“表单生成器”项目开发文档

准备表单数组

使用多维数组保存数据

根据案例的需求分析可知，表单项的相关数据将统一保存到一个多维数组中。利用数字键名区分不同的表单项，每个表单项又是一个二维的关联数组，具体保存形式如下所示。

```
// 利用多维数组保存表单元素
[
    0 => [],      // 表单项
    1 => [],      // 表单项
    2 => [],      // 表单项
    3 => [],      // 表单项
    ……
];
```

```
// 每个表单项的数组结构
0 => [
    'tag' => '',          // 标签
    'text' => '',         // 提示文本
    'attr' => [],         // 属性数组
    'option' => [],       // 选项数组
    'default' => ''       // 默认值
],
```

在上述的数据保存形式中，每个表单项的 tag 元素用于保存标签，如 input、textarea、select；text 元素保存提示文本，如“姓名”和“性别”等；attr 元素保存表单元素的属性，如 type 属性、name 属性；option 元素保存单选按钮或复选框中的每个选项；default 保存默认值。

值得一提的是，表单项在数组中存储的顺序决定着 Web 表单定义后显示的顺序。因此，在保存表单项时要考虑好其所在具体位置。

准备表单生成数据

在了解表单数据如何保存后，接下来通过数组保存图 6-1 所示的表单生成数据，具体步骤如下。

（1）准备表单数组

定义一个变量$elements 用于保存需要生成的表单项，具体如下。

```
// $elements 数组保存整个表单
$elements = [
    0 => [],      // 第 1 个表单项数组
    1 => [],      // 第 2 个表单项数组
];
```

在定义$elements 数组后，接下来开始填充文本框、单选按钮等表单项，每个表单项是一个数组。

（2）文本框

下面先定义一个文本框，具体如下。

```
0 => [
    'tag' => 'input',
    'text' => '姓    名：',
    'attr' => ['type' => 'text', 'name' => 'user']
],
```

在上述代码中，数组元素 tag 表示要生成的表单项为 input 标签；text 用于提示该表单项的功能；attr 利用一个关联数组保存 input 元素的相关属性设置。其中，type 值为 text 表示当前的表单项是一个文本框。按照这样的结构，继续设置“邮箱”和“手机号码”两个文本框即可，这里就不再展示代码。

（3）单选按钮

接着为$elements 数组变量添加一个元素，用于生成一组单选按钮，具体如下。

```
3 => [
    'tag' => 'input',
    'text' => '性    别：',
    'attr' => ['type' => 'radio', 'name' => 'gender'],
    'option' => ['m' => '男', 'w' => '女'],
    'default' => 'm'
 ],
```

在上述代码中，数组元素 attr 利用关联数组保存了 input 标签的 type 属性，值为 radio，表示当前的表单项是单选按钮；option 利用关联数组保存具体的单选项，键名 m、w 为单选按钮的 value 属性值，对应的值“男”“女”为该单选项的提示信息。default 的值为 option 关联数组中的一个键名，表示默认选中哪一项。

（4）复选框

继续为$elements 数组添加一个元素，用于生成一组复选框，具体如下。

```
4 => [
    'tag' => 'input',
    'text' => '爱    好：',
    'attr' => ['type' => 'checkbox', 'name' => 'hobby[]'],
    'option' => ['swimming' => '游泳', 'reading' => '读书', 'running' => '跑步'],
    'default' => ['swimming', 'reading']
],
```

在上述代码中，复选框的 name 属性值为 hobby[]，表示该表单字段以数组形式提交；default 元素通过数组保存 option 中默认选中项。

（5）下拉列表

继续为$elements 数组添加一个元素，用于生成一个下拉列表，具体如下。

```
5 => [
    'tag' => 'select',
    'text' => '住    址：',
    'attr' => ['name' => 'area'],
    'option' => ['' => '--请选择--', 'BJ'=>'北京', 'SH'=>'上海', 'SZ'=>'深圳']
],
```

（6）文本域

继续为$elements 数组添加一个元素，用于生成一个文本域，具体如下。

```
6 => [
    'tag' => 'textarea',
    'text' => '自我介绍：',
    'attr' => ['name' => 'introduce', 'cols' => 50, 'rows' => 5]
],
```

（7）提交按钮

最后为$elements 数组添加一个元素，用于生成一个提交按钮，具体如下。

```
    7 => [
        'tag' => 'input',
        'attr' => ['type' => 'submit', 'value' => '提交']
    ]
```

按照上面的步骤完成数据的保存后，就可以编写函数来读取这个数组，按照数组中保存的表单项自动生成表单。

表单的自动生成

定义表单生成函数

编写一个 generate.php 文件，专门用于保存表单生成函数，具体代码如下。

```
1  <?php
2  function generate($elements)
3  {
4      return '生成结果';
5  }
```

接下来编写 form.php，用于定义$elements 数组，调用 generate()函数展示表单，具体代码如下。

```
1  <?php
2  require 'generate.php';
3  $elements = [
4      // 定义表单元素……
5  ];
6  ?>
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <meta charset="UTF-8">
11     <title>Web 表单生成器</title>
12   </head>
13   <body>
14       <div>个人信息</div>
15       <form method="post">
16         <?=generate($elements)?>
17       </form>
18   </body>
19 </html>
```

在上述代码中，第 2 行用于引入表单生成函数；第 3~5 行通过$elements 数组保存表单中的元素；第 16 行调用了 generate()函数，将$elements 数组作为参数传入，函数执行后将返回表单生成结果，然后通过<?=?>标签将返回结果输出到 HTML 中。

读取$elements 数组

若要实现表单的自动生成，就需要读取$elements 数组，按照数组来生成表单。接下来在 generate()函数中编写代码，具体如下。

```
1  function generate($elements)
2  {
3      $items = '';
4      $default = ['tag' => '', 'text'=>'', 'attr' => [], 'option' => [],
5                  'default' => ''];
6      foreach ($elements as $v) {
7          $v = array_merge($default, $v);
8          $generate = 'generate_' . array_shift($v);
9          $items .= '<tr>' . call_user_func_array($generate, $v) . '</tr>';
10     }
11     return "<table>$items</table>";
12 }
```

在上述代码中，变量$items 用于保存拼接结果；$default 保存了表单项的默认元素，用于在第 7 行与$elements 中的表单项$v 进行数组合并，从而确保待处理的数组结构符合要求。第 8 行利用 array_shift()函数将数组$v 中的第 1 个元素（即 tag）移出并返回，拼接到字符串“generate_”中，然后在第 9 行将这个字符串作为函数名调用，$v 数组作为参数传入。在拼接表单 HTML 时，generate()函数通过<table>标签进行布局，将每个表单项作为表格的一行，第 11 行代码返回了所有表单项的拼接结果。

以上编写的 generate()函数，实现了根据 tag 键值的不同，将生成表单项的工作分派给了其他函数来完成。以 tag 值为 input 为例，为了完成该类表单项的生成，需要按照如下格式定义函数。

```
1  function generate_input($text, $attr, $option, $default)
2  {
3      return '生成结果';
4  }
```

在上述代码中，函数参数的顺序取决于 generate()函数中的$default 数组中指定的顺序，当 generate()函数的第 7 行代码使用 array_merge()函数合并数组时，会按照$default 数组中每个元素的顺序进行合并，从而实现了无论给定数组采用什么样的顺序，都不会影响此处函数参数的顺序。

拼接表单元素的属性

在表单自动生成时，针对表单元素的属性拼接是每个表单项都需要编写的代码，因此可以将这些代码统一封装到函数中，以提高代码的复用性。

下面编写函数 generate_attr()，实现表单元素属性的拼接。具体代码如下所示。

```
1  function generate_attr($attr, $items = '')
2  {
3      foreach ($attr as $k => $v) {
4          $items .= " $k=\"$v\" ";
5      }
6      return $items;
7  }
```

上述代码中，参数$attr 是一个一维关联数组保存的表单元素属性，参数$items 用于保存属性拼接的字符串。第 3~5 行代码用于遍历属性数组，并按照 HTML 标签中属性的编写格式进行拼接。其中，$k 表示属性名称，$v 表示属性的值，并且在属性拼接的前后要留出空格，用以区分多个属性。第 6 行代码用于返回拼接的结果。

接下来，为了让大家更加清晰地理解 generate_attr()的使用，通过如下代码进行测试。

```
$attr = ['type' => 'radio', 'name' => 'gender'];
echo generate_attr($attr);   // 输出结果：type="radio" name="gender"
```

完成 generate_attr()函数后，接下来在 generate()函数中的第 9 行代码的上面，添加如下一行代码，实现在调用生成表单项的函数前，将属性数组转换为 HTML 格式。

```
$v['attr'] = generate_attr($v['attr']);
```

拼接 input 元素

继续编写函数 generate_input()完成指定 input 控件的生成，具体代码如下。

```
1  function generate_input($text, $attr, $option, $default)
2  {
3      if(empty($option)){
4          $items = "<input $attr value=\"$default\">";
5      } else {
6          $items = '';
7          foreach ($option as $k => $v) {
8              $checked = in_array($k, (array)$default, true) ? 'checked' : '';
9              $items .= "<label><input $checked $attr value=\"$k\">$v</label>";
10         }
11     }
12     return "<th>$text</th><td>$items</td>";
13 }
```

在上述代码中，第 3 行用于判断当前需要生成的是单个 input 元素（如文本框），还是组合元素（如单选按钮）。当$option 为空时，执行第 4 行代码拼接单个 input 元素，如 type 属性值为 text、password、hidden、reset、submit 的元素属于此类型。当$option 不为空时，执行第 6~10 行代码，拼接 type 属性值为 radio、checkbox 的元素。

其中，第 8 行代码通过调用 in_array()函数判断当前 input 元素的 value 属性值$k 是否在默认值数组$default 中，如果存在则拼接“checked”表示该项默认处于选中状态；第 12 行返回完整的拼接结果。

拼接 select 元素

继续编写函数 generate_select()完成下拉列表的拼接。具体代码如下所示。

```
1  function generate_select($text, $attr, $option, $default)
2  {
3      $items = '';
4      foreach ($option as $k => $v) {
5          $selected = ($default === $k) ? 'selected' : '';
6          $items .= "<option $selected value=\"$k\">$v</option>";
```

```
7       }
8       $select = "<select $attr>$items</select>";
9       return "<th>$text</th><td>$select</td>";
10  }
```

上述第 4~7 行代码用于拼接下拉列表的选项 option；第 8 行代码用于完成<select>标签的完整拼接；第 9 行代码用于返回含有描述信息的拼接结果。

拼接 textarea 元素

最后编写函数 generate_textarea()完成文本域的拼接，具体代码如下所示。

```
1   function generate_textarea($text, $attr, $option, $default)
1   {
2       $textarea = "<textarea $attr>$default</textarea>";
3       return "<th>$text</th><td>$textarea</td>";
4   }
```

上述第 2 行代码用于实现 textarea 元素的完整拼接，第 3 行代码返回拼接结果。

至此，一个简易的 Web 表单生成器已经完成了。该表单的 CSS 样式可以参考配套源代码。