arquillian

Maven - POM-Elements

```
<!-- The JBoss Community public repository is a composite repository of
      several major repositories -->
<repository>
      <id>jboss-public-repository</id>
      <name>JBoss Repository</name>
      <url>http://repository.jboss.org/nexus/content/groups/public</url>
</repository>

<dependencyManagement>
      <dependencies>
            <dependency>
                  <groupId>org.jboss.arquillian</groupId>
                  <artifactId>arquillian-bom</artifactId>
                  <version>1.0.0.CR5</version>
                  <scope>import</scope>
                  <type>pom</type>
            </dependency>
      </dependencies>
</dependencyManagement>

<!-- Needed for running tests (you may also use TestNG) -->
<dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
</dependency>

<!-- Optional, but highly recommended -->
<!-- Arquillian allows you to test enterprise code such as EJBs and
      Transactional(JTA) JPA from JUnit/TestNG -->
<dependency>
      <groupId>org.jboss.arquillian.junit</groupId>
      <artifactId>arquillian-junit-container</artifactId>
      <scope>test</scope>
</dependency>


<!-- An optional Arquillian testing profile that executes tests in a remote
      JBoss AS instance -->
<dependency>
      <groupId>org.jboss.as</groupId>
      <artifactId>jboss-as-arquillian-container-remote</artifactId>
      <version>7.0.2.Final</version>
      <scope>test</scope>
</dependency>
```

Description

Arquillian enables you to test your business logic in a remote or embedded container. Alternatively, it can deploy an archive to the container so the test can interact as a remote client.

The mission of the Arquillian project is to provide a simple test harness that abstracts away all container lifecycle and deployment from the test logic so developers can easily produce a broad range of integration tests for their enterprise Java applications.

Arquillian is part of the JBoss Testing initiative, an umbrella project focused on providing a comprehensive testing tool set for application developers.

Arquillian can either execute a test case inside the container, in which case the test class is deployed by Arquillian along with the code under test, or hold back the test class so it can act as a remote client to the deployed code. All the developer has to do is write the test logic.

Extensions

Drone

The Arquillian Drone extension for Arquillian provides a simple way of including functional tests for your web based application. Arquillian Drone manages the life cycle of web testing tool, which is either Arquillian Ajocado, Selenium or WebDriver. Arquillian Drone automatically manages life cycle of objects required for interaction between browser and deployed application.

Performance

The performance extension to Arquillian is a simple way of checking that the code you want to test performs within the range you want it to. It's can also automatically catch any performance regressions that might be added to your applications. - and as Arquillian itself, its very easy to use.

JSFUnit

JSFUnit is a test framework for JSF applications. It is designed to allow complete integration testing and unit testing of JSF applications using a simplified API. JSFUnit tests run inside the container, which provides the developer full access to managed beans, the FacesContext, EL Expressions, and the internal JSF component tree. At the same time, you also have access to parsed HTML output of each client request.

JaCoCo

JaCoCo is a free code coverage library for Java. It allows arquillian to analyse code coverage and reported with Sonar.

And more

Containers

> Glassfish 3.1 - remote
> Glassfish 3.1 - embedded

>JBoss 4.2 - remote
>JBoss 4.2 - managed
>JBoss 5 - remote
>JBoss 5.1 - remote
>JBoss 5.1 - managed
>JBoss 5.1 - embedded
>JBoss 6 - remote
>JBoss 6 - managed
>JBoss 6 - embedded
>JBoss 7 - remote
>JBoss 7 - managed
>JBoss 7- managed-domain
>JBoss 7- managed-clustering

>Weld SE 1 - embedded
>Weld SE 1.1 - embedded
>Weld EE 1.1 - embedded

>Tomcat 6 - managed
>Tomcat 6 - embedded
>Tomcat 7 - managed
>Tomcat 7 - embedded

>Jetty 6.1 - embedded
>Jetty 7 - embedded

> OSGi

> ...

arquillian

Example - CDI

Test-Objekt

```
public class Greeter {
    public String greet(String userName) {
        return "Hello, " + userName;
    }
}
```

Test

```
import javax.inject.Inject;

import org.jboss.arquillian.container.test.api.Deployment;
import org.jboss.arquillian.junit.Arquillian;
import org.jboss.shrinkwrap.api.ShrinkWrap;
import org.jboss.shrinkwrap.api.asset.EmptyAsset;
import org.jboss.shrinkwrap.api.spec.JavaArchive;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(Arquillian.class)
public class GreeterManagedBeanTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
                .addClass(Greeter.class)
                .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }

    @Inject
    Greeter greeter;

    @Test
    public void should_greet_earthlings() throws Exception {
        String name = "Earthlings";
        Assert.assertEquals("Hello, " + name, greeter.greet(name));
    }
}
```

Architecture

Arquillian combines a unit testing framework (JUnit or TestNG), ShrinkWrap, and one or more supported target containers (Java EE container, servlet container, Java SE CDI environment, etc) to provide a simple, flexible and pluggable integration testing environment.

At the core, Arquillian provides a custom test runner for JUnit and TestNG that turns control of the test execution lifecycle from the unit testing framework to Arquillian. From there, Arquillian can delegate to service providers to setup the environment to execute the tests inside or against the container. An Arquillian test case looks just like a regular JUnit or TestNG test case

Since Arquillian works by replacing the test runner, Arquillian tests can be executed using existing test IDE, Ant and Maven test plugins without any special configuration. Test results are reported just like you would expect. That's what we mean when we say using Arquillian is no more complicated than basic unit testing.

The test case is dispatched to the container's environment through coordination with ShrinkWrap, which is used to declaratively define a custom Java EE archive that encapsulates the test class and its dependent resources. Arquillian packages the ShrinkWrap-defined archive at runtime and deploys it to the target container. It then negotiates the execution of the test methods and captures the test results using remote communication with the server. Finally, Arquillian undeploys the test archive.

Links

Arquillian:	http://www.jboss.org/arquillian
Arquillian Doc:	https://docs.jboss.org/author/display/ARQ/Home
Quickstart:	http://www.jboss.org/jbossas/downloads
FAQ:	http://community.jboss.org/en/arquillian/faq
Showcase:	https://github.com/arquillian/arquillian-showcase
Repository:	https://repository.jboss.org/nexus/index.html#welcome