[From flask/app.py]

import os  
import sys  
from threading import Lock  
from datetime import timedelta  
from itertools import chain  
from functools import update\_wrapper  
  
from werkzeug.datastructures import ImmutableDict  
from werkzeug.routing import Map, Rule, RequestRedirect, BuildError  
from werkzeug.exceptions import HTTPException, InternalServerError, \  
 MethodNotAllowed, BadRequest, default\_exceptions  
  
from .helpers import \_PackageBoundObject, url\_for, get\_flashed\_messages, \  
 locked\_cached\_property, \_endpoint\_from\_view\_func, find\_package, \  
 get\_debug\_flag  
from . import json, cli  
from .wrappers import Request, Response  
from .config import ConfigAttribute, Config  
from .ctx import RequestContext, AppContext, \_AppCtxGlobals  
from .globals import \_request\_ctx\_stack, request, session, g  
from .sessions import SecureCookieSessionInterface  
from .templating import DispatchingJinjaLoader, Environment, \  
 \_default\_template\_ctx\_processor  
from .signals import request\_started, request\_finished, got\_request\_exception, \  
 request\_tearing\_down, appcontext\_tearing\_down  
from .\_compat import reraise, string\_types, text\_type, integer\_types

from werkzeug.routing import Map, Rule, RequestRedirect, BuildError

from werkzeug.exceptions import HTTPException, InternalServerError, \  
 MethodNotAllowed, BadRequest, default\_exceptions  
  
from .wrappers import Request, Response

from .ctx import RequestContext, AppContext, \_AppCtxGlobals

from .globals import \_request\_ctx\_stack, request, session, g

from .signals import request\_started, request\_finished, got\_request\_exception, \  
 request\_tearing\_down, appcontext\_tearing\_down

[From flask/wrappers.py]

from werkzeug.wrappers import Request as RequestBase, Response as ResponseBase  
from werkzeug.exceptions import BadRequest  
  
from . import json  
from .globals import \_request\_ctx\_stack  
  
\_missing = object()  
  
  
def \_get\_data(req, cache):  
 getter = getattr(req, **'get\_data'**, None)  
 if getter is not None:  
 return getter(cache=cache)  
 return req.data  
  
  
class Request(RequestBase):

[From werkzeug/wrappers.py]

class Request(BaseRequest, AcceptMixin, ETagRequestMixin,  
 UserAgentMixin, AuthorizationMixin,  
 CommonRequestDescriptorsMixin):  
  
 *"""Full featured request object implementing the following mixins:  
  
 - :class:`AcceptMixin` for accept header parsing  
 - :class:`ETagRequestMixin` for etag and cache control handling  
 - :class:`UserAgentMixin` for user agent introspection  
 - :class:`AuthorizationMixin` for http auth handling  
 - :class:`CommonRequestDescriptorsMixin` for common headers  
 """*