验证数据

不受信任的数据来自许多来源（用户、第三方网站，甚至您自己的数据库！），所有这些数据在使用之前都需要进行检查。

请记住：即使管理员也是用户，用户也会有意或无意地输入错误的数据。你的工作就是保护他们免受伤害。

验证输入是根据预定义模式（或多个模式）测试数据的过程，并得出明确的结果：有效或无效。与清理相比，验证是一种更具体的方法，但两者都有其作用。

简单的验证示例：

- 检查必填字段是否未留空
- 检查输入的电话号码是否仅包含数字和标点符号
- 检查请求的字符串是否是五个有效选项之一
- 检查数量字段是否大于 0

应尽早进行数据验证。这意味着在执行任何操作之前验证数据。

验证理念

关于如何进行验证有几种不同的理念。每个都适合不同的场景。

安全名单

仅接受来自已知和可信值的有限列表的数据。

将不受信任的数据与安全列表进行比较时，确保使用严格的类型检查非常重要。否则，攻击者可能会以某种方式制作输入，使其通过安全列表，但仍然具有恶意效果。

比较运算符

```
$untrusted_input = '1 malicious string';  // will evaluate to integer 1 during loose comparisons


if ( 1 === $untrusted_input ) {  // == would have evaluated to true, but === evaluates to false
    echo '<p>Valid data';
} else {
    wp_die( 'Invalid data' );
}
```

in_array()

```
$untrusted_input = '1 malicious string';  // will evaluate to integer 1 during loose comparisons
$safe_values     = array( 1, 5, 7 );

if ( in_array( $untrusted_input, $safe_values, true ) ) {  // `true` enables strict type checking
    echo '<p>Valid data';
} else {
    wp_die( 'Invalid data' );
}
```

switch()

```
$untrusted_input = '1 malicious string';  // will evaluate to integer 1 during loose comparisons

switch ( true ) {
    case 1 === $untrusted_input:  // do your own strict comparison instead of relying on switch()'s loose
comparison
        echo '<p>Valid data';
        break;

    default:
        wp_die( 'Invalid data' );
}
```

黑名单

拒绝来自已知不可信值的有限列表中的数据。这很少是一个好主意。

格式检测

测试数据的格式是否正确。如果是的话，只接受它。

```
if ( ! ctype_alnum( $data ) ) {
  wp_die( "Invalid format" );
}

if ( preg_match( "/[^0-9.-]/", $data ) ) {
  wp_die( "Invalid format" );
}
```

格式修正

接受大部分数据，但删除或更改危险部分。

```
$trusted_integer = (int) $untrusted_integer;
$trusted_alpha = preg_replace( '/[^a-z]/i', "", $untrusted_alpha );
$trusted_slug = sanitize_title( $untrusted_slug );
```

实例一

假设我们有一个输入字段旨在接受美国邮政编码：

```
<input type="text" id="wporg_zip_code" name="my-zipcode" maxlength="10" />
```

在这里`11221`[`eval()`入十个字符.....但是对于可以输入

这就是验证的用武之地。在处理表单时，我们编写代码来检查每个字段的数据类型是否正确，如果不正确则将其丢弃。

`my-zipcode`字段，我们可能会这样做：

```
/**
 * Validate a US zip code.
 *
 * @param string $zip_code    RAW zip code to check.
 *
 * @return bool               true if valid, false otherwise.
 */
function wporg_is_valid_us_zip_code( string $zip_code ):bool {
    // Scenario 1: empty.
    if ( empty( $zip_code ) ) {
        return false;
    }

    // Scenario 2: more than 10 characters.
    // The `maxlength` attribute is only enforced by
    // the browser, so we still need to validate the
    // length of the input on the server to protect
    // against a manual submission.
    if ( 10 < strlen( trim( $zip_code ) ) ) {
        return false;
    }

    // Scenario 3: incorrect format.
```

```
    if ( ! preg_match( '/^d{5}(-?d{4})?$/', $zip_code ) ) {
        return false;
    }

    // Passed successfully.
    return true;
}
```

`wporg_zip_code`

```
if ( isset( $_POST['wporg_zip_code'] ) && wporg_is_valid_us_zip_code( $_POST['wporg_zip_code'] ) ) {
    // $_POST['wporg_zip_code'] is valid; carry on
}
```



```
$allowed_keys = array( 'author', 'post_author', 'date', 'post_date' );
$orderby      = sanitize_key( $_POST['orderby'] );
if ( in_array( $orderby, $allowed_keys, true ) ) {
    // $orderby is valid; carry on
}
```

`orderby`

`sanitize_key()` `in_array()`

`true` `in_array()`

“”

- `balanceTags( $html )` `force_balance_tags( $html )` – HTML XML
- `count()`
- `in_array()`
- `is_email()`
- `in_array()`
- `mb_strlen()` `strlen()`
- `preg_match()` `strpos()`

- `sanitize_html_class( $class, $fallback )` – 清理 html 类名以确保它仅包含有效字符。将字符串剥离为 AZ,az,0-9,'-'，如果这导致空字符串，则它将返回提供的
- `tag_escape( $html_tag_name )` – 清理 HTML 标签名称（不转义任何内容，尽管函数名称如此）。
- term_exists()检查标签、类别或其他分类术语是否存在。
- username_exists()检查用户名是否存在。
- validate_file()将验证输入的文件路径是否为真实路径（但不验证文件是否存在）。

WordPress `*_exists()` `*_validate()` [`is_*()`。并非所有这些都是验证函数，但许多都是有帮助的。

版本号 #1
由 Vcanson 创建于 18 九月 2023 11:02:41
由 Vcanson 更新于 26 九月 2023 15:34:10