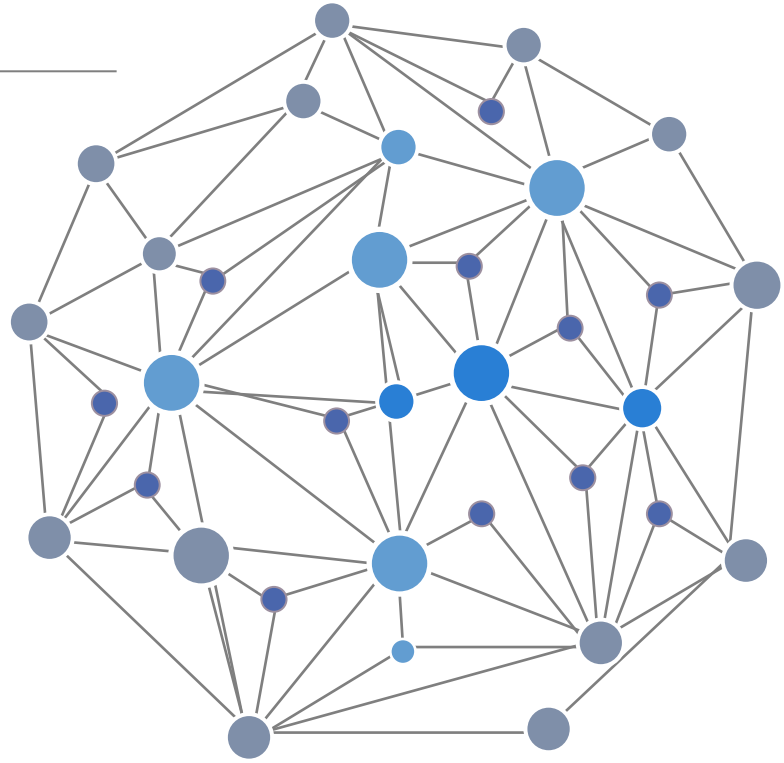
Web 程序设计

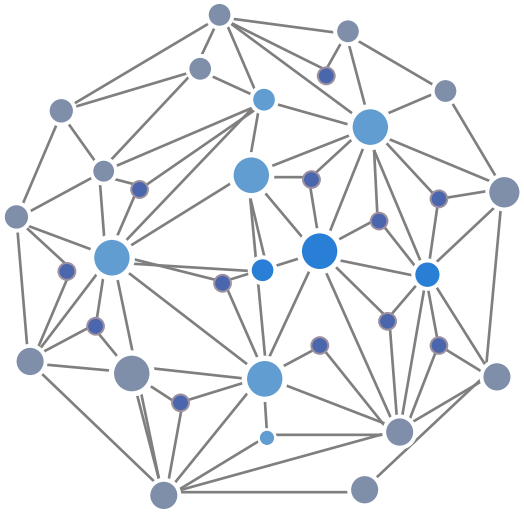


第二讲 HTTP 协议简介与 Web 应用程序组成分析

福州大学 计算机与大数据学院
软件工程系 陈昱

内容提要

- HTTP 协议及其相关标准简介
  - 统一资源定位符 URL
  - HTTP 协议
- Web 应用程序的组成分析
  - 客户端代码
  - 服务器代码

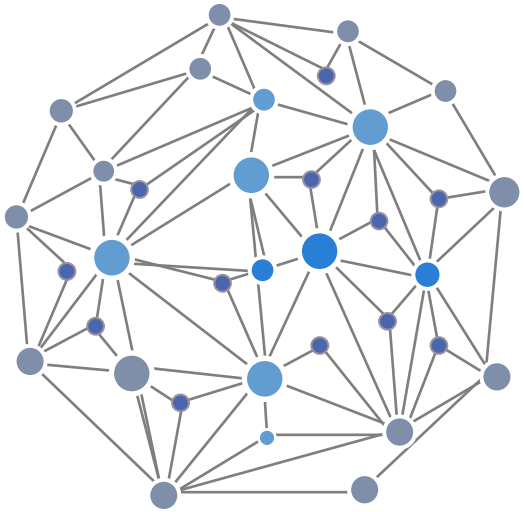


HTTP 协议
及其相关标准简介

HTTP 协议及其相关标准简介

- 统一资源定位符 URL

- HTTP 协议
  - 请求报文
  - 响应报文



统一资源定位符 URL

Uniform Resource Locators

RFC 1738

统一资源定位符 URL

- 怎样标识分布在整个因特网上无数的万维网文件？
- 使用 **统一资源定位符 URL** (Uniform Resource Locator) 来标识万维网上的文档
- 使每一个资源在整个因特网的范围内具有唯一的标识符： ***URL***

URL 的一般形式

- 由（**://**）隔开的两大部分组成

- URL 的一般形式是：

<URL的访问协议>://<主机名>:<端口>/<路径>/<文件名>

HTTP 使用的 URL：主机名

- HTTP使用的 URL 的一般形式

http://<**主机**>:<端口>/<路径>/<文件名>

这里是服务器的主机名 Hostname

- **域名** *DomainName* 是在域名商处注册，并可以通过全球 **DNS** 系统解析的全球唯一的主机名
  - Godaddy.com

DNS（ Domain Name System ）

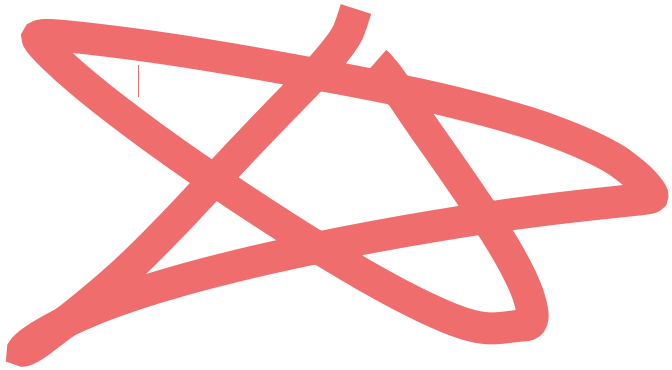
- 域名系统是将域名和 IP 地址相互映射的一个分布式数据库
- 能够使人更方便的访问互联网，而不用去记忆被机器使用的数字 IP
- 在互联网上我们可以通过付费在域名商那里注册域名，通过其 DNS 管理系统将域名与我们的服务器 IP 绑定
- DNS 服务使用 TCP 和 UDP 端口 53

HTTP 使用的 URL：端口号

- HTTP 使用的 URL 的一般形式

http://<主机>:<**端口**>/<路径>/<文件名>

HTTP 的默认端口号是 80，通常可省略；
如果不是 80，则必须标注出来，例如 8080



HTTP 使用的 URL：路径信息

- HTTP 使用的 URL 的一般形式

http://<主机>:<端口>/<路径>/<文件名>

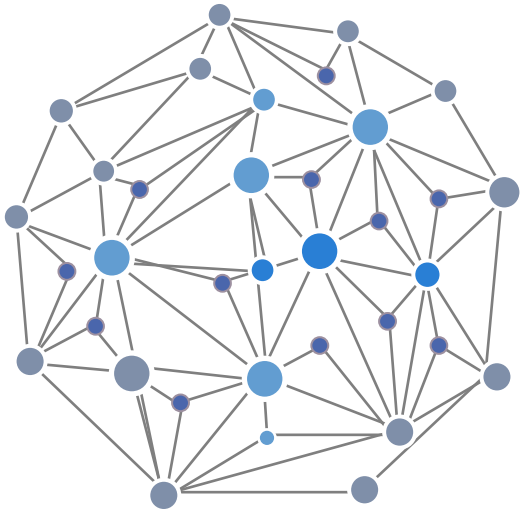
若省略文件的<路径>/<文件名>项，则
URL 就指向网站服务器的某个默认文件

- 默认文件名通常为 index.html, index.htm, index.php 等

HTTP 使用的 URL：查询字符串

http://zh.wikipedia.org:80/wiki/Search?search=铁路&go=Go

？后面的部分是查询(Query)，是客户端发送给服务器端程序的参数，参数之间用 & 分隔



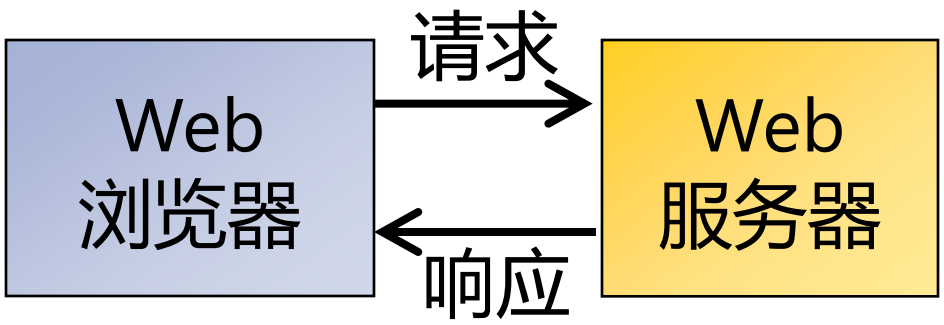
HTTP 协议

WWW 的基础协议

RFC 2616

HTTP HyperText Transfer Protocol

- 超文本传输协议
  - WWW (Web) 的基础协议
- 客户端/服务器模式
  - 客户端：浏览器请求、接收、展示 Web 内容
  - 服务器：Web 服务器对请求进行响应,发送内容



HTTP 协议的版本

- HTTP/0.9
  - 古老的版本
  - 1991, Tim Berners-Lee
- HTTP/1.0
  - 1995, 开始成为重要的面向事务的应用层协议
- HTTP/1.1
  - 1999, 目前最常用的版本
  - 增加了持久连接，缓存等新功能，改进了性能

HTTP 协议的版本

- HTTP/2.0
  - 大幅度的提升了 Web 性能，在与 HTTP/1.1 完全语义兼容的基础上，进一步减少了网络延迟
  - 二进制分帧，多路复用，请求优先级，流量控制，服务器端推送以及首部压缩等新改进
- HTTP/2 简介
- 用 Chrome 查看 HTTP2 Demo
  - https://http2.akamai.com/demo

HTTP 协议的工作流程

HTTP 是应用层协议，工作在 TCP 之上：

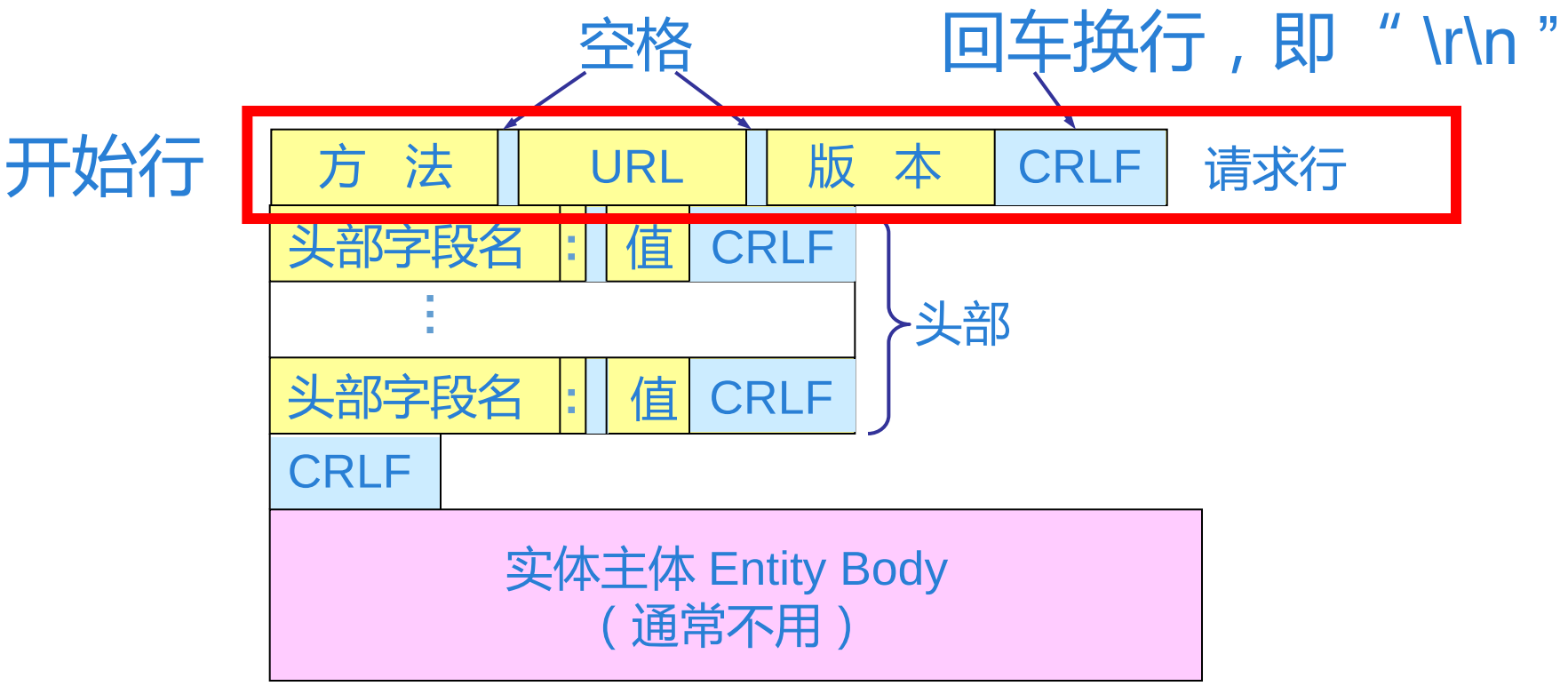
1. 客户端发起 TCP 连接到服务器, 端口 80
2. 服务器接受来自客户端的 TCP 连接
3. HTTP 报文 (应用层协议报文) 在浏览器 (http client) 和 Web服务器 (http server) 之间进行交换
4. 关闭 TCP 连接

HTTP 的协议报文结构

- HTTP 有两类报文：***Request***, ***Response***

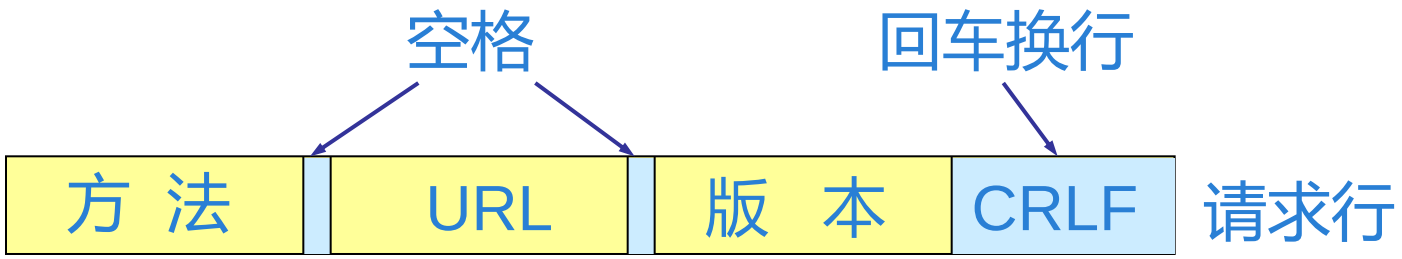
- **请求报文:** 客户端向服务器发送的请求消息
- **响应报文:** 服务器给客户端的回应消息

HTTP 请求报文结构



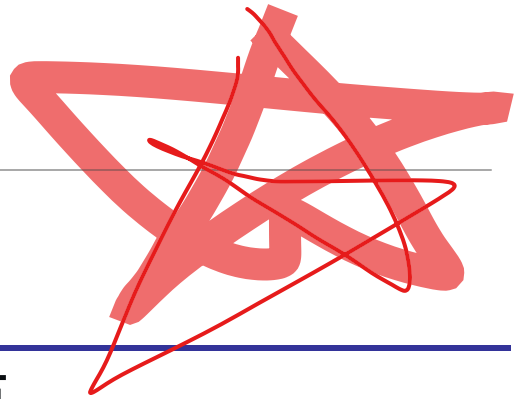
报文由三个部分组成，即开始行、头部行和实体主体
在请求报文中，开始行称为请求行

HTTP 请求报文 — 请求行



- “方法” 就是对所请求的对象进行的操作
- “URL” 是所请求的资源的 URL 路径
- “版本” 是 HTTP 的版本号

HTTP 请求报文方法



方法	意义
GET	向服务器请求指定的资源
HEAD	只返回HTTP头部而没有内容
POST	请求服务器修改存储在服务器上的信息
PUT	请求服务器在服务器上生成和替换资源
DELETE	请求服务器删除服务器上的资源
OPTION	请求服务器列出对指定资源所有可用的请求方法
TRACE	请求服务器在收到请求头部后返回它
CONNECT	用于SSL和代理服务器

GET 方法

- 最简单的 GET 方法报文：

  GET URL HTTP/1.1

  Host: HostName

- 请求行后跟随一些可选的字段
  - 字段由字段名和字段值构成
  - 字段名和字段值之间用冒号隔开
  - Host 字段对于 HTTP 1.1 是必须的

Firefox 发起连接到 fzu

- GET / HTTP/1.1

Host	www.fzu.edu.cn
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding	gzip, deflate, br
Accept-Language	zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Connection	keep-alive
Cookie	JSESSIONID=0000t2ZZAmW1tw3UtPrc6Essu4Y:18f1dopuc; ElvisLives-47873=AGOHENDLFAAA

过滤消息头 拦截 重发

GET https://www.fzu.edu.cn/

状态 200 OK

版本 HTTP/1.1

传输 42.61 KB（大小 42.38 KB）

响应头 (236 字节) 原始

Cache-Control: no-cache
Connection: close
Content-Language: en-US
Content-Type: text/html; charset=utf-8
Date: Tue, 03 Nov 2020 01:00:09 GMT
Pragma: No-cache
Server: IBM_HTTP_Server
Transfer-Encoding: chunked

请求头 (473 字节) 原始

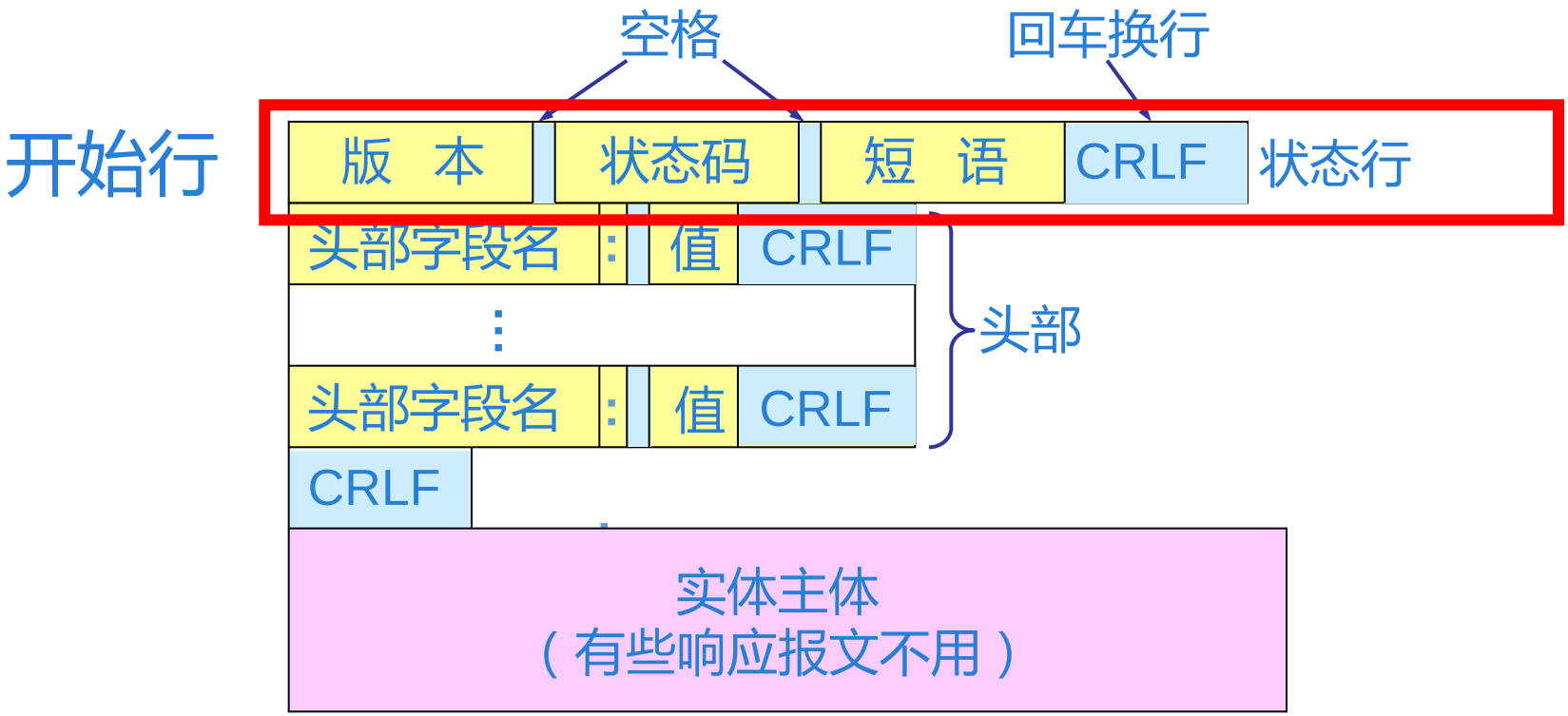
```
GET / HTTP/1.1
Host: www.fzu.edu.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: JSESSIONID=0000t2ZZAmW1tw3UtPrc6Essu4Y:18f1dopuc; ElvisLives-47873=AGOHENDLFAAA
Upgrade-Insecure-Requests: 1
```

请求报文的头部信息

- Host　　服务器主机名(服务器通过它支持虚拟主机)
- User-Agent　　指明浏览器的版本
- Referer　　告诉服务器用户从哪里跳转来
- Accept　　浏览器支持的媒体类型
- Authorization　　可以发送用户名、密码，用于访问受限的文档

……

HTTP 的响应报文结构

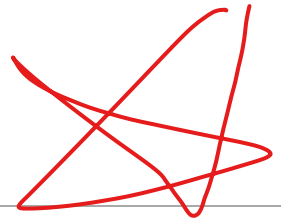


响应报文的开始行是**状态行**。
状态行包括三项内容，即 **HTTP 的版本**，**状态码**，以及解释状态码的**简单短语**。

状态码都是三位数字

- 1xx　　表示通知信息，如请求收到了或正在进行处理。
- 2xx　　表示成功，如接受或知道了。
- 3xx　　表示重定向，表示要完成请求还必须采取进一步的行动。
- 4xx　　表示客户的差错，如请求中有错误的语法或不能完成。
- 5xx　　表示服务器的差错，如服务器失效无法完成请求。

常见状态码和短语



- **200 OK** 请求已经成功处理
- 301 Moved permanently 文档已经转移到新地址
- 304 Not modified Cache中的文档是最新的
- 400 Bad request 客户端请求存在错误
- 403 Forbidden 客户端不允许访问该资源
- **404 Not found** 文件未找到
- 500 Internal server error 服务器内部错误
- 503 Service unavailable 服务器过载(超负荷)

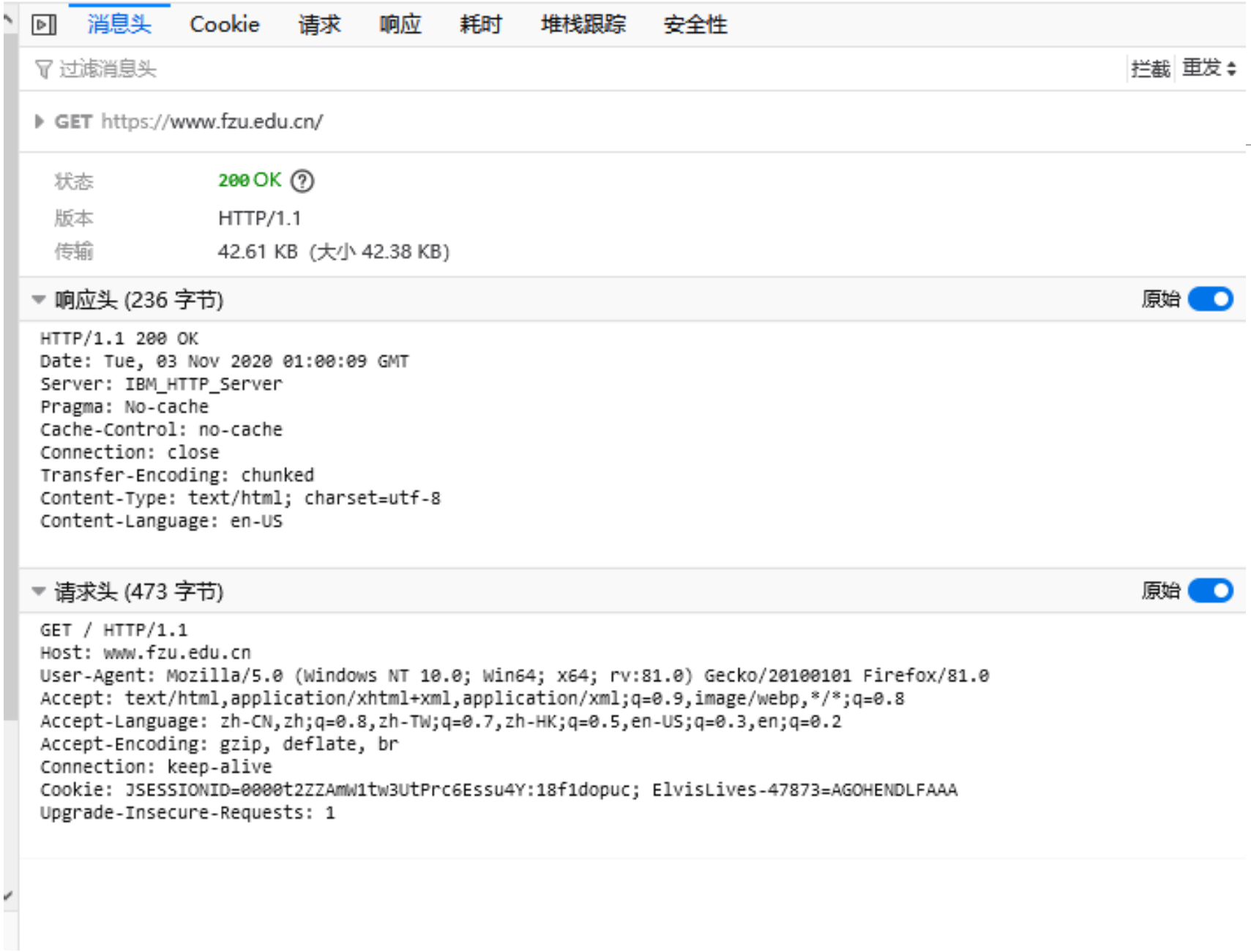
响应报文的其他头部信息

- Date　　当前的日期/时间
- Server　　Web服务器信息
- Last Modified　　请求文档的最近修改时间
- Expires　　请求文档的过期时间
- Content-Type　　连接的媒体类型
- Connect-length　　数据的长度
- Connect-encoding　　说明有无使用压缩技术
- Transfer-encoding　　说明采用的编码变换类型
- ……

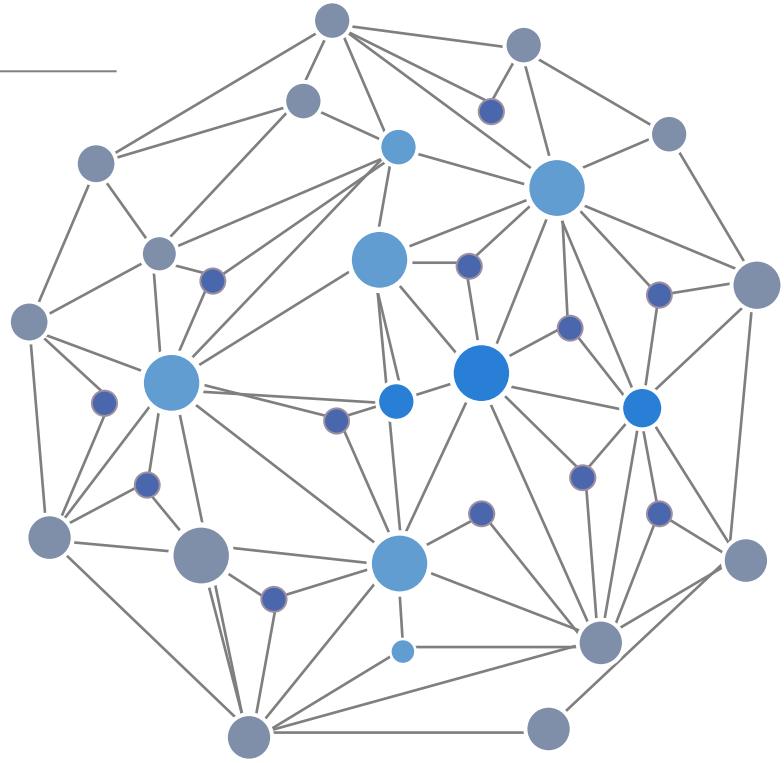
fzu 返回给 Firefox 的信息

- HTTP/1.1 200 OK

Date	Tue, 03 Nov 2020 01:00:09 GMT
Server	IBM_HTTP_Server
Cache-Control	No-cache
Transfer-Encoding	chunked
Content-Type	**text/html; charset=utf-8**
Content-Language	en-US
Pragma	No-cache


消息头
Cookie
请求
响应
耗时
堆栈跟踪
安全性
过滤消息头
拦截
重发
GET https://www.fzu.edu.cn/
状态 200 OK
版本 HTTP/1.1
传输 42.61 KB（大小 42.38 KB）
响应头 (236 字节)
原始
HTTP/1.1 200 OK
Date: Tue, 03 Nov 2020 01:00:09 GMT
Server: IBM_HTTP_Server
Pragma: No-cache
Cache-Control: no-cache
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
Content-Language: en-US
请求头 (473 字节)
原始
GET / HTTP/1.1
Host: www.fzu.edu.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: JSESSIONID=0000t2ZZAmW1tw3UtPrc6Essu4Y:18f1dopuc; ElvisLives-47873=AGOHENDLFAAA
Upgrade-Insecure-Requests: 1

Web 应用程序的组成分析



Web 应用程序的组成分析

- 程序功能，架构解说
- 客户端代码（HTML+CSS）
- 服务器端代码（PHP）
- 安装部署

程序功能

- 这是一个简单的 Web 应用程序
- 包括简单的客户端表单和服务器端处理程序
- 实现简单的货物订单处理

Web 应用程序常见的工作流程 1/2

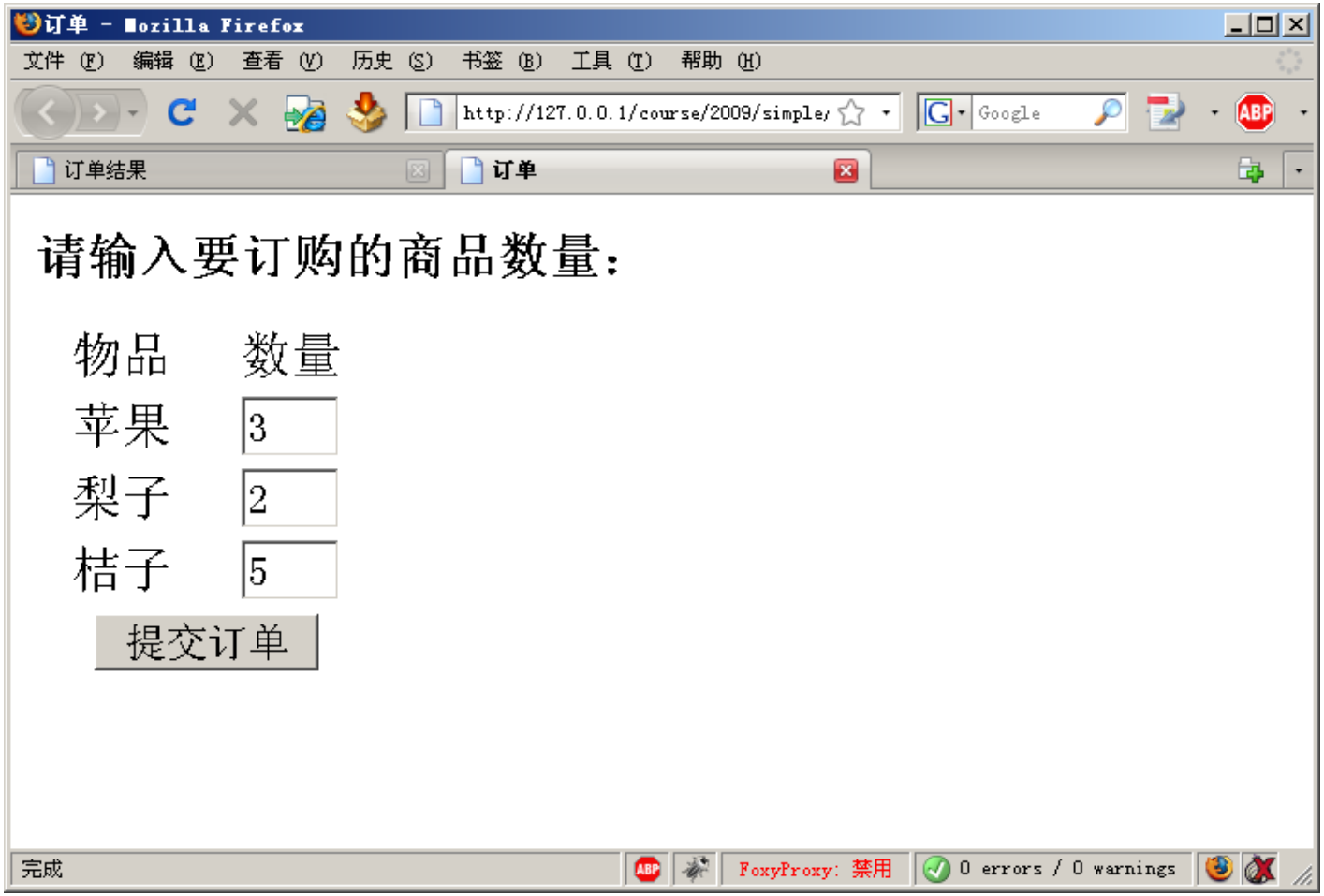
1. 服务器提供一个页面，让用户通过浏览器输入信息
2. 用户完成了信息输入后，浏览器将数据提交给服务器
3. 服务器端程序利用提交的数据进行计算，然后向浏览器返回一个新的页面，将计算结果通知给用户

Web 应用程序常见的工作流程 2/2

- 有时，浏览器可能还会直接请求执行服务器中的某个程序，程序执行的结果将返回给浏览器

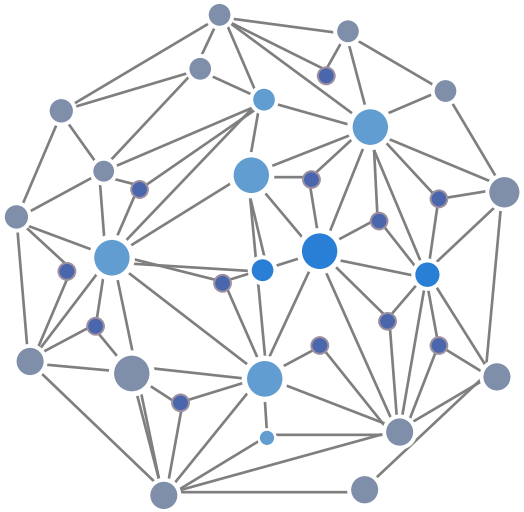
- 比如访问大多数动态网站的首页的时候，这时并未提交信息

订单提交


订单 - Mozilla Firefox
文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)
http://127.0.0.1/course/2009/simple/
Google
订单结果
订单
请输入要订购的商品数量:
物品 数量
苹果 3
梨子 2
桔子 5
提交订单
完成
FoxyProxy: 禁用
0 errors / 0 warnings

订单处理的返回结果


订单结果 - Mozilla Firefox
文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)
http://127.0.0.1/cours
Google
订单结果
订单结果
网上水果店
订单结果
订单接受时间： 2009年2月13日 09:54
你的订单如下：
共 10 个水果：
3 个苹果
2 个梨子
5 个桔子
总价：￥9.50
完成
FoxyProxy: 禁用
0 errors / 0 warnings



客户端代码

客户端

- 客户端代码采用简单的 HTML+CSS 形式
- HTML 用于存储页面结构层的信息，CSS 用于存储页面的排版布局信息
- 出于简单考虑，省略了一些内容的介绍

客户端代码 1

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta http-equiv="Content-type" content="text/html;
    charset=UTF-8" />
  <title>订单</title>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
```

一些常用标记 1

- <!DOCTYPE...> 文档类型声明
- <html> 根元素
- <head> 头元素
- <meta>主要用于设置字符集
- <title> 文档标题
- <link> 链接 CSS 样式表

客户端代码 2

```
<body>
<h1>请输入要订购的商品数量：</h1>
<form action="processorder.php" method="get">
<table>
<tr>
  <td>物品</td>
  <td>数量</td>
</tr>
<tr>
  <td>苹果</td>
  <td><input type="text" name="appleqty" size="3"
  maxlength="3" /></td>
</tr>
```

一些常用标记 2

- <body> 文档主体
- <h1> 标题一，类似的还有 <h2> <h3>...
- <p> 段落
- <table>表格
  - <tr> 行
  - <td> 单元格
- 无序列表
  - 列表中的元素（list item）

表单 Form

- 表单是重要的输入工具，用户通过它提交数据给服务器端处理程序
- 表单本身只是负责输入，需要编写一个程序来处理表单传递的数据

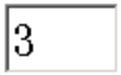
```
<form action="processorder.php"
        method="get">
    ……
</form>
```

表单标记中的属性

- 行为属性 **action** 告诉表单提交的时候将内容发往哪个程序（ **processorder.php** ）
- 可选的方法属性 **method** 告诉表单数据将怎样发送，有 **get** 和 **post** 两种方法

```
<form action="processorder.php"
        method="get">
    ……
</form>
```

表单中的元素

- <input type="text" name="appleqty" size="3" maxlength="3" />
  - type="text" 是文本输入框 

- <input type="submit" value="提交订单" />
  - type="submit" 是提交按钮 

GET 方法

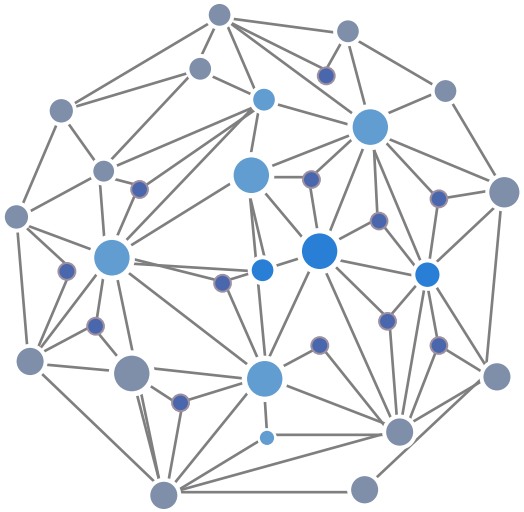
- http://xxxxxxxxxxx/processorder.php?**appleqty=3**&**pearqty=2**&**orangeqty=5**

- get 方法将表单数据按照 name=value 的形式添加到 action 所指向的 URL 后面

- 剩下的事情由 processorder.php 完成

配上简单的 CSS

```
h1 {
   font-size: 1em;
}
tr {
   text-align: center;
}
td {
   width: 50px;
}
```



服务器端代码

服务器端程序的作用

- 服务器端程序通过表单获得用户提交的数据，并完成相应计算
- 通常，表单中的每一项输入在程序中表现为一个变量
- 服务器端程序完成计算后返回一个结果页面

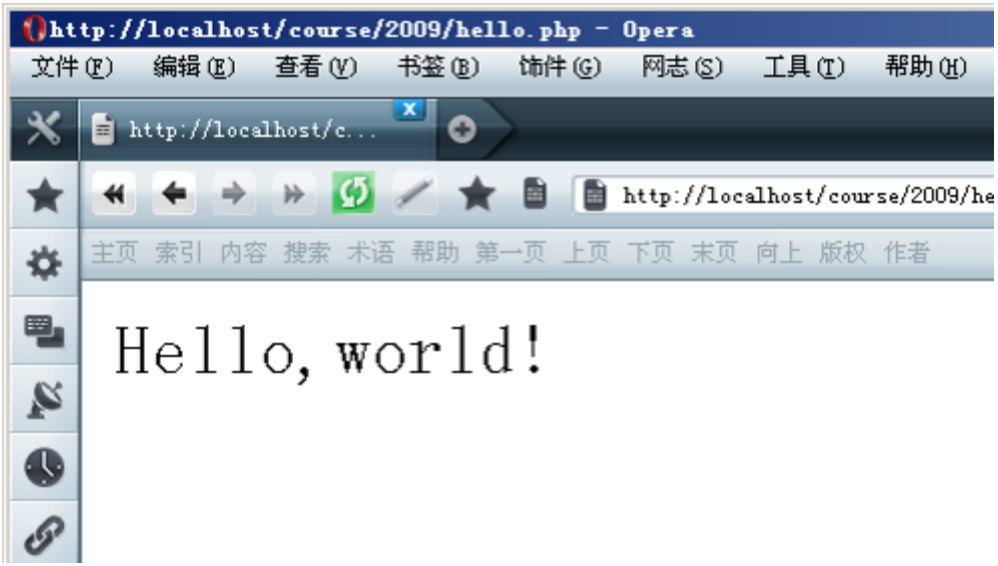
PHP 的特点

- PHP 是一种服务器端的解释型语言
- PHP 最重要的用途是编写服务器端的处理程序，包括表单处理，数据库访问，文件操作等功能
- PHP → PHP Hypertext Preprocessor

PHP 程序工作原理

- 当客户端请求一个 PHP 页面的时候，Web 服务器会调用 PHP 解释器解析执行 PHP 文件，并将 PHP 程序产生的输出（HTML 等代码）发送给浏览器

```
<?php
  echo "Hello,world!";
?>
```



PHP 基础语法 1

- PHP 代码使用 <?php ... ?> 标记嵌入 (X)HTML/XML 文档中
  - 当 PHP 解析一个文件时，会寻找开始和结束标记，并执行其中的代码
  - 凡是在一对开始和结束标记之外的内容都会被 PHP 解析器当成文本忽略（而直接输出给客户端）
- 每个语句后用分号结束指令

服务器端代码 1

```
<?php
  // 获取表单提交的数据
  $appleqty = $_GET['appleqty'];
  $pearqty = $_GET['pearqty'];
  $orangeqty = $_GET['orangeqty'];

  // 定义一些常量
  define('APPLE_PRICE', 1 .00);
  define('PEAR_PRICE', 2.00);
  define('ORANGE_PRICE', 0.50);
?>
```

PHP 基础语法 2

- PHP 是一种弱类型的程序语言
- 也就是说一个变量可以存储任意类型的数据，使用变量之前无须声明变量是字符型还是整型
- 与 C/C++/Java 语言有本质区别
- 变量用一个美元符号后面跟变量名来表示

服务器端代码 2

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta http-equiv="Content-type" content="text/html;
    charset=UTF-8" />
  <title>订单结果</title>
</head>
<body>
<h1>网上水果店</h1>
<h2>订单结果</h2>
```

在<?php ... ?>之外，
不做改变直接输出到客户端

这是结果页面的开始部分

服务器端代码 3

```
<?php
date_default_timezone_set('Asia/Shanghai');
echo '<p>订单接受时间：  ';
echo date('Y年n月j日 H:i');
echo "</p>\n";

echo '<p>你的订单如下: </p>';

$totalqty = 0;
$totalqty = $appleqty + $pearqty + $orangeqty;
echo "<p>共 ".$totalqty. " 个水果：</p>\n";
```

设置时区，输出当前时间

输出水果数量

服务器端代码 3 执行结果

- 订单接受时间： 2009年2月22日 10:09
- 你的订单如下: 共 10 个水果：

服务器端代码 4

```
if( $totalqty == 0 ) {
  echo '<p>你没有在前一页中订购任何物品~ </p>';
} else {
  echo "<ul>\n";
  if ( $appleqty>0 ）  echo "<li>".$appleqty." 个苹果</li>\n";
  if ( $pearqty>0 ）  echo "<li>".$pearqty." 个梨子</li>\n";
  if ( $orangeqty>0 ) echo "<li>".$orangeqty." 个桔子
    </li>\n";
  echo "</ul>\n";
}
```

服务器端代码 4 执行结果

-
- 2 个苹果
- 3 个梨子
- 5 个桔子
-

服务器端代码 5

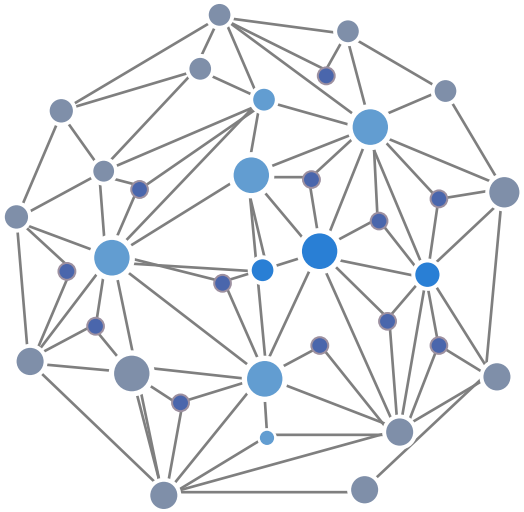
```
$totalamount = 0.00;

$totalamount = $appleqty * APPLE_PRICE
             + $pearqty * PEAR_PRICE
             + $orangeqty * ORANGE_PRICE;

echo "<p>总价:
    ￥".number_format($totalamount,2)."</p>\n";
?>
</body>
</html>
```

服务器端代码 5 执行结果

- 总价: ￥10.50
- </body>
- </html>



安装部署

Apache + PHP

- Apache 与 PHP 皆为开源软件
- 支持各大操作系统
- 可从官方网站下载 Windows 版自行安装配置

WAMP 套装

- 不过执行安装配置比较繁琐
- 所以有人专门制作了 Apache + MySQL + PHP 的打包，称为 AMP 套装
- 例如：WampServer

  http://www.wampserver.com/en/

上传（拷贝）文件，就这么简单

- 将编写好的 网页，PHP程序拷贝到 WAMP 安装目录下的 www 目录里即可通过 http://localhost 访问

- 如果是远程的服务器，通常用 FTP 或 SFTP 上传到 Apach 的 DocumentRoot 文档根目录
  - Linux 常为 /var/www

总结 Conclusion

- Web 应用程序分成两个部分
  - 客户端程序
  - 服务器端程序
- 表单是用于收集用户输入的工具
- 服务器端程序常用于处理表单输入

第二讲课后练习

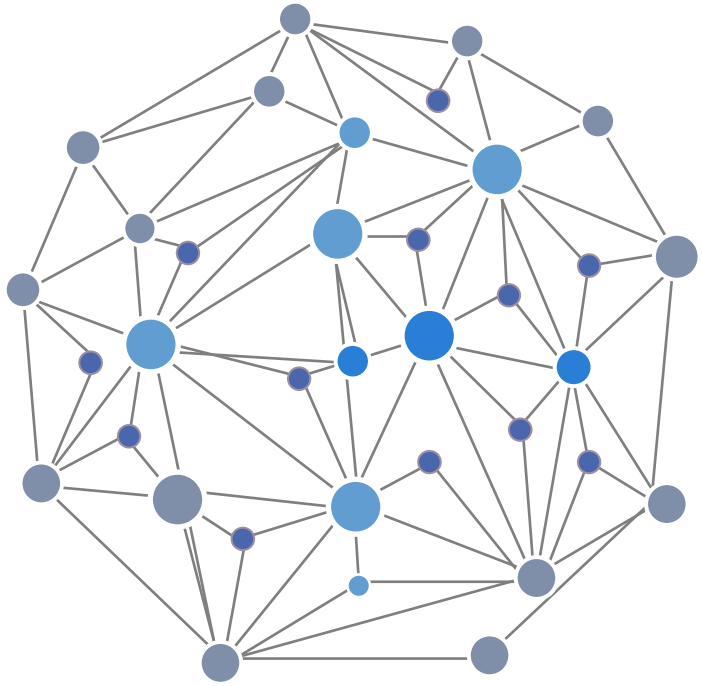
- 从群共享下载 WAMP 并安装

- 将第二讲的程序在自己本机运行调试
  - 例子目录下提供该程序源代码
  - 也可以自己根据课件编写出该程序

- 测试页面（校园网）：
  - http://sw.fzu.edu.cn/WebProgramming/class_samples/PHP/simpleorder/orderform.html

THANKS

本章结束



福州大学 计算机与大数据学院 软件工程系