第4章 PHP进阶

4.1 错误处理

1、错误处理

错误处理：指的是系统（或者用户）在对某些代码进行执行的时候，发现有错误，就会通过错误处理的形式告知程序员。

2、错误分类

（1）语法错误:用户书写的代码不符合PHP的语法规范，语法错误导致在编译过程中不通过，所以代码不会执行（Parse error）

（2）运行时错误：代码编译通过，但是代码在执行的过程中会出现一些条件不满足导致的错误（runtime error）

（3）逻辑错误：程序员在写代码的时候不够规范，出现了一些逻辑性的错误，导致代码正常执行，但是得不到想要的结果。

3、错误代号：

（1）系统错误：E_ERROR，E_WARNING，E_NOTICE

（2）用户错误：E_USER_ERROR,E_USER_WAENING，E_USER_NOTICE

（3）其他：E_ALL，代表着所有从错误（通常在进行错误控制的时候使用比较多），建议在开发过程中（开发环境）使用。

4.1.1 错误级别

错误级别：PHP中有多种错误类型，每个错误都对应了错误级别，使用常量表示错误级别。常见的错误级别：

级别常量	值	描述
E_ERROR	1	致命的运行时错误，这类错误不可恢复，会导致脚本停止运行
E_WARNING	2	运行时警告，仅给出提示信息，但是脚本不会停止运行
E_PARSE	4	编译时语法解析错误，说明代码存在语法错误，无法执行
E_NOTICE	8	运行时通知，表示脚本遇到可能会表现为错误的情况
E_CORE_ERROR	16	类似E_ERROR，是由PHP引擎核心产生的
E_CORE_WARNING	32	类似E_WARNING，是由PHP引擎核心产生的
E_COMPILE_ERROR	64	类似E_ERROR，是由Zend脚本引擎产生的

级别常量	值	描述
E_COMPILE_WARNING	128	类似E_WARNING，是由Zend脚本引擎产生的
E_USER_ERROR	256	类似E_ERROR，由用户在代码中使用trigger_error()产生的
E_USER_WARNING	512	类似E_WARNING，由用户在代码中使用trigger_error()产生的
E_USER_NOTICE	1024	类似E_NOTICE，由用户在代码中使用trigger_error()产生的
E_STRICT	2048	严格语法检查，确保代码具有互用性和向前兼容性
E_DEPRECATED	8192	运行时通知，对未来版本中可能无法正常工作的代码给出警告
E_ALL	32767	表示所有的错误和警告信息（在PHP 5.4之前不包括E_STRICT）

1.Notice(E_NOTICE)

Notice：是代码不严谨造成的，不会影响脚本继续执行。

Notice错误信息代码示例**

```
// ① 使用未定义的变量
echo $var;  // 提示信息“Notice:Undefined variable…”
// ② 使用未定义的常量
echo PI;        // 提示信息“Notice:Use of undefined constant…”
```

2.Warning(E_WARNING)

Warning：比Notice更严重一些，不会影响脚本继续执行。

Warning警告信息代码示例**

```
// ① 除法运算时,除数为0
echo 5 / 0;     // 提示信息“Warning:Division by zero…”
// ② 使用include包含不存在的文件
include '1234';     // 提示信息“Warning:include():Failed opening…”
```

3.Fatal error(E_ERROR)

Fatal error：致命错误，一旦发生这种错误，PHP脚本会立即停止执行。

Fatal error错误信息代码示例**

```
display();                  // Fatal error:Uncaught Error:Call to undefined
function …
echo 'hello';             // 前一行发生错误,此行代码不会执行
```

例:

```
<?php
//parse error
$a=100
echo $a;

//warning error
$b=0;
echo $a/$b;
if($b==0){
    //人为触发
    trigger_error('除数不能为0');//默认notice，会继续执行
    trigger_error('除数不能为0', E_USER_ERROR);//默认error，代码不会执行
}
echo $a/$b;
echo 'hello';
```

4.1.2 错误处理

熟悉错误处理的方法，能够控制错误的显示、隐藏，或将错误信息记录到日志文件中

1.在浏览器中显示错误信息

在浏览器中显示错误信息的两种方式：通过修改配置文件来显示错误报告，通过PHP提供的函数来显示错误报告。

2、可以在运行的 PHP脚本中去设置:在脚本中定义的配置项级别比配置文件高（通常在开发当中都会在代码中去进行控制和配置）

2.记录错误日志

错误日志设置，在实际生产环境中，不会直接让错误赤裸裸的展示给用户:

- 不友好
- 不安全:错误会暴露网站很多信息（路径、文件名）
  所以在生产环境中，一般不显示错误（错误也比较少），但是不可能避免会出现错误（测试的时候不会发现所有的问题），这个时候不希望看到，但是又希望捕捉到可以让后台程序员去修改:需要保存到日志文件中，需要在PHP配置文件中或者代码中（ini _set）设置对应error_log配置项。
- 开启日志功能
- 指定路径

记录错误日志的两种方式：通过配置文件和error_log()函数记录错误日志信息。

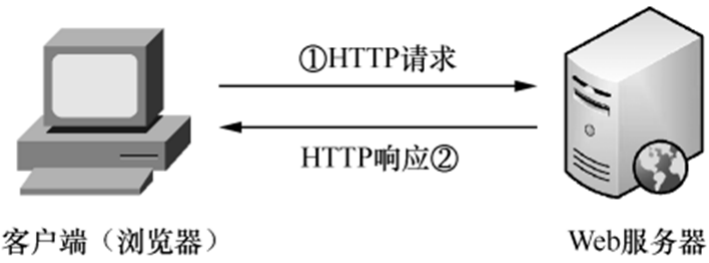
4.2HTTP

目标：熟悉什么是HTTP， 能够说出HTTP的概念和特点。

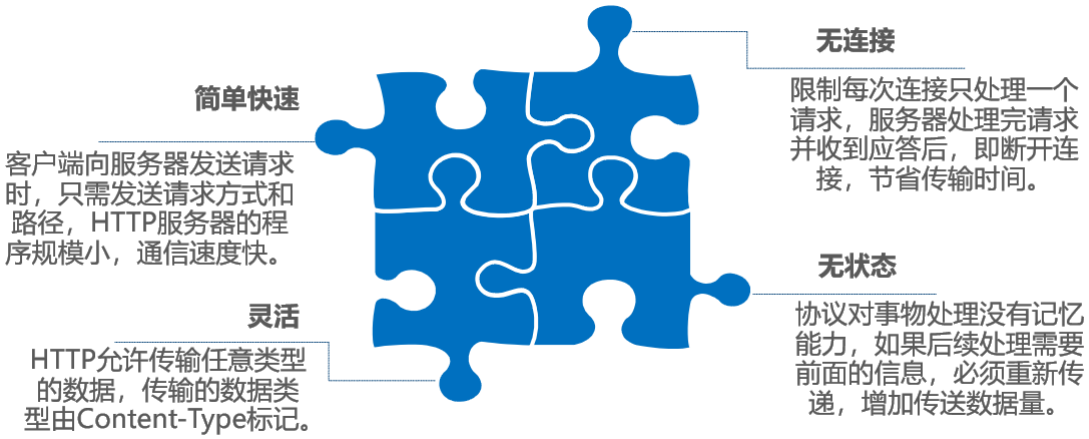
4.2.1 什么是HTTP

HTTP：超文本传输协议（HyperText Transfer Protocol）由W3C组织推出，用于定义浏览器与Web服务器之间数据交换的格式。

浏览器与Web服务器交互过程：



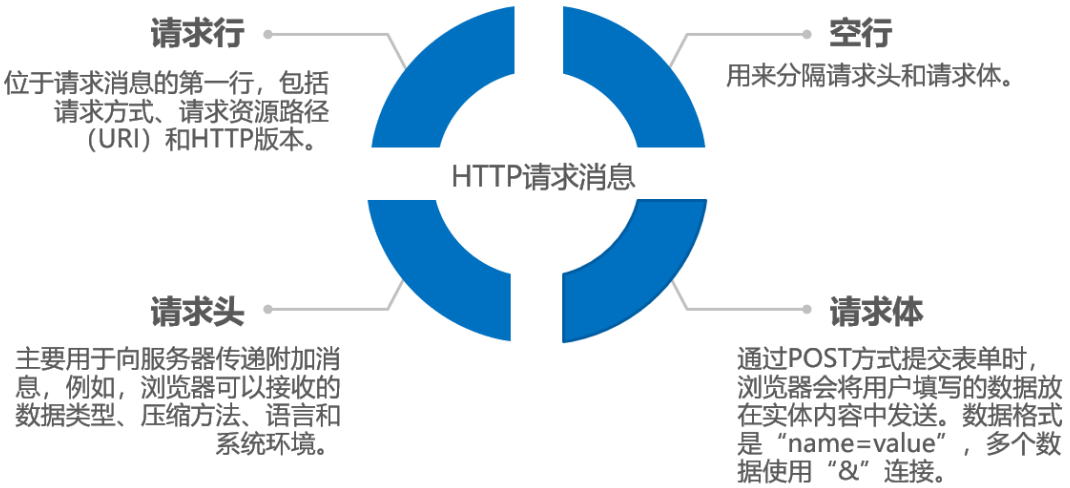
HTTP在Web开发中占据重要位置的原因：



4.2.2 HTTP请求

目标：掌握HTTP请求，能够说出HTTP请求的各个部分的含义：

HTTP请求消息：用户在浏览器中访问某个URL地址时，浏览器会向服务器发送请求。



HTTP请求方式：

请求方式	说明
HEAD	通过发送HTTP请求从服务器获取数据
GET	与HEAD一样，但是GET是通过自身携带的数据来获取服务器数据
POST	向指定资源提交数据进行处理请求
PUT	向指定资源位置来提交数据
DELETE	通过指定数据来删除服务器的数据
OPTIONS	获取URL所支持的方式

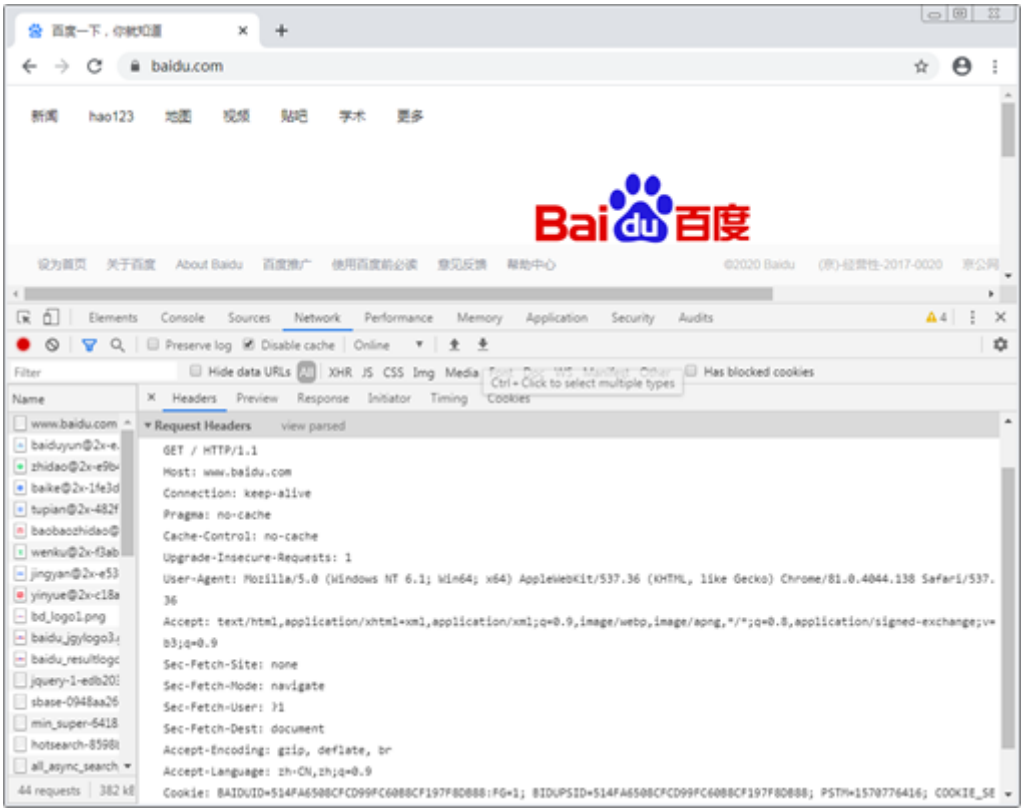
4.2.3 查看请求信息

目标：熟悉如何查看请求信息，能够利用Chrome浏览器开发者工具查看请求信息

借助工具查看请求信息：

使用Chrome浏览器，按F12键打开开发者工具，切换到Network选项卡刷新网页，就可以看到当前网页从第1个请求开始，依次发送的所有请求。

以查看百度网站请求信息为例：



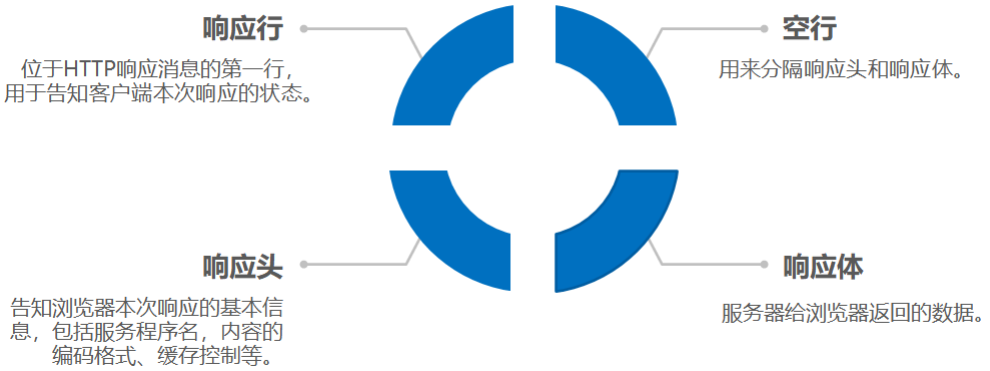
常见的请求头字段：

请求头	描述
Accept	客户端浏览器支持的数据类型
Accept-Charset	客户端浏览器采用的编码
Accept-Encoding	客户端浏览器支持的数据压缩格式
Accept-Language	客户端浏览器所支持的语言包，可以指定多个
Host	客户端浏览器想要访问的服务器主机
If-Modified-Since	客户端浏览器对资源的最后缓存时间
Referer	客户端浏览器指向的Web页的URL
User-Agent	客户端的系统信息，包括使用的操作系统、浏览器版本号等
Cookie	客户端向服务器发送请求时发送
Cache-Control	客户端浏览器的缓存控制
Connection	请求完成后，客户端希望是保持连接还是关闭连接

4.2.4 HTTP响应

目标：掌握HTTP响应，能够说出HTTP响应的各个部分的含义。

HTTP响应消息：服务器接收到请求数据后，将处理后的数据返回给客户端。



以百度网站为例，查看响应信息：

响应状态码：服务器对客户端请求的各种不同的处理结果和状态，由一个三位十进制数表示。

响应状态码分类：

1xx：成功接收请求，要求客户端继续提交下一次请求才能完成整个处理过程。

2xx：成功接收请求并已完成整个处理过程。

3xx：未完成请求，客户端需要进一步细化请求。

4xx：客户端的请求有错误。

5xx：服务器端出现错误。

常见响应状态码：

状态码	含义	说明
200	正常	客户端的请求成功，响应消息返回正常的请求结果
403	禁止	服务器理解客户端的请求，但是拒绝处理。通常由服务器上文件或目录的权限设置导致
404	找不到	服务器上不存在客户端请求的资源
500	内部服务器错误	服务器内部发生错误，无法处理客户端的请求

常见的响应头字段：

响应头	含义
Server	服务器的类型和版本信息
Date	服务器的响应时间
Expires	控制缓存的过期时间
Location	控制浏览器显示哪个页面
Accept-Ranges	服务器是否支持分段请求，以及请求范围
Cache-Control	服务器控制浏览器如何进行缓存
Content-Disposition	服务器控制浏览器以下载方式打开文件
Content-Encoding	实体内容的编码格式
Content-Length	实体内容的长度
Content-Language	实体内容的语言和国家名
Content-Type	实体内容的类型和编码类型
Last-Modified	请求文档的最后一次修改时间
Transfer-Encoding	文件传输编码
Set-Cookie	发送Cookie相关的信息
Connection	是否需要持久连接

4.2.5 PHP设置响应头

掌握如何设置响应头，能够利用header()函数设置响应头

设置响应头：通过header()函数设置。

header（Location：url）让浏览器重定向请求，立即发起（浏览器不会解析剩下的响应头和响应体）

header（refresh：时间；url）让浏览器指定时间后重定向，浏览器会继续执行后面所有内容（时间内）

header（content-type）:让浏览器按照明确的格式解析内容

header（content-disposition）：让浏览器如何处理内容（选择应用程序）

```php
<?php
#立即重定向
header('Location:http://www.baidu.com');
echo 'hello world';

#延时重定向
header('Refresh:3;url=http://www.baidu.com');
echo 'hello world';

#解析图片：jpg图片
header('Content-Type:image/jpeg');
#输出图片
echo file_get_contents('php.jpg');
```

```
#应用程序：让浏览器当附件保存，命名为美女.jpg(下载使用)
header('Content-Type:application/octet-stream');
header('Content-disposition:attachment;filename="美女.jpg"');
echo file_get_contents('php.jpg');
```

MIME：表示内容的类型，表示方法为“大类别/具体类型”。

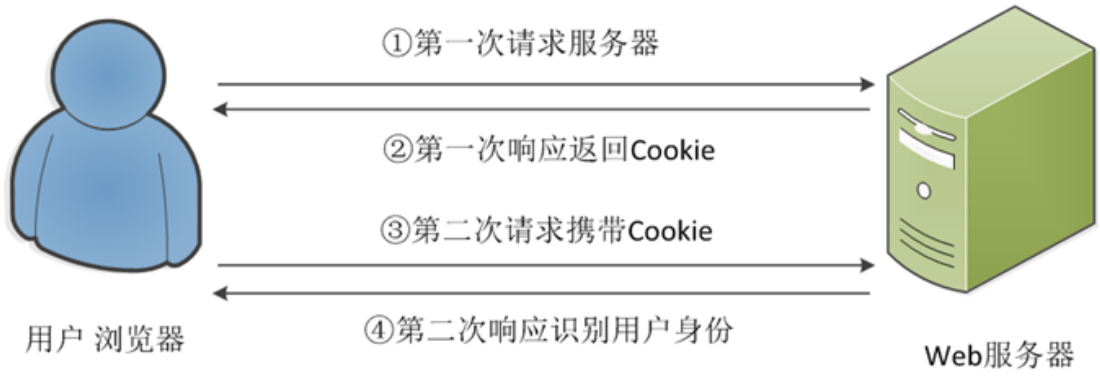
类型	含义
text/plain	普通文本（.txt）
text/xml	XML文档（.xml）
text/html	HTML文档（.html）
image/gif	GIF图像（.gif）
image/png	PNG图像（.png）
image/jpeg	JPEG图像（.jpg）

4.3会话技术

4.3.1 Cookie简介

目标：熟悉什么是Cookie，能够说出Cookie的用途和传输过程。

Cookie在浏览器和服务器之间的传输过程：



4.3.2 Cookie的基本使用

目标：掌握Cookie的基本使用，能够使用setcookie()函数创建或修改Cookie。

创建Cookie：使用setcookie()函数创建或修改Cookie。

语法格式：

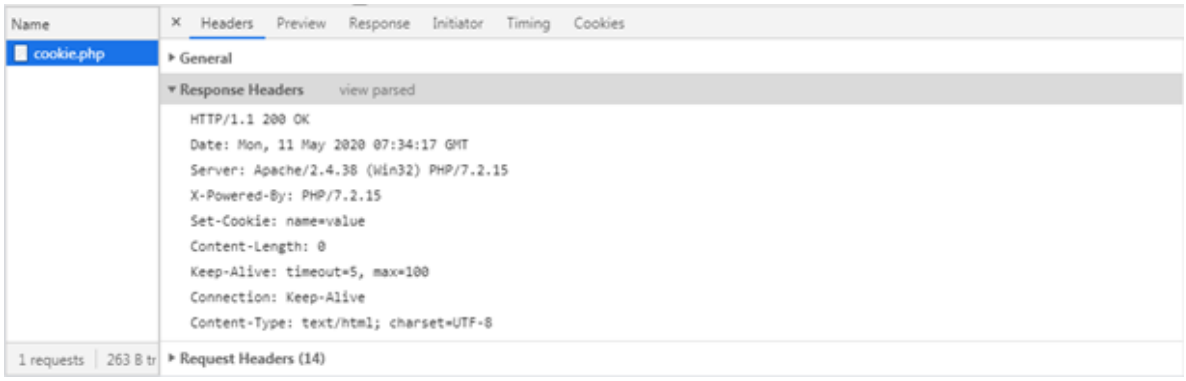
```
bool setcookie(
        string $name,        // Cookie的名称（必须）
        string $value = '',  // Cookie的值（可选）
        int $expire = 0,     // Cookie的有效期（可选）
        string $path = '',   // Cookie在服务器端的路径（可选）
        string $domain = '', // Cookie的有效域名（可选）
        bool $secure = false,// 指定是否通过安全的HTTPS连接来传输（可选）
        bool $httponly = false // 指定Cookie只能通过HTTP协议访问（可选）
)
```

创建Cookie：使用setcookie()函数创建Cookie。

```
<?php
setcookie('name', 'value');
```

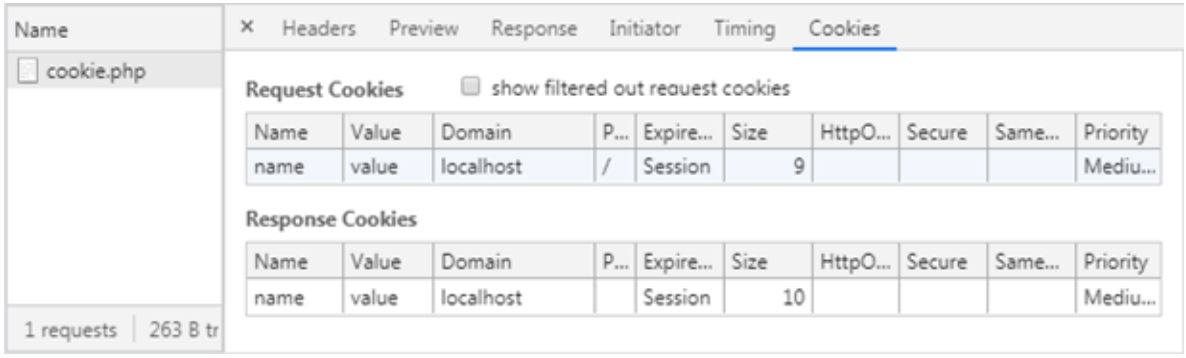
查看响应头信息：

浏览器访问，查看响应头信息。



HTTP中的Cookie信息：在开发者工具中

切换到【Network】→【Cookie】。



读取Cookie：使用超全局变量$_COOKIE读取Cookie。

读取Cookie代码示例

```
var_dump($_COOKIE);  // 输出结果：array(2) { ["name"]=> string(5) "value" }
```

注意：当PHP第一次通过setcookie()函数创建Cookie时，*$COOKIE中没有数据，只有当浏览器下次请求并携带Cookie时，才能通过*$COOKIE获取到。

Cookie存储多个值：在Cookie名后添加“[]”进行设置。

Cookie存储复杂数据代码示例

```
// Cookie存储复杂数据
setcookie('user[name]', 'tom');
setcookie('user[age]', 30);
// 输出结果：array{["user"]=>array{["name"]=>"tom"["age"]=>"30"}}
var_dump($_COOKIE);
```

4.3.3 Cookie的高级应用

Cookie高级使用：通过第3个参数设置Cookie的生命周期，通过第4个参数设置Cookie的访问路径，通过第5个参数设置访问网站。

```
// 第3个参数设置Cookie有效期
setcookie('name', 'value', time() + 1800);          // 30分钟后过期
setcookie('name', 'value', time() + 60 * 60 *24);   // 一天后过期
setcookie('name', '', time() - 1);                  // 立即过期（相当于删除Cookie）
// 第4个参数设置Cookie的访问路径
setcookie('name', 'value', 0, '/');                 // 当前整个网站都可访问
// 第5个参数设置Cookie的有效域名
setcookie('name', 'value', 0, '/', '.com');         // 所有.com的网站都可以访问
```

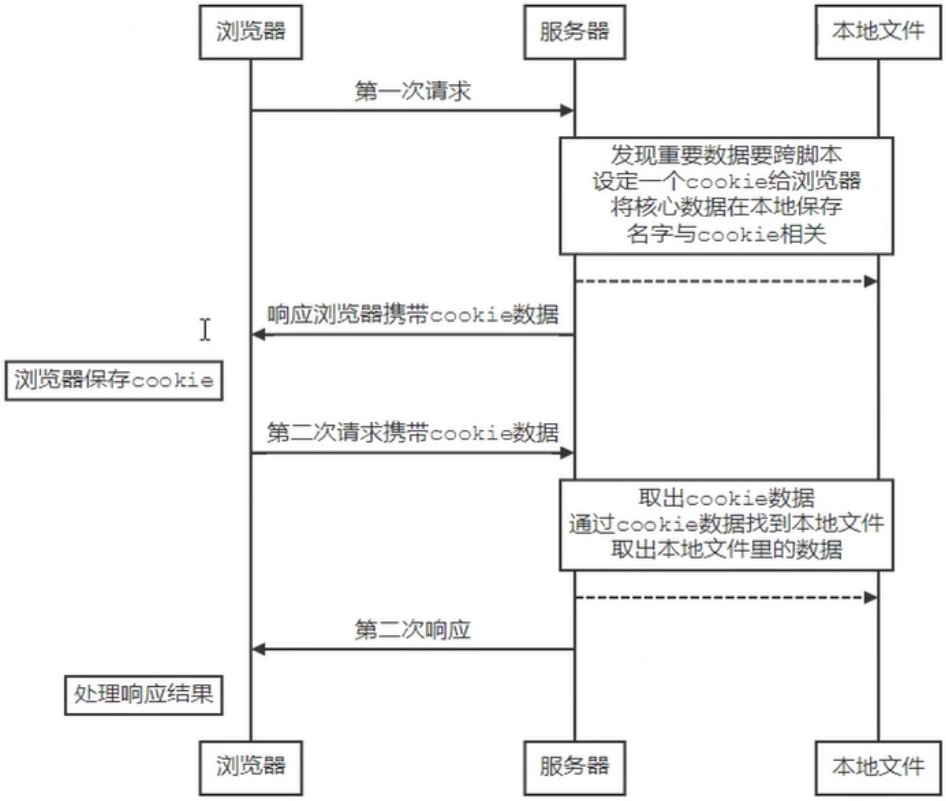
总结：Cookie存储在浏览器上。

4.3.4 Session简介

思考:有了cookie技术之后，服务器可以实现多个不同的脚本文件共享该数据了。但是数据是通过浏览器来存储的，安全吗?
引入:浏览器上存储的cookie数据是用户可见的，而且js文件可以从浏览器读取所有cookie，所以显然是不安全的。而且如果数据量大的话，浏览器在每次请求服务器时都会带着cookie，这样也是浪费资源的。所以，如果有一些核心数据想要跨脚本共享的话，就要用到session技术。

Session存储在服务器端，实现数据跨脚本共享的会话技术，Session技术的实现依赖于Cookie技术。

Session实现原理：



思考：session技术的本质是识别操作文件，问题是服务器那么多文件，系统是如何保证文件名字不冲突的呢?

原因：session文件的名字对应的cookie是有一套内部机制生成32位随机字符串，冲突的可能性几乎不可能。

1、session的使用必须开启session:利用PHP函数session_start()， session_start完成了几件事情

（1）sessionlD获取,sessionid就是session文件名字的核心部分

- 读取cookie:固定cookie名字，PHPSESSID，读取到了就直接使用
- 自动生成:没有读到cookie，就自动生成一个sessionlD，同时设置cookie，把该id拼凑到响应头中

（2）寻找sessionid对应的session文件

- 没有:创建对应文件
- 有:打开文件，读取里面的内容

（3）初始化$SESSION系统变量

- 没有数据:数组为空
- 有数据:存放到$SESSION中

2、session操作继续，不管$_SESSION中是否已经有数据了，我们都可以往$__SESSION中读写数据。

3、session_start开启后，一直在等待:等待脚本执行结束,session系统自动将$_SESSION中的数据取出来，然后写入到session文件。session文件保存在**D:\phpstudy_pro\Extensions\tmp\tmp**

4、销毁session:即删除当前session文件，同时关闭session系统。session文件默认的是不会自动删除的，除非程序主动删除:可以使用session_desroy()函数来删除session文件

示例：

```php
<?php
#访问$_SESSION
var_dump($_SESSION); #错误不能直接访问，没有初始化
#激活session
session_start();
#存储数据到session中：往$_SESSION中添加数据
$_SESSION['name'] = 'value';
$_SESSION['user'] = array(
    'name' => 'Tom',
    'age' => 18
);
var_dump($_SESSION);
session_destroy();
```

4.3.5 Session的基本使用

目标：掌握Session的基本使用，能够实现Session的开启、添加数据、删除数据等操作。

```php
session_start();                    // 开启Session
$_SESSION['name'] = 'tom';          // 向Session添加数据（字符串）
$_SESSION['id'] = [1,2,3];          // 向Session添加数据（数组）
unset($_SESSION['name']);           // 删除单个数据
$_SESSION = [];                     // 删除所有数据
session_destroy();                  // 结束当前会话
```

4.3.6 Session的配置

php.ini中关于Session的配置项：

配置项	含义
session.name	指定Cookie的名字，只能由字母和数字组成，默认为PHPSESSID
session.save_path	读取或设置当前会话文件的保存路径，默认为“C:\Windows\Temp”
session.auto_start	指定是否在请求开始时自动启动一个会话，默认为0（不启动）
session.cookie_lifetime	以秒数指定发送到浏览器的Cookie生命周期，默认为0（直到关闭浏览器）
session.cookie_path	指定要设定会话Cookie的路径，默认为“/”
session.cookie_domain	指定要设定会话Cookie的域名，默认为无
session.cookie_secure	指定是否仅通过安全连接发送Cookie，默认为off
session.cookie_httponly	指定是否仅通过HTTP访问Cookie，默认为off

通过session_start()函数配置Session：

```php
session_start(['name' => 'MySESSID']);
```

注意：session_start()函数对配置项的修改只在PHP脚本的运行周期内有效，不影响php.ini中的原有设置。

4.4图像处理

4.4.1 开启GD扩展

目标：掌握GD扩展的开启方法，能够动手完成GD扩展的开启

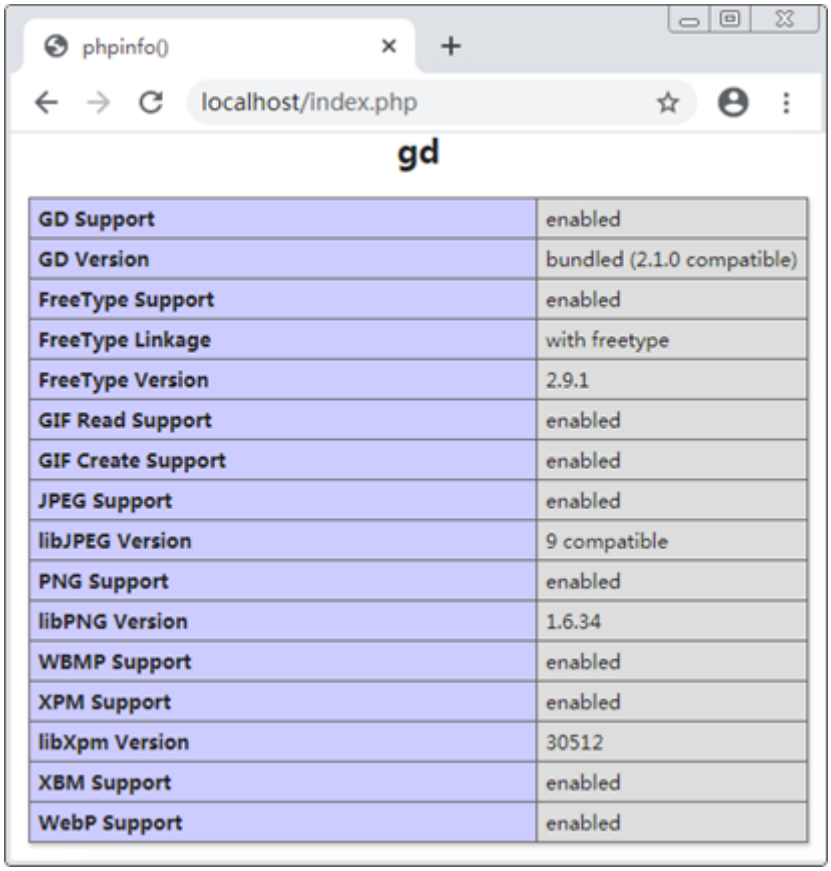
开启GD扩展：在PHP的配置文件php.ini，找到

“;extension=gd2”，去掉分号“;”。

```
extension=gd2
```

查看开启结果：重启Apache使配置生效，通过

phpinfo()函数查看GD库是否开启成功。



4.4.2 常用图像处理函数

目标：掌握常用图像处理函数的使用，能够实现图像的处理。

函数	作用
imagecreatetruecolor()	创建指定宽高的真彩色空白画布图像
imagecolorallocate()	为画布分配颜色
imagefill()	为画布填充颜色
imagestring()	将字符串写入到画布中

函数	作用
imagettftext()	将文本写入到画布中
imageline()	在画布中绘制直线
imagecreatefromjpeg()	创建JPEG格式的图像

函数	作用
imagecreatefrompng()	创建PNG格式的图像
imagecopymerge()	合并两个图片
imagecopyresampled()	复制一部分图像到目标图像中
imagepng()	输出PNG格式的图像
imagejpeg()	输出JPEG格式的图像
imagedestroy()	销毁图像
getimagesize()	获取图像的大小

4.4.3【案例】制作验证码

目标：掌握制作验证码案例的实现，能够动手完成验证码功能的代码编写

需求分析

在有数据输入的功能开发中，如果恶意向服务器提交数据，那么网站会产生大量的脏数据，使用验证码成为一种防御手段。通过图像处理函数实现制作验证码的功能。

1.验证码制作流程:

- 制作画布:imagecreatetruecolor
- 填充背景色: imagecolorallocate分配颜色，imagefill填充颜色
- 写入内容:imagestring写简单内容，imagettftext写入字体文字
- 增加干扰: imageline增加线段，imagestring增加其他符号
- 保存图片:imagepng输出或者保存图片

```php
<?php
function getCaptcha($width, $height, $lines = 10, $length = 4)
{
    //创建画布
    $img = imagecreatetruecolor($width, $height);
    //2.填充背景色
    //2.1给画布分配背景颜色，RGB
    $bg_color = imagecolorallocate($img, mt_rand(200,255),  mt_rand(200,255), mt_rand(200,255));
    //2.2填充颜色,从0,0坐标处开始上色，自动渲染相同颜色像素点
    imagefill($img, 0, 0, $bg_color);

    //3.写入字符串
    $str_arr = range('A', 'Z');
    shuffle($str_arr);
    $captcha = '';
    for($i = 0; $i < $length; $i++){
        $captcha .= $str_arr[$i];
```

```
    }
    //3.1给要写入的字符串分配颜色
    $str_color = imagecolorallocate($img, mt_rand(0, 80), mt_rand(0, 80),
mt_rand(0, 80));
    //3.2写入文字
    imagestring($img, 5, 30, 10, $captcha, $str_color);

    //3.3增加干扰线
    for ($i = 0; $i < $lines; $i++) {
        $line_color = imagecolorallocate($img, mt_rand(100, 160), mt_rand(100,
160), mt_rand(100, 160));
        imageline($img, mt_rand(0, $width), mt_rand(0, $height), mt_rand(0,
$width), mt_rand(0, $height), $line_color);
    }
    //4.输出图片
    header('content-type:image/png');
    imagepng($img);
    //5.销毁资源
    imagedestroy($img);
}
getCaptcha(100, 30);
```

其中:

```
bool imagestring ( resource $image , int $font , int $x , int $y , string $s , int $col )
```

imagestring()用col颜色将字符串s画到image所代表的图像的x,y坐标处(这是字符串左上角坐标，整幅图像的左上角为0，0)，如果font是1，2，3，4或5，则使用内置字体。

4.4.4 【案例】生成水印图片

目标：掌握生成水印图片案例的实现，能够动手完成生成水印图片功能的代码编写

需求分析:

在项目开发时，考虑到网站中所上传的图片不被他人盗用，对用户上传的图片进行添加水印标记的处理，根据原图的路径，找到图片资源，完成添加水印的功能。

```
<?php
//封装水印图制作函数
/*
*制作水印图
*@param1 string $image,要添加水印的图片地址
*@param2 string $path,水印图制作完成后存储路径
*@param3 string $water,水印图图片地址
*@param4 int $pct=50,透明度，默认为50
 @return string 新生成图片的名字
*/
function getWaterMark($image, $path, $water, $pct = 50){
    // 1.打开原图资源
    $dst = imagecreatefromjpeg($image);
    // 2.打开水印资源
    $src = imagecreatefrompng($water);
    // 3.获取水印信息
    $dst_info = getimagesize($image);
    print_r($dst_info);
```

```
    $src_info = getimagesize($water);
    //4.采样合并
    imagecopymerge($dst, $src, $dst_info[0]-$src_info[0], $dst_info[1]-
$src_info[1], 0, 0, $src_info[0], $src_info[1], $pct);
    $dst_info = pathinfo($image);
    $water_file = $dst_info['filename'] . '_water.' . $dst_info['extension'];
    //5.保存输出
    imagejpeg($dst, $path . '' . $water_file);
    //6.销毁资源
    imagedestroy($dst);
    imagedestroy($src);
}

getWaterMark('desktop.jpg', './', 'water.png');
```

4.4.5 【案例】制作缩略图

需求分析:

在项目开发中，为了解决用户上传图片大小不一的问题，需要对用户上传的图片进行相应的处理，可以让其在指定大小的地方显示。制作缩略图，是将原图放到一个固定大小的图片资源里，形成一张新的图片，称之为缩略图。通过图像处理函数实现制作缩略图的功能。

```
<?php

function thumb($image, $path, $width = 100, $height = 100){
    $src = imagecreatefromjpeg($image);                 // 原图资源
    $dst = imagecreatetruecolor($width, $height);    // 创建缩略图画布
    $bg_color = imagecolorallocate($dst, 255, 255, 255);
    imagefill($dst, 0, 0, $bg_color);

    $src_info = getimagesize($image);
    $src_c = $src_info[0] / $src_info[1];
    $thu_c = $width / $height;
    if($src_c > $thu_c){
        $thu_w = $width;
        $thu_h = ceil($thu_w / $src_c);
    }else{
        $thu_h = $height;
        $thu_w = ceil($thu_h * $src_c);
    }

    $thu_x = ceil(abs($thu_w - $width) / 2);
    $thu_y = ceil(abs($thu_h - $height) / 2);

    imagecopyresampled($dst, $src, $thu_x, $thu_y, 0, 0, $thu_w, $thu_h,
        $src_info[0], $src_info[1]);
    $src_info = pathinfo($image);
    $thumb_file = $src_info['filename'] . '_thumb.' . $src_info['extension'];
    imagejpeg($dst, $path . '' . $thumb_file);
    imagedestroy($dst);
    imagedestroy($src);
}

thumb('desktop.jpg', './');

?>
```

4.5目录和文件操作

4.5.1 目录操作

目标：掌握目录操作，能够实现目录的创建、删除、重命名以及读取操作

1.创建目录

使用mkdir()函数创建目录，该函数执行成功返回true，失败返回false。

语法格式：

```
bool mkdir( string $pathname [, int $mode = 0777[, bool $recursive = false[,
resource $context ]]] )
```

- $pathname表示要创建的目录地址，地址的格式可以是绝对路径也可以是相对路径；
- $mode指定目录的访问权限（用于Linux环境），默认为0777；
- $recursive指定是否递归创建目录，默认为false。

  使用示例

```
mkdir('upload');
```

2.删除目录

使用rmdir()函数删除目录，该函数执行成功返回true，失败返回false

语法格式

```
bool rmdir( string $dirname [, resource $context ] )
```

使用示例

```
rmdir('upload');
```

注意：删除的目录不存在，会删除失败，并提示Warning错误。

删除非空目录时，同样也会删除失败并提示Warning错误，只有先清空里面的文件，才能够删除目录。

3.重命名目录

使用rename()函数实现目录或文件的重命名，该函数执行成功返回true，失败返回false。

语法格式

```
bool rename( string $oldname , string $newname[, resource $context ] )
```

- $oldname表示要重命名的目录
- $newname表示新的目录名称

rename()**函数使用示例**

```
mkdir('upload');
rename('upload', 'uploads');
```

4.读取目录

读取目录两种方式：一种是使用scandir()函数获取目录下的所有文件名；另外一种方式是使用opendir()函数获取资源，然后使用readdir()函数进行访问。

scandir()函数语法格式

```
bool scandir( string $directory [, int $order, resource $context ] )
```

$directory表示要查看的目录

$order规定排列排序，默认是0，表示按字母升序排列

使用scandir()函数查看当前目录下的所有内容：

scandir()函数使用示例

```
$dir_info = scandir('./');
foreach ($dir_info as $file) {
    echo $file . '<br>';
}
```

4.读取目录

opendir()函数用于打开一个目录句柄，readdir()函数从目录句柄中读取条目。

opendir*()函数语法格式**

```
resource opendir( string $path [, resource $context ] )
```

- $path表示要打开的目录路径，
- 函数执行成功返回目录句柄的resource，失败返回false

readdir*()函数语法格式**

```
resource readdir( [ resource $dir_handle ] )
```

- $dir_handle表示已经打开的目录句柄资源，
- 函数执行成功返回文件名称，失败返回false

使用opendir()函数和readdir()函数读取目录中的内容：

```
$resource = opendir('./');
$file = '';
while ($file = readdir($resource)) {
    echo $file . '<br>';
}
closedir($resource);
```

其他目录函数：

函数	作用
bool is_dir(string $filename)	判断给定的文件名是否是一个目录
getcwd()	若成功会返回当前目录，失败则返回false
rewinddir(resource $dir_handle)	将打开的资源指针重置到目录的开头
bool chdir(string $directory)	改变当前的目录，若成功会返回true，失败则返回false

路径的有效性：为了保证代码的严谨性，通常会使用is_dir()函数来判断。

4.5.2 文件操作

目标：掌握文件操作，能够实现文件的打开、删除、修改、重命名以及读取操作

1.打开文件

fopen()函数：打开文件，该函数执行成功后，返回资源类型的文件指针，用于其他操作。

fopen*()函数语法格式**

```
resource fopen( string $filename, string $mode [, bool $use_include_path
= false [,resource $context ]] )
```

- $filename表示打开的文件路径，可以是本地文件，也可以是HTTP或FTP协议的URL地址；
- $mode表示文件打开的模式。

fopen()函数常用的文件打开模式：

模式	说明
r	只读方式打开，将文件指针指向文件头
r+	读写方式打开，将文件指针指向文件头
w	写入方式打开，将文件指针指向文件头并将文件大小截为0
w+	读写方式打开，将文件指针指向文件头并将文件大小截为0
a	写入方式打开，将文件指针指向末尾
a+	读写方式打开，将文件指针指向末尾
x	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则fopen()调用失败，返回false，并生成E_WARNING级别的错误信息
x+	创建并以读写方式打开，其他行为和“x”相同

fopen*()函数使用示例**

```
// 只读方式打开，不会创建文件，文件不存在会提示warning信息
$f1 = fopen('test1.html', 'r');
$f2 = fopen('test2.html', 'w');        // 写入方式打开，文件不存在自动创建
fclose($f1);
fclose($f2);
```

2.删除文件

unlink()函数：删除文件，该函数执行成功后成功返回值为true，失败返回false。

unlink()***函数语法格式****

```
bool unlink( string $filename [, resource $context ] )
```

unlink()***函数使用示例****

```
unlink('./test2.html');
```

3.修改文件

fwrite()函数：修改文件内容。

fwrite***()函数语法格式****

```
int fwrite( resource $handle , string $string [, int $length ] )
```

- $handle表示文件指针
- $string表示要写入的字符串
- $length表示指定写入的字节数，如果省略，表示写入整个字符串。

```
$f3 = fopen('test3.html', 'w');
fwrite($f3, '<html><body>Hello world<body></html>');
fclose($f3);
```

5.读取文件

fread()函数：读取文件内容，返回读取到的内容，读取失败返回false。

fread()函数语法格式

```
string fread( resource $handle , int $length )
```

- $handle表示文件指针
- $length指定读取的字节数

fread()函数使用示例

```
$filename = 'test3.html';
$f3 = fopen($filename, 'r');
$data = fread($f3, filesize($filename));
echo $data;       // 输出内容: <html><body>Hello world<body></html>
fclose($f3);
```

6.file_get_contents()和file_put_contents()函数

（1）file_get_contents()函数：将文件的内容全部读取到一个字符串中。

file_get_contents***()函数语法格式****

```
string file_get_contents( string $filename [, bool $use_include_path = false
[, resource $context [, int $offset = 0 [, int $maxlen ]]]] )
```

$filename指定要读取的文件路径

（2）file_put_contents()函数：写入内容，成功返回写入到文件内数据的字节数，失败返回false。

file_put_contents**()函数语法格式****

```
int file_put_contents( string $filename , mix $data [, int $flags = 0
[, resource $context ]] )
```

- $filename指定要写入的文件路径；
- $data指定要写入的内容；
- $flags指定写入选项，使用常量FILE_APPEND表示追加写入。

函数使用示例

```
$filename = 'test3.html';
$content = file_get_contents($filename);
echo $content;   // 输出内容: <html><body>Hello world<body></html>
// 文件内容不会改变，覆盖原文件内容
$str = '<html><body>Hello world<body></html>';
file_put_contents($filename, $str);
// 追加内容
file_put_contents($filename, $str, FILE_APPEND);
```

4.5.3 【案例】递归遍历目录

需求分析

掌握递归遍历目录案例，能够编写代码完成目录的递归遍历

递归：指一个函数在其函数体内调用自身的过程，这种函数称为递归函数。

递归需要有递归点和递归出口。

- 递归点：当需要解决的问题与当前函数解决的问题相同时，进行递归调用。
- 递归出口：当没有再需要解决的问题时，结束递归。

以遍历目录下的所有文件为例，获取根目录的文件列表后，从列表中找到目录的地址，根据该地址再次调用函数，获取子目录下的文件。

```php
<?php

function dirfile($dir)
{
    if (!is_dir($dir)) {
        return;
    }
    $files = scandir($dir);
    foreach ($files as $file) {
        if ($file == '.' || $file == '..') {
            continue;
        }
        $tmp_dir = $dir . '/' . $file;
        echo $tmp_dir, '<br>';
        if (is_dir($tmp_dir)) {
            dirfile($tmp_dir);
        }
```

```
    }
}
dirfile('.');    // 调用函数，递归当前目录
```

4.6表单传值

目标：熟悉表单传值方式，能够利用GET方式和POST方式进行表单传值

4.6.1 表单传值方式

表单传值：实现动态网站非常重要的一步，不仅能够实现用户的数据提交，还能帮助用户实现动态数据定制。

表单传值示例代码

```
<form action="表单提交地址" method="post">
    <!-- 表单内容 -->
</form>
```

4.6.2 接收表单数据

熟悉表单数据的接收，能够利用超全局变量接收表单数据。

接收表单数据：在服务端使用超全局变量接收表单数据。

变量名	说明
$_GET	接收GET方式提交的数据
$_POST	接收POST方式提交的数据
$_REQUEST	接收GET、POST以及Cookie数据

4.6.3 表单提交数组值

熟悉如何使用表单提交数组值，能够实现表单中的复选框数据的提交和接收

表单提交数组：将表单相同元素的name设置成数组的形式即可，表单将会以数组的形式提交。

设置表单示例代码

(1)index.html代码

```
<form action="index.php" method="post">
    <input type="checkbox" name="hobby[]" value="basketball" />篮球
    <input type="checkbox" name="hobby[]" value="football" />足球
    <input type="checkbox" name="hobby[]" value="vollyball" />排球
    <input type="submit" value="提交">
</form>
```

接收表单提交的数组数据：当选择“篮球”和“足球”两个选项时，接收并输出复选框的值。

（2）index.php

```
var_dump($_POST);
// 输出结果
array(1){
    ["hobby"]=>array(2){
        [0]=>string(10) "basketball"
        [1]=>string(8) "football"
    }
}
```

案例：用户登录

1、创建login.html，显示用户登录的界面。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="login.php" method="post">
      用户名:<input type="text" name="user"><br>
      密码:<input type="password" name="pwd"><br>
        <input type="submit" value="登录">
    </form>
</body>
</html>
```

2、编写login.php，接收用户登录的表单。

```
<?php
session_start();
if ($_POST) {
    $user = isset($_POST['user']) ? trim($_POST['user']) : '';
    $pwd = isset($_POST['pwd']) ? trim($_POST['pwd']) : '';
    $data = ['user' => 'Tom', 'pwd' => '123456'];
    if (($user == $data['user']) && ($pwd == $data['pwd'])) {
        $_SESSION['user']=$data['user'];
        header('Location:index.php');
        exit;
    } else {
        echo '用户名或者密码输入不正确，登录失败';
    }
```

```
}
require './login.html';
```

3、创建index.php。

```
<?php
session_start();
if(isset($_SESSION['user'])){
    echo '当前登录用户：'.$_SESSION['user'].' 。';
    echo '<a href="logout.php">退出</a>';
}else{
    header('Location:login.html');
    exit;
}
```

4、创建logout.php，实现退出功能。

```
<?php
session_start();
$_SESSION['user']=[];
session_destroy();
header('Location:login.html');
```