Chapter 1. What's New?

Table of Contents

Simultaneous engines

Sometimes applications will contain more than one *NoSQL* engine, for example some parts of your model will be expressed better as a graph (Neo4J for example), but other parts will be more natural in a column way (for example using Cassandra). **NoSQLUnit** supports this kind of scenarios by providing in integration tests a way to not load all datasets into one system, but choosing which datasets are stored in each backend.

For declaring more than one engine, you must give a name to each database *Rule* using `connectionIdentifier()` method in configuration instance.

Example 1.1. Given a name database rule

```
@Rule
public MongoDbRule remoteMongoDbRule1 = new MongoDbRule(mongoDb()
                                        .databaseName("test").connectionIdentifier
```

And also you need to provide an identified dataset for each engine, by using `withSelectiveLocations` attribute of `@UsingDataSet` annotation. You must set up the pair "named connection" / datasets.

Example 1.2. Selective dataset example

```
@UsingDataSet(withSelectiveLocations =
    { @Selective(identifier = "one", locations = "test3") },
   loadStrategy = LoadStrategyEnum.REFRESH)
```

In example we are refreshing database declared on previous example with data located at *test3* file.

Also works in expectations annotation:

Example 1.3. Selective expectation example

```
@ShouldMatchDataSet(withSelectiveMatcher =
    { @SelectiveMatcher(identifier = "one", location = "test3")
    })
```

For more information see chapter about advanced features.

Support for JSR-330

NoSQLUnit supports two annotations of JSR-330 aka Dependency Injection for Java. Concretely `@Inject` and `@Named` annotations.

During test execution you may need to access underlying class used to load and assert data to execute extra operations to backend. **NoSQLUnit** will inspect `@Inject` annotations of test fields, and try to set own driver to attribute. For example in case of MongoDb, `com.mongodb.Mongo` instance will be injected.

Example 1.4. Injection example

```
@Rule
public MongoDbRule remoteMongoDbRule1 = new MongoDbRule(mongoDb()
      .databaseName("test").build() ,this);

@Inject
private Mongo mongo;
```

Warning

Note that in example we are setting `this` as second parameter to the Rule.

But if you are using more than one engine at same time (see chapter) you need a way to distinguish each connection. For fixing this problem, you must use `@Named` annotation by putting the identifier given in configuration instance. For example:

Example 1.5. Named injection example

```
@Rule
public MongoDbRule remoteMongoDbRule1 = new MongoDbRule(mongoDb()
     .databaseName("test").connectionIdentifier("one").build() ,this);

@Rule
public MongoDbRule remoteMongoDbRule2 = new MongoDbRule(mongoDb()
     .databaseName("test2").connectionIdentifier("two").build() ,this);

@Named("one")
@Inject
private Mongo mongo1;

@Named("two")
@Inject
private Mongo mongo2;
```

For more information see advanced features chapter.