Web-Anwendungen mit Arquillian testen

Michael Kotten | **open** knowledge

@michaelkotten
@_openKnowledge


Wozu denn testen?
Ich mach doch keine
Fehler!

Wozu denn testen?

- Notwendig bei komplexen Systemen
- Sicherung von
  - Qualität
  - Funktionalität
- Wichtig bei agilen Entwicklungsprozessen
  - Schnelle Entwicklungszyklen
  - „sichere“ Refactorings

Probleme mit klassischen Tests

- Manuelles Testen von User Interfaces
  - Zeitaufwändig
  - Keine „specified conditions“ möglich
  - Keine komplette Testabdeckung möglich
  - Kein „continuous testing“
- Unit Tests
  - Außerhalb des Zielcontainers
  - Keine ausreichende Kontrolle der Funktionen im Container (DI, Transaktionen etc.)


Und jetzt?

Selenium?

- Pro:
  - Java-basiert
  - Unterstützung für div. Browser
  - Wiederverwendbarkeit
  - Gute Dokumentation
- Contra:
  - Testdeployment ist umständlich
  - API umständlich

BED-Con 2015

Beispiel Selenium

Web-Anwendungen mit Arquillian testen

```
// The Firefox driver supports javascript
WebDriver driver = new FirefoxDriver();
// Go to the Google Suggest home
driver.get("http://www.google.com/webhp?complete=1&hl=en");
// Enter the query string "Berlin Expert Days 2015"
WebElement query = driver.findElement(By.name("q"));
query.sendKeys("Berlin Expert Days 2015");
// Sleep until the div we want is visible or 5 seconds is over
long end = System.currentTimeMillis() + 5000;
while (System.currentTimeMillis() < end) {
  WebElement resultsDiv =
        driver.findElement(By.className("gssb_e"));
          if (resultsDiv.isDisplayed())
            break;
          }
        }
List<WebElement> allSuggestions =
        driver.findElements(By.xpath("//td[@class='gssb_a gbqfsf']"));
```

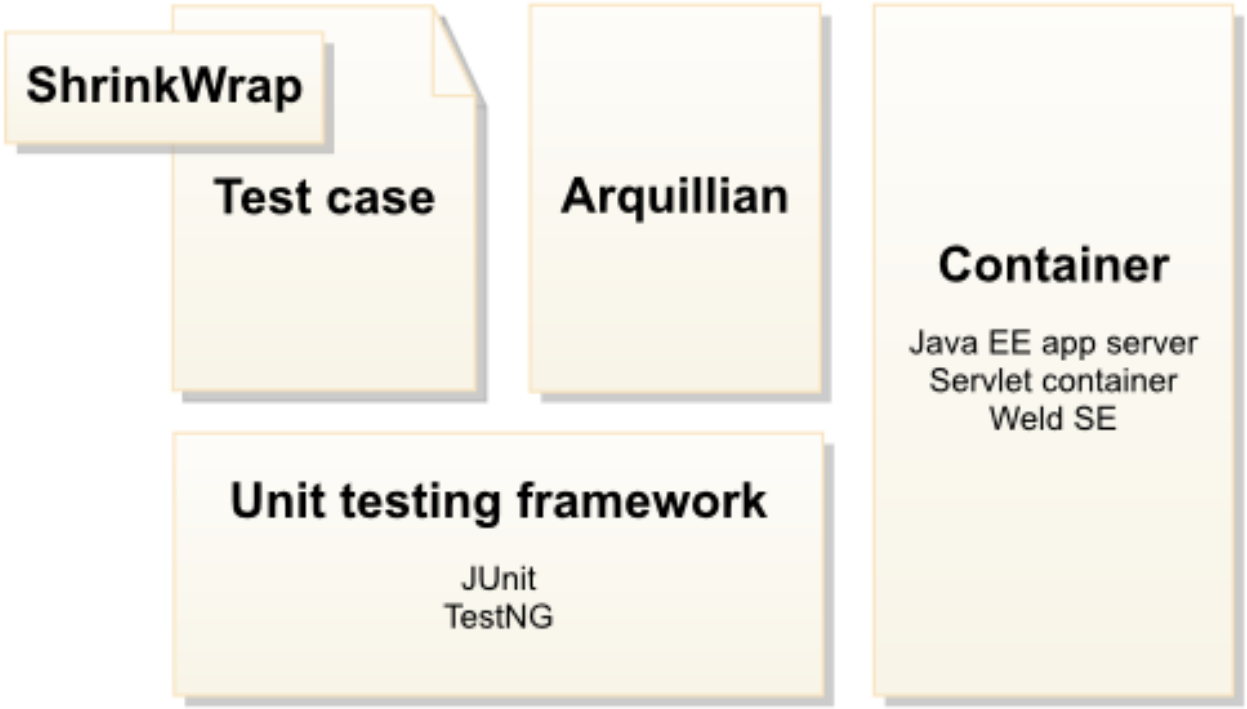
Die Lösung!

JBoss Arquillian

Web-Anwendungen mit Arquillian testen

- Automatisierte Integration Tests
  - Ausführbar im Container (embedded, remote oder managed)
  - Steuert Lifecycle des Containers
  - Deployment von Archiven per Shrinkwrap
  - Dependency Injection im Test
  - Erweiterbar über Service Provider Interface (SPI)
  - Div. Container verfügbar (u. a. Jboss AS, Glassfish, Tomcat, Jetty, Weld SE, Wildfly)

Architektur



JBoss Shrinkwrap

- Erzeugung von Archiven (JARs, WARs und EARs) zur Laufzeit
- Deploybar in jeden unterstützen Container (Jboss AS, Glassfish, Jetty, OpenEJB etc.)
- Voll integriert in den Deployment Mechanismus von Arquillian

Beispiel Shrinkwrap

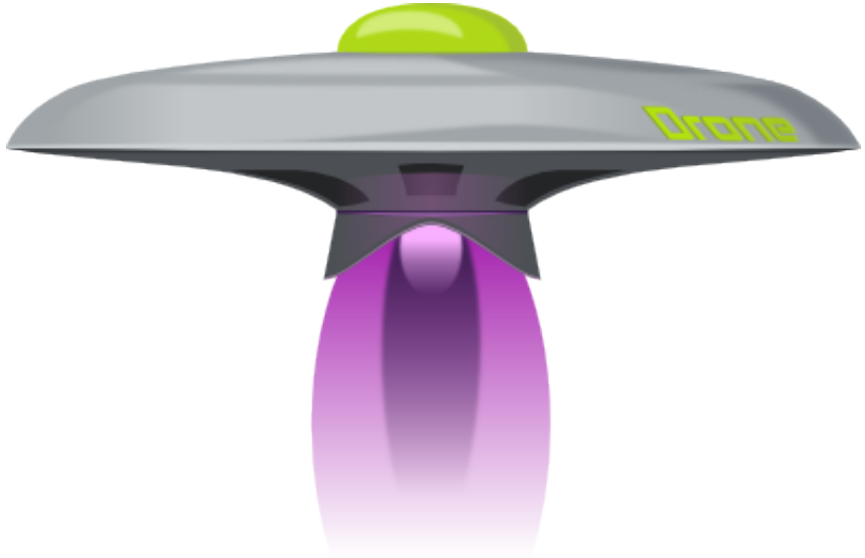

```
private static final String WEBAPP_SRC = "src/main/webapp";

@Deployment(testable = false)
public static Archive<?> createDeployment() {
    WebArchive archive = ShrinkWrap.create(WebArchive.class);
    archive.addClass(LoginController.class)
          .setWebXML(new File(WEBAPP_SRC, "WEB-INF/web.xml"))
          .addAsWebInfResource(
             new File(WEBAPP_SRC, "WEB-INF/faces-config.xml"))
          .addAsWebInfResource(EmptyAsset.INSTANCE, "beans.xml")
          .addAsWebResource(new File(WEBAPP_SRC, "login.xhtml");
    return archive;
}
```

BED-Con 2015

Arquillian Drone

Web-Anwendungen mit Arquillian testen

- Browser Lifecycle Management
- Zugriff auf Arquillian Deployments und Container
- Mehrere Browser pro Test möglich
- Steuerung des Browsers per WebDriver
  - Navigation
  - Zugriff auf WebElements
  - Screenshots



Beispiel Drone


```
@Drone WebDriver browser;

@ArquillianResource URL deploymentUrl;

@FindBy(id = "loginForm:username") WebElement username;
@FindBy(id = "loginForm:password") WebElement password;

@Test
public void should_login_successfully() {
    browser.navigate().to(deploymentUrl + "login.xhtml");
    username.sendKeys("demo");
    password.sendKeys("secret");
    browser.findElement(By.id("loginForm:login").click();

    assertTrue("User should be logged in!",
        browser.findElement(
            By.xpath("//li[contains(text(), 'Welcome')]")
               .isDisplayed());
}
```

Arquillian Graphene 2

Web-Anwendungen mit Arquillian testen

- Erweiterung für Selenium WebDriver
- Vereinfachte Fluent-API
- Abstraktion durch Page Objects und Page Fragments
- JQuery Selectors
- AJAX-enabled Testing
- Waiting API
- Request Guards

Page Fragments

- Bei wiederverwendbaren Komponenten
- Wiederverwendbar auch in Tests
- Kapselung von
  - Templates (ui:insert)
  - Widgets, wie z.B. calendar
  - Composite Components

Beispiel Page Fragments


```
public class AutocompleteFragment {

    @Root
    private WebElement root;

    @FindBy(css = "input[type='text']")
    private WebElement inputToWrite;

    public List<String> getSuggestions() {
        ...
    }
}
```

Beispiel Page Fragments


```
@Test
@RunWith(Arquillian.class)
@RunAsClient
public class AutocompleteTest {

    @FindBy(css = "div[class=\"rf-au\"]:nth-of-type(1)")
    private AutocompleteFragment autocomplete;

    ...

}
```

Page Objects

- Kapselung einzelner Seiten in Java Klassen
- Injection von WebElements (per @FindBy)
- Fachliche Methoden möglich, wie z.B. login(username, password)
  - Einfache Verwendung der WebElements
- Können Page Fragments beinhalten
- Per @Location inkl. URL

BED-Con 2015

Beispiel Page Objects

Web-Anwendungen mit Arquillian testen

```
@Location("login.xhtml")
public class LoginPage {

    @FindBy(id = "loginForm:username")
    private WebElement usernameInput;

    @FindBy(id = "loginForm:password")
    private WebElement passwordInput;

    @FindBy(id = "loginForm:login")
    private WebElement loginButton;

    public void login(String username, String password) {
        usernameInput.sendKeys(username);
        passwordInput.sendKeys(password);
        loginButton.click();
    }
}
```

BED-Con 2015

Beispiel Page Objects


```
@RunWith(Arquillian.class)
@RunAsClient
public class LoginTest {

    @Test
    public void testLogin(@InitialPage LoginPage page){
        page.login("demo", "secret");
        waitGui()
            .withTimeout(2, TimeUnit.SECONDS)
            .until()
            .element(By.xpath("//li[contains(text(),'Welcome')]"))
            .is()
            .visible();
    }
}
```

Arquillian Warp

„Warp fills the void between client-side and server-side testing“

Arquillian Warp

Web-Anwendungen mit Arquillian testen

- Gray-Box-Testing
- Client-seitige Aktionen (via WebDriver)
- Server-seitige Tests

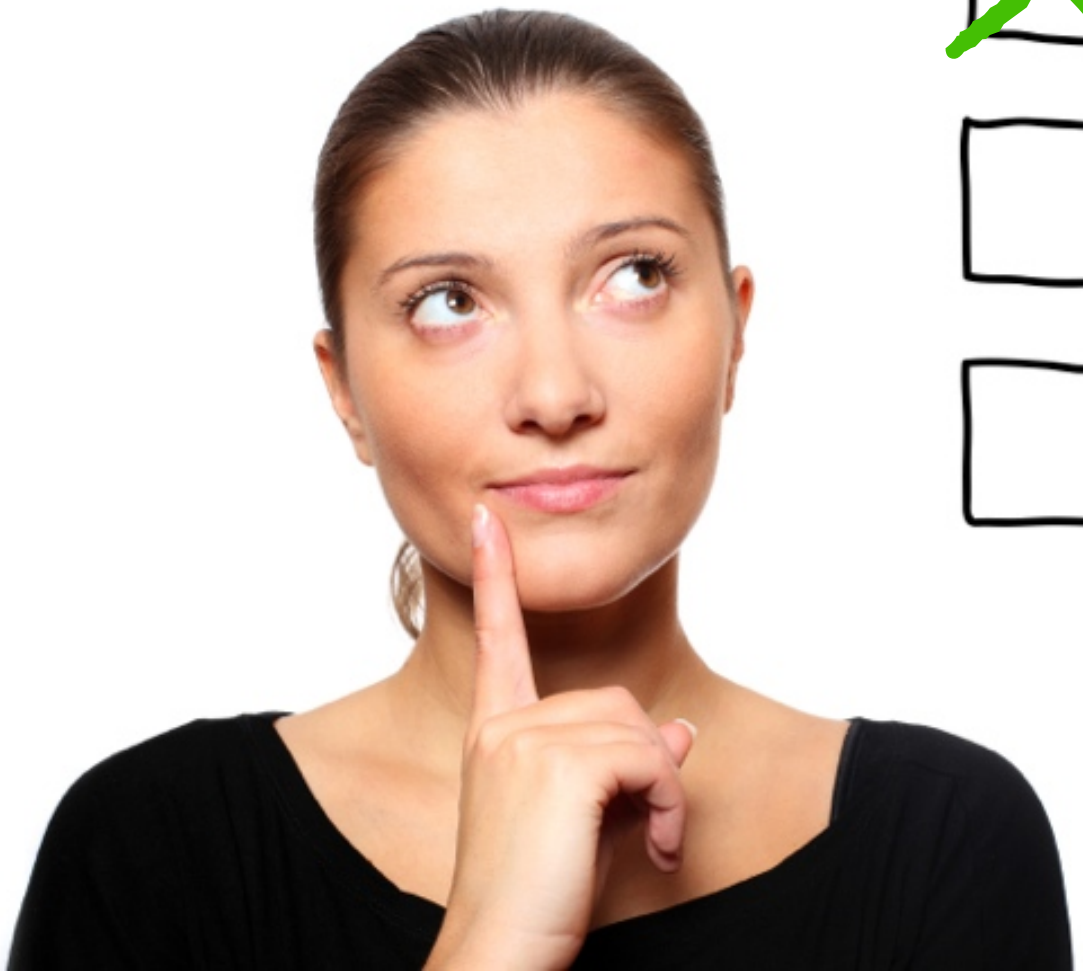
Beispiel Warp Test


```
@RunWith(Arquillian.class)
@WarpTest
@RunAsClient
public class LoginTest {

    @Deployment(testable = true)
    public static WebArchive deployment() {
        ...
    }

    @Test
    public void testLogin(@InitialPage LoginPage page) {
        ...
    }
}
```

Beispiel Warp Test


```
@Test
public void testLogin(@InitialPage LoginPage page){
    Warp.initiate(new Activity() {
            @Override
            public void perform() {
                page.login("demo", "secret");
            }
        }).observe(request().method().equal(HttpMethod.POST))
        .inspect(new Inspection() {
            private static final long serialVersionUID = -1L;

            @Inject
            private LoginController loginController;

            @AfterPhase(Phase.RENDER_RESPONSE)
            public void checkLogin() {
                assertTrue(loginController.isLoggedIn());
            }
        });
}
```

Fazit:



- [x] yes
- [ ] no
- [ ] maybe

Fazit

- Arquillian bietet...
  - Realitätsnahe Tests im echten Java EE Container
  - Einfache Kapselung einzelner Seiten und Komponenten
  - Intuitive Browsersteuerung per Selenium
  - Unterstützung div. Browser
  - Continuous Testing
- Aber...
  - Erweiterungen teilweise noch als Alpha-Version
  - Kann wartungsintensiv sein

Fragen?

Gibt es noch Fragen?

Dann los ...

Links

- Selenium: http://docs.seleniumhq.org
- Arquillian: http://www.arquillian.org
- Shrinkwrap: http://www.jboss.org/shrinkwrap
- Graphene: https://docs.jboss.org/author/display/ARQGRA2/Home
- Drone: https://docs.jboss.org/author/display/ARQ/Drone
- Warp: https://github.com/lfryc/arquillian.github.com/blob/warp-docs/docs/warp.adoc

Web-Anwendungen mit Arquillian testen

Michael Kotten | **open** knowledge

@michaelkotten
@_openKnowledge