ARQUILLIAN IN DER PRAXIS

Stefan Hildebrandt / @hildebrandttk

FOLIEN ALS HTML-PRÄSENTATION

http://bit.ly/1FzQAkg

MEIN EINSTIEG

JAVA 2 EE

- jUnit
- Client-zugriff
  - Proxies im Container
  - InitialContextFactory
    - NameNotFoundException
    - NamingException
  - RemoteException

J2EE MIT SPRING

- war-Deployment
- spring-test
    - Projektspezifische "Frameworks"
        - Deployment mit Mocks
        - Datenbanksetup
        - Testdaten

JAVA EE 6

JAVA EE OHNE SPRING
... OHNE SPRINGTEST

WAS NUN?

ARQUILLIAN!

HISTORIE VON ARQUILLIAN

- Entstand bei der Implementierung des JSR 299
- Weiterentwicklung durch JBoss
  - Fokus: Test ihrer Server und Frameworks
  - Euer Projektfokus?

FEATURES

- Steuerung des Container–Lebenszyklus
- Erstellung von Artefakten
- Installieren von Artefakten
- Bereitstellung von Komponenten in den Test
- Plug-in-Konzept für Erweiterungen

EINSATZBEREICHE

UNIT-TESTS

```java
public class PetTypeControllerTest {

@Test
public void testSwitchSortOrder_ascending_to_descending() {
   PetTypeController petTypeController
         = new PetTypeController(SortOrder.ascending);

   petTypeController.switchSortOrder();

   assertEquals(SortOrder.descending, petTypeController.getPetTypeSortOrder())
}
```

```java
@Test
public void testSwitchSortOrder_descending_to_ascending() {
   PetTypeController petTypeController
            = new PetTypeController(SortOrder.descending);

   petTypeController.switchSortOrder();

   assertEquals(SortOrder.ascending, petTypeController.getPetTypeSortOrder());
}
}
```

UNIT-TESTS

- Kein Arquillian
  - Zu hohe Komplexität
  - Relativ langsam

UNIT-TEST

... FÜR JAVA EE ELEMENTE

- Interceptor
- Decorator
- Filter
- JSF-Komponenten
- ...

JPA

- TDD / BDD
  - Entwicklung von komplexen Mappings
  - Schrittweise Erstellung von Queries
  - Als Input- -> Output-Tests
  - Sehr gute Absicherung für Refactorings

```java
@RunWith(Arquillian.class)
@PersistenceTest
@UsingDataSet("OwnerDaoImplTest.yml")
@Cleanup(strategy = CleanupStrategy.USED_TABLES_ONLY)
public class OwnerDaoImplTest {

   //Need this instance for arquillian-dbunit loading
   @PersistenceContext(unitName = "javaee7petclinic")
   private EntityManager entityManager;

   @Inject
   private OwnerDao ownerDao;
```

```yaml
types:
  - id: 1
    name: Katze
  - id: 2
    name: Hund
owners:
  - id: 1
    first_name: Stefan
    last_name: Hildebrandt
    address: Hauptstr.
    city: Oldenburg
    telephone: 0441-666
  - id: 2
    first_name: Horst
    last_name: Müller
    address: Hauptstr.
    city: Oldenburg
    telephone: 0441-765
```

```
@Test
public void testFindOwnersWithVisitWithinGivenTimeFrame_DayForLastVisit() {
   final List<Owner> owners
            = ownerDao.findOwnersWithVisitWithinGivenTimeFrame(
         SIMPLE_DATE_FORMAT.parse("2015-03-11"), new Date());
   assertThat(owners, CoreMatchers
            .<Owner>hasItem(hasProperty("firstName", is(equalTo("Stefan")))));
   assertThat(owners, not(CoreMatchers
            .<Owner>hasItem(hasProperty("firstName", is(equalTo("Horst"))))));
}
```

```java
@Test
public void testFindOwnersWithVisitWithinGivenTimeFrame_OnlyOneVisitOnOneDayIr
   final List<Owner> owners = ownerDao.findOwnersWithVisitWithinGivenTimeFrame
            SIMPLE_DATE_FORMAT.parse("2015-03-10"),
            SIMPLE_DATE_FORMAT.parse("2015-03-10"));
   assertThat(owners, not(CoreMatchers
            .<Owner>hasItem(hasProperty("firstName",is(equalTo("Stefan"))))));
   assertThat(owners, CoreMatchers
            .<Owner>hasItem(hasProperty("firstName", is(equalTo("Horst")))));
}
```

```
@Test
public void testFindOwnersWithVisitWithinGivenTimeFrame_AllVisitsUntilSomeDate
   final List<Owner> owners
            = ownerDao .findOwnersWithVisitWithinGivenTimeFrame(
            SIMPLE_DATE_FORMAT.parse("1990-03-10"),
            SIMPLE_DATE_FORMAT.parse("2015-03-10"));
   assertThat(owners, CoreMatchers
            .<Owner>hasItem(hasProperty("firstName", is(equalTo("Stefan")))));
   assertThat(owners, CoreMatchers
            .<Owner>hasItem(hasProperty("firstName", is(equalTo("Horst")))));
}
```

SERVICES

- Zusammenspiel mehrerer Elemente eines Services
- Mit/ohne Datenbank
- Mocks oder Simulatoren
  - Angepasstes Deployment
  - @Alternative

```java
@RunWith(Arquillian.class)
public class VetWebserviceTest {
  private static final String BASE = "../../../../";
  private static final String POM_PATH = BASE + "pom.xml";

  @Deployment(testable = false)
  public static WebArchive createDaoDeployment() {
    File[] deps = Maven.resolver().loadPomFromFile(POM_PATH)
         .importRuntimeDependencies().resolve().withTransitivity().asFile();
    WebArchive war;
       = ShrinkWrap.create(WebArchive.class, "remote-service-deployment.war")
         .addClasses(VetWebservice.class, JaxRsActivator.class, Vets.class)
         .addClasses(VetDao.class, Vet.class, Specialty.class)
         .addClasses(VetDaoMock.class)
         .addAsLibraries(deps);
    return war
  }
```

```java
@Test
@RunAsClient
public void testFeed() throws IOException {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    final String uri = baseURL + "rest/vets/feed";
    HttpGet httpGet = new HttpGet(uri);
    CloseableHttpResponse response = httpClient.execute(httpGet);
    assertEquals(200, response.getStatusLine().getStatusCode());
    final String datePattern = new SimpleDateFormat("yyyy-MM-dd'T'hh:mm'.*'").
    final HttpEntity entity = response.getEntity();
    final ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    entity.writeTo(outputStream);
    assertThat(outputStream.toString(), RegexMatcher.matchesRegex(
        "..."));
}
```

SYSTEM

- DB mit Echtdaten (obfuscated) in VM / Container
- Simulatoren für Nachbarsysteme
- Testschnittstellen:
  - Service-Ebene
    - Remote
    - Local
  - UI

DEMO UI-TEST

SYSTEMINTEGRATION MIT NACHBARSYSTEMEN

- Komplett definierter und exklusiver Stand
  - Kein Setup über reguläre UI/Services
  - Sehr robuste und verlässliche Tests
  - Relativ schnell
  - Idealerweise Ausführung im CI vor Integration
- z.B. mit vagrant oder docker (mit compose)
- Simulatoren für nicht verfügbare Systeme (SAP, ...)

AKZEPTANZTESTS

- Externe, fachliche Sicht auf Tests
- Verlinkung zu Anforderungen
- Langfristige Absicherung von Anforderungen
- Klassische Probleme:
  - Langsam
  - Seltene Ausführung
  - Weit weg vom Code
  - Kaum Feedback für Entwicklung

CUKESPACE

- Cucumber-Runner für Arquillian
- Akzeptanztests von Unit- bis Systemintegrationsebene
- Setup wie bei Systemintegrationstests
  - Wesentlich Robuster
- Lokale Ausführung im CI -> Schnelles Feedback
- Regelmäßige Ausführung im CI -> Schnelles Feedback
- Eigener Vortrag

NICHT-FUNKTIONALE TESTS

- Mit JMeter...
- Über Messpunkte bei der Testausführung
- Indirekt: Laufzeit einer größeren Suite

ZUSAMMENFASSUNG EINSATZBEREICHE

- Tests immer auf kleinstmöglicher Ebene
- Erst bei konkreten Problemen Tests in höherer Ebene
- Wenige vollumfängliche Integrationstests
- Durch die integrierte Kombination wertvoll

CLIENT-/SERVER-MODE

CLIENT

- @Deployment(testable = false)
- Nur Deployment des Archivs
- Tests laufen in der Client-JVM

SERVER

- @Deployment
- Archiv wird erweitert
- Tests laufen im Container
- Übertragung der Ergebnisse an den Test
- Voller Zugriff auf Ressourcen/Komponenten

CONTAINERMANAGEMENT

REMOTE

Verwendung einer bestehenden laufenden Instanz

↗ Am schnellsten

↗ Ergebnis realitätsnah

↘ Manuelles Setup

↘ Manuelles Starten/Stoppen

↘ Hängenbleiben von Deployments

MANAGED

Verwendung einer bestehenden Installation
Start/Stop durch Arquillian

- ↗ Ergebnis realitätsnah
- ↗ Automatisches Starten/Stoppen
- ↘ Etwas längere Laufzeiten
- ↘ Manuelles Setup
- ↘ Hängenbleiben von Deployments

EMBEDDED

Verwendung eines embedded-Containers in der Test-JVM

- ↗ Kein Hängenbleiben von Deployments
- ↗ Automatisches Startup
- ↗ Automatisches Setup
- ↘ Etwas längere Laufzeiten
- ↘ Ergebnis ggf. nicht vollständig realitätsnah
- ↘ Konflikte von Bibliotheken (z.B. guava)

DEBUGGING

↗ Embedded: Alles
→ Client: Nur den Test
↘ Ansonsten: Remote-Debugging

ARTEFAKT-ERSTELLUNG

SHRINKWRAP

- Erstellung von Artefakten
- Programmatisch
- Sehr flexibel -> Gut für Unit-tests

DOPPLUNG DER BUILDSYSTEMBESCHREIBUNG

- ↗ Maven/Gradle-Importer
- ↘ Resources, Pre-/Postprocessing, ...

AUSFÜHRUNGSZEIT

DEPLOYMENT

- Ein Deployment je Testklasse
  → Unit-Test
  ↘ Integrationstests

ARQUILLIAN SUITE EXTENSION

- Idee: Ein Deployment je Test-Classpath
  - Auswirkungen auf Projektlayout
- Alternativ: Mehrere Deployments
  - Benannte Deployments
  - Zuordnung über Namen
- Eigene Events für Hooks
- Nicht im Fokus anderer der Kernentwicklung und anderer Extensions
  - Kompatibilitätsprobleme mit anderen Extensions

TOMEE

- Benannte Deployments
- Geänderte Deployment-Events
- Ab Tomee 1.7.2

DEPENDENCY HANDLING

- maven-/gradle-Importer
  - Analyse komplexer Projekte dauert
- Alternativ: directory-Importer
  - Befüllung mittels maven/gradle

DATENBANK SETUP (UNIT)

- In-Memory-Datenbanken
- Ggf. Testbezogen laden

DATENBANK SETUP (INTEGRATION+)

- Once-Per-Suite
- Dedizierte Daten je Test
- Mögliche Verfahren
  - DB Unit
  - Spezialwerkzeuge für Datenbanken
  - Fertige Container oder VMs

UI-TESTS

- Definierter Systemzustand
  - Datenbank
  - Sonstige Daten/Zustände
- Langsamere Systeme mocken
- Inspections in den Mocks
- Bedienung des UIs vereinfachen

MULTI-THREADING

- Viel Komplexität
- Ab 4 Threads deutlicher Gewinn
- Interne vs. externe Lösung

INTERNES MULTITHREADING

- Parallele Ausführung der Tests in eine Suite in einer VM
- Notwendig:
  - Synchronisatisiertes Suite-Deployment
  - Disjunkte Testdaten
  - Threadsave Mocks-/Simulatoren
- junittoolbox-ParallelSuite
  - Nicht mit Arquillian kombinierbar

EXTERNES MULTITHREADING

- Parallele Testausführung mit maven/gradle
  - mehrere JVMs
  - Notwendig:
    - Mehrere Container, DBs und Umsysteme
      - Arquillian Cube Extension

INTERESSANTE ERWEITERUNGEN

DRONE/GRAPHENE

- Unterstützung für Seleniumtests
- Bereitstellung von Webdriver-Instanz
- Implementierung des Page-Objekt-Patterns
- Erweiterung des Page-Objekt-Patterns um Komponenten
- Eigener Vortrag

WARP

- Untersuchung von Servlet-Requests im Test
- Mocking von Servlet-Responses im Test

TRANSACTION

- Transaktionssteuerung in Tests

PERSISTENCE

- Ausführung von SQL-Skripten für das Schema
- DBUnit-Integration
  - Befüllung einer DB
  - Assert auf DB-Ebene

DOKUMENTATION

- Offizielle Guides
- Arquillian Showcase
- Quellcode der Tests
- Quellcode der Tests von Open Source Projekten

ZUSAMMENFASSUNG

- ↗ Ideal für Java EE-Unit-Tests
- ↗ Auf allen Teststufen verwendbar
- ↗ Viel Flexibilität
- ↘ Viel Komplexität
- ↘ Suite nicht im Fokus

ALTERNATIVEN

- cdi-unit
- Apache DeltaSpike CdiCtrl
- spring-test
- spring-boot

FOLIEN

http://bit.ly/1FzQAkg

STEFAN HILDEBRANDT - CONSULTING.HILDEBRANDT.TK

- Beratung, Coaching und Projektunterstützung
- Java EE
- Buildsysteme gradle und maven/ant-Migration
- Testautomatisierung
- Coach in agilen Projekten
- DevOps