# EJB单元测试及其总结

## 综述

本文从安装步骤，注意事项和参考文献几个方面说明EJB单元测试相关内容。

安装步骤主要描述了Netbeans开发环境中如何创建单个EJB的单元测试，如何配置maven的pom.xml文件，以及如何运行单元测试等问题。

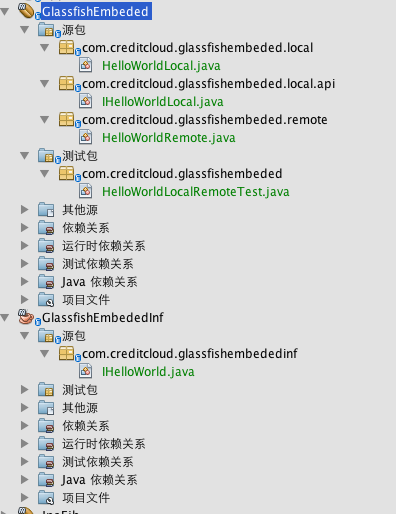
注意事项列出了作者在测试中遇到的一些问题，提供一些参考。

参考文献列出了探究EJB单元测试所使用的一些网络资源。

## 安装步骤

### Session Bean例子

Session Bean的例子总体的结构如图所示。



我们只要建立了两个工程，GlassfishEmbededInf和GlassfishEmbeded， 前者是远程接口，后者是实现了远程接口的会话Bean。

* 首先建立一个EJB远程接口的工程GlassfishEmbededInf，里面包含一个IHelloWorld接口。

public interface IHelloWorld {

public String say();

}

* 建立一个实现此远程接口的EJB工程，同时也包含了一个本地Session Bean接口和本地Bean实现。

@Stateless

@Remote(IHelloWorld.class)

public class HelloWorldRemote implements IHelloWorld {

@Override

public String say() {

return "HelloWorld";

}

}

一个本地接口代码：

public interface IHelloWorldLocal {

public String sayLocal();

}

一个本地bean的具体实现：

@Stateless

@Local(IHelloWorldLocal.class)

public class HelloWorldLocal implements IHelloWorldLocal {

@Override

public String sayLocal() {

return "HelloWorldLocal";

}

}

此工程的pom.xml文件的设置如下文所示：

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.creditcloud</groupId>

<artifactId>GlassfishEmbeded</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>ejb</packaging>

<name>GlassfishEmbeded</name>

<parent>

<groupId>com.creditcloud</groupId>

<artifactId>Parent</artifactId>

<version>1.0.0</version>

</parent>

<dependencies>

<dependency>

<groupId>org.glassfish.main.extras</groupId>

<artifactId>glassfish-embedded-all</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.glassfish.main.extras</groupId>

<artifactId>glassfish-embedded-static-shell</artifactId>

<version>4.0</version>

</dependency>

<!-- 此处是依赖的远程接口 -->

<dependency>

<groupId>com.creditcloud</groupId>

<artifactId>GlassfishEmbededInf</artifactId>

<version>1.0-SNAPSHOT</version>

<type>jar</type>

</dependency>

</dependencies>

<！--关于Build的一些配置--->

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-ejb-plugin</artifactId>

<version>2.3</version>

<configuration>

<ejbVersion>3.1</ejbVersion>

</configuration>

</plugin>

</plugins>

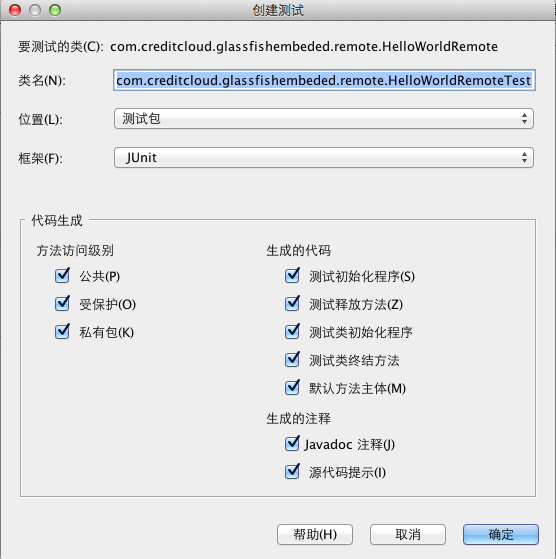
</build>

</project>

在pom.xml中，依赖关系中的glassfish-embedded-all和glassfish-embedded-static-shell是两个核心的依赖。 glassfish-embedded-all就是glassfish的嵌入式容器，大小近80M，下载比较慢。所有运行EJB单元测试的工程pom.xml文件中要包含这两个依赖。本地安装的glassfish包含glassfish-embedded-static-shell这jar包，位置在${install\_location}/lib/embeded下面。

* 编写代码的测试用例。

Netbeans支持根据实现类创建自动的测试。点击工具->创建测试，会弹出创建测试类的窗口，如下所示，会在测试包下为HelloWorldRemote创建HelloWorldRemoteTest类，此代码已经包含了一些核心的代码。为了测试方便，我们为一个工程的所有的Bean只建立一个测试类，在这个测试类中初始化和关闭容器，并测试所有Bean的所有业务方法，因为这样，容器只打开和关闭一次，提升测试效率，同时，避免出现一个测试用例结束还没有关闭容器的情况下，另一个测试用例打开容器，出现“Address already in use”的错误。



所以，我们将此远程回话Bean和本地会话Bean的测试放在一个Java文件(HelloWorldLocalRemoteTest)中，代码如下所示：

@RunWith(JUnit4.class)

public class HelloWorldLocalRemoteTest {

static EJBContainer container;

static Context context;

public HelloWorldLocalRemoteTest() {}

@BeforeClass

public static void setUpClass() {

Map properties = new HashMap();

// 指定的编译的位置

properties.put(EJBContainer.MODULES, new File[]{new File("target/classes"), new File("target/test-classes")});

// 服务器的安装的位置

properties.put("org.glassfish.ejb.embedded.glassfish.installation.root", "/Users/sobranie/majinming/glassfish4/glassfish");

//指定服务器的配置文件的位置

properties.put("org.glassfish.ejb.embedded.glassfish.configuration.file", "/Users/sobranie/majinming/commomfile/domain.xml");

container = EJBContainer.createEJBContainer();

context = container.getContext();

}

@AfterClass

public static void tearDownClass() throws Exception {

container.close();

}

@Test //运行正式的测试内容

public void testSayLocal() throws Exception {

IHelloWorldLocal helloWorld = (IHelloWorldLocal) context.lookup("java:global/classes/HelloWorldLocal");

Assert.assertEquals("HelloWorldLocal", helloWorld.sayLocal());

}

@Test

public void testSayRemote() throws NamingException {

IHelloWorld helloWorld = (IHelloWorld) context.lookup("java:global/classes/HelloWorldRemote");

Assert.assertEquals("HelloWorld", helloWorld.say());

}

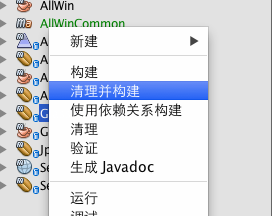
}

这个测试文件中最重要的是@Beforeclass和@AfterClass， 拥有@BeforeClass标签的方法只会在这个测试开始执行所有的方法前面执行一次，所以，我们将容器的创建和初始化等过程放在这儿，拥有@AfterClass标签的方法会在所有的test方法执行结束之后，执行一次，所以，我们将关闭容器的过程放在这儿。

在setUpClass方法中，我们首先配置一系列容器的定制化属性，主要有三个。第一个，容器的安装根目录，由org.glassfish.ejb.embedded.glassfish.installation.root指定，如果本地安装有glassfish的话，可以指定本地的glassfish安装目录。否则，容器在启动的时候可能会报错，也有可能讲应用程序部署到系统临时目录中去。第二个， 代码字节码的位置。由EJBContainer.MODULES，否则容器可能找不到需要部署的Bean的字节码的位置。 第三，domain.xml文件。 domain.xml文件中，可以配置应用程序的数据库连接等信息。这个在测试Entity Bean的时候很有用。

对于Bean的具体的测试，我们使用JNDI将Bean取出来，直接测试其方法即可。对于每个方法，需要标注@Test， 如果想忽略这个测试，使用@Ignore标签。

* 测试用例编写完整之后，只需要清理构建即可。选中单个EJB工程，选择清理构建即可。

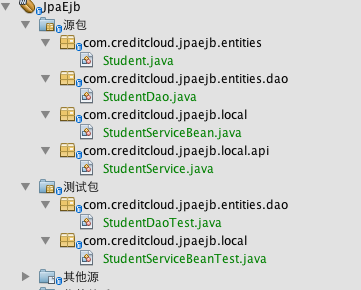


此时，如果测试用例通过，则会Build Success.

### Entity Bean 例子

Entity Bean测试的例子与Session Bean基本相同。不同的是，需要配置数据源。在pom.xml 文件中，需要加入数据库驱动程序依赖（本例使用msql）。在这个例子中，连接mysql数据库，并测试DAO。

系统总体的工程结构图如图所示。



最终需要测试StudentServiceBean和StudentDao.

* 首先，编写Student, StudentDao, StudengServiceBean, StudentService代码。

Student代码如下所示。

@Entity

@Table(name = "student")

public class Student implements Serializable {

@Id

@Column(nullable = false)

private int id;

@Column

private String name;

@Column

private String school;

public void setId(int id) {

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public void setSchool(String school) {

this.school = school;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public String getSchool() {

return school;

}

}

StudentDao的代码如下所示：

@Stateless

@LocalBean

public class StudentDao {

@PersistenceContext(unitName = "Student")

private EntityManager em;

public StudentDao() {}

public Student queryUserById(int id) {

Query query = em.createQuery("select s from Student s where s.id=:id");

query.setParameter("id", id);

Student student = (Student) query.getSingleResult();

return student;

}

}

StudentService 代码如下所示：

public interface StudentService {

public String getSchool(int id);

}

StudentServiceBean的代码如下所示：

@Singleton

@Stateless

@Local(StudentService.class)

public class StudentServiceBean implements StudentService {

@EJB //注入StudentDao

StudentDao studentDao;

@Override

public String getSchool(int id) {

Student student = studentDao.queryUserById(id);

if (student != null) {

return student.getSchool();

}

return null;

}

}

* 编写测试代码，StudentDaoTest代码如下所示。

StudentDaoTest代码如下所示。

@RunWith(JUnit4.class)

public class StudentDaoTest {

static EJBContainer container;

static Context context;

public StudentDaoTest() {}

@BeforeClass

public static void setUpClass() {

Map properties = new HashMap();

properties.put(EJBContainer.MODULES, new File[]{new File("target/classes"), new File("target/test-classes")});

//在数据中，需要给系统指定新的安装的位置。

properties.put("org.glassfish.ejb.embedded.glassfish.installation.root", "/Users/sobranie/majinming/glassfish4/glassfish");

//配置文件的位置

properties.put("org.glassfish.ejb.embedded.glassfish.configuration.file", "/Users/sobranie/majinming/commomfile/domain.xml");

container = EJBContainer.createEJBContainer(properties);

context = container.getContext();

}

@AfterClass

public static void tearDownClass() throws Exception {

container.close();

}

@Test

public void testQueryUserById() throws NamingException {

StudentDao dao = (StudentDao) context.lookup("java:global/classes/StudentDao");

Assert.assertEquals("bupt", dao.queryUserById(1).getSchool());

}

@Test

public void testGetSchool() throws NamingException {

StudentService bean = (StudentService) context.lookup("java:global/classes/StudentServiceBean");

Assert.assertEquals("bupt", school);

}

}

* 配置pom.xml和persistence.xml文件的内容。

pom.xml的内容如下所示，基本相同，加入了mysql驱动包和eclipselink包。

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.creditcloud</groupId>

<artifactId>JpaEjb</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>ejb</packaging>

<name>JpaEjb</name>

<parent>

<groupId>com.creditcloud</groupId>

<artifactId>Parent</artifactId>

<version>1.0.0</version>

</parent>

<dependencies>

<!-- Persist -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

<dependency>

<artifactId>eclipselink</artifactId>

<groupId>org.eclipse.persistence</groupId>

<type>jar</type>

<version>2.5.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.glassfish.main.extras</groupId>

<artifactId>glassfish-embedded-all</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.glassfish.main.extras</groupId>

<artifactId>glassfish-embedded-static-shell</artifactId>

<version>4.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-ejb-plugin</artifactId>

<version>2.3</version>

<configuration>

<ejbVersion>3.1</ejbVersion>

</configuration>

</plugin>

</plugins>

</build>

</project>

persisten.xml中只需要指定使用的数据源即可。代码如下所示。

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">

<persistence-unit name="Student" transaction-type="JTA">

<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

<!-- 此处需要修改数据源，和domain.xml中的数据源相互对应即可-->

<jta-data-source>jdbc/\_\_default</jta-data-source>

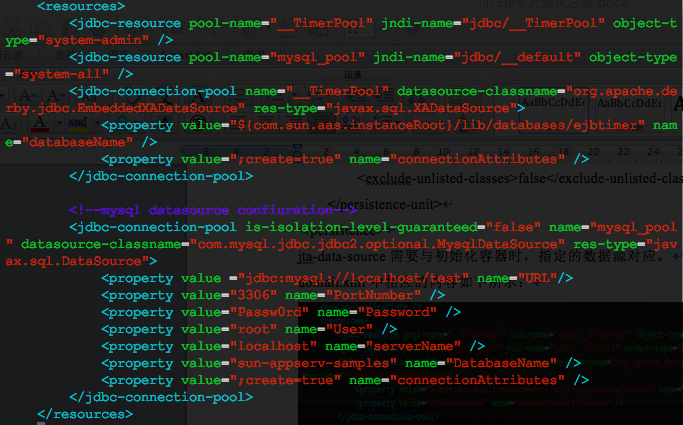
<exclude-unlisted-classes>false</exclude-unlisted-classes>

</persistence-unit>

</persistence>

jta-data-source需要与初始化容器时，指定的数据源对应。

domail.xml中相应的内容如下所示：



此处，指定了连接mysql\_pool（在jdbc\_connection-pool中指定）的各项配置。

* 最后，清理构建项目即可。

## 注意事项

* pom.xml的配置是最关键的，必须引入glassfish-embedded-all和glassfish-embedded-static-shell两个依赖。
* 编写测试用例的时候，需要添加属性EJBContainer.MODULES，org.glassfish.ejb.embedded.glassfish.installation.root 和org.glassfish.ejb.embedded.glassfish.configuration.file 属性。
* 为了提高测试效率，一个工程中的所有Bean(Session Bean, Entity Bean)都写在一个文件中，完成测试。
* 整个项目在github上的地址为https://github.com/iqequal0/ejb\_unit\_test

## 常见错误

* Container initialize error. 远程接口和实现不能再同一个工程里。

## 参考文献

### EJB单元测试参考文献

表1. 重要的几篇参考文献

|  |  |
| --- | --- |
| URL | 文献内容说明 |
| https://netbeans.org/kb/docs/javaee/javaee-entapp-junit\_zh\_CN.html?print=yes | 使用嵌入式glassfish测试EJB的官网教程，有详细的流程和说明 |
| https://netbeans.org/kb/docs/javaee/maven-entapp-testing\_zh\_CN.html?print=yes | 使用maven测试EJB的教程。里面有实体bean的测试说明。 |
| http://docs.oracle.com/cd/E18930\_01/html/821-2424/gjlde.html | Oracle上使用嵌入式glassfish测试EJB的一篇文章 |
| http://zhengxl.me/blog/2013/09/10/ejb3-unit-test-using-glassfish-embedded-ejb-container | 国人写的一篇使用Glassfish嵌入式容器和Maven测试单个EBJ的文章，很有用。 |
| http://www.verborgh.be/articles/2010/08/05/using-the-glassfish-ejbcontainer-in-netbeans/ | 如何给嵌入式容器指定domain.xml文件的文章，domian.xml中可以配置数据库连接资源，这个在测试entity Bean时很重要。 |
| http://www.hascode.com/2011/01/enterprise-java-bean-ejb-3-1-testing-using-maven-and-embedded-glassfish/ | EJB单元测试另外一篇重要的文章 |
| http://hi.baidu.com/rltmhffswlfgiwr/item/cca3b4dd56b4349e260ae713 | 百度上的一篇EJB单元测试的文章。 |
|  |  |

表2. 不太重要的几篇参考文献

|  |  |
| --- | --- |
| URL | 文献内容说明 |
| http://www.junitee.org/tutorial.html | JunitEE测试框架的一篇介绍文档 |
| http://www.ibm.com/developerworks/rational/library/07/0102\_woldemichael/ | IBM使用JUnitEE框架进行EJB元测试的一篇文章 |
| https://netbeans.org/kb/docs/javaee/maven-entapp-testing\_zh\_CN.html?print=yes | 使用maven测试EJB的教程。里面有实体bean的测试说明。 |
| http://docs.oracle.com/cd/E18930\_01/html/821-2424/gjlde.html | Oracle上使用嵌入式glassfish测试EJB的一篇文章 |
| http://www.oracle.com/technetwork/cn/articles/java/unittesting-455385-zhs.html | 模拟测试EJB的一个参考文献 |
| http://www.verborgh.be/articles/2010/01/29/configuring-a-datasource-for-the-embedded-glassfish-maven-plugin/ | 安装maven和glassfish的插件的一片文 |
| http://www.oracle.com/technetwork/cn/articles/java/unittesting-455385-zhs.html | 模拟测试EJB的一个参考文献 |
| http://yyq2007aa.blog.163.com/blog/static/441816092010101601814865/ | 网易上的单独跑EJB容器的一篇文章，其中的文件配置具有参考意义。 |
| http://www.eclipse.org/eclipselink/documentation/2.4/solutions/testingjpa002.htm | 测试Entity Bean时，如何配置persistence.xml的问题，只需要在domain.xml中配置即可。 |

### JUnit测试参考文献

|  |  |
| --- | --- |
| URL | 文献内容说明 |
| http://www.blogjava.net/iamtin/archive/2006/11/17/81802.html | 此文献主要说明在JUnit测试中，使用Setup和Teardown的用法，同时对@eforeclass, @AfterClass, @Before, @After四个标签的区别给出了说明。针对EJB测试，我们只使用@BeforeClass和@AfterClass,因为这两个标签在所有的测试方法中只运行一次。 |
| https://github.com/junit-team/junit/wiki/\_pages | Junit项目的使用文档说明的官网地址。 |