# 环境搭建

Ubuntu

sudo apt-get install python-virtualenv

**Windows**

pip install virtualenv

激活虚拟环境

venv\Scripts\activate

deactivate

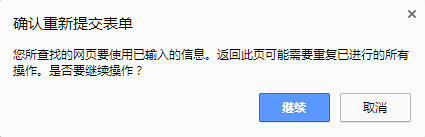
安装flask

pip install flask

# 浏览器警告

## 刷新表单重新提交

* **问题描述：**用户提交表单后，点击浏览器的刷新按钮，会看到一个警告，要求再次提交表单之前进行确认。



* **问题原因：**刷新会导致浏览器重新发送之前已经发送过的最后一个请求，如果最后一个请求是一个包含表单数据的POST请求，那么浏览器会再次提交表单，这并非理想的处理方式。
* **解决方法：**Post/重定向/Get 模式解决。不要把Post请求作为浏览器的最后一个请求，响应POST请求的是一个重定向URL，这个URL会发出Get请求，那么始终可以保证最后一个请求不是Post请求。

# 扩展

pip install flask-script

pip install flask-bootstrap

本地化日期和时间

pip install flask-moment

<http://momentjs.com/docs/#/displaying/>

## Web表单 WTForms

pip install flask-wtf

**支持HTML标准字段**

StringField 文本字段

TextAreaField 多行文本字段

PasswordField 密码文本字段

HiddenField 隐藏文本字段

DateField 文本字段，值为datetime.date 格式

DateTimeField 文本字段，值为datetime.datetime 格式

IntegerField 文本字段，值为整数

DecimalField 文本字段，值为decimal.Decimal

FloatField 文本字段，值为浮点数

BooleanField 复选框，值为True 和False

RadioField 一组单选框

SelectField 下拉列表

SelectMultipleField 下拉列表，可选择多个值

FileField 文件上传字段

SubmitField 表单提交按钮

FormField 把表单作为字段嵌入另一个表单

FieldList 一组指定类型的字段

**内建的校验函数**

Email 验证电子邮件地址

EqualTo 比较两个字段的值；常用于要求输入两次密码进行确认的情况

IPAddress 验证IPv4 网络地址

Length 验证输入字符串的长度

NumberRange 验证输入的值在数字范围内

Optional 无输入值时跳过其他验证函数

Required 确保字段中有数据

Regexp 使用正则表达式验证输入值

URL 验证URL

AnyOf 确保输入值在可选值列表中

NoneOf 确保输入值不在可选值列表中

# JinJa2 模板引擎

**变量过滤器**

safe 渲染值时不转义

capitalize 把值的首字母转换成大写，其他字母转换成小写

lower 把值转换成小写形式

upper 把值转换成大写形式

title 把值中每个单词的首字母都转换成大写

trim 把值的首尾空格去掉

striptags 渲染之前把值中所有的HTML 标签都删掉

完整的过滤器列表

<http://jinja.pocoo.org/docs/templates/#builtin-filters>

Flask-Bootstrap 模板定义

Flask-Bootstrap基模板中定义的块

块　　名说　　明

doc 整个HTML 文档

html\_attribs <html> 标签的属性

html <html> 标签中的内容

head <head> 标签中的内容

**title** <title> 标签中的内容

metas 一组<meta> 标签

styles 层叠样式表定义

body\_attribs <body> 标签的属性

body <body> 标签中的内容

**navbar** 用户定义的导航条

**content** 用户定义的页面内容

scripts 文档底部的JavaScript 声明

# CSRF 跨站请求伪造

攻击者盗用了你的身份，以你的名义发送恶意请求。CSRF能够做的事情包括：以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账......造成的问题包括：个人隐私泄露以及财产安全。



从上图可以看出，要完成一次CSRF攻击，受害者必须依次完成两个步骤：

　　1.登录受信任网站A，并在本地生成Cookie。

　　2.在不登出A的情况下，访问危险网站B。

　　看到这里，你也许会说：“如果我不满足以上两个条件中的一个，我就不会受到CSRF的攻击”。是的，确实如此，但你不能保证以下情况不会发生：

　　1.你不能保证你登录了一个网站后，不再打开一个tab页面并访问另外的网站。

　　2.你不能保证你关闭浏览器了后，你本地的Cookie立刻过期，你上次的会话已经结束。（事实上，关闭浏览器不能结束一个会话，但大多数人都会错误的认为关闭浏览器就等于退出登录/结束会话了......）

　　3.上图中所谓的攻击网站，可能是一个存在其他漏洞的可信任的经常被人访问的网站。