as\_view方法代码如下：

@classmethod

def as\_view(cls, \*\*initkwargs):

"""

Store the original class on the view function.

This allows us to discover information about the view when we do URL

reverse lookups. Used for breadcrumb generation.

"""

if isinstance(getattr(cls, 'queryset', None), models.query.QuerySet):

def force\_evaluation():

raise RuntimeError(

'Do not evaluate the `.queryset` attribute directly, '

'as the result will be cached and reused between requests. '

'Use `.all()` or call `.get\_queryset()` instead.'

)

cls.queryset.\_fetch\_all = force\_evaluation

view = super(APIView, cls).as\_view(\*\*initkwargs)

view.cls = cls

view.initkwargs = initkwargs

# Note: session based authentication is explicitly CSRF validated,

# all other authentication is CSRF exempt.

return csrf\_exempt(view)

原来APIView类是继承View类，view类正式from django.views import View下的View,

先看as\_view方法中的view = super(APIView, cls).as\_view(\*\*initkwargs)的这行代码,

是调用了父类View中的as\_view方法，这里的initkwargs，及其父类的View中的as\_view方法执行流程，之类就不在赘述了，简单说就是在如下IndexView类的执行流程就是

先去执行print("dispatch")–>然后在去执行print("get")方法–>然后在去执行super(IndexView,self).dispatch(request, \*args, \*\*kwargs)–>最后执行return HttpResponse(ret)

