WEB应用开发——后端

PHP基础

目录

练习1——PHP练习：计算阶乘

练习要求

编写一段PHP代码，计算1到10的阶乘之和，最后直接使用echo输出结果即可

代码：

```php
<?php
$sum = 0; // 初始化总和变量，用于存储1到10的阶乘之和
// 外层循环：计算1到10每个数字的阶乘并累加
for($i=1; $i<=10; $i++){
    $factorial = 1; // 初始化阶乘变量，每个数字的阶乘计算从1开始
    // 内层循环：计算当前数字i的阶乘
    for($j=1; $j<=$i; $j++){
        $factorial *= $j; // 累乘计算阶乘值
    }
    $sum += $factorial; // 将当前数字的阶乘值加到总和中
}
echo $sum; // 输出1到10的阶乘之和
?>
```

输出：

```
4037913
```

练习2——PHP练习：单词排序

练习要求

编写一段PHP代码，将字符串“hello world my name is php”中的所有单词按照长度排序并按照如下方式输出:

提示:
拆分字符串使用explode()
将单词数组拼接成字符串用implode()
自定义排序使用usort()
遍历数组使用foreach()或for循环
换行打印

按照从小到大次序输出:
1 my
2 is
3 php
4 name
5 hello
6 world
排序后重新拼接字符串:
my is php name hello world

练习参考代码

编写一段PHP代码，将字符串“hello world my name is php”中的所有单词按照长度排序并按照如下方式输出：

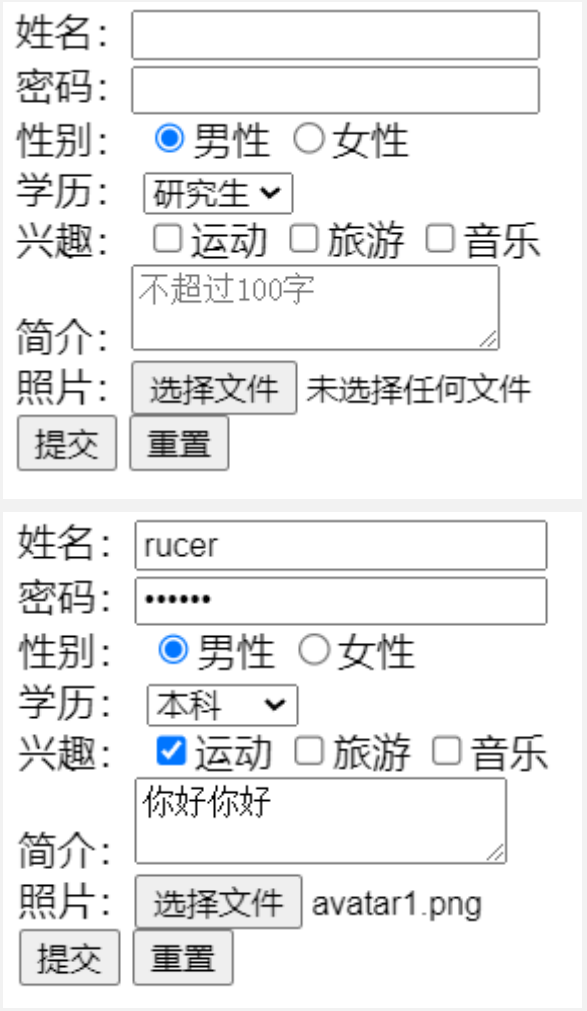
代码：

```php
<?php
// 定义一个包含多个单词的字符串
$str = "hello world my name is php";
// 使用explode函数将字符串按空格分割成单词数组
$words = explode(" ", $str);
// 使用usort函数对数组进行自定义排序
// 这里使用匿名函数比较两个单词的长度，按长度从小到大排序
usort($words, function($a, $b){
    return strlen($a) - strlen($b);
});
// 输出排序后的标题
echo "按照从小到大次序输出：<br/>";
// 遍历数组并按指定格式打印
// 使用foreach循环遍历排序后的单词数组
// $index是当前元素的索引，$word是当前单词
foreach ($words as $index => $word) {
    // 输出序号(索引+1)和单词，每个单词占一行
    echo ($index + 1) . " " . $word . "<br/>";
}
// 输出拼接后的单词标题
echo "排序后重新拼接字符串：<br/>";
// 使用implode函数将数组中的单词用空格连接成字符串并输出
echo implode(" ", $words);
?>
```

练习3——单文件上传

练习要求

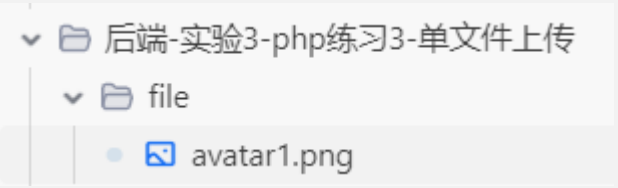
给你一个写好的HTML文件，请你编写一段PHP代码，能够接受HTML传来的表单数据，并使用echo打印，如果传输了图片，则需要打印图片的信息



点击提交

输出结果1：跳转到submit.php页面，并且显示如下

输出结果2：根目录的file文件夹下应该有上传到后端的图像



练习参考代码

给你一个写好的HTML文件，请你编写一段PHP代码，能够接受HTML传来的表单数据，并使用echo打印，如果传输了图片，则需要打印图片的信息

```php
<?php
/*
 * 使用可变变量+foreach循环
 * 将$_POST数组中的所有键值对转换为变量
 * 例如：$_POST['name'] => $name
 */
foreach ($_POST as $key => $value) {
    $$key = $value;
}
```

```php
// 输出表单提交的文本字段值
echo "name: " . $name . "<br/>";          // 显示姓名
echo "password: " . $password . "<br/>"; // 显示密码(注意：实际应用中不应明文显示密码)
echo "sex: " . $sex . "<br/>";            // 显示性别
echo "education: " . $education . "<br/>"; // 显示教育程度
echo "hobby: " . $hobby . "<br/>";        // 显示兴趣爱好
echo "intro: " . $intro . "<br/>";        // 显示个人介绍
```

练习参考代码

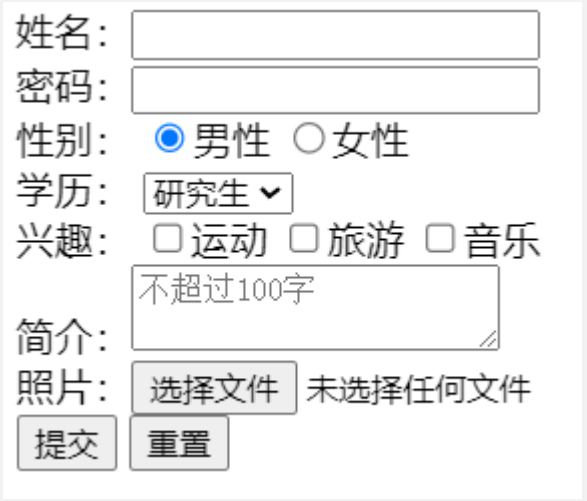
给你一个写好的HTML文件，请你编写一段PHP代码，能够接受HTML传来的表单数据，并使用echo打印，如果传输了图片，则需要打印图片的信息

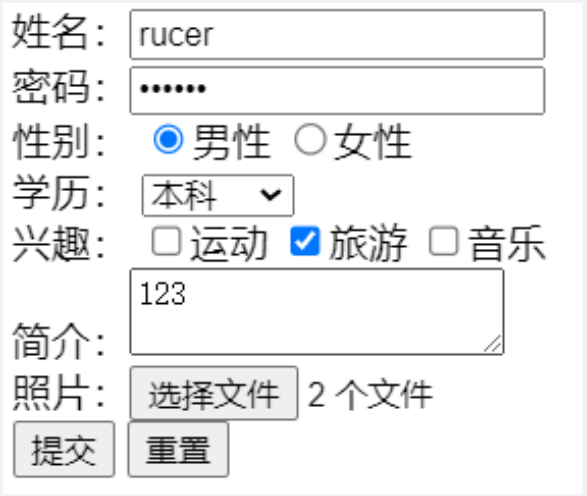
```php
/*
 * 文件上传处理部分
 * 检查是否有文件被上传且没有错误
 */
if (is_uploaded_file($_FILES["photo"]["tmp_name"]) && $_FILES["photo"]["error"] == 0) {
    // 显示上传文件的基本信息
    echo "Upload: " . $_FILES["photo"]["name"] . "<br/>";    // 原始文件名
    echo "Type: " . $_FILES["photo"]["type"] . "<br/>";      // 文件MIME类型
    echo "Size: " . ($_FILES["photo"]["size"] / 1024) . " Kb<br/>"; // 文件大小(KB)
    echo "Stored in: " . $_FILES["photo"]["tmp_name"] . "<br/>"; // 临时存储路径

    // 文件路径处理
    $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）

    /*
    * 文件移动操作
    * 将临时文件移动到目标目录
    */
    if (move_uploaded_file($temporary_path, $absolute_path)) {
        echo "文件保存成功: " . $relative_path . "<br/>";
        // 显示上传的图片(限制最大宽度为300px)
        echo "<img src=\"$relative_path\" style=\"max-width: 300px; margin: 10px;\"/><br/>";
    } else {
        echo "文件保存失败<br/>"; // 移动文件失败时的提示
    }
}
```

练习4——多文件上传

练习要求

给你一个写好的HTML文件，请你编写一段PHP代码，能够接受HTML传来的表单数据，并使用echo打印。如果传输了多张图片，则需要依次打印图片的信息





点击提交

输出结果1：跳转到submit.php页面，并且显示如下

name: rucer
password: 123456
sex: male
education: undergraduate
hobby: travel
intro: 123

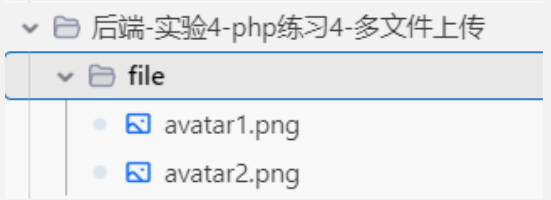
文件 1:

Upload: avatar1.png
Type: image/png
Size: 24.958984375 Kb
Stored in: /tmp/phpPR7ula
文件保存成功: /file/avatar1.png

文件 2:

Upload: avatar2.png
Type: image/png
Size: 28.4970703125 Kb
Stored in: /tmp/php0oh4g2
文件保存成功: /file/avatar2.png

输出结果2：根目录的file文件夹下应该有上传到后端的图像



练习参考代码

```php
/*
 * 多文件上传处理部分
 * 检查是否有名为"photo"的文件上传字段
 */
if (!empty($_FILES["photo"])) {
    /*
     * 遍历所有上传的文件
     * $_FILES["photo"]["error"]数组包含每个文件的上传状态码
     */
    foreach ($_FILES["photo"]["error"] as $key => $error) {
        // 检查当前文件是否上传成功(UPLOAD_ERR_OK表示成功)
        if ($error == UPLOAD_ERR_OK) {
            // 显示文件上传信息标题
            echo "<h3>文件 " . ($key + 1) . ":</h3>";

            // 显示文件详细信息
            echo "Upload: " . $_FILES["photo"]["name"][$key] . "<br/>"; // 原始文件名
            echo "Type: " . $_FILES["photo"]["type"][$key] . "<br/>";   // 文件MIME类型
            echo "Size: " . ($_FILES["photo"]["size"][$key] / 1024) . " Kb<br/>"; // 文件大小(KB)
            echo "Stored in: " . $_FILES["photo"]["tmp_name"][$key] . "<br/>"; // 临时存储路径
```

练习参考代码

```php
            /*
             * 文件路径处理
             * 准备文件移动的源路径和目标路径
             */
            $temporary_path = $_FILES["photo"]["tmp_name"][$key]; // 临时文件路径
            $relative_path = "/file/" . $_FILES["photo"]["name"][$key]; // 相对路径(web可访问)
            $absolute_path = $_SERVER["DOCUMENT_ROOT"] . $relative_path; // 绝对路径(服务器文件系统)

            /*
             * 文件移动操作
             * 将临时文件移动到目标目录
             */
            if (move_uploaded_file($temporary_path, $absolute_path)) {
                echo "文件保存成功: " . $relative_path . "<br/>";
                // 显示上传的图片(限制最大宽度为300px)
                echo "<img src=\"$relative_path\" style=\"max-width: 300px; margin: 10px;\"/><br/>";
            } else {
                echo "文件保存失败<br/>"; // 移动文件失败时的提示
            }
        } else {
            // 文件上传出错时的处理
            echo "文件 " . ($key + 1) . " 上传出错: " . $error . "<br/>";
        }
        echo "<hr/>"; // 分隔线，区分不同文件的上传结果
    }
}
```

示例1——微人大登录界面（PHP）

微人大登录界面（PHP）

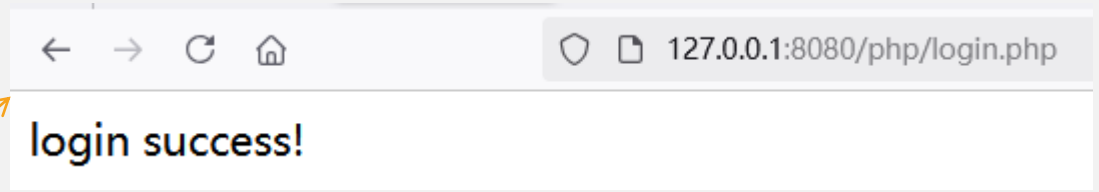
- 在使用JS之后，我们将登录界面变成“动态”界面了，但是这仍然不够。这是因为JS存在以下问题：
  - JS运行在用户浏览器中，负责交互性和动态内容（如表单验证、动画、DOM操作），无法直接访问服务器资源（如数据库、文件系统）
  - JS不能持久化存储敏感数据
  - JS代码对用户完全透明，所有逻辑可被查看或篡改（如通过浏览器开发者工具）

- 因此我们需要引入PHP来做后端，以保证web应用的安全性

微人大登录界面（PHP）

- 在这个示例中，我们将之前的登录验证逻辑从由JS负责改为由PHP负责
- 效果展示：

验证成功则跳转并显示login success



验证则不跳转，继续显示登陆页面

微人大登录界面（PHP

修改HTML代码中form的action和method属性

```
<form action="php/login.php" method="post">
```

login.php中先接收传来的参数

```php
<?php
// 将POST请求中的所有参数转换为变量
// 例如：$_POST['name'] 会变成 $name 变量
foreach($_POST as $key=>$val) {
    $$key = $val;
}
```

创建验证码对照表

```php
// 验证码对照表，数字索引对应验证码字符串
$captcha = [
    0=>"pupr", 1=>"khrb", 2=>"huks",
    3=>"egwb", 4=>"ekdv", 5=>"khns",
    6=>"wuxc", 7=>"tcyk", 8=>"nupf",
    9=>"brpk"
];
```

检查用户输入

```php
// 检查用户输入的验证码是否正确
if ($code === $captcha[$id]) {
    // 验证码正确后，检查用户名和密码
    // 这里硬编码了用户名admin和密码123456作为示例
    if ($name === 'admin' and $passwd === "123456") {
        // 登录成功，返回成功消息
        echo "login success!";
    }
} else {
    // 验证码错误，重定向回登录页面
    echo "<script>location.href='../index.html'</script>";
    // 终止脚本执行
    die();
}
```

示例2——微人大登录界面（SESSION）

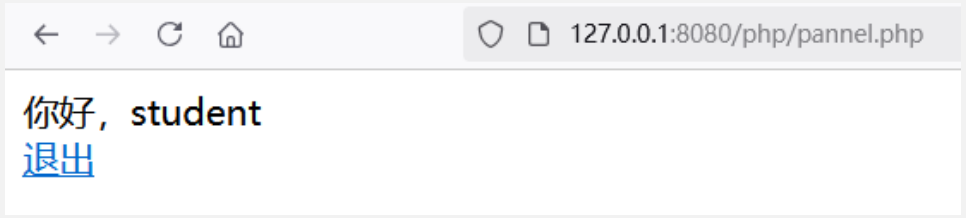
微人大登录界面（SESSION）

- 上一个示例中，我们为微人大登录页面搭建了简单的后端。但是新的问题接踵而至：我们一旦刷新网页，登录状态就重置了，需要重新登录。如何保持用户登录呢?

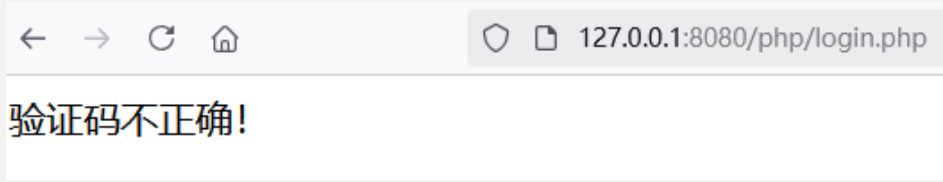
未登录时，输入用户名密码登录

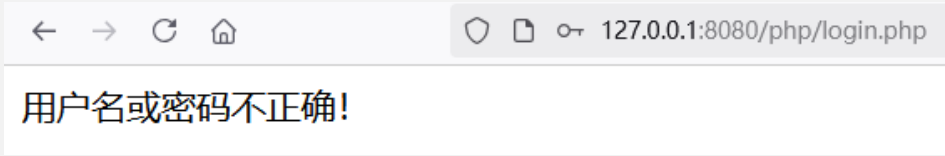
验证成功则跳转并显示当前role

点击退出回到登录页



验证失败会有相应的提示



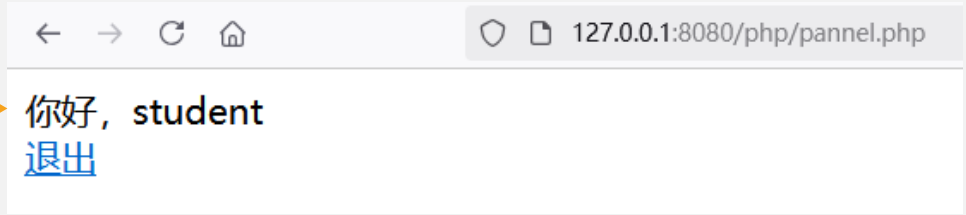


微人大登录界面（SESSION）

- 上一个示例中，我们为微人大登录页面搭建了简单的后端。但是新的问题接踵而至：我们一旦刷新网页，登录状态就重置了，需要重新登录。如何保持用户登录呢?

如果已经登录了

直接跳转



微人大登录界面（SESSION）

index.php中，进入页面时启动会话

```php
<?php
// 启动会话，用于跟踪用户登录状态
session_start();

// 检查用户是否已登录(通过检查session中是否有username字段)
if (isset($_SESSION['username'])) {
    // 如果已登录，重定向到控制面板页面
    header("Location: php/pannel.php");
    // 终止脚本执行，确保重定向立即生效
    exit();
}
?>
```

login.php 登录验证逻辑，验证成功设置角色

```php
// 验证用户输入的验证码是否正确
if ($code == $correct_code) {

    // 检查用户名和密码组合
    if ($username == "admin001" && $password == "admin001")
        $_SESSION["role"] = "admin";  // 管理员角色
    else if ($username == "teacher001" && $password == "teacher001")
        $_SESSION["role"] = "teacher";  // 教师角色
    else if ($username == "student001" && $password == "student001")
        $_SESSION["role"] = "student";  // 学生角色

    // 如果成功设置了角色
    if (isset($_SESSION["role"])) {
        // 存储用户名到session
        $_SESSION["username"] = $username;
        // 设置角色cookie（httponly标志为true，防止XSS攻击）
        setcookie("role", $_SESSION["role"], 0, "/", "", false, true);
        // 重定向到面板页面
        header("Location: pannel.php");
    } else {
        // 用户名或密码不正确
        echo "用户名或密码不正确！";
    }
} else {
    // 验证码不正确
    echo "验证码不正确！";
}
```

微人大登录界面（SESSION）

panel.php中启动会话，并且根据role打印信息

```php
// 启动会话，用于验证用户登录状态
session_start();

// 检查用户是否已登录（session中是否存在username）
if (isset($_SESSION["username"])) {
    // 从cookie获取用户角色 - 存在安全风险：
    // 1. cookie容易被篡改
    // 2. 建议改为从session或数据库获取角色信息
    $role = $_COOKIE["role"];

    // 显示欢迎信息和用户角色
    echo "你好，" . $role . "<br/>";

    // 提供退出登录的链接
    echo "<a href='logout.php'>退出</a>";
} else {
    // 未登录用户显示无权访问提示
    echo "无权访问！";
}
```

logout.php中清除会话中的变量

```php
// 启动会话，允许访问$_SESSION变量
session_start();

// 清除当前会话中所有已注册的变量
// 注意：这不会销毁会话本身，只是清空会话数据
session_unset();

// 完全销毁当前会话
// 这会删除会话数据并终止会话
session_destroy();

// 重定向用户回到首页
// 使用相对路径 "../" 表示上一级目录
header("Location: ../index.php");

// 确保在重定向后立即退出脚本执行
exit();
```

示例3——RUC小喇叭（SESSION）

RUC小喇叭（SESSION）

- 同样的，我们将RUC小喇叭的逻辑功能由JS实现改为由PHP实现，并且引入Session机制

发帖后记录在SESSION中

刷新之后帖子依然存在

RUC小喇叭（SESSION）

index.php中，进入页面时启动会话并从$_SESSION中获取posts数组

```php
<?php
  session_start();

  if(!isset($_SESSION["posts"])) {
    $_SESSION["posts"] = array();
  }

  $posts = $_SESSION["posts"];
```

遍历post数组，并从中获取帖子信息

```php
foreach ($posts as $post) {
  $post_id = $post["post_id"];
  $post_time = $post["post_time"];
  $post_content = $post["content"];
  $post_like_num = $post["like_num"];
  $post_comment_num = $post["comment_num"];
```

根据帖子的发帖时间构建出对应的字符串

```php
// 计算时间差并转换
$currentTime = time();
$diff = $currentTime - $post_time;

if ($diff < 60) {
    $timeAgo = "刚刚";
} elseif ($diff < 3600) {
    $minutes = floor($diff / 60);
    $timeAgo = $minutes . "分钟前";
} elseif ($diff < 86400) {
    $hours = floor($diff / 3600);
    $timeAgo = $hours . "小时前";
} else {
    $days = floor($diff / 86400);
    $timeAgo = $days . "天前";
}
```

根据已有信息，调用之前写好的js函数，展示帖子

```php
echo "<script>createPostContainer('{$post_id}', '{$timeAgo}', '{$post_content}', '{$post_like_num}', '{$post_comment_num}')</script>";
```

RUC小喇叭（SESSION）

index.php中，设置form的action和method属性

```
<div class="input_container">
  <form action="./php/send_post.php" method="post" id="input_form">
    <textarea id="input_textarea" name="content" placeholder="分享你的快乐，分担你的忧愁" required></textarea>
    <div class="btn_row">
      <img id="add_img_btn" src="./img/add_images.svg" />
      <button type="submit"><span>发表</span></button>
    </div>
  </form>
</div>
```

构造数据

```
10    $currentTime = time();
11    $post_id = count($_SESSION["posts"]) + 1;
12    $post_like_num = 0;
13    $post_comment_num = 0;
```

submit_post.php中，先获取前端传来的信息

```
3    session_start();
4
5    // 在session_start()后添加
6    if (!isset($_SESSION["posts"])) {
7        $_SESSION["posts"] = array();
8    }
9
10   if (!isset($_POST['content'])) {
11       echo "前端数据错误：表单中的content字段不能为空";
12   }
13
14   $content = $_POST['content'];
```

将帖子添加到$_SESSION中

```
$post = array(
    "post_id" => $post_id,
    "content" => $content,
    "post_time" => $currentTime,
    "like_num" => $post_like_num,
    "comment_num" => $post_comment_num
);
array_push($_SESSION["posts"], $post);

echo "<script>location.href='../index.php'</script>";
```

WEB应用开发——后端

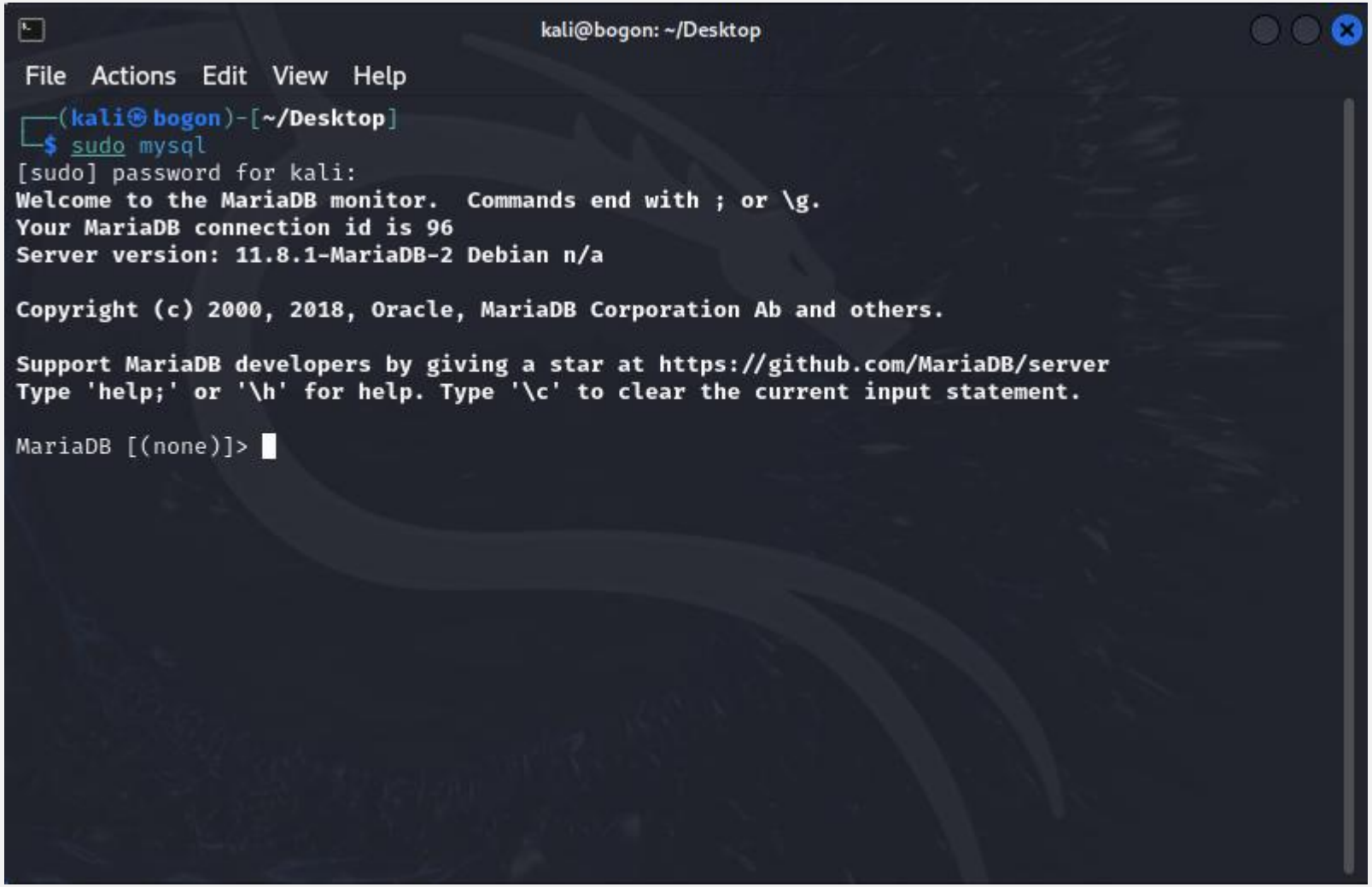
MySQL

目录

MYSQL配置

MySQL配置

课程给的Kali Linux虚拟机中已经默认安装了MySQL，只需要在命令行中输入sudo mysql即可打开mysql终端



MySQL配置

我们通过该方式是以root用户访问mysql，root用户权限太高，现实应用中我们往往会创建新的用户并赋予其有限的权限（比如只能修改某个数据库，只能进行查询操作等）。登录root之后，建议通过以下命令创建新用户：

创建本次上机课需要用到的用户：

```
CREATE USER 'vruc'@'localhost' IDENTIFIED BY 'vruc123';
CREATE USER 'rucwall'@'localhost' IDENTIFIED BY 'rucwall123';
```

创建本次上机课需要用到的数据库：

```
CREATE DATABASE vruc;
CREATE DATABASE rucwall;
```

赋权：

```
GRANT ALL PRIVILEGES ON vruc.* TO 'vruc'@'localhost';
GRANT ALL PRIVILEGES ON rucwall.* TO 'rucwall'@'localhost';
```

以vruc身份登录：

```
sudo mysql -u vruc -p
```

TIPS

tips

web开发中零碎的知识点:

1. 在浏览器中按F12可以打开开发者调试工具

2. 在Kali Linux虚拟机环境中，如果要运行起来某一个php代码，可以使用以下方式:

   ```
   php -S localhost:8080
   ```

   该命令会启动一个小web服务器，该服务器监听localhost（本机）的8080端口。启动之后，我们在浏览器中直接访问网址localhost:8080便可查看到php对应的界面

3. 以某一用户登录mysql可以在终端中输入:

   ```
   sudo mysql -u <你的用户名> -p
   ```

4. 课下有时间可以学一点Linux系统简单的操作，如cd,ls等

练习5——MYSQL练习：基础的增删改查

练习要求

在终端中连接mysql依次完成以下操作:

(1) 创建一个数据库vruc

(2) 在数据库vruc中创建表user，包含以下字段:

字段名	类型
id	int
name	text
role	text
passwd	text

(3) 向表user中加入几条数据:

(1, admin, admin, 123456)
(2, student1, student, stu001)
(3, teacher1, teacher, tea001)
(4, student2, student, stu002)

(4) 查询user中所有的学生

(5) 修改id为4的用户的role为admin

(6) 删除id为4的用户

练习参考代码

(1) 创建一个数据库vruc

```
MariaDB [MyWall]> create database vruc;
Query OK, 1 row affected (0.000 sec)
```

(2) 在数据库vruc中创建表user

```
MariaDB [vruc]> create table user(
    -> id int primary key,
    -> name text,
    -> role text,
    -> passwd text
    -> );
Query OK, 0 rows affected (0.007 sec)
```

(3) 向表user中加入几条数据

```
MariaDB [vruc]> insert into user values(1, 'admin', 'admin', '123456');
Query OK, 1 row affected (0.001 sec)

MariaDB [vruc]> insert into user values(2, 'student1', 'student', 'stu001');
Query OK, 1 row affected (0.001 sec)

MariaDB [vruc]> insert into user values(3, 'teacher1', 'teacher', 'tea001');
Query OK, 1 row affected (0.001 sec)

MariaDB [vruc]> insert into user values(4, 'student2', 'student', 'stu002');
Query OK, 1 row affected (0.002 sec)
```

(4) 查询user中所有的学生

```
MariaDB [vruc]> select * from user where role = 'student';
+----+-----------+---------+--------+
| id | name      | role    | passwd |
+----+-----------+---------+--------+
|  2 | student1 | student | stu001 |
|  4 | student2 | student | stu002 |
+----+-----------+---------+--------+
2 rows in set (0.001 sec)
```

(5) 修改id为4的用户的role为admin

```
MariaDB [vruc]> update user set role = 'admin' where id = 4;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [vruc]> select * from user where id = 4;
+----+-----------+-------+--------+
| id | name      | role  | passwd |
+----+-----------+-------+--------+
|  4 | student2 | admin | stu002 |
+----+-----------+-------+--------+
1 row in set (0.001 sec)
```

练习参考代码

(6) 删除id为4的用户

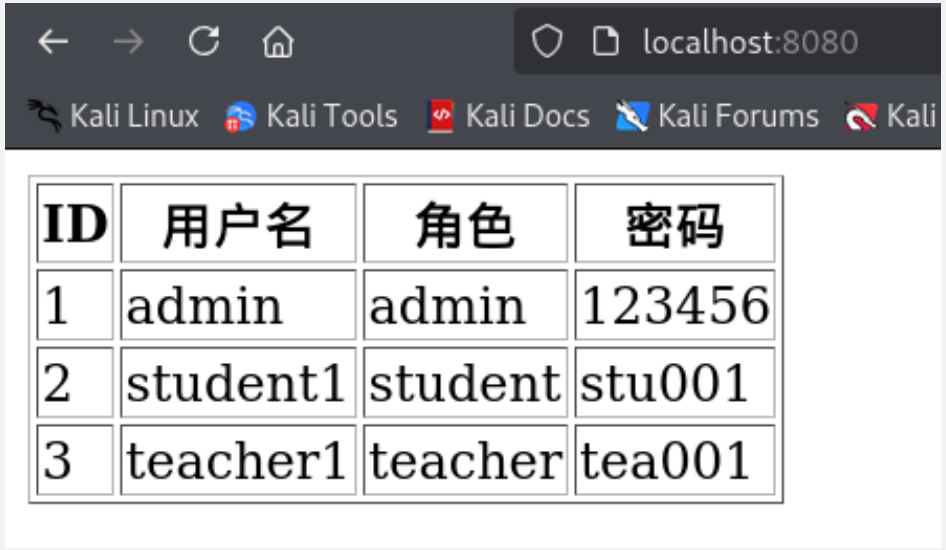
```
MariaDB [vruc]> select * from user;
+----+-----------+----------+---------+
| id | name      | role     | passwd  |
+----+-----------+----------+---------+
|  1 | admin     | admin    | 123456  |
|  2 | student1  | student  | stu001  |
|  3 | teacher1  | teacher  | tea001  |
|  4 | student2  | admin    | stu002  |
+----+-----------+----------+---------+
4 rows in set (0.001 sec)
```

```
MariaDB [vruc]> delete from user where id = 4;
Query OK, 1 row affected (0.001 sec)

MariaDB [vruc]> select * from user;
+----+-----------+----------+---------+
| id | name      | role     | passwd  |
+----+-----------+----------+---------+
|  1 | admin     | admin    | 123456  |
|  2 | student1  | student  | stu001  |
|  3 | teacher1  | teacher  | tea001  |
+----+-----------+----------+---------+
3 rows in set (0.000 sec)
```

练习6——MYSQL练习：使用PHP连接MYSQL

练习要求

请写一个php网页能够从上一个练习中创建的数据库vruc中查询所有的用户，并使用echo打印出他们的信息：



ID	用户名	角色	密码
1	admin	admin	123456
2	student1	student	stu001
3	teacher1	teacher	tea001

练习参考代码

```php
<?php
// 数据库连接配置
$servername = "127.0.0.1:3306";  // MySQL服务器地址和端口
$username = "root";              // 数据库用户名
$passwd = "kali";                // 数据库密码
$database = "vruc";              // 要连接的数据库名称

// 创建MySQLi连接对象
$conn = new mysqli($servername, $username, $passwd, $database);

// 检查连接是否成功
if ($conn->connect_error) {
    die("连接失败: " . $conn->connect_error);  // 连接失败时终止脚本并显示错误信息
}
```

练习参考代码

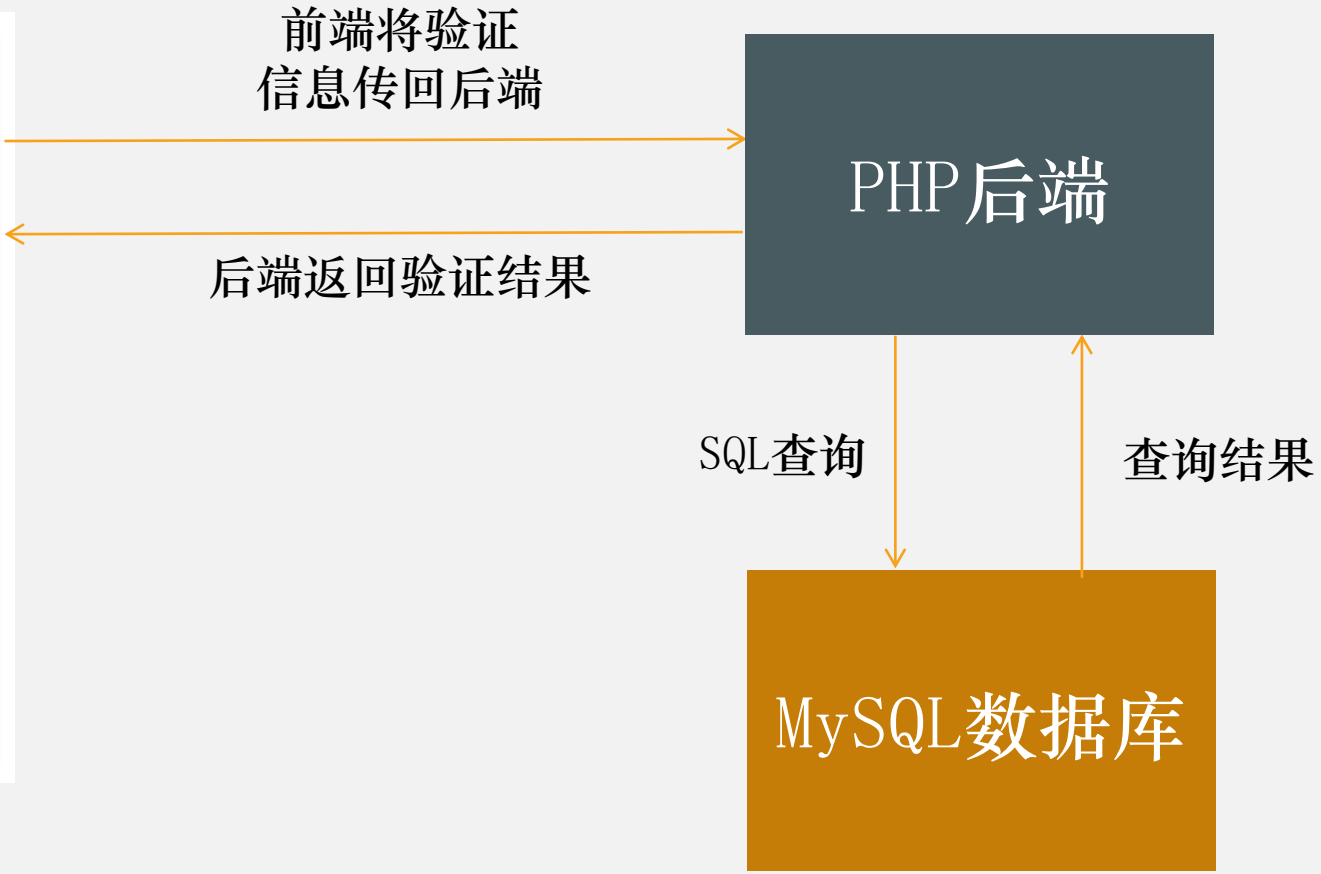
```php
// SQL查询语句，从user表中获取所有数据
$sql = "SELECT * FROM user";

// 执行SQL查询并获取结果集
$res = $conn->query($sql);

// 检查结果集中是否有数据
if ($res->num_rows > 0) {
    // 如果有数据，开始创建HTML表格
    echo "<table border='1'>";
    // 表格标题行
    echo "<tr><th>ID</th><th>用户名</th><th>角色</th><th>密码</th></tr>";

    // 循环遍历结果集中的每一行数据
    while($row = $res->fetch_assoc()) {
        echo "<tr>";
        // 输出每列数据
        echo "<td>".$row['id']."</td>";
        echo "<td>".$row['name']."</td>";
        echo "<td>".$row['role']."</td>";
        echo "<td>".$row['passwd']."</td>";
        echo "</tr>";
    }
    echo "</table>";

}
```

示例4——微人大登录界面（MYSQL）

微人大登录界面（MYSQL）

上一个示例中，我们为微人大登录页面搭建了有会话功能的后端，但是我们的用户名，密码等都是写死在代码中的。那么我们如何将所有的数据持久化地存储下来呢？这就需要用到数据库

未登录时，输入用户名密码登录



微人大登录界面（MYSQL）

其余的代码均不变，只需要修改login.php中的逻辑

```php
<?php
// 开启session，用于存储用户登录状态
session_start();

// 数据库连接配置
$servername = "127.0.0.1:3306";  // MySQL服务器地址和端口
$username = "root";              // 数据库用户名
$passwd = "kali";                // 数据库密码
$database = "vruc";              // 要连接的数据库名

// 创建数据库连接
$conn = new mysqli($servername, $username, $passwd, $database);
// 检查连接是否成功
if ($conn->connect_error) {
    die("连接失败: " . $conn->connect_error);  // 连接失败时终止脚本并显示错误
}

// 将POST请求中的所有变量转换为PHP变量
// 例如：$_POST['name'] 转换为 $name
foreach($_POST as $key=>$val) {
    $$key = $val;
}

// 验证码映射表，数字索引对应验证码字符串
$captcha = [0=>"pupr", 1=>"khrb", 2=>"huks",
            3=>"egwb", 4=>"ekdv", 5=>"khns",
            6=>"wuxc", 7=>"tcyk", 8=>"nupf",
            9=>"brpk"];

```

微人大登录界面（MYSQL）

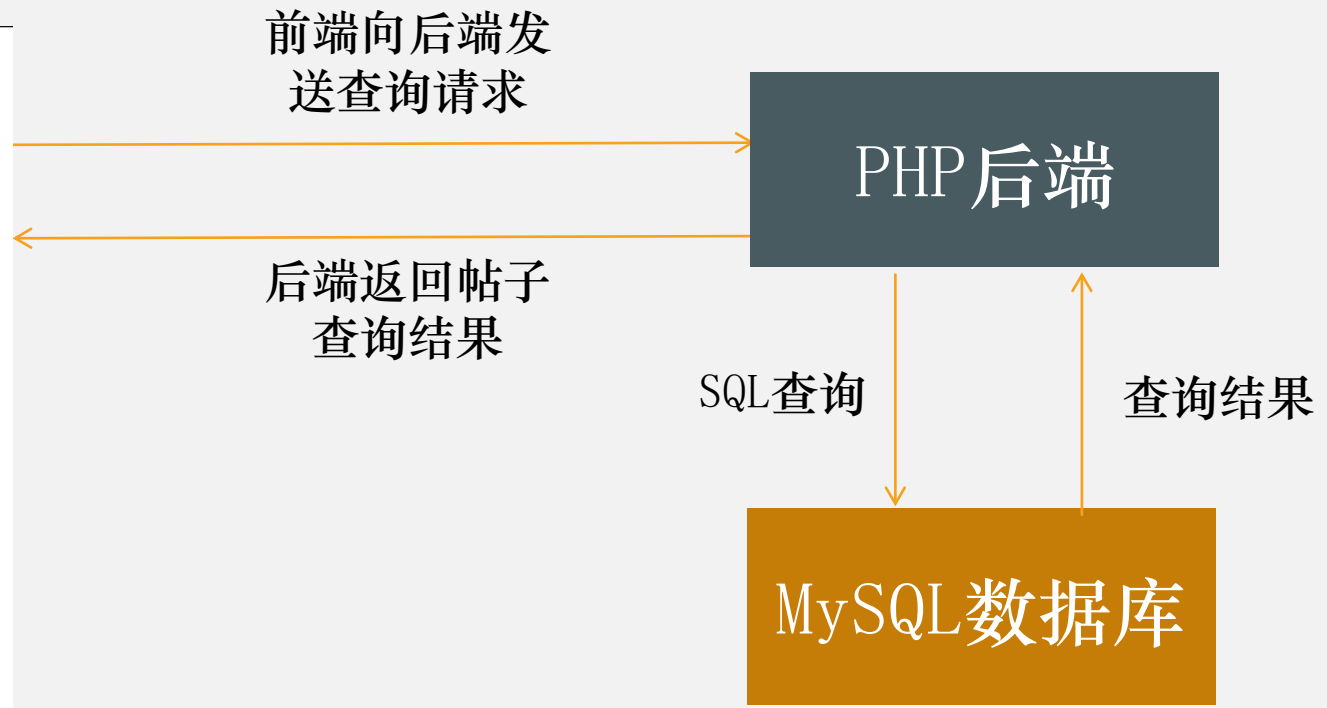
其余的代码均不变，只需要修改login.php中的逻辑

```php
// 验证用户输入的验证码是否正确
if ($code === $captcha[$id]) {
    // 验证码正确，准备SQL查询语句
    // 注意：这里存在SQL注入风险，直接拼接用户输入
    $sql = "SELECT * FROM user where name='$name' and passwd='$passwd'";
    $res = $conn->query($sql);

    // 检查查询结果是否有匹配的用户
    if ($res->num_rows > 0) {
        // 登录成功，存储用户信息到session
        $row = $res->fetch_assoc();
        $_SESSION["role"] = $row["role"];  // 用户角色
        $_SESSION["name"] = $row["name"];  // 用户名
    } else {
        // 用户名或密码错误，返回登录页
        echo "<script>alert('wrong username and password!');location.href='../index.php'</script>";
        die();
    }

    // 登录成功，跳转到面板页面
    echo "<script>location.href='pannel.php'</script>";
} else {
    // 验证码错误，返回登录页
    echo "<script>location.href='../index.php'</script>";
    die();
}
```

示例5——RUC小喇叭（MYSQL）

RUC小喇叭（MYSQL）

之间的SESSION版本中，我们实现了将帖子存储在会话中，每次会话结束，浏览器清理缓存等，数据都会消失。现在我们将帖子数据存储到数据库中




```
MariaDB [rucwall]> select * from posts;
+---------+---------------------+---------+----------+-------------+
| post_id | time                | content | like_num | comment_num |
+---------+---------------------+---------+----------+-------------+
|       1 | 2025-04-10 10:58:40 | 123     |        0 |           0 |
|       2 | 2025-04-10 10:58:47 | nihao   |        0 |           0 |
+---------+---------------------+---------+----------+-------------+
2 rows in set (0.001 sec)
```

RUC小喇叭（MYSQL）

我们首先要创建一下数据库和表
我们的数据库起名为rucwall，然后创建表posts，表结构如下：

字段名	类型	默认值	备注
post_id	int		AUTO_INCREMENT NOT NULL PRIMARY KEY
time	timestamp	CURRENT_TIMESTAMP	NOT NULL
content	text		NOT NULL
like_num	int	0	NOT NULL
comment_num	int	0	NOT NULL

值得注意的是，AUTO_INCREMENT表示该字段会自增，常用于id字段。CURRENT_TIMESTAMP表示当前的时间戳

RUC小喇叭（MYSQL）

我们首先要创建一下数据库和表，对应的SQL语句如下：

```
MariaDB [vruc]> create database rucwall;
Query OK, 1 row affected (0.000 sec)

MariaDB [vruc]> use rucwall;
Database changed
MariaDB [rucwall]> create table posts(
    -> post_id int AUTO_INCREMENT PRIMARY KEY,
    -> time timestamp default CURRENT_TIMESTAMP NOT NULL,
    -> content text NOT NULL,
    -> like_num int default 0 NOT NULL,
    -> comment_num int default 0 NOT NULL
    -> );
Query OK, 0 rows affected (0.008 sec)
```

RUC小喇叭（MYSQL）

创建好之后，我们需要修改两个文件。首先是index.php

```php
<?php
// 启动会话
session_start();
// 数据库连接配置
$servername = "127.0.0.1:3306";
$username = "root";
$passwd = "kali";
$database = "rucwall";

// 创建数据库连接
$conn = new mysqli($servername, $username, $passwd, $database);
// 检查连接是否成功
if ($conn->connect_error) {
    die("连接失败: " . $conn->connect_error);
}
```

```php
// 查询所有帖子
$sql = "SELECT * FROM posts";
$res = $conn->query($sql);
// 存储帖子数据的数组
$posts = array();
// 遍历查询结果
while ($row = $res->fetch_assoc()) {
    $posts[] = $row;
}
```

RUC小喇叭（MYSQL）

创建好之后，我们需要修改两个文件。还需要修改submit_post.php

```php
<?php
// 启动会话
session_start();

// 数据库连接配置
$servername = "127.0.0.1:3306";  // 数据库服务器地址和端口
$username = "root";              // 数据库用户名
$passwd = "kali";                // 数据库密码
$database = "rucwall";           // 数据库名称

// 创建数据库连接
$conn = new mysqli($servername, $username, $passwd, $database);
// 检查连接是否成功
if ($conn->connect_error) {
    die("连接失败: " . $conn->connect_error);
}
```

```php
// 获取前端提交的帖子内容
$content = $_POST['content'];

// 准备SQL插入语句，使用参数化查询防止SQL注入
$sql = "INSERT INTO posts (content) VALUES (?)";

// 预处理SQL语句
$stmt = $conn->prepare($sql);
// 绑定参数
$stmt->bind_param("s", $content);

// 执行SQL语句
$res = $stmt->execute();
```

WEB应用开发——后端

AJAX

目录

拓展——JS中的PROMISE对象

参考教程

https://www.runoob.com/js/js-promise.html

https://www.bilibili.com/video/BV1WP4y187Tu/

Promise

我们之前遇到的异步任务都是一次异步，如果需要多次调用异步函数呢？例如，如果我想分三次输出字符串，第一次间隔 1 秒，第二次间隔 4 秒，第三次间隔 3 秒：

```
setTimeout(function () {
    console.log("First");
    setTimeout(function () {
        console.log("Second");
        setTimeout(function () {
            console.log("Third");
        }, 3000);
    }, 4000);
}, 1000);
```

这段程序实现了这个功能，但是它是用〝函数瀑布〞来实现的。可想而知，在一个复杂的程序当中，用〝函数瀑布〞实现的程序无论是维护还是异常处理都是一件特别繁琐的事情，而且会让缩进格式变得非常冗赘。

Promise

- Promise 是 JavaScript 中用于处理异步操作的对象，它代表一个异步操作的最终完成（或失败）及其结果值。以下是 Promise 的核心概念：
  - 三种状态：pending （进行中）：初始状态；fulfilled （已成功）：操作成功完成；fulfilled （已成功）：操作成功完成
  - 不可逆性 ：状态一旦改变（从 pending 到 fulfilled 或 rejected）就不可再变

```javascript
const promise = new Promise((resolve, reject) => {
  // 异步操作
  if (/* 成功条件 */) {
    resolve(value); // 状态变为 fulfilled
  } else {
    reject(error); // 状态变为 rejected
  }
});
```

Promise

- Promise 构造函数是 JavaScript 中用于创建 Promise 对象的内置构造函数。
- Promise 构造函数接受一个函数作为参数，该函数是同步的并且会被立即执行，所以我们称之为起始函数。起始函数包含两个参数 resolve 和 reject，分别表示 Promise 成功和失败的状态。
- 起始函数执行成功时，它应该调用 resolve 函数并传递成功的结果。当起始函数执行失败时，它应该调用 reject 函数并传递失败的原因。
- Promise 构造函数返回一个 Promise 对象，该对象具有以下几个方法：
  - then：用于处理 Promise 成功状态的回调函数。
  - catch：用于处理 Promise 失败状态的回调函数。
  - finally：无论 Promise 是成功还是失败，都会执行的回调函数

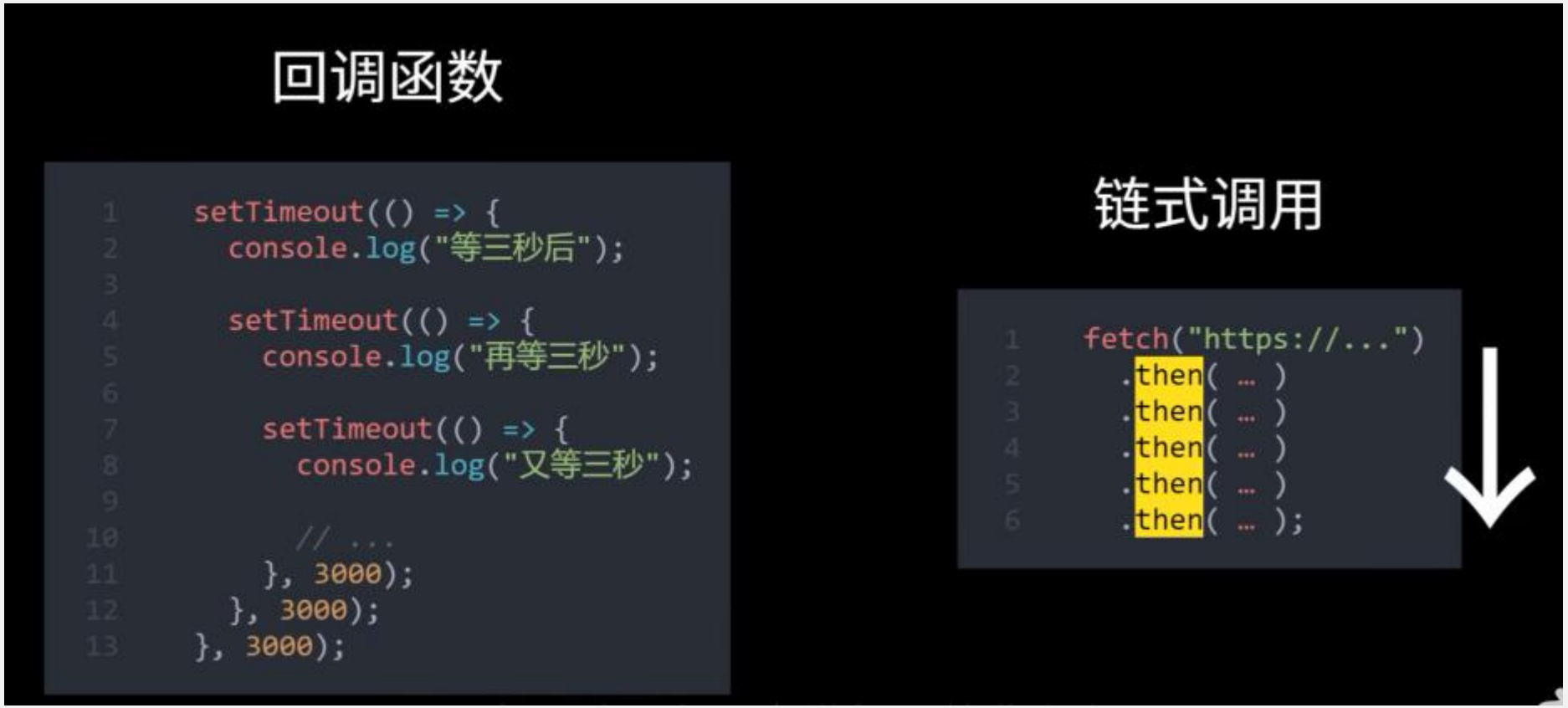
```
const promise = new Promise((resolve, reject) => {
  // 异步操作
  setTimeout(() => {
    if (Math.random() < 0.5) {
      resolve('success');
    } else {
      reject('error');
    }
  }, 1000);
});

promise.then(result => {
  console.log(result);
}).catch(error => {
  console.log(error);
});
```

Promise

现在我们用 Promise 来实现同样的功能：

```
new Promise(function (resolve, reject) {
    setTimeout(function () {
        console.log("First");
        resolve();
    }, 1000);
}).then(function () {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            console.log("Second");
            resolve();
        }, 4000);
    });
}).then(function () {
    setTimeout(function () {
        console.log("Third");
    }, 3000);
});
```

Promise



仅从这个例子，我们似乎还体会不到Promise的妙处，这是因为我们的异步操作过于简单，后面的练习会让大家有更直观的体验。

练习7——AJAX练习

练习要求

给你了三个后端文件server_a.php,server_b.php,server_c.php，请你依次向他们发送请求，并且将返回的结果输出到html页面上。
基础：使用xhr来请求，并使用回调函数来处理展示
进阶：使用fetch来请求，并使用promise对象来处理展示

练习要求

如果使用xhr，则我们需要嵌套三个xhr请求：

```
48      document.getElementById('sendRequest').addEventListener('click', function () {
49          const xhr1 = new XMLHttpRequest();
50          xhr1.open('POST', '/server_a.php', true);
51          xhr1.setRequestHeader('Content-Type', 'application/json');
52
53  >       xhr1.onload = function () {···
113
114         xhr1.onerror = function () {
115             showResp({ status: false, received_data: '网络错误' });
116         };
117
118         xhr1.send('Hello Server A!');
119     });
```

练习要求

如果使用xhr，则我们需要嵌套三个xhr请求：

```
xhr1.onload = function () {
    if (xhr1.status === 200) {
        try {
            const data = JSON.parse(xhr1.responseText);
            showResp(data);

            const xhr2 = new XMLHttpRequest();
            xhr2.open('POST', '/server_b.php', true);
            xhr2.setRequestHeader('Content-Type', 'application/json');

            xhr2.onload = function () {
                if (xhr2.status === 200) {···
                    showResp({ status: false, received_data: '请求失败: ' + xhr2.status });
                }
            };

            xhr2.onerror = function () {
                showResp({ status: false, received_data: '网络错误' });
            };

            xhr2.send('Hello Server B!');
        } catch (e) {
            showResp({ status: false, received_data: '解析响应失败' });
        }
```

练习要求

如果使用xhr，则我们需要嵌套三个xhr请求：

```
xhr2.onload = function () {
    if (xhr2.status === 200) {
        try {
            const data = JSON.parse(xhr2.responseText);
            showResp(data);

            const xhr3 = new XMLHttpRequest();
            xhr3.open('POST', '/server_c.php', true);
            xhr3.setRequestHeader('Content-Type', 'application/json');

            xhr3.onload = function () {
                if (xhr3.status === 200) {
                    try {
                        const data = JSON.parse(xhr3.responseText);
                        showResp(data);
                    } catch (e) {
                        showResp({ status: false, received_data: '解析响应失败' });
                    }
                } else {
                    showResp({ status: false, received_data: '请求失败: ' + xhr3.status });
                }
            };

            xhr3.onerror = function () {
                showResp({ status: false, received_data: '网络错误' });
            };

            xhr3.send('Hello Server C!');
```

练习要求

如果使用promise，我们可以“顺序”请求：

```
document.getElementById('sendRequest').addEventListener('click', function () {
    // 1. 发送第一个AJAX请求到server_a.php
    sendRequest('server_a.php', 'Hello, Server A!')
        // 将响应转换为JSON格式
        .then(response => response.json())
        .then(data => {
            // 在页面上显示第一个请求的响应
            showResp(data);
            // 2. 发送第二个AJAX请求到server_b.php
            return sendRequest('server_b.php', 'Hello, Server B!');
        })
        .then(response => response.json())
        .then(data => {
            // 在页面上显示第二个请求的响应
            showResp(data);
            // 3. 发送第三个AJAX请求到server_c.php
            return sendRequest('server_c.php', 'Hello, Server C!');
        })
        .then(response => response.json())
        .then(data => {
            // 在页面上显示第三个请求的响应
            showResp(data);
        })
        .catch(error => {
            // 捕获并处理任何请求错误
            console.error('请求出错:', error);
            document.getElementById('responseContainer').textContent = '请求出错: ' + error.received_data;
        });
```

示例8——微人大登录界面（AJAX）

微人大登录界面（AJAX）

- 上一个示例中，我们为微人大登录页面的后端连接了数据库。现在有个新的问题，如果我们用户名、密码、验证码输入错误，页面会进行相应提示，但是要展示出提示内容需要刷新一遍网页。当网页比较小的时候感觉不到，但如果网页比较复杂，或者网络信号比较弱，每次都刷新全部网页不仅没必要，而且用户体验很差。AJAX的诞生就是为了实现“局部刷新”

如果密码输入错误，只会更新红框中的内容，不会重新加载整个网页

微人大登录界面（AJAX）

- 这里我们主要修改index.php和login.php。首先来看index.php

```
58  // 获取表单和登录按钮元素
59  const loginForm = document.getElementById('loginForm');
60  const loginBtn = document.getElementById('loginBtn');
61
62  // 登录按钮点击事件处理
63  loginBtn.addEventListener('click', function(e) {
64      e.preventDefault();  // 阻止表单默认提交行为
65
66      // 收集表单数据
67      const formData = new FormData(loginForm);
68
69      // 创建XMLHttpRequest对象用于AJAX请求
70      const xhr = new XMLHttpRequest();
71
72      // 配置请求: POST方法, 目标URL为php/login.php, 异步请求
73      xhr.open('POST', 'php/login.php', true);
74
```

微人大登录界面（AJAX）

- 这里我们主要修改index.php和login.php。首先来看index.php

```
75      // 请求完成后的回调函数
76      xhr.addEventListener('loadend', () => {
77        // 解析服务器返回的JSON响应
78        const response = JSON.parse(xhr.responseText);
79
80        // 根据响应结果处理
81        if (response.success) {
82          // 登录成功，跳转到面板页面
83          location.href='php/pannel.php';
84        } else {
85          // 登录失败，显示错误信息
86          document.getElementById('errorMessage').firstChild.innerText = response.message;
87        }
88      })
89
90      // 发送请求，附带表单数据
91      xhr.send(formData);
92    });
```

微人大登录界面（AJAX）

- 这里我们主要修改index.php和login.php。然后是login.php

```php
<?php
// 开启会话，用于存储用户登录状态
session_start();

// 数据库连接配置
$servername = "localhost";  // 数据库服务器地址
$username = "root";         // 数据库用户名
$passwd = "kali";  // 数据库密码
$database = "vruc";         // 数据库名称

// 创建数据库连接
$conn = new mysqli($servername, $username, $passwd, $database);
if ($conn->connect_error) {
    // 连接失败时终止脚本并显示错误
    die("连接失败: " . $conn->connect_error);
}

```

```php
// 验证验证码是否正确
if ($code === $captcha[$id]) {
    // 存在SQL注入漏洞的查询语句，生产环境中需要改成参数化查询
    $sql = "SELECT * FROM user where name='$name' and passwd='$passwd'";
    $res = $conn->query($sql);

    // 检查查询结果
    if ($res->num_rows > 0) {
        // 登录成功，设置session变量
        $row = $res->fetch_assoc();
        $_SESSION["role"] = $row["role"];  // 用户角色
        $_SESSION["name"] = $row["name"];  // 用户名

        // 设置成功响应
        $response['success'] = true;
        $response['message'] = '登录成功';
    } else {
        // 用户名或密码错误
        $response['message'] = '用户名或密码错误';
        echo json_encode($response);  // 返回JSON格式响应
        die();  // 终止脚本执行
    }
    // 返回成功响应
    echo json_encode($response);
```

示例9——RUC小喇叭（AJAX）

RUC小喇叭（AJAX）

同样的，我们每次发送了帖子之后，上方的输入框、搜索框没有发生变化，所以没有必要刷新整个页面，只需要局部刷新下方的帖子列表即可

RUC小喇叭（AJAX)

为了实现此功能，我们需要修改js.js，send_post.php两个文件，新增get_post.php一个文件。
首先来看send_post.php
先定义一个响应数组

```php
// 初始化响应数组 - 统一接口返回格式
$response = [
    'success' => false,    // 操作是否成功标志位
    'message' => '未知错误', // 返回给客户端的消息
];
```

然后进行一些检查

```php
// 检查数据库连接是否成功
if ($conn->connect_error) {
    http_response_code(500); // 设置HTTP状态码为500(服务器内部错误)
    $response['message'] = '数据库连接失败';
    echo json_encode($response); // 返回JSON格式的错误信息
    die(); // 终止脚本执行
}
```

RUC小喇叭（AJAX)

```php
// 验证请求方法是否为POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405); // 405 Method Not Allowed
    die('只允许POST请求'); // 直接返回文本错误信息
}
```

获取请求体中的json数据并解析

```php
/**
 * 处理请求体数据
 */
$json_data = file_get_contents('php://input'); // 从输入流获取原始JSON数据
if (empty($json_data)) {
    http_response_code(400); // 400 Bad Request
    die('请求体不能为空');
}
```

RUC小喇叭（AJAX)

验证前端发来的字段

```php
// 验证必须字段content是否存在且不为空
if (!isset($data['content']) || empty(trim($data['content']))) {
    http_response_code(400);
    $response['message'] = 'content字段不能为空';
    echo json_encode($response);
    die();
}
```

执行插入操作

```php
// 去除content字段两端的空白字符
$content = trim($data['content']);

/**
 * 数据库操作部分
 * 使用预处理语句防止SQL注入
 */
$sql = "INSERT INTO posts (content) VALUES (?)"; // 参数化查询SQL

$stmt = $conn->prepare($sql); // 预处理SQL语句
$stmt->bind_param("s", $content); // 绑定参数(s表示字符串类型)

// 执行SQL语句
$res = $stmt->execute();
```

返回处理结果

```php
// 处理执行结果
if($res) {
    // 发布成功
    $response['success'] = true;
    $response['message'] = '发布成功';
} else {
    // 发布失败
    http_response_code(500);
    $response['message'] = '发布失败';
}

// 返回JSON格式的响应
echo json_encode($response);
```

RUC小喇叭（AJAX)

再来看get_post.php。该文件的作用就是以json格式返回posts列表。与数据库建立连接部分与send_post.php相同，只是后面的sql语句和操作不同。

```php
$sql = "SELECT * FROM posts";

$res = $conn->query($sql);

// 处理执行结果
if ($res) {
    // 存储帖子数据的数组
    $posts = array();

    // 遍历查询结果
    while ($row = $res->fetch_assoc()) {
        $posts[] = $row;
    }

    // 设置成功响应
    $response['success'] = true;
    $response['message'] = '获取帖子信息成功';
    $response['data'] = $posts;
} else {
    // 获取帖子信息失败，显示错误并重定向回首页
    http_response_code(500);
    $response['message'] = '获取帖子信息失败';
}
echo json_encode($response);
```

RUC小喇叭（AJAX)

最后是前端的js部分。首先为最顶上输入表单绑定事件响应

```
110  // 添加表单提交事件监听器
111  intputForm.addEventListener("submit", function(event) {
112      event.preventDefault(); // 阻止表单的默认提交行为
113
114      // 获取并清理输入内容
115      const post_content = inputTextarea.value.trim();
116      // 如果内容为空，直接返回
117      if(post_content.length === 0 || post_content === null){
118          return;
119      }
120
121      // 发送AJAX请求到服务器，提交新的帖子内容
122      const xhr = new XMLHttpRequest();
123
124      xhr.open("POST", "/php/send_post.php", true); // 这里的路径是相对路径，需要根据实际情况调整
125      xhr.setRequestHeader("Content-Type", "application/json"); // 设置请求头
```

RUC小喇叭（AJAX）

```
127     // 监听服务器响应
128     // 请求完成后的回调函数
129     xhr.addEventListener('loadend', () => {
130         // 解析服务器返回的JSON响应
131         const response = JSON.parse(xhr.responseText);
132         console.log(response);
133         // 根据响应结果处理
134         if (response.success) {
135           // 发送帖子成功，更新页面显示
136           refreshPosts();
137         } else {
138           // 发送帖子错误，显示错误信息
139           alert(response.message);
140         }
141       })
142
143     // 构造请求体
144     const requestBody = JSON.stringify({ content: post_content });
145     // 发送请求
146     xhr.send(requestBody);
```

RUC小喇叭（AJAX)

然后定义局部刷新帖子列表的函数refreshPosts

```
186  function refreshPosts() {
187      // 发送AJAX请求到服务器，获取所有帖子
188      const xhr = new XMLHttpRequest();
189
190      xhr.open("GET", "/php/get_post.php", true); // 这里的路径是相对路径，需要根据实际情况调整
191      xhr.setRequestHeader("Content-Type", "application/json"); // 设置请求头
192      // 监听服务器响应
193      xhr.addEventListener('loadend', () => {
194          // 解析服务器返回的JSON响应
195          const response = JSON.parse(xhr.responseText);
196          console.log(response);
197          // 处理每个帖子并显示
198          if (response.success) {
199              // 获取帖子成功
200              // 删除当前所有帖子
201              removeAllPosts();
202              // 更新页面显示
203              showPosts(response.data);
204          } else {
205              // 获取帖子错误，显示错误信息
206              alert(response.message);
207          }
208      })
209
210      // 发送请求
211      xhr.send();
212  }
```

RUC小喇叭（AJAX)

然后定义局部刷新帖子列表的函数refreshPosts。其中removeAllPosts是删除当前所有帖子的函数。showPosts是根据posts列表将post添加到页面上的函数

```
186  function refreshPosts() {
187      // 发送AJAX请求到服务器，获取所有帖子
188      const xhr = new XMLHttpRequest();
189
190      xhr.open("GET", "/php/get_post.php", true); // 这里的路径是相对路径，需要根据实际情况调整
191      xhr.setRequestHeader("Content-Type", "application/json"); // 设置请求头
192      // 监听服务器响应
193      xhr.addEventListener('loadend', () => {
194          // 解析服务器返回的JSON响应
195          const response = JSON.parse(xhr.responseText);
196          console.log(response);
197          // 处理每个帖子并显示
198          if (response.success) {
199              // 获取帖子成功
200              // 删除当前所有帖子
201              removeAllPosts();
202              // 更新页面显示
203              showPosts(response.data);
204          } else {
205              // 获取帖子错误，显示错误信息
206              alert(response.message);
207          }
208      })
209
210      // 发送请求
211      xhr.send();
212  }
```