03

项目 3 PHP 语言基础

学思课堂

PHP 有一套独立的语法规则，若没有语法规则的约束，PHP 就无法正常运行。在当前全面推进依法治国、加快建设社会主义法治国家的时代背景下，“讲规矩、守规矩”必然成为我们的共同追求和自觉行动。大学生作为社会主体的一部分、未来社会的重要力量，其规矩意识的养成，是时代发展的现实要求，也是大学生受社会发展影响的必然结果。

学习目标

知识目标

1.了解 PHP 语法规则、编码规范，并能熟练使用。

2.掌握 PHP 中变量与常量、数据类型的概念，以及数据类型转换的方法。

3.熟悉运算符与表达式的用法和运算符优先级。

4.掌握数组的基本内容，以及数组常用的处理函数。

技能目标

1.能够熟练运用 PHP 语法规范。

2.能够对数据进行数据类型转换。

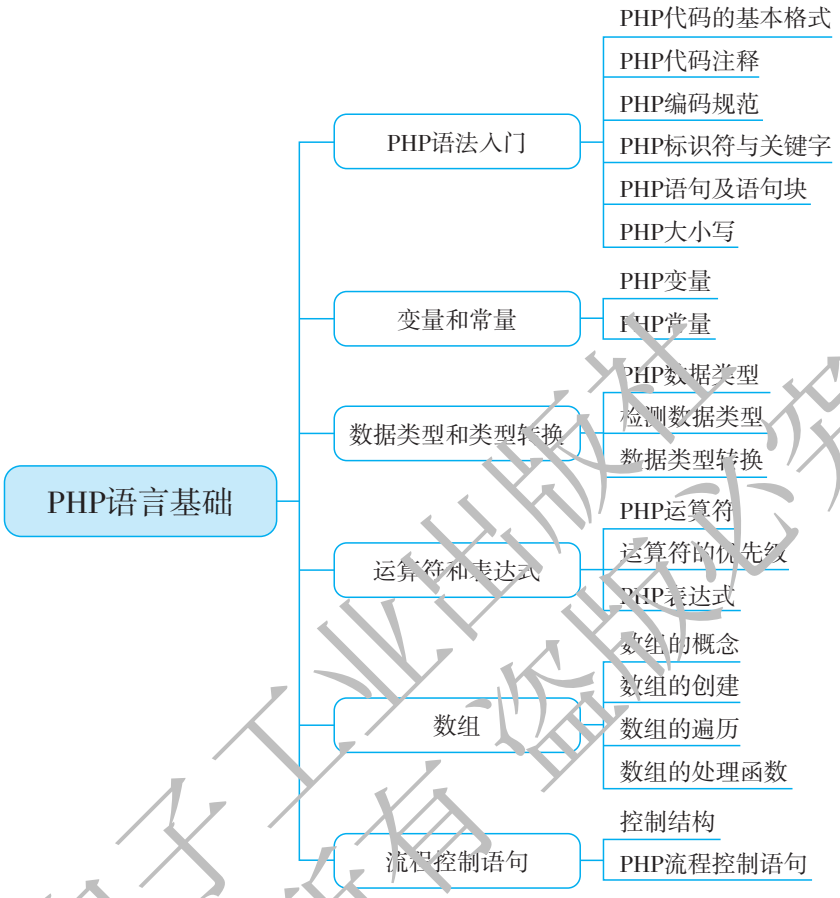
3.能够使用 PHP 代码编写程序。

素养目标

1.培养综合运用所学知识的能力。

2.培养学习毅力，提高学习专注力。

思维导图



任务3.1 PHP语法入门

PHP语法入门

3.1.1 PHP代码的基本格式

PHP是一种可嵌入HTML中的脚本语言。PHP的代码一般由两部分组成:一是HTML代码,还包括嵌入其中的CSS代码和JavaScript代码;二是服务器端脚本位于PHP定界符“<?”与“?>”之间的代码。其中,HTML代码是静态网页也具备的,它通过浏览器解释执行,又称为客户端代码。因此,可以通俗地认为PHP把服务器端脚本放在“<?”和“?>”之间,再嵌入静态网页中。

根据定界符的不同,PHP代码有4种风格,即XML标签风格、脚本风格、简短风格和ASP风格。使用任意一种风格都可将PHP代码嵌入HTML中。

1.XML 标签风格

XML 标签风格示例代码如下。

```
<?php
    echo "这是 XML 标签风格";
?>
```

从上面的代码中可以看到,XML 标签风格是以“<?php”开始(<?与 php 之间不能有空格),以“?>”结尾的,中间包含的就是 PHP 代码。这是 PHP 最常用的标签风格,推荐使用这种标签风格,因为它不会被服务器禁用,在 XML、XHTML 中都可以使用。

2.脚本风格

脚本风格示例代码如下。

```
<script language="php">
    echo "这是脚本风格";
</script>
```

脚本风格以“<script…>”开头,以“</script>”结尾。

3.简短风格

简短风格示例代码如下。

```
<?
    echo "这是简短风格";
?>
```

将定界符“<?php”中的 php 省略,就成了简短风格,它的定界符是“<?”和“?>”。如果要使用这种风格开发 PHP 程序,则必须确保 PHP 配置文件 php.ini 中的“short_open_tag”选项值设置为“on”(默认值即为“on”)。

4.ASP 风格

ASP 风格示例代码如下。

```
<%
    echo "这是 ASP 风格";
%>
```

如果要使用这种风格开发 PHP 程序,则必须确保 PHP 配置文件 php.ini 中的“asp_tags”选项值设置为“on”。不推荐使用这种风格。

3.1.2 PHP 代码注释

注释是指在程序编写过程中,对程序文件或者代码片段的解释说明,是程序中不可或缺的一个重要元素。使用注释不仅能够提高程序的可读性,还有利于程序的后期维护工作。程序执行时,注释会被 PHP 解释器忽略,因此浏览器端看不到 PHP 代码的注释。

PHP 的注释有 3 种风格,下面分别进行介绍。

1.Shell 脚本风格的单行注释(#)

```
<?php
    echo "使用 Shell 脚本风格的注释"; #这里的内容是看不到的
?>
```

运行结果为:

```
使用 Shell 脚本风格的注释
```

因为使用了注释符号“#”,所以在该注释符号后面的内容是不会被程序执行的。

2.C++风格的单行注释(//)

```
<?php
    echo "使用 C++风格的注释";
    //echo "这就是 C++风格的注释";
?>
```

运行结果为:

```
使用 C++风格的注释
```

上面代码虽然有两条 echo 输出语句,但因为使用了注释符号对第 2 条输出语句进行了注释,所以这条输出语句不会被程序执行。

注意:在使用单行注释时,注释内容中不要出现“?>”标志,因为解释器会认为这是 PHP 脚本,从而去执行“?>”后面的代码。

示例代码如下。

```
<?php
    echo "这样会出错的!";                                    //不会看到?> 会看到
?>
```

运行结果为：

```
这样会出错的！会看到?>
```

3.C 风格的多行注释(/ *…*/)

```
<?php
    /*
        echo "这是第 1 行注释信息";
        echo "这是第 2 行注释信息";
    */
    echo "使用 C 风格的注释";
?>
```

运行结果为：

```
使用 C 风格的注释
```

上面代码虽然有 3 条 echo 输出语句，但因为使用注释符号对前面两条输出语句进行了注释，所以前面两条输出语句不会被程序执行。如果要添加大段注释，则使用多行注释更方便。

程序注释是书写规范程序时的一个重要环节。注释主要是对代码的解释和说明，用来解释脚本的用途、版权说明、版本号、生成日期、作者、内容等，有助于对程序进行阅读与理解。使用注释应注意以下 9 项。

(1)注释可以书写在代码中的任意位置，但一般写在代码的开头或结束位置。

(2)注释在编译代码时会被忽略，不会被编译到最后的可执行文件中，因此，注释不会增加可执行文件的大小。

(3)注释语言必须准确、易懂、简洁。

(4)修改程序代码时，一定要同时修改相关的注释，保持代码和注释的同步。

(5)在程序块的结束行右边添加注释，以表明某程序块的结束。

(6)在实际的代码规范中，要求注释占程序代码的比例为 20% 左右，即 100 行程序中包含约 20 行注释。

(7)避免在一行代码或表达式的中间插入注释，否则容易使代码的可理解性变差。

(8)避免在注释中使用缩写，特别是不常用的缩写。

(9)注释与所描述内容进行同样的缩排，可使程序排版整齐，并方便阅读与理解注释。

3.1.3 PHP 编码规范

编码规范对于编程人员来说非常重要。在如今的 Web 项目开发中，一个人不可能完成

所有的工作,尤其是一些大型的项目,需要很多人共同完成。在项目开发中,难免会有新的开发人员参与进来,如果前面编写的代码没有按编码规范编写,新的开发人员在阅读代码时就会有许多问题。遵循编码规范能提高代码的可读性,有利于相关开发人员的交流;有利于开发人员了解任意代码,厘清程序的状况;有利于团队管理,重用部分资源;有利于程序的维护,降低软件成本。

以 PHP 开发为例,编码规范就是融合了开发人员长时间积累下来的经验,形成了一种良好统一的编程风格,这种良好统一的编程风格会在团队开发或二次开发时,起到事半功倍的效果。编码规范是一种总结性的说明和介绍,并不是强制性的规则。从项目长远的发展,以及团队效率来考虑,遵守编码规范是十分必要的。常见的编码规范如下。

1.书写规范

(1)缩进。使用制表符(Tab 键)缩进,缩进单位为 4 个空格。开发工具的种类多样,需要在开发工具中统一设置。

(2)大括号。有两种大括号“{}”放置规则可以使用。

一种是将大括号放到关键字的下方且同列。

```
if ($isTrue)
{
    …
}
```

另一种是首括号与关键词同行,而尾括号与关键字同列。

```
if ($isTrue){
    …
}
```

(3)关键字与小括号。小括号和关键字不能紧贴在一起,要用空格隔开它们。如:

```
if ($isTrue){                                    //if 和小括号之间有一个空格
    …
}
```

(4)函数与小括号。小括号和函数要紧贴在一起,以便区分关键字和函数。如:

```
round($n);                                       //round 和小括号之间没有空格
```

尽量不要在 return 返回语句中使用小括号。

(5)运算符。运算符与两边的变量或表达式之间要有一个空格(字符连接运算符“.”除

外)。示例代码如下。

```
while(a == b){                                    //“==”与“a”“b”之间都有一个空格
    …
}
```

当代码块较大时,上、下应当添加空白行,两个代码块之间可添加一个空行,但不建议添加多行。

2.命名规范

使用良好的命名规范也是重要的编程习惯,描述性强的名称让代码更容易阅读、理解和维护。命名遵循的基本原则是:以标准计算机英文为蓝本,杜绝一切拼音或拼音与英文混杂的命名方式,建议使用语义化的方式命名。

(1)变量命名。变量所有字母均使用小写,并使用下画线“_”来分隔每个词。

例如,$first_name、$last_name 等为较好的命名。

(2)引用变量。引用变量应该使用前缀“r”。

示例代码如下。

```
class Student{
    public $mSex="";
    public $mAge="";
    function SetAge(&$rAge){
        …
    }
    function &GetAge(){
        …
    }
}
```

(3)全局变量。全局变量应该使用前缀“g”,如 global $gTest、global $g。

(4)常量/全局常量。常量/全局常量应该全部使用大写字母,单词之间用下画线“_”来分隔。

示例代码如下。

```
define('DEFAULT_SCORE_AVE',90);
define('DEFAULT_SCORE_SUM',500);
```

(5)静态变量。静态变量应该使用前缀“s”。

示例代码如下。

```
static $sStatus=1;
```

(6)函数命名。函数名称均使用小写字母,多个单词使用下画线“_”来分隔。

示例代码如下。

```
function format_date(){
    …
}
```

以上的各种命名规则,可以组合在一起使用。

示例代码如下。

```
class Example{
    $msValue="";                                    //该参数既是类属性,又是静态属性
}
```

(7)类命名。类名的首字母使用大写。不要使用数字或下画线“_”。以大写字母作为词的分隔,其他的字母均使用小写。

例如,Score、ProductInfo 等为较好的命名方式。

(8)类属性命名。类属性名应该使用前缀“m”。“m”总是在名字的开头,起修饰作用。前缀“m”后采用与类命名一致的规则。

例如,mPrice、mWeight 等为较好的命名方式。

(9)方法命名。方法命名应说明该方法的作用,方法名称的前缀和后缀一般都有一定的规律,如 Get(获取)、Set(设置)、Is(判断)。方法的命名规范与类命名是一致的。

示例代码如下。

```
class Student{
    $mSex="";
    $mAge="";
    function GetAge(){
        …
    }
}
```

(10)方法中的参数命名。方法参数的第一个字母使用小写。在首字母后的所有字符都按照类命名的规范,将首字母大写。

示例代码如下。