Flask入门

课程介绍

课程介绍

- 第一部分：Flask微框架的介绍
- 第二部分：URL配置及报文显示

Flask微框架的介绍

- 第一个Flask程序
- MTV模型介绍
- 启动选项及调试
- Flask的扩展

URL配置及报文显示

- URL及HTTP基础知识
- URL配置及路由
- 请求-响应及上下文对象
- 请求报文
- 响应报文
- 重定向等内部视图

什么是Flask框架

Flask介绍

- Python 实现的 Web 开发微框架
- 小而美，丰富的周边扩展

Flask的特点

- 轻松掌握
- 灵活扩展
- 免费开源

Flask使用情况

- 使用Flask搭建的网站

豆瓣douban　果壳科技有意思 Guokr.com　下厨房

- 国外公司

Netflix、Reddit、Twilio、Mailgun

安装Flask

Flask安装

- 使用pip命令安装

```
pip install flask

pip install flask -i
https://pypi.tuna.tsinghua.edu.cn/simple/
```

Flask安装

- 源码安装

```
python setup.py install
```

验证是否已经安装成功

- 验证

```
>>> import flask
>>> flask.__version__
' 1.1.2'
```

使用虚拟环境安装

- 使用virtualenv
- 使用pipenv

第一个Flask程序

第一个Flask程序

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

```

程序解释

- 参数__name__

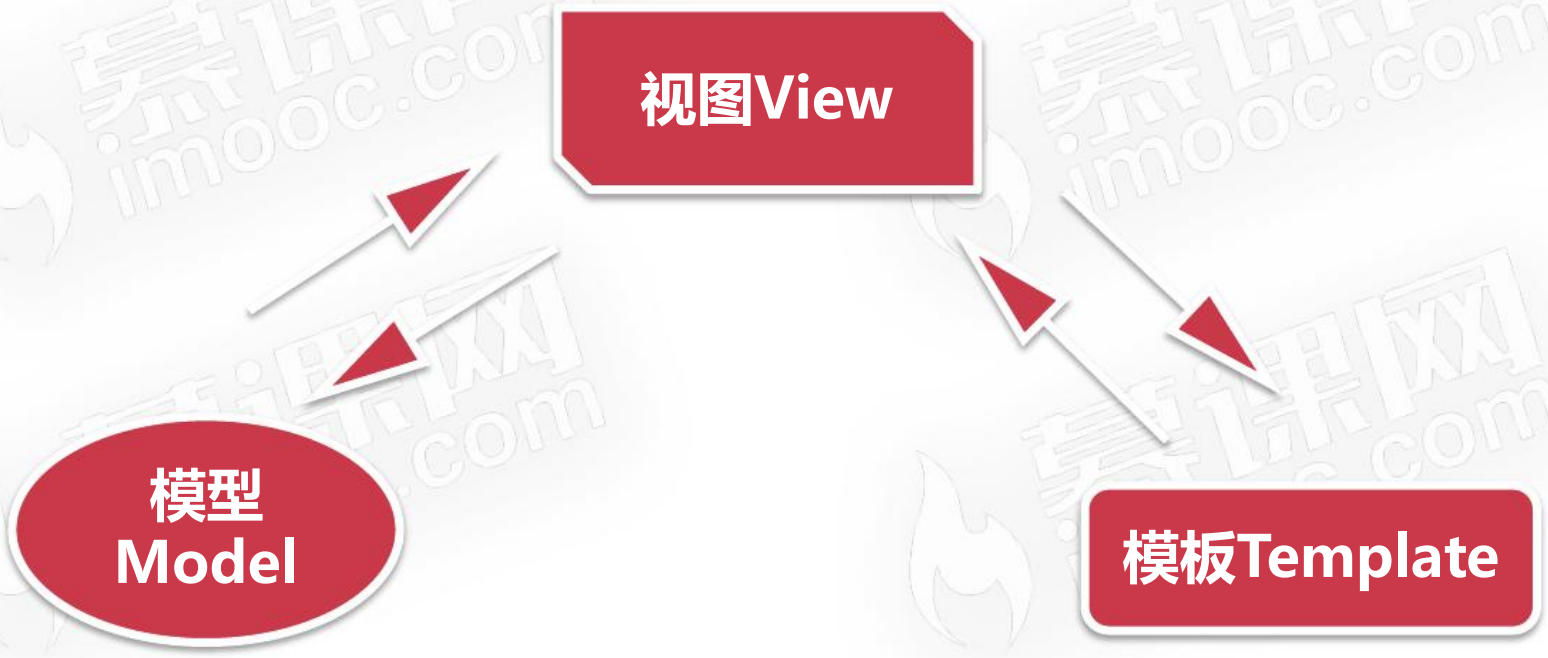
  表示Flask应用的主模块或包的名称。Flask使用该参数确定应用的位置，然后找到应用中其他文件的位置，如网页中的图片目录，模板目录

- 装饰器app.route()

  表示一个路由配置，即：用户在浏览器输入URL，使用对应的函数处理其中的业务逻辑（可写多个）

MTV模型介绍

MTV模型介绍



MTV模型介绍

- 模型Model

  Flask-PyMongo/Flask-MongoKit

  Flask-SQLAlchemy

- 视图View

  Flask-WTF/bootstrap-flask/Flask-Uploads

- 模板Template

  Jinja2

启动选项及调试

启动服务器

- 步骤1：设置环境变量

  Windows： set FLASK_APP=app.py

  *inux： export FLASK_APP=app.py

- 步骤2： flask run 启动内置web服务器

  指定IP及端口：

  flask run --host=0.0.0.0 --port=8001

  或：

  flask run -h 0.0.0.0 -p 8001

开启调试模式

- 代码修改后服务器自动重启
- 步骤1：设置环境变量

  Windows： set FLASK_ENV=development

  *inux： export FLASK_ENV=development
- 步骤2： flask run 启动内置web服务器

开启调试模式

- 直接执行app.py(v1.0以前)
- 步骤1：在文件中添加启动代码

```
if __name__ == '__main__':
    app.run(debug=True)
```

- 步骤2： python app.py
- 提示：切勿在生产环境下开启调试模式

Flask的扩展

Flask的扩展

- 什么是扩展
- 简而言之：为 Flask 应用增加功能的包

Flask的扩展

- 有哪些常用的Flask扩展？

Flask SQLAlchemy　Flask WTF

flask-mail　Flask RESTful

Flask的扩展

- 如何使用Flask扩展？

一搜 ——不要重复发明轮子

二看——查看文档、源码库

三用——对照文档使用

Flask的扩展

- 知识拓展：为Flask写扩展

英文链接>

中文链接>

URL配置及报文显示

URL及HTTP基础知识

URL介绍

网站开发基础知识补充

- 什么是URL?
- 常见的网络协议有哪些？
- 常见的HTTP请求方式有哪些？
- GET请求与POST请求有什么区别？
- 不同的HTTP响应状态码有什么含义？

URL介绍

- URL 是统一资源定位符，对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址
- 互联网上的每个文件都有一个唯一的URL

URL介绍

- 基本URL包含模式（或称协议）、服务器名称（或IP地址）、路径和文件名

  scheme://host[:port#]/path/.../[;url-params][?query-string][#anchor]

URL举例

- http://i1.mifile.cn/a4/xmad_15481237431678_LbHXJ.jpg
- https://www.mi.com
- ftp://192.168.100.2/help/readme.txt
- https://docs.djangoproject.com/en/1.11/topics/http/urls/

URL协议

- http——超文本传输协议资源
- https——用安全套接字层传送的超文本传输协议
- ftp——文件传输协议

常见的HTTP请求方式

- GET
- POST

GET 请求

- 可以用浏览器直接访问
- 请求可以携带参数，但是有长度限制
- 请求参数直接放在URL后面

POST请求

- 不能使用浏览器直接访问
- 对请求参数的长度没有限制
- 可以用来上传文件等需求

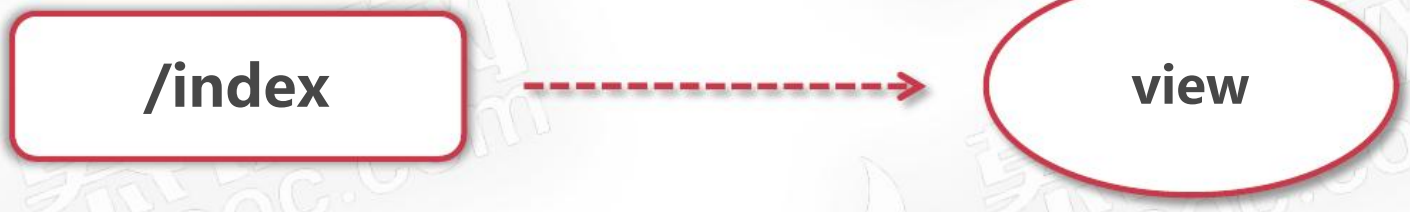
HTTP常见状态码

- 2xx 请求成功
- 3xx 重定向
- 4xx 请求错误
- 5xx 服务器错误

URL配置及路由

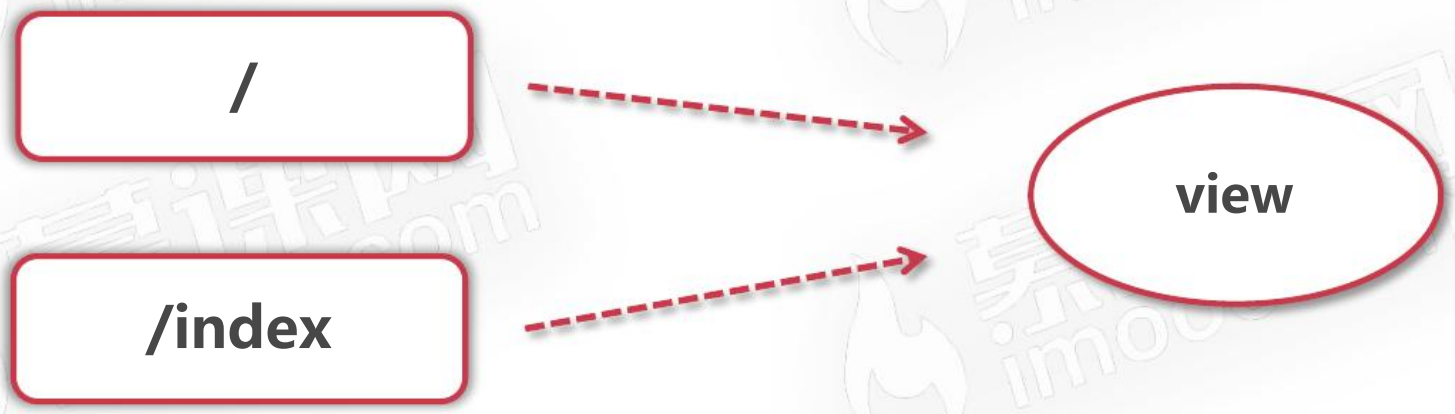
回顾一下

◆ URL与函数的关系



URL配置及路由

◆ URL与函数的关系



在IDE开发

- PyCharm专业版

路由配置

- 方式一：使用装饰器
  @app.route(url_name, methods)

- 方式二：使用API配置
  app.add_url_rule(url, url_name, view_name)

方式一：使用装饰器

- 语法规则

  @app.route(url, methods)

- 参数解释

  url：匹配的URL地址

  methods: 所支持的请求方式（['GET', 'POST']）

- 示例：

  @app.route('/login', methods=['GET', 'POST'])

方式二：使用API配置

- 语法规则

  app.add_url_rule(url, url_name, view_name)

- 参数解释

  url：匹配的URL地址

  url_name: 给URL的命名

  view_name: 视图函数

路由匹配规则

- 匹配整个文字

```
@app.route('/hello')
```

- 传递参数

```
@app.route('/user/<username>')
```

- 指定参数类型

```
@app.route('/post/<int:post_id>')
```

URL参数类型

类型	描述
string	接受任何不包含斜杠的文本（默认值）
int	接受正整数
float	接受正浮点数
path	类似 string ，但可以包含斜杠
uuid	接受 UUID 字符串

URL配置及路由

- 查看URL规则列表

  app.url_map

- URL逆向解析（根据名称解析成URL字符串）

  <1>url_for(url_name, **kwargs)

  <2>静态文件(js/css/图片)引用

  url_for('static', filename='style.css')

视图函数中获取页面传值

- URL中的值

  @app.route('/page/<page>')

  def list_user(page):

- URL中的值为可选

  @app.route('/page/<page>')

  def list_user(page=None):

视图函数中获取页面传值

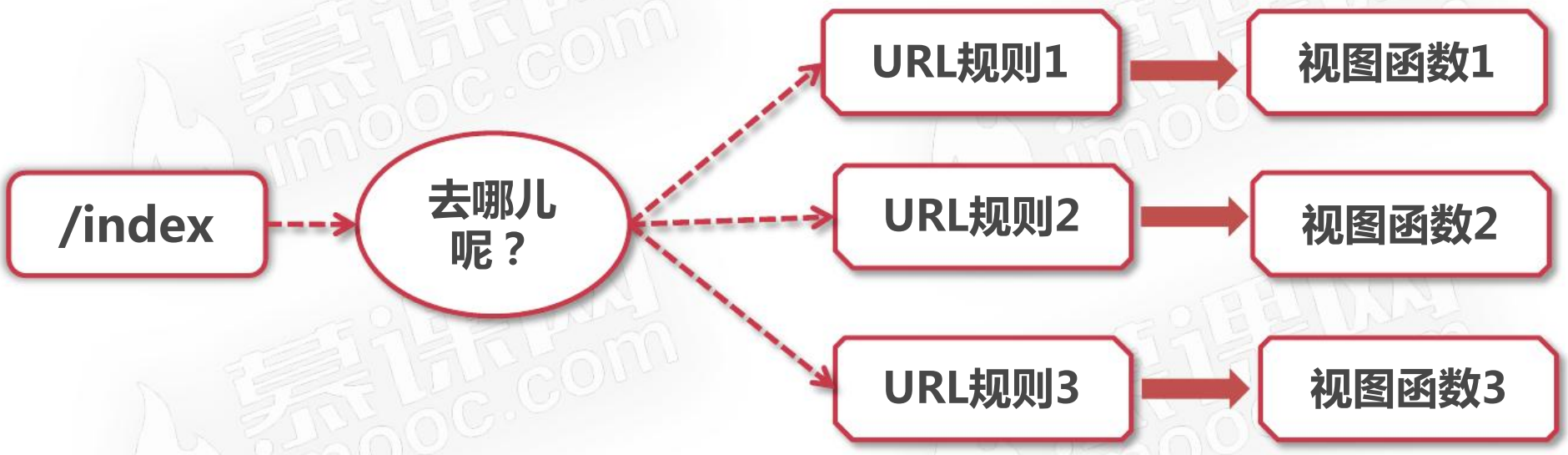
- 思考：

  GET/POST参数如何获取？

请求-响应及上下文对象

请求-响应

请求分派



url_map

上下文对象

- 应用上下文
- 请求上下文
- 在分派请求之前激活应用上下文和请求上下文
- 在请求处理完成后将其删除

理解上下文

- 带着餐具去食堂打饭

应用上下文对象

- current_app

  当前应用的实例

- g

  处理请求时的临时存储对象，每次请求都会重设这个变量

请求上下文对象

- request

  请求对象，封装了客户端发出的HTTP请求中的内容

- session

  用户会话(dict)，各请求之间的数据共享

请求报文

请求报文常用参数

- method: 请求的类型(GET/POST/OPTIONS等)
- form: POST请求数据dict
- args: GET请求数据dict
- values: POST请求和GET请求数据集合dict

请求报文常用参数

- files: 上传的文件数据dict
- cookies: 请求中的cookie dict
- headers: HTTP请求头

请求报文练习

- 获取GET参数
- 解析请求头中的IP地址

请求钩子

- 思考：如下场景如何实现？

  <1>每个请求中都要验证用户信息（是否已登录、是否有权限访问）

  <2>限制来着某些IP的恶意请求

- 使用钩子函数可以减少重复代码的编写，便于维护

请求钩子

- before_first_request

  服务器初始化后第一个请求到达前执行

- before_request

  每一个请求到达前执行

请求钩子

- after_request

  每次请求处理完成后执行，如果请求过程中产生了异常，则不执行

- teardown_request

  每次请求处理完成之后执行，如果请求过程中产生了异常也执行

响应报文

响应

- 可以是字符串
- 可以是元组（tuple）

  (response, status, headers)

  或

  (response, headers)

响应报文

- 响应元组

  <1>response——响应内容

  <2>status——响应状态码

  <3>headers——响应头信息

- 使用make_response代替

重定向等内部视图

重定向等内部视图

- redirect（ ）实现重定向
- abort（ ）处理错误

课程总结

知识点总结