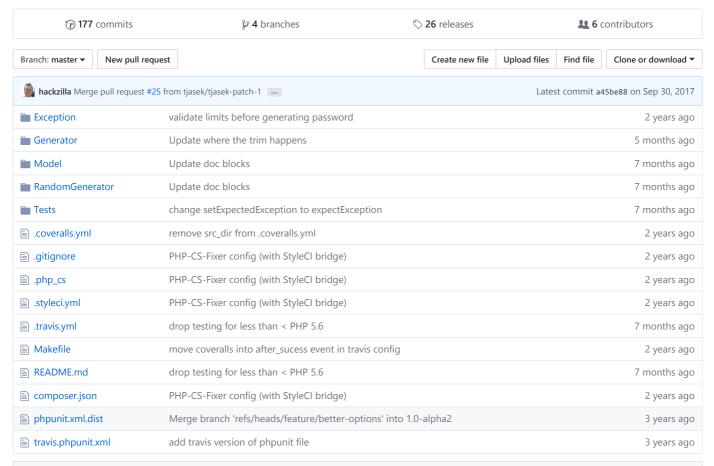
hackzilla / password-generator

PHP Library to generate random passwords http://hackzilla.org

#php #composer #password-generator #php7 #php-library



■ README.md

Password Generator Library

Simple library for generating random passwords.



Requirements

 \bullet PHP >= 5.3.2 (No longer testing <= PHP 5.6, and next version will drop support)

Installation

Install Composer

curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer

Now tell composer to download the library by running the command:

\$ composer require hackzilla/password-generator

Composer will add the library to your composer json file and install it into your project's vendor/hackzilla directory.

Simple Usage

```
use Hackzilla\PasswordGenerator\Generator\ComputerPasswordGenerator;

$generator = new ComputerPasswordGenerator();

$generator
    ->setOptionValue(ComputerPasswordGenerator::OPTION_UPPER_CASE, true)
    ->setOptionValue(ComputerPasswordGenerator::OPTION_LOWER_CASE, true)
    ->setOptionValue(ComputerPasswordGenerator::OPTION_NUMBERS, true)
    ->setOptionValue(ComputerPasswordGenerator::OPTION_SYMBOLS, false)

$password = $generator->generatePassword();
```

More Passwords Usage

If you want to generate 10 passwords that are 12 characters long.

```
use Hackzilla\PasswordGenerator\Generator\ComputerPasswordGenerator;

$generator = new ComputerPasswordGenerator();

$generator
   ->setUppercase()
   ->setLowercase()
   ->setNumbers()
   ->setSymbols(false)
   ->setLength(12);

$password = $generator->generatePasswords(10);
```

Hybrid Password Generator Usage

```
use Hackzilla\PasswordGenerator\Generator\HybridPasswordGenerator;

$generator = new HybridPasswordGenerator();

$generator
   ->setUppercase()
   ->setLowercase()
   ->setNumbers()
   ->setSymbols(false)
   ->setSegmentLength(3)
   ->setSegmentCount(4)
   ->setSegmentSeparator('-');

$password = $generator->generatePasswords(10);
```

If you can think of a better name for this password generator then let me know.

The segment separator will be remove from the possible characters.

Human Password Generator Usage

```
use Hackzilla\PasswordGenerator\Generator\HumanPasswordGenerator;

$generator = new HumanPasswordGenerator();

$generator
   ->setWordList('/usr/share/dict/words')
   ->setWordCount(3)
   ->setWordSeparator('-');

$password = $generator->generatePasswords(10);
```

Requirement Password Generator Usage

```
use Hackzilla\PasswordGenerator\Generator\RequirementPasswordGenerator;
$generator = new RequirementPasswordGenerator();
  ->setLength(16)
 ->setOptionValue(RequirementPasswordGenerator::OPTION_UPPER_CASE, true)
  ->setOptionValue(RequirementPasswordGenerator::OPTION_LOWER_CASE, true)
  ->setOptionValue(RequirementPasswordGenerator::OPTION_NUMBERS, true)
  ->setOptionValue(RequirementPasswordGenerator::OPTION_SYMBOLS, true)
  ->setMinimumCount(RequirementPasswordGenerator::OPTION_UPPER_CASE, 2)
  ->setMinimumCount(RequirementPasswordGenerator::OPTION_LOWER_CASE, 2)
  ->setMinimumCount(RequirementPasswordGenerator::OPTION_NUMBERS, 2)
  ->setMinimumCount(RequirementPasswordGenerator::OPTION_SYMBOLS, 2)
  ->setMaximumCount(RequirementPasswordGenerator::OPTION_UPPER_CASE, 8)
  ->setMaximumCount(RequirementPasswordGenerator::OPTION_LOWER_CASE, 8)
 ->setMaximumCount(RequirementPasswordGenerator::OPTION_NUMBERS, 8)
  ->setMaximumCount(RequirementPasswordGenerator::OPTION_SYMBOLS, 8)
$password = $generator->generatePassword();
```

A limit can be removed by passing null

```
$generator
->setMinimumCount(RequirementPasswordGenerator::OPTION_UPPER_CASE, null)
->setMaximumCount(RequirementPasswordGenerator::OPTION_UPPER_CASE, null)
;
```

When setting the minimum and maximum values, be careful of unachievable settings.

For example the following will end up in an infinite loop.

```
$generator -> setLength(4) -> setOptionValue(RequirementPasswordGenerator::OPTION_UPPER_CASE, true) -> setOptionValue(RequirementPasswordGenerator::OPTION_LOWER_CASE, false) -> setMinimumCount(RequirementPasswordGenerator::OPTION_UPPER_CASE, 5) -> setMaximumCount(RequirementPasswordGenerator::OPTION_LOWER_CASE, 1);
```

For the moment you can call \$generator->validLimits() to test whether the counts will cause problems. If the method returns true, then you can proceed. If false, then generatePassword() will likely cause an infinite loop.

Example Implementations

- Password Generator App [https://github.com/hackzilla/password-generator-app]
- Password Generator Bundle [https://github.com/hackzilla/password-generator-bundle]

Caution

This library uses mt_rand which is does not generate cryptographically secure values. Basically an attacker could predict the random passwords this library produces given the right conditions.

If you have a source of randomness you can inject it into the PasswordGenerator, using RandomGeneratorInterface.

PHP 7 has random_int function which they say is good to use for cryptographic random integers.

```
use Hackzilla\PasswordGenerator\Generator\HumanPasswordGenerator;
use Hackzilla\PasswordGenerator\RandomGenerator\Php7RandomGenerator;

$generator = new HumanPasswordGenerator();

$generator
   ->setRandomGenerator(new Php7RandomGenerator())
   ->setWordList('/usr/share/dict/words')
   ->setWordCount(3)
```

```
->setWordSeparator('-');

$password = $generator->generatePasswords(10);
```