WORKSHOP ZU UNIT TESTS MIT MOCKERY

TESTEN MIT PHPUNIT

EINLEITUNG

- Wie unterscheidet sich Unit von Integrations-, Funktional- und Acceptancetests
- Welche Erwartungen werden an die einzelnen Testgruppen gestellt
- Wie deutet man den Code Coverage Report?
- Wie arbeitet der Code Coverage Report?
- Welche Bedeutung haben Mocks und Stubs?
- Was ist der Unterschied zwischen PHPUnit Mock und Mockery?

UNIT VS INTEGRATION VS FUNCTIONAL

- Unit Test testen die kleinste Einheit
 - Funktioniert diese wie gewünscht?
 - Wird aus Entwicklersicht geschrieben
 - ▶ Testet nicht die Integration/Interaktion (mit) anderer Klassen
 - Keine Daten(bank)abhängigkeiten
 - Abhängige Klassen werden über Stubs oder Mocks bereitgestellt
 - Coverage sollte immer mit den Funktionen deklariert werden die man testet

- Integrationstest
 - Prüfen die Zusammenarbeit der verschiedenen Klassen
 - Abhängige Klassen sollten nicht gemocked werden
 - Ausgehende HTTP/S Anfragen sollten gemocked werden
 - APIs sollen ihre "Verträge" wie sie Daten bereitstellen beibehalten

- Funktionale Tests
 - Werden aus User-Sicht geschrieben
 - Es sollte eine produktiv nahe Umgebung geschaffen werden
 - So wenig wie möglich (am Besten nichts) mocken
 - Testet den gesamten Ablauf von Eingabe bis Ausgabe

- Acceptance Test UI
 - Prüfen die Funktionalität einer oder mehrerer (verbundener) Webseiten
 - Automatische Eingabe in Formfelder (positiv wie negativ)
 - Prüfen der Ergebnisse
 - Vergleichen der Stylings vor und nach Änderungen

CODE COVERAGE

NUTZEN UND ERGEBNISSE

NUTZEN VON CODE COVERAGE

- Dient zur Darstellung der getesteten/durchlaufenen Codezeilen
- Wurden mit den Tests alle Zeilen des Codes geprüft
- Keine Garantie das die Funktionen logisch korrekt funktionieren
- Detailliertere Tests möglich
- Coverage Abdeckung (in %) als mindest Standard festlegen
- Kann jedoch sinken wenn Klassen (beim Refactoring) entfallen

ERGEBNISSE VON CODE COVERAGE

- Aufschluss über getestete Codezeilen
- Aufschluss über nicht getestete Codezeilen
- Gibt immer alle durchlaufenen Codezeilen zurück
- Kann jedoch auch falsch dargestellt werden
- Tests können irrtümlich Codezeilen als getestet markieren

PHPUNIT MOCK VS MOCKERY

- Mock Logik von PHPUnit vereinfacht das unabhängige Testen
- Mockery ermöglicht einen komplexeren Mock
- Expects von PHPUnit Mocks werden beim Durchlaufen als erfolgreiche Asserts gewertet
- Mockery erlaubt das Testen der erwarteten aufgerufen Methoden des Mocks

NACHTEILE PHPUNIT MOCK

- Es kann sich zu sehr auf die selbst angelegten Mocks verlassen werden
- Rückgabewerte werden nicht mehr getestet
- Es werden die Annahmen einer Funktion aus einem Mock getestet statt die Funktionalität der primären Funktion
- Die Mocks können ein falsches Bild über die getestete Funktionalität liefern

NACHTEILE MOCKERY

- Externe Bibliothek die extra eingebunden werden muss
- Neue Logik zur Erstellung von Mock-Objekten