WebOTX V8.1 新規機能　EJB 3.0

WebOTX V8.1より、Java EE 5(Java Platform, Enterprise Edition 5)に対応しました。これによりいろいろな機能追加が行われていますが、特に大きな変更であるEJB 3.0対応についてご紹介いたします。なお、WebOTX V7で対応したEJB 2.1についてもWebOTX V8.1で引き続き利用することが可能です。

1. EJB 2.1 とEJB 3.0 の違い

EJB 3.0対応を始めとする、Java EE 5では「EoD(Ease of Development:開発の容易さ)の実現」を主な目的としています。そのため、EJB 3.0においてもEJB 2.1と比較して、開発の面においてさまざまな改良が行われています。

具体的には、以下のような点が改良されています。

- ビジネスロジック部分を単純なJavaオブジェクト(POJO：Plain Old Java Object)として実装
  →Java さえ知っていれば、ビジネスロジックが実装可能に！
- 複雑で記述ミスが発生しがちな配備記述子が不要
  →配備記述子の形式、設定を学習する必要なし！
- 従来ビジネスロジックのあちこちに埋め込みが必要だったコード（ログ出力コード等）を、ビジネスロジックを変更せずに埋め込み可能
  →分かりやすく、メンテナンスしやすいソースコードに！
- 従来EJBのテストに必要だったEJBコンテナが不要
  →評価が容易に行えるようになり、開発効率が格段にアップ！

以下ではこれらそれぞれについて説明します。

1. 1. ビジネスロジック部分を単純なJavaオブジェクト(POJO)として実装

EJB 2.1では、Bean作成の際には、Beanクラスそのものに加え、EJB独自のインタフェース定義を行う必要がありました。ステートレスセッションBeanの場合、以下の実装が必要でした。

- SessionBeanクラスを継承し、ビジネスロジックを含んだBeanクラス
- EJBHomeインタフェースを継承したホームインタフェース
- EJBObject インタフェースを継承し、ビジネスインタフェースを含んだコンポーネントインタフェース

一方、EJB 3.0では、アノテーション(@Stateless等の@で始まる記述)を利用することにより、EJB独自インタフェースは利用せずに、ビジネスインタフェースとビジネスロジックの実装だけで十分です。つまり、単純なJavaオブジェクト(POJO)として実装可能となります。

これにより、EJBを知らなくてもJavaさえ知っていれば、ビジネスロジックが実装可能になります。

- EJB 2.1 のプログラム例

インタフェース定義
```
public interface HelloHome extends EJBHome {

    public Hello create() throws CreateException, java.rmi.RemoteException;
}
public interface Hello extends EJBObject {
    public String hello(String str) throws RemoteException;
}
```

Bean クラス定義
```
public class HelloBean implements SessionBean {

    private SessionContext sessionContext;
    public HelloBean() {
    }

    public void setSessionContext(SessionContext arg0) {
        this.sessionContext = arg0;
    }

    public void ejbRemove() {
    }

    public void ejbActivate() {
    }

    public void ejbPassivate() {
    }

    public void ejbCreate() {
    }

    public String hello(String str) {
        return "Hello : " + str;
    }
}
```

- EJB 3.0 のプログラム例

インタフェース定義
```
public interface Hello {
    String hello(String str);
}
```

Bean クラス定義
```
@Stateless public class HelloBean implements Hello {

    public String hello(String str) {
        return "Hello : " + str;
    }
}
```

1.2. 複雑で記述ミスが発生しがちな配備記述子が不要

EJB 2.1 では、配備記述子を作成し、アプリケーション中にパッケージする必要がありましたが、EJB 3.0 ではこれが必須ではなくなりました。配備記述子を作成しない場合、既定値で動作することになります。また、配備記述子の代わりにアノテーションを利用して属性の指定を行うことも可能です。

これにより、配備記述子の形式や設定内容を学習しなくても、EJB を利用することが可能となります。

- EJB 2.1 の配備記述子(ejb-jar.xml)の例

```
<?xml version="1.0" encoding="MS932"?>
<ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
    <enterprise-beans>
        <session>
            <ejb-name>Hello</ejb-name>
            <home>sample.HelloHome</home>
            <remote>sample.Hello</remote>
            <ejb-class>sample.HelloBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
    </enterprise-beans>
</ejb-jar>
```

- 配備記述子の内容を EJB 3.0 のアノテーションで記載した例

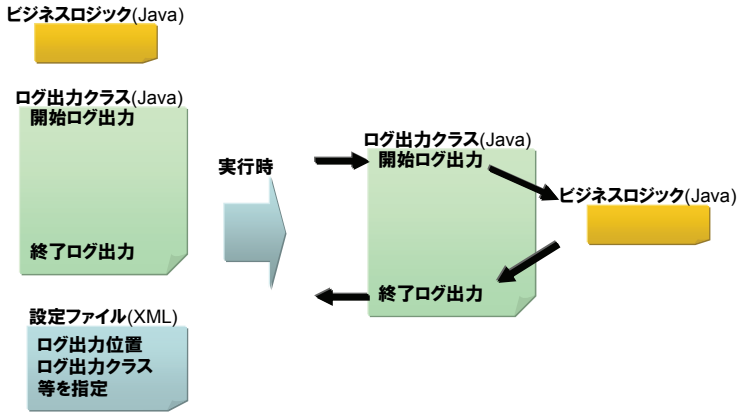
```
@Stateless
@Remote
@TransactionManagement(TransactionManagementType.CONTAINER)
public class HelloBean implements Hello {
```

1.3. ビジネスロジックのあちこちに埋め込みが必要だったコードを、ビジネスロジックを変更せずに埋め込み可能

EJB 3.0 では、新たにアスペクト指向プログラム(AOP)という概念が導入されています。これはオブジェクト指向と少し違う方法で、ソフトウェアの複雑さの低減や再利用性を向上させるための仕組みです。

アスペクトの例としてよく使われるのが、ログやトランザクション処理の例です。一般にログの出力コードはプログラム中にあちこち分散します。アスペクト指向を使うと、ログ出力はコード中には記述しません。代わりに、「どこにログを出すか」、「どんなログを出すか」を別に定義します。

これにより、Bean クラス内にはビジネスロジックのみがコーディングされることになり、分かりやすく、メンテナンスしやすいコードとなります。



1. 4. 従来EJBのテストに必要だったEJBコンテナが不要

EJB 2.1 では、EJB 独自のインタフェースやクラスを継承して実装する必要があったため、実装したプログラムのテストを行う際は、EJB コンテナが必要でした。

EJB 3.0 では、POJO で実装可能となり、EJB 独自のインタフェースやクラスが不要となったため、EJB コンテナが無くてもテストが実施できるようになりました。

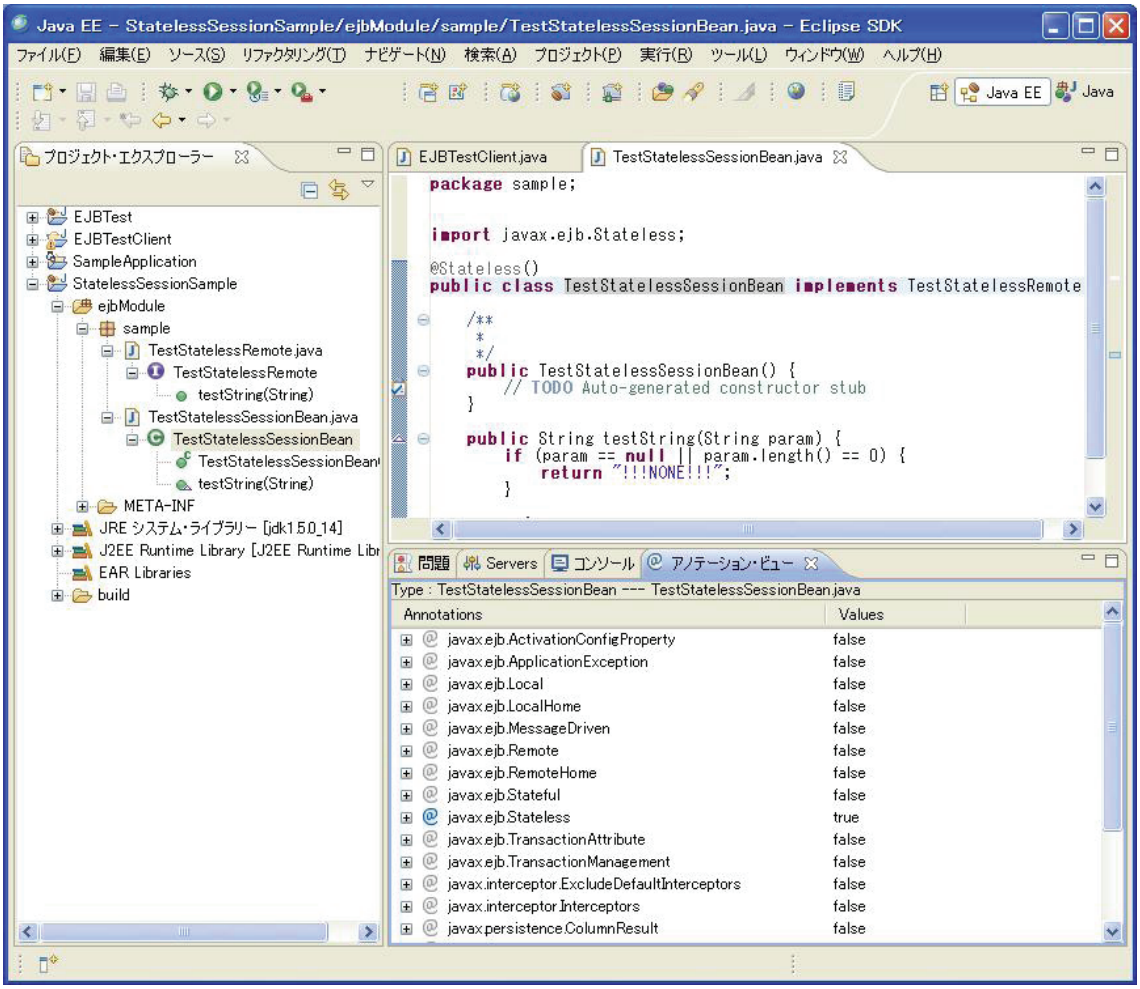
これにより、テストが容易に行えるようになり、開発効率が格段にアップします。

2. WebOTX Developerで提供するアノテーションエディタの利用

WebOTX Developer は、新たにアノテーションエディタを導入しています。これは以下の機能を提供しています。

- アノテーションの一覧表示・編集機能
  Java ソースに持つアノテーションを一覧表示します。それに、GUI でアノテーションの付与・更新・削除機能を提供します。
- アノテーション表示選択・ビュー機能
  用途別にアノテーションをカテゴリに分け、そのカテゴリ毎にビューに表示するかしないか選択できる機能を提供します。この機能により、ユーザが表示するアノテーションを制御できるようになります。
- アノテーションの検証機能
  ある要素にアノテーションが付与できるかどうかの検証やアノテーションの属性などに入力される値の検証機能を提供します。

これらの機能により、アノテーションを容易に定義、編集することができるようになり、EJB 3.0 アプリケーションの作成をより効率的に行うことができるようになります。



詳細については、WebOTX マニュアルのアプリケーション開発ガイドの「第４部　プログラミング・開発」－「1.5. アノテーションエディタ」を参照してください。

以上。