```
#  cookie和session组件

#  使用django放置cookie/session
    response.set_cookie(key,value)
    request.session ['key'] = value
      #此处的cookie为HttpResponse对象即可， 包括render， JsonResponse， redirect
    #加密：
    response.set_signed_cookie(key,value,salt = "加密盐")

    '''COOKIES参数
      *max_age        以秒为单位的cookie存在时间，默认为none即为浏览器关闭为止
      expires         超时时间（用的少，IE用）
      path            cookie生效的路径
      domain          cookie生效的域名
      secure          浏览器通过Https来回传cookie，默认false代表可以
      http            只能http协议传输，无法被JS获取（不是绝对，底层抓包）
    '''
    # 删除：
    response.delete_cookie(key)
    del response.session(key)

    # 获取
    request.COOKIES['key']
    request.session ['key']
    request.COOKIES.get('key')
    request.session.get('key')
    request.session.setdefault('key', 'value')     # 存在则不设置
    '''使用COOKIE的登录认证装饰器'''
      def login_auth(func):
        def inner(request,*args,**kwargs):
          next_url=request.get_full_path()
          if request.COOKIES.get('is_login'):
            return func(request,*args,**kwargs)
          else:
            return redirect('cookie_login/?next=%s'%next_url)
        return inner

    '''SESSION方法'''
      # 获取、设置、删除Session中数据
      request.session['k1']
      request.session.get('k1',None)
      request.session['k1'] = 123
      request.session.setdefault('k1',123) # 存在则不设置
      del request.session['k1']


      # 所有 键、值、键值对
      request.session.keys()
      request.session.values()
      request.session.items()
      request.session.iterkeys()
      request.session.itervalues()
      request.session.iteritems()

      # 会话session的key
      request.session.session_key

      # 将所有Session失效日期小于当前日期的数据删除
```

```
request.session.clear_expired()

# 检查会话session的key在数据库中是否存在
request.session.exists("session_key")

# 删除当前会话的所有Session数据(只删数据库)
request.session.delete()

# 删除当前的会话数据并删除会话的Cookie（数据库和cookie都删）
request.session.flush()
    这用于确保前面的会话数据不可以再次被用户的浏览器访问
    例如，django.contrib.auth.logout() 函数中就会调用它

# 设置会话Session和Cookie的超时时间
request.session.set_expiry(value)
    #* 如果value是个整数，session会在些秒数后失效
    #* 如果value是个datatime或timedelta，session就会在这个时间后失效
    #* 如果value是0,用户关闭浏览器session就会失效
    #* 如果value是None,session会依赖全局session失效策略

'''Django中的SESSION配置'''
    #1.数据库Session
    SESSION_ENGINE = 'django.contrib.sessions.backends.db'   # 引擎（默认）

    #2. 缓存Session
    SESSION_ENGINE = 'django.contrib.sessions.backends.cache'  # 引擎
    SESSION_CACHE_ALIAS = 'default'
    # 使用的缓存别名（默认内存缓存，也可以是memcache），此处别名依赖缓存的设置

    #3. 文件Session
    SESSION_ENGINE = 'django.contrib.sessions.backends.file'    # 引擎
    SESSION_FILE_PATH = None
    # 缓存文件路径，如果为None，则使用tempfile模块获取一个临时地址tempfile.gettempdir()

    #4. 缓存+数据库
    SESSION_ENGINE = 'django.contrib.sessions.backends.cached_db'        # 引擎

    #5. 加密Cookie Session
    SESSION_ENGINE = 'django.contrib.sessions.backends.signed_cookies'   # 引擎

    其他公用设置项：
    SESSION_COOKIE_NAME ＝ "sessionid"
    # Session的cookie保存在浏览器上时的key，即：sessionid＝随机字符串（默认）
    SESSION_COOKIE_PATH ＝ "/"
    # Session的cookie保存的路径（默认）
    SESSION_COOKIE_DOMAIN = None
    # Session的cookie保存的域名（默认）
    SESSION_COOKIE_SECURE = False
    # 是否Https传输cookie（默认）
    SESSION_COOKIE_HTTPONLY = True
    # 是否Session的cookie只支持http传输（默认）
    SESSION_COOKIE_AGE = 1209600
    # Session的cookie失效日期（2周）（默认）
    SESSION_EXPIRE_AT_BROWSER_CLOSE = False
    # 是否关闭浏览器使得Session过期（默认）
    SESSION_SAVE_EVERY_REQUEST = False
    # 是否每次请求都保存Session，默认修改之后才保存（默认）
```