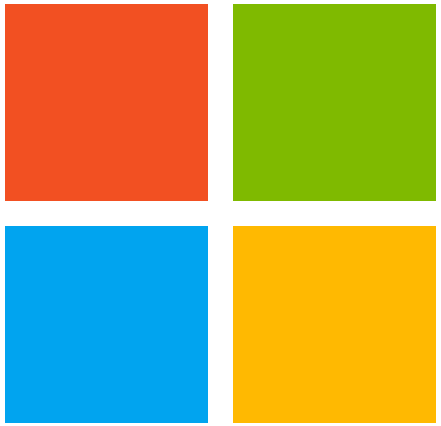


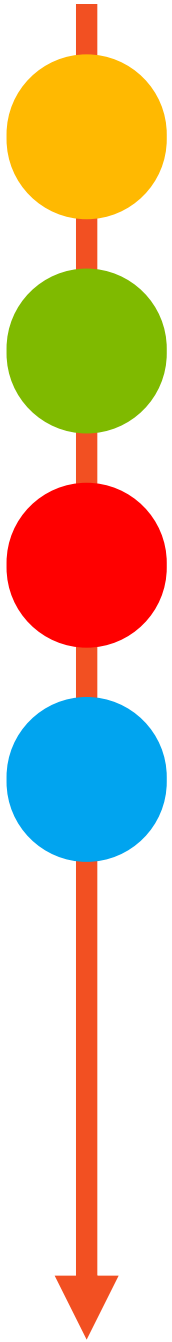
2018 TJMSC Tech. Courses

Play with Flask

施程航
Tongji Microsoft Student Club

5 19, 2019
南校区第一教学楼 102
SSE, Tongji Univ

同济微软俱乐部
Tongji Microsoft Student Club



Life is short, 不学Python

Web基础

Flask!

Misc

Python?

众所周知，Python是一门动态类型的、解释型的、需要游标卡尺的、拥有活跃社区（轮子很多）的语言

语法基础

1.Play with Python

2.内置数据结构

我们今天大概只会涉及list和dict（用起来像stl的list和map）

类型名	举例	描述
list	[1, 2, 3]	列表，有序集合
tuple	(1, 2, 3)	元组，不可变的有序集合
dict	{'a':1, 'b':2, 'c':3}	字典，无序的键值对映射
set	{1, 2, 3}	集合，具有无序性和唯一性

Web开发中的一些概念

HTML: HyperText Markup Language 超文本标记语言

CSS: Cascading Style Sheets 层叠样式表

Javascript：一门脚本语言

Web开发中的一些概念

HTML **不是**一门编程语言

HTML 是一种用来告知**浏览器如何组织页面**的**标记语言**

HTML 由一系列的**元素（elements）**组成

一对标签（ tags ）可以为一段文字或者一张图片添加超链接，将文字设置为斜体，改变字号等等

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8">
5       <title>我的测试站点</title>
6     </head>
7     <body>
8       <p>这是我的页面</p>
9     </body>
10  </html>
```

Web开发中的一些概念

CSS 用来**样式化和排版你的网页。**

例如更改网页内容的字体、颜色、大小和间距，将内容分割成多列或者加入动画以及别的装饰型效果。

Web开发中的一些概念

Javascript：用户端动态脚本，比如可以实现按钮的点击事件，发送表单

Web开发中的一些概念

HTML: 网页内容的含义与结构

CSS: 网页的表现与展示效果

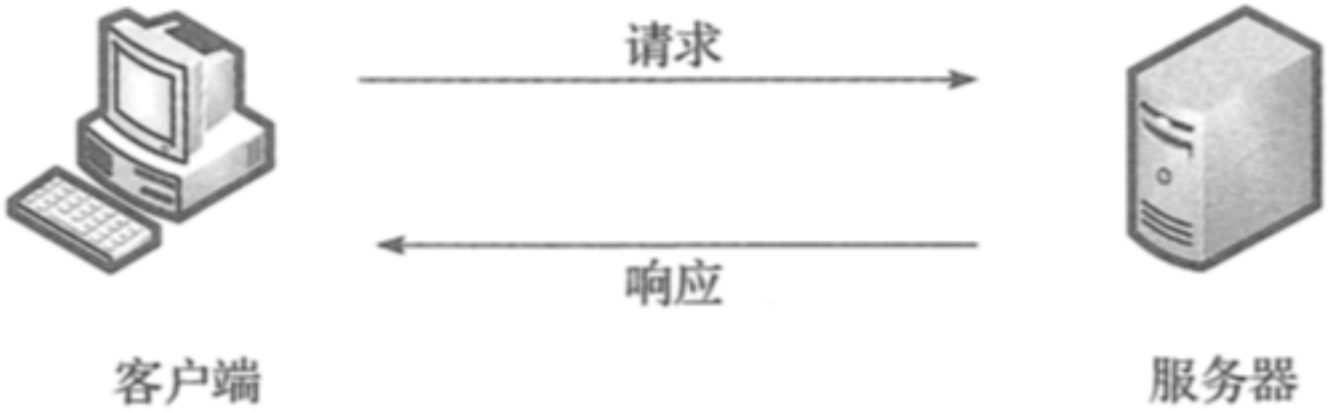
Javascript: 功能与行为

HTTP协议

HTTP协议：Hypertext Transfer Protocol,超文本传输协议

HTTP是一个客户端（用户）和服务端（网站）之间请求和应答的**标准**，通过使用网页浏览器、网络爬虫等，客户端发起一个HTTP请求到服务器上指定端口

设计HTTP最初的目的是为了提供一种发布和接收 HTML 页面的方法



HTTP请求方法

GET : 向指定的资源发出显示请求

POST : 向指定资源提交数据，请求服务器进行处理。比如上传文件

GET
HEAD
DELETE
TRACE
CONNECT
PUT
OPTIONS

HTTP状态码

200 : 请求成功

301 : 资源（网页等）永久转移至其他URL

404 : 请求的资源（网页等）不存在

500 : 内部服务器错误

HTTP状态码

1xx 消息：请求已被服务器接收，继续处理

2xx 成功：请求已成功被服务器接收、理解、并接受

3xx 重定向：需要后续操作才能完成这一请求

4xx 请求错误：请求含有词法错误或者无法被执行

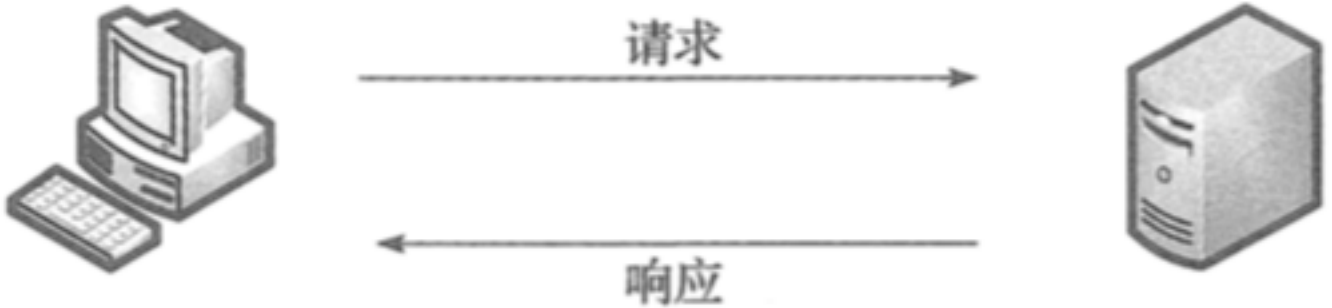
5xx 服务器错误：服务器在处理某个正确请求时发生错误

Web应用的本质

同济微软俱乐部
Tongji Microsoft Student Club

总结下：

1. 浏览器发送一个HTTP请求
2. 服务器收到请求，进行逻辑处理，生成一个HTML文档
3. 服务器把HTML文档放进HTTP响应，发送给浏览器
4. 浏览器收到HTTP响应，示



Flask!!!

Flask is a micro**framework** for Python based on ~~Werkzeug~~, ~~Jinja 2~~ and good intentions.

Armin Ronacher

Flask的由来：flask是使用Python编写的Web框架，诞生于一个愚人节玩笑。

框架是啥？大概就是帮你干了很多脏东西的那个ta。Web框架让我们不用关心底层的请求响应处理，更方便高效地编写Web程序。

不严谨地说，有了Flask，我们的主要任务大概就是绑定对应的URL（统一资源定位符）和html界面

浏览器和服务器如何对话

具体一点

发出请求？处理请求？返回响应？

Show me demo!

如果没有CSS

视图函数(View function)

视图函数，大概就是“请求处理函数”，规定**什么样的URL返回什么HTML页面**

```
@app.route('/')
def helloworld():
    return '<h1>Hello world</h1>'
```

以上代码注册了一个处理函数，它处理某个URL的请求，并返回一个Hello World 页面

视图(View)函数

注意下app.route()，事实上这是Python所谓的“装饰器”，所谓的注册事实上就是给函数套上一个装饰器帽子。

通过这个装饰器，我们把函数绑定到对应的URL，当用户在浏览器访问这个URL时，就会触发该函数，执行响应的处理逻辑并返回右侧 Hello World 页面

app.route()括号里的内容便是我们说的URL。
这里我们只要给出相对地址，可以省去前面那一串

表单(form)

HTML表单是**用户和web站点或应用程序之间交互**的主要内容之一。它们允许用户将数据发送到web站点。

HTML表单是由一个或多个小部件组成的。这些小部件可以是文本字段(单行或多行)、选择框、按钮、复选框或单选按钮。

发送和验证表单数据

表单(form)

同济微软俱乐部
Tongji Microsoft Student Club

Name

Message

Submit

在我们的程序里提交的请求是如何被处理的呢？

```
<div class="hello-form">
    <form action="/" method="post"          class="form" role="form">
        <div class="form-group required"><label class="form-control-label" for="name">Name</label>
            <input class="form-control" id="name" name="name" required type="text" value="">
        </div>
        <div class="form-group required"><label class="form-control-label" for="body">Message</label>
            <textarea class="form-control" id="body" name="body" required></textarea>
        </div>
        <input class="btn btn-secondary" id="submit" name="submit" type="submit" value="Submit">
    </form>
</div>
```

模板渲染

在一般的Web程序里，一个HTML页面通常包含各种信息。由于程序是**动态**的，所以我们需要根据不同的情况调整页面信息。

比如我们针对**不同的登录用户显示不同的欢迎信息**？把变的部分称为变量并标识出来，拿到具体的用户信息再进行**渲染（实例化）**并返回给对应的用户。

我们把包含变量和运算逻辑的HTML或其他格式的文本叫做模板，通常放在template目录下。

模板渲染

用法跟C++的模板其实差不多，就是给参数（类型T可以看成某种意义上的参数/非类型模板参数），然后实例化。

目的差不多，都是因为懒

```
<p>{{ message_board|length }} Messages</p>
<ul class="movie-list">
    {% for message in message_board %}
    <li>{{ message.name }} : {{ "      " + message.body }}</li>
    {% endfor %}
</ul>
```

模板渲染

经过渲染，变成了右侧的样子

优点？：
动态生成，不是写死的

避免冗余

```
<p>6 Messages</p>
<ul class="movie-list">

    <li>Tony :      我们</li>

    <li>Jenny :      一起</li>

    <li>Gogo :      学猫叫</li>

    <li>张三 :      一起</li>

    <li>李四 :      喵喵喵</li>

    <li>X五 :      喵喵</li>

</ul>
```

访问网页时用的不是IP地址？

Web工作方式：

在浏览器中输入URL，首先浏览器请求DNS(Domain Name System，一项服务)服务器，获取对应域名对应的IP

通过IP地址找到IP对应的服务器后，要求建立TCP连接，浏览器发送HTTP请求包

服务器接收请求包，调用自身服务，返回HTTP响应包

客户端从响应中拿出Response包的主体，断开连接，渲染页面

```
@app.route('/msg-board', methods=['POST', 'GET'])
def index():
    # 如果是POST请求
    if request.method == 'POST':
        _name = request.form.get('name')
        _body = request.form.get('body')

        # TODO: 验证数据

        message_board.append({'name': _name, 'body' : _body})
    #     flash('Im added!')
        return redirect(url_for('index'))

    return render_template('index.html', name=name, message_board =
message_board)
```

访问网页时用的不是IP地址？

默认端口：http 80, https 443

Chrome开发者工具

现实场景下的Web开发

从stateless(cookie, session)到keep-alive

每次处理表单请求都返回一个html? **AJAX**(Asynchronous JavaScript and XML)

安全

数据持久化

……

其他

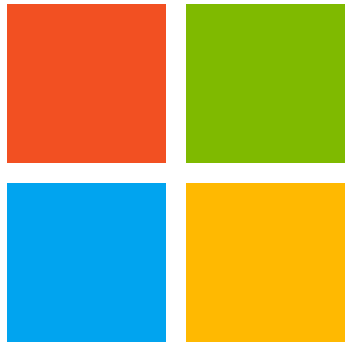
Web基础可以看下MDN https://developer.mozilla.org/zh-CN/docs/learn

Flask相关可以看下 http://helloflask.com/tutorial/ （demo借鉴处..）

官方tutorial http://flask.pocoo.org/docs/1.0/

如果还有兴趣可以看看狗书、狼书一类的

可以稍微了解下pipenv



Microsoft

同济微软俱乐部
Tongji Microsoft Student Club

加入TJMSC QQ群

关注TJMSC微信公众号