```python
"""
Base class for Scrapy spiders

See documentation in docs/topics/spiders.rst
"""
import logging
import warnings

from scrapy import signals
from scrapy.http import Request
from scrapy.utils.trackref import object_ref
from scrapy.utils.url import url_is_from_spider
from scrapy.utils.deprecate import create_deprecated_class
from scrapy.exceptions import ScrapyDeprecationWarning
from scrapy.utils.deprecate import method_is_overridden


class Spider(object_ref):
    """Base class for scrapy spiders. All spiders must inherit from this
    class.
    """

    name = None         # 类变量 用来标识爬虫程序
    custom_settings = None         # 爬虫的一些配置

    def __init__(self, name=None, **kwargs):        # __init__ 函数 接收一个参数name 以及多个键值对
        if name is not None:        # 如果name不为None
            self.name = name        # 则赋值给类变量name
        elif not getattr(self, 'name', None):        # 获取类变量name 如果不为真 比如说'', [], False
            raise ValueError("%s must have a name" % type(self).__name__)    # 然后抛出异常，提示说该类必须有一个名字
        self.__dict__.update(kwargs)        # 更新类变量，将键值对都更新到类变量中
        if not hasattr(self, 'start_urls'):        # 如果不存在'start_urls'这个类变量
            self.start_urls = []        # 初始化self.start_urls = []

    @property        # property python内置的装饰器 将函数封装成属性调用
    def logger(self):        # 定义函数logger
        logger = logging.getLogger(self.name)        # 以类变量name为名获取logger对象
        return logging.LoggerAdapter(logger, {'spider': self})        # 实例化
```

```
LoggerAdapter，传递键值对给logger

    def log(self, message, level=logging.DEBUG, **kw):        # 定义函数 参数
message, level
        """Log the given message at the given log level

        This helper wraps a log call to the logger within the spider, but you
        can use it directly (e.g. Spider.logger.info('msg')) or use any other
        Python logger too.
        """
        self.logger.log(level, message, **kw)        # 将类属性（其实是函数）
logger在level级别以上记录日志

    @classmethod        # classmethod python内置的装饰器 可以通过类来调用该方法，
而不用类的实例来调用
    def from_crawler(cls, crawler, *args, **kwargs):        # 定义函数
from_crawler cls代表类本身，并且不用self参数
        spider = cls(*args, **kwargs)        # 生成一个Spider实例spider
        spider._set_crawler(crawler)        # 调用_set_crawler
        return spider        # 返回spider实例

    def set_crawler(self, crawler):
        warnings.warn("set_crawler is deprecated, instantiate and bound the "
                      "spider to this crawler with from_crawler method "
                      "instead.",
                      category=ScrapyDeprecationWarning, stacklevel=2)
 # warnings警告信息
        assert not hasattr(self, 'crawler'), "Spider already bounded to a " \
                                             "crawler"
        self._set_crawler(crawler)        # 运行_set_crawler

    def _set_crawler(self, crawler):        # _set_crawler 约定俗成的私有类
        self.crawler = crawler
        self.settings = crawler.settings
        crawler.signals.connect(self.close, signals.spider_closed)

    def start_requests(self):        # start_request
        cls = self.__class__        #
        if method_is_overridden(cls, Spider, 'make_requests_from_url'):
    # 判断函数是否被覆盖
            warnings.warn(
                "Spider.make_requests_from_url method is deprecated; it "
                "won't be called in future Scrapy releases. Please "
                "override Spider.start_requests method instead (see %s.%s)." %
(
                    cls.__module__, cls.__name__
```

```python
                ),
            )
            for url in self.start_urls:        # 若被覆盖 遍历start_urls 交给覆盖后的make_requests_from_url
                yield self.make_requests_from_url(url)
        else:
            for url in self.start_urls:        # 否则 Request
                yield Request(url, dont_filter=True)

    def make_requests_from_url(self, url):        # return Request
        """ This method is deprecated. """
        return Request(url, dont_filter=True)

    def parse(self, response):        # 若没有覆盖 则抛出NotImlementedError
        raise NotImplementedError

    @classmethod
    def update_settings(cls, settings):        # 更新设置的函数
        settings.setdict(cls.custom_settings or {}, priority='spider') # 重写后的custom_settings或者{}

    @classmethod
    def handles_request(cls, request):
        return url_is_from_spider(request.url, cls)

    @staticmethod
    def close(spider, reason):
        closed = getattr(spider, 'closed', None)
        if callable(closed):
            return closed(reason)

    def __str__(self):
        return "<%s %r at 0x%0x>" % (type(self).__name__, self.name, id(self))

    __repr__ = __str__


BaseSpider = create_deprecated_class('BaseSpider', Spider)


class ObsoleteClass(object):
    def __init__(self, message):
        self.message = message

    def __getattr__(self, name):
        raise AttributeError(self.message)
```

```python

spiders = ObsoleteClass(
    '"from scrapy.spider import spiders" no longer works - use '
    '"from scrapy.spiderloader import SpiderLoader" and instantiate '
    'it with your project settings"'
)

# Top-level imports
from scrapy.spiders.crawl import CrawlSpider, Rule
from scrapy.spiders.feed import XMLFeedSpider, CSVFeedSpider
from scrapy.spiders.sitemap import SitemapSpider

```