

Seasar2: 第 1 回 WebSphere Application Server へ Web アプリケーションをデプロイする

Seasar2 と Teeda を利用した Web アプリケーションを WebSphere Application Server にデプロイする。

中村 年宏 (nakamura.toshihiro@isid.co.jp)
株式会社 電通国際情報サービス

2008年 2月 01日

本連載では、軽量コンテナ Seasar2 を IBM WebSphere Application Server (以降 WAS と表記) や IBM DB2 と連携させる方法について紹介します。連載の第1回目となるこの記事では、Seasar2 を利用するWebアプリケーションをWASへデプロイする際に気をつけるべき事柄について説明します。

[このシリーズの他の記事を見る](#)

はじめに

Seasar2 は DI (Dependency Injection) と AOP (Aspect Oriented Programming) をサポートする軽量コンテナです。Seasar2 はスタンドアロン環境ではもちろん、[Apache Tomcat](#) のようなサーブレットコンテナ、WAS のような J2EE1.4 準拠のアプリケーションサーバー、また JavaEE5 準拠のアプリケーションサーバー上で動作します。

Seasar2 を使い Web アプリケーションを開発するには、Seasar2 と連携する Web フレームワーク Teeda が利用可能です。

Teeda とは Java の標準仕様である JSF (JavaServer Faces) をベースにしたフレームワークです。JSF のバージョン1.1 に対応していますが、それだけでなく、JSF を独自に拡張した機能や Ajax に対応した機能も備えます。テンプレートに JSP (JavaServer Pages) ではなく HTML を利用できることが大きな特徴になっています。

以下では、Seasar2 と Teeda を利用した Web アプリケーションを WAS へデプロイする際の考慮事項について解説します。

動作環境

コラム1. Seasar2 が動作可能な WAS のバージョン

WAS はバージョン5.1 から JDK1.4 に対応し、バージョン6.1 から JDK5 に対応しています。Seasar2 は JDK1.4 以上の環境を必要とします。したがって、Seasar2 が動作可能な WAS のバージョンは 5.1 以降です。

Teeda HTML Example は Java5 のアノテーションを使用していますが、バージョン6.1 より以前の WAS では、上記の理由により Java5 の機能であるアノテーションやジェネリクスを使用できません。Teeda HTML Example を以前のバージョンで動作させるには、Java5 のアノテーションの代わりに相当する定数アノテーションを使用するように修正してください。定数アノテーションは、Java の static な定数を利用して表されるメタ情報です。ジェネリクスについては使用しないように修正してください。

次のプラットフォームを使用します。

OS

Windows XP Professional SP2

アプリケーションサーバー

WebSphere Application Server Network Deployment V6.1.0.13 (以降 WAS と表記します)

Seasar プロジェクトで開発されている次のプロダクトを使用します。

- S2Container 2.4.22
- S2-Tiger 2.4.22
- Teeda 1.0.12-rc2
- Teeda HTML Example 1.0.12-rc2

Teeda HTML Example は Seasar2 と Teeda を使った動作確認用 Web アプリケーションです。ここでは、WAS 上にデプロイするサンプルとして使用します。

Teeda HTML Example は [Sysdeo/SQLI Eclipse Tomcat Launcher plugin](#) を使った Tomcat プロジェクトとして作成されています。このプラグインを用意し、Teeda HTML Example を Eclipse にインポートすると、Tomcat 上で即座に動作します。

以下の記事では、これらのバージョンやエディションの表記は省略します。

デプロイ時の考慮事項

デプロイ時の考慮事項は2つあります。

1 つ目は WAS へのデプロイに含める JAR ファイルについてです。デプロイにあたってどの JAR ファイルを含めるべきで、どの JAR ファイルを含めるべきではないかを説明します。

2 つ目は WAS のクラス・ローダーに関してです。WAS のクラス・ローダー順序を変更する必要性について説明します。

JAR ファイル

Seasar2 は JTA (Java Transaction API) の実装を提供しているため、JTA の API を含む `geronimo-jta_1.1_spec-1.0.jar` を同梱して配布しています。Tomcat など、JTA が備わっていない環境で Seasar2 の JTA 実装を利用するには、この JAR ファイルが必要です。WAS のように標準で JTA 実装

を提供している環境では、逆にこの JAR ファイルは不要です。すでに必要なクラスが WAS 内に存在するからです。

しかし、ここで注意すべきことがあります。それは、WAS と Seasar2 で使用する JTA のバージョンが違うことです。WAS は J2EE1.4 準拠のアプリケーションサーバーなので JTA のバージョン 1.0 に対応しています。一方、Seasar2 は JTA のバージョン 1.1 を使用します。Seasar2 では、JTA1.0 と JTA1.1 の差分のインタフェースのみを含む `geronimo-jta_1.1_spec-1.0-tsr.jar` を提供しています。そして、差分の実装は Seasar2 が提供するため、WAS 上でも JTA1.1 が使用できるようになっています。デプロイの際には、`geronimo-jta_1.1_spec-1.0.jar` ではなく `geronimo-jta_1.1_spec-1.0-tsr.jar` を含めるようにしてください。

ここでは JTA に関する JAR ファイルについて述べましたが、他の J2EE (JavaEE) 関連の JAR ファイルについても、要/不要が環境により変わります。WAS にデプロイする場合において、Seasar2 が同梱している J2EE (JavaEE) 関連 JAR ファイルの要/不要をまとめました (表1)。ただし、Feature Pack で WAS に新機能を追加した場合にはこの限りではないことがあります。

表1. J2EE (Java5) 関連JARファイルの要/不要

JARファイルの名称	要/不要
<code>geronimo-jta_1.1_spec-1.0.jar</code>	不要
<code>geronimo-jta_1.1_spec-1.0-tsr.jar</code>	S2JTA(注1) または S2Tx(注2) を使うならば必要
<code>geronimo-ejb_2.1_spec-1.0.jar</code>	不要
<code>geronimo-jpa_3.0_spec-1.0.jar</code>	Seasar2 の S2JDBC(注3) 拡張機能または JPA を使うならば必要
<code>geronimo-annotation_1.0_spec-1.0.jar</code>	Seasar2 の EJB3 機能を使うならば必要
<code>geronimo-ejb_3.0_spec-1.0.jar</code>	Seasar2 の EJB3 機能を使うならば必要
<code>geronimo-interceptor_3.0_spec-1.0.jar</code>	Seasar2 の EJB3 機能を使うならば必要

注1: S2Tx とは、Seasar2 の AOP によるトランザクション管理機能です。

注2: S2JTA とは、Seasar2 の JTA 実装です。

注3: S2JDBC とは、Seasar2 の JDBC フレームワークです。

クラス・ローダー

WAS のクラス・ローダーは階層構造になっており、上位には WAS が自身の実行に必要なクラスを読み込む WebSphere 拡張クラス・ローダー、その下位にはアプリケーションモジュールのクラス・ローダー、さらにその下位には Web モジュールのクラス・ローダーが連なります。

WAS は、デフォルトでは上位のクラス・ローダーから優先してクラスを読み込みます。仮に、アプリケーションモジュール (EAR ファイル) や Web モジュール (WAR ファイル) に、WAS で使われているクラスと同名のクラスがあったとしても読み込みません。

WAS は JSF の実装を提供しています。一方、Teeda は JSF の実装そのものです。すなわち、Teeda のライブラリを WAS にデプロイすると、クラス・ローダーの階層の中に JSF の同名クラスが複数存在することになります。デフォルトのロード順序では、WAS のクラスが優先され、Teeda のクラスは使用されません。しかし、アプリケーションは Teeda と連携するように作られているため、アプリケーションが動作するには Teeda のクラスがロードされねばなりません。

したがって、この問題を解決するには、WebSphere 拡張クラス・ローダーよりもアプリケーションモジュールや Web モジュールのクラス・ローダーを優先するように「クラス・ローダー順序」の設定を変更する必要があります。

コラム2. WAS 6.0 以前での注意点

WAS バージョン6.0 以前では、WAS内部で使用されるライブラリとアプリケーションで利用するライブラリの重複に注意する必要があります。アプリケーションのライブラリ・クラスが使用されなかったり、クラスの衝突が発生したりするからです。代表的なライブラリとしては、Xerces や commons-logging が挙げられます。

WAS バージョン6.1 からは、クラスローダーの仕組みが変更され、アプリケーションから WAS 内部のライブラリ・クラスは見えなくなりました。したがって WAS バージョン6.1 以降では、ここに挙げる注意点は当てはまりません。

Xerces

Teeda では、XHTML の解析のために Xerces2.6.2 を使用します。そのため、Teeda を利用する Web アプリケーションは WEB-INF/lib の下に Xerces2.6.2 を持ちます。Teeda は、利便性を高めるために、Xerces2.6.2 の実装クラスを独自に拡張し、XHTML としては不正な属性を意図的に許容する対応を行っています。たとえば、XHTML の属性中に登場する「&」を認め、エラーにしません（正しい XHTML としては「&」ではなく「&」と記述すべきです）。

Teeda が認めている XHTML としての記述例

```
<a id="goLinkResult"
href="linkResult.html?arg1=1111&arg2=2222">Link</a>
```

正しい XHTML としての記述例

```
<a id="goLinkResult" href="linkResult.html?arg1=1111&amp;arg2=2222">Link</a>
```

しかし、WAS バージョン6.0 以前では、WEB-INF/lib の Xerces2.6.2 ではなく、IBM JDK の Xerces が優先して使用されてしまいます。優先順位は、WAS の「クラス・ローダー順序」の設定を利用しても変更不可能です。

IBM JDK が抱える Xerces を含む Xerces2.6.2 よりも上位のバージョンでは、Xerces2.6.2 において Teeda が拡張しているクラスの実装に変更が加えられています。そのため、Teeda による拡張処理が有効になりません（たとえば、属性中の「&」は認められずエラーとなります）。

このことは、WAS 上の環境と WAS 以外の環境で XHTML 解析の挙動が異なるということを意味します。これでは不便なので、Teeda では、Xerces の解析方法をカスタマイズ可能にすることで、どのような環境であっても本来の XHTML として解析できるようにしています。挙動を統一するために、WAS 上で動かすことを想定する場合は、`app.dicon` に次の記述を必ず追加してください。

```
<component class="org.seasar.teeda.extension.html.impl.TeedaXMLReaderFactory$STRICT"/>
```

commons-logging

WASは内部で commons-logging の LogFactory のカスタム実装を利用しますが、アプリケーションで commons-logging を利用する場合に、LogFactory が衝突します。

WAS で commons-logging を利用する上での注意点は、[Commons LoggingのTroubleshooting Guide](#)で詳しく解説されています。

デプロイ

Web アプリケーションを WAS にデプロイする手順を進めながら、JAR ファイルやクラス・ローダーの問題を解消する方法を以下に示します。

サンプルアプリケーション Teeda HTML Example の取得

Teeda HTML Example は [Teeda のサイトから ZIP ファイルをダウンロード](#)し、Eclipse にインポートしてください。Teeda を使ったアプリケーションに必要なディレクトリ構成がわかると思います。インポート後、Eclipse でプロジェクトをビルドしてください。WAR ファイルを作成するには次の「WAR ファイルの作成」へ進んでください。

デプロイをすぐに試したい場合は、[WAR ファイルをダウンロード](#)し、「アプリケーションのインストール」へ進んでください。

WAR ファイルの作成

Eclipse にインポートしたできるプロジェクト `teeda-html-example` は、`build.xml` を含んでいます。Eclipse の Ant 実行機能でこのファイルの `create-war-was61` ターゲットを実行してください。WAR ファイルが `teeda-html-example` プロジェクト直下の `target` フォルダに作成されます。

`create-war-was61` ターゲットでは `geronimo-jta_1.1_spec-1.0.jar` の代わりに `geronimo-jta_1.1_spec-1.0-tsr.jar` を `WEB-INF/lib` に含めています。

アプリケーションのインストール

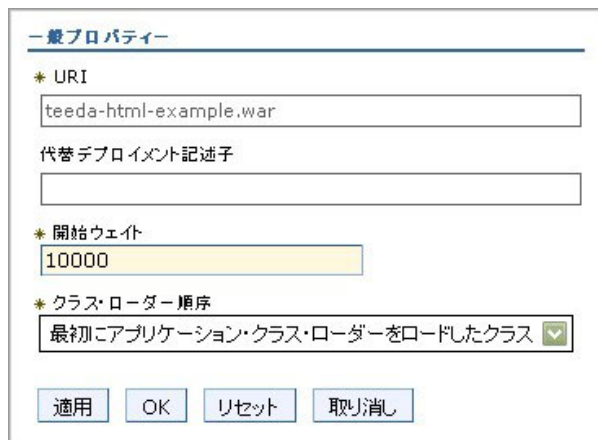
WAS の管理ソールから war ファイルをインストールします。

コンテキストルート名には「`teeda-html-example`」と指定してください。画面を進めてインストールを完了してください。コンテキストルート名以外はデフォルトの設定を使用します。

クラス・ローダーの設定変更

インストールが完了したら「エンタープライズ・アプリケーション」 - 「`teeda-html-example_war`」 - 「モジュールの管理」 - 「`teeda-html-example.war`」に進み、「クラス・ローダー順序」のプルダウンメニューで「最初にアプリケーション・クラス・ローダーをロードしたクラス」を選択します。（図1）

図1. Webモジュールのクラス・ローダー順序を変更する



この設定により WebSphere 拡張クラス・ローダーよりも Web モジュールのクラス・ローダーが優先されるようになります。設定が終わったら変更を保管しデプロイを完了します。

もし、複数の Web モジュールから共通に使われる JAR ファイルを EAR ファイルにパッケージングしてデプロイする場合は、アプリケーション・モジュール・クラス・ローダーについても同様に變更してください。變更画面へは、「エンタープライズ・アプリケーション」 - 「teeda-html-example.war」 - 「クラス・ロードおよび更新の検出」で辿れます。「クラス・ローダー順序」のラジオボタンで「最初にアプリケーション・クラス・ローダーをロードしたクラス」を選択します。（図2）

図2. アプリケーションモジュールのクラス・ローダー順序を変更する



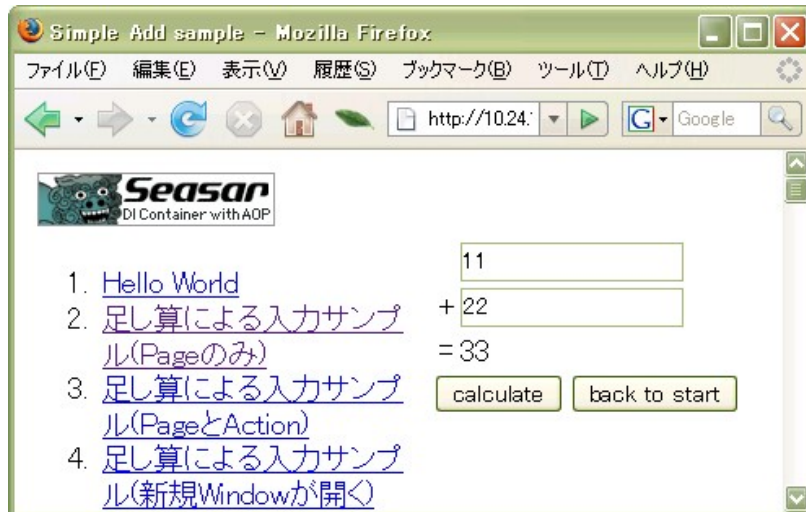
この設定により WAS のクラス・ローダーよりもアプリケーションモジュールのクラス・ローダーが優先されるようになります。

デプロイしたアプリケーションの動作確認

インストールしたアプリケーションを起動し、「<http://#####/teeda-html-example>」にアクセスしてください。Teeda HTML Example の画面が表示されます。

動作確認として「足し算による入力サンプル (Page のみ)」のリンクをクリックしてください。入力項目が開かれたら適当な値を入れ、calculate ボタンを押してください。足し算の結果が正しく表示されればデプロイは正しく行われています。(図3)

図3. デプロイ成功の確認



最後に

Seasar2 と Teeda を利用した Web アプリケーションを WAS にデプロイする際の考慮事項として、JAR ファイルとクラス・ローダーについて説明しました。

著者について

中村 年宏

IBM AS/400 の RPG 言語を使ったプログラミングを経験して以来、データベースアクセス技術に興味を持つ。Seasar2 プロジェクトでは、S2Daoや Kuina-Dao などデータベースアクセスを簡単に行うためのプロダクトを中心にコミットを務めている。

© Copyright IBM Corporation 2008

(www.ibm.com/legal/copytrade.shtml)

商標

(www.ibm.com/developerworks/jp/ibm/trademarks/)