# Запуск тестов из командной строки

Список параметров, которые можно указать в командной строке можно получить, указав параметр --help

phpunit --help

**phpunit UnitTest**

Выполняет тесты, представленные в классе **UnitTest**. Предполагается что этот класс объявлен в файле **UnitTest.php**.

Класс **UnitTest** должен наследовать **PHPUnit\_Framework\_TestCase** или предоставлять метод **public static suite()**, который возвращает объект типа **PHPUnit\_Framework\_Test**. Например, экземпляр класса **PHPUnit\_Framework\_TestSuite**.

**phpunit UnitTest UnitTest.php**

Выполняет тесты из класса **UnitTest**. Этот класс должен быть объявлен в указанном файле.

**--log-junit**

Генерирует файл логов в формате JUnit XML для всех пройденных тестов

**--log-tap**

Генерирует файл логов, используя формат [Test Anything Protocol (TAP)](http://yapro.ru/system/redirect?to=http://testanything.org/) для пройденных тестов.

**--log-json**

Генерирует файл логов, используя [JSON](http://yapro.ru/system/redirect?to=http://www.json.org/) формат

**--coverage-html**

Генерирует отчёт о покрытии кода в формате HTML. Более детальную информацию см. [*Code Coverage Analysis*](https://phpunit.de/manual/current/en/code-coverage-analysis.html).

Обратите внимание, что эта функциональность доступна, если установлены расширения tokenizer и Xdebug.

**--coverage-clover**

Генерирует файл логов в формате XML для пройденных тестов.

Обратите внимание, что эта функциональность доступна, если установлены расширения tokenizer и Xdebug.

**--coverage-php**

Генерирует сериализованый (serialized) объект класса PHP\_CodeCoverage с информацией о покрытии.

Обратите внимание, что эта функциональность доступна, если установлены расширения tokenizer и Xdebug.

**--coverage-text**

Генерирует файл логов или вывод в командной строке в человеко-читаемом формате.

Обратите внимание, что эта функциональность доступна, если установлены расширения tokenizer и Xdebug.

**--testdox-html** **и** **--testdox-text**

Генерирует agile документацию в текстовом или HTML формате для всех запущеных тестов. Более детальную информацию см. [Глава 15,](http://yapro.ru/web-master/php/other-uses-for-tests.html) [*Другие использования тестов*](http://yapro.ru/web-master/php/other-uses-for-tests.html).

**--filter**

Вызывает только те тесты, которые содержат в себе заданный фильтр. Фильтром может быть название (или подстрока) теста, или же [регулярное выражение](http://yapro.ru/system/redirect?to=http://ru.php.net/pcre), которое соответствует нескольким названиям тестов.

**--group**

Выполняет только тесты из указанных групп. Тест может принадлежать группе, если он помечен аннотацией **@group**.

Аннотация **@author** является синонимом аннотации **@group** и позволяет фильтровать тесты по их авторам.

**--exclude-group**

Исключает тесты указанных групп. Тест может принадлежать группе, если он помечен аннотацией **@group**.

**--list-groups**

Выводит список возможных групп тестов.

**--loader**

Указывает какую реализацию загрузчика **PHPUnit\_Runner\_TestSuiteLoader** использовать.

Стандартная реализация загрузчика будет искать исходные файлы тестов в текущей директории и всех директориях указанных в директиве **include\_path**. Названия классов должны следовать конвенции PEAR Naming Conventions. Например, класс **Project\_Package\_Class** должен быть реализован в файле **Project/Package/Class.php**.

**--printer**

Указывает какую реализацию форматирования вывода использовать. Этот класс должен наследовать **PHPUnit\_Util\_Printer** и реализовывать интерфейс **PHPUnit\_Framework\_TestListener**.

**--repeat**

Уазывает сколько раз прогнать тесты по кругу.

**--tap**

Выводит отчет в формате [Test Anything Protocol (TAP)](http://yapro.ru/system/redirect?to=http://testanything.org/).

**--testdox**

Выводит результат тестов в виде документации agile. Более детальную информацию см. [*Другие использования тестов*](http://yapro.ru/web-master/php/other-uses-for-tests.html).

**--colors**

Использовать цвета при выводе.

**--stderr**

Выводить в **STDERR** вместо **STDOUT**.

**--stop-on-error**

Остановить выполнение при возникновении первой ошибки.

**--stop-on-failure**

Остановить выполнение при возникновении первой ошибки или провала.

**--stop-on-skipped**

Остановить выполнение при первом пропущенном тесте.

**--stop-on-incomplete**

Остановить выполнение при первом незавершенном тесте.

**--strict**

Выполнять тесты в строгом (strict) режиме.

**--verbose**

Выводить дополнительную информацию. Например, названия тестов, которые были пропущены или не завершены.

**--process-isolation**

Выполнять каждый тест в отдельном потоке PHP.

**--no-globals-backup**

Не сохранять и восстанавливать $GLOBALS.

**--static-backup**

Сохранять и восстанавливать статические аттрибуты тестируемых классов.

**--bootstrap**

"Bootstrap" - PHP файл, который будет запущен перед запуском тестов.

**--configuration,** **-c**

Прочитать настройки из файла в формате XML.

Если файл **phpunit.xml** или **phpunit.xml.dist** существует (в этом порядке) в текущей директории и опция **--configuration** *не* использована - файл настроек автоматически будет прочитан.

**--no-configuration**

Игнорировать файл **phpunit.xml** и **phpunit.xml.dist**, находящиеся в текущей папке.

**--include-path**

Добавить указанный(е) пути в начало директивы PHP **include\_path**.

**-d**

Устанавливает заданный(е) параметр(ы) конфигурации PHP.

**--debug**

Выводить отладочную информацию, такую как название теста, перед тем как он будет запущен.

Пожалуй, единственным минусом будет то что, при таком подходе, нельзя запускать тесты в определённом порядке.

 Использование файла настроек XML для составления тест-комплектов

Также, для составления тест-комплектов, можно использовать файл настроек XML для PHPUnit показывает элементарный пример, который добавит все классы с маской **\*Test**, которые находятся в файлах **\*Test.php**. При поиске тест-кейсов директория **Tests** обходится рекурсивно.

**Пример Использование файла настроек XML для составления тест-комплектов**

<phpunit>

<testsuites>

<testsuite name="Object\_Freezer">

<directory>Tests</directory>

</testsuite>

</testsuites>

</phpunit>

Порядок, в котором будут выполнены тест-кейсы, можно указать следующим образом:

**Использование файла настроек XML для составления тест-комплектов с указанием точного порядка выполнения тест-кейсов**

<phpunit>

<testsuites>

<testsuite name="Object\_Freezer">

<file>Tests/Freezer/HashGenerator/NonRecursiveSHA1Test.php</file>

<file>Tests/Freezer/IdGenerator/UUIDTest.php</file>

<file>Tests/Freezer/UtilTest.php</file>

<file>Tests/FreezerTest.php</file>

<file>Tests/Freezer/StorageTest.php</file>

<file>Tests/Freezer/Storage/CouchDB/WithLazyLoadTest.php</file>

<file>Tests/Freezer/Storage/CouchDB/WithoutLazyLoadTest.php</file>

</testsuite>

</testsuites>

</phpunit>

Для каждого теста PHPUnit выводит на экран один символ:

**.** Выводится, если тест прошел.

**F** Выводится, если тест провалился.

**E** Выводится когда возникает ошибка.

**S** Выводится, если тест был пропущен

**I** Выводится, если тест помечен как "незавершён" или "ещё не реализован"

PHPUnit разделяет статусы *провал* и *ошибка*. Провал выводится в случае, если утверждение не выполняется **assertEquals()**. Ошибкой может являться непойманное исключение или ошибка PHP. Иногда это разграничение становится полезным, потому что, обычно, ошибки проще исправить чем провалы. Если в проекте выявлен большой список проблем - следует сперва обратить внимание на ошибки, и, затем, проверить оставшиеся провалы.