运用DBUnit进行高效的单元测试 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/273/2021_2022__E8_BF_90_E7_94_A8DBUn_c104_273297.htm 现实系统中通常会有一些具有外部依赖性的对象，这些对象和数据库或者其他对象存在诸多关联。如果我们对这样的对象编写单元和组件级测试的话，可以想象将是非常麻烦的一件事.因为这种外部依赖性的存在，使的我们很难将对象孤立出来进行测试。 经常提及的白盒测试法，基本上就是通过控制对象的外部依赖性来达到隔离对象的目的，使的可以操作这些对象的状态和相关行为。 运用 模拟对象(mock objects) 或者stubs，就是一个控制对象外部依赖性的解决方案。通过隔离那些关联的数据库访问类，象JDBC的相关操作类，对于控制对象外部依赖性将是很有效的。但模拟对象的解决方案对一些特殊的应用系统架构就显得力不从心了，象那些运用了EJB的CMP(container-managed persistence)或者 JDO(java Data Objects)的应用系统架构，在这些架构里，数据库的访问对象是在最底层的而且很隐蔽。 由Manuel Laflamme 编写的开放源代码的DBUnit架构体系，对于控制系统内部的数据库依赖性提供了一个非常不错的解决方案。他允许程序员在整个的测试过程中自由的管理控制数据库的状态，这很重要。利用DBUnit，在测试之前，我们可以给目标数据库植入我们需要的数据集，而且，在测试完毕后，数据库完全能够回溯到测试前的状态。 在很多成功的软件项目中，测试自动化往往是关键的层面。DBUnit允许开发人员创建测试用例代码，在这些测试用例的生命周期内我们可以很好的控制数据库的状态。而且,这些测试用例是很容易

实现自动化的。这样在测试过程中我们无须对它进行人工的干预，为人工干预造成的后果而担心就更没必要了。 简单介绍 配置使用DBUnit的第一步我们首先需要知道如何生成数据库schema，这个文件是XML格式的，其中包括了数据库的表及相关数据信息。 例如，这里有一个数据库表employee ，我们可以用SQL的形式这样将他表示出来。 而且，我们可以看到，一个简单的数据集可以这样表示 在DBUnit中，上面这个表和抽样数据信息可以用XML文件的形式这样表示： start_date=2001-11-01 first_name=Andrew ssn=xxx-xx-xxxx last_name=Glover /> 这个生成的XML格式的文件可以作为系统所需的所有种子文件(seed files)的样本模版. 为相互关联的测试场景创建多个种子文件是一个很有效的策略，就象通过不同的数据库文件来区分隔离数据库状态是一个道理。多种子文件策略可以将我们的测试目标锁定到较小的范围，目标数据可以只针对数据库的表，而不是整个数据库。 为了给目标数据库植入不同的职员记录，我们需要的XML数据文件如下所示： start_date=2001-01-01 first_name=Drew ssn=000-29-2030 last_name=Smith /> start_date=2002-04-04 first_name=Nick ssn=000-90-0000 last_name=Marquiss /> start_date=2003-06-03 first_name=Jose ssn=000-67-0000 last_name=Whitson /> 现在，要让DBUnit和我们所需的数据库schema一起工作了，对于程序员来说，我们使用DBUnit进行测试可以有两种选择：通过直接编码方式进行测试或者与Ant结合. 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com