

Living Documentation

Version 1.0.0-SNAPSHOT

Table of Contents

| | |
|---|-----------|
| 1. Introduction | 1 |
| 2. Summary | 2 |
| 3. Features | 3 |
| 3.1. Manage database with Database Rider Core | 3 |
| 3.1.1. Scenario: Seed database using yml dataset | 3 |
| 3.2. Manage database with Database Rider CDI | 4 |
| 3.2.1. Scenario: Seed database using yml dataset | 5 |
| 3.3. Manage database with Database Rider Cucumber | 7 |
| 3.3.1. Scenario: Seed database using Database Rider in Cucumber tests | 8 |
| 3.4. Manage database with Database Rider and JUnit 5 | 10 |
| 3.4.1. Scenario: Seed database using Database Rider in JUnit5 tests | 10 |
| 3.5. Dynamic data using scribable datasets | 11 |
| 3.5.1. Scenario: Seed database with groovy script in dataset | 12 |
| 3.5.2. Scenario: Seed database with javascript in dataset | 13 |
| 3.6. Database assertion using expected datasets | 14 |
| 3.6.1. Scenario: Database assertion with yml dataset | 14 |
| 3.6.2. Scenario: Database assertion with regular expression in expected dataset | 15 |
| 3.6.3. Scenario: Database assertion with seeding before test execution | 16 |
| 3.6.4. Scenario: Failling database assertion | 17 |
| 3.6.5. Scenario: Database assertion using automatic transaction | 18 |

Chapter 1. Introduction

Database Rider aims for bringing [DBUnit](#) closer to your JUnit tests so **database testing will feel like a breeze!**. 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 integration](#) via interceptor to seed database without rule instantiation;
- JSON, YAML, XML, XLS, and CSV support;
- [Configuration](#) via annotations or yml files;
- [Cucumber](#) integration;
- Multiple database support;
- Date/time support in datasets;
- Scriptable datasets with groovy and javascript;
- Regular expressions in expected datasets;
- [JUnit 5](#) integration;
- [DataSet export](#);
- [Connection leak detection](#);
- Lot of [examples](#).

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;
- [JUnit5](#): Comes with an [extension](#) for JUnit5.
- [Examples module](#).

Chapter 2. Summary

| Scenarios | | | Steps | | | | | | | Features: 6 | |
|---|--------|-------|--------|--------|---------|---------|-----------|---------|-------|-------------|--------|
| Passed | Failed | Total | Passed | Failed | Skipped | Pending | Undefined | Missing | Total | Duration | Status |
| Manage database with Database Rider Core | | | | | | | | | | | |
| 1 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 4 | 000ms | passed |
| Manage database with Database Rider CDI | | | | | | | | | | | |
| 1 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 4 | 01s 310ms | passed |
| Manage database with Database Rider Cucumber | | | | | | | | | | | |
| 1 | 0 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 000ms | passed |
| Manage database with Database Rider and JUnit 5 | | | | | | | | | | | |
| 1 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 000ms | passed |
| Dynamic data using scribable datasets | | | | | | | | | | | |
| 2 | 0 | 2 | 7 | 0 | 0 | 0 | 0 | 0 | 7 | 000ms | passed |
| Database assertion using expected datasets | | | | | | | | | | | |
| 5 | 0 | 5 | 16 | 0 | 0 | 0 | 0 | 0 | 16 | 001ms | passed |
| Totals | | | | | | | | | | | |
| 11 | 0 | 11 | 39 | 0 | 0 | 0 | 0 | 0 | 39 | 01s 313ms | |

Chapter 3. Features

3.1. Manage database with Database Rider Core

In order to manage database state in JUnit tests
As a developer
I want to use DBUnit in my tests.

Database Rider Core module brings [DBUnit](#) to your unit tests via [JUnit rules](#).

Dependencies

To use it just add the following maven dependency:

```
<dependency>
  <groupId>com.github.database-rider</groupId>
  <artifactId>rider-core</artifactId>
  <scope>test</scope>
</dependency>
```

3.1.1. Scenario: Seed database using yaml dataset

Given

The following junit rules 🍌 (000ms)

```
@RunWith(JUnit4.class)
public class DBUnitRulesIt {
  Unresolved directive in documentation.adoc -
  include:../../../src/test/java/com/github/database/rider/core/DBUnitRules
  It.java[tags=rules]
}
```

- ① [EntityManagerProvider](#) is a simple Junit rule that creates a JPA entityManager for each test. DBUnit rule don't depend on EntityManagerProvider, it only needs a **JDBC connection**.
- ② DBUnit rule responsible for reading [@DataSet](#) annotation and prepare the database for each test.

And

The following dataset 🍌 (000ms)

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

```
Unresolved directive in documentation.adoc -  
include:../../../src/test/resources/datasets/yml/users.yml[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../src/test/java/com/github/database/rider/core/DBUnitRules  
It.java[tags=seedDatabase]
```



Source code of the above example can be [found here](#).

Then

The database should be seeded with the dataset content before test execution 🍌 (000ms)

3.2. Manage database with Database Rider CDI

In order to manage database state in **CDI** based tests
As a developer
I want to use DBUnit in a CDI test environment.

DBUnit CDI integration is done through a [CDI interceptor](#) which reads `@DataSet` to prepare database for CDI based tests.

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



```
@RunWith(CdiTestRunner.class) ①  
@DBUnitInterceptor ②  
public class DBUnitCDITest {  
  
}
```

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

② Needed to activate DBUnit interceptor

Dependencies

To use this module just add the following maven dependency:

```
<dependency>  
  <groupId>com.github.database-rider</groupId>  
  <artifactId>rider-cdi</artifactId>  
  <scope>test</scope>  
</dependency>
```

3.2.1. Scenario: Seed database using yaml dataset

Given

DBUnit interceptor is enabled in your test beans.xml: 🍌 (01s 310ms)

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.database.rider.cdi.DBUnitInterceptorImpl</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

```
Unresolved directive in documentation.adoc -
include:../../../../cdi/src/test/resources/datasets/yml/users.yml[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../cdi/src/test/java/com/github/database/rider/cdi/DBUnit  
tCDIIt.java[tags=seedDatabase]
```



Source code of the above example can be [found here](#).

Then

The database should be seeded with the dataset content before test execution 🍌 (000ms)

3.3. Manage database with Database Rider Cucumber

In order to manage database state in **BDD** tests
As a BDD developer
I want to use DBUnit along side my BDD tests.

DBUnit enters the BDD world through a dedicated JUnit runner which is based on [Cucumber](#) and [Apache DeltaSpike](#).

This runner just starts CDI within your BDD tests so you just have to use [Database Rider CDI interceptor](#) on Cucumber steps, here is the so called Cucumber CDI runner declaration:

```
Unresolved directive in documentation.adoc -  
include:../../../../src/test/java/com/github/database/rider/bdd/DBUnitRulesBdd.java[]
```



As cucumber doesn't work with JUnit Rules, see [this issue](#), you won't be able to use Cucumber runner with *Database Rider Core* because its based on JUnit rules, but you can use DataSetExecutor in [@Before](#), see [example here](#).

Dependencies

Here is a set of maven dependencies needed by Database Rider Cucumber:



Most of the dependencies, except CDI container implementation, are bring by Database Rider Cucumber module transitively.

```
<dependency>
  <groupId>com.github.dbunit-rules</groupId>
  <artifactId>cucumber</artifactId>
  <scope>test</scope>
</dependency>
```

Cucumber dependencies

Unresolved directive in documentation.adoc -
include:../../../../cucumber/pom.xml[tags=cucumber-deps]

- ① You don't need to declare because it comes with Database Rider Cucumber module dependency.

DeltaSpike and CDI dependency

Unresolved directive in documentation.adoc -
include:../../../../cucumber/pom.xml[tags=deltaspike-cdi-deps]

- ① Also comes with DBUnit Rules Cucumber.
- ② You can use CDI implementation of your choice.

To use this module just add the following maven dependency:

3.3.1. Scenario: Seed database using Database Rider in Cucumber tests

Given

The following feature 🍌 (000ms)

Unresolved directive in documentation.adoc -
include:../../../../examples/jpa-productivity-boosters/src/test/resources/features/contacts.feature[]

And

The following dataset 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../examples/jpa-productivity-  
boosters/src/test/resources/datasets/contacts.yml[]
```

And

The following Cucumber test 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../examples/jpa-productivity-  
boosters/src/test/java/com/github/database/rider/examples/cucumber/Cont  
actFeature.java[]
```

When

The following cucumber steps are executed 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../examples/jpa-productivity-  
boosters/src/test/java/com/github/database/rider/examples/cucumber/Cont  
actSteps.java[]
```

- ① As the Cucumber cdi runner enables CDI, you can use injection into your Cucumber steps.
- ② Here we use the Database Rider CDI interceptor to seed the database before step execution.



Source code for the example above can be [found here](#).

Then

The database should be seeded with the dataset content before step execution 🍌 (000ms)

3.4. Manage database with Database Rider and JUnit 5

In order to manage database state in [JUnit 5](#) integration tests

As a developer

I want to use DBUnit along side my JUnit 5 tests.

DBUnit is enabled in JUnit 5 tests through an [extension](#) named **DBUnitExtension**.

Dependencies

To use the extension just add the following maven dependency:

```
<dependency>
  <groupId>com.github.dbunit-rules</groupId>
  <artifactId>junit5</artifactId>
  <scope>test</scope>
</dependency>
```

3.4.1. Scenario: Seed database using Database Rider in JUnit5 tests

Given

The following dataset 🍌 (000ms)

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

```
Unresolved directive in documentation.adoc -  
include:../../../../junit5/src/test/resources/datasets/users.yml[]
```

When

The following junit5 test is executed 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../junit5/src/test/java/com/github/database/rider/junit5/  
/DBUnitJUnit5It.java[tags=declaration;connectionField;test]
```

- ① Enables DBUnit;
- ② JUnit 5 runner;
- ③ As JUnit5 requires **Java8** you can use lambdas in your tests;
- ④ DBUnitExtension will get connection by reflection so just declare a field or a method with **ConnectionHolder** as return type.



Source code of the above example can be [found here](#).

Then

The database should be seeded with the dataset content before test execution 🍌 (000ms)

3.5. Dynamic data using scritable datasets

In order to have dynamic data in datasets
As a developer
I want to use scripts in DBUnit datasets.

Scritable datasets are backed by JSR 223. [2: Scripting for the Java Platform, for more information access the official [docs here](#)].

3.5.1. Scenario: Seed database with groovy script in dataset

Given

Groovy script engine is on test classpath 👍 (000ms)

```
<dependency>
  <groupId>org.codehaus.groovy</groupId>
  <artifactId>groovy-all</artifactId>
  <version>2.4.6</version>
  <scope>test</scope>
</dependency>
```

And

The following dataset 👍 (000ms)

```
Unresolved directive in documentation.adoc -
include:../../../../core/src/test/resources/datasets/yml/groovy-with-
date-replacements.yml[]
```

① Groovy scripting is enabled by **groovy**: string.

When

The following test is executed: 👍 (000ms)

```
Unresolved directive in documentation.adoc -
include:../../../../core/src/test/java/com/github/database/rider/ScriptRe
placementsIt.java[tags=groovy]
```

Then

Dataset script should be interpreted while seeding the database 👍 (000ms)

3.5.2. Scenario: Seed database with javascript in dataset



Javascript engine comes within JDK so no additional classpath dependency is necessary.

Given

The following dataset 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/js-with-calc-  
replacements.yml[]
```

① Javascript scripting is enabled by `js:` string.

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/java/com/github/database/rider/ScriptRe  
placementsIt.java[tags=javascript-likes]
```

Then

Dataset script should be interpreted while seeding the database 🍌 (000ms)

3.6. Database assertion using expected datasets

In order to verify database state after test execution
As a developer
I want to assert database state with datasets.

3.6.1. Scenario: Database assertion with yml dataset

Given

The following dataset 🍌 (000ms)

expectedUsers.yml

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/expectedUsers.yml[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/java/com/github/database/rider/Expected  
DataSetIt.java[tags=expectedDeclaration;expected]
```

① Clear database before to avoid conflict with other tests.

Then

Test must pass because database state is as in expected dataset. 🍌 (000ms)

3.6.2. Scenario: Database assertion with regular expression in expected dataset

Given

The following dataset 🍌 (000ms)

expectedUsersRegex.yml

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/expectedUsersReg  
ex.yml[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/java/com/github/database/rider/Expected  
DataSetIt.java[tags=expectedRegex]
```

Then

Test must pass because database state is as in expected dataset. 🍌 (000ms)

3.6.3. Scenario: Database assertion with seeding before test execution

Given

The following dataset 🍌 (000ms)

user.yml

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/user.yml[]
```

And

The following dataset 🍌 (000ms)

expectedUser.yml

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/expectedUser.yml  
[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/java/com/github/database/rider/Expected  
DataSetIt.java[tags=expectedWithSeeding]
```

Then

Test must pass because database state is as in expected dataset. 🍌 (000ms)

3.6.4. Scenario: Failing database assertion

Given

The following dataset 🍌 (000ms)

expectedUsers.yml

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/expectedUsers.yml[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/java/com/github/database/rider/Expected  
DataSetIt.java[tags=faillingExpected]
```

Then

Test must fail with following error: 🍌 (000ms)



```
junit.framework.ComparisonFailure: value (table=USER, row=0,  
col=name) expected:<[]expected user1> but was:<[non ]expected  
user1> at  
org.dbunit.assertion.JUnitFailureFactory.createFailure(JUnitFailur  
eFactory.java:39) at  
org.dbunit.assertion.DefaultFailureHandler.createFailure(Default  
FailureHandler.java:97) at  
org.dbunit.assertion.DefaultFailureHandler.handle(DefaultFailure  
Handler.java:223) at ...
```

3.6.5. Scenario: Database assertion using automatic transaction

Given

The following dataset 🍌 (000ms)

expectedUsersRegex.yml

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/resources/datasets/yml/expectedUsersReg  
ex.yml[]
```

When

The following test is executed: 🍌 (000ms)

```
Unresolved directive in documentation.adoc -  
include:../../../../core/src/test/java/com/github/database/rider/Transact  
ionIt.java[tags=transaction]
```



Transactional attribute will make Database Rider start a transaction before test and commit the transaction **after** test execution but **before** expected dataset comparison.

Then

Test must pass because inserted users are committed to database and database state matches expected dataset. 🍌 (000ms)