表单处理

表单的概念在生活中很常见，就像是问卷调查表一样，别人先把问卷发给你，你照着问卷的要求填写，完事过后再将填完的问卷发给别人，从而达到一个将别人需要的信息传递给别人的一种方式。

传统的网页大多数的作用都是展示数据，就是将信息传递给用户。而在现代化的 Web 开发中，非常注重信息交互，所以表单也随处可见，只是形式上变成网页，性质上还是一模一样的。主要的作用任然是**收集指定的用户信息**。

> 信息交互：例如 简书 这个平台，除了展示文章（展示信息），还可以发布文章（收集信息）

1. 表单基本使用

HTML 中有一个专门用于提交数据的标签：`<form>`，通过这个标签可以很容易的收集用户输入。

> form 标签有两个必要属性：
>
> - action：表单提交地址（填完了，交给谁）
> - method：表单以什么方式提交

例如，我们需要在登录界面上收集用户输入的用户名和密码：

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>登录</title>
</head>
<body>
  <form action="login.php" method="post">
    <div>
      <label for="username">用户名</label>
      <input type="text" id="username" name="username">
    </div>
    <div>
      <label for="password">密码</label>
      <input type="password" id="password" name="password">
    </div>
    <button type="submit">登录</button>
  </form>
</body>
</html>
```

按照目前的情况，用户第一次请求得到这个表单页面，填写完表单内容，点击登录，表单会自动发送到 `login.php`，剩下的问题就是要考虑如何在 `login.php` 中获取到用户提交过来的内容。

> PHP 中有三个超全局变量专门用来获取表单提交内容：
>
> - `$_GET`：用于获取以 GET 方式提交的内容，更标准的说法：接收 URL 地址问号参数中的数据
> - `$_POST`：用于获取以 POST 方式提交的内容，更标准的说法：接收 请求体 中的数据
> - `$_REQUEST`：用于获取 GET 或 POST 方式提交的内容

借助 `$_POST` 或者 `$_REQUEST` 就可以获取到表单提交的内容：

```php
<?php
// 获取表单提交的用户名和密码
echo '用户名：' . $_REQUEST['username'];
echo '密码：' . $_REQUEST['password'];
```

1.1. 提交地址

`action` 提交地址指的是这个表单填写完成过后点击提交，发送请求的请求地址是什么。

从便于维护的角度考虑，一般我们最常见的都是提交给当前文件，然后在当前文件中判断是否是表单提交请求：

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  // 表单提交请求
}
```

另外，建议使用 `$_SERVER['PHP_SELF']` 动态获取当前页面访问路径，这样就不用因为文件重命名或者网站目录结构调整而修改代码了：

```php
<!-- 这样写死 action 地址，当文件重命名就需要修改代码 -->
<form action="/foo/login.php">
  <!-- ... -->
</form>
<!-- 通过 `$_SERVER['PHP_SELF']` 获取路径，可以轻松避免这个问题 -->
<form action="<?php echo $_SERVER['PHP_SELF']; ?>">
  <!-- ... -->
</form>
```

> 鲁棒性：指的是我们的程序应对变化的能力

1.2. 提交方式

`method` 可以用于设置表单提交的方式，目前我们所认识的就是最常见两种表单提交方式：`GET` 和 `POST`。

从效果上来看，两者都可以将数据提交到服务端，但是从实现提交的原理上两者有很大的不同：

- GET
  - **表单数据是通过 URL 中的？参数传递到服务端的**
  - 可以在地址栏中看到提交的内容
  - 数据长度有限制，因为 URL 地址长度有限（2000个字符）
- POST
  - **表单数据是通过请求体传递到服务端的**，我们在界面上看不到
  - 可以提交任何类型的数据，包括文件
  - 由于界面上看不见，浏览器也不储存，所以更安全

至于什么情况下应该选用哪种方式，这个需要结合业务场景和这两种方式各自的特点来决定，没有绝对的答案，只能给出一些原则：

- 绝不能使用 GET 来发送密码或其他敏感信息！！！
- **应该想清楚这次请求到底主要是去拿东西，还是去送东西**

2. 常见表单元素处理

至于表单元素中的文本框文本域一类的元素，都是直接将元素的 `name` 属性值作为键，用户填写的信息作为值，发送到服务端。

但是表单元素中还有一些比较特殊的表单元素需要单独考虑：

2.1. 单选按钮

```
<!-- 最终只会提交选中的那一项的 value -->
<input type="radio" name="gender" value="male">
<input type="radio" name="gender" value="female">
```

2.2. 复选按钮

```
<!-- 没有设置 value 的 checkbox 选中提交的 value 是 on -->
<input type="checkbox" name="agree">
<!-- 设置了 value 的 checkbox 选中提交的是 value 值 -->
<input type="checkbox" name="agree" value="true">
```

如果需要同时提交多个选中项，可以在 `name` 属性后面 跟上 `[]`：

> http://php.net/manual/zh/faq.html.php#faq.html.arrays

```
<input type="checkbox" name="funs[]" value="football">
<input type="checkbox" name="funs[]" value="basketball">
<input type="checkbox" name="funs[]" value="world peace">
```

最终提交到服务端，通过 `$_POST` 接收到的是一个索引数组。

2.3. 选择框

```
<select name="subject">
  <!-- 设置 value 提交 value -->
  <option value="1">语文</option>
  <!-- 没有设置 value 提交 innerText -->
  <option>数学</option>

</select>
```

3. 案例

3.1. 基于文件的注册和登录

1. 注册
    1. 用户请求一个注册页面
    2. 服务端返回一个注册表单
    3. 用户填写表单并提交
    4. 服务端接收用户提交的信息
    5. 判断是否正确填写内容以及是否勾选同意
    6. 如果出现异常界面给出提示，并返回表单
    7. 如果都正常，则保存到文件中（每个用户一行）
2. 登录
    1. 自己分析

4. 文件上传

> http://php.net/manual/zh/features.file-upload.php
>
> **注意**：
>
> - 修改 `php.ini` 中的 `post_max_size` 配置，让服务端可以接受更大的请求体体积
> - 修改 `php.ini` 中的 `upload_max_filesize` 配置，让服务端支持更大的单个上传文件
>
> *暂时作为了解

`type` 属性为 `file` 的 `input` 元素可以通过表单提交文件（上传文件），服务端 PHP 可以通过 `$_FILES` 获取上传的文件信息。

```php
<?php
// 如果选择了文件 $_FILES['file']['error'] => 0
// 详细的错误码说明：http://php.net/manual/zh/features.file-upload.errors.php
if ($_FILES['file']['error'] === 0) {
  // PHP 在会自动接收客户端上传的文件到一个临时的目录
  $temp_file = $_FILES['file']['tmp_name'];
  // 我们只需要把文件保存到我们指定上传目录
  $target_file = '../static/uploads/' . $_FILES['file']['name'];
  if (move_uploaded_file($temp_file, $target_file)) {
    $image_file = '/static/uploads/' . $_FILES['file']['name'];
  }
}
```

`$_FILES` 同样也是一个关联数组，键为表单的 `name` ，内容如下：

```
array(1) {
  ["avatar"]=>
  array(5) {
    ["name"]=>
    string(17) "demo.jpg"
    ["type"]=>
    string(10) "image/jpeg"
    ["tmp_name"]=>
    string(27) "C:\Windows\Temp\php786C.tmp"
    ["error"]=>
    int(0)
    ["size"]=>
    int(29501)
  }
}
```

5. 音乐列表案例

> 基于文件存储的音乐数据**增删改查**

5.1. 思路分析

> 绝大多数情况下我们编写的应用功能都是在围绕着某种类型的数据做增删改查（Create / Read / Update / Delete）。

对于被增删改查的数据有几点是可以明确的：

- 结构相同的多条数据（可以认为是：一个数组，数组中的元素都具有相同的属性结构）
- 可以持久化（长久保存）

5.1.1. 数据放在哪？

我们第一件事就是考虑数据放在哪（怎么存怎么取）？

目前我们接触到的技术方案中，只有文件可以持久化的保存内容（数据），所以一定是用文件存我们要操作的数据。

但是由于我们要存放的是一个有着复杂结构的数据，并不是简简单单的值，所以我们必须设计一种**能表示复杂结构数据的方式**。

例如，一行为一条数据，不同信息之间用 `|` 分割，同时约定好每个位置的数据含义：

```
59d632855434e | 错过 | 梁咏琪 | /uploads/img/1.jpg | /uploads/mp3/1.mp3
59d632855434f | 开始懂了 | 孙燕姿 | /uploads/img/2.jpg | /uploads/mp3/2.mp3
59d6328554350 | 一生中最爱 | 谭咏麟 | /uploads/img/3.jpg | /uploads/mp3/3.mp3
59d6328554351 | 爱在深秋 | 谭咏麟 | /uploads/img/4.jpg | /uploads/mp3/4.mp3
```

这种方式很简单，但是缺点也非常明显，所以我们迫切需要一个更好更方便的表示有结构数据的方式。

5.2. JSON

JSON（JavaScript Object Notation）是一种通过普通字符串描述数据的手段，用于表示有结构的数据。类似于编程语言中字面量的概念，语法上跟 JavaScript 的字面量非常类似。

5.2.1. 数据类型

- null

```
null
```

- string

```
"hello json"
```

- number

```
2048
```

- boolean

```
true
```

- object

```
{
  "name": "zce",
  "age": 18,
  "gender": true,
  "girl_friend": null
}
```

- array

```
["zhangsan", "lisi", "wangwu"]
```

5.2.2. 注意

1. JSON 中属性名称必须用双引号包裹
2. JSON 中表述字符串必须使用双引号
3. JSON 中不能有单行或多行注释
4. JSON 没有 `undefined` 这个值

5.2.3. JSON 表述

有了 JSON 这种格式，我们就可以更加容易的表示拥有复杂结构的数据了。

```json
[
  {
    "id": "59d632855434e",
    "title": "错过",
    "artist": "梁咏琪",
    "images": ["/uploads/img/1.jpg"],
    "source": "/uploads/mp3/1.mp3"
  },
  {
    "id": "59d632855434f",
    "title": "开始懂了",
    "artist": "孙燕姿",
    "images": ["/uploads/img/2.jpg"],
    "source": "/uploads/mp3/2.mp3"
  },
  {
    "id": "59d6328554350",
    "title": "一生中最爱",
    "artist": "谭咏麟",
    "images": ["/uploads/img/3.jpg"],
    "source": "/uploads/mp3/3.mp3"
  },
  {
    "id": "59d6328554351",
    "title": "爱在深秋",
    "artist": "谭咏麟",
    "images": ["/uploads/img/4.jpg"],
    "source": "/uploads/mp3/4.mp3"
  }
]
```

5.3. 功能实现

> 在服务端开发领域中所谓的**渲染**指的是经过程序执行得到最终的 HTML 字符串这个过程。

5.3.1. 列表数据展示（展示类）

- 文件读取
- JSON 反序列化
  - json_decode 需要注意第二个参数
  - 如果希望以关联数组的方式而非对象的方式操作数据，可以将 json_decode 的第二个参数设置为 true
- 数组遍历 foreach

- PHP 与 HTML 混编

5.3.2. 新增数据（表单类）

- 表单使用（form action method enctype，input name label for id）
- 服务端表单校验并提示错误消息
  - empty 判断一个成员是否没定义或者值为 false（可以隐式转换为 false）
- 上传文件
  - 文件数量
  - 文件种类
  - 如果需要考虑文件重名的情况，可以给上传的文件重新命名（唯一名称）
- 单文件域多文件上传
  - name 一定 以 [] 结尾，服务端会接收到一个数组
- JSON 序列化
- 文件写入

5.3.3. 删除数据

- 问号传参
  - 一般情况下，如果需要超链接点击发起的请求可以传递参数，我们可以采用 `?` 的方式

```
<a href="/delete.php?id=123">删除</a>
```

- 数组移除元素
  - array_splice

6. 参考链接

- HTML 中的 form 标签：http://www.w3school.com.cn/html/html_forms.asp
- PHP 中处理表单：http://www.w3school.com.cn/php/php_forms.asp
- Emmet 手册：https://docs.emmet.io/cheat-sheet/