**DBUnit**

* Pentru a crea teste ce testeaza cum functioneaza codul cu baza de date, pentru a evita sa lucram direct du baza de date, proces care ar fi costisitor, cream un set de date cu DBUnit
* DBUnit este o extenstie la Junit ce ne ajuta sa testam aplicatii ce depind de baze de date
* Baza de date creata de Junit e creata pentru fiecare test
* Dbunit ofera un mecanism simplu XML pentru a crea date
* DBUnit ne ajuta sa cream seturi de date de la fisiere XML
* **@ContextConfiguration** este o anotatie folosita in loc de @SpringBootTest, in care putem specifica noi anume ce contexts vrem sa fie incarcate.

Are parametrii:

classes – specificam clasele ce sunt @Configuration

locations – fisierele xml

* Partea buna la DBUnit e ca nu ne trebuie dubluri sau Mock caci lucram ca si cum chiar cu baza de date
* Dependente:

<dependency>  
 <groupId>org.dbunit</groupId>  
 <artifactId>dbunit</artifactId>  
 <version>2.7.3</version>  
 <scope>test</scope>  
</dependency>

* Acum, avem nevoie de un fisier .xml pentru a scrie datele nostre in el. Fiecare tuple va fi scris asa:

<NumeTabel column1=”” column2=”” ....>

<dataset>  
 <Teacher id="100" first\_name="Test1" last\_name="User1" expirience="2"/>  
 <Teacher id="101" first\_name="Test2" last\_name="User2" expirience="20"/>  
 <Teacher id="102" first\_name="Test3" last\_name="User3" expirience="10"/>  
 <Teacher id="103" first\_name="Test4" last\_name="User4" expirience="5"/>  
 <Teacher id="104" first\_name="Test2" last\_name="User5" expirience="8"/>  
</dataset>

Nu folosim fields din clasa Teacher,ci asa cum sunt ele in baza de date

* Pentru a ne folosi de DBUnit, clasa de test trebuie sa extinda clasa **DBTestCase**
* **DBUnit preia datele legate de baza de date din System, de aia cel mai bine e ca in constructor sa setam datele de conectare la baza de date reala:**
* public DBunitApplicationTests()  
  {  
   System.*setProperty*( PropertiesBasedJdbcDatabaseTester.*DBUNIT\_DRIVER\_CLASS*, com.mysql.jdbc.Driver.class.getName() );  
   System.*setProperty*( PropertiesBasedJdbcDatabaseTester.*DBUNIT\_CONNECTION\_URL*, "jdbc:mysql://localhost:3306/test" );  
   System.*setProperty*( PropertiesBasedJdbcDatabaseTester.*DBUNIT\_USERNAME*, "testuser" );  
   System.*setProperty*( PropertiesBasedJdbcDatabaseTester.*DBUNIT\_PASSWORD*, "Frb2eshox!" );  
   // System.setProperty( PropertiesBasedJdbcDatabaseTester.DBUNIT\_SCHEMA, "" );  
  }
* Acum, este necesar de a suprascrie metoda getDataSet pentru a obtine datele din fisierul .xml
* protected IDataSet getDataSet() throws Exception  
  {  
   return new FlatXmlDataSetBuilder().build(new FileInputStream("src/test/resources/data.xml"));  
  }
* In mod default, DBUnit face CLEAN\_INSERT inaintea fiecarui test, adica sterge datele puse de testul precedent si le pune pe cele din xml inapoi, dar nu face nimic la finisarea testului. Metodele care fac asta sunt:
* protected DatabaseOperation getSetUpOperation() throws Exception  
  {  
   return DatabaseOperation.*REFRESH*;  
  }  
    
  protected DatabaseOperation getTearDownOperation() throws Exception  
  {  
   return DatabaseOperation.*NONE*;  
  }

Le putem da override si modifica comportamentul.