# Basic Techniques for Securing the Apps

1. Filter input
   1. [trim()](https://www.php.net/manual/en/function.trim.php)
   2. [Prepared statement](https://www.w3schools.com/php/php_mysql_prepared_statements.asp)s
   3. [filters](https://www.php.net/manual/en/filter.filters.php)
      1. [filter\_var()](https://www.php.net/manual/en/function.filter-var.php)
      2. [Validate filters](https://www.php.net/manual/en/filter.filters.validate.php)
      3. [Sanitize filters](https://www.php.net/manual/en/filter.filters.sanitize.php) + [strip\_tags()](https://www.php.net/manual/en/function.strip-tags.php)
2. Escape/Sanitize output – *Pay attention to not escape the data more than once, you must escape only when you received it or when you need to output it!*
   1. [htmlentities()](https://www.php.net/manual/en/function.htmlentities.php)
   2. [htmlspecialchars()](https://www.php.net/manual/en/function.htmlspecialchars.php)
   3. [addslashes()](https://www.php.net/manual/en/function.addslashes.php)

The problem with htmlentities() is that it is not very powerful, in fact, it does not escape **single quotes**, cannot detect the character set and does not validate HTML as well. **To solve this problem –**

* The first argument that this function accepts is the string we need to sanitize.
* The second one must include a flag, in our case we want to use the  **ENT\_QUOTES** constant - **It prompts the function to encode single quotes.**
* Eventually, the last argument allows you to specify **the character set** you are using in your application.

A basic example would look like this:

**echo htmlentities($string, ENT\_QUOTES, 'UTF-8');**

1. Encrypt sensitive data *– user passwords are never saved in the data base without encryption!*
   1. [*password\_hash()*](https://www.php.net/manual/en/function.password-hash.php)

[more](https://dev.to/anastasionico/good-practices-how-to-sanitize-validate-and-escape-in-php-3-methods-139b)