Unit-Tests mit PHPUnit

Ein Überblick zu „Test Driven Development“

- Unit-Tests mit PHPUnit
- Philosophie des TDD
- Vorgehensweise / Regeln
- Vorteile des TDD mit Unit-Tests
- Werkzeuge
- Stolpersteine
- Andere Testmethoden

Philosophie des TDD

- Tests werden vor dem Programmcode erstellt
  - „Grey-Box-Test“
  - Betriebsblindheit wird damit vermieden
- Entwicklung erfolgt in möglichst kleinen Einheiten – den Units.

Vorgehensweise / Regeln

- Schreibe Tests für das erwünschte fehlerfreie Verhalten, für schon bekannte Fehlschläge oder für das nächste Teilstück an Funktionalität, das neu implementiert werden soll. Diese Tests werden vom bestehenden Programmcode erst einmal nicht erfüllt bzw. es gibt diesen noch gar nicht.
- Ändere/schreibe den Programmcode mit möglichst wenig Aufwand, bis nach dem anschließend angestoßenen Testdurchlauf alle Tests bestanden werden.
- Räume dann im Code auf (Refactoring): Entferne Wiederholungen (Code-Duplizierung), abstrahiere wo nötig, richte ihn nach den verbindlichen Code-Konventionen aus etc.

Vorteile des TDD mit Unit-Tests

- Einfachere Metrik des Programmcodes (Cyclomatic und NPath Complexity)
- Weniger und lockerere Abhängigkeiten
- Einfacheres und „sichereres“ Refactoring
- Leichterer Einstieg in existierenden Code
- Tests dienen auch als Dokumentation und Spezifikation

Werkzeuge

- Test-Framework: PHPUnit, PHPSpec
- Job-Runner: Jenkins, CruiseControl

Stolpersteine

- Konsequenz und Disziplin
- Vermeintlicher Mehraufwand
- Zu sehr verzahnte Module (viele Tests müssen angepasst werden, wenn etwas geändert wird)
- Trügerische Sicherheit der 100 % Codeabdeckung

Andere Testmethoden

- „Behavior Driven Development“ (BDD)
  - an natürlicher Sprache orientiert (Wenn-Dann-Sätze)
  - interne Programmstruktur wenig relevant
  - QA, Kunde, Product Owner können diese Tests schreiben und lesen