“趣 PHP”网站项目开发文档

准备工作

准备公共函数和配置文件

将之前在第 7 章、第 10 章的动手实践项目代码中已经写好的 config()、input()、thumb()、page_html()、page_sql()函数复制到本项目的 common\function.php 文件中，这些函数在开发中会经常用到。

接下来，在 common 目录中创建项目的配置文件 config.php，具体代码如下。

```
<?php
return [
    'DB_CONNECT' => ['host' => 'localhost', 'user' => 'root', 'pass' => '123456',
                     'dbname' => 'php_fun', 'port' => '3306'],  // 数据库连接配置
    'DB_CHARSET' => 'utf8',      // 数据库字符集
    'DB_PREFIX' => 'fun_',       // 项目的数据库表前缀
];
```

上述配置中，“DB_PREFIX”用于在程序中自动为数据表加上前缀进行访问。例如，为“user”数据表添加前缀“fun_”后变为“fun_user”。实际开发中，为项目的数据表设置前缀是一个好习惯，可以用来区分一个数据库中的多个项目。

引入公共文件

创建项目的初始化文件 common\init.php，实现相关文件的引入和环境配置，具体代码如下。

```
<?php
date_default_timezone_set('Asia/Shanghai');            // 配置时区
session_set_cookie_params(0, null, null, null, true); // 开启 Session 的 HttpOnly
mb_internal_encoding('UTF-8');                         // 配置 mbstring 扩展的字符集
require './common/function.php';                       // 引入函数库
session_start();                                       // 开启 Session
if(!isset($_SESSION['fun'])){
    $_SESSION = ['fun' => []];      // 为项目创建 Session，统一保存到 fun 中
}
```

完成上述代码后，在具体功能脚本中引入 common\init.php 文件，即可实现项目的初始化。

配置网页标题和顶部导航

在项目的 HTML 模板中，每个页面都需要显示网页标题和顶部导航，当需要修改这些内容时，直接编辑模板文件会非常麻烦。因此，将这些数据保存到配置文件中，然后在模板中直接读取配置即可，使代码更好维护。编辑 common\config.php 文件，新增配置如下。

```
// 网页标题
'APP_TITLE' => '趣 PHP - 学习分享网站',
// 顶部导航
'APP_NAV' => ['hot' => '热门', 'new' => '新鲜', 'pic' => '趣图',
              'text' => '趣文', 'video' => '视频'],
```

完成上述代码后，就可以在 HTML 模板中通过如下代码获取网页标题。

```
<title><?=config('APP_TITLE')?></title>
```

考虑到项目的顶部导航在每个页面中都需要显示，将这部分代码保存到公共模板目录中。在 view 目录下创建 common 目录，然后编写 top.html 用于保存顶部导航，该文件的关键代码如下。

```
<?php foreach (config('APP_NAV') as $k => $v): ?>
  <a class="<?=(isset($type) && ($type == $k)) ? 'curr' : ''?>"
  href="./?type=<?=$k?>"><?=$v?></a>
<?php endforeach; ?>
```

上述代码第 2 行使用了变量$type，该变量表示当前位于顶部导航的某个链接指向的页面中，值可以是 hot、new、pic、text 或 video。当 APP_NAV 数组的键名与$type 相同时，为链接添加 class 为 curr 的样式。$type 的值是通过 URL 参数 type 获取的，因此，在载入 HTML 模板前，应先接收该参数。

完成 view\common\top.html 后，在其他 HTML 模板中直接引入该文件即可获得顶部导航。

数据库操作类

创建类文件

项目的开发离不开数据库，为了使数据库操作的代码易于维护，下面编写一个数据库操作类，将代码保存到 common\library\Db.php 文件中，Db 类的代码如下。

```
class Db
{
    private static $instance;       // 保存本类的单例对象
    private static $mysqli;         // 保存数据库连接对象
    private function __construct()  // 在私有构造方法中连接数据库
    {
        $config = config('DB_CONNECT');
        self::$mysqli = new mysqli($config['host'], $config['user'],
                        $config['pass'], $config['dbname'], $config['port']);
        if (self::$mysqli->connect_error) {
            exit('数据库连接失败：' . self::$mysqli->connect_error());
        }
        self::$mysqli->set_charset(config('DB_CHARSET'));
    }
    private function __clone() {}           // 阻止克隆
    public static function getInstance()    // 获得单例对象
    {
        return self::$instance ?: (self::$instance = new self());
    }
```

```
20 }
```

上述代码通过单例模式实现了对 MySQLi 扩展的封装。需要注意的是，第 7、13 行调用自定义函数 config()，以及在第 11 行通过 exit()停止脚本，都会影响 Db 类的通用性。若要考虑 Db 类的通用性，建议通过 getInstance()方法的参数传入数据库连接配置；若连接失败，则先记录错误信息，然后编写一个 getError()方法专门用于错误信息的获取。此处为了方便代码编写，暂时不考虑通用性。

创建数据库操作类之后，在 common\init.php 中引入类文件，代码如下。

```
require './common/library/Db.php';      // 引入数据库操作类
```

封装常用方法

在 Db 类中完成数据库连接后，下面开始编写一些常用的方法，用于实现数据库操作。

（1）执行 SQL 语句

编写 query()方法实现预处理方式执行 SQL 语句，并支持二维数组批量执行。具体代码如下。

```
1  public function query($sql, $type = '', array $data = [])
2  {
3      if (!$stmt = self::$mysqli->prepare($sql)) {    // 预处理 SQL
4          exit("SQL[$sql]预处理失败: " . self::$mysqli->error);
5      }
6      if ($data) {     // 当有数据部分时，执行参数绑定
7          $data = is_array(current($data)) ? $data : [$data];  // 自动转换为二维数组
8          $params = array_shift($data);          // 将$data 中的第 1 个元素出栈
9          $args = [$type];                       // 准备$stmt->bind_param()的参数
10         foreach ($params as &$args[]) ;        // 为$args 数组添加引用元素
11         call_user_func_array([$stmt, 'bind_param'], $args);  // 自动完成参数绑定
12     }
13     if (!$stmt->execute()) {          // 执行第 1 次操作
14         exit('数据库操作失败: ' . $stmt->error);
15     }
16     foreach ($data as $row) {         // 进入批量模式
17         foreach ($row as $k => $v) {
18             $params[$k] = $v;         // 更新每个字段的值
19         }
20         if (!$stmt->execute()) {      // 执行第 2~N 次操作
21             exit('数据库操作失败: ' . $stmt->error);
22         }
23     }
24     return $stmt;
25 }
```

在上述代码中，query()函数的参数$sql 表示 SQL 语句，$type 表示参数绑定的类型，$data 表示参数绑定的数据部分。第 6~12 行实现了为$data 进行参数绑定。其中，第 11 行调用 call_user_func_array()函数时传递的第 1 个参数表示$stmt 对象的 bind_param()方法，第 2 个参数表示调用指定方法时传入的参数。

接下来，通过具体代码演示 query()方法的使用，示例如下。

```
$db = Db::getInstance();
$sql = 'UPDATE `test` SET `name`=? WHERE `id`=?';
```

```
// 执行单次操作
$db->query($sql, 'si', ['test', 123]);
// 执行批量操作
$db->query($sql, 'si', [['test01', 123], ['test02', 456]]);
```

通过示例可以看出，利用 query()方法可以很方便地执行 SQL 语句，并支持批量操作。

（2）自动添加表前缀

当为项目的数据表添加前缀后，每次书写 SQL 时都要加上表前缀，这会在修改表前缀时非常麻烦，还容易遗漏。因此，为了更好地处理表前缀，在 SQL 中使用模板语法来代替表名。例如，若要查询数据表“fun_user”中的数据，则模板语法中使用“__USER__”表示表名。示例如下。

```
SELECT * FROM `fun_user`;       # 原始 SQL
SELECT * FROM __USER__;         # 模板语法
```

为了实现这个效果，可以利用正则表达式对 SQL 语句模板语法进行自动替换。下面将如下代码添加到 Db 类 query()方法中调用$mysqli->prepare()方法预处理 SQL 之前的位置。

```
$sql = preg_replace_callback('/__([A-Z0-9_-]+)__/sU', function($match) {
    return '`' . config('DB_PREFIX') . strtolower($match[1]) . '`';
}, $sql);
```

上述代码可以将 SQL 语句中的“__USER__”自动替换为“fun_user”。其中，preg_replace_callback()函数的第 2 个参数是回调函数，该回调函数的参数$match 表示正则表达式中子模式匹配到的结果，其返回值将会替换子模式匹配到的字符串。使用“$match[1]”取得匹配到模板语法表名后，通过 strtolower()函数转换为小写，然后在前面拼接了项目配置的表前缀，最后将表名用反引号（`）包裹。

（3）返回执行结果

继续编写 Db 类，实现 fetchAll()、fetchRow()、execute()方法，用于根据不同需求返回不同形式的结果，具体代码如下。

```
// 查询所有结果
public function fetchAll($sql, $type = '', array $data = [])
{
    return $this->query($sql, $type, $data)->get_result()->fetch_all(MYSQLI_ASSOC);
}
// 查询一行结果
public function fetchRow($sql, $type = '', array $data = [])
{
    return $this->query($sql, $type, $data)->get_result()->fetch_assoc();
}
// 执行 SQL，对于 INSERT 语句返回最后插入的 id，其他语句返回受影响行数
public function execute($sql, $type = '', array $data = [])
{
    $stmt = $this->query($sql, $type, $data);
    return (strtoupper(substr(trim($sql), 0, 6)) == 'INSERT') ?
            $stmt->insert_id : $stmt->affected_rows;
}
```

（4）封装 SELECT 查询

在项目开发中，对于数据的增加、删除、修改、查找都是常见操作。为了方便代码的编写，可以在 Db 类中增加这些操作。下面编写 select()方法用于执行 SELECT 查询，具体代码如下。

```
// 执行 SELECT 查询
```

```
public function select($table, $fields, $type = '', array $data = [],
                       $method = 'fetchAll')
{
    $fields = str_replace(',', '`,`', $fields);
    $where = implode(' AND ', self::buildFields(array_keys($data)));
    $limit = ($method == 'fetchRow') ? 'LIMIT 1' : '';
    return $this->$method("SELECT `$fields` FROM $table WHERE $where $limit",
           $type, $data);
}
// 将字段数组转换为 SQL 形式
private static function buildFields(array $fields)
{
    return array_map(function($v) { return "`$v`=?"; }, $fields);
}
```

在 select()方法的参数中，$table 表示表名；$fields 表示待查询的字段，多个字段用逗号分隔；$type 和$data 用于传递数据进行参数绑定。

完成 select()方法后，可以按照如下示例代码进行 SELECT 查询。

```
$db = Db::getInstance();
$data = $db->select('__USER__', 'name,password', 'i', ['id' => 1]);
// 实际执行的 SQL 语句：
// SELECT `name`,`password` FROM `fun_user` WHERE `id`=1
```

利用 select()方法可以返回所有查询结果，当只需要结果中的第一行或某个字段的值时，还可以通过 find()方法、value()方法来实现，具体代码如下。

```
// 执行 SELECT 查询，只返回一行结果
public function find($table, $fields, $type = '', array $data = [])
{
    return $this->select($table, $fields, $type, $data, 'fetchRow');
}
// 执行 SELECT 查询，返回$fields 字段查询到的值
public function value($table, $field, $type = '', array $data = [])
{
    return $this->find($table, $field, $type, $data, 'fetchRow')[$field];
}
```

（5）封装 INSERT 语句

通过分析 INSERT 语句的规律，通过如下代码实现自动插入数据，具体代码如下。

```
// 执行 INSERT 语句
public function insert($table, $type, array $data)
{
    $fields = self::arrayFields($data);
    $sql = "INSERT INTO $table SET " . implode(',', self::buildFields($fields));
    return $this->execute($sql, $type, $data);
}
// 从一维或二维数组中获取字段
private static function arrayFields(array $data)
```

```
10 {
11     $row = current($data);
12     return array_keys(is_array($row) ? $row : $data);
13 }
```

接下来通过代码演示 insert()方法的使用。

```
$db = Db::getInstance();
$id = $db->insert('__USER__', 'is', ['id' => 1, 'name' => 'test']);
// 实际执行的 SQL 语句：
// INSERT INTO `fun_user` SET `id`=1, `name`='test'
```

（6）封装 UPDATE 和 DELETE 语句

继续编写 Db 类，通过如下代码实现 update()方法和 delete()方法。

```
1  // 执行 UPDATE 语句
2  public function update($table, $type, array $data, $where = 'id')
3  {
4      $where = explode(',', $where);
5      $fields = array_diff(self::arrayFields($data), $where);
6      return $this->execute("UPDATE $table SET " . implode(',',
7             self::buildFields($fields)) . ' WHERE ' . implode(' AND ',
8             self::buildFields($where)), $type, $data);
9  }
10 // 执行 DELETE 语句
11 public function delete($table, $type, array $data)
12 {
13     $fields = implode(' AND ', self::buildFields(self::arrayFields($data)));
14     return $this->execute("DELETE FROM $table WHERE $fields", $type, $data);
15 }
```

接下来，通过代码演示 update()方法和 delete()方法的使用。

```
$db = Db::getInstance();
$data = ['email' => 'a@b.com', 'id' => 1, 'name' => 'test'];
$db->update('__USER__', 'sis', $data, 'id,name');
$db->delete('__USER__', 'i', ['id' => 1]);
// 实际执行的 SQL 语句：
// UPDATE `fun_user` SET `email`='a@b.com' WHERE `id`=1 AND `name`='test'
// DELETE FROM `fun_user` WHERE `id`=1
```

文件上传类

创建类文件

文件上传是项目开发中的常见功能，本项目的用户头像上传、栏目图片上传等功能都会用到。下面编写一个文件上传类，将代码保存到 common\library\Upload.php 文件中。Upload 类的代码如下。

```
1  class Upload
2  {
```

```
    private $file = [];          // 上传文件信息
    private $allow_ext = [];     // 允许上传的文件扩展名
    private $upload_dir = '';    // 上传文件保存目录
    private $new_dir = '';       // 在保存目录中创建的子目录
    private $error = '';         // 错误信息
    /**
     * 构造方法
     * @param array $file 上传文件 $_FILES['xx'] 数组
     * @param string $upload_dir 上传文件保存目录
     * @param array $new_dir 在保存目录中创建的子目录
     * @param array $allow_ext 允许上传的文件扩展名
     * @param int $limit 文件数量限制，默认 20，至少为 1
     */
    public function __construct(array $file, $upload_dir = '.',
    $new_dir = '', array $allow_ext = [], $limit = 20)
    {
        if (isset($file['error'])) {
            $this->file = $this->parse($file, max((int)$limit, 1));
        }
        $this->upload_dir = trim($upload_dir, '/');
        $this->new_dir = trim($new_dir, '/');
        $this->allow_ext = $allow_ext;
    }
    /**
     * 获取错误信息
     * @return string 错误信息
     */
    public function getError()
    {
        return $this->error;
    }
    // 解析$file 数组（该方法在下一步中实现）
    private function parse($file, $limit) { }
}
```

在实例化文件上传类时，用户可以指定文件的保存目录、自动创建的子目录、允许上传的文件名以及上传文件数量限制。下面的代码演示了 Upload 类的使用方法。

```
$file = isset($_FILES['up']) ? $_FILES['up'] : [];
$up = new Upload($file, './uploads', date('Y'), ['jpg', 'png'], 2);
```

在上述代码中，实例化 Upload 时传递的第 1 个参数表示上传“<input type="file" name="up">”文件，若将 name 属性值改为“up[]”则表示上传多个文件；第 2 个和第 3 个参数指定了文件成功上传后保存的路径为“./uploads/当前年份/文件名.jpg”，第 4 个参数指定了只允许上传 jpg、png 扩展名的文件，第 5 个参数指定了当进行多文件上传时，最多允许上传 2 个文件。

解析文件上传数组

在 Upload 类中，私有方法 parse()用于解析给定的“$_FILES['xx']”数组，自动识别单文件上传和多文件上传两种情况，同时进行错误检查、控制文件数量在$limit 的限制内。为了在遇到错误时给予提示信息，下面为 Upload 类增加成员属性$msg 保存这些提示信息，代码如下。

```
private $msg = [
    UPLOAD_ERR_INI_SIZE => '文件大小超过了服务器设置的限制！',
    UPLOAD_ERR_FORM_SIZE => '文件大小超过了表单设置的限制！',
    UPLOAD_ERR_PARTIAL => '文件只有部分被上传！',
    UPLOAD_ERR_NO_TMP_DIR => '上传文件临时目录不存在！',
    UPLOAD_ERR_CANT_WRITE => '文件写入失败！'
];
```

接下来，编写 parse()方法，具体代码如下。

```
private function parse($file, $limit)
{
    $result = [];
    if (is_array($file['error'])) {
        foreach ($file['error'] as $k => $v) {
            $v = (int) $v;
            $this->error = isset($this->msg[$v]) ? $this->msg[$v] : '';
            if ($v == UPLOAD_ERR_OK && ( --$limit >= 0)) {
                $result[$k] = ['name' => $file['name'][$k],
                  'tmp_name' => $file['tmp_name'][$k], 'type' => $file['type'][$k],
                  'size' => $file['size'][$k], 'error' => $file['error'][$k]];
            }
        }
    } elseif ($file['error'] == UPLOAD_ERR_OK) {
        $result[] = $file;
    } else {
        $this->error = isset($this->msg[$file['error']]) ? $this->msg[$v] : '';
    }
    return $result;
}
```

从上述代码可以看出，parse()方法会将$file 数组整理成一维数组的形式返回。如果文件存在错误，则记录错误信息并跳过。在记录错误信息后，通过 getError()方法可以获取错误信息。

实现单文件和多文件上传

编写用于多文件上传的 upload()方法和用于单文件的上传的 uploadOne()方法，具体代码如下。

```
/**
 * 上传多个文件
 * @return array 返回保存文件名的数组
 */
```

```
5  public function upload()
6  {
7      $result = [];
8      foreach ($this->file as $k => $v) {
9          if (!$save = $this->save($v)) {
10             continue;
11         }
12         $result[$k] = $save;
13     }
14     return $result;
15 }
16 /**
17  * 上传一个文件
18  * @return string 成功返回文件名，失败返回 false
19  */
20 public function uploadOne()
21 {
22     return $this->save(current($this->file));
23 }
24 // 保存上传文件（该方法在下一步中实现）
25 private function save($file) { }
```

从上述代码可以看出，upload()方法和 uploadOne()方法的区别在于返回值的形式不同，前者返回的是数组，后者返回的是字符串。值得一提的是，将 Upload()类的参数$limit 设为 1 时也可以实现单文件上传，但 uploadOne()方法在获取结果时会更加方便。

保存上传文件

编写 save()方法，实现上传一个指定文件，具体代码如下。

```
1  private function save($file)
2  {
3      if ($this->error || $file['error'] == UPLOAD_ERR_NO_FILE) {
4          return false;
5      }
6      $ext = pathinfo($file['name'], PATHINFO_EXTENSION);
7      if (!in_array(strtolower($ext), $this->allow_ext)) {
8          $this->error = '文件上传失败：只允许扩展名：' . implode(', ', $this->allow_ext);
9          return false;
10     }
11     $upload_dir = $this->upload_dir . '/' . $this->new_dir;
12     if (!is_dir($upload_dir) && !mkdir($upload_dir, 0777, true)) {
13         $this->error = '文件上传失败：无法创建保存目录！';
14         return false;
15     }
16     $new_name = md5(microtime(true)) . ".$ext";
```

```
17    if (!move_uploaded_file($file['tmp_name'], "$upload_dir/$new_name")) {
18        $this->error = '文件上传失败：无法保存文件！';
19        return false;
20    }
21    return trim($this->new_dir . '/' . $new_name, '/');
22 }
```

完成上述代码后，即可在实例化 Upload 类后调用 upload()或 uploadOne()方法实现文件上传。

用户登录与退出

设计用户表

在数据库中，user 表用于保存网站的注册用户，该表的具体结构如下表所示。

字段	数据类型	说明
id	INT UNSIGNED PRIMARY KEY AUTO_INCREMENT	用户 id
group	ENUM('admin','user') DEFAULT 'user' NOT NULL	用户组
name	VARCHAR(10) UNIQUE DEFAULT '' NOT NULL	用户名
email	VARCHAR(32) DEFAULT '' NOT NULL	电子邮箱
password	CHAR(32) DEFAULT '' NOT NULL	密码
salt	CHAR(6) DEFAULT '' NOT NULL	密码盐
avatar	VARCHAR(255) DEFAULT '' NOT NULL	头像地址
time	INT UNSIGNED DEFAULT 0 NOT NULL	注册时间

在上表中，需要重点关注 password 和 salt 字段。从安全角度考虑，通常不建议将用户的密码明文存储到数据库中，以避免数据泄露。因此，这里的 password 和 salt 字段只用于存储密码加密后的信息。

关于密码加密的方式有很多，其中 MD5 是一种非常普遍的方式。由于 MD5 算法的不可逆性，通过计算后的结果将无法还原出计算前的内容，而对于完全相同的内容，其计算后的结果也是相同的，因此 MD5 算法可用于密码的存储。但这种方式也有缺点，如果将世界上所有的密码与计算结果全部存储起来，进行检索，则密码就会被逆向查找出来。为此，通常会为每个用户生成一个 salt（密码盐），用于在对密码进行 MD5 计算时加入 salt，增加破解难度。

下面的代码演示了基于 MD5 与 salt 的密码加密方式。

```
$salt = substr(uniqid(), -6);            // 生成 6 位密钥
$password = md5(md5('密码') . $salt);    // 加密密码
```

上述代码中，uniqid()函数用于根据当前微秒时间生成不重复的字符串。生成后，使用 substr()函数截取出了后 6 位字符作为 salt 字段。在对密码进行第 1 次 md5()运算后，拼接 salt 进行第 2 次 md5()运算，形成了用于保存到数据库中的 password 字段结果。

因此，当需要验证用户输入的密码是否正确时，将给定密码与数据库中保存的 salt 按照加密时的算法进行计算，通过比较计算结果即可确定给定的密码是否正确。

在了解密码加密算法后，接下来在 user 表中插入一条用户记录，具体 SQL 如下。

```
INSERT INTO `fun_user` VALUES
(1, 'admin', 'admin', 'admin@example.com', MD5(CONCAT(MD5('123456'),
'a9E3iN')), 'a9E3iN', '', UNIX_TIMESTAMP());
```

在上述 SQL 中，用户的 id 为 1，用户组为 admin（表示网站管理员），用户名为 admin，用户的邮箱为 admin@example.com，密码为 123456，salt 为 a9E3iN，头像为空，注册时间为当前时间戳。

实现用户登录

（1）显示登录页面

创建 login.php 用于显示用户登录页面，具体代码如下。

```
<?php
require './common/init.php';
require './view/login.html';
```

编写 view\login.html 文件，创建一个用户登录表单，关键代码如下。

```
<form method="post">
  用户名 <input type="text" name="name">
  密　码 <input type="password" name="password">
  <input type="submit" value="登录">
</form>
```

（2）接收表单

完成表单的创建后，就可以在 login.php 中接收表单进行处理。考虑到程序的安全性和严谨性，下面先在 common\function.php 文件中编写用于验证用户名和密码格式的函数，代码如下。

```
function input_check($field, $data, &$msg = '')
{
    switch ($field) {
        case 'user_name':
            $msg = '2~10 位中文、字母、数字、下划线。';
            return (bool)preg_match('/^[\w\x{4E00}-\x{9FA5}]{2,10}$/u', $data);
        case 'user_password':
            $msg = '6~12 位字母、数字、下划线。';
            return (bool)preg_match('/^\w{6,12}$/', $data);
    }
}
```

上述代码中，$field 表示待验证的字段；$data 表示待验证的内容；$msg 用于保存格式信息，当验证失败时可以用于输出，以提醒用户更正格式问题。第 6 行正则表达式中的“\x{4E00}-\x{9FA5}”用于匹配中文汉字。

为了判断当前是否有表单提交，在 common\init.php 中定义常量 IS_POST，具体代码如下。

```
define('IS_POST', ($_POST || $_FILES));
```

上述代码根据$_POST 和$_FILES 数组是否为空，来判断当前是否有表单提交。

继续编写 login.php，在载入 HTML 模板之前，先接收表单进行处理，具体代码如下。

```
if (IS_POST) {
    $name = input('post', 'name', 's');
    $password = input('post', 'password', 's');
    if (!input_check('user_name', $name, $error)) {
        exit("登录失败，用户名格式有误，要求：$error");
    }
    if (!input_check('user_password', $password, $error)) {
```

```
        exit("登录失败，密码格式有误，要求：$error");
    }
    // 到数据库中根据用户名查询密码，判断密码是否正确……
}
```

从上述代码可以看出，当程序中的每一步验证失败时，就会停止脚本并输出提示信息。

（3）友好提示登录失败信息

在登录过程中遇到失败的情况时，为了增强用户体验，应该给用户提供再次登录的机会，并弹出一个提示框告知失败原因。修改 login.php，将载入 HTML 模板的代码封装到 display()函数中，如下所示。

```
function display($tips = null, $name = '')
{
    require './view/login.html';
    exit;
}
```

在上述代码中，参数$tips 表示提示信息；参数$name 表示在登录表单中，用户名输入框默认显示的值，用于在登录失败时保留上次输入的内容。

在完成 display()函数后，下面演示该函数的使用，示例代码如下。

```
// 在登录失败时，显示登录页面并弹出提示信息
display('登录失败，验证码输入有误。', $name);
// 没有表单提交时，显示登录页面
display();
```

接下来，修改 view\login.html，为填写用户名的文本框添加默认值，如下所示。

```
用户名 <input type="text" name="name" value="<?=htmlspecialchars($name)?>">
```

为了在页面中显示提示信息，下面将所有 HTML 模板中相同的页脚部分抽取到 view\common 目录下的 footer.html 文件中，然后在 footer.html 中判断$tips 变量，如果有值则输出提示信息，代码如下。

```
<?php if(!empty($tips)): ?>
  <script>alert(<?=json_encode($tips)?>);</script>
<?php endif; ?>
```

（4）判断用户名和密码是否正确

继续编写 login.php，判断表单提交的用户名和密码是否正确。具体代码如下。

```
// 到数据库中根据用户名查询密码，判断密码是否正确
if (!login_form($name, $password, $error)) {
    display("登录失败，$error");
}
```

编写 login_form()函数，具体代码如下。

```
function login_form($name, $password, &$error = '')
{
    // 根据用户名取出密码
    $data = Db::getInstance()->find('__USER__', 'id,group,name,password,salt',
            's', ['name' => $name]);
    // 判断密码是否正确
    if ($data && (password($password, $data['salt']) == $data['password'])) {
        login_success($data['id'], $data['name'], $data['group']);
    }
    $error = '用户名或密码错误。';
```

```
11 }
```

上述第 7 行代码在根据给定的用户名取出数据库中保存的密码后，调用了 password()函数对输入的密码进行加密，然后比较加密结果与数据库中保存的结果是否一致。

在 common\function.php 中编写 password()函数，代码如下。

```
1  function password($password, $salt)
2  {
3      return md5(md5($password) . $salt);
4  }
```

当密码验证通过后，就会调用 login_success()函数保存当前的登录状态，该函数的代码如下。

```
1  function login_success($id, $name, $group)
2  {
3      // 利用 Session 保存登录状态
4      $_SESSION['fun']['user'] = ['id' => $id, 'name' => $name, 'group' => $group];
5      // 跳转到首页
6      redirect('index.php');
7  }
```

上述第 4 行代码用于将当前登录的用户 id、用户名和用户组保存到 Session 中；第 6 行代码调用了 redirect()函数用于实现页面跳转。在 common\function.php 中编写 redirect()函数，具体代码如下。

```
1  function redirect($url)
2  {
3      header("Location: $url");
4      exit;
5  }
```

从上述代码可以看出，当页面跳转后，就会通过第 4 行代码停止脚本继续执行。

获取登录信息

当 login_success()函数将用户的登录状态保存到了 Session 中以后，就可以在每个功能脚本中判断用户当前是否已经登录。在 common\init.php 中编写用于检查用户登录的代码，具体如下。

```
1  // 检查用户登录
2  define('IS_LOGIN', isset($_SESSION['fun']['user']));
3  // 如果用户已经登录，取出用户信息
4  IS_LOGIN && user(null, $_SESSION['fun']['user']);
```

上述第 4 行代码用于在用户登录后，将 Session 中的用户信息保存到 user()函数中。

下面在 common\function.php 文件中编写 user()函数，具体代码如下。

```
1  function user($name, $user = null)
2  {
3      static $data = null;
4      return $user ? ($data = $user) : $data[$name];
5  }
```

从上述代码可以看出，当 user()函数的第 2 个参数不为空时，表示将第 2 个参数保存到 user()函数中的静态变量$data 中；若省略第 2 个参数，则表示根据第 1 个参数获取用户信息，示例代码如下。

```
user('id');          // 获取当前登录用户的 id
user('name');        // 获取当前登录用户的用户名
```

```
user('group');      // 获取当前登录用户的用户组（admin、user）
```

为了确定当前登录用户是否为网站管理员，在 common\init.php 中定义常量 IS_ADMIN，如下所示。

```
define('IS_ADMIN', IS_LOGIN && user('group') == 'admin');
```

从上述代码可以看出，如果当前处于用户已登录的状态，且当前用户的用户组为“admin”，就表示当前用户是网站的管理员。在后面的代码中，通过 IS_ADMIN 常量可以方便地进行管理员身份验证。

接下来，编写 view\common\top.html 文件，在未登录状态下显示“登录”和“注册”链接；在已登录状态下显示当前登录的用户名，并提供“退出”链接。具体代码如下。

```
<?php if(IS_LOGIN): ?>
  欢迎您，<a href="user.php"><?=user('name')?></a> | <a href="#">退出</a>
<?php else: ?>
    <a href="login.php">登录</a> | <a href="register.php">注册</a>
<?php endif; ?>
```

用户退出

用户退出功能的实现较为简单，这里通过为 login.php 传递 URL 参数 action=logout 来实现。首先打开 view\common\top.html 文件，修改用户登录后显示的“退出”链接，如下所示。

```
<a href="login.php?action=logout">退出</a>
```

然后在 login.php 中接收 action 参数，如果值为 logout，表示用户需要退出，清除 Session 中保存的用户信息即可，具体代码如下。

```
$action = input('get', 'action', 's');
if ($action == 'logout') {
    unset($_SESSION['fun']['user']);
    redirect('./');   // 跳转到首页
}
```

验证码

在开发用户登录功能时，需要考虑一个问题，就是除了浏览器，其他软件也可以向服务器提交数据。从系统安全角度看，如果软件自动大批量向服务器提交表单，那么网站管理员的密码将会被穷举，导致管理员账号被盗取。为此，验证码就成为了一种防御手段。

通常情况下，验证码是一张带有文字的图片，要求用户输入图片中的文字。对于图片中的文字，人类识别非常容易，而软件识别则非常困难。因此，验证码是一种区分由人类还是由计算机操作的程序。

生成验证码文本

在项目中创建 common\library\Captcha.php 文件，用于保存验证码类。在类中编写一个 create()方法用于验证码文本的生成，具体代码如下。

```
class Captcha
{
    // 生成验证码文本，参数$count 表示验证码的位数
    public static function create($count = 5)
```

```
5     {
6         $charset = 'ABCDEFGHJKLMNPQRSTUVWXYZ23456789';  // 随机因子
7         $code = '';
8         $len = strlen($charset) - 1;
9         for ($i = 0; $i < $count; ++$i) {
10            $code .= $charset[mt_rand(0, $len)];
11        }
12        return $code;
13    }
14 }
```

在上述代码中，第 8~11 行用于从$charset 字符串中随机取出$count 个字符，保存到$code 中。该方法执行后，返回生成的验证码文本。

生成验证码图像

通过 PHP 提供的 GD 库扩展，可以绘制一张带有文字的图片。在 Captcha 类中编写 show()方法，实现根据指定文本输出验证码图像，具体代码如下。

```
1  // 输出验证码图像，参数$code 表示验证码字符串，$x 和$y 表示图像宽高像素值
2  public static function show($code, $x = 250, $y = 62)
3  {
4      // 创建图像资源，随机生成背景颜色
5      $im = imagecreate($x, $y);
6      imagecolorallocate($im, mt_rand(50, 200), mt_rand(0, 155), mt_rand(0, 155));
7      // 设置验证码文本的颜色和字体
8      $fontcolor = imagecolorallocate($im, 255, 255, 255);
9      $fontfile = realpath('./common/library/fonts/captcha.ttf');
10     // 在图像中绘制验证码
11     for ($i = 0, $len = strlen($code); $i < $len; ++$i) {
12         imagettftext($im,                          // 图像资源
13             30,                                    // 字符尺寸
14             mt_rand(0, 20) - mt_rand(0, 25),       // 随机设置字符倾斜角度
15             32 + $i * 40, mt_rand(30, 50),         // 随机设置字符坐标
16             $fontcolor,                            // 字符颜色
17             $fontfile,                             // 字符样式
18             $code[$i]                              // 字符内容
19         );
20     }
21     // 添加 8 条干扰线，颜色、位置随机
22     for ($i = 0; $i < 8; ++$i) {
23         $linecolor = imagecolorallocate($im, mt_rand(0, 255), mt_rand(0, 255),
24                         mt_rand(0, 255));
25         imageline($im, mt_rand(0, $x), 0, mt_rand(0, $x), $y, $linecolor);
26     }
27     // 添加 250 个噪点，位置随机
```

```
28    for ($i = 0; $i < 250; ++$i) {
29        imagesetpixel($im, mt_rand(0, $x), mt_rand(0, $y), $fontcolor);
30    }
31    header('Content-Type: image/png');    // 设置消息头
32    imagepng($im);                        // 输出图片
33    imagedestroy($im);                    // 释放图像资源
34 }
```

上述代码实现了将验证码文本按照随机倾斜的角度和坐标写入到图像中，并添加了干扰线和噪点。在图像生成后，通过 header()函数告知浏览器当前内容是一张 PNG 图片。

输出验证码图像

在完成了验证码类的编写后，接下来创建 captcha.php 文件，实现验证码图像的输出，代码如下。

```
1  <?php
2  require './common/init.php';
3  require './common/library/Captcha.php';
4  $code = Captcha::create();      // 生成验证码
5  captcha_save($code);            // 将验证码保存到 Session 中
6  Captcha::show($code);           // 输出验证码
```

上述代码第 5 行通过调用 captcha_save()函数将验证码保存到 Session 中。在 common\function.php 中编写 catpcha_save()函数，具体代码如下。

```
1  // 将参数$code 保存到 Session
2  function captcha_save($code)
3  {
4      $_SESSION['fun']['captcha'] = $code;
5  }
```

修改 view\login.html 文件，在登录表单中添加输入验证码的文本框和验证码图像，代码如下。

```
输入验证码：<input type="text" name="captcha">
验证码图像：<img src="captcha.php">
```

上述代码保存后，在浏览器中访问用户登录页面，即可看到验证码效果。

验证码的验证

当用户提交表单后，在判断用户名和密码之前，应该先判断验证码是否正确，如果不正确则停止登录。接下来在 common\function.php 文件中编写 captcha_check()函数用于验证码的验证，具体如下。

```
1  // 对验证码$code 进行验证
2  function captcha_check($code)
3  {
4      $captcha = isset($_SESSION['fun']['captcha']) ?
5                 $_SESSION['fun']['captcha'] : '';
6      if (!empty($captcha)) {
7          unset($_SESSION['fun']['captcha']);  // 清除验证码，防止重复验证
8          return strtoupper($captcha) == strtoupper($code);  // 不区分大小写
```

```
9       }
10      return false;
11  }
```

完成上述代码后，就可以在 login.php 中调用 captcha_check()函数。当用户提交表单后，在判断用户名和密码之前，应该先判断验证码是否正确，代码如下。

```
1  if (!captcha_check(input('post', 'captcha', 's'))) {
2      display('登录失败，验证码输入有误。', $name);
3  }
```

接下来，通过浏览器访问进行测试，判断验证码功能是否能够正确进行验证。

用户注册和记住登录状态

用户注册

创建 register.php 显示用户注册页面，具体代码如下。

```
1  <?php
2  require './common/init.php';
3  display();
4  function display($tips = null, $name = '', $email = '', $type = '')
5  {
6      require './view/register.html';
7      exit;
8  }
```

上述代码的 display()函数用于显示页面并退出，参数$name 和$email 分别表示注册表单中默认显示的用户名和电子邮箱。

编写 view\register.html 文件，创建一个用户注册表单，关键代码如下。

```
1  <form method="post">
2    用户名 <input type="text" name="name" value="<?=htmlspecialchars($name)?>"
3          placeholder="2~10 位中文、字母、数字、下划线" required>
4    密码 <input type="password" name="password"
5        placeholder="6~12 位字母、数字、下划线" required>
6    确认密码 <input type="password" placeholder="再次输入密码进行确认" required>
7    邮箱 <input type="text" name="email" value="<?=htmlspecialchars($email)?>"
8        placeholder="输入有效的邮箱地址" required>
9    验证码 <input type="text" name="captcha" required>
10         <img src="captcha.php">
11   <input type="submit" value="注册">
12 </form>
```

完成表单的创建后，就可以接收表单进行处理。接下来，在 register.php 中的 display()函数调用之前添加如下代码，实现表单的接收、验证和处理。

```
1  if (IS_POST) {
2      $name = input('post', 'name', 's');
3      $password = input('post', 'password', 's');
```

```
    $email = input('post', 'email', 's');
    if (!captcha_check(input('post', 'captcha', 's'))) {
        display('注册失败，验证码输入有误。');
    }
    if (!input_check('user_name', $name, $error)) {
        display("注册失败，用户名格式有误，要求：$error");
    }
    if (!input_check('user_password', $password, $error)) {
        display("注册失败，密码格式有误，要求：$error");
    }
    if (!input_check('user_email', $email)) {
        display('注册失败，邮箱格式有误。');
    }
    if (!register($name, $password, $email, $error)) {
        display("注册失败，$error", $name, $email);
    }
}
```

在上述代码中，第 14~16 行用于验证邮箱格式，在 input_check()函数的 switch 语句中增加邮箱格式验证的代码，具体如下。

```
case 'user_email':
    return (bool)preg_match('/^(\w+(\_|\-|\.)*)+@(\w+(\-)?)+(\.\w{2,})+$/', $data);
```

在验证成功后，调用 register()函数实现新用户的注册。register()函数的具体代码如下。

```
function register($name, $password, $email, &$error = '')
{
    $db = Db::getInstance();
    if ($db->value('__USER__', 'id', 's', ['name' => $name])) {
        $error = '用户名已经存在。';
        return false;
    }
    $salt = substr(uniqid(), -6);
    $data = ['group' => 'user', 'name' => $name, 'password' => password($password,
            $salt), 'salt' => $salt, 'email' => $email, 'time' => time()];
    if (!$id = $db->insert('__USER__', 'sssssi', $data)) {
        $error = '数据库保存失败。';
        return false;
    }
    register_success($id, $name, 'user');
}
```

上述代码中，第 4~7 行用于判断用户名是否已经存在，由于用户名是登录的凭据，因此不允许出现相同的用户名；第 8 行用于生成用于加密密码的盐；第 9~14 行将新用户的相关数据保存到数据库中。当用户注册成功后，通过第 15 行代码调用 register_success()函数，将用户信息保存到 Session，然后跳转到首页，该函数的具体代码如下。

```
function register_success($id, $name, $group)
{
```

```
3      $_SESSION['fun']['user'] = ['id' => $id, 'name' => $name, 'group' => $group];
4      redirect('index.php');
5  }
```

记住登录状态

为了增强用户体验，大部分网站在用户登录的表单中提供“下次自动登录”或“记住登录状态”的复选框，当用户选中后进行登录，服务器就会利用 Cookie 保存用户的登录信息。在保存后，如果 Session 已经过期，则可以利用 Cookie 进行登录。

下面在 view\login.html 中增加一个“记住登录状态”的复选框，代码如下。

```
<input type="checkbox" name="auto" value="1"> 记住登录状态
```

当用户选中复选框后，处理登录表单时，如果登录成功，将用户名和密码保存到 Cookie 中，用于在 Session 过期后进行登录验证。需要注意的是，由于 Cookie 安全性差，网站不应将用户的密码明文存储在 Cookie 中。下面在 common\config.php 中添加如下配置，用于提高安全性。

```
1  // 加密 Cookie 的密钥
2  'AUTOLOGIN_SERCET' => 'df9jEn3HdpSoe',
3  // 默认保存时间（30 天）
4  'AUTOLOGIN_EXPIRES' => 3600 * 24 * 30,
```

在 login.php 中处理登录表单时，当密码验证成功后，在调用 login_success()函数之前，判断用户是否选中了表单中的“记住登录状态”复选框，如果选中了，将用户名和加密后的密码保存到 Cookie 中，具体代码如下。

```
1  if (input('post', 'auto', 's')) {
2      $expires = time() + config('AUTOLOGIN_EXPIRES');
3      setcookie('fun_auto_name', $data['name'], $expires);
4      setcookie('fun_auto_pass', autologin_cookie($data['password']), $expires);
5  }
```

在上述代码中，第 4 行的 autologin_cookie()函数用于将数据库中已经保存的密码再次进行加密，用于限制 Cookie 密码的有效时间。在 common\function.php 中编写 autologin_cookie()函数，代码如下。

```
1  function autologin_cookie($data)
2  {
3      $time = time();
4      return $time . '|' . password($data, $time . config('AUTOLOGIN_SERCET'));
5  }
```

上述代码将当前时间戳与加密后的 Cookie 密码通过“|”分隔拼接成一个字符串，在调用 password()函数时，将数据库保存的密码$data、当前时间戳$time 和配置文件中的 Cookie 密钥一起进行运算。其中，“|”前面的$time 用于在验证 Cookie 密码时判断当前是否在有效期内，而 password()函数参数中的$time 用于确保“|”前面的$time 无法被伪造。

下面演示一个生成后的 Cookie 加密结果。当网站修改了 AUTOLOGIN_SERCET 配置，或“|”前面的时间戳超过了当前时间减去 AUTOLOGIN_EXPIRES 时，Cookie 密码将失效。

```
1493100144|6681659860efdec0b08f42942c0cb7e4
```

为了验证 Cookie 密码的有效性，接下来编写 autologin_check()函数，代码如下。

```
1  function autologin_check($cookie, $data)
2  {
3      $arr = explode('|', $cookie, 2);
```

```
4     return isset($arr[1]) && (time() - config('AUTOLOGIN_EXPIRES') < (int)$arr[0])
5             && (password($data, $arr[0] . config('AUTOLOGIN_SERCET')) == $arr[1]);
6 }
```

在上述代码中，函数的参数$cookie 表示来自 Cookie 中保存的密码，$data 是数据库中保存的密码。首先通过第 3 行代码根据“|”分割字符串，然后在第 4 行代码判断$arr 数组是否有 2 个元素。如果有，再判断当前时间戳减去有效时间后是否超过了 Cookie 中的时间戳，超过即表示 Cookie 密码已过期。如果时间戳没有过期，继续利用数据库中的密码判断 Cookie 中的时间戳和 Cookie 密钥加密后的结果是否与 Cookie 中的密码相同，只有相同的情况下，才能保证这个 Cookie 密码是有效的。

接下来，编写 login.php，判断当前用户是否已经在 Cookie 中保存了登录状态，代码如下。

```
define('IS_AUTOLOGIN', isset($_COOKIE['fun_auto_name']));
```

当用户保存了登录状态时，在 view\login.html 中显示已保存的用户信息，代码如下。

```
1 <?php if(IS_AUTOLOGIN): ?>
2   用户名 <input type="text" value="<?=htmlspecialchars(input('cookie',
3    'fun_auto_name', 's'))?>" disabled>
4   密码 <input type="text" value="已保存，可直接登录" disabled>
5 <?php endif; ?>
```

上述代码第 2 行通过 input()函数获取$_COOKIE 中保存的信息，需要在该函数中增加针对 Cookie 的处理，如下所示。

```
switch ($method) {
    // case ……（get、post）
    case 'cookie': $method = $_COOKIE; break;  // 新增$_COOKIE 数组的支持
}
```

需要注意的是，通过 Cookie 登录时，如果不需要验证码，则验证码形同虚设。因此，当用户通过表单登录时，限制用户即使保存了登录状态，也需要在表单中输入验证码才能进行登录。下面在 login.php 中编写代码处理来自 Cookie 的用户登录，具体如下。

```
1  if (IS_POST) {
2      if (IS_AUTOLOGIN) {
3          // 提交表单后，通过 Cookie 登录
4          $name = input('cookie', 'fun_auto_name', 's');
5          $pass = input('cookie', 'fun_auto_pass', 's');
6          if (!captcha_check(input('post', 'captcha', 's'))) {
7              display('登录失败，验证码输入有误。', $name);
8          }
9          if (!input_check('user_name', $name, $error)) {
10             display("登录失败，用户名格式有误，要求：$error", $name);
11         }
12         if (!login_cookie($name, $pass, $error)) {
13             display("登录失败：$error", $name);
14         }
15     } else {
16         // 通过表单登录……
17     }
18 }
```

上述代码第 12 行调用了 login_cookie()函数，该函数的具体代码如下。

```
1  function login_cookie($name, $pass, &$error = '')
2  {
3      $data = Db::getInstance()->find('__USER__', 'id,group,name,password',
4              's', ['name' => $name]);
5      if ($data && autologin_check($pass, $data['password'])) {
6          login_success($data['id'], $data['name'], $data['group']);
7      }
8      $error = '保存的登录信息无效，请重新登录。';
9      setcookie('fun_auto_name', '', -1);
10     setcookie('fun_auto_pass', '', -1);
11 }
```

通过上述代码可以看出，若 Cookie 中的密码验证成功，则调用 login_success()函数将用户信息保存到 Session 中。如果密码验证失败，则清除 Cookie，要求用户重新登录。

另外，“记住登录状态”功能通常是提供给在登录状态下直接关闭浏览器的用户，而若用户单击了网页中的“退出”链接，则表示用户需要完全退出。因此，下面修改 login.php 中实现用户退出的代码，在清除 Session 中的用户信息后，再清除 Cookie 中保存的登录信息，具体代码如下。

```
1  setcookie('fun_auto_name', '', -1);
2  setcookie('fun_auto_pass', '', -1);
```

用户头像上传

在用户登录的状态下，单击网页顶部显示的用户名之后，就会来到用户个人中心，用户可以设置自己的头像。下面编写 user.php，实现在用户登录的情况下显示上传头像的页面，具体代码如下。

```
1  <?php
2  require './common/init.php';
3  require './common/library/Upload.php';
4  if (!IS_LOGIN) {
5      redirect('login.php');      // 若用户没有登录，跳转到登录页面
6  }
7  display();
8  function display($tips = null, $type = '')
9  {
10     $avatar = Db::getInstance()->value('__USER__', 'avatar',
11             'i', ['id' => user('id')]);
12     require './view/user.html';
13     exit;
14 }
```

上述代码中，第 10~11 行用于根据当前登录用户的 id 查询用户头像，保存到$avatar 变量中。

编写显示当前用户头像和上传头像的页面 view\user.html，具体代码如下。

```
1  <!-- 当前头像 -->
2  <?php if($avatar): ?>
3      <img src="./uploads/avatar/<?=$avatar?>">
4    <?php else: ?>
5      <img src="./images/noavatar.gif">
```

```
6    <?php endif; ?>
7  <!-- 设置新头像-->
8  <form method="post" enctype="multipart/form-data">
9    <input type="file" name="avatar" required>
10   <input type="submit" value="上传头像">
11 </form>
```

从上述代码可以看出，当用户头像$avatar 值不为空时，显示“./uploads/avatar/”中保存的用户头像，否则显示默认头像“./images/noavatar.gif”。

继续编写 user.php，实现接收用户上传头像的表单，具体代码如下。

```
1  if (IS_POST) {
2      // 上传头像
3      if (!$avatar = user_avatar_upload(input('file', 'avatar', 'a'), $error)) {
4          display("头像上传失败，$error");
5      }
6      // 删除原来的头像
7      $db = Db::getInstance();
8      $avatar_old = './uploads/avatar/' . $db->value('__USER__', 'avatar',
9                    'i', ['id' => user('id')]);
10     is_file($avatar_old) && unlink($avatar_old);
11     // 更新头像记录
12     $db->update('__USER__', 'si', ['avatar' => $avatar, 'id' => user('id')], 'id');
13 }
```

上述第 3 行代码调用了 input()函数，用于获取$_FILES 数组中的信息，需要在该函数中增加对$_FILES 数组的支持，具体如下。

```
switch ($method) {
    // case …… （get、post、cookie）
    case 'file': $method = $_FILES; break;       // 新增$_FILES 数组的支持
}
```

编写 user_avatar_upload()函数，用于实现头像文件上传，该函数的代码如下。

```
1  function user_avatar_upload($file, &$error = '')
2  {
3      $upload_dir = './uploads/temp';
4      $up = new Upload($file, $upload_dir, '', ['jpg', 'jpeg', 'png']);
5      if(!$result = $up->uploadOne()) {
6          $error = $up->getError() ?: '没有上传文件。';
7          return false;
8      }
9      $new_dir = date('Y-m/d');
10     $avatar_dir = "./uploads/avatar/$new_dir";
11     // 创建保存目录
12     if (!is_dir($avatar_dir) && !mkdir($avatar_dir, 0777, true)) {
13         $error = '无法创建头像保存目录! ';
14         return false;
15     }
```

```
16      $upload_file = "$upload_dir/$result";
17      thumb($upload_file, "$avatar_dir/$result", 120);
18      is_file($upload_file) && unlink($upload_file);
19      return "$new_dir/$result";
20  }
```

在上述代码中，第 4 行实例化了文件上传类 Upload，第 5 行调用了 uploadOne()方法上传单个文件，上传成功后的返回值为自动生成的文件名；第 6 行用于当上传文件失败时获取错误信息。

通过浏览器访问测试，查看用户头像上传完成后的效果。

栏目管理

设计栏目表

在本项目中，栏目是对内容的分类，用户在发布内容时可以选择其所属的栏目。由于栏目会在网页的右侧边栏显示。为了显示效果，网站管理员可以为栏目设置显示图片。栏目表的结构如下表所示。

字段	数据类型	说明
id	INT UNSIGNED PRIMARY KEY AUTO_INCREMENT	栏目 id
name	VARCHAR(12) DEFAULT '' NOT NULL	栏目名称
cover	VARCHAR(255) DEFAULT '' NOT NULL	图片地址
sort	INT DEFAULT 0 NOT NULL	显示顺序

在上表中，sort 字段表示栏目的排序值，数值小的排在前面，数值大的排在后面。

接下来，在栏目表中插入测试数据，具体如下。

```
INSERT INTO `fun_category` (`id`, `name`, `sort`) VALUES
(1, 'ThinkPHP', 0), (2, 'Bootstrap', 1), (3, 'Laravel', 2),
(4, '小道消息', 3), (5, '嘿科技', 4), (6, '趣快报', 5),
(7, '歪果趣闻', 6), (9, '神吐槽', 7), (8, '涨姿势', 8);
```

栏目数据的读取和显示

在 HTML 模板中，侧边栏在首页（index.php）、内容查看页（show.php）、内容发布页（post.php）以及栏目编辑页（category.php）都需要显示，因此将获取栏目数据的代码封装成 category_list()函数保存在 common\function.php 中，具体如下。

```
1  function category_list()
2  {
3      static $data = [];
4      return $data ?: ($data = Db::getInstance()->fetchAll('SELECT
5              `id`,`name`,`cover`,`sort` FROM __CATEGORY__ ORDER BY `sort` ASC'));
6  }
```

接下来，编写栏目编辑页 category.php，具体代码如下。

```
1  <?php
2  require './common/init.php';
```

```
require './common/library/Upload.php';
if (!IS_ADMIN) {
    exit('您无权访问。<a href="./">返回首页</a>');
}
display();
function display($tips = null, $type = '')
{
    $category = category_list();
    require './view/category.html';
    exit;
}
```

上述代码第 10 行通过调用 category_list()函数获取了栏目数据。在侧边栏中输出栏目数据之前，先将侧边栏的 HTML 代码提取出来，保存到 view\common\slide.html 文件中，该文件的代码如下。

```
栏目推荐
<?php if(IS_ADMIN): ?><a href="category.php">[编辑]</a><?php endif; ?>
<!-- 输出有图片的栏目 -->
<?php foreach ($category as $k => $v): if ($v['cover']): ?>
  <a href="./?cid=<?=$v['id']?>" title="<?=$v['name']?>">
  <img src="./uploads/category/<?=$v['cover']?>"></a>
<?php endif; endforeach; ?>
<!-- 输出无图的栏目 -->
<?php foreach ($category as $k => $v): if (!$v['cover']): ?>
  <a href="./?cid=<?=$v['id']?>"><?=$v['name']?></a>
<?php endif; endforeach; ?>
```

完成侧边栏的输出后，编写栏目管理页面 view\category.html，代码如下。

```
<form method="post" enctype="multipart/form-data">
  <!-- 修改栏目 -->
  <?php foreach ($category as $v): ?>
  删除 <input type="checkbox" name="del[]" value="<?=$v['id']?>">
  排序 <input type="text" name="save[<?=$v['id']?>][sort]" value="<?=$v['sort']?>">
  名称 <input type="text" name="save[<?=$v['id']?>][name]"
      value="<?=htmlspecialchars($v['name'])?>" required>
  上传封面 <?php if ($v['cover']): ?>
    <img src="./uploads/category/<?=$v['cover']?>">
    <input type="checkbox" name="del_cover[]" value="<?=$v['id']?>"> 删除
  <?php endif; ?>
  <input type="file" name="save_cover[<?=$v['id']?>]">
  <?php endforeach; ?>
  <!-- 添加栏目 -->
  <span class="js-cate-cancel">[取消]</span>
  排序 <input type="text" name="add[_ID_][sort]" >
  名称 <input type="text" name="add[_ID_][name]" required>
  上传封面 <input type="file" name="add_cover[_ID_]" >
  <span class="js-cate-add">添加栏目</span>
```

```
20   <input type="submit" value="提交修改">
21 </form>
```

上述代码是一个修改栏目的表单，第 2~13 行用于修改栏目，第 14~18 行用于添加栏目。值得一提的是，默认情况下第 14~18 行代码是隐藏且禁用的，因为这部分代码仅用于 JavaScript 进行处理。当用户单击第 19 行的“添加栏目”时，页面中的 JavaScript 程序会读取第 14~18 行代码，将这部分代码复制一份后取消隐藏和取消禁用，并追加到表单中。用户可以单击多次“添加栏目”添加多个，每次复制时，会将 input 元素 name 属性中的占位符“_ID_”替换为数字。该数字从 0 开始，每次添加新栏目后自增 1，从而确保在表单中区分每个新增的栏目。

为了便于理解，下面通过数组赋值语法演示用户提交表单后，$_POST 的数组结构，如下所示。

```
$_POST = [
    'add' => [                  // 添加栏目，新栏目的 id 为自动增长
        0 => ['sort' => '0', 'name' => '新栏目 1'],
        1 => ['sort' => '1', 'name' => '新栏目 2'],
    ],
    'save' => [                 // 修改指定 id 的栏目
        2 => ['sort' => '0', 'name' => '修改栏目 1'];    // 修改 id 为 2 的栏目
        4 => ['sort' => '0', 'name' => '修改栏目 2'];    // 修改 id 为 4 的栏目
    ],
    'del_cover' => [3, 4],      // 删除栏目 id 为 3 和 4 的图片
    'del' => [5, 6]             // 删除栏目 id 为 5 和 6 的记录和图片
];
```

另外，若用户在表单中添加了上传文件，则在$_FILES 数组中可以获取到 add_cover 和 save_cover 两个数组元素。

接下来，在 category.php 中接收表单进行处理。为了便于代码的维护，将添加、修改、删除功能分别封装到 category_add()、category_save()和 category_delete()函数中，如下所示。

```
1  if (IS_POST) {
2      category_add($error);            // 添加栏目
3      $error && display("栏目添加失败，$error");
4      category_save($error);           // 修改栏目
5      $error && display("栏目修改失败，$error");
6      category_delete();               // 删除栏目
7  }
```

栏目添加

编写 category_add()函数实现栏目的添加，具体代码如下。

```
1  function category_add(&$error = '')
2  {
3      $cover = category_cover_upload(input('file', 'add_cover', 'a'), $error);
4      $result = [];
5      foreach (input('post', 'add', 'a') as $k => $v) {
6          $result[] = [
7              'name' => mb_strimwidth(input($v, 'name', 's'), 0, 12),
8              'cover' => input($cover, $k, 's'),
```

```
            'sort' => input($v, 'sort', 'd'),
        ];
    }
    $result && Db::getInstance()->insert('__CATEGORY__', 'ssi', $result);
}
```

在上述代码中，第 3 行调用了 category_cover_upload()函数用于上传图片，第 4~12 行用于搜集表单数据，进行过滤后保存到数据库中。

编写函数 category_cover_upload()实现栏目图片的上传，具体代码如下。

```
function category_cover_upload($file, &$error = '')
{
    $upload_dir = './uploads/category';
    $up = new Upload($file, $upload_dir, date('Y-m/d') , config('PICTURE_EXT'));
    $result = $up->upload();
    $error = $up->getError();
    return $result;
}
```

上述代码实现了将上传图片保存到“./uploads/category”目录中，第 5 行的 upload()方法用于上传多个文件，返回值是一维数组，数组的键为表单中文件上传输入框的 name 属性，其值是“save_cover[键名]”或“add_cover[键名]”中的一种，数组的值为包含子目录和文件名的数组。第 4 行的 config('PICTURE_EXT')表示读取配置文件中允许上传的图片扩展名，下面在 common\config.php 中添加该配置，如下所示。

```
// 允许的图片扩展名（小写）
'PICTURE_EXT' => ['jpg', 'jpeg', 'png', 'gif', 'bmp'],
```

栏目修改

编写 category_save()函数实现栏目的修改，具体代码如下。

```
function category_save(&$error = '')
{
    // 获取待保存数据
    $result = [];
    foreach (input('post', 'save', 'a') as $k => $v) {
        $result[] = [
            'name' => mb_strimwidth(input($v, 'name', 's'), 0, 12),
            'sort' => input($v, 'sort', 'd'),
            'id' => abs($k),      // 通过 abs()函数确保 id 不为负数
        ];
    }
    // 保存记录
    $db = Db::getInstance();
    $result && $db->update('__CATEGORY__', 'sii', $result, 'id');
    $cover = [];
    // 获取待删除图片
    $cover_del = input('post', 'del_cover', 'a');
    foreach ($cover_del as $v) {
```

```
19        $cover[$v] = ['cover' => '', 'id' => abs($v)];
20    }
21    // 获取新上传的图片
22    $cover_save = category_cover_upload(input('file', 'save_cover', 'a'), $error);
23    foreach ($cover_save as $k => $v) {
24        $cover[$k] = ['cover' => $v, 'id' => abs($k)];
25    }
26    if ($cover) {
27        category_cover_delete(array_keys($cover));          // 删除原图文件
28        $db->update('__CATEGORY__', 'si', $cover, 'id');    // 更新图片字段
29    }
30 }
```

上述第 3~14 行代码实现了对栏目记录的修改；第 16~20 行代码用于当用户选中栏目图片上的“删除”复选框时，通过$cover 保存需要删除图片的栏目记录；第 21~25 行代码用于调用 category_cover_upload()函数上传图片并将返回结果保存到$cover 中；第 27 行代码用于根据$cover 中保存的栏目 id 删除该栏目的图片文件；第 28 行用于将栏目图片修改结果保存到数据库中。

下面继续编写 category_cover_delete()函数，实现栏目图片的删除，具体代码如下。

```
1  function category_cover_delete($cover)
2  {
3      $data = Db::getInstance()->fetchAll('SELECT `cover` FROM __CATEGORY__
4              WHERE `id` IN (' . implode(',', $cover) . ')');
5      foreach ($data as $v) {
6          $path = './uploads/category/' . $v['cover'];
7          is_file($path) && unlink($path);
8      }
9  }
```

栏目删除

编写 category_delete()函数实现栏目的删除，具体代码如下。

```
1  function category_delete()
2  {
3      $del = array_map('abs', input('post', 'del', 'a'));
4      if ($del) {
5          category_cover_delete($del);
6          $db = Db::getInstance();
7          $sql_del = implode(',', $del);
8          $db->execute("DELETE FROM __CATEGORY__ WHERE `id` IN ($sql_del) ");
9          // 将 POST 表中的相关记录设为空栏目
10         $db->execute("UPDATE __POST__ SET `cid`=0 WHERE `cid` IN ($sql_del)");
11     }
12 }
```

上述代码实现了接收表单提交的 del 数组，该数组中的每个元素是待删除的栏目 id。第 5~8 行代码实现了先删除栏目图片再删除栏目记录，第 10 行用于将被删除栏目下 POST 表中的记录更改为空栏目。

内容发布与修改

设计数据表

根据项目的需求分析，用于保存用户发布内容的 post 表的结构如下表所示。

字段	数据类型	说明
id	INT UNSIGNED PRIMARY KEY AUTO_INCREMENT	内容 id
cid	INT UNSIGNED DEFAULT 0 NOT NULL	栏目 id
uid	INT UNSIGNED DEFAULT 0 NOT NULL	用户 id
type	ENUM('pic','text','video') DEFAULT 'text' NOT NULL	内容类型
content	TEXT NOT NULL COMMENT	内容文本
time	INT UNSIGNED DEFAULT 0 NOT NULL	发布时间
hits	INT UNSIGNED DEFAULT 0 NOT NULL	阅读量
reply	INT UNSIGNED DEFAULT 0 NOT NULL	回复量

当用户发布内容时，有 pic（趣图）、text（趣文）、video（视频）3 种类型可以选择。其中，pic 类型所包含的图片地址，以及 video 类型所包含的视频地址，将保存到如下表所示的 attachment 表中。

字段	数据类型	说明
id	INT UNSIGNED PRIMARY KEY AUTO_INCREMENT	附件 id
pid	INT UNSIGNED DEFAULT 0 NOT NULL	内容 id
content	VARCHAR(255) DEFAULT '' NOT NULL	附件内容

值得一提的是，对于 pic 类型的内容，在内容发布页面会提供图片上传的功能。而考虑到视频文件的体积比较大，在发布 video 类型的内容时，只能使用第三方视频网站提供的视频链接。

内容发布

在侧边栏中提供内容发布按钮，打开 view\common\slide.html 文件，新增代码如下。

```
发布 <a href="post.php?type=pic">趣图</a>
发布 <a href="post.php?type=text">趣文</a>
发布 <a href="post.php?type=video">视频</a>
```

接下来编写 post.php，该文件有两个功能，当传递 URL 参数“id”时执行添加操作，不存在时执行修改操作，具体代码如下。

```
<?php
require './common/init.php';
require './common/library/Upload.php';
if (!IS_LOGIN) {
    redirect('login.php');
}
$id = input('get', 'id', 'd');
$type = input('get', 'type', 's');
```

```
9  if (!in_array($type, ['pic', 'text', 'video'])) {
10     $type = 'text';
11 }
12 display(null, $id, $type);
```

上述代码第 9~11 行用于限制 URL 参数 type 的值只能是 pic、text 或 video，如果是其他值，则自动修改为 text；第 12 行调用了 display()函数用于显示页面并退出，该函数的代码如下。

```
1  function display($tips = null, $id = 0, $type = 'text')
2  {
3      $atch = [];
4      $post = ['cid' => 0, 'content' => ''];
5      if ($id) {
6          $db = Db::getInstance();
7          $post = $db->find('__POST__', 'cid,uid,type,content', 'i', ['id' => $id]);
8          if (!$post || (!IS_ADMIN && $post['uid'] != user('id'))) {
9              exit('您无权编辑此内容，或内容不存在。');
10         }
11         $type = $post['type'];
12         if ($type == 'pic' || $type == 'video') {
13             $atch = $db->select('__ATTACHMENT__', 'id,content', 'i', ['pid' => $id]);
14         }
15     }
16     $category = category_list();
17     require './view/post.html';
18     exit;
19 }
```

在上述代码中，函数的参数$tips、$id、$type，以及第 3、4、16 行的变量$atch、$post、$category 都是用来在模板中使用的。当$id 的值不为 0 时，执行第 5~15 行代码，根据 id 查找对应的内容，其中第 7 行用于查找 post 记录，保存到变量$post 中；第 8~10 行用于判断当$post 为空，或当前登录用户不是管理员且不是内容作者时，禁止编辑；第 13 行用于查找 attachment 记录，保存到变量$atch 中。

创建表单

编写 view\post.html 文件，创建内容发布和修改的表单，其关键代码如下。

```
1  <form method="post" action="?id=<?=$id?>&type=<?=$type?>"
2    enctype="multipart/form-data">
3    所属栏目：<select name="cid">
4      <option value="0">- 未选择 -</option>
5      <?php foreach ($category as $v): ?>
6        <option value="<?=$v['id']?>" <?=($post['cid']==$v['id']) ?
7        'selected' : ''?>><?=$v['name']?></option>
8      <?php endforeach; ?>
9    </select>
10   <?php if ($type=='pic'): ?>
11     上传图片： <input type="file" name="pic[]" <?=$id ? '' : 'required'?>> [+] [-]
```

```
12    <?php if(!empty($atch)): ?>
13      已上传图片：
14      <?php foreach($atch as $v): ?>
15        <a href="./uploads/picture/<?=$v['content']?>" target="_blank">
16        <img src="./uploads/picture/<?=$v['content']?>"></a>
17        <input type="checkbox" name="del[]" value="<?=$v['id']?>">删除
18      <?php endforeach; ?>
19    <?php endif; ?>
20  <?php endif; ?>
21  <?php if($type == 'video'): ?>
22    <p>链接视频：<span>（支持优酷、腾讯、爱奇艺、bilibili 等主流视频网站）</span></p>
23    <?php foreach (($atch ?: [['id' => 0, 'content' => '']]) as $v): ?>
24      <input type="text" name="video[]" value="<?=htmlspecialchars($v['content'])?>"
25      placeholder="视频嵌入代码" required> [+] [-]
26    <?php endforeach; ?>
27  <?php endif; ?>
28  文字内容： <textarea name="content" placeholder="1000 个字以内">
29          <?=htmlspecialchars($post['content'])?></textarea>
30  <input type="submit" value="<?=$id ? '编辑' : '发布'?>">
31 </form>
```

上述第 3~9 行代码输出了栏目下拉菜单，用户可以通过这个菜单选择当前发布内容的所属栏目；第 10~20 行代码用于发布类型为 pic 时，提供上传图片的输入框；如果当前编辑内容已有上传图片，则通过第 12~19 行代码将图片输出，并提供“删除”复选框来删除该图片；第 21~27 行代码用于输出当前内容关联的视频链接，其中第 23 行代码在使用 foreach 遍历$atch 数组时，确保了当$atch 为空时在页面中输出一个空白的视频链接输入框，用于添加视频链接；第 28~29 行代码用于输入文本内容。

值得一提的是，在上传图片和链接视频的输入框右边，页面提供了“[+]”和“[-]”按钮，单击后会执行 JavaScript 程序来增加或减少输入框，从而实现添加多个图片或视频内容。在增加或减少时，JavaScript 程序会判断当前提供的输入框的数量，确保页面中至少保留一个输入框。

考虑到每个内容所能添加的图片或视频应该是有限的，下面在配置文件中指定数量限制。

```
'APP_ATTACHMENT_MAX' => 10,     // 每个内容允许的附件最大数量
```

添加上述配置后，在表单中增加如下隐藏域，告知页面中的 JavaScript 程序当前允许的附件最大数量值，如下所示。一旦当前图片或视频的输入框达到限制值，将不能继续添加。

```
1  <!-- 用于告知 JavaScript 程序验证 -->
2  <input type="hidden" name="atch_max"
3  value="<?=config('APP_ATTACHMENT_MAX')?>" disabled>
```

接收表单

在 post.php 中接收用户提交的表单，实现数据的添加与修改，具体代码如下。

```
1  if (IS_POST) {
2      $data = [
3          'cid' => input('post', 'cid', 'd'),
4          'content' => mb_strimwidth(input('post', 'content', 's'), 0, 1000),
5          'uid' => user('id'),
```

```
    ];
    $db = Db::getInstance();
    if ($id) {
        // 更新记录，在更新前先验证权限
        $result = $db->find('__POST__', 'uid,type', 'i', ['id' => $id]);
        if (!$result || (!IS_ADMIN && $result['uid'] != user('id'))) {
            exit('您无权编辑此内容，或内容不存在。');
        }
        $type = $result['type'];
        $data['id'] = $id;
        $db->update('__POST__', 'isii', $data, 'id');
        // 删除关联的图片或视频……
    } else {
        // 新增记录
        $data['type'] = $type;
        $data['time'] = time();
        $id = $db->insert('__POST__', 'isisi', $data);
    }
    // 接收图片或视频……
}
```

上述代码中，第 2~6 行用于获取表单数据，第 8~17 行用于修改记录，第 19~22 行用于添加记录。

处理图片和视频

修改内容时删除关联的图片或视频

在编辑内容时，用户可以对关联的图片或视频进行添加、修改、删除操作。对于图片，如果用户选中了“删除”复选框，则删除图片记录和文件；对于视频，为了方便处理，直接将原来的记录全部删除，然后重新添加记录。接下来，在 post.php 中更新记录时对 attachment 表中的内容进行处理。

```
// 删除关联的图片或视频
if ($type == 'pic') {
    $del = array_map('abs', input('post', 'del', 'a'));
    $del && post_picture_delete($id, $del);
} elseif ($type == 'video') {
    $db->delete('__ATTACHMENT__', 'i', ['pid' => $id]);
}
```

上述代码第 4 行调用了 post_picture_delete()函数用于删除图片，该函数的代码如下。

```
function post_picture_delete($pid, $del)
{
    $db = Db::getInstance();
    $where = "`pid`=$pid AND `id` IN (" . implode(',', $del) . ')';
    $data = $db->fetchAll("SELECT `content` FROM __ATTACHMENT__ WHERE $where");
    foreach ($data as $v) {
```

```
7         $path = './uploads/picture/' . $v['content'];
8         is_file($path) && unlink($path);
9     }
10    $db->execute("DELETE FROM __ATTACHMENT__ WHERE $where");
11 }
```

上述代码实现了根据 attachment 表中的 pid 和 id 字段查询记录，查询后删除图片文件，然后再删除数据库中的记录。

接收图片或视频

继续编写 post.php，实现图片的上传和视频链接的接收，代码如下。

```
1  $atch = post_attachment($id, $type, $error);
2  $atch && $db->insert('__ATTACHMENT__', 'si', $atch);
3  $error ? display($error, $id, $type) : redirect("show.php?id=$id");
```

上述代码中，post_attachment()函数用于接收图片或视频，该函数的返回值是一个数组，保存了每个新增图片或视频记录的 content、pid 字段的值。第 3 行代码用于当整个流程没有错误时，跳转到内容查看页面，查看当前编辑后的内容；若流程中发生错误，则显示内容编辑页面并提示错误信息。

接下来编写 post_attachment()函数，具体代码如下。

```
1  function post_attachment($id, $type, &$error = '')
2  {
3      $atch = [];
4      if ($type == 'pic') {
5          $pic = post_picture_upload(input('file', 'pic', 'a'), $error);
6          $error && $error = "图片上传失败：$error";
7          foreach ($pic as $v) {
8              $atch[] = ['content' => $v, 'pid' => $id];
9          }
10     } elseif ($type == 'video') {
11         foreach (input('post', 'video', 'a') as $v) {
12             $v = is_string($v) ? parse_video_url(trim($v)) : '';
13             if (!input_check('post_video', strtolower($v))) {
14                 $error = '链接视频失败，URL 中包含不支持的域名。';
15                 continue;
16             }
17             $atch[] = ['content' => substr($v, 0, 255), 'pid' => $id];
18         }
19     }
20     return $atch;
21 }
22 function parse_video_url($url)
23 {
24     preg_match('/ src=[\'"](.*?)[\'"]/', $url, $match);
25     return isset($match[1]) ? $match[1] : $url;
26 }
```

在上述代码中，$atch 数组用于保存图片文件地址或视频链接，该数组最终将保存到数据库中。第 5~9 行代码用于保存图片，其中第 5 行调用了 post_picture_upload()函数用于上传图片；第 11~18 行代码用于保存视频链接，在从表单中接收后通过第 13 行代码进行格式验证，该验证用于确保用户填写的视频链接是主流视频网站提供的播放器，防止填写恶意网址影响网站的安全。

编写 post_picture_upload()函数实现图片的上传，具体代码如下。

```
1  function post_picture_upload($file, &$error = '')
2  {
3      $up = new Upload($file, './uploads/picture/', date('Y-m/d'),
4            config('PICTURE_EXT'), config('APP_ATTACHMENT_MAX'));
5      $result = $up->upload();
6      $error = $up->getError();
7      return $result;
8  }
```

在上述代码中，实例化 Upload 类时传递了最后一个参数，用于限制文件最多上传的数量。

验证视频地址

为了限制用户在发布视频时只能使用优酷、土豆等主流视频网站，下面在 input_check()函数中增加对于视频 URL 的验证，具体代码如下。

```
    case 'post_video':
        $domain = parse_url($data);
        return isset($domain['host']) && in_array($domain['host'],
                config('APP_VIDEO_ALLOW'));
```

上述代码中的 parse_url()是 PHP 的内置函数，用于解析 URL 地址，其返回值是一个关联数组，数组的元素 host 保存了 URL 中的域名部分。第 4 行从配置文件中读取了域名白名单，具体如下所示。

```
    'APP_VIDEO_ALLOW' => [
        'player.youku.com',          // 优酷
        'v.qq.com',                  // 腾讯
        'open.iqiyi.com',            // 爱奇艺
        'tv.sohu.com',               // 搜狐
        'www.acfun.cn',              // AcFun
        'player.bilibili.com',       // bilibili
    ],
```

上述代码中的域名是在各大视频网站中搜集的，下面演示一些视频网站的外链播放器地址。

优酷：

```
    <iframe height=498 width=510 src='https://player.youku.com/embed/XNDU4ODgyNzczNg==' f
rameborder=0 'allowfullscreen'></iframe>
```

腾讯：

```
    <iframe frameborder="0" src="https://v.qq.com/txp/iframe/player.html?vid=l3148ywitdy"
 allowFullScreen="true"></iframe>
```

爱奇艺：

```
    <iframe src="http://open.iqiyi.com/developer/player_js/coopPlayerIndex.html?vid=64a7f
f438dbe2cc52bbbd619de9bc86e&tvId=32689187009&accessToken=2.ef9c39d6c7f1d5b44768e38e524315
7d&appKey=8c634248790d4343bcae1f66129c1010&appId=1368&height=100%&width=100%" frameborder
```

```
="0" allowfullscreen="true" width="100%" height="100%"></iframe>
```

搜狐：

```
    <iframe frameborder="0" src="https://tv.sohu.com/s/sohuplayer/iplay.html?bid=92716757
&autoplay=true&disablePlaylist=true" allowFullScreen="true" scrolling="no"></iframe>
```

AcFun：

```
    <iframe style="min-width: 500px;min-height: 300px"  src="https://www.acfun.cn/player/
ac14920979" id="ACPlayer-re"  scrolling="no" border="0" frameborder="no" framespacing="0"
allowfullscreen="true"></iframe>
```

bilibili：

```
    <iframe src="//player.bilibili.com/player.html?aid=327989530&bvid=BV1UA4114773&cid=18
3561661&page=1" scrolling="no" border="0" frameborder="no" framespacing="0" allowfullscree
n="true"> </iframe>
```

内容查看

获取查看的内容

编写 show.php 实现指定 id 内容的查看，具体代码如下。

```
 1  <?php
 1  require './common/init.php';
 2  $id = input('get', 'id', 'd');
 3  // 增加阅读计数
 4  $db = Db::getInstance();
 5  $db->execute("UPDATE __POST__ SET `hits`=`hits`+1 WHERE `id`=$id");
 6  // 查询记录
 7  $sql = 'SELECT p.`uid`,p.`type`,p.`content`,p.`time`,p.`hits`,p.`reply`,'
 8      . ' u.`name`,u.`avatar`,c.`name` cname FROM __POST__ p'
 9      . ' LEFT JOIN __USER__ u ON p.`uid`=u.`id`'
 10     . ' LEFT JOIN __CATEGORY__ c ON p.`cid`=c.`id`'
 11     . " WHERE p.`id`=$id LIMIT 1";
 12 if (!$post = $db->fetchRow($sql)) {
 13     exit('您查看的内容不存在。');
 14 }
 15 $type = $post['type'];
 16 // 查询关联的图片或视频
 17 $atch = [];
 18 if ($type == 'pic' || $type == 'video') {
 19     $atch = $db->select('__ATTACHMENT__', 'content', 'i', ['pid' => $id]);
 20 }
 21 $category = category_list();
 22 require './view/show.html';
```

输出到页面中

编写 view\show.html 文件，输出查询到的记录，代码如下。

```
<!-- 当前内容所属栏目 -->
<?=htmlspecialchars($post['cname'])?>
<!-- 作者名称和头像 -->
<img src="<?=$post['avatar'] ? ('./uploads/avatar/' . $post['avatar']) :
 './images/noavatar.gif'?>">
<a href="./?author=<?=$post['uid']?>" target="_blank"><?=$post['name']?></a>
<!-- 发布时间 -->
<?=date('Y-m-d', $post['time'])?>
<!-- 文本内容 -->
<?=nl2br(str_replace(' ', ' ', htmlspecialchars($post['content'])))?>
<!-- 图片或视频 -->
<?php foreach ($atch as $v): ?>
  <?php if ($type == 'pic'): ?>
    <img src="./uploads/picture/<?=$v['content']?>">
  <?php elseif ($type == 'video'): ?>
    <iframe src="<?=htmlspecialchars($v['content'])?>" width="100%" height="335" scrol
ling="no" border="0" frameborder="no" framespacing="0" allowfullscreen="true"></iframe>
  <?php endif; ?>
<?php endforeach; ?>
<!-- 阅读量和回复量 -->
<a href="show.php?id=<?=$id?>" title="阅读量"><?=$post['hits']?></a>
<a href="show.php?id=<?=$id?>#reply" title="回复量"><?=$post['reply']?></a>
<!-- 编辑和删除链接 -->
<?php if (IS_ADMIN || $post['uid'] == user('id')): ?>
  <a href="./post.php?id=<?=$id?>">[编辑]</a>
  <a href="#">[删除]</a>
<?php endif; ?>
```

完成上述代码后，即可将指定 id 的内容，包括栏目名称、作者用户名、作者头像、发布时间、文本内容、图片或视频内容、阅读量和回复量等信息输出。第 23~26 行提供了编辑和删除的链接，该链接只有当前登录用户为网站管理员和内容作者时输出。

内容删除

通过链接进行删除

在内容查看页面 view\show.html 文件中，为“删除”链接添加链接地址，如下所示。

```
<a href="./post.php?action=del&id=<?=$id?>">[删除]</a>
```

然后在 post.php 中接收参数，实现指定 id 内容的删除，代码如下。

```
$action = input('get', 'action', 's');
```

```
2  if ($action == 'del') {
3      $db = Db::getInstance();
4      $result = $db->find('__POST__', 'uid,type', 'i', ['id' => $id]);
5      if (!$result || (!IS_ADMIN && $result['uid'] != user('id'))) {
6          exit('您无权删除此内容，或内容不存在。');
7      }
8      if ($result['type'] == 'pic') {
9          foreach ($db->select('__ATTACHMENT__', 'content', 'i', ['pid' => $id]) as $v) {
10             $path = './uploads/picture/' . $v['content'];
11             is_file($path) && unlink($path);
12         }
13     }
14     $db->delete('__POST__', 'i', ['id' => $id]);
15     $db->delete('__ATTACHMENT__', 'i', ['pid' => $id]);
16     redirect('./');
17 }
```

上述第 4~7 行代码用于根据 id 取出 post 表中的记录后验证权限，只有网站管理员和内容作者可以执行当前的删除操作；第 8~13 行用于当待删除的内容是 pic 类型时，先删除其关联的图片文件；第 14~15 行用于执行 post 表和 attachment 表的删除操作。当删除完成后，通过第 16 行代码返回首页。

解决 CSRF 问题

在前面实现删除功能时，直接通过访问一个 URL 地址就实现了删除数据，这种方式在 Web 开发中存在安全隐患。例如，当管理员在已登录状态下进行其他操作时，若访问了其他用户恶意构造的 URL 地址，就会导致一些危险的操作被执行，这个问题被称为 CSRF（跨站请求伪造）。

防御 CSRF 的一个有效措施，就是为所有涉及更改数据的操作加上令牌保护，该令牌将在用户登录时随机生成，每个更改的操作都需要附加上令牌，如果没有令牌时将无法执行操作。

下面在项目的 common\function.php 文件中添加令牌生成和验证的函数，具体代码如下。

```
1  function token_get()          // 生成令牌
2  {
3      if(isset($_SESSION['fun']['token'])){
4          $token = $_SESSION['fun']['token'];
5      }else{
6          $token = md5(microtime(true));
7          $_SESSION['fun']['token'] = $token;
8      }
9      return $token;
10 }
11 function token_check($token) // 验证令牌
12 {
13     return token_get() === $token;
14 }
```

上述代码中，第 6 行的 md5(microtime(true))用于根据一个精确到微秒的时间生成一个 32 位字符串，生成后通过第 7 行代码保存到 Session 中。

完成令牌的生成和验证函数后，在 common\init.php 中添加如下代码，将令牌保存为 TOKEN 常量。

```
define('TOKEN', token_get());
```

接下来修改 view\show.html 文件，在“删除”链接的 URL 参数中增加 token 参数，如下所示。

```
<a href="./post.php?action=del&id=<?=$id?>&token=<?=TOKEN?>">[删除]</a>
```

完成上述修改后，就可以根据 URL 参数中是否含有 action 参数来确认是否需要令牌验证。需要注意的是，除了 GET 方式，POST 方式也有遭受 CSRF 攻击的风险。因此，接下来在 common\init.php 中针对 POST 方式，和 URL 中包含 action 参数的情况进行令牌验证，具体代码如下。

```
1  if((IS_POST || isset($_GET['action'])) && !token_check(input('get', 'token', 's'))) {
2      exit('操作失败：非法令牌。');
3  }
```

添加令牌验证后，需要修改项目中所有的表单提交地址和带有 action 参数的链接地址，在 URL 中增加 token 参数，如下所示。

```
1  <form method="post" action="?token=<?=TOKEN?>" enctype="multipart/form-data">
2  </form>
```

内容列表

在项目中，网站的首页 index.php 用于显示内容列表，支持排序、分页、按照栏目筛选、按照作者筛选等功能。对于图片和视频内容，还可以取出第一条数据作为预览显示在列表中。

查询内容列表记录

编写 index.php 实现根据 URL 参数查询内容列表，具体代码如下。

```
1  <?php
2  require './common/init.php';
3  $cid = input('get', 'cid', 'd');                    // 根据栏目 id 筛选记录
4  $type = input('get', 'type', 's');                  // 根据类型筛选记录，或排序
5  $author = input('get', 'author', 'd');              // 根据作者筛选记录
6  $page = max(input('get', 'page', 'd'), 1);          // 当前查看的页码
7  if (!array_key_exists($type, config('APP_NAV'))) {
8      $type = 'hot';
9  }
10 // 拼接查询条件和排序
11 $where = 'WHERE 1=1 ';
12 $where .= $author ? "AND `uid`=$author" : ($cid ? "AND `cid`=$cid" : '');
13 $where .= in_array($type, ['pic', 'text', 'video']) ? "AND `type`='$type'" : '';
14 $limit = 'LIMIT ' . page_sql($page, config('APP_PAGESIZE'));
15 $sort = ['hot' => 'ORDER BY p.`reply` DESC,p.`id` DESC', 'new' => 'ORDER BY p.`id` DESC'];
16 $order = isset($sort[$type]) ? $sort[$type] : $sort['hot'];
17 // 获取总记录数
18 $db = Db::getInstance();
19 $total = $db->fetchRow("SELECT COUNT(*) `total` FROM __POST__ $where")['total'];
20 // 查询列表
```

```
$sql = 'SELECT p.`id`,p.`uid`,p.`type`,p.`content`,p.`time`,p.`hits`,p.`reply`,'
    . ' u.`name`,u.`avatar`,c.`name` cname FROM __POST__ p'
    . ' LEFT JOIN __USER__ u ON p.`uid`=u.`id`'
    . ' LEFT JOIN __CATEGORY__ c ON p.`cid`=c.`id`'
    . " $where $order $limit";
$list = $db->fetchAll($sql);
// 查询结果为空时，自动返回第 1 页
if (empty($list) && $page > 1) {
    redirect("index.php?type=$type&cid=$cid&page=1");
}
// 查询预览图或预览视频
foreach ($list as $k => $v) {
    if ($v['type'] == 'pic' || $v['type'] == 'video') {
        $list[$k]['preview'] = $db->value('__ATTACHMENT__', 'content',
                                'i', ['pid' => $v['id']]);
    }
}
$category = category_list();
require './view/index.html';
```

在上述代码中，第 14 行从配置文件中取出了每页显示的记录数，其配置具体如下。

```
'APP_PAGESIZE' => 5,         // 每页显示的记录数
```

输出到页面中

在 view\index.html 文件中输出查询到的记录，具体代码如下。

```
<!-- 显示当前列表的筛选依据（根据用户或栏目进行筛选） -->
<?php if ($author): ?>
  <a href="?author=<?=$author?>">查看用户发表的内容</a>
<?php elseif ($cid): foreach ($category as $v): if ($v['id'] == $cid): ?>
  <a href="?cid=<?=$cid?>"><?=$v['name']?></a>
<?php endif; endforeach; endif; ?>
<?php foreach ($list as $v): ?>
  <!-- 所属的栏目名称 -->
  <?php if ($v['cname']): ?>
    <?=htmlspecialchars($v['cname'])?>
  <?php endif; ?>
  <!-- 作者头像、用户名、发布时间 -->
  <img src="<?=($v['avatar'] ? './uploads/avatar/' . $v['avatar'] :
   './images/noavatar.gif')?>">
  <a href="?author=<?=$v['uid']?>"><?=$v['name']?></a>
  <?=date('Y-m-d', $v['time'])?>
  <!-- 内容预览 -->
  <a href="show.php?id=<?=$v['id']?>" target="_blank">
  <?=htmlspecialchars(mb_strimwidth($v['content'], 0, 200))?>...</a>
```

```
 20   <!-- 预览图或预览视频 -->
 21   <?php if (!empty($v['preview'])): if ($v['type'] == 'pic'): ?>
 22     <a href="show.php?id=<?=$v['id']?>" target="_blank">
 23     <img src="./uploads/picture/<?=$v['preview']?>"></a>
 24   <?php elseif ($v['type'] == 'video'): ?>
 25     <iframe src="<?=htmlspecialchars($v['preview'])?>" width="100%" height="335" scrol
ling="no" border="0" frameborder="no" framespacing="0" allowfullscreen="true"></iframe>
 26   <?php endif; endif; ?>
 27   <!-- 阅读量和回复量 -->
 28   <a href="show.php?id=<?=$v['id']?>" title="阅读量"><?=$v['hits']?></a>
 29   <a href="show.php?id=<?=$v['id']?>#reply" title="回复量"><?=$v['reply']?></a>
 30   <!-- 编辑和删除操作 -->
 31   <?php if (IS_ADMIN || $v['uid'] == user('id')): ?>
 32     <a href="post.php?id=<?=$v['id']?>">[编辑]</a>
 33     <a href="post.php?action=del&id=<?=$v['id']?>&token=<?=TOKEN?>">[删除]</a>
 34   <?php endif; ?>
 35 <?php endforeach; ?>
 36 <!-- 分页导航 -->
 37 <?=page_html("./?type=$type&cid=$cid&page=", $total, $page,
 38 config('APP_PAGESIZE'))?>
```

通过浏览器访问测试，观察运行结果是否正确。

发表回复

设计回复表

在 show.php 内容查看页面，用户可以发表对内容的回复。reply 表的结构如下表所示。

字段	数据类型	说明
id	INT UNSIGNED PRIMARY KEY AUTO_INCREMENT	回复 id
pid	INT UNSIGNED DEFAULT 0 NOT NULL	内容 id
uid	INT UNSIGNED DEFAULT 0 NOT NULL	用户 id
content	VARCHAR(255) DEFAULT '' NOT NULL	回复内容
time	INT UNSIGNED DEFAULT 0 NOT NULL	回复时间

发表回复

在 view\show.html 中添加发表评论的表单，具体代码如下。

```
1  发表评论
2  <form method="post" action="?action=reply&id=<?=$id?>&token=<?=TOKEN?>">
3    <textarea name="reply" placeholder="200 字以内" required></textarea>
4    <input type="submit" value="评论">
```

```
5  </form>
```

上述代码将发表评论后的表单提交到了 show.php 中，并传递了参数 action=reply。接下来，在 show.php 中处理表单，具体代码如下。

```
1  $db = Db::getInstance();
2  if (IS_POST && $action == 'reply') {
3      if (!IS_LOGIN) {
4          redirect('login.php');
5      }
6      $reply = mb_strimwidth(input('post', 'reply', 's'), 0, 200);
7      $result = $db->insert('__REPLY__', 'iisi', ['pid' => $id, 'uid' => user('id'),
8                'content' => $reply, 'time' => time()]);
9      $result && $db->execute("UPDATE __POST__ SET `reply`=`reply`+1 WHERE `id`=$id");
10     redirect("show.php?id=$id#reply");
11 }
```

完成上述代码后，即可实现回复的发表功能。其中，第 9 行代码用于增加被回复内容在 post 表中的回复量统计数。回复成功后，程序跳转到 show.php 页面，并添加锚点“#reply”定位到查看回复的位置。

查看回复

当内容查看页面中有了回复之后，将回复查询出来，并提供分页显示的功能。编辑 show.php，在载入模板之前新增如下代码，分页查询回复记录。

```
1  $total = $db->fetchRow("SELECT COUNT(*) `total` FROM __REPLY__
2            WHERE `pid`=$id")['total'];
3  $limit = 'LIMIT ' . page_sql($page, config('APP_REPLY_PAGESIZE'));
4  $reply = $db->fetchAll('SELECT r.`id`,r.`uid`,r.`content`,r.`time`,'
5      . ' u.`name`,u.`avatar` FROM __REPLY__ r'
6      . ' LEFT JOIN __USER__ u ON r.`uid`=u.`id`'
7      . " WHERE r.`pid`=$id ORDER BY r.`id` DESC $limit");
```

接下来在 view\show.html 中输出回复列表，其关键代码如下。

```
1  <!-- 顶部分页导航 -->
2  <?php $page_html = page_html("show.php?id=$id&page=", $total, $page,
3  config('APP_REPLY_PAGESIZE'), '#reply'); ?>
4  <?php if($page_html): echo $page_html; endif; ?>
5  <!-- 输出回复 -->
6  <?php foreach ($reply as $v): ?>
7    <!-- 显示用户头像、用户名、发表时间 -->
8    <img src="<?=($v['avatar'] ? './uploads/avatar/' . $v['avatar'] :
9    './images/noavatar.gif')?>"><?=$v['name']?>
10   <span><?=date('Y-m-d H:i', $v['time'])?></span>
11   <!-- 回复内容 -->
12   <?=htmlspecialchars($v['content'])?>
13 <?php endforeach; ?>
14 <!-- 底部分页导航 -->
15 <?=$page_html?>
```

完成上述代码后，通过浏览器进行访问测试，观察程序是否执行成功。

删除回复

在 view\show.html 文件中输出回复后，为每个回复内容增加一个删除链接，用于管理员或发表回复的用户删除回复。具体修改如下。

```
<?php if (IS_ADMIN || $v['uid'] == user('id')): ?>
  <a href="?id=<?=$id?>&action=reply_del&del=<?=$v['id']?>&token=<?=TOKEN?>">
  [删除]</a>
<?php endif; ?>
```

接下来，在 show.php 中处理删除回复的请求，具体代码如下。

```
1  if ($action == 'reply_del') {
2      if (!IS_LOGIN) {
3          redirect('login.php');
4      }
5      $del = input('get', 'del', 'd');
6      $result = IS_ADMIN ? $db->delete('__REPLY__', 'i', ['id' => $del]) :
7                $db->delete('__REPLY__', 'ii', ['id' => $del, 'uid' => user('id')]);
8      $result && $db->execute("UPDATE __POST__ SET `reply`=`reply`-1 WHERE `id`=$id
9                  AND `reply` > 0");
10     redirect("show.php?id=$id&page=$page");
11 }
```

上述第 6 行代码在删除回复时先判断是否为管理员，如果是管理员则直接删除指定 id 的回复，否则在删除回复的同时将当前用户 id 作为 WHERE 条件，防止普通用户删除其他其他用户的回复。