中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION

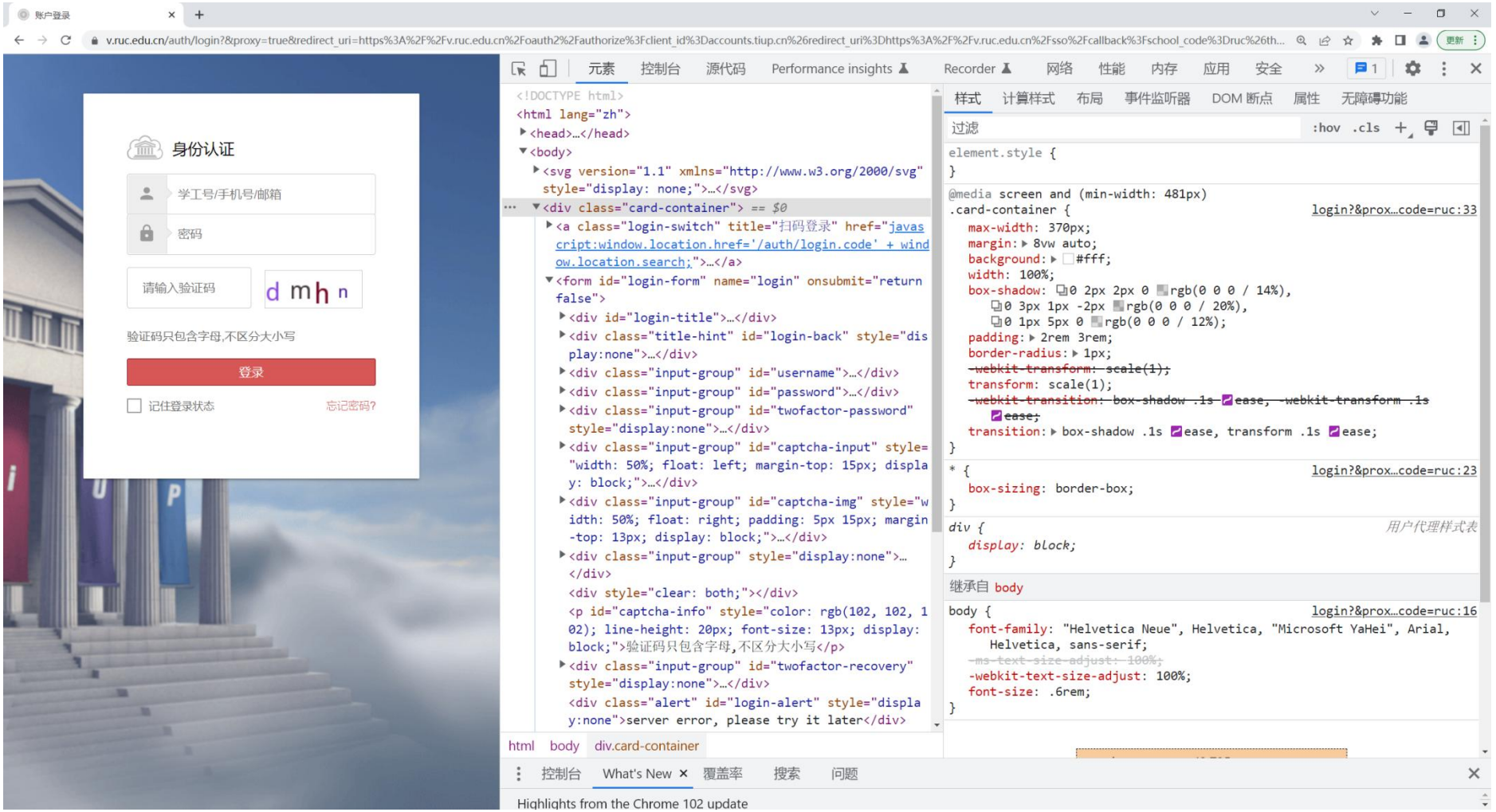
3. Web应用开发——后端

授课教师：游伟 副教授

授课时间：周五16:00 – 17:30（教二2406）

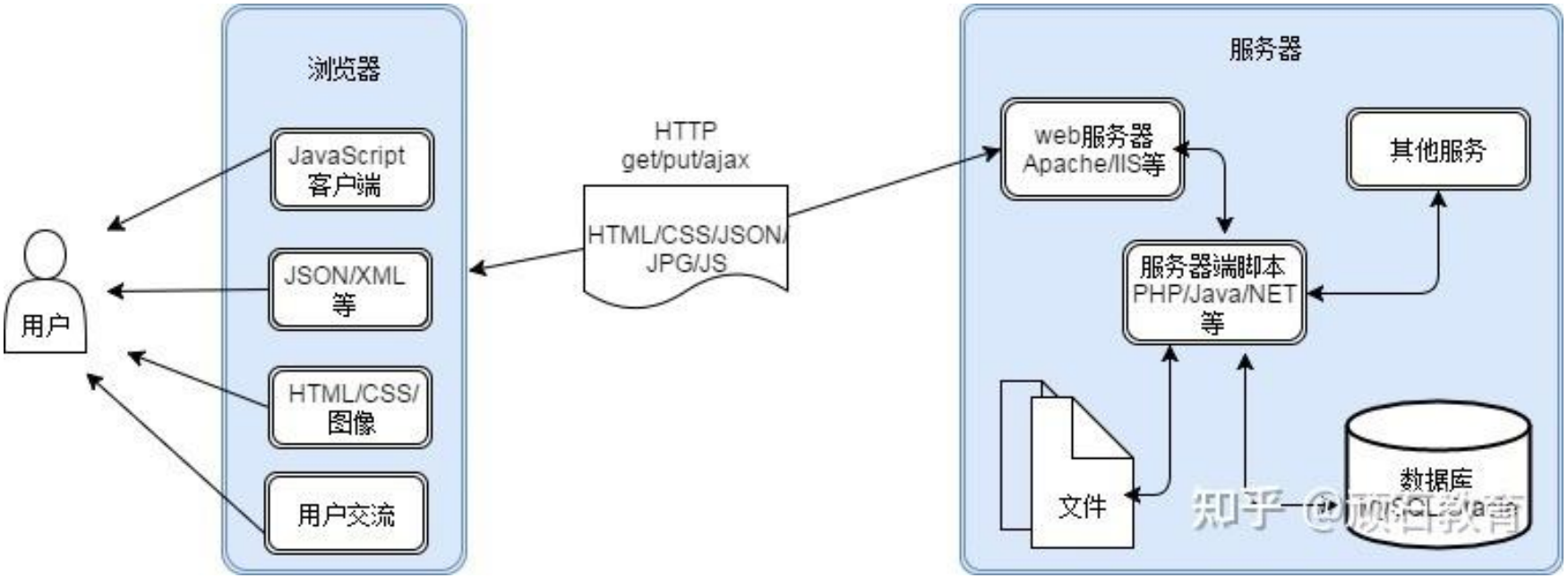
课程主页：https://www.youwei.site/course/websecurity

引子

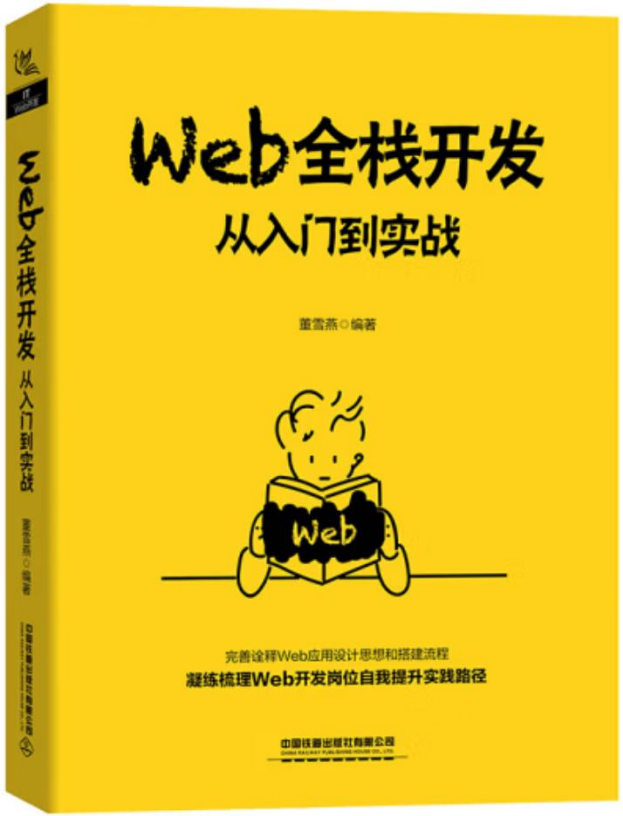

账户登录
v.ruc.edu.cn/auth/login?&proxy=true&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Foauth2%2Fauthorize%3Fclient_id%3Daccounts.tiup.cn%26redirect_uri%3Dhttps%3A%2F%2Fv.ruc.edu.cn%2Fsso%2Fcallback%3Fschool_code%3Druc%26th...
身份认证
学工号/手机号/邮箱
密码
请输入验证码
d mh n
验证码只包含字母,不区分大小写
登录
记住登录状态
忘记密码?
元素 控制台 源代码 Performance insights Recorder 网络 性能 内存 应用 安全
样式 计算样式 布局 事件监听器 DOM 断点 属性 无障碍功能
过滤 :hov .cls
element.style {
}
@media screen and (min-width: 481px)
.card-container { login?&prox…code=ruc:33
max-width: 370px;
margin: 8vw auto;
background: #fff;
width: 100%;
box-shadow: 0 2px 2px 0 rgb(0 0 0 / 14%), 0 3px 1px -2px rgb(0 0 0 / 20%), 0 1px 5px 0 rgb(0 0 0 / 12%);
padding: 2rem 3rem;
border-radius: 1px;
-webkit-transform: scale(1);
transform: scale(1);
-webkit-transition: box-shadow .1s ease, -webkit-transform .1s ease;
transition: box-shadow .1s ease, transform .1s ease;
}
* { login?&prox…code=ruc:23
box-sizing: border-box;
}
div { 用户代理样式表
display: block;
}
继承自 body
body { login?&prox…code=ruc:16
font-family: "Helvetica Neue", Helvetica, "Microsoft YaHei", Arial, Helvetica, sans-serif;
-ms-text-size-adjust: 100%;
-webkit-text-size-adjust: 100%;
font-size: .6rem;
}
html body div.card-container
控制台 What's New 覆盖率 搜索 问题
Highlights from the Chrome 102 update

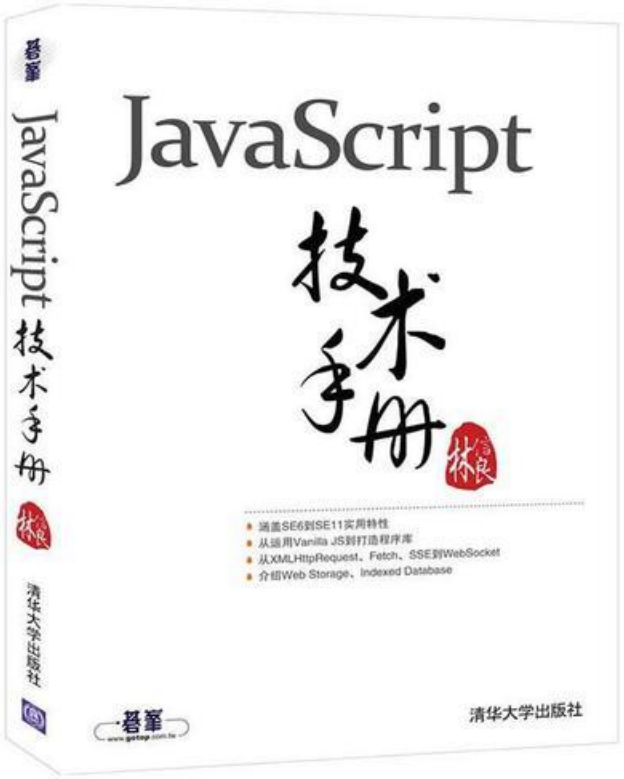
Web应用开发模式

- 前端技术：HTML + CSS + JavaScript
- 后端技术：PHP/ASP/JSP + 数据库 + 数据处理
- 前后端通信：HTTP + AJAX



推荐图书

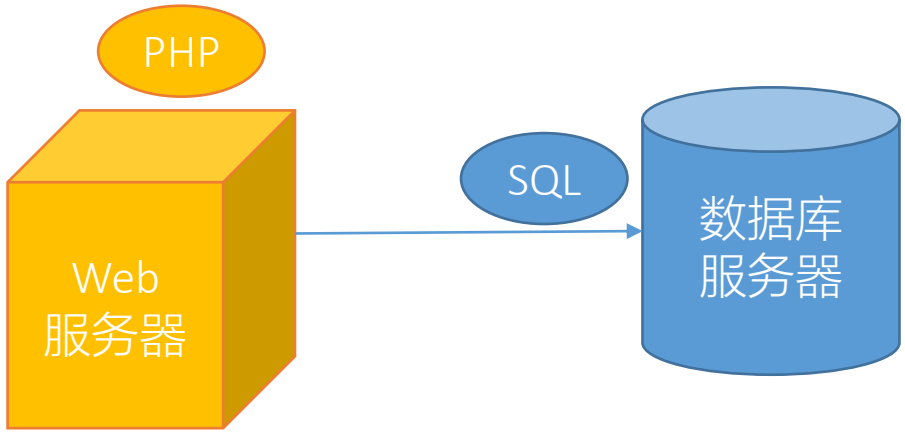

Web全栈开发
从入门到实战
董雪燕 编著
Web
完善诠释Web应用设计思想和搭建流程
凝练梳理Web开发岗位自我提升实践路径
中国铁道出版社有限公司


JavaScript
技术手册
林良
涵盖SE6到SE11实用特性
从运用Vanilla JS到打造程序库
从XMLHttpRequest、Fetch、SSE到WebSocket
介绍Web Storage、Indexed Database
碁峰
清华大学出版社


第2版
PHP核心技术与最佳实践
PHP Core Technology and Best Practice, Second Edition
PHP核心技术
与最佳实践
机械工业出版社
China Machine Press

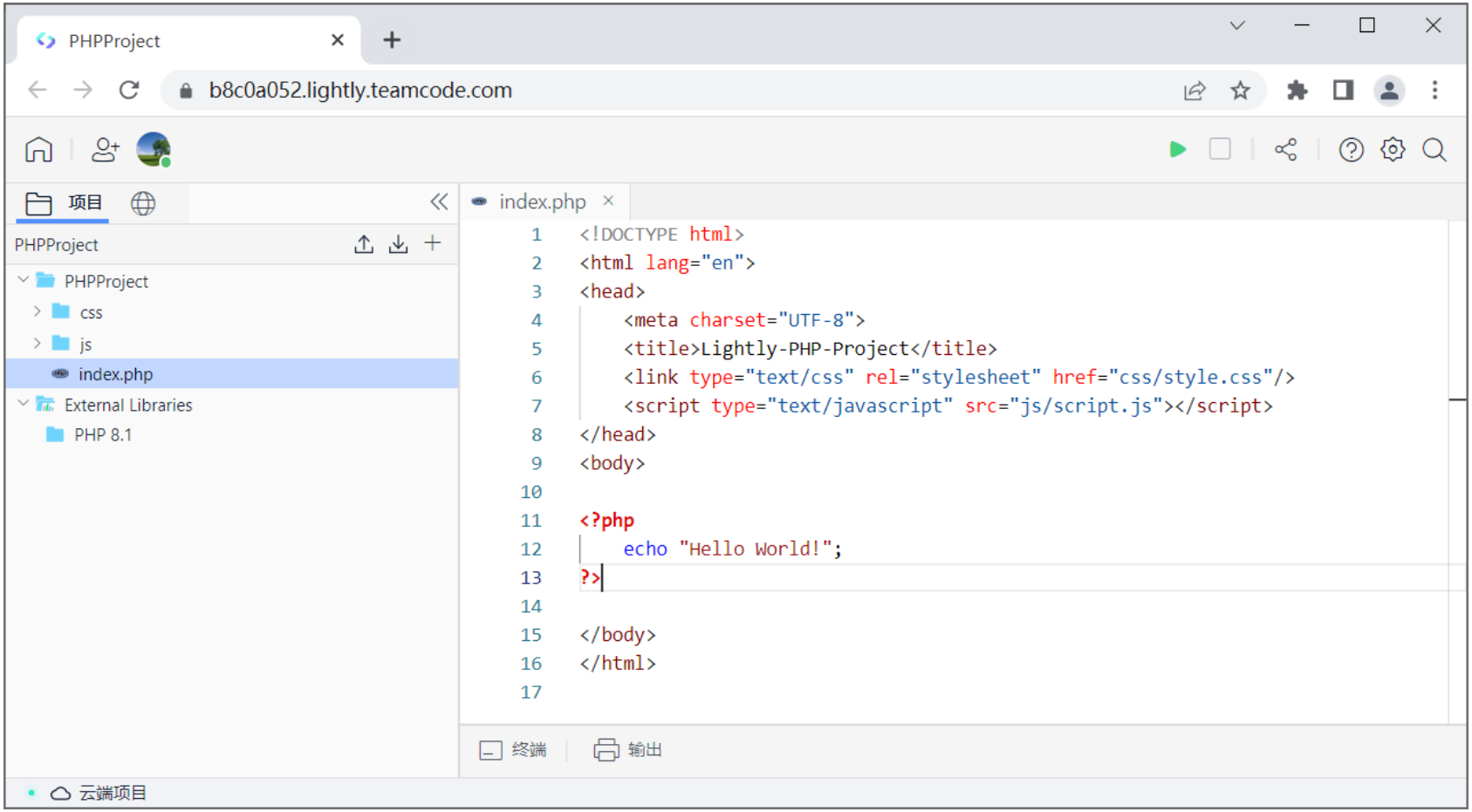
Web后端架构

- Web服务器：Apache / IIS
- 数据库服务器：MySQL / SQLServer
- 服务器端脚本语言：PHP / JSP / ASP
- 数据库结构化查询语言：SQL



后端PHP在线编辑平台

- https://lightly.teamcode.com/php

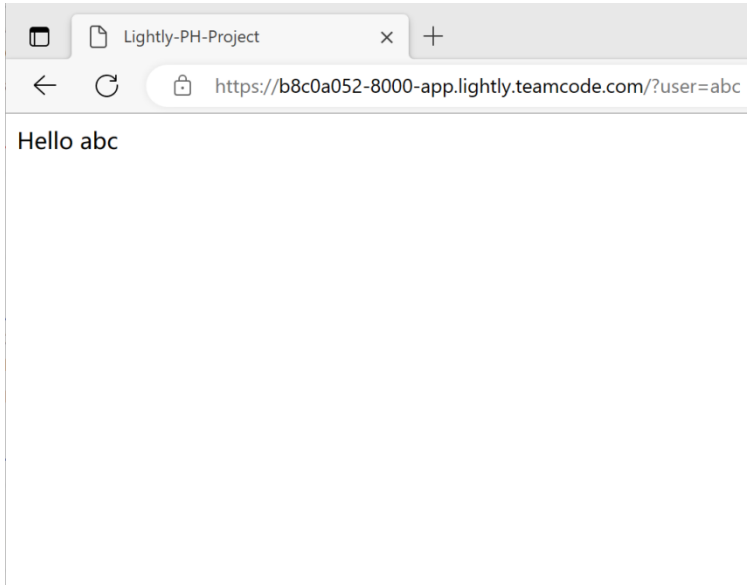


目录

3.1 PHP

- PHP是PHP：Hypertext Preprocessor（超文本预处理器）的缩写，是一种服务器端执行的脚本语言，用于生成动态页面内容

```
<html>
  <head>
    <title>第一个PHP代码</title>
  </head>

  <body>
    <span>Hello </span>
    <?php
      $user = $_GET["user"];
      echo $user;
    ?>
  </body>
</html>
```



学习更多：https://www.w3school.com.cn/php/index.asp

3.1.1 在网页中引入PHP代码

- XML标记风格（推荐）

```
<?php
  echo "this is xml style";
?>
```

- 简短标记风格

```
<?
  echo "this is xml style";
?>
```

- 脚本标记风格

```
<script language="php">
  echo "this is xml style";
</script>
```

- ASP标记风格

```
<%
  echo "this is xml style";
%>
```

3.1.3 常量

- 预定义常量

常 量 名	功 能
__FILE__	默认常量，PHP程序文件名
__LINE__	默认常量，PHP程序行数
PHP_VERSION	内建常量，PHP程序的版本，如“3.0.8_dev”
PHP_OS	内建常量，执行PHP解析器的操作系统名称，如“Windows”
TRUE	这个常量是一个真值（True）
FALSE	这个常量是一个假值（False）
NULL	一个null值
E_ERROR	这个常量指到最近的错误处
E_WARNING	这个常量指到最近的警告处
E_PARSE	这个常量指解析语法有潜在问题处
E_NOTICE	这个常量为发生不寻常，但不一定是错误处

```php
<?php
  echo "本文件路径和文件名为：".__FILE__."<br/>";
  echo "当前行数为：".__LINE__."<br/>";
  echo "当前的操作系统为：".PHP_OS."<br/>";
  echo "当前的PHP版本为：".PHP_VERSION."<br/>";
?>
```

```
本文件路径和文件名为：/workspace/PHPProject/index.php
当前行数为：13
当前的操作系统为：Linux
当前的PHP版本为：8.1.9
```

```php
<?php
  define("DEFAULT_PATH", "/var/www/");
  define("NORMAL_USER", "0");
  echo DEFAULT_PATH."<br/>";
  echo constant("NORMAL_USER")."<br/>";
  echo defined("DEFAULT_PATH")."<br/>";
  echo defined("HELLO_WORLD")."<br/>";
?>
```

```
/var/www/
0
1

END
```

- 自定义常量
  - 使用define()函数声明常量
  - 使用constant()函数或常量名获取常量的值
  - 使用defined()函数判断常量是否已经被定义

3.1.4 变量

- 变量采用美元符号（$）加一个变量名的方式来表示
- 变量在使用前不需要预先定义，可以直接在使用时定义变量
- 变量赋值：给变量一个具体的数据值
  - 传值赋值：等号两边的变量值互不影响，任何一个变量值的改变都不会影响另一个变量
  - 引用赋值：等号两边的变量指向内存中同一存储空间，任何一个变量值的改变都会引起另一个变量的变化

```php
<?php
  echo "使用传值方式赋值";
  $a = "hello";
  $b = $a;                          //将变量$a的值赋值给变量$b
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
  $a = "hello world";               //改变变量$a的值
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
?>
```

```
使用传值方式赋值变量a的值为hello
变量b的值为hello
变量a的值为hello world
变量b的值为hello
```

```php
<?php
  echo "使用传值方式赋值";
  $a = "world";
  $b = &$a;                         //将变量$a的引用赋值给变量$b
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
  $a = "hello world";               //改变变量$a的值
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
?>
```

```
使用传值方式赋值变量a的值为world
变量b的值为world
变量a的值为hello world
变量b的值为hello world
```

3.1.4 变量

- 变量检测与销毁
  - 使用isset()函数判断给定变量是否存在
  - 使用unset()函数销毁给定变量
- 可变变量：变量名称不是预先定义好的，而是动态地设置和使用
  - 指使用一个变量的值作为另一个变量的名称，又称为变量的变量
  - 可变变量通过在一个变量名称前使用两个美元符号（$$）符号实现

```php
<?php
  $a = 123;
  if (isset($a)) echo "\$a exists<br/>";

  $a = null;
  if (!isset($a)) echo "\$a not exist<br/>";
  echo "value of \$a: $a<br/>";

  unset($a);
  if (!isset($a)) echo "\$a not exist<br/>";
  echo "value of \$a: $a<br/>";
?>
```

```
$a exists
$a not exist
value of $a:
$a not exist

Warning: Undefined variable $a in /workspace/PHPProject/index.php on line 21
value of $a:
```

```php
<?php
  $a = "hello"; // 声明变量 $a
  $$a = "world"; // 声明变量 $hello
  echo $a."<br/>";
  echo $hello."<br/>";
  echo $$a."<br/>";
?>
```

```
hello
world
world
```

3.1.5 数据类型

基本数据类型

类　型	说　明	检测函数
boolean（布尔型）	这是最简单的类型。只有两个值，真值（true）和假值（false）	is_bool
string（字符串型）	字符串就是连续的字符序列，可以是计算机能表示的一切字符的集合	is_string
integer（整型）	整型数据类型只能包含整数。可以是正整数或负整数	is_integer/is_int
float/double（浮点型）	浮点数据类型wygd用来存储数字，和整型不同的是它有小数位	is_float/is_double

is_numeric

复合数据类型

类　型	说　明	检测函数
array（数组）	将多个相互关联的数据组织在一起形成一个整体，作为一个单元使用	is_array
object（对象）		is_object

特殊数据类型

类　型	说　明	检测函数
resource（资源）	资源是一种特殊的数据类型，保存着对外部资源的引用，如打开的文件、数据库连接等	--
null（空值）	空值表示没有为变量设置任何值	is_null

3.1.6 类型转换

- PHP是弱类型脚本语言，无需声明变量类型，运行时根据需要，自动确定变量类型并进行数据类型转换
- 类型转换分为：隐式类型转换和显式类型转换
  - 隐式类型转换：对不同类型值使用运算符时，值可在类型之间自动转换
  - 显式类型转换：开发人员在编写代码时，强制指定对值的类型进行转换

3.1.6 类型转换

Converting to boolean

When converting to bool, the following values are considered **false**:

- the boolean **false** itself
- the integer 0 (zero)
- the floats 0.0 and -0.0 (zero)
- the empty string, and the string "0"
- an array with zero elements
- the special type NULL (including unset variables)
- SimpleXML objects created from attributeless empty elements, i.e. elements which have neither children nor attributes.

Every other value is considered **true** (including any resource and **NAN**).

Converting to string

A bool **true** value is converted to the string "1". bool **false** is converted to "" (the empty string). This allows conversion back and forth between bool and string values.

An int or float is converted to a string representing the number textually (including the exponent part for floats). Floating point numbers can be converted using exponential notation (`4.1E+6`).

Arrays are always converted to the string "Array"; because of this, echo and print can not by themselves show the contents of an array. To view a single element, use a construction such as `echo $arr['foo']`. See below for tips on viewing the entire contents.

In order to convert objects to string, the magic method __toString must be used.

Resources are always converted to strings with the structure "Resource id #1", where 1 is the resource number assigned to the resource by PHP at runtime. While the exact structure of this string should not be relied on and is subject to change, it will always be unique for a given resource within the lifetime of a script being executed (ie a Web request or CLI process) and won't be reused. To get a resource's type, use the get_resource_type() function.

null is always converted to an empty string.

As stated above, directly converting an array, object, or resource to a string does not provide any useful information about the value beyond its type. See the functions print_r() and var_dump() for more effective means of inspecting the contents of these types.

Most PHP values can also be converted to strings for permanent storage. This method is called serialization, and is performed by the serialize() function.

Converting to integer

From booleans

false will yield `0` (zero), and **true** will yield `1` (one).

From floating point numbers

When converting from float to int, the number will be rounded *towards zero*. As of PHP 8.1.0, a deprecation notice is emitted when implicitly converting a non-integral float to int which loses precision.

If the float is beyond the boundaries of int (usually `+/- 2.15e+9 = 2^31` on 32-bit platforms and `+/- 9.22e+18 = 2^63` on 64-bit platforms), the result is undefined, since the float doesn't have enough precision to give an exact int result. No warning, not even a notice will be issued when this happens!

From strings

If the string is numeric or leading numeric then it will resolve to the corresponding integer value, otherwise it is converted to zero (`0`).

From NULL

null is always converted to zero (`0`).

From other types

The behaviour of converting to int is undefined for other types. Do *not* rely on any observed behaviour, as it can change without notice.

学习更多：
https://www.php.net/manual/en/language.types.type-juggling.php

3.1.7 字符串

- 字符串的定义
  - 使用单引号定义字符串：单引号串中的内容被认为是普通字符直接输出
  - 使用双引号定义字符串：双引号串中的内容可以被解释而且替换
  - 使用定界符定义字符串：原样输出定界符中的内容，变量用其值来替换

```php
<?php
  $ruc = "中国人民大学";
  $str1 = "I Like \"$ruc\"<br/>";
  $str2 = 'I Like \"$ruc\"<br/>';

  $html = <<<str
    <font color="#AE0B2A"> $ruc 是中国共产党创办的第一所新型正规大学，<br/>
                           是一所以人文社会科学为主的综合性研究型全国重点大学，<br/>
                           直属于教育部，由教育部与北京市共建。</font>
  str;

  echo $str1;
  echo $str2;
  echo $html;
?>
```

I Like "中国人民大学"
I Like \"$ruc\"
中国人民大学 是中国共产党创办的第一所新型正规大学，
是一所以人文社会科学为主的综合性研究型全国重点大学，
直属于教育部，由教育部与北京市共建。

字符串常用操作

数组与字符串的转换

Syntax

```
explode(separator,string,limit)
```

Parameter Values

Parameter	Description
separator	Required. Specifies where to break the string
string	Required. The string to split
limit	Optional. Specifies the number of array elements to return. Possible values: • Greater than 0 - Returns an array with a maximum of *limit* element(s) • Less than 0 - Returns an array except for the last -*limit* elements() • 0 - Returns an array with one element

Syntax

```
implode(separator,array)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies what to put between the array elements. Default is "" (an empty string)
array	Required. The array to join to a string

学习更多：
https://www.w3schools.com/php/php_ref_string.asp

子串操作

Syntax

```
substr(string,start,length)
```

Parameter Values

Parameter	Description
string	Required. Specifies the string to return a part of
start	Required. Specifies where to start in the string • A positive number - Start at a specified position in the string • A negative number - Start at a specified position from the end of the string • 0 - Start at the first character in string
length	Optional. Specifies the length of the returned string. Default is to the end of the string. • A positive number - The length to be returned from the start parameter • Negative number - The length to be returned from the end of the string • If the *length* parameter is 0, NULL, or FALSE - it return an empty string

```php
<?php
  $str = "Hello world. It's a beautiful day.";
  $arr = explode(" ", $str);
  print_r($arr);
?>
```

Array ([0] => Hello [1] => world. [2] => It's [3] => a [4] => beautiful [5] => day.)

```php
<?php
  $arr = array('Hello','World!','Beautiful','Day!');
  $str = implode(" ", $arr);
  echo $str;
?>
```

Hello World! Beautiful Day!

```php
<?php
  $str = "Hello World!";
  $substr = substr($str, 6, 5);
  echo $substr;
?>
```

World

3.1.8 数组

- 数组的类型
  - 索引数组：键名（下标）由数字组成，默认从0开始
  - 关联数组：键名是字符串，或者数字和字符串混合的形式
- 创建数组
  - 通过数组标识符“[]”创建数组
  - 使用array()函数创建数组

```php
<?php
  $arr1 = array("中国人民大学", "北京大学", "清华大学");
  $arr2[1]="中国人民大学"; $arr2[2]="北京大学"; $arr2[3]="清华大学";

  echo "打印数组键和值如下 :<br/>";
  var_dump($arr1); echo "<br/>";

  echo "打印单独某一数组元素 :<br/>";
  echo "$arr2[0]:".$arr2[0]."<br/>"; // 打印数组$arr2第0个元素
  echo "$arr2[1]:".$arr2[1]."<br/>"; // 打印数组$arr2第1个元素
  echo "$arr2[2]:".$arr2[2]."<br/>"; // 打印数组$arr2第2个元素
  echo "$arr2[3]:".$arr2[3]."<br/>"; // 打印数组$arr2第3个元素
?>
```

```
打印数组键和值如下 :
array(3) { [0]=> string(18) "中国人民大学" [1]=> string(12) "北京大学" [2]=> string(12) "清华大学" }
打印单独某一数组元素 :

Warning: Undefined array key 0 in /workspace/PHPProject/index.php on line 21
$arr2[0] :
$arr2[1] :中国人民大学
$arr2[2] :北京大学
$arr2[3] :清华大学
```

```php
<?php
  $arr3 = array("ruc"=>"中国人民大学","pku"=>"北京大学","thu"=>"清华大学");
  $arr4 = ["ruc"=>"中国人民大学", "pku"=>"北京大学", "thu"=>"清华大学"];

  echo "打印数组键和值如下 :<br/>";
  var_dump($arr3); echo "<br/>";

  echo "打印单独某一数组元素 :<br/>";
  echo '$arr4["ruc"]:'.$arr4["ruc"]."<br/>"; // 打印键值为“ruc”的元素
  echo '$arr4["pku"]:'.$arr4["pku"]."<br/>"; // 打印键值为“pku”的元素
  echo '$arr4["thu"]:'.$arr4["thu"]."<br/>"; // 打印键值为“thu”的元素
?>
```

```
打印数组键和值如下 :
array(3) { ["ruc"]=> string(18) "中国人民大学" ["pku"]=> string(12) "北京大学" ["thu"]=> string(12) "清华大学" }
打印单独某一数组元素 :
$arr4["ruc"] :中国人民大学
$arr4["pku"] :北京大学
$arr4["thu"] :清华大学
```

3.1.8 数组

- 遍历数组
  - 使用list()函数遍历数组

```php
<?php
  $arr = array("中国人民大学", "北京大学", "清华大学");
  list($ruc, $pku, $thu) = $arr;        // 将数组$arr中三个元素分别赋值给$ruc, $pku, $thu
  echo "ruc: ".$ruc."<br/>";
  echo "pku: ".$pku."<br/>";
  echo "thu: ".$thu."<br/>";
?>
```

```
ruc: 中国人民大学
pku: 北京大学
thu: 清华大学
```

  - 使用foreach循环遍历数组

```php
<?php
  $arr1 = array("中国人民大学", "北京大学", "清华大学");
  $arr3 = array("ruc"=>"中国人民大学","pku"=>"北京大学","thu"=>"清华大学");

  foreach($arr1 as $value) {              // 遍历索引数组
    echo $value."<br/>";
  }

  foreach($arr3 as $key=>$value) {        // 遍历关联数组
    echo $key.":".$value."<br/>";
  }
?>
```

```
中国人民大学
北京大学
清华大学

ruc:中国人民大学
pku:北京大学
thu:清华大学
```

3.1.8 数组

- 多维数组：包含一个或多个数组的数组
  - 对于n维数组，需要n个索引来选取元素
  - 可以用n重循环遍历n维数组的每一个元素

```php
<?php
  $sites = array
  (
    "ruc" => array("中国人民大学", "https://www.ruc.edu.cn"),
    "pku" => array("北京大学", "https://www.pku.edu.cn"),
    "thu" => array("清华大学", "https://www.tsinghua.edu.cn")
  );

  print("<pre>");
  print_r($sites);
  print("</pre>");

  echo $sites["ruc"][0].": ".$sites["ruc"][1]."<br/>";

  foreach($sites as $key=>$values) {
    echo $key.": ";
    foreach($values as $value) {
      echo $value." ";
    }
    echo "<br/>";
  }
?>
```

```
Array
(
    [ruc] => Array
        (
            [0] => 中国人民大学
            [1] => https://www.ruc.edu.cn
        )

    [pku] => Array
        (
            [0] => 北京大学
            [1] => https://www.pku.edu.cn
        )

    [thu] => Array
        (
            [0] => 清华大学
            [1] => https://www.tsinghua.edu.cn
        )

)
```

中国人民大学: https://www.ruc.edu.cn
ruc: 中国人民大学 https://www.ruc.edu.cn
pku: 北京大学 https://www.pku.edu.cn
thu: 清华大学 https://www.tsinghua.edu.cn

数组常用操作

统计数组元素个数

Syntax

```
count(array, mode)
```

Parameter Values

Parameter	Description
array	Required. Specifies the array
mode	Optional. Specifies the mode. Possible values: • 0 - Default. Does not count all elements of multidimensional arrays • 1 - Counts the array recursively (counts all the elements of multidimensional arrays)

```php
<?php
  $sites = array
  (
    "ruc" => array("中国人民大学", "https://www.ruc.edu.cn"),
    "pku" => array("北京大学", "https://www.pku.edu.cn"),
    "thu" => array("清华大学", "https://www.tsinghua.edu.cn")
  );

  echo "一维数组\$sites[\"ruc\"]元素个数: ".count($sites["ruc"])."<br/>";
  echo "二维数组\$sites中的一维数组个数: ".count($sites)."<br/>";
  echo "二维数组\$sites递归所有元素个数: ".count($sites, 1)."<br/>";
?>
```

```
一维数组$sites["ruc"]元素个数: 2
二维数组$sites中的一维数组个数: 3
二维数组$sites递归所有元素个数: 9
```

检查数组中是否存在某个值

Syntax

```
in_array(search, array, type)
```

Parameter Values

Parameter	Description
search	Required. Specifies the what to search for
array	Required. Specifies the array to search
type	Optional. If this parameter is set to TRUE, the in_array() function searches for the search-string and specific type in the array.

```php
<?php
  $postcodes = array
  (
    "ruc"=>"100872",
    "pku"=>100871,
    "thu"=> array("100084")
  );

  if (!in_array("pku", $postcodes)) echo "pku does not exist <br/>";
  if (in_array(100872, $postcodes)) echo "100872 exists (loose) <br/>";
  if (!in_array(100872, $postcodes, true))
    echo "100872 does not exist (strict) <br/>";
  if (in_array([100084], $postcodes)) echo "[100084] exists <br/>";
?>
```

```
pku does not exist
100872 exists (loose)
100872 does not exist (strict)
[100084] exists
```

学习更多:
https://www.w3schools.com/php/php_ref_array.asp

3.1.9 函数

- 参数传递方式
  - 按值传递
  - 按引用传递
  - 默认参数值
  - 变长参数
- 返回值传递方式
  - 按值传递
  - 按引用传递

/* 参数按值传递 */

```php
<?php
  function example($m) {
    $m = $m*5+10;
    echo "<p>在函数内：\$m = $m</p>";
  }

  $m = 1;
  example($m) ;
  echo "<p>在函数外：\$m = $m</p>" ;
?>
```

在函数内：$m = 15

在函数外：$m = 1

/* 参数按引用传递 */

```php
<?php
  function example(&$m) {
    $m = $m*5+10;
    echo "<p>在函数内：\$m = $m</p>";
  }

  $m = 1;
  example($m) ;
  echo "<p>在函数外：\$m = $m</p>" ;
?>
```

在函数内：$m = 15

在函数外：$m = 15

/* 默认参数值 */

```php
<?php
  function values($price, $tax=0) {
    $price = $price+($price*$tax);
    echo "<p>价格：$price</p>";
  }

  values(100, 0.25);
  values(100);
?>
```

价格：125

价格：100

/* 变长参数 */

```php
<?php
  function sum(){
    $total = 0;
    $vars = func_get_args();//参数数组
    foreach($vars as $var)
      $total += $var;
    return $total;
  }

  echo sum(1, 2)."<br/>";
  echo sum(1, 2, 3)."<br/>";
?>
```

3
6

/* 返回值传递方式 */

```php
<?php
  function &test() {
    static $b = 0;
    $b++;
    return $b;
  }
?>
```

```php
<?php
  $a = test();
  echo "$a<br/>";
  $a = 5;
  $a = test();
  echo "$a<br/>";
?>
```

1
2

```php
<?php
  $a = &test();
  echo "$a<br/>";
  $a = 15;
  $a = test();
  echo "$a<br/>";
?>
```

1
16

3.1.9 函数

- 变量作用域与生命周期
  - 局部变量：在函数内部定义的变量，作用域和生命周期是其所在函数
  - 全局变量：定义在所有函数以外的变量，作用域和生命周期是整个PHP文件，需使用global关键字声明，才可在用户自定义函数内部使用全局变量
  - 静态变量：作用域是其所在函数，生命周期是整个PHP文件

```php
<?php
  $str = "函数外部定义的变量";
  function fun() {
    echo "函数内部输出变量值：".$str."<br/>";
  }
  fun();
  echo "在函数外部输出变量值：".$str."<br/>";
?>
```

Warning: Undefined variable $str in **/workspace/PHPProject/index.php** on line **15**
函数内部输出变量值：
在函数外部输出变量值：函数外部定义的变量

```php
<?php
  $str = "函数外部定义的变量";
  function fun() {
    $str = "...函数内部定义的变量...";
    echo "函数内部输出变量值：".$str."<br/>";
  }
  fun();
  echo "在函数外部输出变量值：".$str."<br/>";
?>
```

函数内部输出变量值：...在函数内部定义的变量...
在函数外部输出变量值：函数外部定义的变量

```php
<?php
  $str = "函数外部定义的变量";
  function fun() {
    global $str;
    echo "函数内部输出变量值：".$str."<br/>";
    $str = "...在函数内部改变变量的值 ...";
    echo "改变后的变量值：".$str."<br/>";
  }
  fun();
  echo "在函数外部输出变量值：".$str."<br/>";
?>
```

函数内部输出变量值：函数外部定义的变量
改变后的变量值：...在函数内部改变变量的值 ...
在函数外部输出变量值：...在函数内部改变变量的值 ...

```php
<?php
  function fun1() {
    static $num = 0;  //初始化静态变量
    $num += 1;
    echo $num."  ";
  }
?>
```

```php
<?php
  function fun2() {
    $num = 0;  //函数内部局部变量
    $num += 1;
    echo $num."  ";
  }
?>
```

```php
<?php
  echo "fun1()中num的值：";
  for ($i = 0; $i < 9; $i++) fun1();
  echo "fun2()中num的值：";
  for ($i = 0; $i < 9; $i++) fun2();
?>
```

fun1()中num的值：1 2 3 4 5 6 7 8 9 fun2()中num的值：1 1 1 1 1 1 1 1 1

3.1.9 函数

- 文件的引用
  - include：使用include引用外部文件时，只有代码执行到include语句时，调用的外部文件才会被引用并读取；当引用的文件发生错误时，系统只给出个警告错误，而整个php文件会继续执行
  - require：在php文件被执行之前，php解析器会用被引用的文件的全部内容替换require语句形成新的php文件，最后按新的php文件执行程序；如果调用的文件没有找到，require语句会输出错误信息，立即终止执行
  - include_once/require_once：与include/require相同，但能确保一个被包含的文件只能被包含一次

```php
<?php
  include "nonexist.php";
  echo "continue execution";
?>
```

Warning: include(nonexist.php): Failed to open stream: No such file or directory in **/workspace/PHPProject/index.php** on line **12**

Warning: include(): Failed opening 'nonexist.php' for inclusion (include_path='.:/usr/local/lib/php') in **/workspace/PHPProject/index.php** on line **12**
continue execution

```php
<?php
  require "nonexist.php";
  echo "continue execution";
?>
```

Warning: require(nonexist.php): Failed to open stream: No such file or directory in **/workspace/PHPProject/index.php** on line **12**

Fatal error: Uncaught Error: Failed opening required 'nonexist.php' (include_path='.:/usr/local/lib/php') in /workspace/PHPProject/index.php:12 Stack trace: #0 {main} thrown in **/workspace/PHPProject/index.php** on line **12**

3.2 与Web页面交互

- 提交表单
- 上传文件
- 会话控制
- 其它信息

请求网址: https://www.youwei.site/training/form/submit.php?name=user&password=abcd&sex=male&education=graduate&hobby=sports&hobby=music&intro=I+am+me.&photo=head.jpg

请求方法: GET

3.2.1 提交表单

- GET方法
  - 请求数据以地址的形式表现在请求行，对传输数据的大小有限制
  - 从全局数组变量$_GET[]获得通过GET方法传递的数据

```
<form action="submit.php" method="GET" enctype="multipart/form-data">
  姓名: <input type="text" name="name"/> <br/>
  密码: <input type="password" name="password"/> <br/>
  性别:
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历:
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣:
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介: <textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片: <input type="file" name="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

```
<?php
  /* 逐个定义变量 */
  $name = $_GET["name"];
  $password = $_GET["password"];
  $sex = $_GET["sex"];
  $education = $_GET["education"];
  $hobby = $_GET["hobby"];
  $intro = $_GET["intro"];
  $photo = $_GET["photo"];

/* 输出变量值 */
  echo "name: ".$name."<br/>";
  echo "password: ".$password."<br/>";
  echo "sex: ".$sex."<br/>";
  echo "education: ".$education."<br/>";
  echo "hobby: ".$hobby."<br/>";
  echo "intro: ".$intro."<br/>";
?>
```

请求网址: https://0c660a4e-8000-app.lightly.teamcode.com/submit.php

请求方法: POST

3.2.1 提交表单

- POST方法
  - 将请求参数封装在HTTP请求数据中，对传输数据的大小无限制
  - 从全局数组变量$_POST[]获得通过POST方法传递的数据

```
<form action="submit.php" method="POST" enctype="multipart/form-data">
  姓名: <input type="text" name="name"/> <br/>
  密码: <input type="password" name="password"/> <br/>
  性别:
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历:
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣:
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介: <textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片: <input type="file" name="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

```
<?php
  /* 使用可变变量+foreach循环 */
  foreach($_POST as $key=>$value) {
    $$key = $value;
  }

  /* 输出变量值 */
  echo "name: ".$name."<br/>";
  echo "password: ".$password."<br/>";
  echo "sex: ".$sex."<br/>";
  echo "education: ".$education."<br/>";
  echo "hobby: ".$hobby."<br/>";
  echo "intro: ".$intro."<br/>";
?>
```

3.2.2 上传文件

- 全局数组变量$_FILES[]：存储上传的文件信息
  - 是一个二维数组（上传单个文件）或三维数组（上传多个文件）
  - 其一维数组的键值是<input>标签的name属性值

元 素 名	说　明
$_FILES["filename"]["name"]	存储上传文件的文件名。如text.txt、title.jpg等
$_FILES["filename"]["size"]	存储文件大小，单位为字节
$_FILES["filename"]["tmp_name"]	存储文件在临时目录中使用的文件名。因为文件在上传时，首先要将其以临时文件的身份保存在临时目录中
$_FILES["filename"]["type"]	存储上传文件的MIME类型，MIME类型规定各种文件格式的类型。每种MIME类型都是由"/"分隔的主类型和子类型组成的。例如："image/gif"，主类型为"图像"，子类型为GIF格式的文件，"text/html"代表HTML格式的文本文件
$_FILES["filename"]["error"]	存储了上传文件的结果。如果返回0，则说明文件上传成功

- 关键函数
  - is_uploaded_file(string filename)：检查是否上传过指定文件
  - move_uploaded_file(string src, string dest)：移动文件位置

上传单个文件

```
<form action="submit.php" method="POST" enctype="multipart/form-data">
  ……
  照片：<input type="file" name="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

```
<?php
  if (is_uploaded_file($_FILES["photo"]["tmp_name"]) && $_FILES["photo"]["error"] == 0) {
    echo "Upload: ".$_FILES["photo"]["name"]."<br/>";
    echo "Type: ".$_FILES["photo"]["type"]."<br/>";
    echo "Size: " .($_FILES["photo"]["size"] / 1024)." Kb<br/>";
    echo "Stored in: ".$_FILES["photo"]["tmp_name"]."<br/>";
    echo "Path: ".$_SERVER["DOCUMENT_ROOT"]."<br/>";
    $temporary_path = $_FILES["photo"]["tmp_name"];                    // 上传文件的临时存放路径
    $relative_path = "file/".$_FILES["photo"]["name"];                 // 上传文件的目标存放路径（相对路径）
    $absolute_path = $_SERVER["DOCUMENT_ROOT"]."/".$relative_path;     // 上传文件的目标存放路径（绝对路径）
    move_uploaded_file($temporary_path, $absolute_path);               // 从临时存放路径移动到目标存放路径
    echo "<img src=\"$relative_path\"/>";
  }
?>
```

上传多个文件

```
<form action="submit.php" method="POST" enctype="multipart/form-data">
  姓名：<input type="text" name="name"/> <br/>
  密码：<input type="password" name="password"/> <br/>
  性别：
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历：
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣：
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介：<textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片：<input type="file" name="photo" multiple/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

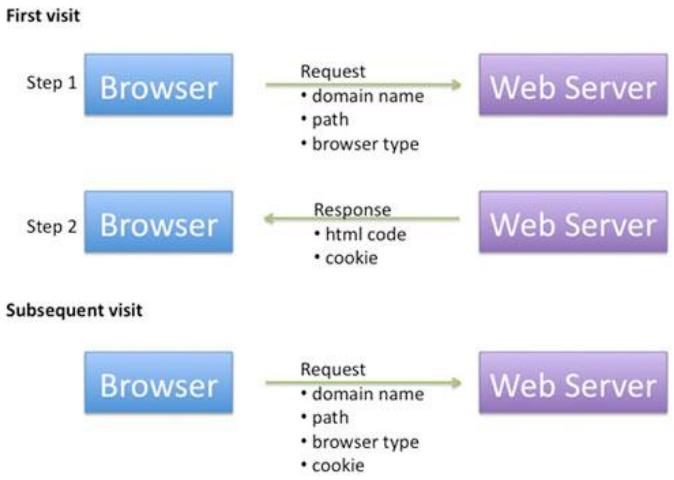
```
<?php
  /* 使用可变变量+foreach循环 */
  foreach($_POST as $key=>$value) {
    $$key = $value;
  }

  /* 输出变量值 */
  echo "name: ".$name."<br/>";
  echo "password: ".$password."<br/>";
  echo "sex: ".$sex."<br/>";
  echo "education: ".$education."<br/>";
  echo "hobby: ".$hobby."<br/>";
  echo "intro: ".$intro."<br/>";
  echo "photo: ".$photo."<br/>";
?>
```

3.2.3 会话控制

- **会话**的含义是指某个用户在网站上的有始有终的一系列动作的集合。例如，用户在访问网站时， 会话就是指从用户登入站点到到关闭浏览器所经过的这段过程
- Session是将信息保存在服务器上，并通过一个Session ID来传递客户端的信息，服务器在接收到Session ID后根据这个ID来提供相关的Session信息资源
- Cookie是将所有的信息以文本文件的形式保存在客户端，并由浏览器进行管理和维护

Cookie

- Cookie是一段存放在客户端的文本数据，由服务器端生成，发送给客户端浏览器。客户端浏览器如启用cookie，则会将这个小文本数据保存到其某个目录下的文本文件内（**永久Cookie**）或内存中（**临时Cookie**）
- 客户端下次登录同一域名下的网页，浏览器则会自动将Cookie读入之后，传给服务器端。服务器端可以对该Cookie进行读取并验证。一般情况下，Cookie中的值是以key-value的形式进行表达的
- HTTP是一种无状态协议。Cookie可认为是一种**跨页面**的数据共享机制，常被用来保存用户认证相关的信息
- 问题：什么时候需要跨页面数据共享？

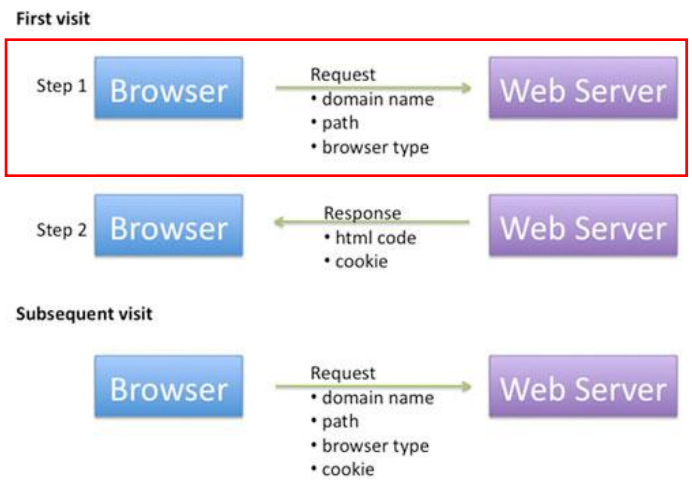


Cookie

- 常见属性

cookie属性	基本描述	举例	备注
name=value	cookie键值对	id=a3fWa	
expires	cookie过期时间	expires=Tue, 10-Jul-2013 08:30:18 GMT	
domain	指定哪些主机可以接收cookie	Domain=mozilla.org; 不设置则等于当前页面domian	定义cookie将被种植在哪些地方
path	指示哪些路径的请求会发送cookie	Path=/docs	定义cookie将会种植在哪些地方
secure	指定通过https请求发送cookie		限制后续请求访问该cookie的姿势
httponly	指示是否允许通过JavaScript Document.cookie API访问cookie		限制后续请求访问该cookie的姿势
hostonly	标记该cookie与创建cookie的页面主机名的相关性		如果host-only-flag为true时，只有当前域名与该Cookie的Domain属性完全相等，才检查path、secure、httponly等属性的匹配性
samesite	让服务器指定是否允许跨站请求发送cookie	SameSite=Lax	限制后续的跨站访问是否允许携带cookie

微人大：登录请求



▼ 常规

请求网址: https://v.ruc.edu.cn/auth/login

请求方法: POST

状态代码: 200 OK

远程地址: 10.21.1.142:443

引荐来源网址政策: strict-origin-when-cross-origin

▶ 响应标头 (13)

▼ 请求标头 查看源代码

Accept: application/json

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Connection: keep-alive

Content-Length: 505

Content-type: application/x-www-form-urlencoded

Host: v.ruc.edu.cn

Origin: https://v.ruc.edu.cn

Referer: https://v.ruc.edu.cn/auth/login?&proxy=true&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Foauth2%2Fauthorize%3Fclient_id%3Daccounts.tiup.cn%26redirect_uri%3Dhttps%253A%252F%252Fv.ruc.edu.cn%252Fsso%252Fcallback%253Fschool_code%253Druc%2526theme%253Dschools%26response_type%3Dcode%26school_code%3Druc%26scope%3Dall%26state%3DfwQ6exD-ERNF_Ep0%26theme%3Dschools&school_code=ruc

sec-ch-ua: "Google Chrome";v="105", "Not)A;Brand";v="8", "Chromium";v="105"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

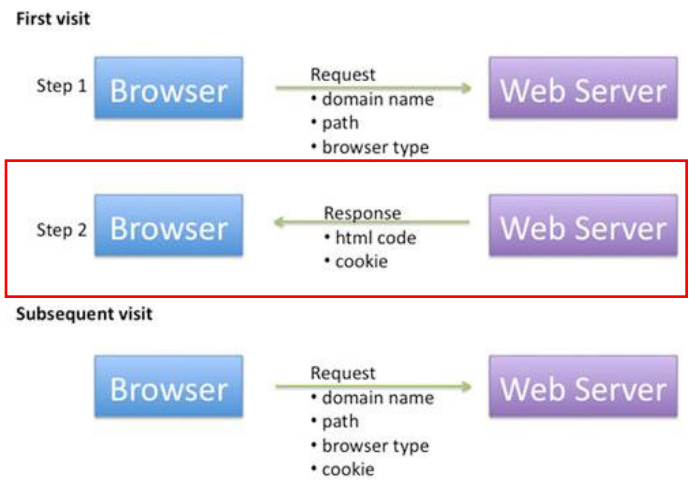
Sec-Fetch-Site: same-origin

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36

▼ 表单数据 查看源代码 查看网址编码格式的数据

{"username":"ruc[redacted]","password":"[redacted]","code":"qeew","remember_me":"false","redirect_uri":"https://v.ruc.edu.cn/oauth2/authorize?client_id=accounts.tiup.cn&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Fsso%2Fcallback%3Fschool_code%3Druc%26theme%3Dschools&response_type=code&school_code=ruc&scope=all&state=fwQ6exD-ERNF_Ep0&theme=schools","twofactor_password":"","twofactor_recovery":"","token":"kgo4glwg1r","captcha_id":"1uKgeOyhNjYEjsOyMYYY"}:

微人大：登录响应



▼ 常规

请求网址: https://v.ruc.edu.cn/auth/login

请求方法: POST

状态代码: 200 OK

远程地址: 10.21.1.142:443

引荐来源网址政策: strict-origin-when-cross-origin

▼ 响应标头　　查看源代码

Access-Control-Allow-Origin: *

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

Cache-Control: no-store

Connection: keep-alive

Content-Length: 290

Content-Type: application/json; charset=utf-8

Date: Mon, 26 Sep 2022 07:11:56 GMT

Expires: Fri, 01 Jan 1990 00:00:00 GMT

Pragma: no-cache

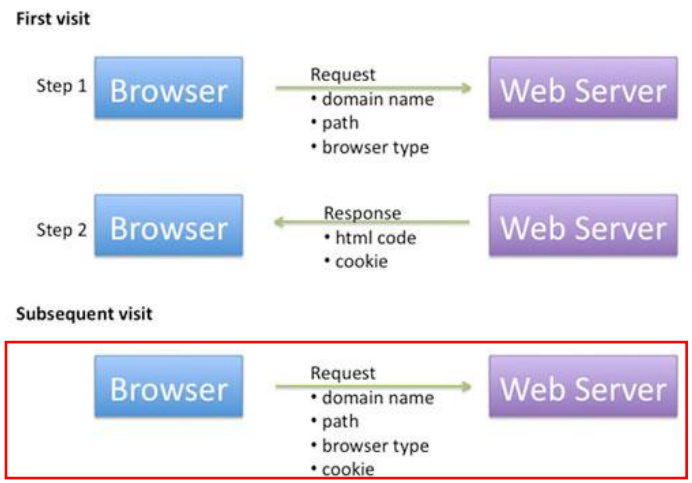
Server: none

Set-Cookie: is_simple=1; Path=/; Max-Age=3600; HttpOnly

Set-Cookie: session=6fd37[redacted]792fa5; Path=/; HttpOnly

X-Content-Type-Options: nosniff

微人大：后续请求



▼ 常规

请求网址: https://v.ruc.edu.cn/me

请求方法: GET

状态代码: 200 OK

远程地址: 10.21.1.142:443

引荐来源网址政策: strict-origin-when-cross-origin

▶ 响应标头 (12)

▼ 请求标头 查看源代码

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Connection: keep-alive

Cookie: is_simple=1; session=6fd37[redacted]792fa5; access_token=jeAlhChnRhOVkJl7J6xJlg; tiup_uid=[redacted]652

Host: v.ruc.edu.cn

Referer: https://v.ruc.edu.cn/auth/login?&proxy=true&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Foauth2%2Fauthorize%3Fclient_id%3Daccounts.tiup.cn%26redirect_uri%3Dhttps%253A%252F%252Fv.ruc.edu.cn%252Fsso%252Fcallback%253Fschool_code%253Druc%2526theme%253Dschools%26response_type%3Dcode%26school_code%3Druc%26scope%3Dall%26state%3DfwQ6exD-ERNF_Ep0%26theme%3Dschools&school_code=ruc

sec-ch-ua: "Google Chrome";v="105", "Not)A;Brand";v="8", "Chromium";v="105"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

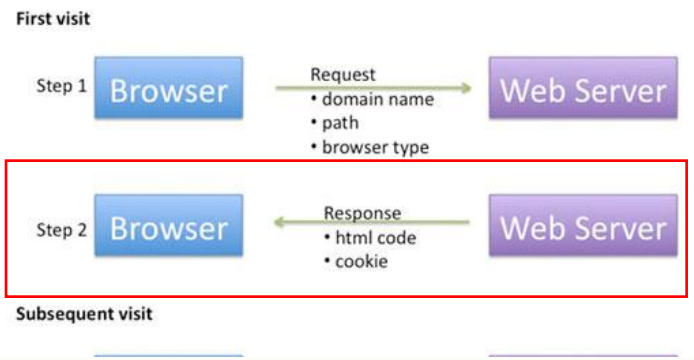
Sec-Fetch-Site: same-origin

Sec-Fetch-User: ?1

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36

服务器端操作Cookie



创建Cookie

语法

```
setcookie(name,value,expire,path,domain,secure,httponly)
```

参数	描述
name	必需。规定 cookie 的名称。
value	必需。规定 cookie 的值。
expire	可选。规定 cookie 的有效期。
path	可选。规定 cookie 的服务器路径。
domain	可选。规定 cookie 的域名。
secure	可选。规定是否通过安全的 HTTPS 连接来传输 cookie。
httponly	可选。规定是否设置http-only选项。

语法

```
header(string,replace,http_response_code)
```

参数	描述
string	必需。规定要发送的报头字符串。
replace	可选。指示该报头是否替换之前的报头，或添加第二个报头。 默认是 true（替换）。false（允许相同类型的多个报头）。
http_response_code	可选。把 HTTP 响应代码强制为指定的值。（PHP 4 以及更高版本可用）

```php
<?php
  header("Set-Cookie: is_simple=1; Path=/; Max-Age=3600; HttpOnly");
  setcookie("session","6fd37[redacted]792fa5", 0, "/", "", false, true);
?>
<html>
</html>
```

Set-Cookie: is_simple=1; Path=/; Max-Age=3600; HttpOnly

Set-Cookie: session=6fd373[redacted]792fa5; path=/; HttpOnly

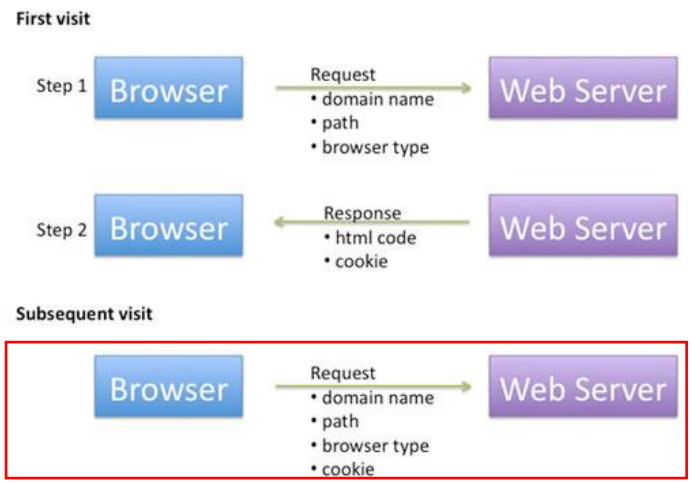
注意：setcookie()和header()是通过HTTP请求响应的Header来完成的，需要在请求响应内容输出之前执行。

删除Cookie

```php
<?php
  setcookie("session", "", 0);  // 设置value为空或者expire为0，即可清空Cookie
?>
```

服务器端操作Cookie

- 获取Cookie：全局数组变量$_COOKIE[]




```php
<?php
  setcookie("session","6fd37                                    792fa5", 0, "/", "", false, true);
  if (isset($_COOKIE["session"])) echo "名为session的Cookie项目的值为".$_COOKIE["session"];
  else echo "名为session的Cookie项目不存在";
?>
```

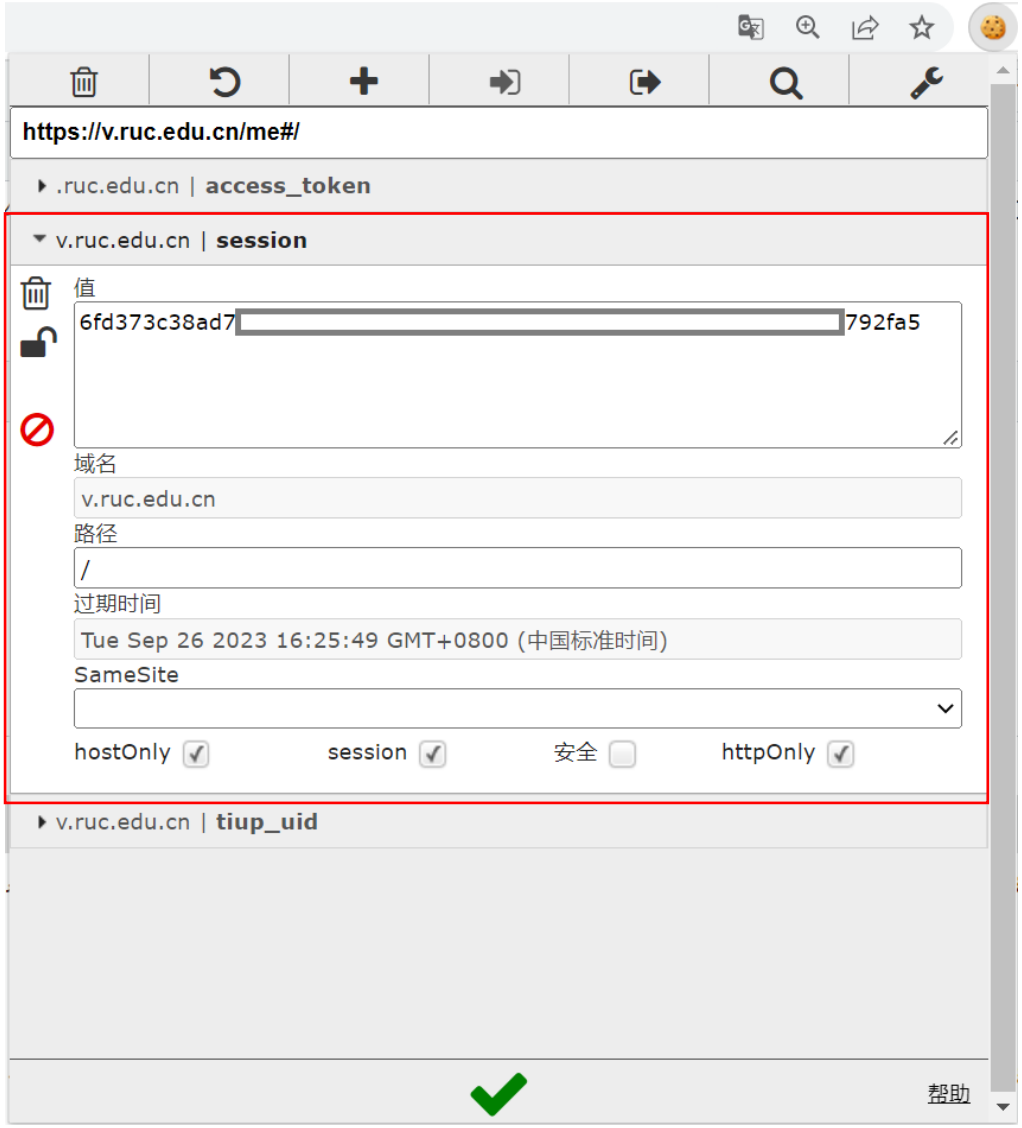
第一次访问： 名为session的Cookie项目不存在

第二次访问： 名为session的Cookie项目的值为6fd37   792fa5

注意：

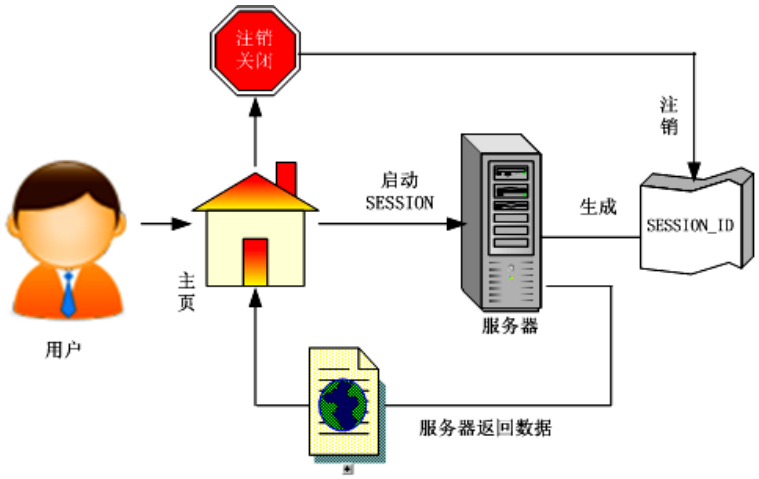
- 通过setcookie()函数创建Cookie后，在当前页应用echo $_COOKIE["name"]不会有任何输出
- 在刷新后或者到达下一个页面时才可以看到Cookie值
- 因为setcookie()函数执行后，会向客户端发送一个Cookie，若不刷新或者浏览下一个页面，客户端就不能将Cookie送回

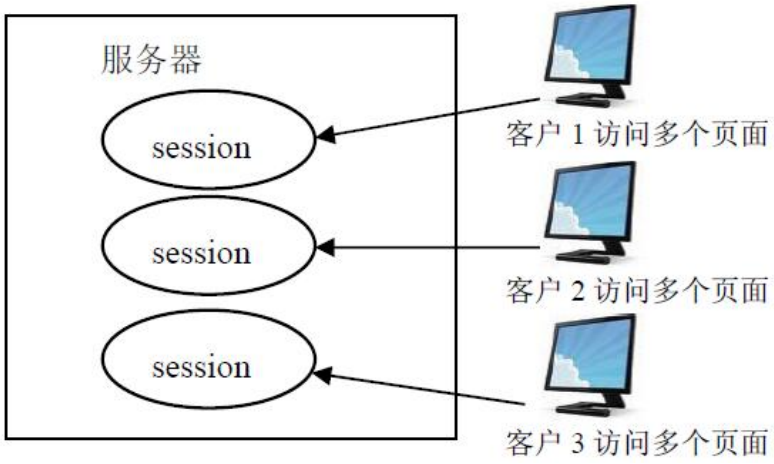
客户端操作Cookie：EditThisCookie扩展



Session

- Session中的数据可以被同一个客户在网站的一次会话过程共享。但是对于不同客户来说，每个人的Session是不同的
- Session可认为是一种保存在服务器端的跨页面（跨请求）的数据共享机制
- 当用户结束会话时（如关闭浏览器），服务端的Session并不会被立即删除（若非主动告知，服务端并不会知道用户结束了会话）。除非程序通知服务器删除一个Session，否则服务器会一直保留这个Session对象，直到其超时失效，被垃圾收集机制收集掉



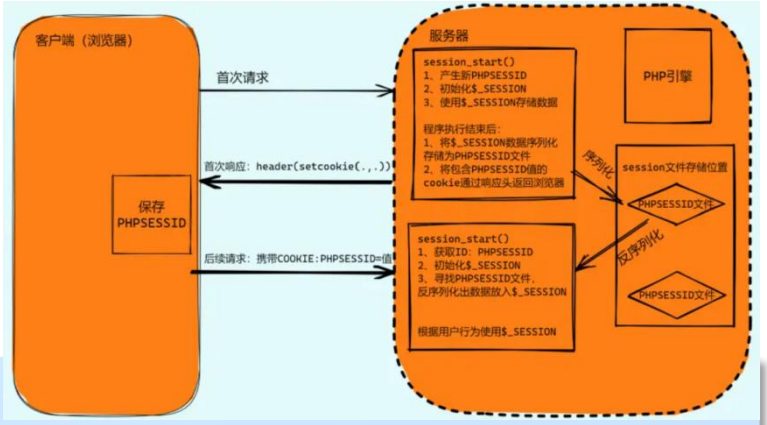


Session

- 当程序需要为某个客户端的请求创建一个Session的时候，服务器可以首先检查这个客户端的请求里是否已包含了一个Session标识：称为Session ID（通常为一个随机的长字符串）
  - 如果已包含一个Session ID则说明以前已经为此客户端创建过Session，服务器就按照Session ID把这个 Session检索出来使用
  - 如果客户端请求不包含Session ID，则为此客户端创建一个Session并且生成 一个与此Session相关联的Session ID， Session ID的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串，这个Session ID可在响应中返回给客户端保存
- 一般情况下， **Session ID被保存在客户端的Cookie中**（用户登录成功后，将用户认证成功的信息存放在一个Session中，并将此Session ID设置存放在客户端Cookie中）

Session

- 开启Session：session_start()
- 终结Session：session_destroy()
- 获得Session ID：session_id()
- 存储/获取Session变量：全局数组变量$_SESSION[]



注意：

- 如果客户端发送给服务器的Cookie中包含名为“PHPSESSID”的项，session_start()会使用$_COOKIE["PHPSESSID"]对应的session
- 客户端发送给服务器的Cookie中不包含名为“PHPSESSID”的项，session_start()会创建一个名为“PHPSESSION”的Cookie项发送给客户端
- session_destroy()只销毁服务器端session，不会销毁客户端名为“PHPSESSID”的Cookie项

```php
<?php
  session_start();                    // 开启Session

  $username = $_POST["username"];     // 获取表单中的用户名
  $password = $_POST["password"];     // 获取表单中的密码

  if ($username == "admin001" && $password == "admin001")
    $_SESSION["role"] = "admin";                          // 管理员身份
  else if ($username == "teacher001" && $password == "teacher001")
    $_SESSION["role"] = "teacher";                        // 存储Session变量
  else if ($username == "student001" && $password == "student001")
    $_SESSION["role"] = "student";                        // 教师身份
  else session_destroy();                                 // 存储Session变量
                                                          // 学生身份
  echo "Session ID: ".session_id()."<br/>";     // 获得Session ID   // 存储Session变量
  echo "Role: ".$_SESSION["role"]."<br/>";      // 获取Session变量  // 终结Session
?>
```

Session ID: 028c70308f8682092859519569ad4787
Role: admin

实例：微人大认证页面会话控制


```php
<?php
  /* pannel.php: 面板页面 */
  session_start();
  if (isset($_SESSION["username"])) {
    $role = $_COOKIE["role"]; //安全问题?
    echo "你好，".$role."<br/>";
    echo "<a href='logout.php'>退出</a>";
  } else echo "无权访问！";
?>
```

```php
<?php
  /* logout.php: 退出页面 */
  session_start();    // 启用Session
  session_unset();    // 销毁$_SESSION数组
  session_destroy(); // 销毁Session
  header("Location: index.php");
?>
```

```php
<?php
  /* index.php: 首页 */
  session_start(); // 启用Session
  if (isset($_SESSION["username"])) { // 判断是否处于成功登录状态
    header("Location: pannel.php"); // 跳转到面板页面
    exit(); // 终止后续代码的执行
  }
?>
<form action="login.php" method="POST" id="login-form">
<?php $_SESSION["code"] = rand()%10; ?>
  <img src="img/code_<?=$_SESSION["code"]?>.png">
</form>
```

```php
<?php
  /* login.php: 登录页面 */
  session_start();                          // 启用Session
  $username = $_POST["username"];           // 获取表单中的用户名
  $password = $_POST["password"];           // 获取表单中的密码
  $code = $_POST["code"];                   // 获取表单中的验证码

  //维护一个验证码正确值和验证码图片下标的映射
  $checks = ["pupr", "khrb", "huks", "egwb", "ekdv", "khns", "wuxc", "tcyk", "nupf", "brpk"];
  $correct_code = $checks[$_SESSION["code"]];

  if ($code == $correct_code) {
    if ($username == "admin001" && $password == "admin001")                 // 管理员
      $_SESSION["role"] = "admin";                                          // 在Session中记录角色
    else if ($username == "teacher001" && $password == "teacher001")        // 教师
      $_SESSION["role"] = "teacher";                                        // 在Session中记录角色
    else if ($username == "student001" && $password == "student001")        // 学生
      $_SESSION["role"] = "student";                                        // 在Session中记录角色

    if (isset($_SESSION["role"])) {                                         // 登录成功
      $_SESSION["username"] = $username;                                    // 在Session中记录用户名
      setcookie("role",$_SESSION["role"], 0, "/", "", false, true);         // 设置名为“role”的Cookie项
      header("Location: pannel.php");                                       // 跳转到用户面板
    } else echo "用户名或密码不正确！";                                     // 错误提示
  } else echo "验证码不正确！";                                             // 错误提示
?>
```

3.2.4 其它信息

- 全局数组变量$_ENV[]：PHP的环境变量信息
- 全局数组变量$_GLOBAL[]：引用全局作用域中可用的全部变量
- 全局数组变量$_SERVER[]：与PHP服务器相关的信息
- 全局数组变量$_REQUEST[]：获取GET方法、POST方法和通过Cookie传到服务器的信息，是$_GET[]、$_POST[]和$_COOKIE[]的汇总

3.2.4 其它信息

- 全局数组变量$_SERVER[]：与PHP服务器相关的信息

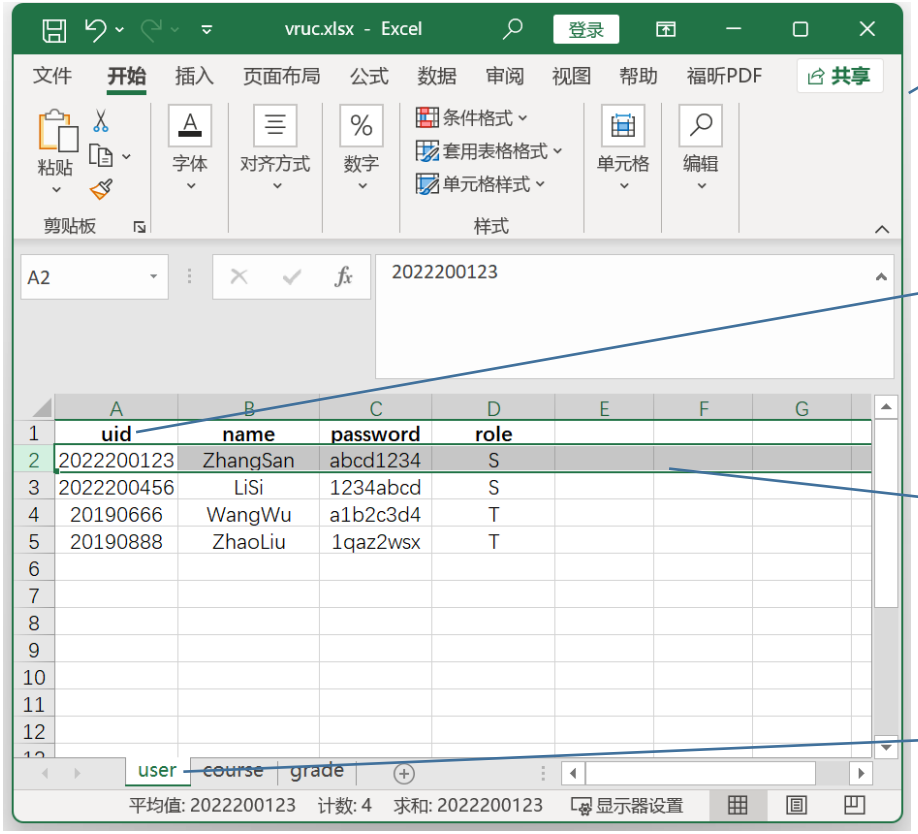
数组元素	说明
$_SERVER['PHP_SELF']	当前执行脚本的文件名，与document root有关。例如，在地址为http://c.biancheng.net/test.php/foo.bar的脚本中使用 $_SERVER['PHP_SELF'] 将得到 /test.php/foo.bar
$_SERVER['SERVER_ADDR']	当前运行脚本所在服务器的 IP 地址
$_SERVER['SERVER_NAME']	当前运行脚本所在服务器的主机名。如果脚本运行于虚拟主机中，该名称就由那个虚拟主机所设置的值决定
$_SERVER['SERVER_PROTOCOL']	请求页面时通信协议的名称和版本。例如，“HTTP/1.0”
$_SERVER['REQUEST_METHOD']	访问页面使用的请求方法。例如“GET”“HEAD”“POST”“PUT”
$_SERVER['DOCUMENT_ROOT']	当前运行脚本所在的文档根目录。在服务器配置文件中定义
$_SERVER['HTTP_ACCEPT_LANGUAGE']	当前请求头中 Accept-Language: 项的内容（如果存在）。例如，“en”
$_SERVER['REMOVE_ADDR']	浏览当前页面的用户 IP 地址，注意与 $_SERVER['SERVER_ADDR'] 的区别
$_SERVER['SCRIPT_FILENAME']	当前执行脚本的绝对路径
$_SERVER['SCRIPT_NAME']	包含当前脚本的路径
$_SERVER['REQUEST_URI']	URI 用来指定要访问的页面。例如，“index.html”
$_SERVER['PATH_INFO']	包含由客户端提供的、跟在真实脚本名称之后并且在查询语句（query string）之前的路径信息（如果存在）。例如，当前脚本是通过 URL http://c.biancheng.net/php/path_info.php/some/stuff?foo=bar 被访问的，那么 $_SERVER['PATH_INFO'] 将包含 /some/stuff

```php
<?php
  echo $_SERVER["SERVER_NAME"]."<br/>";
  echo $_SERVER["SERVER_PROTOCOL"]."<br/>";
  echo $_SERVER["REQUEST_METHOD"]."<br/>";
  echo $_SERVER["REQUEST_URI"]."<br/>";
  echo $_SERVER["DOCUMENT_ROOT"]."<br/>";
  echo $_SERVER["SCRIPT_FILENAME"]."<br/>";
?>
```

```
0.0.0.0
HTTP/1.1
GET
/?port=8000&dcsId=b8c0a052&token=Q_OuBnzfQaa4Hb_odoc3yQ
/workspace/PHPProject
/workspace/PHPProject/index.php
```

3.3 SQL

- SQL（Structured Query Language）结构化查询语言，关系数据库事实上的标准操作语言



工作簿 ↔ 数据库

列 ↔ 字段

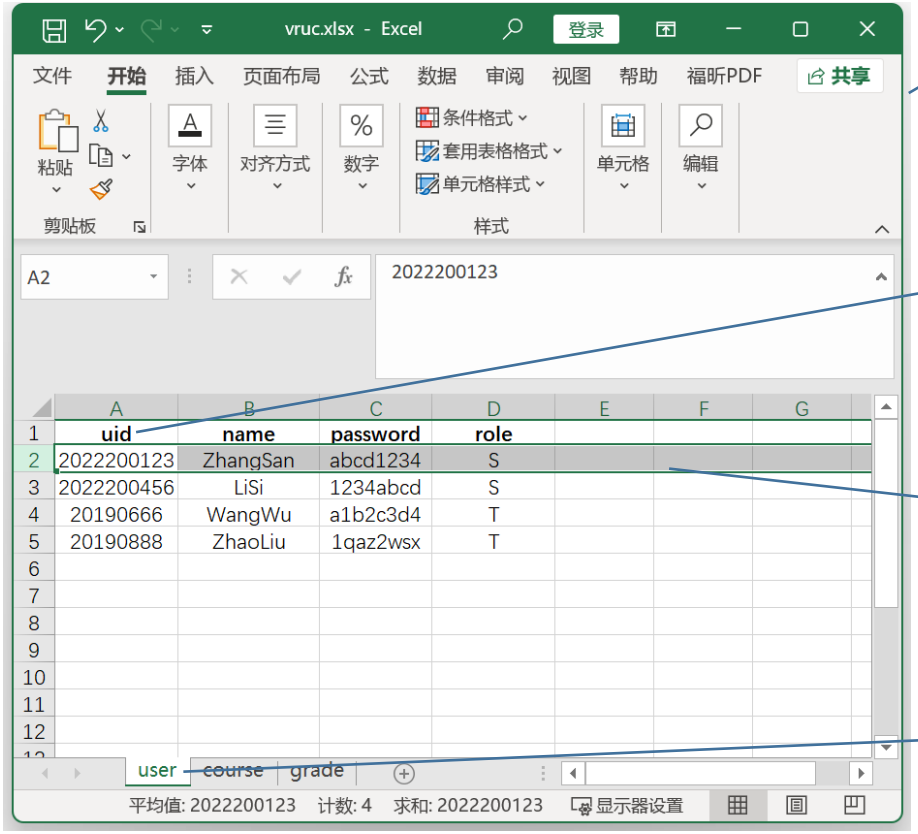
行 ↔ 记录

工作表 ↔ 表

学习更多：https://www.w3school.com.cn/sql/index.asp

3.3 SQL

- SQL（Structured Query Language）结构化查询语言，关系数据库事实上的标准操作语言



工作簿 ↔ 数据库

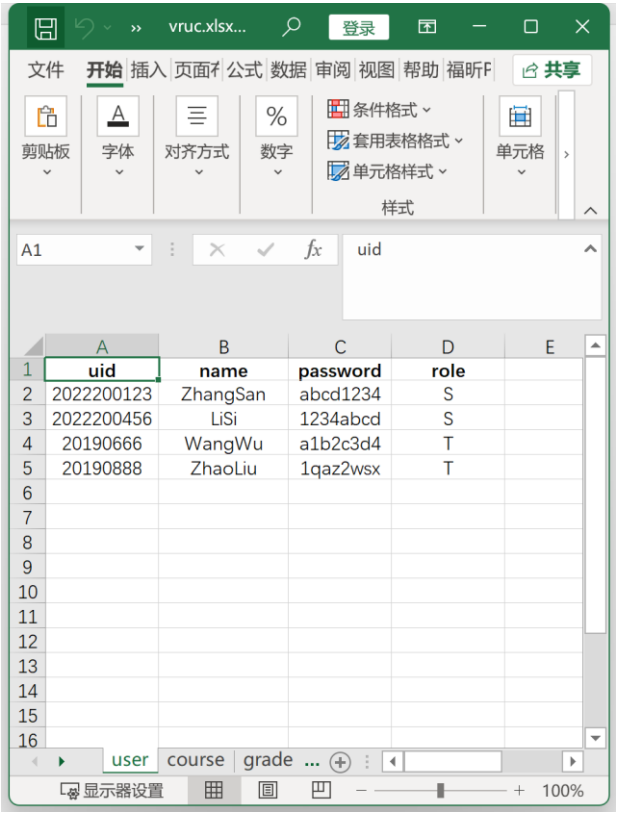
列 ↔ 字段

行 ↔ 记录

工作表 ↔ 表

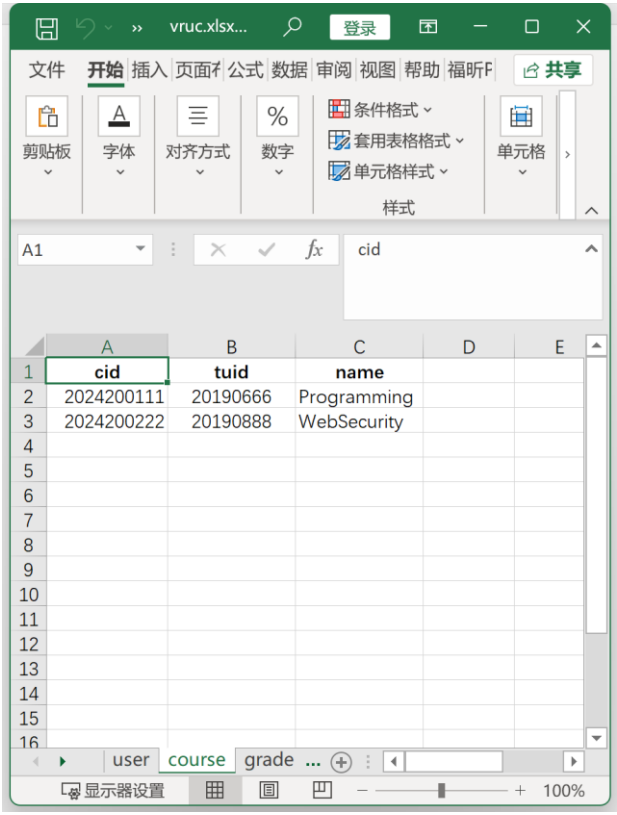
学习更多：https://www.w3school.com.cn/sql/index.asp

3.3 SQL



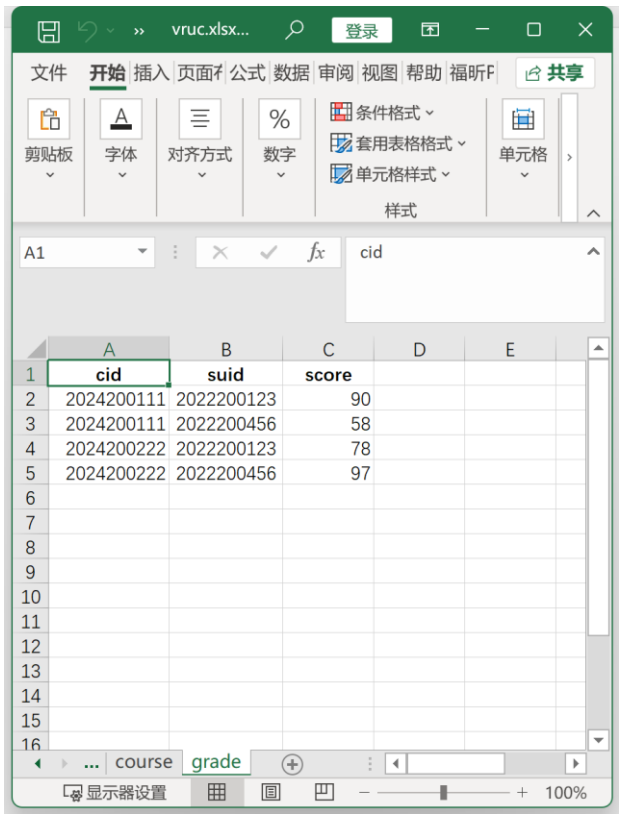
	A	B	C	D	E
1	uid	name	password	role	
2	2022200123	ZhangSan	abcd1234	S	
3	2022200456	LiSi	1234abcd	S	
4	20190666	WangWu	a1b2c3d4	T	
5	20190888	ZhaoLiu	1qaz2wsx	T	

```
MariaDB [test]> select * from user;
+------------+----------+----------+------+
| uid        | name     | password | role |
+------------+----------+----------+------+
| 2022200123 | ZhangSan | abcd1234 | S    |
| 2022200456 | LiSi     | 1234abcd | S    |
|   20190666 | WangWu   | a1b2c3d4 | T    |
|   20190888 | ZhaoLiu  | 1qaz2wsx | T    |
+------------+----------+----------+------+
4 rows in set (0.000 sec)
```



	A	B	C	D	E
1	cid	tuid	name		
2	2024200111	20190666	Programming		
3	2024200222	20190888	WebSecurity		

```
MariaDB [test]> select * from course;
+------------+----------+-------------+
| cid        | tuid     | name        |
+------------+----------+-------------+
| 2024200111 | 20190666 | Programming |
| 2024200222 | 20190888 | WebSecurity |
+------------+----------+-------------+
2 rows in set (0.001 sec)
```



	A	B	C	D	E
1	cid	suid	score		
2	2024200111	2022200123	90		
3	2024200111	2022200456	58		
4	2024200222	2022200123	78		
5	2024200222	2022200456	97		

```
MariaDB [test]> select * from grade;
+------------+------------+-------+
| cid        | suid       | score |
+------------+------------+-------+
| 2024200111 | 2022200123 |    90 |
| 2024200111 | 2022200456 |    58 |
| 2024200222 | 2022200123 |    78 |
| 2024200222 | 2022200456 |    97 |
+------------+------------+-------+
4 rows in set (0.001 sec)
```

3.3 SQL

- 数据表创建和删除操作
- 数据查询操作
- 数据增加操作
- 数据删除操作
- 数据更新操作

3.3.1 数据表创建和删除操作

- 语法：

  CREATE TABLE 表名 (列名1 数据类型1, 列名2 数据类型2, ...)

  DROP TABLE 表名

- 示例：

```
1 CREATE TABLE user (
2   uid int,
3   name varchar(256),
4   password varchar(256),
5   role char
6 );
```

```
MariaDB [test]> describe user
    -> ;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| uid      | int(11)      | YES  |     | NULL    |       |
| name     | varchar(256) | YES  |     | NULL    |       |
| password | varchar(256) | YES  |     | NULL    |       |
| role     | char(1)      | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.002 sec)
```

```
1 DROP TABLE user;
```

3.3.2 数据查询操作

- 语法：

```
1 SELECT 列名 FROM 表名
2 WHERE 条件表达式
3 GROUP BY 列名
4 HAVING 条件表达式
5 ORDER BY 字段  ASC|DESC
6 LIMIT m, n
```

- 示例：

```
1 SELECT uid, name FROM user;
```

```
MariaDB [test]> SELECT uid, name FROM user;
+------------+----------+
| uid        | name     |
+------------+----------+
| 2022200123 | ZhangSan |
| 2022200456 | LiSi     |
|   20190666 | WangWu   |
|   20190888 | ZhaoLiu  |
+------------+----------+
4 rows in set (0.001 sec)
```

```
1 SELECT * FROM course WHERE tuid=20190666;
```

```
MariaDB [test]> SELECT * FROM course WHERE tuid=20190666;
+------------+----------+-------------+
| cid        | tuid     | name        |
+------------+----------+-------------+
| 2024200111 | 20190666 | Programming |
+------------+----------+-------------+
1 row in set (0.000 sec)
```

3.3.2 数据查询操作

- 示例

```sql
SELECT name, score FROM user, grade
WHERE user.uid = grade.suid
ORDER BY name ASC, score DESC
LIMIT 1, 2;
```

```
MariaDB [test]> SELECT name, score FROM user, grade WHERE user.uid = grade.suid ORDER BY name ASC, score DESC LIMIT 1, 2;
+----------+-------+
| name     | score |
+----------+-------+
| LiSi     |    58 |
| ZhangSan |    90 |
+----------+-------+
2 rows in set (0.001 sec)
```

```sql
SELECT cid, avg(score) FROM grade GROUP BY cid;
```

```
MariaDB [test]> SELECT cid, avg(score) FROM grade GROUP BY cid;
+------------+------------+
| cid        | avg(score) |
+------------+------------+
| 2024200111 |    74.0000 |
| 2024200222 |    87.5000 |
+------------+------------+
2 rows in set (0.001 sec)
```

```sql
SELECT suid, avg(score) FROM grade
GROUP BY suid
HAVING avg(score)≥80;
```

```
MariaDB [test]> SELECT suid, avg(score) FROM grade GROUP BY suid HAVING avg(score)≥80;
+------------+------------+
| suid       | avg(score) |
+------------+------------+
| 2022200123 |    84.0000 |
+------------+------------+
1 row in set (0.001 sec)
```

3.3.3 数据增加操作

- 语法：

```
1 INSERT INTO 表名 (列名) VALUES (列值)
```

- 示例：
  - INSERT INTO user VALUES (2022200123, "ZhangSan", "abcd1234", 'S');

  - INSERT INTO user (uid, name) VALUES (2022200456, "LiSi");

  - INSERT INTO user (uid, name) VALUES (20190666, "WangWu"), (20190888, "ZhaoLiu");

3.3.4 数据删除操作

- 语法：

```
1 DELETE FROM 表名 WHERE 条件表达式
```

- 示例：
  - DELETE FROM user WHERE uid=2022200456;

```
MariaDB [test1]> select * from user;
+------------+-----------+----------+------+
| uid        | name      | password | role |
+------------+-----------+----------+------+
| 2022200123 | ZhangSan  | abcd1234 | S    |
| 2022200456 | LiSi      | NULL     | NULL |
|   20190666 | WangWu    | NULL     | NULL |
|   20190888 | ZhaoLiu   | NULL     | NULL |
+------------+-----------+----------+------+
4 rows in set (0.000 sec)
```

```
MariaDB [test1]> select * from user;
+------------+-----------+----------+------+
| uid        | name      | password | role |
+------------+-----------+----------+------+
| 2022200123 | ZhangSan  | abcd1234 | S    |
|   20190666 | WangWu    | NULL     | NULL |
|   20190888 | ZhaoLiu   | NULL     | NULL |
+------------+-----------+----------+------+
3 rows in set (0.000 sec)
```

  - DELETE FROM user;

```
MariaDB [test1]> select * from user;
+------------+-----------+----------+------+
| uid        | name      | password | role |
+------------+-----------+----------+------+
| 2022200123 | ZhangSan  | abcd1234 | S    |
|   20190666 | WangWu    | NULL     | NULL |
|   20190888 | ZhaoLiu   | NULL     | NULL |
+------------+-----------+----------+------+
3 rows in set (0.000 sec)
```

```
MariaDB [test1]> DELETE FROM user;
Query OK, 3 rows affected (0.003 sec)

MariaDB [test1]> select * from user;
Empty set (0.001 sec)
```

3.3.5 数据更新操作

- 语法：

```
1 UPDATE 表名 SET 列名 = 列值 WHERE 条件表达式
```

- 示例：
  - UPDATE user SET name="ZhangSan*" WHERE uid=2022200123;

```
MariaDB [test1]> select * from user;
+------------+----------+----------+------+
| uid        | name     | password | role |
+------------+----------+----------+------+
| 2022200123 | ZhangSan | abcd1234 | S    |
| 2022200456 | LiSi     | NULL     | NULL |
|   20190666 | WangWu   | NULL     | NULL |
|   20190888 | ZhaoLiu  | NULL     | NULL |
+------------+----------+----------+------+
4 rows in set (0.000 sec)
```

```
MariaDB [test1]> select * from user;
+------------+-----------+----------+------+
| uid        | name      | password | role |
+------------+-----------+----------+------+
| 2022200123 | ZhangSan* | abcd1234 | S    |
| 2022200456 | LiSi      | NULL     | NULL |
|   20190666 | WangWu    | NULL     | NULL |
|   20190888 | ZhaoLiu   | NULL     | NULL |
+------------+-----------+----------+------+
4 rows in set (0.000 sec)
```

  - UPDATE user SET role='S';

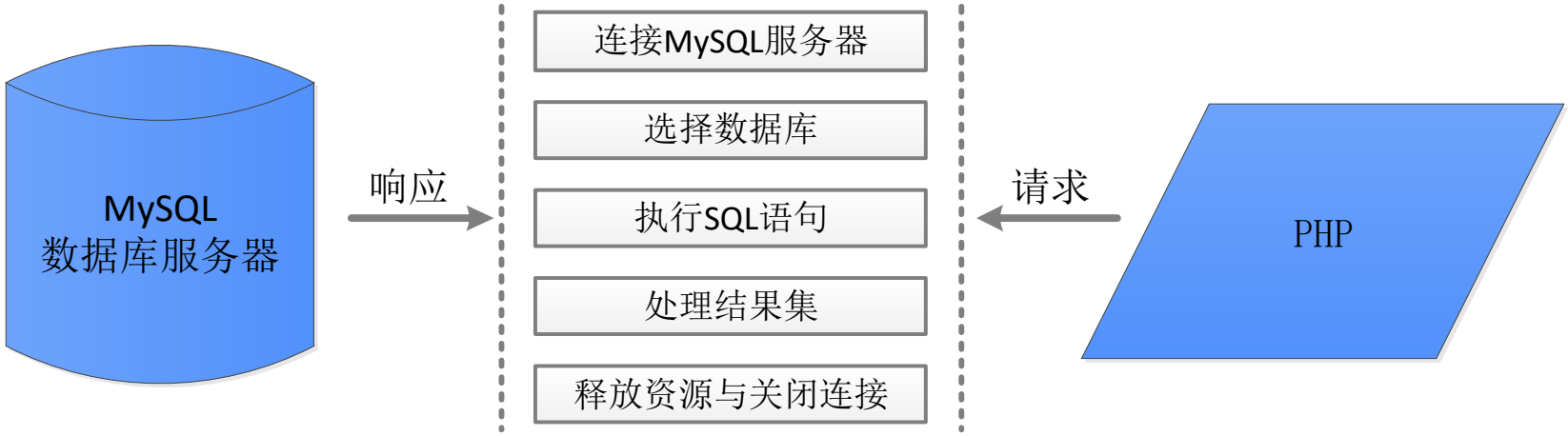
```
MariaDB [test1]> select * from user;
+------------+-----------+----------+------+
| uid        | name      | password | role |
+------------+-----------+----------+------+
| 2022200123 | ZhangSan* | abcd1234 | S    |
| 2022200456 | LiSi      | NULL     | NULL |
|   20190666 | WangWu    | NULL     | NULL |
|   20190888 | ZhaoLiu   | NULL     | NULL |
+------------+-----------+----------+------+
4 rows in set (0.000 sec)
```

```
MariaDB [test1]> select * from user;
+------------+-----------+----------+------+
| uid        | name      | password | role |
+------------+-----------+----------+------+
| 2022200123 | ZhangSan* | abcd1234 | S    |
| 2022200456 | LiSi      | NULL     | S    |
|   20190666 | WangWu    | NULL     | S    |
|   20190888 | ZhaoLiu   | NULL     | S    |
+------------+-----------+----------+------+
4 rows in set (0.000 sec)
```

目录

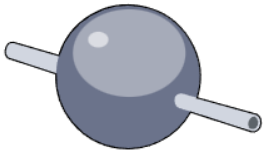
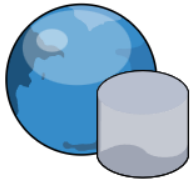
3.4 数据库操作

- PHP提供了大量的MySQL数据库操作函数，可以方便地实现访问MySQL数据库的各种需要，从而轻松完成Web应用程序开发



数据库操作

PHP、终端与数据库连接对比图

 PHP环境	 数据库	 终端
`$mysqli = new mysqli( $servername, $username, $passwd, $database );`	连接	sudo mysql
	选择数据库	use databaseName;
`$res = $mysqli->query($sql);`	执行sql语句	sqlSentence;
`$row = $res->fetch_assoc();`	处理结果集	结果直接显示在终端上
`$res->free();`	释放资源	exit
`$mysqli->close();`	关闭连接	

3.4.1 连接MySQL数据库

在操作MySQL数据库之前，需先与MySQL数据库服务器建立连接。我们使用面向对象的方式

参数	参数解释
host	Optional. 确定连接域名或IP地址
username	Optional. 确定连接数据库用户名
password	Optional. 上述用户对应的密码
dbname	Optional. 确定连接哪个数据库
port	Optional. 确定连接host的哪个端口，默认3306
socket	Optional. 确定使用哪个socket或pipe

```php
$mysqli =
new mysqli(
  $servername,
  $username,
  $passwd,
  $database
);
```

```php
<?php
  $mysqli = new mysqli("localhost","my_user","my_password","my_db");
  if ($mysqli -> connect_errno) {
    echo "Failed to connect to MySQL: " . $mysqli -> connect_error;
    exit();
  }
```

3.4.2 执行Sql语句

- 我们首先将需要执行的sql语句保存到变量中->$sql

```
$res = $mysqli->query($sql);
```

参数	参数解释
connection	Required. 确定用哪个连接执行sql语句，对应上图$conn
query	Required. 确定执行的sql语句，对应上图$sql
resultmode	Optional. 常量，下面两个二选一 •MYSQLI_USE_RESULT (用于获取超大量数据) •MYSQLI_STORE_RESULT (默认模式)

```
1 // Perform query
2 if ($result = $mysqli → query("SELECT * FROM Persons")) {
3   echo "Returned rows are: " . $result → num_rows;
4   // Free result set
5   $result → free_result();
6 }
```

3.4.3 处理结果集

- 处理结果集有多种形式，形式为

```
1 $mysqli_result → fetch_someMode()
```

函数	作用	返回格式	示例
fetch_all	一次性获取结果集中的所有行。	返回一个二维数组，每一行是一个子数组。	
fetch_array	获取结果集中的下一行。	返回一个数组，包含关联数组和索引数组两种形式。	
fetch_assoc	获取结果集中的下一行。	返回一个关联数组。	
fetch_object	获取结果集中的下一行。	返回一个对象，字段名作为属性。	

```
1 $data = $result→fetch_all(MYSQLI_ASSOC);
2 print_r($data);
```

```
Array
(
    [0] => Array
        (
            [id] => 1
            [name] => John
        )
    [1] => Array
        (
            [id] => 2
            [name] => Jane
        )
)
```

```
1 while ($row = $result→fetch_array(MYSQLI_ASSOC))
2 { echo $row['id'] . " - " . $row['name'] . "<br>"; }
```

```
1 - John
2 - Jane
```

```
1 while ($row = $result→fetch_assoc())
2 { echo $row['id'] . " - " . $row['name'] . "<br>"; }
```

```
1 - John
2 - Jane
```

```
1 while ($row = $result→fetch_object())
2 { echo $row→id . " - " . $row→name . "<br>"; }
```

```
1 - John
2 - Jane
```

3.4.3 处理结果集

到了这里，我们已经能够从数据库中查询到结果了。

```php
<?php
  $mysqli = new mysqli("localhost","my_user","my_password","my_db");
  if ($mysqli -> connect_errno) {
    echo "Failed to connect to MySQL: " . $mysqli -> connect_error;
    exit();
  }
  $sql = "SELECT Lastname, Age FROM Persons ORDER BY Lastname";
  $result = $mysqli -> query($sql);
  // Fetch all
  $result -> fetch_all(MYSQLI_ASSOC);
```

3.4.4 释放资源和关闭连接

- 使用面向对象的方式时，由于PHP本身优秀的垃圾回收机制，你并不需要手动释放结果集，如果你想要手动释放，可以调用结果集的free()函数

```
$res->free();
```

- 当你的数据库服务器的吞吐量较大时，你也许需要关闭不需要的连接来减轻数据库服务器的负担，调用连接的close()函数

```
$mysqli->close();
```

实例：微人大认证页面数据库查询

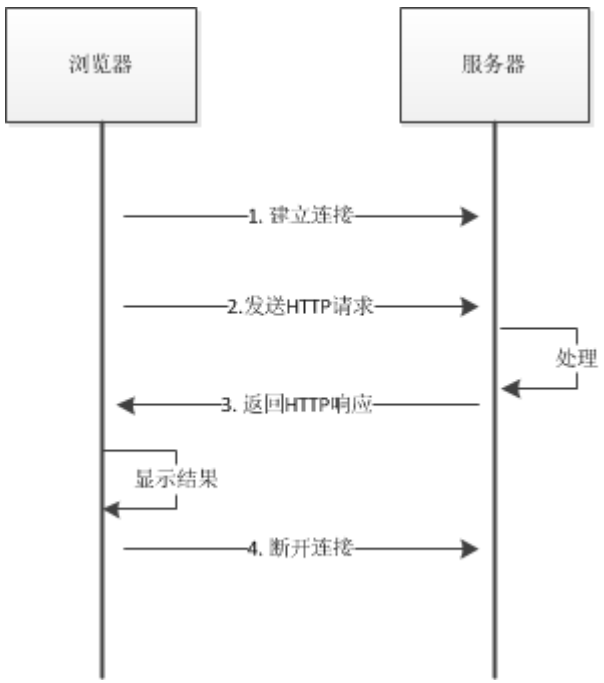
```php
<?php
session_start();
$servername = "localhost";
$username = "clever";
$passwd = "aStrongPasswd";
$database = "MyWall";

$mysqli = new mysqli($servername, $username, $passwd, $database);
if ($mysqli->connect_error) {
    die("连接失败: " . $conn->connect_error);
}

foreach($_POST as $key=>$val) {
    $$key = $val;
}

$captcha = [0=>"pupr", 1=>"khrb", 2=>"huks",
            3=>"egwb", 4=>"ekdv", 5=>"khns",
            6=>"wuxc", 7=>"tcyk", 8=>"nupf",
            9=>"brpk"];

if ($code === $captcha[$id]) {
    $sql = "SELECT * FROM user where name='$name' and passwd='$passwd'";
    $res = $mysqli->query($sql);

    if ($res->num_rows > 0) {
        $row = $res->fetch_assoc();
        $_SESSION["role"] = $row["role"];
        $_SESSION["name"] = $row["name"];
    } else {
        echo "<script>alert('wrong username and password!');location.href='../index.php'</script>";
        die();
    }
    $res->free_result();
    $conn->close();
    echo "<script>location.href='pannel.php'</script>";
} else {
    echo "<script>location.href='../index.php'</script>";
    die();
}

```

3.5 前后端通信

- HTTP协议与消息
- AJAX

3.5.1 HTTP协议与消息

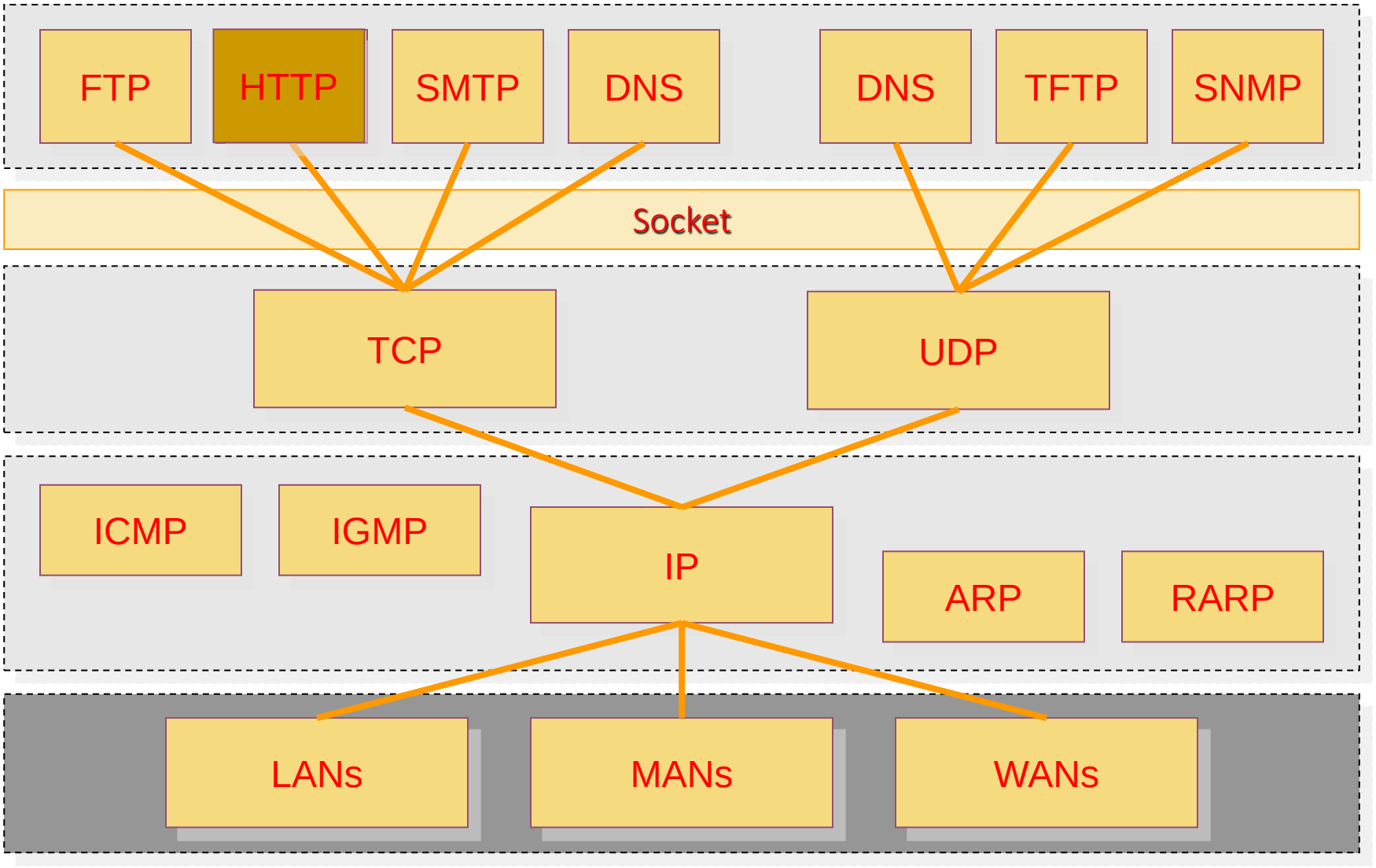
- HTTP：超文本传输协议（HyperText Transfer Protocol），基于TCP/IP之上的应用协议



HTTP协议通讯过程

1. 客户机与服务器建立连接。只要单击某个超级链接，HTTP 的工作开始。
2. 客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是MIME信息包括请求修饰符、客户机信息和可能的内容。
3. 服务器接到请求后，处理并返回响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是MIME 信息包括服务器信息、实体信息和可能的内容。
4. 客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

HTTP协议


FTP
HTTP
SMTP
DNS
DNS
TFTP
SNMP
Socket
TCP
UDP
ICMP
IGMP
IP
ARP
RARP
LANs
MANs
WANs

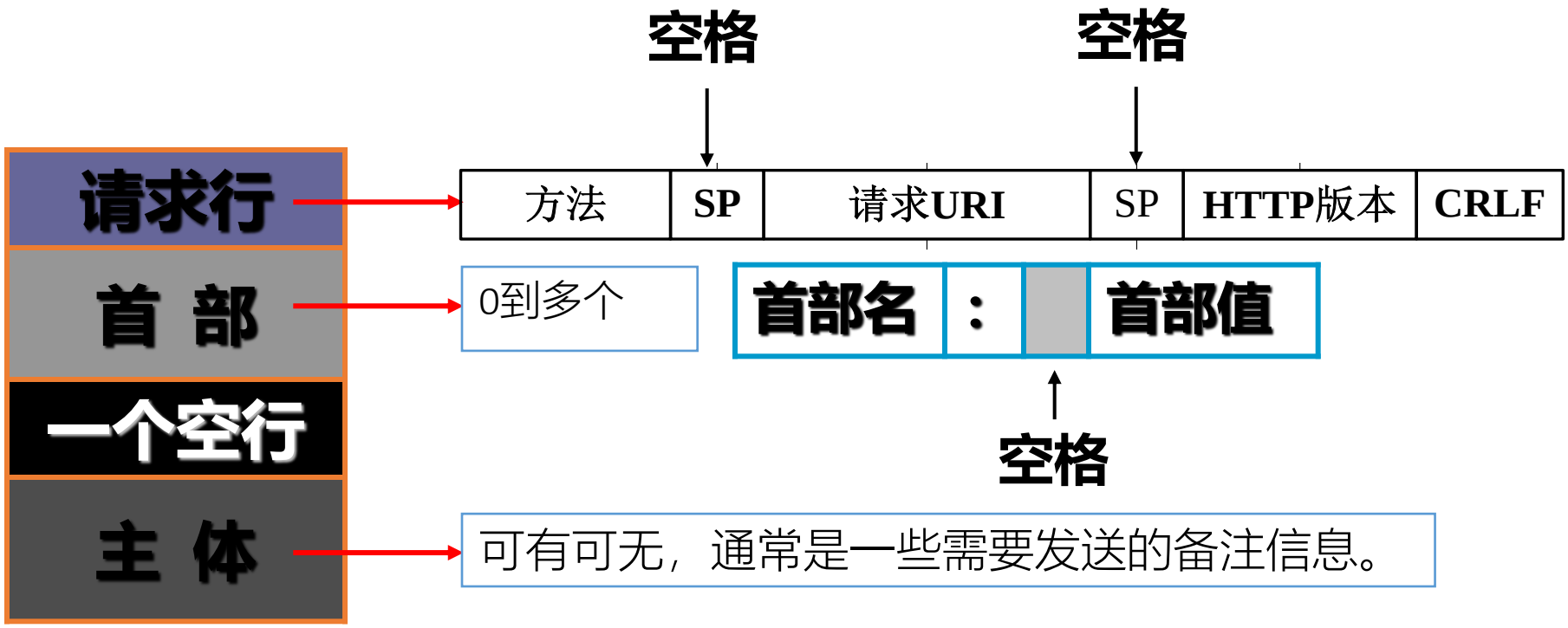
HTTP协议

- 无连接
  - 每次连接只处理一个请求
  - 服务器处理完客户的请求，并收到客户的应答后，即断开连接
- 无状态
  - 协议对于事务处理没有记忆能力
  - 如果后续处理需要前面的信息，则它必须重传
- 媒体独立的
  - 只要客户端和服务器知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送
  - 客户端以及服务器指定使用适合的内容类型

HTTP消息

- 消息类型：
  - 请求：客户端->服务器端
  - 响应：服务器端->客户端
- 消息构成：
  - 请求/响应行
  - 消息头
  - 消息体

HTTP请求


空格
空格
请求行
方法
SP
请求URI
SP
HTTP版本
CRLF
首 部
0到多个
首部名
:
首部值
空格
一个空行
主 体
可有可无，通常是一些需要发送的备注信息。

HTTP请求

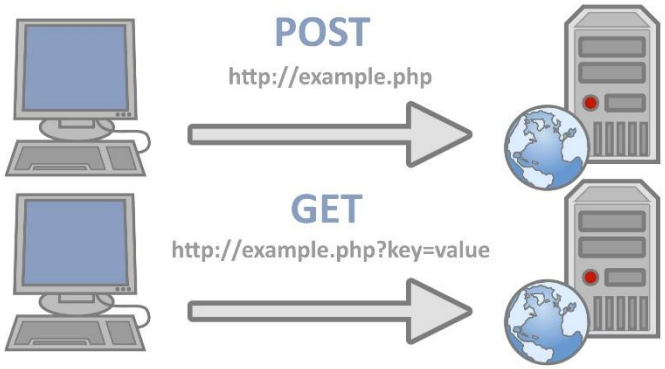
① 是请求方法，HTTP/1.1 定义的请求方法有8种。一般常用的是GET和POST。
② 为请求对应的URL地址，它和报文头的Host属性组成完整的请求URL
③ 是协议名称及版本号。
④ 是HTTP的报文头，包含若干个属性，格式为"属性名:属性值"，服务端据此获取客户端信息。
⑤ 是报文体，承载请求参数的数据等内容

HTTP请求

方法名	备注
GET	获取一个URL指定的资源,即资源实体
POST	从客户端向服务器端发送一些信息（需要主体部分）
HEAD	只请求文档的首部信息，而不包含文档的内容
PUT	从服务器向客户端发送一些信息
DELETE	请求服务器删除指定的页面
TRACE	网络跟踪,允许客户端查看消息回收过程（用于测试）
CONNECT	与PROXY之间的连接管理
OPTIONS	查询能力,允许客户端查看服务器的性能

GET还是POST?

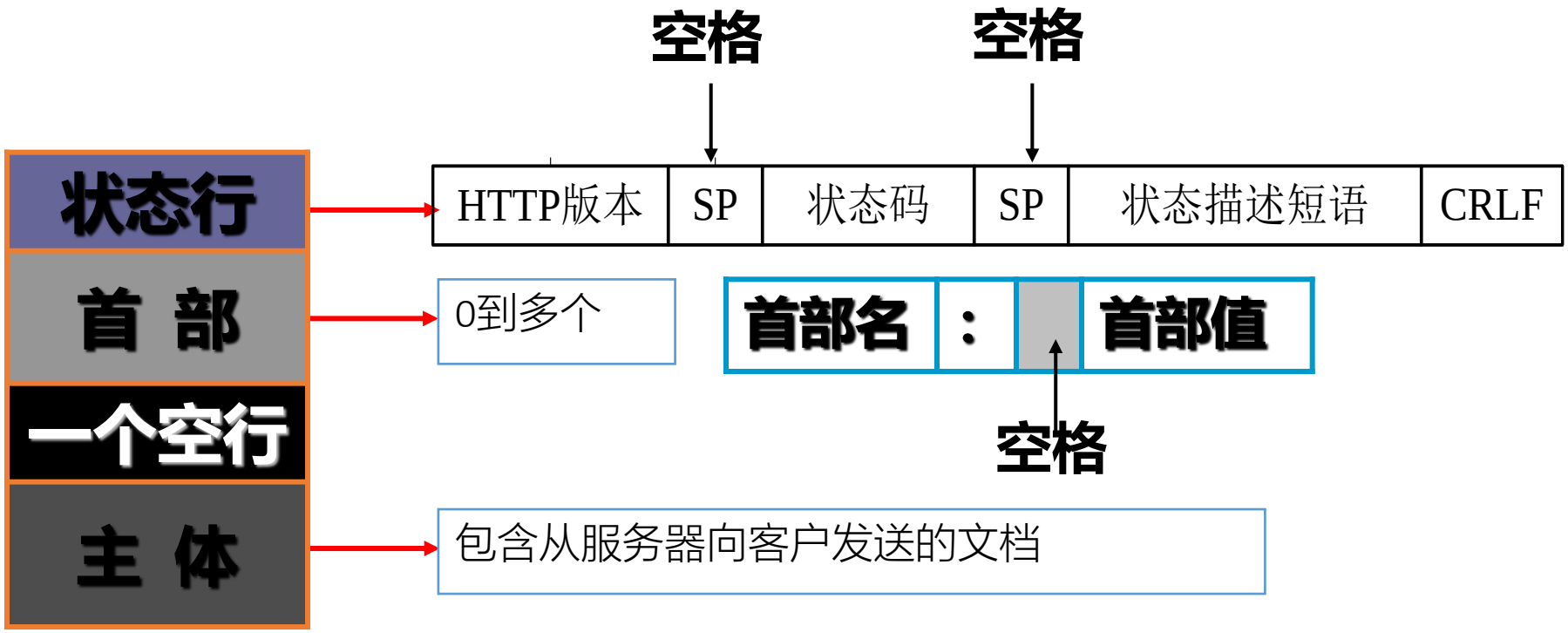
- 与 POST 相比，GET 更简单也更快，且在大部分情况下都能用。
- 然而，在以下情况中，请使用 POST 请求：
  - 不愿使用缓存文件（更新服务器上的文件或数据库）
  - 向服务器发送大量数据（POST 没有数据量限制）
  - 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠



HTTP请求

首部字段	含义
User-agent	标志客户程序
Accept	客户端能够接受的媒体格式
Accept-charset	客户端能够处理的字符集
Accept-encoding	客户端能够处理的编码方案
Accept-language	客户端能够接受的语言
Authorization	客户端具有何种准许
From	用户的电子邮件地址
Host	客户端的主机和端口号
If-modified-since	只当比指明日期更加新时才发送这个文档
If-match	只当与给定标记匹配时才发送这个文档
If-not-match	只当与给定标记不匹配时才发送这个文档
If-range	只发送缺少的那部分文档
If-unmodified-since	若在指明日期之后未改变，则发送文档
Referrer	指明被链接的文档的URL
User-went	标志客户程序

HTTP响应



响应首部：指明服务器的配置或主体信息类型、长度、压缩方法、

HTTP响应

① 报文协议及版本；
② 状态码及状态描述；
③ 响应报文头，也是由多个属性组成；
④ 响应报文体，即我们真正要的“干货”。

HTTP响应

HTTP的响应状态码由5段组成：

```
1xx 消息，一般是告诉客户端，请求已经收到了，正在处理，别急...
2xx 处理成功，一般表示：请求收悉、我明白你要的、请求已受理、已经处理完成等信息.
3xx 重定向到其它地方。它让客户端再发起一个请求以完成整个处理。
4xx 处理发生错误，责任在客户端，如客户端的请求一个不存在的资源，客户端未被授权，禁止访问等。
5xx 处理发生错误，责任在服务端，如服务端抛出异常，路由出错，HTTP版本不支持等。
```

状态码	原因短语
200	正确 ok
201	创建 created
202	接收 accepted
204	无内容 no content
300	多种选择
301	永久移动
302	暂时移动
304	未被修改

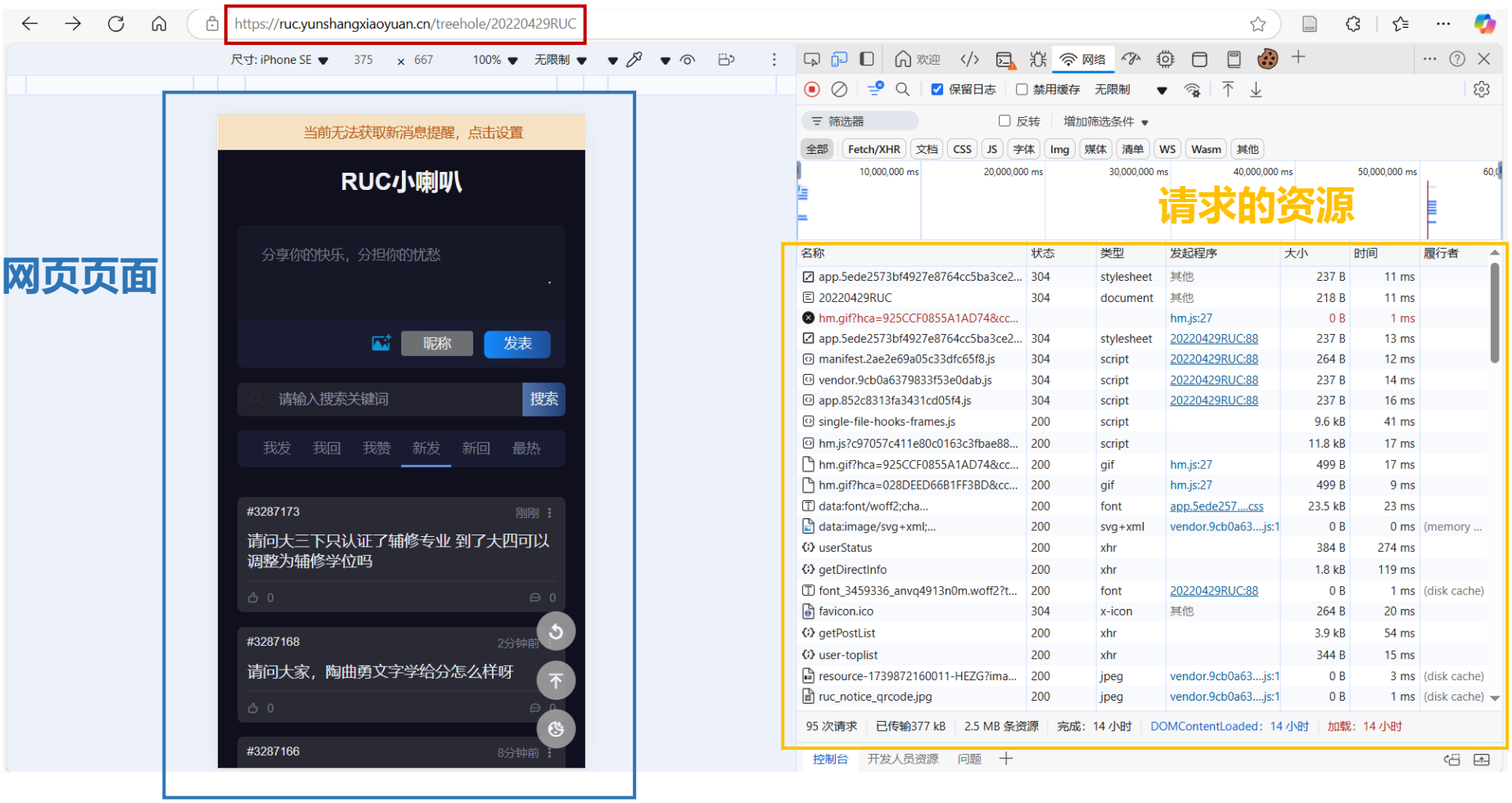
状态码	原因短语
400	错误请求
401	未授权
403	禁止
404	未发现
500	内部服务错误
501	未实现
502	错误网关
503	服务未提供

HTTP响应

首部字段	说明
Date	给出当前日期
Last-Modified	请求文档最近修改时间
Content-encoding	主体所用的编码
Content-length	数据长度（字节）
Content-type	数据类型
Accept-Range	服务器指明它能接受请求的字节范围
Age	表示发送者对响应（或重验证）在源服务器上产生以来的时间估计
Location	用于为完成请求或识别一个新资源，使接收者能重定向于由它指示的URI而不是请求URI
Proxy-Authenticate	必须包含在407响应中，由一个challenge和parameters组成，challenge指明认证方案，而parameters应用于此请求URI的代理
Retry-After	用于一个503响应，向请求端指明服务不可得的时长；用于3xx（重定向）响应，指明用户代理再次提交已重定向请求之前的最小等待时间
Server	包含源服务器用于处理请求的软件信息
WWW-Authenticate	必须包含在401响应中，字段值至少应该包含一个指明认证方案的callenge和适用于请求URI的参数，用于通知客户方需要的认证信息（如用户名、口令等）

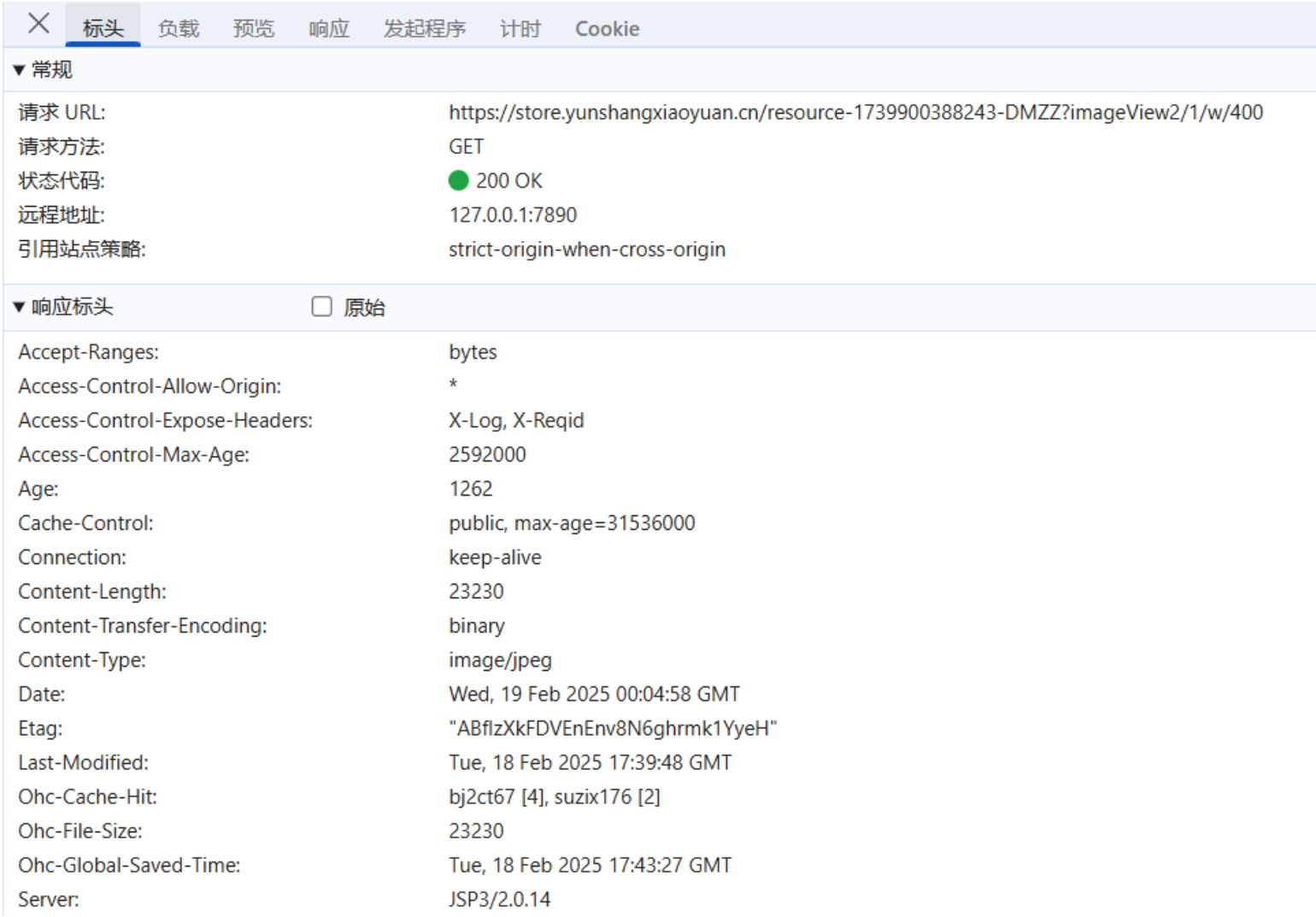
实践

打开浏览器，访问一个网页，按下F12或Ctrl+Shift+i 可以打开浏览器的开发工具（devtool）。在devtool中可以看到网页的详细信息，包括页面布局，请求的网络资源等。



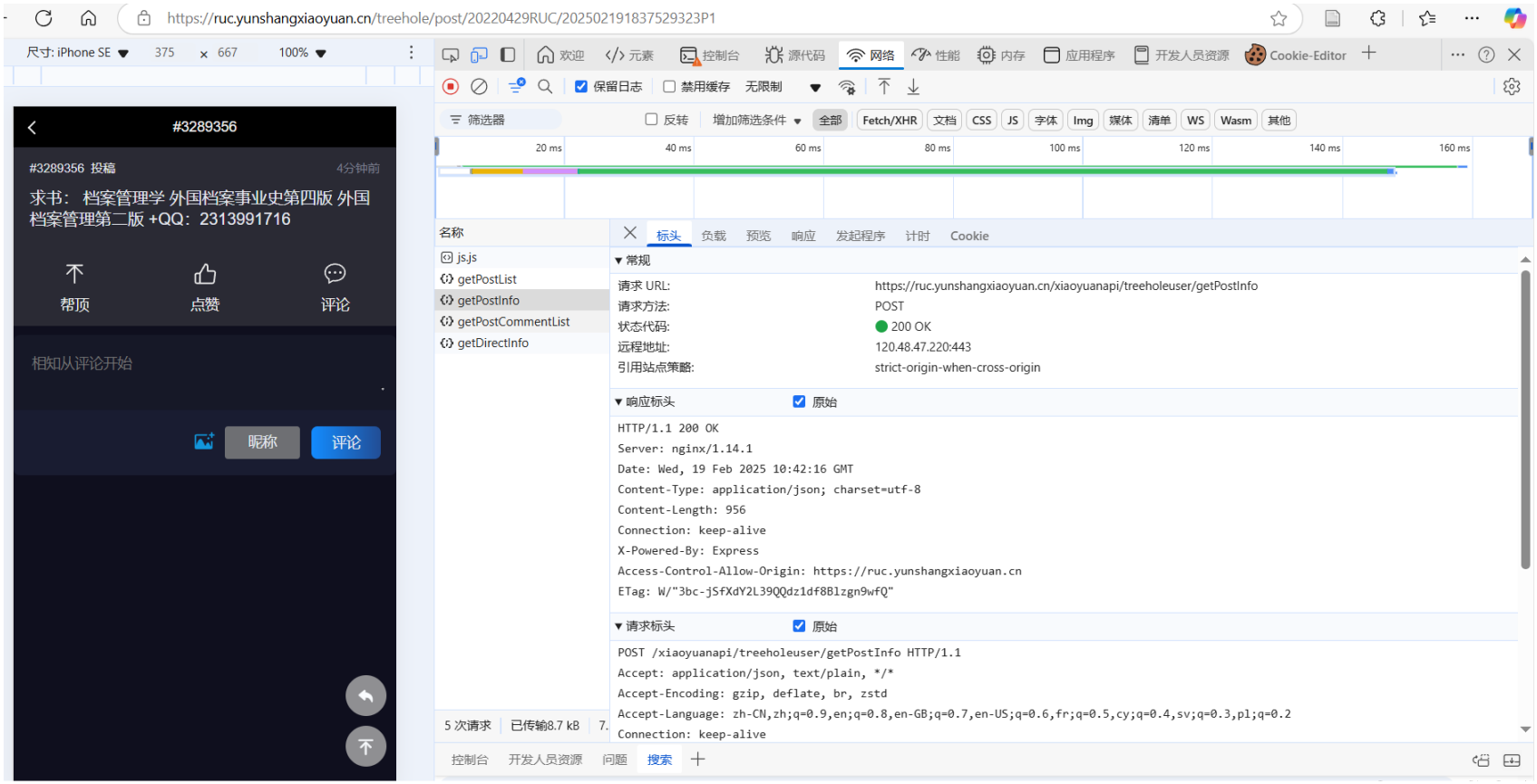
实践

点击某一个资源，便可查看这个资源的详细信息



实例：HTTP请求报文

- 点击匿名墙的一个帖子，以其中请求帖子内容的HTTP报文为例



实例：HTTP请求报文

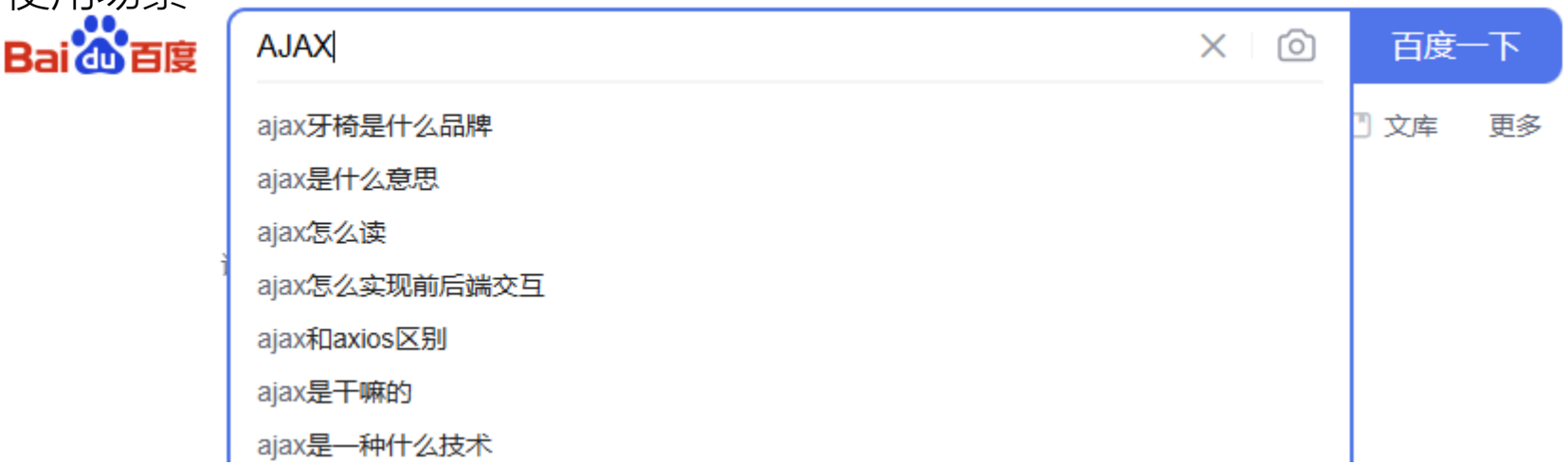
- 点击匿名墙的一个帖子，以其中请求帖子内容的HTTP报文为例

```
POST /xiaoyuanapi/treeholeuser/getPostInfo HTTP/1.1
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6,fr;q=0.5,cy;q=0.4,sv;q=0.3,pl;q=0.2
Connection: keep-alive
Content-Length: 209
Content-Type: application/json;charset=UTF-8
Cookie: Hm_lvt_e64b246d87369ea1e8501aa005f04829=1739872487;
Hm_lvt_c97057c411e80c0163c3fbae882ff468=1739864467,1739961668;
Hm_lpvt_c97057c411e80c0163c3fbae882ff468=1739961668; HMACCOUNT=925CCF0855A1AD74
Host: ruc.yunshangxiaoyuan.cn
Origin: https://ruc.yunshangxiaoyuan.cn
Referer: https://ruc.yunshangxiaoyuan.cn/treehole/post/20220429RUC/20250219183752932 3P1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 16_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like
Gecko) Version/16.6 Mobile/15E148 Safari/604.1 Edg/133.0.0.0

...（报文主体省略）
```

3.5.2 AJAX

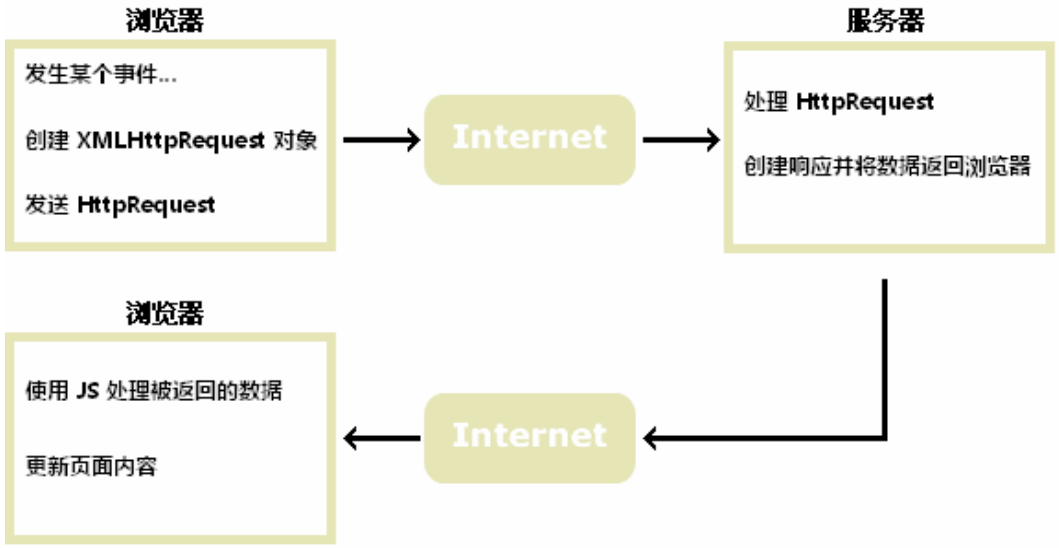
- Ajax(Asynchronous JavaScript and XML)：异步的JavaScript和XML技术，是一种支持异步请求的技术
- 它可以异步地向服务器发送请求，**在等待响应的过程中，不会阻塞当前页面**，在这种情况下，浏览器可以做自己的事情。直到成功获取响应后，浏览器才开始处理响应数据。

使用场景

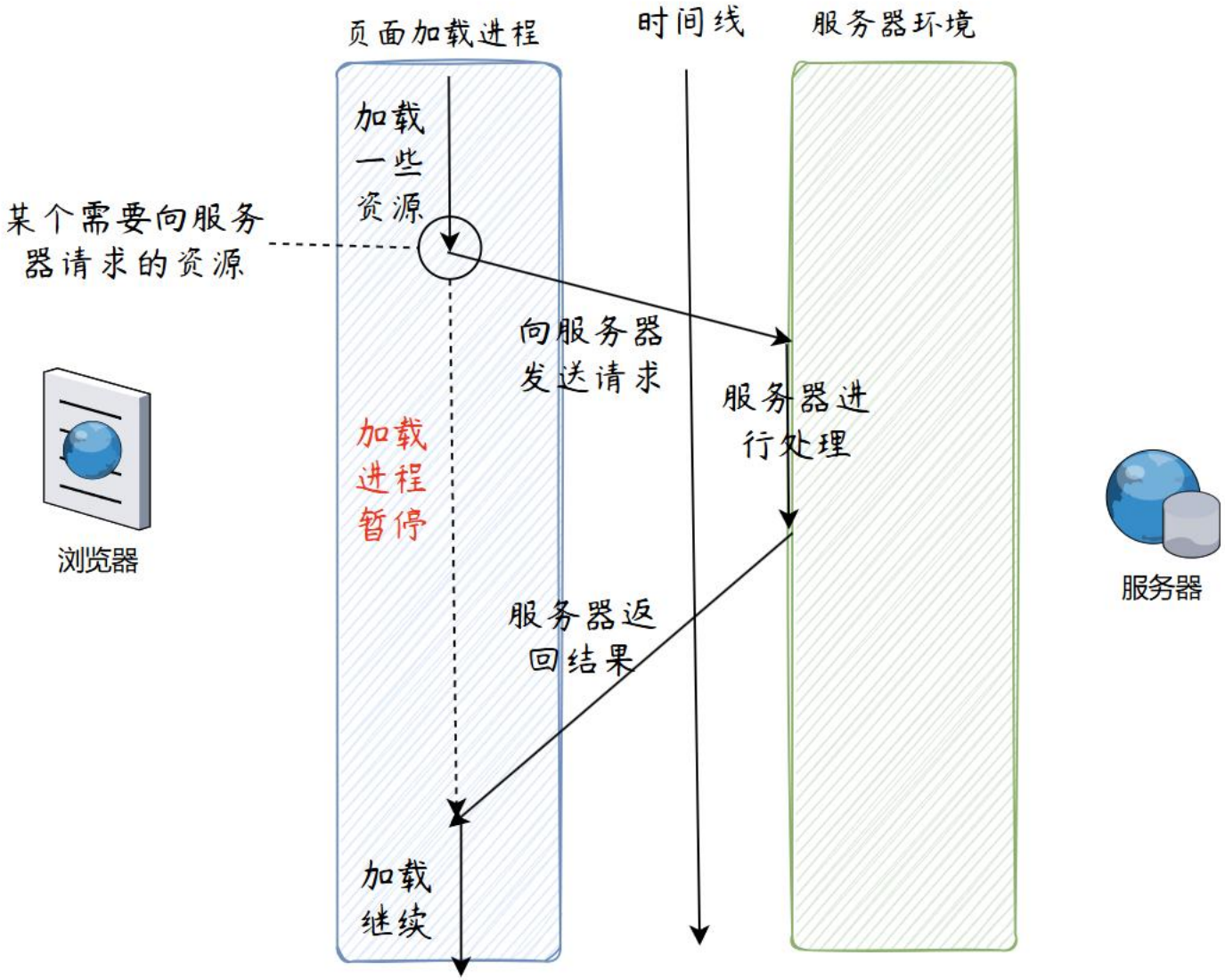


AJAX与传统页面的区别

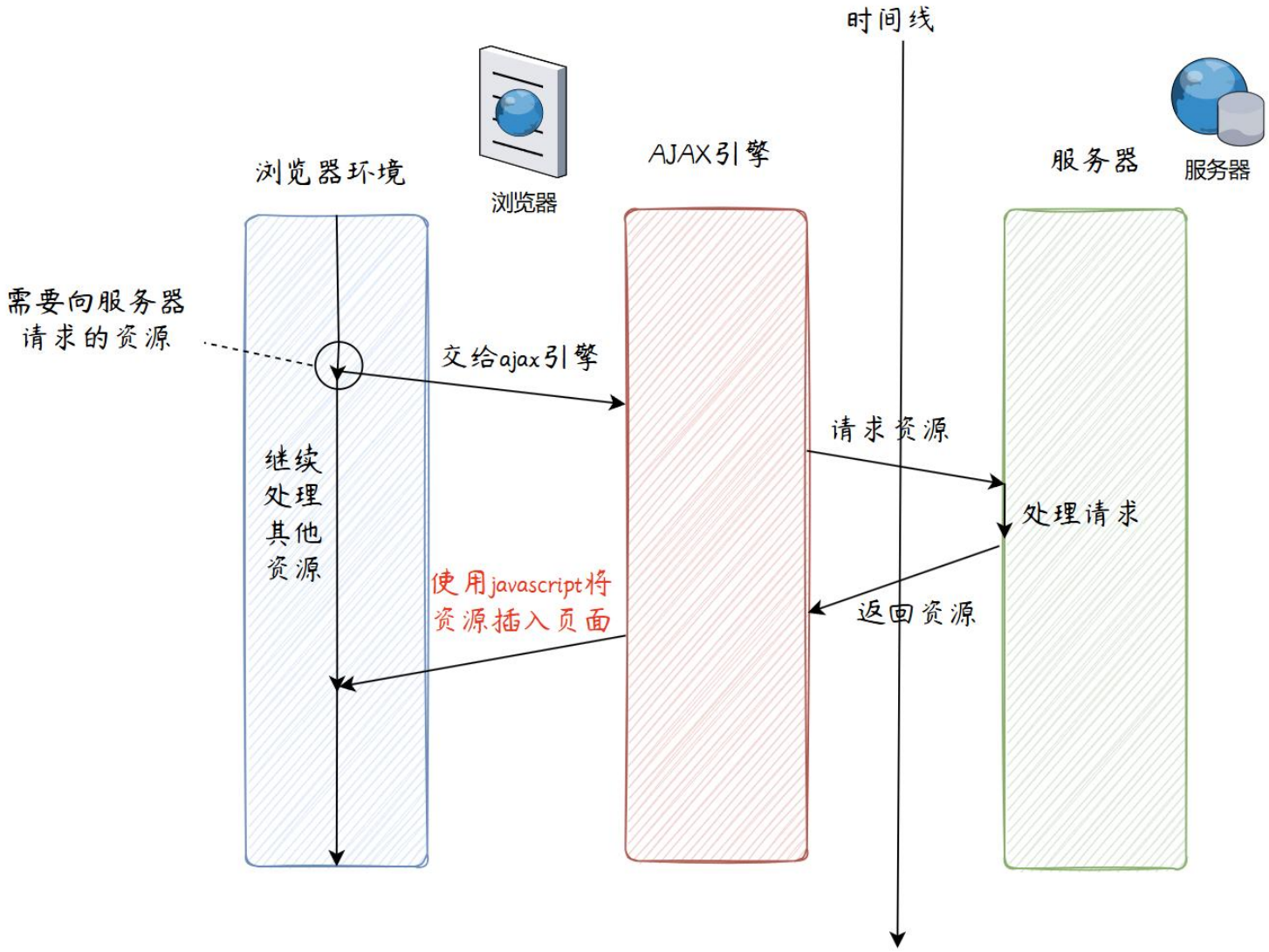
- AJAX 是一种用于创建快速动态网页的技术。
- 通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。
- 传统的网页（不使用 AJAX）如果需要更新内容，**必需重载整个页面。**



传统Web应用的工作模式

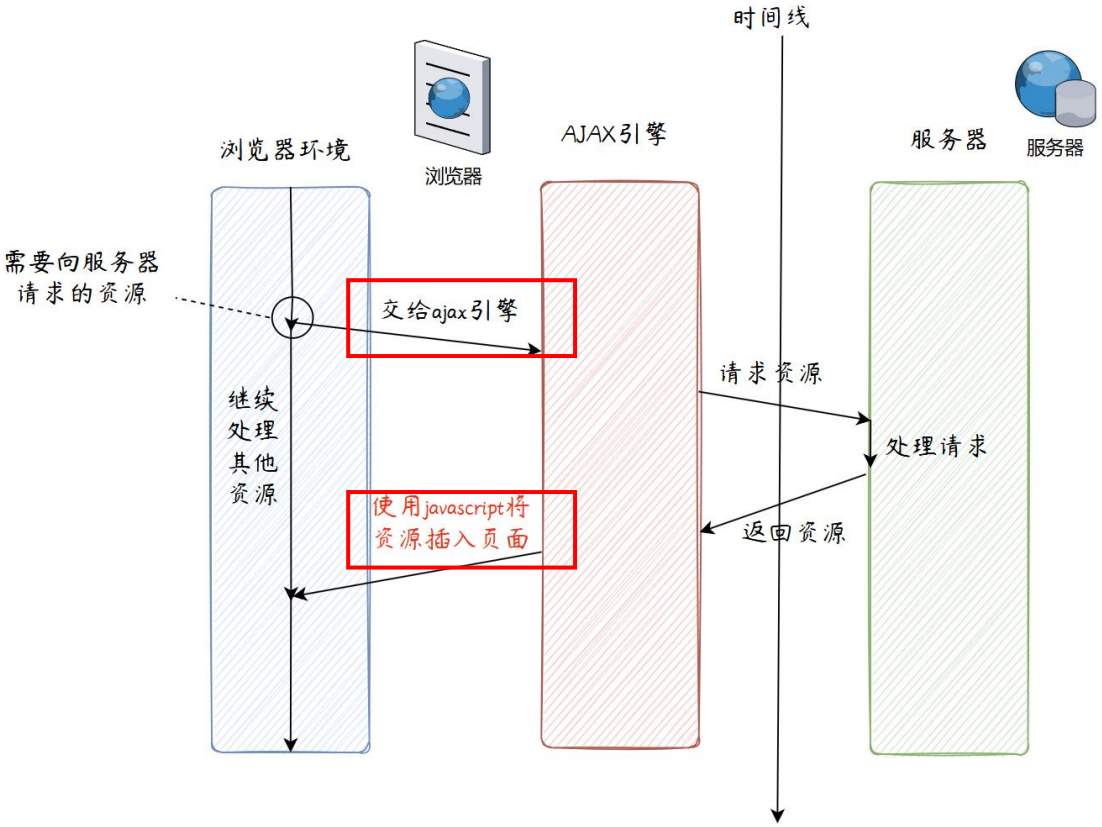


Ajax Web应用的工作模式



AJAX

- 基于ajax的流程，不难发现ajax最重要的有两部分：
  - 浏览器将请求资源的行为交给ajax引擎
  - 接收到服务器返回值后，ajax引擎将资源插入页面中

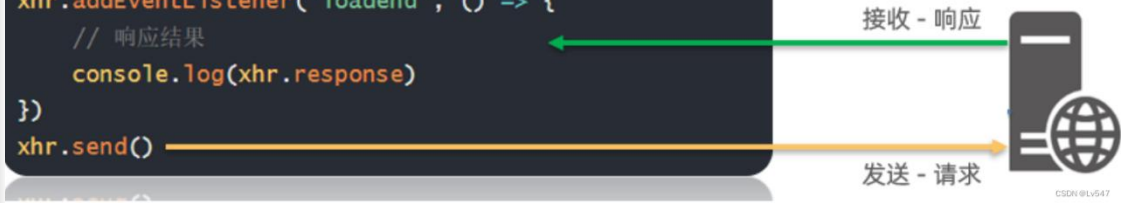


浏览器 -> AJAX引擎

- 目前有两种方式
  - XMLHttpRequest
  - fetch（了解即可）

```fetch
fetch("请求url", {
  method: "请求方式",
  body: "请求携带的参数"
})
  .then((res) => {
    return res.json();
  })
  .then((res) => {
    console.log(res);
  });
```

```xhr
const xhr = new XMLHttpRequest()
xhr.open('请求方法', '请求url网址')
xhr.addEventListener('loadend', () => {
    // 响应结果
    console.log(xhr.response)
})
xhr.send()
```



XMLHttpRequest

- 流程
  - 创建XMLHttpRequest对象

```
1 const xhr = new XMLHttpRequest()
```

  - 配置请求方法和请求 url 地址

```
1 xhr.open('GET', 'https://www.example.com/getapi')
```

```
1 xhr.open('POST', 'https://www.example.com/POSTapi')
```

  - 设置收到请求结果后的行为：定义onloadend事件处理函数

```
1 xhr.addEventListener('loadend', () => {
2   console.log(xhr.response)
3 })
```

  - 向服务器发送请求
    - 如果是post方法，将参数放在send中传输
    - 如果是get方法，则在第二步中直接写在url中

```
1 xhr.send(postdata)
```

XMLHttpRequest

- 总览

```
<script>
  /**
   * 目标：使用XMLHttpRequest对象与服务器通信
   *  1. 创建 XMLHttpRequest 对象
   *  2. 配置请求方法和请求 url 地址
   *  3. 监听 loadend 事件，接收响应结果
   *  4. 发起请求
   */
  // 1. 创建 XMLHttpRequest 对象
  const xhr = new XMLHttpRequest()

  // 2. 配置请求方法和请求 url 地址
  xhr.open('GET', 'http://hmajax.itheima.net/api/province')

  // 3. 监听 loadend 事件，接收响应结果
  xhr.addEventListener('loadend', () => {
    console.log(xhr.response);
  })

  // 4. 发起请求
  xhr.send()
</script>
```

实例：vruc登录AJAX实例

```
<script>
  const loginForm = document.getElementById('loginForm');
  const loginBtn = document.getElementById('loginBtn');
  // 登录按钮点击事件
  loginBtn.addEventListener('click', function(e) {
    e.preventDefault();
    // 收集表单数据
    const formData = new FormData(loginForm);
    // 创建XHR对象
    const xhr = new XMLHttpRequest();
    // 配置请求
    xhr.open('POST', 'php/login.php', true);
    // 结果处理
    xhr.addEventListener('loadend', () => {
        const response = JSON.parse(xhr.responseText);
        if (response.success) {
        location.href='php/pannel.php'
      } else {
        document.getElementById('errorMessage').firstChild.innerText = response.message;
      }
    })
    // 发送请求
    xhr.send(formData);
  });
</script>
```

前端代码，在这里，我们假设后端返回的是json数据并且格式为
{"success": false, "message": "用户名或密码错误"}

实例：vruc登录AJAX实例

```php
foreach($_POST as $key⇒$val) {
    $$key = $val;
}

$response = [
    'success' ⇒ false,
    'message' ⇒ '未知错误',
    'redirect' ⇒ null
];

if ($code === $captcha[$id]) {
    $sql = "SELECT * FROM user where name='$name' and passwd='$passwd'";
    $res = $conn→query($sql);

    if ($res→num_rows > 0) {
        $row = $res→fetch_assoc();
        $_SESSION["role"] = $row["role"];
        $_SESSION["name"] = $row["name"];
        $response['success'] = true;
    $response['message'] = '登录成功';
    } else {
        $response['message'] = '用户名或密码错误';
        echo json_encode($response);
        die();
    }
    echo json_encode($response);
} else {
        $response['message'] = 'wrong captcha';
        echo json_encode($response);
    die();
}
```

后端代码，首先取出POST数据，然后再数据库中查询，并设置对应的返回值，最后转化为json格式发送

实例：vruc登录AJAX实例

- 效果展示