疯狂的菠菜

永久域名 http://macrochen.iteye.com

macrochen

unitils中spring module初始化源码解读 | 淘宝数据平台招聘

2009-12-02

解决unitils中的dbunit使用spring中定义的多数据源

博客分类: 测试

Spring SQL 单元测试 Bean Oracle

最近在研究unitils, dbunit来适应目前的单元测试.

在unitils中将要使用的数据源都定义在unitils.properties中, 而在我们的测试配置中, 同样的数据源也在spring中配置了一份儿, 因此本人希望同样的配置不应该出现在两个方面, 从而增加维护成本. 另一方面还可以通过spring来解决多数据源的问题. 于是开始看unitils的源代码, 原来在dbunit module中, 获取数据源的代码:

Java代码

```
1.  public DbUnitDatabaseConnection getDbUnitDatabaseConnection(String schemaName) {
2.          DbUnitDatabaseConnection dbUnitDatabaseConnection = dbUnitDatabaseConnections.get(schema
    Name);
3.          if (dbUnitDatabaseConnection == null) {
4.              dbUnitDatabaseConnection = createDbUnitConnection(schemaName);
5.              dbUnitDatabaseConnections.put(schemaName, dbUnitDatabaseConnection);
6.          }
7.          return dbUnitDatabaseConnection;
8.      }
9.      protected DbUnitDatabaseConnection createDbUnitConnection(String schemaName) {
10.         // A DbSupport instance is fetched in order to get the schema name in correct case
11.         DataSource dataSource = getDatabaseModule().getDataSourceAndActivateTransactionIfNeeded(
    );
12.         ...
13.     }
```

也就是说, 最终的数据源是通过DatabaseModule来获取的, 因此我们需要对这些代码进行改造, 但是还有另外一个问题, unitils的spring module对所有测试的spring context都是按照以测试类为key, context为value进行缓存的. 因此要获取spring的context, 必须知道当前的测试类, 因此这里需要整体调整Dbunit module的insertDataset()方法的内部方法调用的参数. 在处理流程中增加testObject参数. 这个需要修改的代码非常多, 这里不一一展开, 对于从spring中获取数据源的代码如下:

Java代码

```
1.  private DataSource getDataSource(String schemaName, Object testObject) {
2.      // 从spring中取
3.      SpringModule springModule = Unitils.getInstance().getModulesRepository().getModuleOfType(Spr
    ingModule.class);
4.      DataSource dataSource = (DataSource) springModule.getSpringBean(testObject, schemaName);
5.      if (dataSource == null) {
6.          throw new RuntimeException(String.format("datasource[%s]在spring配置文件中不存在", schemaN
    ame));
7.      }
8.      return dataSource;
9.  }
```

为了将spring的datasource与dbunit能够关联起来, 本人对构造的测试数据做了一个约定, 对于操作的数据表, 必须加

- 电影 (3)
- Ajax (14)
- 代码 (7)
- Eclipse (95)
- vi (2)
- 音乐 (6)
- groovy (10)
- AutoHotKey (3)
- Dorado (10)
- maven (6)
- scrum (5)
- 英语 (20)
- 理财 (12)
- OSGi (3)
- 综合技术 (3)
- 工具 (6)
- browser (1)
- ppt (1)

我的相册

杭州绿茶

共 777 张

我的留言簿 >>更多留言

- 你好，能不能给我发一些在eclipse中开发dorado项目的小例子，最好有数据库 ...

  -- by pjkl
- http://rdc.taobao.com/team/jm/ar ...

  -- by ordinary
- 你好，刚被录取为淘宝开发实习生，想了解一下现在淘宝的技术类应届研究生的一般待遇，谢 ...

  -- by livingforever

其他分类

- 我的收藏 (82)
- 我的代码 (0)
- 我的书籍 (2)
- 我的论坛主题帖 (50)

数据源标识前缀, 而数据源标识则对应spring bean配置文件中的id, 比如配置了一个datasource bean:

Xml代码

```xml
<bean id="db1" parent="parentDataSource">
        <property name="url">
                <value>jdbc:oracle:oci:@${db1.name}</value>
        </property>
        <property name="username">
                <value>${db1.username}</value>
        </property>
        <property name="password">
                <value>${db1.password}</value>
        </property>
</bean>
```

如果有一个dataset中有一个table(table1)跟该datasource关系, 那么该表名必须这样定义: db1.table1. 也可以将db1前缀理解为schema(但这里实际并不是).对于这个问题, 已经跟unitils的founder进行了交流, 并在jira中增加了一个issue: http://jira.unitils.org/browse/UNI-190 在未来版本中将实现该功能.

在对数据清理的处理上的改进

对于构造的测试准备数据的清理, dbunit在默认情况下, 假定所有测试的数据库表必须建立主键, 否则会抛出异常, 因为在清理数据的时候需要利用主键以及构造的对应主键值来做db的delete操作.而我们目前存在一些关联表是没有主键的, 因此给使用unitils带来了一定的麻烦. 但也是可以搞定的. 另外一个问题就是, 对于所有的构造数据都会根据dataset中定义的table来处理, 而目前我们面临的另外一个问题, 就是在测试前, 希望能通过指定的sql来对测试环境对某些数据进行清理操作, 以避免对测试造成影响. 于是

对unitils的DataSetLoadStrategy和dbunit的DatabaseOperation进行了扩展.

本人定义了一个ExecuteSqlOperation用来执行指定的sql语句:

Java代码

```java
public class ExecuteSqlOperation extends DatabaseOperation {
    private Map<String, List<String>> sqlMap;
    private static final Logger logger = LoggerFactory.getLogger(ExecuteSqlOperation.class);

    public ExecuteSqlOperation(Map<String, List<String>> sqlMap) {
        this.sqlMap = sqlMap;
    }

    @Override
    public void execute(IDatabaseConnection connection, IDataSet dataSet) throws DatabaseUnitException, SQLException {
        logger.debug("execute(connection={}, dataSet={}) - start", connection, dataSet);

        DatabaseConfig databaseConfig = connection.getConfig();
        IStatementFactory factory =
                (IStatementFactory) databaseConfig.getProperty(DatabaseConfig.PROPERTY_STATEMENT_FACTORY);

        // for each table
        Iterator<Entry<String, List<String>>> iterator = sqlMap.entrySet().iterator();
        while (iterator.hasNext()) {
            Entry<String, List<String>> sqlEntry = iterator.next();
            String schema = sqlEntry.getKey();

            // Do not process empty table
            if (StringUtils.isBlank(schema) || sqlEntry.getValue().isEmpty()) {
                logger.warn(String.format("schema{%s} or sqls(%s) is empty", schema, sqlEntry.getValue()));
                continue;
            }
```

- 我的所有论坛帖 (210)
- 我的精华良好帖 (5)

最近加入群组

- Groovy on Grails
- 读书空间
- maven
- DocBook
- 英语学习

存档

- 2011-08 (2)
- 2011-07 (4)
- 2011-06 (3)
- 更多存档...

评论排行榜

- [joke] 史上最强的结对编程^_^
- 用groovy测试java代码的两个障碍及其他
- 《硝烟中的scrum和xp》读书笔记
- 《程序员的思维修炼》读书笔记
- 猎头Fiona总结的加入淘宝网的十个理由<转 ...


```
28.
29.             // don't find the datasource
30.             if (!connection.getSchema().equalsIgnoreCase(schema)) {
31.                 continue;
32.             }
33.
34.             IPreparedBatchStatement statement = null;
35.
36.             try {
37.                 for (String sql : sqlEntry.getValue()) {
38.                     statement = factory.createPreparedBatchStatement(sql, connection);
39.                     statement.addBatch();
40.                 }
41.
42.                 statement.executeBatch();
43.                 statement.clearBatch();
44.
45.                 // clear schema and sql
46.                 iterator.remove();
47.             } finally {
48.                 if (statement != null) {
49.                     statement.close();
50.                 }
51.             }
52.         }
53.     }
54.
55.     public void setSqlMap(Map<String, List<String>> sqlMap) {
56.         this.sqlMap = sqlMap;
57.     }
58. }
```

从代码中我们可以看出, dbunit中的DatabaseOperation 与dataset是有非常强的耦合的(本来嘛, dataset是整个dbunit的模型核心), 而这里我们对dataset是不需要的: 拿到connection, 执行sql, 退出.

然后定义了一个CustemAndInsertDataSetLoadStrategy, 用来引入我们上面定义的这个ExecuteSqlOperation :

Java代码

```
1.  public abstract class CustemAndInsertDataSetLoadStrategy implements DataSetLoadStrategy {
2.      public void execute(DbUnitDatabaseConnection dbUnitDatabaseConnection, IDataSet dataSet) {
3.          try {
4.              doExecute(dbUnitDatabaseConnection, dataSet);
5.              DatabaseOperation.INSERT.execute(dbUnitDatabaseConnection, dataSet);
6.          } catch (DatabaseUnitException e) {
7.              throw new UnitilsException("Error while executing DataSetLoadStrategy", e);
8.          } catch (SQLException e) {
9.              throw new UnitilsException("Error while executing DataSetLoadStrategy", e);
10.         }
11.     }
12.
13.     abstract protected void doExecute(DbUnitDatabaseConnection dbUnitDatabaseConnection, IDataSe
    t dataSet)
14.             throws DatabaseUnitException, SQLException;
15. }
```

在实际测试中使用:

Java代码

```
1.  @DataSet(loadStrategy = MyTest.MyDataSetLoadStrategy.class)
2.  public class MyTest extends IcBaseCase2 {
```

```
3.      public static class MyDataSetLoadStrategy extends CustemAndInsertDataSetLoadStrategy {
4.          private Map<String, List<String>> sqlMap = new HashMap<String, List<String>>();
5.          private ExecuteSqlOperation executeSqlOperation;
6.          public SpuRelationDataSetLoadStrategy() {
7.              sqlMap.put("db1", Collections
8.                      .singletonList("delete table1 where id = 110"));
9.              executeSqlOperation = new ExecuteSqlOperation(sqlMap);
10.         }
11.
12.         @Override
13.         protected void doExecute(DbUnitDatabaseConnection dbUnitDatabaseConnection, IDataSet dat
    aSet)
14.                 throws DatabaseUnitException, SQLException {
15.             executeSqlOperation.execute(dbUnitDatabaseConnection, dataSet);
16.         }
17.     }
```

当然中间省略了一些实现细节, 发现经过这样扩展之后, 基本能满足我们的测试需要.

感觉多数据源的使用是一个比较少的场景, 从dbunit, unitils的feature中就可以看出来, 貌似二者基本上没有做实现, 因此有了上面的代码.

unitils中spring module初始化源码解读 | 淘宝数据平台招聘

20:07 | 评论 / 浏览 (0 / 580) | 相关推荐 MORE

评论

发表评论

您还没有登录,请您登录后再发表评论