Содержание

- Новости
- Идея сайта
- Статьи
  - Операционные системы
    - Linux
    - QNX
  - Программирование
    - Java
      - Примеры
    - **PHP**
    - Web
    - Zend Framework
    - Тестирование
  - СУБД
    - MySQL
    - Oracle
- Наши работы
- Ссылки

Метки

.Net CSS Drupal Eclipse Fedora HTTP Java JavaDB Linux MySQL Oracle Oracle Linux PHP PHPUnit PL/SQL QNX Smarty SWT Twitter UAC UTF8 Web Windows7 XML xUnit Zend Framework Безопасность Книги ОСи разное Регулярные выражения СУБД Тестирование

PHPUnit. Часть 02 Цели PHPUnit

Статьи -> Программирование -> PHP

PHPUnit. Часть 02 Цели PHPUnit

v:1.0 21.03.2010

Перевод статьи Chapter 2. PHPUnit's Goals.
Автор: Sebastian Bergmann
Перевод: Петрелевич Сергей

Предисловие переводчика
Эта статья продолжает цикл про PHPUnit, первая часть находится здесь.

Итак, у нас есть только два теста для встроенного `массива` и функции `sizeof()`. Когда мы начнем тестировать все бесчисленные встроенные функции, которые предлагает PHP для работы с массивом, нам придется написать по одному тесту для каждой из них. Конечно, мы могли бы написать необходимую инфраструктуру с нуля. Однако намного лучше использовать ранее написанную инфраструктуру, и доделывать только отдельные специфические части. PHPUnit - это пример такой инфраструктуры.

Инфраструктура (framework) PHPUnit должна решить ряд ограничений, причем некоторые из этих ограничений пересекаются.
Тесты можно считать хорошими, если их:

Легко научиться писать.
: Если обучиться написанию тестов трудно, разработчики не будут это делать.

Легко написать.
: Если написать тесты трудно, разработчики не будут это делать.

Легко прочитать.
: Тест не должен содержать какие-либо внешние переопределения, чтобы тест не потерялся в шуме, который его окружает.

Легко выполнить.
: Тест должен выполняться легким нажатием кнопки и выводить результат в понятном и недвусмысленном формате.

Быстро выполнить.
: Тесты должны выполняться очень быстро, чтобы была возможность выполнять тестирование сотни тысяч раз в день.

Изолированность.
: Тесты не должны влиять друг на друга. Если порядок выполнения тестов изменяется, итоговый результат меняться не должен.

Комбинируемость.
: У нас должна быть возможность запускать любую комбинацию тестов. Это следствие изолированности.

У этих ограничений есть два противоречия:

Легкость обучения написанию тестов противоречит легкости написания.
: В большинстве своем, тесты не требуют всей гибкости языков программирования. Многие средства тестирования предоставляют свой скриптовый язык, который имеет только минимально необходимые функции для написания тестов. Используя такие скрипты, легко написать нужные тесты, которые

потом можно легко прочитать, такие тесты не содержат синтаксические конструкции, которые отвлекают от содержания.
Однако, изучение скриптового языка и набора программных средств может быть достаточно сложным делом.

Изолированность противоречит скорости выполнения.
Если Вы хотите, чтобы результаты одного теста не влияли на результаты других, то для выполнения каждого теста Вам надо полностью создать состояние среды тестирования и восстановить первоначальное состояние после того как тест будет завершен.
Однако, создание состояния среды тестирования может занять длительное время. Например, подключение к базе данных и инициализация достоверными данными.

PHPUnit старается разрешить эти противоречия, используя PHP как язык написания тестов. Иногда вся мощь PHP избыточна для написания простых тестов и заметно их усложняет, однако применяя PHP, мы используем ранее накопленный опыт и множество уже готовых и отлаженных инструментальных средств. Если мы пытаемся убедить ленивых тестировщиков использовать новое средство, то максимальная простота написания первых тестов становится очень важным условием.

PHPUnit больше склоняется в сторону изолированности в ущерб скорости выполнения. Изолированные тесты очень важны, поскольку они обеспечивают высококачественную обратную связь. Если первый тест серии тестов закончится неудачно, то такая неудача не повлияет на выполнение других тестов. В результате Вы получите отчет о выполнении отдельных независимых тестов, а не запутанный клубок непонятных результатов, из которого не ясно то ли тест не сработал сам, то ли другие тесты на него негативно повлияли.
Использование изолированных тестов поощряет разработчика использовать большое количество простых объектов. Изолированно можно быстро проверить все эти объекты. В результате архитектура становится лучше, а тесты быстрее.

PHPUnit предполагает, что большинство тестов завершается успешно, и нет необходимости выводить детальные сведенья о них. Если тест заканчивается неудачно, этот факт обязательно фиксируется в отчете. Большинство тестов должно завершиться успешно, успешные результаты никак не комментируются, отмечается только общее число выполненных тестов. Это предположение реализовано на уровне классов отчетов, а не ядра PHPUnit. В отчетах о результатах тестирования отмечается общее число выполненных тестов и детальная информация о тестах, которые закончились с ошибкой.

Тест должен быть минимально возможного размера, и должен проверять только одну характеристику объекта. Поэтому, если тест в первый раз заканчивается неудачей, то выполнение тестов прерывается, и PHPUnit сообщает об ошибке. Это своего рода искусство, выполнять тестирование при помощи множества маленьких тестов. Применение микротестов позволяет в целом улучшить архитектуру системы.

Тестировать объект при помощи PHPUnit возможно только через публичный интерфейс. Тестирование, основанное только на публично видимом поведении, позволяет выявить и устранить проблемы архитектуры до того, как эти проблемы распространятся на другие части системы.

Метки: PHP Web PHPUnit Тестирование

Комментарии.

Внимание.
Комментировать могут только зарегистрированные пользователи.
Возможно использование следующих HTML тегов: <a>, <b>, <i>,
.

Пожалуйста авторизуйтесь или зарегистрируйтесь.