

Arquillian - a different kind of testing

Nils Heyer
Christian Neumann

Fontys Hogeschool voor Techniek en Logistiek

December 7, 2015

Introduction

Advantages & Disadvantages

Advantages of testing with
Arquillian

Disadvantages of testing
with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

- In the past and even today it is a problem to test JavaEE applications.
- Often logic is tested only with mockups.
- If run in a server often the runtime needed to managed in the project/ test itself

Arquillian is a possible solution to this problems

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

What is Arquillian

Arquillian - a
different kind of
testing

NH,CN

Arquillian is a Test framework, that can be used for:

- testing inside a remote or embedded container
- deploy an archive to a container so the test can interact as a remote client

It is capable of:

- Unit tests
- Persistence tests
- Functional tests

Arquillian integrates with other testing frameworks (e.g., JUnit or TestNG), allowing the use of IDE, Ant, and Maven test plugins

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Advantages of testing with Arquillian

Arquillian - a
different kind of
testing

NH,CN

- Many various ways of manipulating testing package
 - e.g. adding data source definition or persistence descriptor (persistence.xml) for testing purpose
- 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)

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Disadvantages of testing with Arquillian

Arquillian - a
different kind of
testing

NH,CN

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

Introduction

Advantages &
Disadvantages

Advantages of testing with
Arquillian

Disadvantages of testing
with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Arquillian unit test

Arquillian - a
different kind of
testing

NH,CN

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

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Shrinkwrapper example

Arquillian - a
different kind of
testing

NH,CN

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`

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Shrinkwrapper example

Arquillian - a different kind of testing

NH,CN

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

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Complete test

Arquillian - a different kind of testing

NH,CN

Greeter test

```
@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

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Arquillian persistence test

Arquillian - a different kind of testing

NH,CN

Persistence 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;
}
```

- **WebArchive** a class representing the structure of a war file
- **test-persistence.xml** copy the used persistence.xml into the archive
- **jbossas-ds.xml** add the server config as WEB-INF resource

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Arquillian persistence test

Arquillian - a different kind of testing

NH,CN

Persistence test

```
@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);
}
```

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Arquillian persistence test

Arquillian - a
different kind of
testing

NH,CN

- **UserTransactions** manages the database transactions
- **@PersistenceContext EntityManager** dependency injection of the EntityManager
- **@Before** the test
 - 1 Clean database state
 - 2 Insert sample data
 - 3 Begin transaction
- **@After** the test commit the transaction

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Arquillian functional test

Arquillian - a different kind of testing

NH,CN

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")
        .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");
}
```

- **@Deployment(testable = false)** global deployment on client
- **@RunAsClient** run method on client
- **addAsWebResource** add html files needed for the test to the archive
- **merge** add a new generic shrinkwrapper which contains all html files to the archive

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Arquillian functional test

Arquillian - a different kind of testing

NH,CN

Functional test

```
@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')]"));
}
```

- **@Drone** dependency injection of the browser controller
- **@ArquillianResource** inject the URL
- **@FindBy** find dom object by id or css
- **@FindByjQuery** find dom object by jQuery expressions

Introduction

Advantages & Disadvantages

Advantages of testing with Arquillian

Disadvantages of testing with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Any open issues?

Arquillian - a
different kind of
testing

NH,CN

Not all understood?

- see <http://arquillian.org/guides/>

Questions?

Questions or remarks?

Introduction

Advantages &
Disadvantages

Advantages of testing with
Arquillian

Disadvantages of testing
with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?

Any open issues?

Arquillian - a
different kind of
testing

NH,CN

Death to all bugs!

Introduction

Advantages &
Disadvantages

Advantages of testing with
Arquillian

Disadvantages of testing
with Arquillian

Examples

Unit test

Shrinkwrapper

Complete test

Persistence test

Functional test

Any open issues?