技術評論社発行「BEA WebLogic Server 8.1J EJB 2.0徹底攻略」 Appendix Bより抜粋

Cactus （上）

EJBの単体テスト

皆さんは EJB の単体テストをどのように行っているだろうか。プロトタイピング・プログラム等でなく、本格的なエンタープライズレベル・アプリケーションであれば、EJB の単体テストは実施したほうがよいのは言うまでもない。しかし、EJB2.0 仕様からローカルインタフェースが導入されたことにより、EJB の単体テストは大変面倒なものとなってしまった。中にはセッション Façade の EJB と一緒にテストをしてしまえ！というアプリケーション開発プロジェクトもあるかもしれない。
幸い、WebLogic Server はローカルインタフェースの EJB であっても、EAR ファイルにパッケージ化すればサーブレットから呼ぶことができる。本書のアプリケーションの EJB も、この特性を利用して Cactus を用いて EJB の単体テストを行っている。このセクションでは、WebLogic Server 上の EJB（リモートおよびローカルインタフェース）のテストを Cactus を用いて行う方法について説明する。

Cactus とは？

Cactus は Apache Jakarta プロジェクトが開発しているサーバサイド Java のためのテスティング・フレームワークで、サーブレット、サーブレットフィルタ、タグライブラリおよび EJB のテストを行うことが可能である。Cactus は Java のテスティング・フレームワークである JUnit を拡張している。ここでは、WebLogic Server 上の EJB を Cactus を用いてテストする具体的な方法を説明する。

Cactus のしくみ

図1に Cactus を使ったテスティングフレームワークでのコンポーネントの全体像を示す。

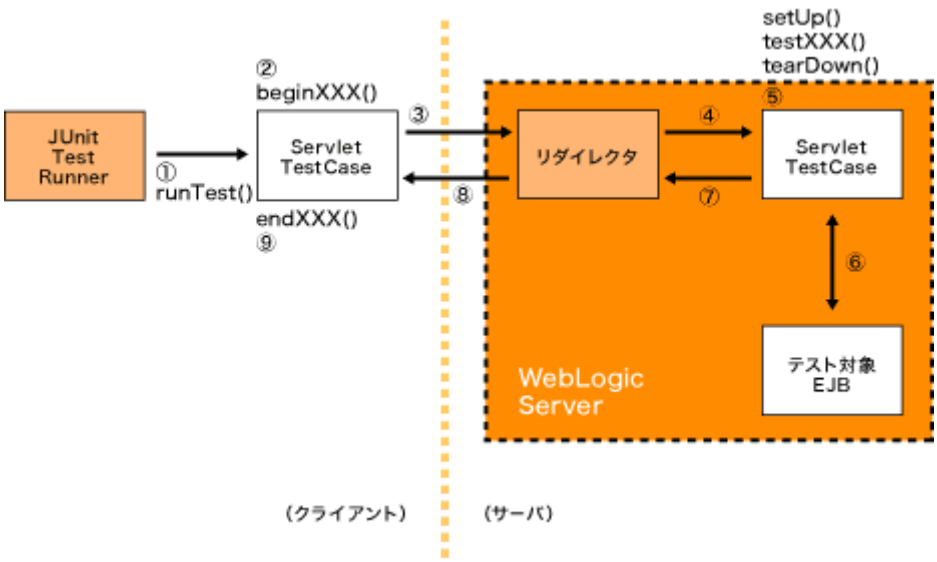

図 1 Cactus のコンポーネント

このうち、「JUnit Test Runner」が JUnit が提供しているコンポーネント、「リダイレクタ」は Cactus が用意しているコンポーネントである。「テスト対象 EJB」と「ServletTestCase」のコンポーネントを開発者が開発する。「テスト対象 EJB」はいうまでもなく、アプリケーションを構成する EJB である。「ServletTestCase」は Cactus が提供する ServletTestCase クラスを extends して実装する「テストケース」と呼ばれるものだ。図 1 には「ServletTestCase」はクライアントサイド、サーバサイドの 2 つ存在するが、同じクラスの別インスタンスがクライアントとサーバ側にそれぞれ生成されることを示している。

これらのコンポーネントの処理フローは次のようになる。

(1) JUnit の Test Runner を起動してテストを実行すると、ServletTestCase の runTest()が呼ばれる。
(2) ServletTestCase の runTest()は、beginXXX()というメソッドを見つけてあればそれを 呼び出す。beginXXX()では HTTP ヘッダー、HTTP パラメータを初期化する。
(3) runTest()は続いてリダイレクタに HTTP コネクションをオープンし、HTTP リクエストとして②で設定したパラメータを渡す。
(4) リダイレクタはサーバサイドで ServletTestCase のインスタンスを生成する。
(5) リダイレクタは続いて ServletTestCase の setUp()、testXXX()、tearDown()メソッドをこの順で呼び出す。setUp()、tearDown()の実装は任意である。
(6) ServletTestCase のメソッド testXXX()の中でテスト対象 EJB のメソッドを呼び出し、テストを実行する。
(7) テストに失敗した場合、testXXX()メソッドはリダイレクタに例外を投げる。
(8) リダイレクタはその例外をクライアント側に投げ、Test Runner 上に表示する。
(9) テストに成功すると、クライアント側の ServletTestCase インスタンスのメソッド endXXX()が探され、あればそれが実行される。

なお、サーブレットフィルタをテストするときは FilterTestCase クラスを、タグライブラリをテストするときは JspTestCase クラスを、それぞれ ServletTestCase クラスの代わりに extends してテストケースを記述する。

テストケースの実装例

EJB のテストケースの実装は、次のようなパターンとなる。

```
import org.apache.cactus.*;
import junit.framework.*;

public class ZZZTestCase extends ServletTestCase {
    public ZZZTestCase( String name ){
        super( name );
    }
    public void setUp() throws Exception {
        ……
    }
    public void tearDown() throws Exception {
        ……
    }
    public void testXXX() throws Exception {
        ……
    }
        ……
}
```

Cactus フレームワークを用いるには、org.apache.cactus.*と、junit.framework.*を import する。クラスは org.apache.cactus.ServletTestCase を extends する（ServletTestCase は JUnit の TestCase クラスを extends している）。String 型を1つとるコンストラクタを作成し、親クラスのコンストラクタを呼び出すようにする。”test”で始まるメソッドを用意し、その中にテストコードを記述する。各”test”メソッドの前には setUp()メソッドが、”test”メソッドの後には tearDown()メソッドが Cactus フレームワークにより呼び出される。これらの実装は任意である。例えば本書のアプリケーションの実装2の Item クラスをテストする ItemTestCase では次のように Item のホームスタブを取得している。

```
public void setUp() throws Exception {
    Context ctx = new InitialContext();
    Object obj = ctx.lookup( JNDI_ITEMHOME );
    m_home =
      (ItemHome)PortableRemoteObject.narrow( obj, ItemHome.class );
    ctx.close();
}
```

以下に同クラスの”test”メソッドのひとつである、testGetItemsAll()メソッドを示す。

```
public void testGetItemsAll() throws Exception {
    final int NUMBER_OF_ITEMS = 44;
    Item item = m_home.create();
    Collection items = item.getItemsAll();
    assertTrue( items.size() == NUMBER_OF_ITEMS );
    item.remove();
}
```

testGetItemsAll()メソッドでは、Item EJB を取得して getItemsAll()を実行し、取得できた Item オブジェクトの数が４４件ならばテストが成功であるとしている。このテストは、付録 A で説明した手順で同梱の CD-ROM からアプリケーションをインストールしてデータベースを初期化し、データベースにあらかじめ商品レコードが４４件あることを前提に書かれているので、そうでない場合テストは失敗する。このテスト対象 EJB は参照系メソッドだけを持ち登録系メソッドを持たないため、この例のようにデータベースにあらかじめ想定したデータが存在していることを前提に記述している。しかし、実際のテスト環境や手順によってはこのような前提がわずらわしい場合もあるので、その場合は”test”メソッドの中で JDBC プログラミングでデータベースにデータを INSERT し、それを EJB で取り出す、というような実装が必要である。いずれにしても、”test”メソッドの中では各テストが完結している必要がある。また、”test”メソッドが呼ばれる順番は決まっていないので、相互で独立したテストを記述するように心がけなければならない。例えば、testA()メソッドでデータベースに INSERT したレコードは、testA()メソッドを終了する前に DELETE しておかなければならない、などである。

その他、assertXXX()メソッドのバリエーションや、意図的に EJB の Exception を起こし Exception がキャッチできなかったときにテストを失敗させる方法、多数のテストケースクラスをまとめてテストスイートにする方法などは、長くなるので説明を省く。興味のある方は本書付属の CD-ROM に入っているテストケースのソースコードや、Cactus/JUnit の Web ページなどを参照いただきたい。