

Living Documentation

Table of Contents

1. Introduction	1
2. Summary	2
3. Features	3
3.1. Seeding database with DBUnit Rules CDI module	3
3.1.1. Scenario: Seed database using yml dataset	3

Chapter 1. Introduction

DBUnit Rules aims for bringing [DBUnit](#) closer to your JUnit tests. Here are the main features:

- [JUnit rule](#) to integrate with DBUnit via annotations:

```
@Rule
public DBUnitRule dbUnitRule = DBUnitRule.instance(jdbcConnection);①

@Test
@DataSet(value = "datasets/yml/users.yml")
public void shouldSeedDataSet(){
    //database is seed with users.yml dataset
}
```

① The rule depends on a JDBC connection.

- CDI interceptor to seed database without rule instantiation;
- Json, Yaml, xml and flat xml support;
- Cucumber integration;
- JPA integration;
- Multiple database support;
- Date/time support in datasets;

The project is composed by 5 modules:

- [Core](#): Contains the dataset executor and JUnit rule;
- [CDI](#): provides the DBUnit interceptor
- [Cucumber](#): a CDI aware cucumber runner;
- [JPA](#): Comes with a dataset executor based on JPA entity manager
- [Examples module](#).

Chapter 2. Summary

Scenarios			Steps							Features: 1	
Passed	Failed	Total	Passed	Failed	Skipped	Pending	Undefined	Missing	Total	Duration	Status
Seeding database with DBUnit Rules CDI module											
1	0	1	4	0	0	0	0	0	4	960ms	passed
Totals											
1	0	1	4	0	0	0	0	0	4	960ms	

Chapter 3. Features

3.1. Seeding database with DBUnit Rules CDI module

DBUnit CDI integration is done through a [CDI interceptor](#).

[IMPORTANT]

=====

CDI must be enabled in your test, see the following example:

[source, java]

```
@RunWith(CdiTestRunner.class) <1>
```

```
public class DBUnitCDITest {
```

```
}
```

<1> [CdiTestRunner](#) is provided by [Apache Deltaspikes](#) but you should be able to use other CDI test runners.

3.1.1. Scenario: Seed database using yml dataset

Given

DBUnit interceptor is enabled in your test beans.xml: 👍 (519ms)

src/test/resources/META-INF/beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">
  <interceptors>

  <class>com.github.dbunit.rules.cdi.DBUnitInterceptor</class>
  </interceptors>
</beans>
```



Your test itself must be a CDI bean to be intercepted. if you're using [Deltaspike test control](#) just enable the following property in `test/resources/META-INF/apache-deltaspike.properties`:

```
deltaspike.testcontrol.use_test_class_as_cdi_bean=true
```

And

The following dataset 🍌 (000ms)

src/test/resources/dataset/yml/users.yml

```
user:
  - id: 1
    name: "@realpestano"
  - id: 2
    name: "@dbunit"
tweet:
  - id: abcdef12345
    content: "dbunit rules!"
    user_id: 1
  - id: abcdef12233
    content: "dbunit rules!"
    user_id: 2
  - id: abcdef1343
    content: "CDI for the win!"
    user_id: 2
follower:
  - id: 1
    user_id: 1
    follower_id: 2
```

When

The following test is executed: 🍌 (000ms)

```
@Test
@UsingDataSet("yml/users.yml")
public void shouldSeedUserDataSetUsingCdiInterceptor() {
    List<User> users = em.createQuery("select u from User u order
by u.id asc").getResultList();
    User user1 = new User(1);
    User user2 = new User(2);
    Tweet tweetUser1 = new Tweet();
    tweetUser1.setId("abcdef12345");
    assertThat(users).isNotNull().hasSize(2).contains(user1,
user2);
    List<Tweet> tweetsUser1 = users.get(0).getTweets();

    assertThat(tweetsUser1).isNotNull().hasSize(1).contains(tweetUser1);
}
```

Then

The database should be seeded with the dataset content before test execution 🍷 (441ms)