Flask 框架: 数据类型有六种 ; unicodeconverter anyconverter integerConverter floatConverter pathConverter UUIDConverter

Anyconverter 字符串拼接

Path 可以匹配 .\*/? & = 与unicode 区别

App = Flask(\_\_name\_\_) import\_name 表示导入模块名称,\_\_name\_\_用来确定程序所在位置,可以传入字符串,如果导入模块名,只会修改默认的static文件夹的访问,不会影响视图函数

@app.route(‘/’) 必须传入一个字符串形式的url路径,该路径以/开始

Url 可以重复 因为相同的url可以指定不同的请求方法

Redirect(可以接受字符串,变量) 重定向 本质上把当前请求返回的相应,向其他的url再次发送请求,跳转页面

作用: 当项目文件目录发生改变的情况下,可以使用重定向

Url视图函数: 带有url 的装饰器 函数

url\_for 生成指定的视图函数对应的url

自定义转换器的步骤:

1. 定义一个类继承基类转换器(baseconverter) (map是路由映射(url视图函数),),
2. 传入两个变量(args是正则表达式匹配规则)
3. 添加转换器到字典中并指定转换器名字
4. 自定义一个装饰器写入正则表达式规则

Abort()是指异常处理,只能抛出协议指定的状态码,配合errorhandler装饰器来自定义给用户提供错误提示.

Request 请求对象

请求勾子(中间件):

初始化方法类似于\_\_init\_\_

@app.

四种:

前两种 Before\_request 都是在请求前执行

区别: 执行一次 每次都执行

后两种

区别: 必须接受一个对象,返回一个对象,没有错误情况下会执行

必须接受一个参数(错误信息) 即使发生错误信息也会执行

response.headers 修改内容形式

装饰器路由实现:

Rule类 用来存储url路径映射,这个rule是个对象

Map 类 只有一个,存储一个flask中所有的rule对象

BaseConverter 负责定义匹配规则

Map Adapter负责匹配工作

Werkzeug (python工具箱 库)

Routing模块

状态保持: http协议底层是tcp/ip.三次握手,四次挥手, 返回数据后,会断开连接,下次连接相当于一次新的请求,不记得之前的请求,没有上下文请求关系.

是因为client server记录用户信息,区分 会给每个用户生成一个编号和值字典cookie,

在服务器生成,第一次请求登陆后把cookie存在于浏览器中,下次请求自动携带

cookie(用户本地数据),session:

两者都由服务器生成:

都是键值对的字符串.

cookie存在浏览器中,不安全,容易被盗取所以出现了session

Session键值对的字符串 基于cookie实现的 区别: session\_id(键)存储在客户端中

值存在服务器数据库当中. 一般会对session进行加密

Cookie(1)都是键值对的字符串 (2)有个同源策略 浏览记录未清除, 访问的网站之间存在合作关系.

有效期 : max\_age = 60 秒

Session flask中的对象session

请求上下文 : 两个对象

Request : 封装了HTTP内容

Session : 记录了用户请求信息

应用上下文 : 两个对象

Current\_app 运行过程中一直存在 保存执行过程, 作用: 在项目当中用来记录项 目日志 模块;方法 用于存储应用程序中的变量(配置,启动脚本,)

G 临时变量 临时存储当前用户信息数据

相当于请求内的全局变量,

区别: 应用上下文生命周期长特指current\_app, 请求结束后就被销毁了

四个请求勾子:

before\_first\_request

第一次请求前执行

before\_request

如果修饰函数中返回了一个响应,视图函数不在调用

after\_request

如果没有抛出错误请求后执行,接收一个响应参数,并返回

teardown\_request

每一次请求后都执行 如果有错误就抛出

flask-script扩展包:

作用: 在终端中以命令的形式启动项目

1. 导入扩展包:

From flask\_script import Manager

1. 实例化管理对象:

Manager = Manager(app)

1. 代替app调用run方法

manager.run()

Runserver----作用 -p 端口 不一样 在终端中手动指定host和post 不用手动修改代 码的host和post

Jinja2模板: 用来接收视图返回的响应数据

思想: 模块化(把具体的功能模块划分出去)功能 降低各个模块之间的关系 解耦合

让专业的事情交给更专业的模块去做

<Html>格式文件 不是特指

包含响应文本:

使用:

render\_template函数(‘模板文件’) 调用了Jinja2 模板引擎(实现过程)

作用: 不能独立使用

语法: 变量{{变量名}} 存储数据

语句: 在变量的基础上控制输出

{% For index in my\_list %}

{% if index % 7 == 0 %}

<p>{{index}}</p>

{% endif %}

{% endfor %}

语句块: python中以缩进来结束

Request请求上下文对象

Args 查询字符串

Url 中 ?后面的以=进行传参,以&号进行分割的,叫做查询字符串.

上下文:

请求上下文 :(save client and server data)

request 封装了http请求内容,get请求参数

Session 记录了用户信息,get获取用户信息

应用上下文:(save app config data demo app.name, )

current\_app 存储应用程序中的变量: 连接的数据库是哪个,ip多少,内存多少

G 变量: 临时全局变量 传递上下文数据 不同的请求,有不同的全局变量

为什么要使用状态保持:

因为 http协议的底层是tcp/ip,不具有事务记忆能力,为了提高用户体验,就需要实现状态保持,让服务器给每个用户进行编号,记录客户的信息,

实现状态保持的方法: cookie (存在于浏览器当中)session(key存在于浏览器当中,相对安全)

GET POST 区别:

两者底层都是tcp/ip GET 请求产生一个数据包(请求头和请求数据一并发送出去), POST产生两个数据包(第一次发送请求头,服务器响应100,第二次发送请求数据,200) GET请求参数会被浏览器保存,POST不会被保留, GET请求会被浏览器隐藏, POST不会除非手动设置,

GET请求效率高, POST请求相对安全