Arquillian - a different kind of testing

Nils Heyer
Christian Neumann

Fontys Hogeschool voor Techniek en Logistiek

December 7, 2015

What is Arquillian

Advantages of testing with Arquillian

- Testing of all kind of beans or methods
- Many various ways of manipulating testing package
- Can keep the package for testing deployment small
- Allows to do different types of tests with one framework
- Able to test in different containers
- Contexts and Dependency Injection
- fast test due to ShrinkWrapper
- server-side debugging (you can just drop a breakpoint in the test or application code and debug the test)
- Many extensions are available/support for new extensions given

Disadvantages of testing with Arquillian

- Needs deployment for each test execution
- Takes some times for making test packages
- initial setup takes some time
- manual adjustments needed for Shrinkwrapper

Arquillian unit test

Simple Greeter test

```
@Inject
Greeter greeter;

@Test
public void should_create_greeting() {
    Assert.assertEquals("Hello, Earthling!",
        greeter.createGreeting("Earthling"));
}
```

- **@Injected** Dependency inject objects that are used during the tests, equals as using jUnits *@before* to set up tests
- **@Test** defines which methods are tests and should be executed
- **Asserts** are the same as in jUnit

Shrinkwrapper example

Greeter test Shrinkwrapper

```
@Deployment
public static JavaArchive createDeployment() {
    JavaArchive jar = ShrinkWrap.create(↩
        JavaArchive.class)
        .addClasses(Greeter.class, PhraseBuilder↩
            .class)
        .addAsManifestResource(EmptyAsset.↩
            INSTANCE, "beans.xml");

    return jar;
}
```

- **@Deployment** defines which method is run before a test is executed. It need to return the minimal archive which is needed to execute the tests of this class
- **JavaArchive** a class representing the structure of a jar file
- **ShrinkWrap.create** allows to create an archive of the specified class, in case of a unit test JavaArchive.class

Shrinkwrapper example

Greeter test Shrinkwrapper

```
@Deployment
public static JavaArchive createDeployment() {
    JavaArchive jar = ShrinkWrap.create(↩
        JavaArchive.class)
        .addClasses(Greeter.class, PhraseBuilder↩
            .class)
        .addAsManifestResource(EmptyAsset.↩
            INSTANCE, "beans.xml");

    return jar;
}
```

- **addClasses** allows to add classes,that are needed during the tests, to the jar
- **addAsManifestResource** allows to add a resource,that are needed for the execution e.g. database configuration file, to the jar
- **EmptyAsset.INSTANCE** creates an empty file

Complete test

Greeter test

```java
@RunWith(Arquillian.class)
public class GreeterTest {

    @Deployment
    public static JavaArchive createDeployment() {
        JavaArchive jar = ShrinkWrap.create(JavaArchive.class)
            .addClasses(Greeter.class, PhraseBuilder.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");

        return jar;
    }

    @Inject
    Greeter greeter;

    @Test
    public void should_create_greeting() {
        Assert.assertEquals("Hello, Earthling!",
            greeter.createGreeting("Earthling"));
    }
}
```

- **@RunWith(Arquillian.class)** is need for every test class to indicate that this test should be executed with Arquillian

Arquillian persistence test

Persistance shrinkwrapper

```
@Deployment
public static Archive<?> createDeployment() {
    // You can use war packaging...
    WebArchive war = ShrinkWrap.create(↩
        WebArchive.class, "test.war")
        .addPackage(Game.class.getPackage())
        .addAsResource("test-persistence.xml", "↩
            META-INF/persistence.xml")
        .addAsWebInfResource("jbossas-ds.xml")
        .addAsWebInfResource(EmptyAsset.INSTANCE↩
            , "beans.xml");
return war;
}
```

Arquillian persistence test

Persistance shrinkwrapper

```
    @PersistenceContext
    EntityManager em;

    @Inject
    UserTransaction utx;

    @Before
    public void preparePersistenceTest() throws Exception {
        clearData(); //private method
        insertData(); //private method
        startTransaction(); //private method
    }

    @Test
    public void shouldFindAllGamesUsingJpqlQuery() throws Exception {
        // given
        String fetchingAllGamesInJpql = "select g from Game g order by↩
              g.id";

        // when
        System.out.println("Selecting (using JPQL)...");
        List<Game> games = em.createQuery(fetchingAllGamesInJpql, Game↩
             .class).getResultList();

        // then
        System.out.println("Found " + games.size() + " games (using ↩
             JPQL):");
        assertContainsAllGames(games);
    }
```

Arquillian functional test

Functional shrinkwrapper

```
@Deployment(testable = false)
public static WebArchive createDeployment() {
    return ShrinkWrap.create(WebArchive.class, "login.war")
        .addClasses(LoginController.class, User.class, Credentials↩
             .class)
        // .addAsWebResource(new File(WEBAPP_SRC), "login.xhtml")
        // .addAsWebResource(new File(WEBAPP_SRC), "home.xhtml")
        .merge(ShrinkWrap.create(GenericArchive.class).as(↩
             ExplodedImporter.class)
            .importDirectory(WEBAPP_SRC).as(GenericArchive.class),
            "/", Filters.include(".*\\.xhtml$"))
        .addAsWebInfResource(EmptyAsset.INSTANCE, "beans.xml")
        .addAsWebInfResource(
            new StringAsset("<faces-config version=\"2.0\"/>"),
            "faces-config.xml");
}
```

Arquillian functional test

Functional shrinkwrapper

```
@Drone
DefaultSelenium browser;

@ArquillianResource
URL deploymentUrl;

@Test
public void should_login_with_valid_credentials() {
    browser.open(deploymentUrl.toString().replaceFirst("/$", "") +↩
          "/login.jsf");

    browser.type("id=loginForm:username", "user1");
    browser.type("id=loginForm:password", "demo");
    browser.click("id=loginForm:login");
    browser.waitForPageToLoad("15000");

    Assert.assertTrue("User should be logged in!",
        browser.isElementPresent("xpath=//li[contains(text(),'↩
             Welcome')]"));
}
```

Any open issues?

Not all understood?

- see `http://arquillian.org/guides/`

Questions?

Questions or remarks?