Sitewards

Unit-Tests in Magento

- Installation der Extension EcomDev_PHPUnit
- Einrichtung der Extension in PHPStorm
- Aufbau der Testordner
- Testen der Konfiguration
- Controller Tests
- Testen von Models / Helpers / Blocks
- Fixtures
- Providers
- Expectations
- Test Doubles

Sitewards

- Herunterladen der Extension unter
  https://github.com/IvanChepurnyi/EcomDev_PHPUnit/tree/dev
- Kopieren in Magento-Verzeichnis
- Patch der Extension
  https://github.com/airbone42/EcomDev_PHPUnit/commit/a71b97f8ee5c715e5d945463a46f5b6b4e1a4824
- Anpassung der Datei app/etc/local.xml.phpunit
- Konfiguration von PHPStorm

  Settings -> PHP -> PHPUnit -> Use configuration file
- Handbuch der Extension unter http://bit.ly/mR6uKc

Vorbereitung der eigenen Extension für Tests

- Erweiterung der Your_Module.xml

```
<depends>
        <EcomDev_PHPUnit />
        ...
</depends>
```

- Erweiterung der config.xml

```
<config>
        <phpunit>
                <suite>
                        <modules>
                                <Sitewards_B2BProfessional />
                        </modules>
                </suite>
        </phpunit>
        ...
</config>
```

- Anlegen des Verzeichnisses „Test“

Sitewards

- Ist Modul aktiv?
- Ist Modul in richtiger Version vorhanden?
- Wurde Setup richtig definiert?
- Wurden Models richtig definiert?
- Wurden Blöcke richtig definiert?
- Wurden Helper richtig definiert?
- Wurden Layouts richtig definiert?

Sitewards

- Wurden die Rewrites richtig gemacht?

  ```
  $this->assertRequestControllerModule('xyz');
  ```

- Beinhaltet der Response bestimmte Texte?

  ```
  $this->assertResponseBodyContains('xyz');
  ```

- Wurden Block-Methoden aufgerufen und wenn ja, wie oft?

  ```
  $this->assertLayoutBlockActionInvokedAtLeast('block_name',
  'blockMethod', $countTimes, '');
  ```

Sitewards

- Models, Blocks und Helpers können alle über EcomDev_PHPUnit_Test_Case getestet werden
- Wurde ein Event dispatched?
- Gebrauch von Test Doubles, Data Providers und Expectations sehr sinnvoll

Sitewards

- Fixtures werden in YAML definiert
- Per Konvention unter Klassenname/fixtures/testMethodName.yaml
- Fixtures für folgende Bereiche:
  - Scope
    - Website
    - Group
    - Store
  - Config / ConfigXml
  - EAV
    - Batch-Möglichkeit über https://github.com/sitewards/Hackathon-FixtureGenerator

Sitewards

- Expectations werden in YAML definiert
- Per Konvention unter

  Klassenname/expectations/testMethodName.yaml
- Keine besondere Form
- Im Test über $this->expected(„xyz“) aufrufbar
- Rückgabewert ist ein Varien_Object mit üblichen getX und setX

Sitewards

- Providers werden in YAML definiert
- Per Konvention unter Klassenname/providers/testMethodName.yaml
- Array mit jedem Element als Parameter für die Testmethode
- Testmethode wird so oft aufgerufen wie die Provider definiert wurden

Sitewards

- Folgende Objekttypen können gemockt werden:
  - Models
  - ResourceModels
  - Blocks
  - Helpers
- Replacement als
  - model
  - resource_model
  - singleton
  - resource_singleton
  - block
  - helper