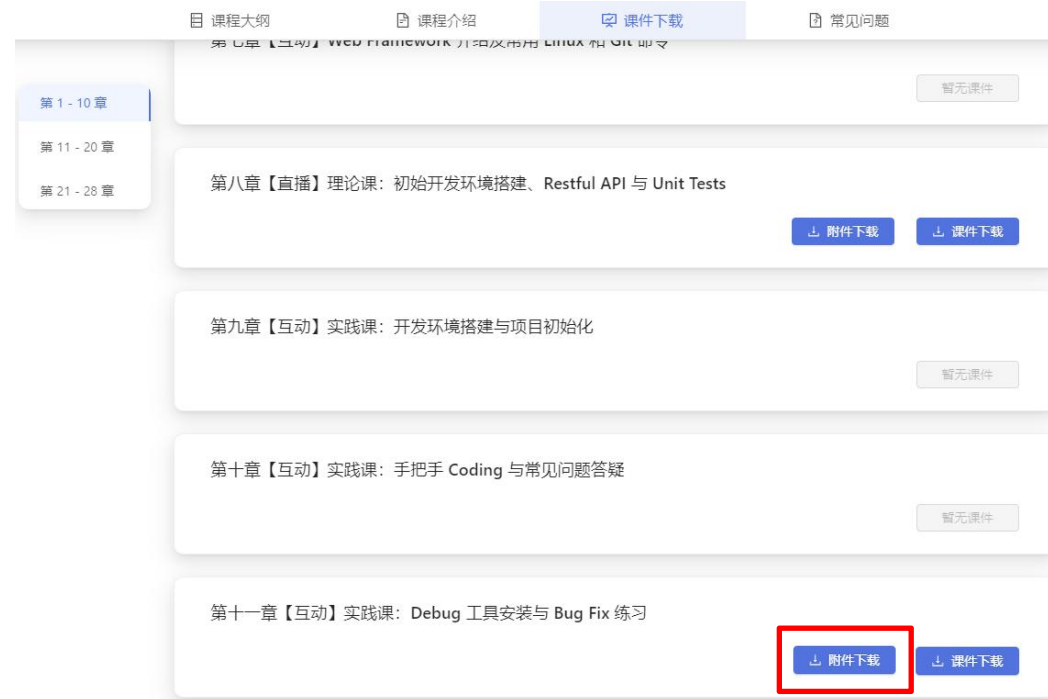
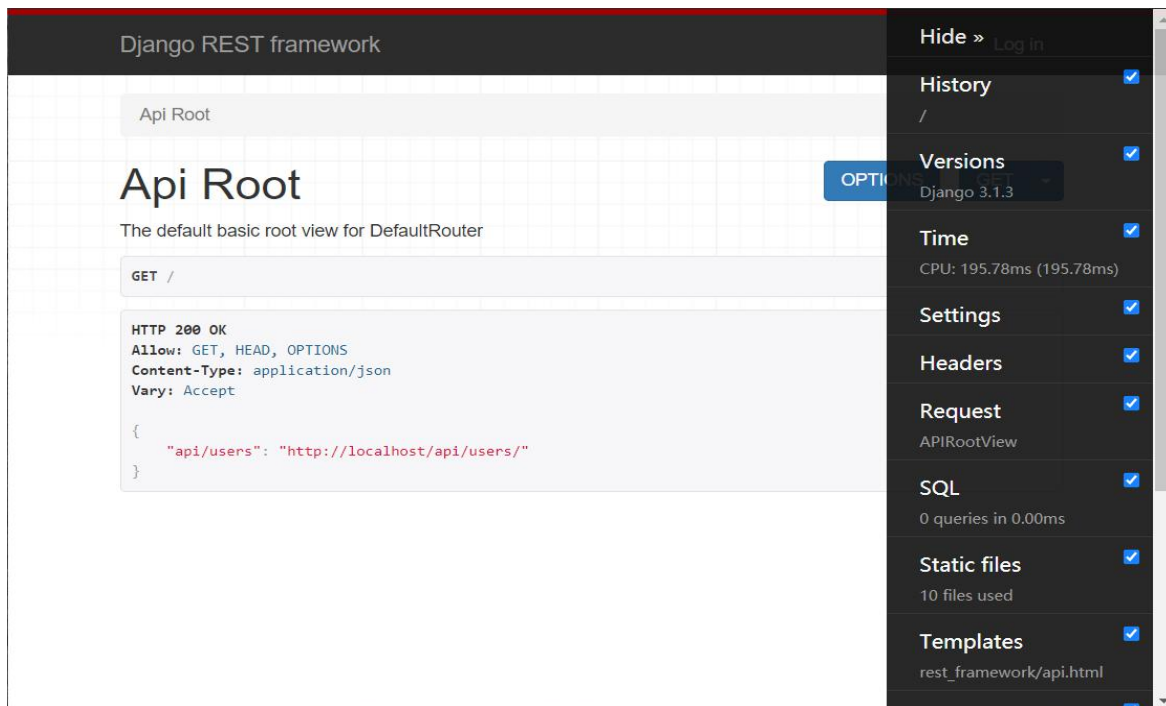
B

Bug Fix 练习

Django-Debug-Toolbar

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项目的运行信息，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。

具体的安装配置见教程



功能简介

选项	功能
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

学会看报错信息

有的 bug 通过看报错信息，很快就能解决，报错可以在很多地方看到，比如当你在浏览器中输入 URL，返回如下界面，就说明你的接口报错了。

AssertionError at /api/accounts/login/

The field 'emial' was declared on serializer SignupSerializer, but has not been included in the 'fields' option.

```
Request Method: GET
Request URL: http://localhost/api/accounts/login/
Django Version: 3.1.3
Exception Type: AssertionError
Exception Value: The field 'emial' was declared on serializer SignupSerializer, but has not been included in the 'fields' option.
Exception Location: /usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py, line 1130, in get_field_names
Python Executable: /usr/bin/python
Python Version: 3.6.9
Python Path: ['/vagrant',
              '/home/vagrant/.pycharm_helpers/pydev',
              '/vagrant',
              '/home/vagrant/.pycharm_helpers/pycharm_display',
              '/home/vagrant/.pycharm_helpers/third_party/thriftpy',
              '/home/vagrant/.pycharm_helpers/pydev',
              '/vagrant/C',
              '/Users/51/AppData/Local/JetBrains/PyCharm2020.1/cythonExtensions',
              '/usr/lib/python36.zip',
              '/usr/lib/python3.6',
              '/usr/lib/python3.6/lib-dynload',
              '/home/vagrant/.local/lib/python3.6/site-packages',
              '/usr/local/lib/python3.6/dist-packages',
              '/usr/lib/python3/dist-packages',
              '/home/vagrant/.pycharm_helpers/pycharm_matplotlib_backend']
Server time: Mon, 19 Apr 2021 01:06:21 +0000
```

Traceback [Switch to copy-and-paste view](#)

/usr/local/lib/python3.6/dist-packages/django/core/handlers/exception.py, line 47, in inner

```
47.         response = get_response(request)
```

► Local vars

表示这个 bug 的错误类型是 **AssertionError**，访问的 URL 是 `/api/accounts/login/`。

这行是具体的报错信息，像这个报错已经十分明显地告诉你问题出在哪了，在序列化器 `SignupSerializer` 中定义了字段 `'emial'`，但是没有包含在 `'fields'` 选项中。

细心的同学可能已近发现了，是 `'emial'` 这个单词拼写错了，不过还是让我们根据具体的代码来确定吧。

AssertionError at /api/accounts/login/

The field 'emial' was declared on serializer SignupSerializer, but has not been included in the 'fields' option.

```
Request Method: GET
Request URL: http://localhost/api/accounts/login/
Django Version: 3.1.3
Exception Type: AssertionError
Exception Value: The field 'emial' was declared on serializer SignupSerializer, but has not been included in the 'fields' option.
Exception Location: /usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py, line 1130, in get_field_names
Python Executable: /usr/bin/python
Python Version: 3.6.9
Python Path: ['/vagrant',
              '/home/vagrant/.pycharm_helpers/pydev',
              '/vagrant',
              '/home/vagrant/.pycharm_helpers/pycharm_display',
              '/home/vagrant/.pycharm_helpers/third_party/thriftpy',
              '/home/vagrant/.pycharm_helpers/pydev',
              '/vagrant/C',
              '/Users/51/AppData/Local/JetBrains/PyCharm2020.1/cythonExtensions',
              '/usr/lib/python36.zip',
              '/usr/lib/python3.6',
              '/usr/lib/python3.6/lib-dynload',
              '/home/vagrant/.local/lib/python3.6/site-packages',
              '/usr/local/lib/python3.6/dist-packages',
              '/usr/lib/python3/dist-packages',
              '/home/vagrant/.pycharm_helpers/pycharm_matplotlib_backend']
Server time: Mon, 19 Apr 2021 01:06:21 +0000
```

Traceback [Switch to copy-and-paste view](#)

/usr/local/lib/python3.6/dist-packages/django/core/handlers/exception.py, line 47, in inner

```
47.         response = get_response(request)
```

► Local vars

果然是我们在定义变量的时候将变量名写错了，只要改正变量名，重启服务即可正常访问。

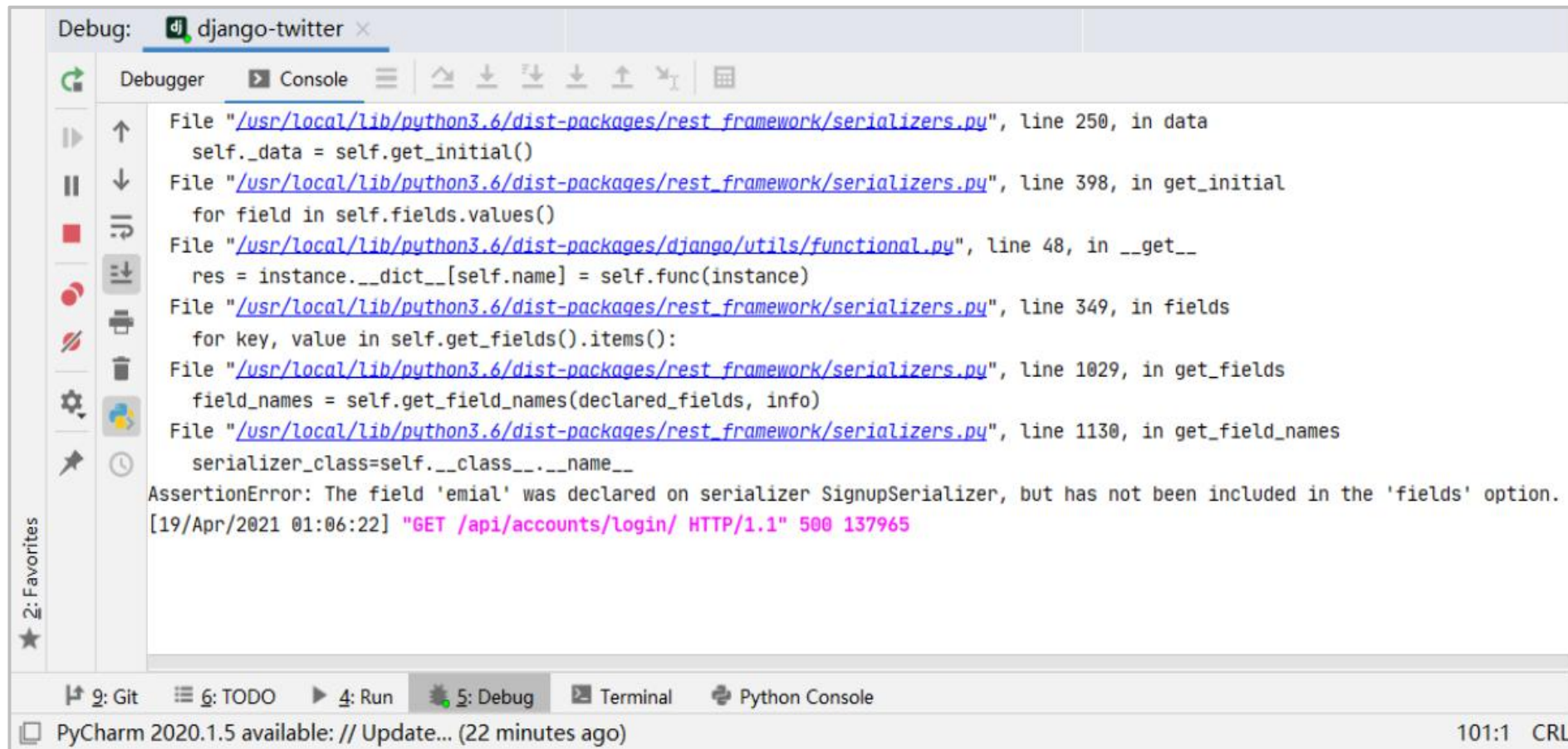
serializers.py

```
class SignupSerializer(serializers.ModelSerializer):
    username = serializers.CharField(max_length=20, min_length=6)
    password = serializers.CharField(max_length=20, min_length=6)
    emial = serializers.EmailField()

    class Meta:
        model = User
        fields = ('username', 'email', 'password')
```

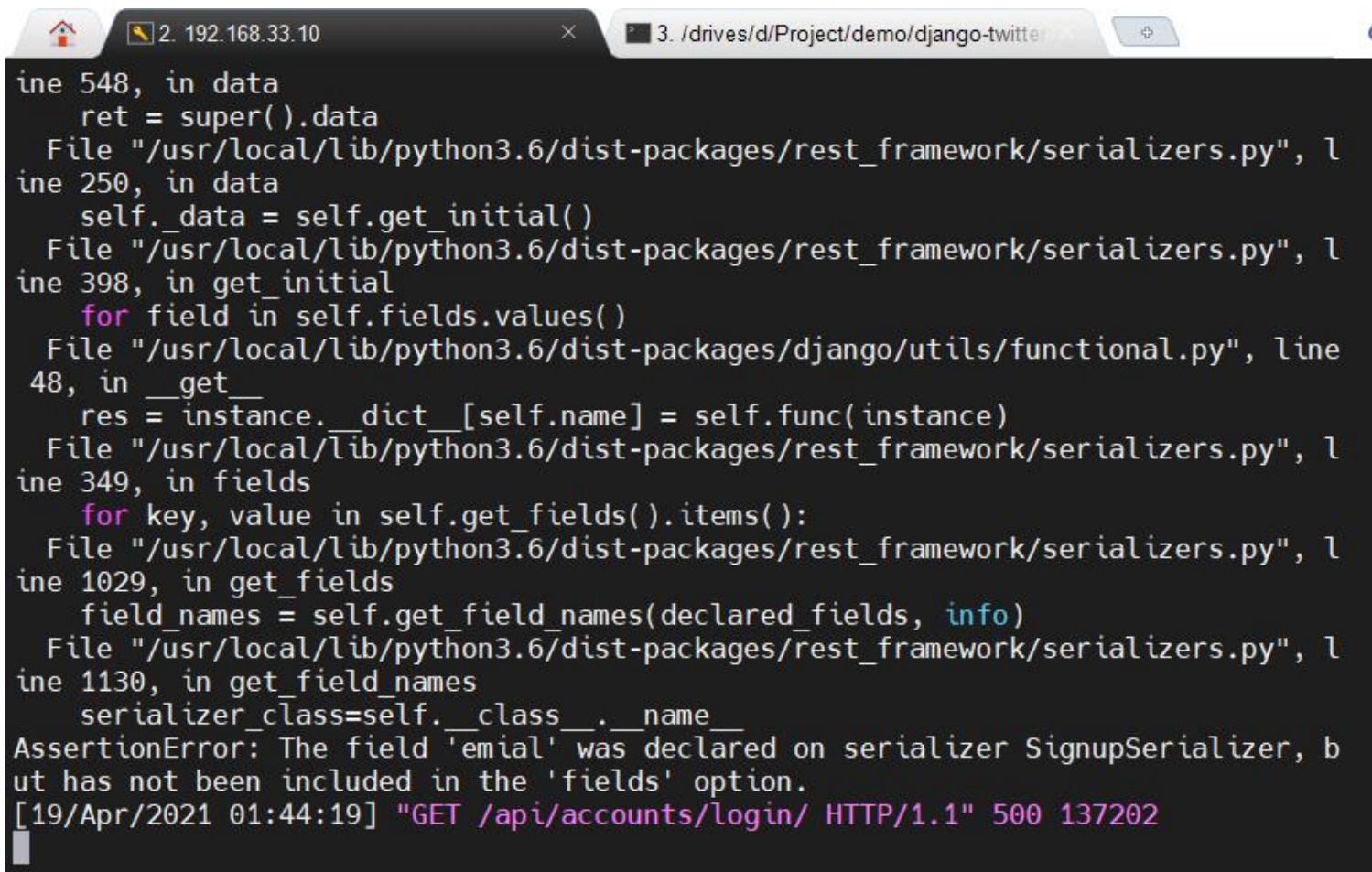

在 PyCharm 看报错信息

前面展示的报错信息在网页中，如果在 **PyCharm** 中运行服务，那么我也可以在 PyCharm 中看报错信息。



在虚拟机看报错信息

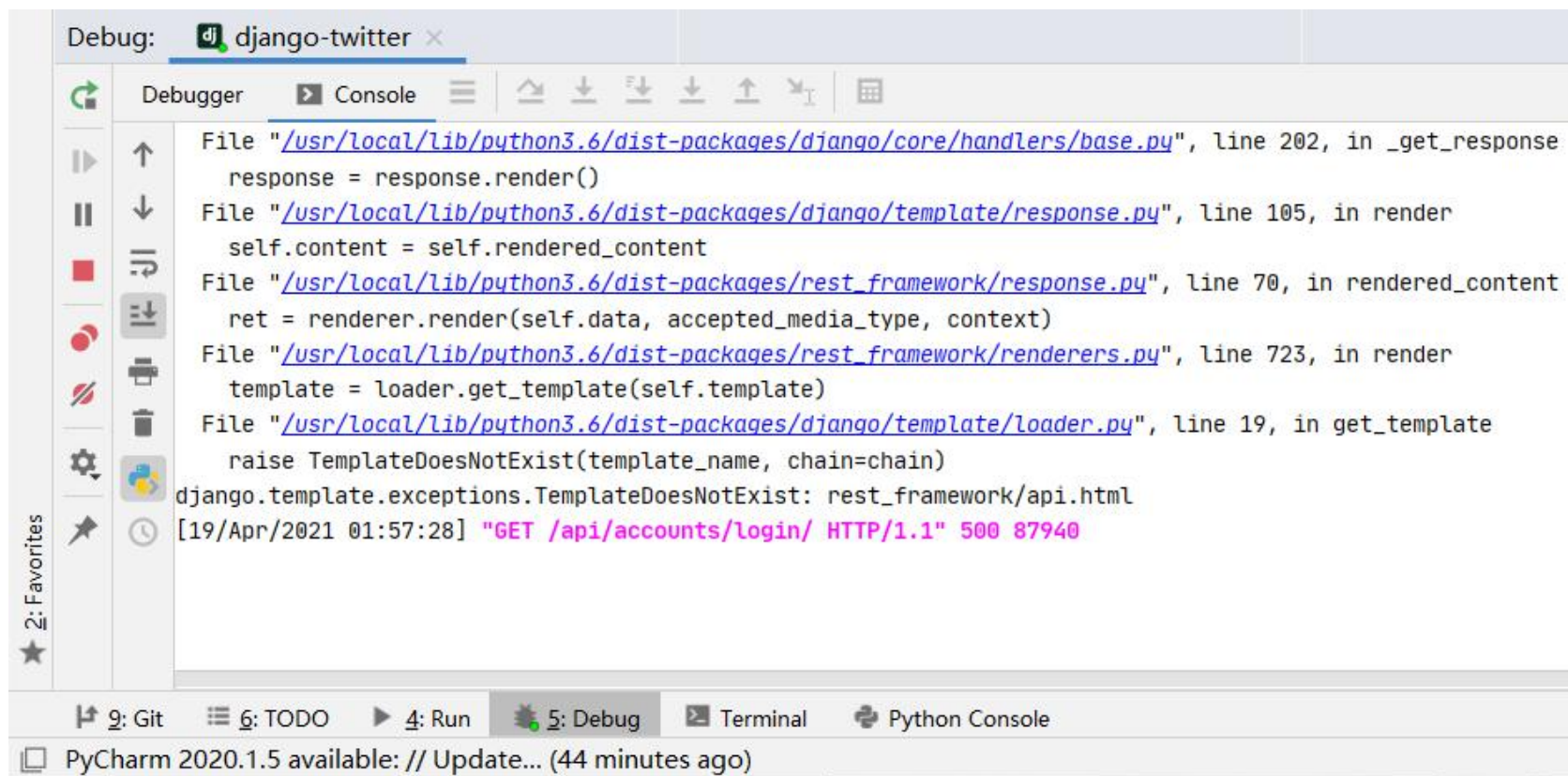
如果在**虚拟机**中通过
python manage.py
runserver 来运行，那么
可以在虚拟机中看到报错
信息。



```
ine 548, in data
    ret = super().data
  File "/usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py", l
ine 250, in data
    self._data = self.get_initial()
  File "/usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py", l
ine 398, in get_initial
    for field in self.fields.values()
  File "/usr/local/lib/python3.6/dist-packages/django/utils/functional.py", line
48, in __get__
    res = instance.__dict__[self.name] = self.func(instance)
  File "/usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py", l
ine 349, in fields
    for key, value in self.get_fields().items():
  File "/usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py", l
ine 1029, in get_fields
    field_names = self.get_field_names(declared_fields, info)
  File "/usr/local/lib/python3.6/dist-packages/rest_framework/serializers.py", l
ine 1130, in get_field_names
    serializer_class=self.__class__.__name__
AssertionError: The field 'email' was declared on serializer SignupSerializer, b
ut has not been included in the 'fields' option.
[19/Apr/2021 01:44:19] "GET /api/accounts/login/ HTTP/1.1" 500 137202
```

根据报错信息修改

有的时候，我们根据报错信息也很难判断问题出在哪里，比如下面这个报错。



对于初学者而言，我们很难根据上面提供的报错信息自己去解决 bug，这个时候我们可以将报错信息复制一下，然后去 **google** 搜索。很多时候我们现在遇到的 bug 都是之前别人踩过的坑，善用 google 能帮你解决很多 bug。



django.template.exceptions.TemplateDoesNotExist: rest_framework/api.htr X



[全部](#) [新闻](#) [购物](#) [地图](#) [视频](#) [更多](#)

[设置](#) [工具](#)

找到约 76,300 条结果 (用时 0.75 秒)

<https://stackoverflow.com/questions/djang...> [翻译此页](#)

Django: TemplateDoesNotExist (rest_framework/api.html ...

2019年4月9日 — I believe this happens because Django REST Framework wants to render a template (rest_framework/api.html), but the template can't be ...

10 个回答 · 218 票: Make sure you have rest_framework in your settings's INSTALLED_APPS

[TemplateDoesNotExist - Django Error - Stack Overflow](#) 2015年10月2日

[Templates does no exist for rest_framework/API.html - Stack ...](#) 2017年3月24日

[TemplateDoesNotExist at /app/api/get - Stack Overflow](#) 2017年9月2日

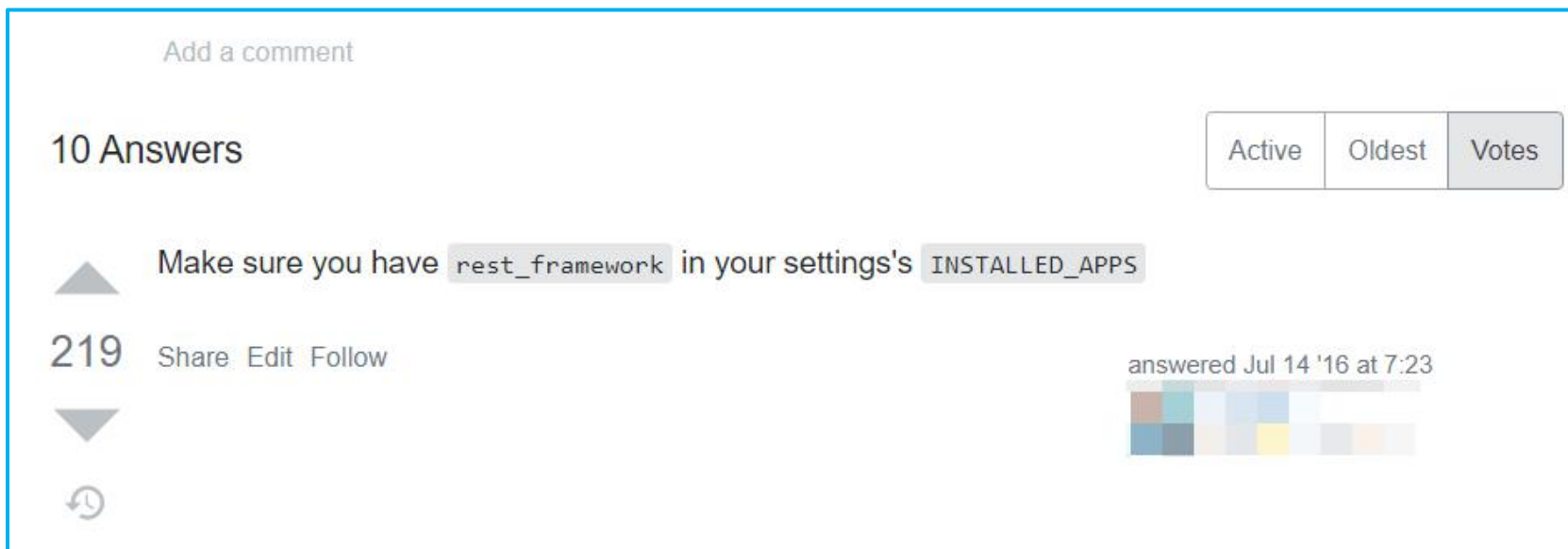
[TemplateDoesNotExist in Heroku using Django Rest Framework](#) 2016年4月22日

[stackoverflow.com](#)站内的其它相关信息

<https://blog.csdn.net/article/details>

Django TemplateDoesNotExist: rest_framework_Rao的博客 ...

2018年9月5日 — 使用restframework 报错. django.template.exceptions.TemplateDoesNotExist: rest_framework/api.html. class ...



出现这个 bug 的原因是在我们安装 `rest_framework` 后，没有在 `INSTALLED_APPS` 添加 `rest_framework`，导致 Django 无法读取 `rest_framework` 自带的 `api.html` 模板。

断点调试

有的时候我们会遇到这样的情况，代码没有报错，我们的操作也没有问题，但是就是操作失败了，这通常是我们的代码逻辑出现了问题。比如我们在进行登录操作是，明明用户名和密码都是正确的，但就是登录不上。我们的用户名和密码如下：

用户名: zhangsan

密码: 123456

Login

Extra Actions ▾

OPTIONS

默认的 username 是 admin, password 也是 admin

POST /api/accounts/login/

HTTP 400 Bad Request

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "success": false,
  "message": "username and password does not match"
}
```

Raw data

HTML form

Username

zhangsan

Email

Password

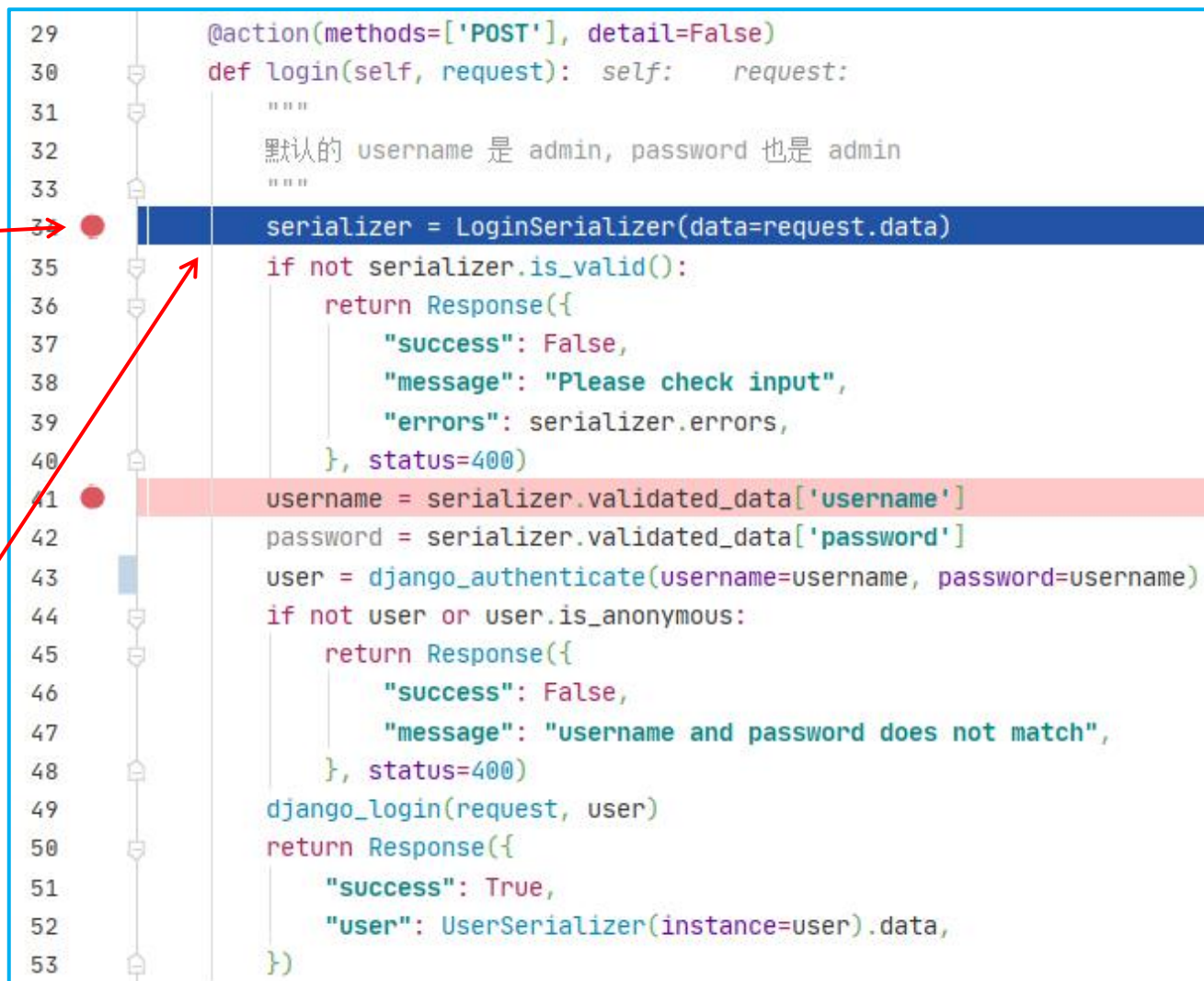
123456

POST

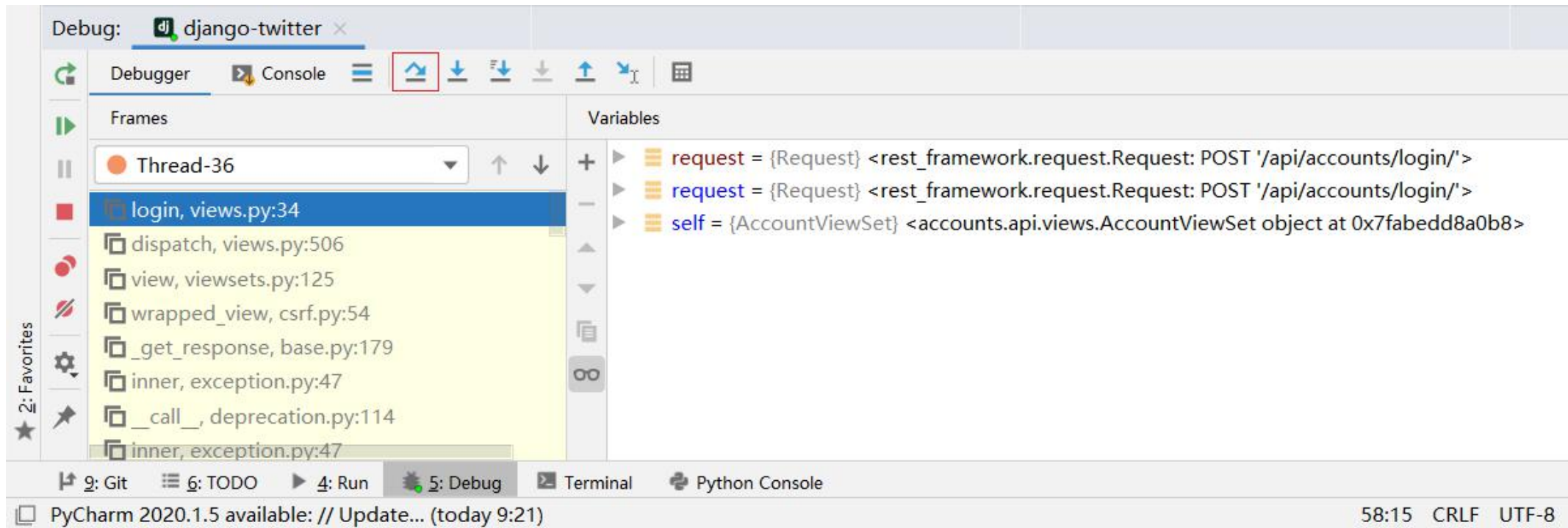
这时候我们可以采用**打断点**的方式进行调试。

如上图所示，我们在第34行代码的位置通过鼠标左键单击，就会在单击的位置出现一个红色的点。

当我们访问登录功能的接口时，Python 解释器会去执行上图所示的代码，当执行到第34行时，就会停止，等待我们进一步的操作。当执行停止在某一行的时候，该行代码用蓝色背景进行表示。



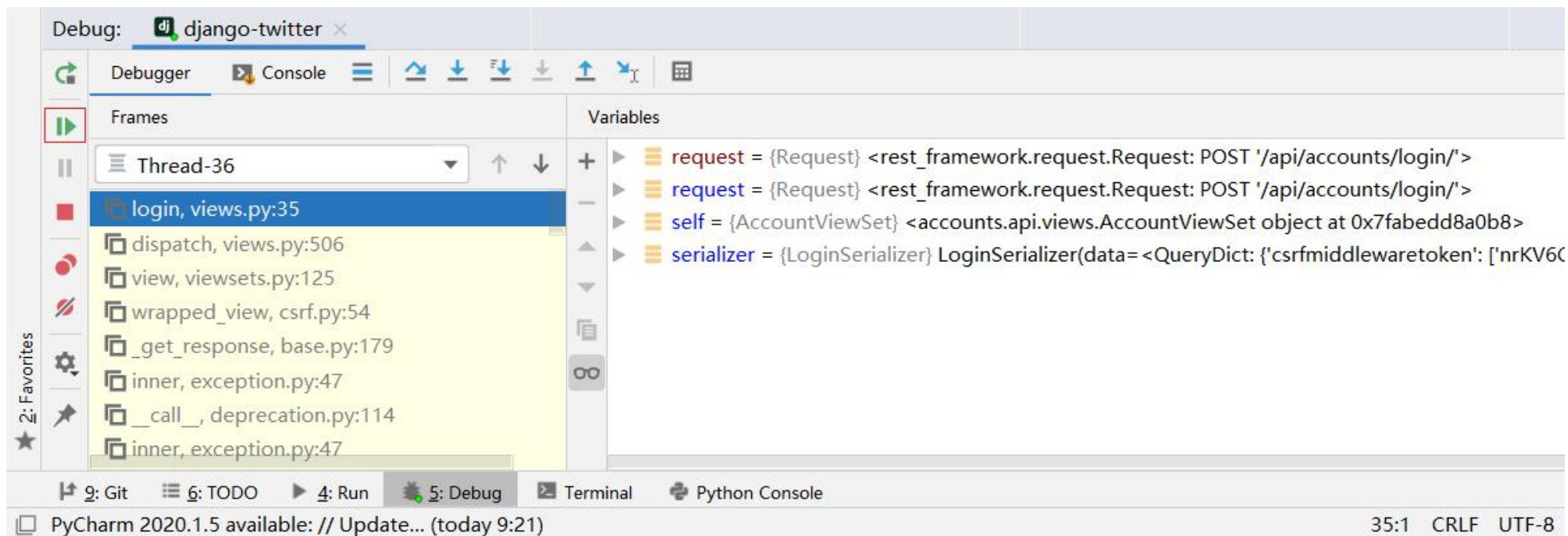
如果我们想让代码接着往下执行，可以点击下图中红框框中的按钮。



点击按钮后，我们的代码向下执行了一行。

```
29 @action(methods=['POST'], detail=False)
30 def login(self, request): self: <accounts.api.views.AccountViewSet ob
31     """
32     默认的 username 是 admin, password 也是 admin
33     """
34     serializer = LoginSerializer(data=request.data) serializer:
35     if not serializer.is_valid():
36         return Response({
37             "success": False,
38             "message": "Please check input",
39             "errors": serializer.errors,
40         }, status=400)
41     username = serializer.validated_data['username']
42     password = serializer.validated_data['password']
43     user = django_authenticate(username=username, password=password)
44     if not user or user.is_anonymous:
45         return Response({
46             "success": False,
47             "message": "username and password does not match",
48         }, status=400)
49     django_login(request, user)
50     return Response({
51         "success": True,
52         "user": UserSerializer(instance=user).data,
53     })
```

当然如果你不想一行一行的执行，也可以直接跳到下一个断点的位置，可以点击下图红框框中的按钮。



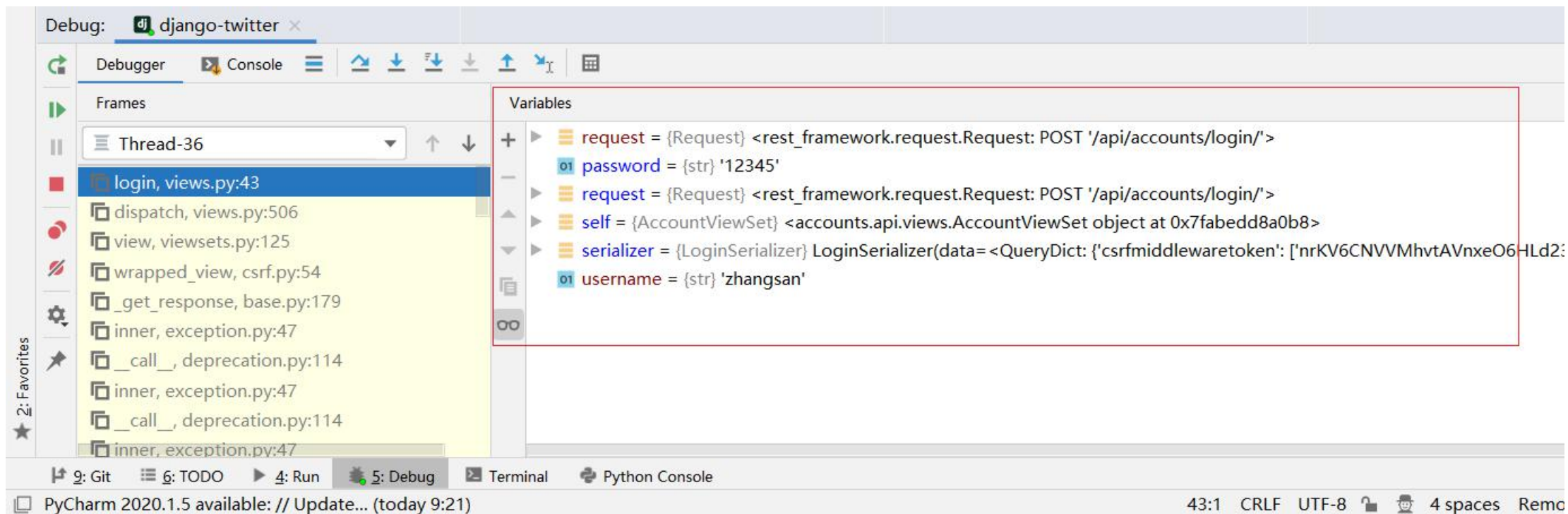
点击按钮后，我们的代码直接跳到了下一个断点的位置。

```
29     @action(methods=['POST'], detail=False)
30     def login(self, request): self: <accounts.api.views.AccountViewSet ob
31         """
32         默认的 username 是 admin, password 也是 admin
33         """
34         serializer = LoginSerializer(data=request.data)  serializer: Login
35         if not serializer.is_valid():
36             return Response({
37                 "success": False,
38                 "message": "Please check input",
39                 "errors": serializer.errors,
40             }, status=400)
41         username = serializer.validated_data['username']
42         password = serializer.validated_data['password']
43         user = django_authenticate(username=username, password=password)
44         if not user or user.is_anonymous:
45             return Response({
46                 "success": False,
47                 "message": "username and password does not match",
48             }, status=400)
49         django_login(request, user)
50         return Response({
51             "success": True,
52             "user": UserSerializer(instance=user).data,
53         })
```

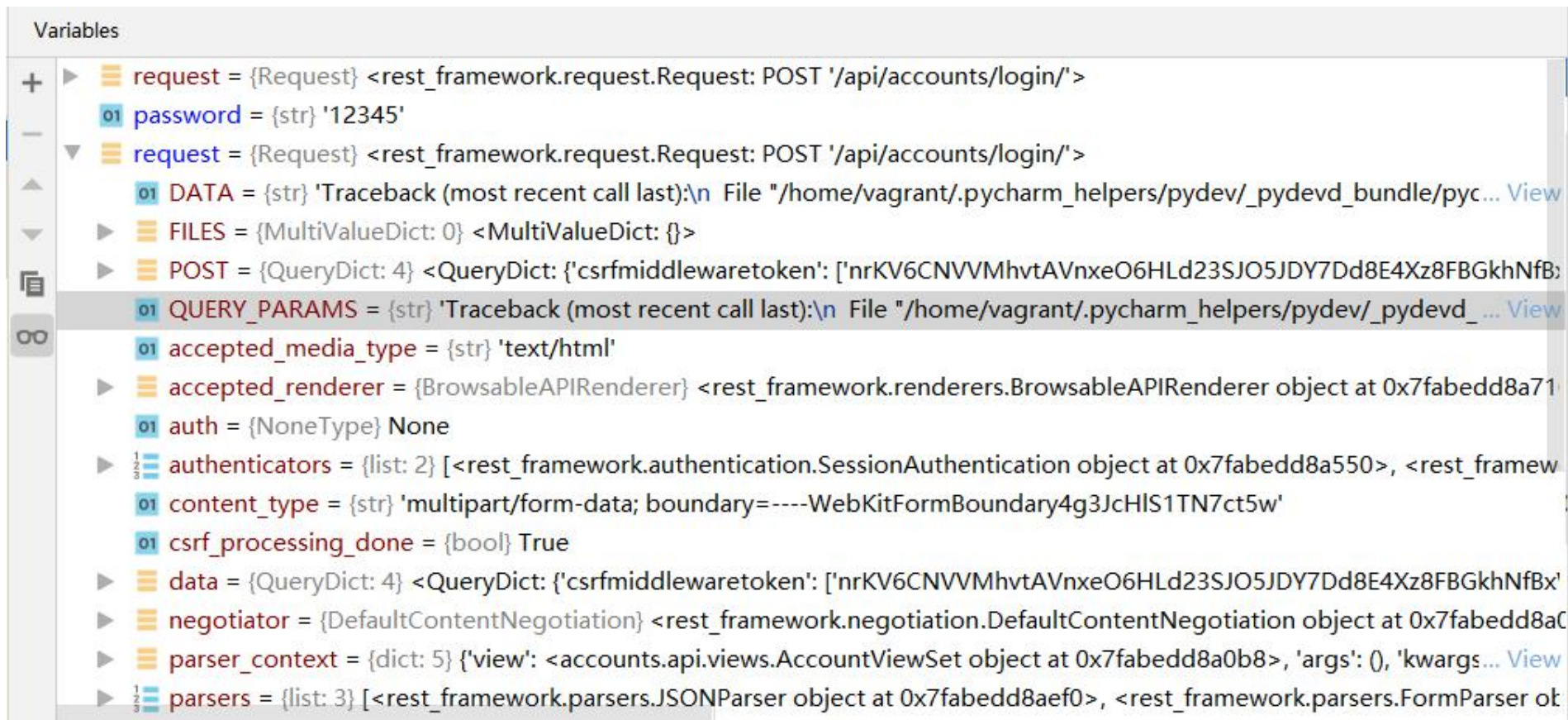

断点调试除了可以逐行执行代码外，你还可以看代码执行过程中变量的值。

```
29 @action(methods=['POST'], detail=False)
30 def login(self, request):
31     """
32     默认的 username 是 admin, password 也是 admin
33     """
34     serializer = LoginSerializer(data=request.data)
35     if not serializer.is_valid():
36         return Response({
37             "success": False,
38             "message": "Please check input",
39             "errors": serializer.errors,
40         }, status=400)
41     username = serializer.validated_data['username']
42     password = serializer.validated_data['password']
43     user = django_authenticate(username=username, password=password)
44     if not user or user.is_anonymous:
45         return Response({
46             "success": False,
47             "message": "username and password does not match",
48         }, status=400)
49     django_login(request, user)
50     return Response({
51         "success": True,
52         "user": UserSerializer(instance=user).data,
53     })
```

我们也可以从 Variables 窗口查看代码执行过程中产生的所有对象。如下图所示，我们可以看到 `username` 和 `password`。



还可以看到一些其他比较复杂的对象，并将其展开，查看它的各种属性。如下图所示我们可以看到 `request` 的详细信息。



```
Variables
+ request = {Request} <rest_framework.request.Request: POST '/api/accounts/login/'>
  01 password = {str} '12345'
  request = {Request} <rest_framework.request.Request: POST '/api/accounts/login/'>
    01 DATA = {str} 'Traceback (most recent call last):\n File "/home/vagrant/.pycharm_helpers/pydev/_pydevd_bundle/pyc... View
    FILES = {MultiValueDict: 0} <MultiValueDict: {}>
    POST = {QueryDict: 4} <QueryDict: {'csrfmiddlewaretoken': ['nrKV6CNVVMhvtAVnxeO6HLd23SJO5JDY7Dd8E4Xz8FBGkhNfB:
    01 QUERY_PARAMS = {str} 'Traceback (most recent call last):\n File "/home/vagrant/.pycharm_helpers/pydev/_pydevd_ ... View
    01 accepted_media_type = {str} 'text/html'
    accepted_renderer = {BrowsableAPIRenderer} <rest_framework.renderers.BrowsableAPIRenderer object at 0x7fabedd8a71
    01 auth = {NoneType} None
    1 2 3 authenticators = {list: 2} [<rest_framework.authentication.SessionAuthentication object at 0x7fabedd8a550>, <rest_framework
    01 content_type = {str} 'multipart/form-data; boundary=----WebKitFormBoundary4g3JcHIS1TN7ct5w'
    01 csrf_processing_done = {bool} True
    data = {QueryDict: 4} <QueryDict: {'csrfmiddlewaretoken': ['nrKV6CNVVMhvtAVnxeO6HLd23SJO5JDY7Dd8E4Xz8FBGkhNfBx'
    negotiator = {DefaultContentNegotiation} <rest_framework.negotiation.DefaultContentNegotiation object at 0x7fabedd8aC
    parser_context = {dict: 5} {'view': <accounts.api.views.AccountViewSet object at 0x7fabedd8a0b8>, 'args': (), 'kwargs... View
    1 2 3 parsers = {list: 3} [<rest_framework.parsers.JSONParser object at 0x7fabedd8aef0>, <rest_framework.parsers.FormParser ok
```


让我们接着往下走，我们发现变量 `user` 的值为 `None`，导致条件判断为真，返回了错误的结果。也就是说当我们的用户名和密码都正确的情况下，`user` 的值肯定不应该是 `None`，因此我们可以判断第43行代码是错误。

```
29 @action(methods=['POST'], detail=False)
30 def login(self, request): self: <accounts.api.views.AccountViewSet object at 0x7f...
31 """
32 默认的 username 是 admin, password 也是 admin
33 """
34 serializer = LoginSerializer(data=request.data)  serializer: LoginSerializer(d...
35 if not serializer.is_valid():
36     return Response({
37         "success": False,
38         "message": "Please check input",
39         "errors": serializer.errors,
40     }, status=400)
41 username = serializer.validated_data['username']  username: 'zhangsan'
42 password = serializer.validated_data['password']  password: '12345'
43 user = django_authenticate(username=username, password=username)  user: None
44 if not user or user.is_anonymous:
45     return Response({
46         "success": False,
47         "message": "username and password does not match",
48     }, status=400)
49 django_login(request, user)
50 return Response({
51     "success": True,
52     "user": UserSerializer(instance=user).data,
53 })
```

仔细看看这行代码，可以发现就可以发现问题所在了，原来是 password 参数接收了 username 变量，导致我们无论输什么密码，密码的值都是用户名的值。

Print

很多时候，我们的代码并没有在 PyCharm 这类 IDE 上运行的，因此也无法打断点进行调试。

因此我们也可以通过 **print** 变量的方式进行调试。如我们可以将 `username` 和 `password` 这两个变量打印出来看看，确保它们的值没有出错，然后将 `user` 也打印出来看看。

```
29 @action(methods=['POST'], detail=False)
30 def login(self, request):
31     """
32     默认的 username 是 admin, password 也是 admin
33     """
34     serializer = LoginSerializer(data=request.data)
35     if not serializer.is_valid():
36         return Response({
37             "success": False,
38             "message": "Please check input",
39             "errors": serializer.errors,
40         }, status=400)
41     username = serializer.validated_data['username']
42     print("username: {}".format(username))
43     password = serializer.validated_data['password']
44     print("password: {}".format(password))
45     user = django_authenticate(username=username, password=password)
46     print("user: {}".format(user))
47     if not user or user.is_anonymous:
48         return Response({
49             "success": False,
50             "message": "username and password does not match",
51         }, status=400)
52     django_login(request, user)
53     return Response({
54         "success": True,
55         "user": UserSerializer(instance=user).data,
56     })
```

在虚拟机中可以看到打印出来的结果：

```
username: zhangsan  
password: 123456  
user: None  
Bad Request: /api/accounts/login/  
[19/Apr/2021 05:34:05] "POST /api/accounts/login/ HTTP/1.1" 400 19371
```

通过打印出来的值，我们就可以对问题代码进行判断了，具体判断的过程和前面打断点调试的过程相似。