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 08 Практика тестирования

Статьи -> Программирование -> PHP

PHPUnit. Часть 08 Практика тестирования

v:1.0 16.04.2010

Перевод статьи Chapter 12. Testing Practices.
Автор: Sebastian Bergmann
Перевод: Петрелевич Сергей

Предисловие переводчика
Эта статья продолжает серию переводов официальной документации по PHPUnit на русский язык.
Часть 1, Часть 2, Часть 3, Часть 4, Часть 5, Часть 6, Часть 7,

Можно написать много тестов. Однако, скоро Вы поймете, что только некоторые тесты действительно полезны. Вы поймете, что нужны тесты, которые провалятся, хотя Вы думаете, что они должны закончиться успешно или которые закончатся успешно, хотя Вы считаете, что они должны провалиться.
Полезно рассматривать тесты с точки зрения цена/выгода. Вполне понятно желание писать тесты, которые вернут полезную информацию.
-- *Erich Gamma*

Во время разработки

Обычно, изменяя внутреннюю структуру программного обеспечения, программисты стараются не изменять внешнее поведение системы. Для выполнения качественного рефакторинга наборы тестов становятся бесценным инструментом. Применение тестов позволяет своевременно обнаружить ошибки, добавленные во время рефакторинга.

Применение модульных тестов позволяет контролировать выполнение следующих условий, которые помогают улучшить код и архитектуру приложения:

1. Все модульные тесты завершились успешно.
2. Код документирует свое поведение.
3. Код не содержит избыточность.
4. Код содержит минимальное количество классов и методов.

Если Вам надо добавить новую функциональность в систему, напишите сначала тест. После завершения разработки выполните все тесты.
Эта практика будет более детально обсуждаться в следующей главе.

Во время отладки

Когда Вы находите ошибку в коде, первым импульсным желанием будет исправить ошибку как можно быстрее. Опыт показывает, что этот импульс не всегда служит хорошую службу; часто исправление ошибки является причиной другой ошибки.

Сдержите свой первый импульс и выполните следующие действия:

1. Убедитесь, что можете воспроизвести эту ошибку.
2. Выявите минимальный путь воспроизведения ошибки, т.е. локализуйте место ошибки.
   Например, если выводится некорректное число,

найдите объект, который рассчитывает это число.
3. Напишите автоматический тест, который провалится, но после исправления ошибки завершится успешно.
4. Исправьте ошибку.

Знание минимального пути воспроизведения ошибки даст Вам возможность проверить корректность исправления. Написанный тест позволяет убедиться в том, что Вы действительно исправили ошибку. Тест имеет значительно меньшую надежность, если написан уже после того как ошибка исправлена. Тесты, написанные до исправления проблемы, повышают вероятность найти новые проблемы небрежного исправления.

Модульное тестирование имеет много преимуществ:

- Тестирование дает уверенность в корректности вносимых в код исправлений.
- Практика написания тестов поощряет разработчика анализировать граничные возможные случаи.
- Тестирование - это хороший способ быстрого поиска ошибок и недопущения их повторного появления.
- Модульные тесты - это работающие примеры использования API. Эти примеры позволяют упростить процесс документирования.

Подводя итог, можно еще раз отметить, что интегрированные модульные тесты снижают цену и уменьшают риск дальнейших изменений, способствуют улучшению качества архитектуры.
-- *Benjamin Smedberg*

Метки: PHP Web PHPUnit Тестирование

Комментарии.

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

Аноним 28.05.2010 0:51:37

Спасибо! Было интересно и познавательно почитать.

Пожалуйста авторизуйтесь или зарегистрируйтесь.