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 07 Тестирование базы данных

Статьи -> Программирование -> PHP

PHPUnit. Часть 07 Тестирование базы данных

v:1.0 01.04.2010

Перевод статьи Chapter 9. Database Testing.
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

Во время разработки тестов программного обеспечения часто возникает необходимость написать модульные тесты для кода, работающего с базой данных. PHPUnit содержит расширение для тестирования базы данных. Расширение выполняет следующие функции: перевод базы данных в заранее известное состояние, выполнение необходимых модификаций данных, проверка, что в базе данных созданы ожидаемые записи.

Самый быстрый способ создания нового модульного теста базы данных заключается в наследовании класса `PHPUnit_Extensions_Database_TestCase`. Этот класс предоставляет функционал установки соединения с базой данных, заполнения базы данными и сравнения полученного результата с ожиданиями. В Примере 9.1 показано как можно использовать функции `getConnection()` и `getDataSet()`.

Пример 9.1: Модульный тест базы данных

```
<?php
require_once
'PHPUnit/Extensions/Database/TestCase.php'
;

class DatabaseTest extends
PHPUnit_Extensions_Database_TestCase
{
    protected function getConnection()
    {
        $pdo = new
PDO('mysql:host=localhost;dbname=testdb',
'root', '');
        return $this-
>createDefaultDBConnection($pdo,
'testdb');
    }

    protected function getDataSet()
    {
        return $this-
>createFlatXMLDataSet(dirname(__FILE__).'
/_files/bank-account-seed.xml');
    }
}
?>
```

* This source code was highlighted with Source Code Highlighter.

Метод `getConnection()` должен вернуть реализацию интерфейса

`PHPUnit_Extensions_Database_DB_IDatabaseConnection`. Метод `createDefaultDBConnection()` можно использовать для получения соединения с базой данных. Первый параметр метода `createDefaultDBConnection()` - объект `PDO`, второй параметр - имя схемы базы данных.

Метод `getDataSet()` должен возвращать реализацию интерфейса `PHPUnit_Extensions_Database_DataSet_IDataSet`. В PHPUnit доступны три вида набора данных (data sets), все они описаны в разделе "Data Sets"

Таблица 9.1. Методы тестирования базы данных

Метод	Назначение
`PHPUnit_Extensions_Database_DB_IDatabaseConnection getConnection()`	Возвращает объект соединения с базой данных.
`PHPUnit_Extensions_Database_DataSet_IDataSet getDataSet()`	Возвращает набор данных, который будет использоваться в операциях перевода базы в требуемое состояние и возврата и исходное.
`PHPUnit_Extensions_Database_Operation_DatabaseOperation getSetUpOperation()`	Метод перегружается для выполнения специфической операции перед началом каждого теста.
`PHPUnit_Extensions_Database_Operation_DatabaseOperation getTearDownOperation()`	Метод перегружается для выполнения специфической операции к конце каждого теста.
`PHPUnit_Extensions_Database_DB_DefaultDatabaseConnection createDefaultDBConnection(PDO $connection, string $schema)`	Возвращает обернутый объект соединения с базой данных, основой обертки служит объект `$connection PDO`. Схема базы данных, которую следует протестировать, указывается вторым параметром `$schema`. Результат работы этого метода можно получить функцией `getConnection()`.
	Возвращает набор данных в формате Flat XML. Набор данных получается в результате

Метод	Назначение
`PHPUnit_Extensions_Database_DataSet_FlatXmlDataSet createFlatXMLDataSet(string $xmlFile)`	загрузки файла, абсолютный путь к которому задается параметром `$xmlFile`. Детальную информацию о Flat XML файлах можно найти в разделе "Наборы данных в формате Flat XML". Результаты работы метода можно получить функцией `getDataSet()`.
`PHPUnit_Extensions_Database_DataSet_XmlDataSet createXMLDataSet(string $xmlFile)`	Возвращает набор данных в формате XML. Набор данных получается в результате загрузки файла, абсолютный путь к которому задается параметром `$xmlFile`. Детальную информацию о XML файлах можно найти в разделе "Наборы данных в формате XML". Результаты работы метода можно получить функцией `getDataSet()`.
`void assertTablesEqual(PHPUnit_Extensions_Database_DataSet_ITable $expected, PHPUnit_Extensions_Database_DataSet_ITable $actual)`	Возвращает ошибку, если содержимое таблицы `$expected` (ожидания) не совпадает с содержимым таблицы `$actual` (фактический результат).
`void assertDataSetsEqual(PHPUnit_Extensions_Database_DataSet_IDataSet $expected, PHPUnit_Extensions_Database_DataSet_IDataSet $actual)`	Возвращает ошибку, если содержимое набора данных `$expected` (ожидания) не совпадает с содержимым набора данных `$actual` (фактический результат).

Наборы данных (Data Sets)

Наборы данных - это простейшие строительные блоки для тестовых окружений (fixture) базы данных и утверждений,

которые проверяются после завершения теста (ожидаемый результат). Если метод `PHPUnit_Extensions_Database_TestCase::getDataSet()` возвращает набор данных в качестве тестового окружения, то по умолчанию PHPUnit автоматически очистит все указанные таблицы и вставит в них данные в порядке, указанном в наборе данных. PHPUnit может работать с несколькими типами наборов данных.

Наборы данных в формате Flat XML

Плоский XML - это очень простой формат XML, в котором каждой записи набора данных соответствует один XML элемент, см. Пример 9.2.

Пример 9.2: Набор данных в формате Flat XML

```
<?xml version="1.0" encoding="UTF-8" ?> <dataset>
<post post_id="1" title="My First Post"
date_created="2008-12-01 12:30:29" contents="This
is my first post" rating="5" /> <post post_id="2"
title="My Second Post" date_created="2008-12-04
15:35:25" contents="This is my second post" />
<post post_id="3" title="My Third Post"
date_created="2008-12-09 03:17:05" contents="This
is my third post" rating="3" /> <post_comment
post_comment_id="2" post_id="2" author="Frank"
content="That is because this is simply an
example." url="http://phpun.it/" /> <post_comment
post_comment_id="1" post_id="2" author="Tom"
content="While the topic seemed interesting the
content was lacking." /> <current_visitors />
</dataset>
```

Как Вы видите, форматирование документа очень простое. Каждый элемент `<dataset>` соответствует записи в тестовой базе данных, за исключением последнего элемента - `<current_visitors />` (будет кратко описан ниже).
Имя элемента соответствует имени таблицы в базе данных. Имя атрибута элемента соответствует имени колонки таблицы. Значение атрибута - это значение, которое будет вставлено в таблицу.

Хотя этот формат и очень прост, но тем не менее требует дополнительные пояснения.
Во-первых, что делать с пустыми таблицами?
С помощью операции `CLEAN_INSERT` можно указать, что надо только очистить таблицу и не вставлять в нее никакие данные. Чтобы достичь этот результат, просто вставьте в набор данных элемент без атрибутов, как это показано в Примере 9.2, элемент `<current_visitors />`.
Чаще всего очистить таблицу надо перед выполнением тестирования вставки записей. Чем меньше данных в таблице, тем быстрее будут выполнены тесты. Итак, если бы я тестировал программное обеспечение моего блога, точнее функционал добавления комментариев в блоге, я бы изменил Пример 9.2, описав `post_comment` как пустую таблицу. При работе с утверждениями часто бывает полезно убедиться, что в таблицу не добавлены ошибочные записи. Использование формата пустой таблицы Flat XML позволяет выполнить эту проверку.

Во-вторых, как описать значения `NULL`?
Природа формата Flat XML позволяет только явно задать значение колонки, причем задать в виде строки. Конечно, в этом случае база данных самостоятельно должна выполнить преобразование строки в число или дату. Однако, нет четкой аннотации для значения `NULL`. Вы можете имитировать значение `NULL`, пропустив колонку в списке атрибутов элемента. Это будет означать - вставить значение NULL в колонку таблицы.

В-третьих, как определить колонки таблицы?
В первом элементе определяются все колонки таблицы.

Список атрибутов элемента соответствует списку колонок таблицы. В Примере 9.2 предполагается, что в таблице `post` есть следующие колонки `post_id`, `title`, `date_created`, `contents` и `rating`. Если первая запись `<post>` будет удалена, то будет неизвестно, что в этой таблице есть колонка `rating`. Все это означает, что первый элемент полностью определяет структуру таблицы. Получается, что первый элемент должен содержать значения для всех колонок, значения которых Вы зададите в следующих элементах. Если в каком-нибудь из следующих элементов будет задано значение атрибута, не описанного в первом элементе, то этот атрибут будет игнорирован. В Примере 9.2 я показал как эта особенность влияет на порядок элементов в наборе данных. По этой же причине я был вынужден второй элемент `<post_comment>` передвинуть на первое место, чтобы иметь возможность задать значение колонки `url`.

Есть способ обойти ограничение явного задания значения `NULL` в наборе данных формата Flat XML, для этого можно использовать набор данных типа "Замена" ("Replacement").

Набор данных в формате XML

Формат Flat XML не только простой, но и ограниченный. У него есть более функциональная альтернатива - формат XML. Это более структурированный xml формат, он позволяет явно и подробно описывать набор данных. Пример 9.3 демонстрирует формат XML и эквивалентен предыдущему примеру.

Пример 9.3: Набор данных в формате XML

```
<?xml version="1.0" encoding="UTF-8" ?> <dataset>
<table name="post"> <column>post_id</column>
<column>title</column>
<column>date_created</column>
<column>contents</column> <column>rating</column>
<row> <value>1</value> <value>My First Post</value>
<value>2008-12-01 12:30:29</value> <value>This is
my first post</value> <value>5</value> </row> <row>
<value>2</value> <value>My Second Post</value>
<value>2008-12-04 15:35:25</value> <value>This is
my second post</value> <null /> </row> <row>
<value>3</value> <value>My Third Post</value>
<value>2008-12-09 03:17:05</value> <value>This is
my third post</value> <value>3</value> </row>
</table> <table name="post_comment">
<column>post_comment_id</column>
<column>post_id</column> <column>author</column>
<column>content</column> <column>url</column> <row>
<value>1</value> <value>2</value>
<value>Tom</value> <value>While the topic seemed
interesting the content was lacking.</value> <null
/> </row> <row> <value>2</value> <value>2</value>
<value>Frank</value> <value>That is because this is
simply an example.</value>
<value>http://phpun.it</value> </row> </table>
<table name="current_visitors">
<column>current_visitors_id</column>
<column>ip</column> </table> </dataset>
```

По сравнению с Flat XML форматирование более многословное, однако несколько проще в понимании. В качестве корневого используется элемент `<dataset>`. В состав корневого элемента входят один или несколько элементов `<table>`. Каждый элемент `<table>` должен иметь атрибут `name`, значение которого эквивалентно таблице, которую представляет элемент. Элемент `<table>` включает один или несколько элементов `<column>`, которые хранят имена колонок. Кроме того, элемент `<table>` может включать несколько элементов `<row>`, эти элементы могут и отсутствовать. Элемент `<row>` в конечном счете и хранит данные, которые должны быть

сохранены/удалены/изменены или использованы для проверки текущего состояния базы данных. В элементе <row> должно быть столько же дочерних элементов сколько элементов <column> описано в <table>. Последовательность этих дочерних элементов определяется последовательностью элементов <column>, описанных в составе таблицы. В элементе <row> может быть два вида дочерних элемента. Вы можете использовать <value>, чтобы задать значение, которое надо вставить в колонку. Используемый формат аналогичен формату, значения атрибута в случае Flat XML. Чтобы явно показать, что колонка должна содержать значение <null>, можно использовать элемент NULL. В элементе <null> не может быть дочерних элементов или атрибутов.

Чтобы проверить набор данных в формате XML, можно воспользоваться DTD, см. Пример 9.4.

Пример 9.4: Набор данных в формате XML DTD

```
<?xml version="1.0" encoding="UTF-8"?> <!ELEMENT
dataset (table+) | ANY> <!ELEMENT table (column*,
row*)> <!ATTLIST table name CDATA #REQUIRED >
<!ELEMENT column (#PCDATA)> <!ELEMENT row (value |
null | none)*> <!ELEMENT value (#PCDATA)> <!ELEMENT
null EMPTY>
```

В Таблице 9.2 приведен перечень XML элементов набора данных.

Таблица 9.2. Описание XML элементов набора данных

Элемент	Цель	Содержание	Атрибуты
<dataset>	Корневой элемент набора данных.	Один или несколько элементов <table>.	нет
<table>	Определяет таблицу набора данных	Один или несколько элементов <column> и несколько необязательных элеменов <row>.	name - имя таблицы.
<column>	Определяет колонку текущей таблицы.	Имя колонки таблицы.	нет
<row>	Определяет ряд в таблице.	Один или несколько элементов <value> или <null>.	нет
<value>	Задает значение колонки.	Содержит значение для соответствующей колонки.	нет
<null>	Здает значение колонки NULL.	нет	нет

Набор данных в формате CSV

Формат XML - это отличное средство структурирования данных. Время от времени появляется необходимость внесения ручных изменений в такой файл. Не многие текстовые редакторы предоставляют возможность простого редактирования XML-файла. Возможно, формат CSV будет для Вас более удобным. Набор данных в формате CSV очень прост для понимания. Каждый CSV файл представляет одну таблицу. Первая строка CSV файла должна описывать структуру таблицы в виде списка имен колонок, все следующие строки файла должны соответствовать структуре, описанной в первой строке.

Для работы с набором данных в формате CSV предназначен класс `PHPUnit_Extensions_Database_DataSet_CsvDataSet`. Конструктор класса принимает три параметра: `$delimiter`, `$enclosure` и `$escape`. Эти параметры позволяют специфицировать формат записей файла. Благодаря этим настройкам файл не обязательно должен полностью соответствовать стандарту CSV. По умолчанию запятая является символом разделения полей, а сами поля должны быть заключены в двойные кавычки. Если в значение входит символ двойной кавычки, то должна быть добавлена еще одна экранирующая. Параметры по умолчанию максимально соответствуют стандарту CSV, однако есть возможность задать свои настройки.

Раз уж мы заговорили про класс `PHPUnit_Extensions_Database_DataSet_CsvDataSet`, то надо упомянуть про метод `addTable()`. Этот метод позволяет добавить CSV файл в набор данных как таблицу. Метод `addTable` принимает два параметра. Первый параметр `$tableName` определяет имя добавляемой таблицы. Второй параметр `$csvFile` указывает путь к файлу CSV. Вы можете вызывать `addTable()` для каждой таблицы, которую хотите добавить в набор данных.

В Примере 9.5 демонстрируется как из трех CSV файлов можно получить тестовое окружение для тестирования базы данных.

Пример 9.5: Набор данных в формате CSV

```
--- fixture/post.csv ---
post_id,title,date_created,contents,ratin
g
1,My First Post,2008-12-01 12:30:29,This
is my first post,5
2,My Second Post,2008-12-04 15:35:25,This
is my second post,
3,My Third Post,2008-12-09 03:17:05,This
is my third post,3

--- fixture/post_comment.csv ---
post_comment_id,post_id,author,content,ur
l
1,2,Tom,While the topic seemed
interesting the content was lacking.,
2,2,Frank,That is because this is simply
an example.,http://phpun.it

--- fixture/current_visitors.csv ---
current_visitors_id,ip

--- DatabaseTest.php ---
<?php
require_once
'PHPUnit/Extensions/Database/TestCase.php'
;
require_once
'PHPUnit/Extensions/Database/DataSet/CsvD
ataSet.php';

class DatabaseTest extends
PHPUnit_Extensions_Database_TestCase
{
    protected function getConnection()
    {
        $pdo = new
PDO('mysql:host=localhost;dbname=testdb',
'root', '');
        return $this-
>createDefaultDBConnection($pdo,
'testdb');
    }
```

```
        protected function getDataSet()
        {
        $dataSet = new
    PHPUnit_Extensions_Database_DataSet_CsvDa
    taSet();
        $dataSet->addTable('post',
    'fixture/post.csv');
        $dataSet->addTable('post_comment',
    'fixture/post_comment.csv');
        $dataSet-
    >addTable('current_visitors',
    'fixture/current_visitors.csv');
            return $dataSet;
        }
    }
    ?>
```

* This source code was highlighted with Source Code Highlighter.

Файл в формате CSV очень удобно редактировать в любом текстовом редакторе, но при работе с эти стандартом есть некоторые сложности с заданием значения `NULL`, мы рассмотрели эти сложности на примере Flat XML. Нет естественного способа явно задать значение `NULL`.

Метки: PHP Web PHPUnit Тестирование

Комментарии.

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

Ratio 14.09.2012 18:26:47

А дальнейший перевод планируется?

Sergey 16.09.2012 9:52:16

А дальнейший перевод планируется?
Нет, не планирую.

Пожалуйста авторизуйтесь или зарегистрируйтесь.