Oracle JDeveloper for Windows NT

リリース・ノート

リリース 3.1

2000 年 7 月

部品番号: J01886-01

原典情報: JDeveloper 3.1 Release Notes (A82929-01)

目次

はじめに

Oracle JDeveloper は、インターネット用データベース・アプリケーションを作成、デバッグおよび配布するためのオラクル社の Java 開発ツールです。今回のリリースでは、Oracle Business Components for Java が導入されました。これは、XML で強化された 100% Java のアプリケーション・コンポーネント・フレームワークで、インターネット用多層 Java アプリケーションの開発、配布およびカスタマイズが大幅に簡略化されます。

この印刷されたリリース・ノートには、Oracle JDeveloper リリースに関する最新情報が記載されています。これらの情報は、CD-ROM 上のオンライン・リリース・ノートに置き換わります。

日本語版 Oracle JDeveloper R3.1 の詳細バージョンは 3.1.1.2 です。リリース・ノートにおいて、これらのバージョンは同一のものを指しています。製品の起動時およびバージョン情報では 3.1.1.2 が表示されますこと、あらかじめご承知置きください。

システム要件

サーバー環境 Oracle8*i* Enterprise Edition for Windows NT R8.1.6

コンピュータ本体	Windows NT 4.0 のハードウェア互換リストに記載された Pentium プロセッサ 200MHz 以上
基本ソフトウェア	Microsoft Windows NT 4.0（2000 年問題修正付き Service Pack 3、Service Pack 5 またはそれ以降）、Windows 2000
メモリ	96MB 以上（推奨 256MB 以上）
ハードディスク	961MB 以上

開発環境 Oracle JDeveloper for Windows NT R3.1

コンピュータ本体	Windows NT 4.0 のハードウェア互換リストに記載された Pentium プロセッサ 200MHz 以上
基本ソフトウェア	Microsoft Windows NT 4.0（Service Pack 5）、Windows 2000
メモリ	96MB 以上（推奨 128MB 以上）
ハードディスク	標準インストール時には 280MB

使用可能なドキュメントおよび他のリソース

JDeveloper 3.1 オンライン・ヘルプ・システム

ドキュメントの 3 つのオプション

JDeveloper 3.1 のオンライン・ヘルプ・システムには、次の 3 つの形式があります。

- HTML Help

  HTML Help 形式のオンライン・ヘルプを使用する場合、マシンに Microsoft Internet Explorer 4.0 以降がインストールされている必要があります。このドキュメントは、圧縮形式の`*.chm` ファイルで提供されています。全文検索機能が使用可能です。

- WebHelp

  WebHelp 形式を使用する場合、Netscape Navigator 4.0 または Microsoft Internet Explorer 4.0 以降のいずれでも使用できます。このドキュメントは、非圧縮の`*.html` ファイルで提供されています。ドキュメントはサイズの小さい多数のファイルで構成されているため、（NTFS ではなく）FAT ファイル・システムにインストールする場合には、大量のディスク容量が必要になります。実際の必要ディスク容量は、ハード・ディスクおよびセクターのサイズにより異なります。

- JavaDoc

  JavaDoc 形式のリファレンスは、JDeveloper 内部から直接使用できます。「ドキュメント」タブで JavaDoc ビューアを使用して表示します。このリファレンスは、圧縮形式の`*.zip` ファイルで提供されています。

オンライン・ヘルプのインストールと設定

Oracle JDeveloper のインストーラにより、コンピュータにどのブラウザがインストールされているかが検出され、適切な形式のヘルプがインストールされます。

- Internet Explorer 4.0 以降がインストールされている場合、自動的に HTML Help がインストールされます。
- Internet Explorer が検出されない場合、WebHelp がインストールされます。
- いずれの場合でも、JDeveloper 内の JavaDoc システムは常に使用可能です。

適切なブラウザをインストールしていない状態で任意のヘルプ・システムを強制インストールするには、Oracle JDeveloper インストーラの「カスタム」インストール・オプションを使用します。希望のオンライン・ヘルプ・オプションを選択してください。

- Internet Explorer 4.0 以降をインストールしていない状態で HTML Help を強制インストールした場合、Oracle JDeveloper のインストール後に Internet Explorer をインストールするよう警告されます。HTML Help システムは、Internet Explorer をインストールせずに起動することはできません。
- Netscape Navigator 4.0 以降または Microsoft Internet Explorer 4.0 以降をインストールしていない状態で WebHelp を強制インストールした場合、Oracle JDeveloper のインストール後にどちらかをインストールするよう警告されます。WebHelp システムは、いずれかのブラウザをインストールせずに起動することはできません。

HTML Help をインストールしても、マシン上で適切なバージョンの Internet Explorer が使用できない場合、ヘルプ・システムの起動時に次のようなエラーが発生します。

- 「`shlwapi.dll` が指定されたパスに見つかりません。」
- JDeveloper が予期せずに終了したことを示す「Windows NT ワトソン博士エラー」

WebHelp をインストールしても、Netscape Navigator で複数のプロファイルを設定している場合、Oracle JDeveloper の「ヘルプ」メニューからヘルプ・システムを起動できません。

Oracle JDeveloper 3.1 の機能紹介

JDeveloper 3.1 の新機能を紹介するページは、以下の URL で閲覧できます。

- http://www.oracle.co.jp/jdev/
- http://www.oracle.co.jp/tools/

サポートされている配布環境

Oracle JDeveloper を使用して、アプレットおよびアプリケーションを様々な環境に配布できます。JDeveloper は Sun Microsystems の JDK 1.2.2 および 1.1.8 に準拠しており、JDK 1.1.8 以降がインストールされているプラットフォームへの配布が可能です。

Oralce JDeveloper は、次の環境をサポートしています。

注意: この印刷されたリリース・ノートには、Oracle JDeveloper リリースに関する最新情報が記載されています。これらの情報は、CD-ROM 上のオンライン・リリース・ノートに置き換わります。

ブラウザ [1]
Netscape Navigator 4.7
Windows Internet Explorer 5.0（Service Pack 1）
Java Runtime Environment 1.1.8 および 1.2.2
Appletviewer 1.1.8 および 1.2.2

1) Netscape Navigator および Microsoft Internet Explorer に搭載されている Java VM は、Oracle JDeveloper で使用される Java VM 1.2.2 より以前のバージョンです。そのため、InfoSwing コントロールおよび InfoProducers を使用して開発したアプレットを実行するには、ユーザーのブラウザに Java VM Plug-in をインストールする必要があります。
Plug-in は、http://java.sun.com/products/plugin/index_ja.html からダウンロードが可能です。

アプリケーション・サーバー
Apache 1.3.12 - JServ 1.1
Apache 1.3.12 - Tomcat 3.1

オペレーティング・システム [1]
Windows 95 および 98
Windows 2000
Windows NT 4.0（Service Pack 3 以降）
Solaris 2.6

1) これらは、JDeveloper 3.1 で作成し適切なアプリケーション・サーバーまたはデータベース・サーバーに配布された、アプリケーション、アプレットおよび JSP 用のクライアント・ランタイム・プラットフォームです。

JDBC
Oracle Thin JDBC
Oracle JDBC-OCI7 [1]
Oracle JDBC-OCI8 [1]
Sun JDBC-ODBC ドライバ

1) OCI ドライバは、アプレットには使用できません。

データベース [1]
Oracle7 RDBMS リリース 7.3.4
Oracle8 RDBMS リリース 8.0.5
Oracle8*i* RDBMS リリース 8.1.5 およびリリース 8.1.6

1) この表は、接続および配布が可能なデータ・ソースのリストです。配布がサポートされているデータベースは、Oracle8*i* 8.1.6 のみです。

Oracle JDeveloper と Oracle8*i* の互換性リスト

Oracle8*i* リリース 2（8.1.6）では、基礎となる Java クラス・ライブラリが Oracle8*i*（8.1.5）から一部変更されています。この変更により、Oracle JDeveloper の Oracle8*i* 関連機能に影響が生じます。次の表は、Oracle JDeveloper 3.1 と、Oracle8*i*（8.1.5）および Oracle8*i* リリース 2（8.1.6）の互換性を示したものです。

JDeveloper 3.1 と Oracle8*i*（8.1.5）および Oracle8*i* リリース 2（8.1.6）の互換性

機能	8.1.5	8.1.6
JDBC	○	○
SQLJ	○	○
Oracle8*i* - JServer への Java ストアド・プロシージャの配布	○	○
Oracle8*i* - JServer への EJB の配布	×	○
Oracle8*i* - JServer への CORBA の配布	×	○
データベース・ブラウザ	○ [1]	○
リモート・デバッグ（Oracle8*i* - JServer）	×	○

1) JDBC 接続のみ

注意: この表は、Oracle JDeveloper の全機能ではなく、Oracle8*i* 関連のみの機能を対象としたものです。

Oracle8*i* パッチセット 8.1.6.1.0

特定のオペレーティング・システムで Oracle Business Components for Java を使用する場合、パッチセット 8.1.6.1.0 が必要な場合があります。このパッチセットには、バグ 1121033 および 1121178 に対する修正が含まれています。Oracle8*i* for Windows NT R8.1.6 にはこのバグの修正が含まれているため、パッチは不要です。Sun Solaris などの他のオペレーティング・システムでは、バグの修正にパッチセット 8.1.6.1.0 が必要です。

> **注意:** Sun Solaris に 8.1.6.1.0 パッチセットをインストールする前に、PATH 環境変数中で、`/usr/ucb` の前に`/usr/bin` が付いていることを確認してください。

Oracle8*i* Lite の使用制限

Connection Manager で接続を設定する際、「JDBC ドライバの選択」ドロップ・ダウン・リスト中に「Oracle Lite JDBC 4.0」が存在しますが、日本語版ではサポートされません。

また、プロジェクト・プロパティ・ダイアログの「ライブラリ」タブで「ライブラリ」ボタンをクリックすることにより表示される「使用可能な Java ライブラリ」リストには、「Oracle8i Lite」が存在します。このライブラリも日本語版ではサポートされません。

XML アプリケーション開発用の新機能

コードをカラー化した XML 編集

XML ベースのファイル、XSL スタイルシートおよび Oracle XSQL Pages を、JDeveloper 3.1 の IDE 内でカラーの構文強調表示を使用して編集できます。

XML 構文チェックのオプション

作成したプロジェクトに編集可能な XML、XSL または XSQL ファイルがある場合、ナビゲータのポップアップ・メニューから「**XML 構文のチェック**」メニューを選択して、XML 適格性を対話的にチェックできます。構文エラーはすべてメッセージ・ビューの「XML エラー」タブに表示され、可能であればエディタでエラーを含む行のエラーを含む文字位置が表示されます。

`<!DOCTYPE` を含む XML ドキュメントで、企業のファイアウォールを越えた DTD が参照されている場合、`.¥lib¥jdeveloper_ja.properties` ファイル内の 2 つのパラメータの値を次のように設定する必要があります。

```
jdeveloper.xml.XmlFileParserAddin.HttpProxyHost=ProxyServer.you.com
jdeveloper.xml.XmlFileParserAddin.HttpProxyPort=80
```

これにより、XML 構文チェックの際に正しい DTD が取得されます。

XML ベースのファイル形式以外のファイル拡張子を扱う場合、`jdeveloper_ja.properties` ファイルで指定される次のリストにファイル拡張子を追加することにより、JDeveloper で XML として認識されるファイル拡張子のセットを拡張できます。

```
jdeveloper.xml.XmlFileParserAddin.XmlFileExtensions=xml,xsl,xsql,xsd
```

この機能では、XML の適格性はチェックされますが、この時点での DTD の妥当性チェックは行われないことに注意してください。

注意: Business Components for Java のウィザードでは、Business Components for Java の XML コンポーネント定義ファイルを常に適切な XML として維持するため、またこれらのファイルは各コード・エディタのバッファで常に読取り専用であるため、これらのファイルまたはその他の XML ファイルには「**XML 構文のチェック**」オプションを使用できません。

Oracle XSQL Pages の実行/デバッグに対する IDE でのサポート

Oracle XSQL Pages の `.xsql` ファイルを含むプロジェクトに XSQL Runtime ライブラリを追加した後、右クリックで「**実行**」を選択して Oracle XSQL Servlet を起動し、プロジェクト中の任意の XSQL Pages を実行できます。カスタムの XSQL Action Handler を Java で作成した場合、その中にブレークポイントを設定し、ページ上で右クリックから「**デバッグ**」を選択してデバッグが可能です。作成した XSQL Pages は、プロジェクト・プロパティの「HTML ソース・ディレクトリ」の「HTML パス」設定で指定されたディレクトリに置く必要があります。Oracle XSQL Pages の詳細は、http://technet.oracle.com/tech/xml（英語）を参照してください。

Business Components for Java ガイド

機能ガイド

JServer における共有読取り専用メタデータの使用をサポート

JServer リリース 8.1.6 では、任意の読取り専用 Java オブジェクトを含む、メモリーキャッシュされたデータベース・スキーマ・オブジェクトへのハンドルとして `SharedDataHandle` があります。

SharedDataHandle が作成される際、参照しているすべてのオブジェクトのみでなく、指定したオブジェクト引数も、データベース内の名前付きスキーマ・オブジェクトへコピーされます。作成した SharedDataHandle は、他のデータベース Java セッションによりアクセスが可能です。

Java オブジェクト・データは、データベースの停止後にも持続します。この Java オブジェクト・データは、SharedDataHandle データ内にインスタンスが保存されるクラスが無効になるまで有効です。SharedDataHandle Java オブジェクト・データは、次のリリース（マイナー・バージョン・アップおよびパッチ・リリースも含む）のデータベース・サーバーがインストールされる際に無効になります。今後リリースされる JServer では、そのリリース以降で作成される SharedDataHandle についてはこの制限が解除される可能性があります。ただし、現行リリース（8.1.6）を含めてそれ以前のリリースの SharedDataHandle クラスで作成された SharedDataHandle の場合、この制限は解除されません。SharedDataHandle 内に格納されたオブジェクトへのアクセスは読取り専用であるため、Oracle8*i* における他のオブジェクト格納メカニズムよりも高速です。

Oracle Business Components for Java は、すべての JNDI メタデータを共有ハンドルに格納するモードで実行できます。そのため、この機能を使用するアプリケーションでは、プロジェクト内の XML ファイルを読取りおよび解析する時間が節約されます。共有ハンドルを作成する最初のユーザーが負担を負い、それ以降のアプリケーションへのコールでは同じハンドルを使用して、メタデータをロードする時間を節約します。

初期コンテキスト環境を通じた置換リストの設定機能

JServer（Oracle8*i* 8.1.6）で実行する場合、Oracle Business Components for Java アプリケーション・モジュールの起動にはコマンド行での操作を必要としませんが、Hashtable 環境内の jbo.project パラメータを設定することが可能です。このパラメータは、 Business Components for Java コンポーネントのファクトリ置換に使用するための .jpx ファイルの完全修飾名（ファイル拡張子は除く）を設定するために、アプリケーション・モジュールの初期参照/作成過程に渡されます。

findByKey を使用した部分的に NULL のキーのマッチングが可能

findByKey では、部分的に NULL のキーの組合せを持つインスタンスをマッチングできます。キーの中の NULL 値を明示的にマッチングするには、Java の null を渡す（その場合、キーのその位置に何があってもよいことを示す）のではなく、oracle.jbo.domain.NullValue ドメインのインスタンスを使用します。次に、DepartmentNumber と ManagerId という 2 つの属性を持つキーで行を決定する例を示します。

```
// Dept 10 の全従業員を検索（任意の Mgr）
findByKey(new Key(new Object[] { new Integer(10), null } ));

// Dept 10 の、Mgr7896 に属する全従業員を検索
findByKey(new Key(new Object[] { new Integer(10), null, new
Integer(7896) } ));

// これは、findByKey(new Key(new Object[] { new Integer(10), null } ));
と同じ
findByKey(new Key(new Object[] { new Integer(10) } ));

// Dept 10 でマネージャを持たない全従業員('new NullValue()')を検索
findByKey(new Key(new Object[] { new Integer(10), null, new
NullValue() } ));
```

デフォルトで一貫性モードのエンティティ Association アクセッサ

エンティティ・オブジェクトの Association アクセッサでは、デフォルトで未処理の変更および新規作成されたエンティティ・インスタンスと一貫した結果を表示します。デフォルトをオーバーライドする場合、コマンド行フラグを次のように設定します。

```
-Djbo.assoc.consistent=false
```

ディテール・ビュー・オブジェクトをエキスパート・モードに変更した後、ビュー・リンクを再定義

マスターとディテールのビュー・オブジェクトの間にビュー・リンクを作成後、ディテール・オブジェクトを通常のビュー・オブジェクトからエキスパート・モードのビュー・オブジェクトに変更した場合、変更したビュー・オブジェクトをディテールとして参照しているビュー・リンクを再定義し、ビュー・リンク・ウィザードで「完了」ボタンをクリックして、適切なビュー・リンク SQL 句を再生成する必要があります。

拡張ビュー・オブジェクトでのエンティティの使用方法をオーバーライド可能

拡張したビュー・オブジェクトにおける既存のエンティティの使用方法は、使用中の既存エンティティ・オブジェクトから直接または間接的に拡張するエンティティの使用方法でオーバーライド可能です。

注意: 「<」または「<<」をクリックして既存のエンティティの使用方法を削除することはできません。この機能の正しい実行方法は、次のとおりです。

- 「**選択済**」領域で、既存の使用方法を選択します。
- 「**使用可能**」リストにある拡張エンティティ・オブジェクトを選択し、「>」をクリックします。

オーバーライド操作を確認するメッセージが表示されます。

JServer のローカル・モードにおける Oracle Business Components for Java クライアントの実行

JServer リリース 8.1.6 内部のローカル・モードで Business Components for Java クライアント・アプリケーションを適切に実行するには、次のようにします。ここでは、J:に JDeveloper がインストールされていると仮定します。

1. scott にパブリック・シノニム作成の権限を付与します。

```
sqlplus system/manager
SQL> grant create public synonym to scott;
```

2. J:¥lib¥jbomt.zip から、テンポラリ・ディレクトリ（C:¥temp など）に、oracle¥jbo¥server¥xml¥DefaultMomContextFactory.class ファイルを解凍します。

3. -s オプションを使用してクラスを再ロードし、パブリック・シノニムを作成します。

```
c:¥> cd temp
c:¥temp> j:¥bin¥setjboenv j: 8i
c:¥temp> loadjava -u scott/tiger -v -r -s -grant PUBLIC,SYS
DefaultMomContextFactory.class
```

4. 次の問合せを実行し、シノニムが定義されているかどうかを検証します。

```
SQL> select dbms_java.longname(synonym_name)
  2  from dba_synonyms
  3  where dbms_java.longname(table_name) like
  4  '%DefaultMomContextFactory%';
```

この問合せで、oracle/jbo/server/xml/DefaultMomContextFactory が返されます。

JServer における共有メタデータの使用方法

Oracle8*i* で共有データ・ハンドルの最適化を有効にするには、次のようにします。

1. JDeveloper インストール・ディレクトリの.¥lib ディレクトリにある jbomt.zip から、使用しやすいディレクトリ（C:¥temp など）に、jboserver.properties ファイルを解凍します。C:¥temp¥oracle¥jbo¥server というディレクトリ構造が作成されます。

   注意: ディレクトリ構造を維持する必要があるため、Zip 解凍ユーティリティで oracle¥jbo¥server ディレクトリを保ってください。

2. C:¥temp¥oracle¥jbo¥server¥jboserver.properties の次の行を編集し、コメント文字（#）を削除します。

   変更前: #ActivateSharedDataHandle = false
   変更後: ActivateSharedDataHandle = true

3. JDeveloper インストール・ディレクトリの.¥lib ディレクトリにある jbomt.zip から、使用しやすいディレクトリ（C:¥temp など）に、Diagnostic.properties ファイルを解凍します。C:¥temp¥oracle¥jbo¥common というディレクトリ構造が作成されます。

4. C:¥temp¥oracle¥jbo¥common¥Diagnostic.properties の次の行を変更します。

   変更前: jbo.debugoutput=silent
   変更後: jbo.debugoutput=console

5. ディレクトリ構造を維持したまま、編集した 2 つのプロパティ・ファイルを含む非圧縮の JAR ファイルを作成します。

```
C:¥> cd temp
C:¥temp> jar cvf0 newproperties.jar oracle/jbo
```

6. loadjava を使用し、Oracle8*i* にプロパティ・ファイルを再ロードします。

```
loadjava -u scott/tiger newproperties.jar
```

7. JDeveloper 3.1 の IDE で Business Component Tester を使用し、Oracle8*i* CORBA/EJB モードでアプリケーション・モジュールに接続します。

8. JServer 内で実行されるアプリケーション・モジュールに正常に接続した後、C:¥oracle¥admin¥<*SID*>¥udump ディレクトリ（または user_dump_dest ディレクトリを指定した任意の場所）にあるデバッグ・トレース・ファイルに、一意の共有ハンドル名が書き込まれます。これは最も最近作成されたファイルである場合が多く、Shared_BC4J という文字列が含まれます。トレース・ファイルには、次のような行があります。

```
Success in creating Shared Data Handle Shared_BC4J_953870408976
```

   次のステップで参照するため、共有ハンドル名「Shared_BC4J_953870408976」を書き留めておきます。

9. C:¥temp¥oracle¥jbo¥server¥jboserver.properties を再度開いて次の行を編集し、ステップ 8 で書き留めた一意の共有データ・ハンドル名を、HandleName の後に追加します。

```
ActivateSharedDataHandle = true
HandleName = Shared_BC4J_953870655396
```

10. ディレクトリ構造を維持したまま、編集した 2 つのプロパティ・ファイルを含む非圧縮の JAR ファイルを再作成します。

```
C:¥> cd temp
C:¥temp> jar cvf0 newproperties.jar oracle/jbo
```

ステップ 6 と同様に、`loadjava` を使用して Oracle8*i* にファイルを再ロードします。

これで、今後作成するアプリケーションでは共有ハンドル機能が有効になります。

Oracle8*i* の共有ハンドルには、プロジェクトの JNDI XML メタデータが格納されます。当該アプリケーションの XML ファイルのうちいずれかが変更された場合には、新規の共有ハンドルを作成し、Oracle8*i* に新規名を再ロードする必要があります。この機能を使用することで、アプリケーション起動時間は最低でも 75%向上します。なおこの最適化機能は、作成するアプリケーションの開発がほぼ安定段階に入り、変更の頻度が少ない場合の使用に最も適しています。

注意: Oracle8*i* の共有データ用領域には制限があります。したがって作成した共有ハンドルが多くなった場合、データベースでそれ以上の共有ハンドルは作成できなくなります。この場合、データベースから手動でハンドルを削除する必要があります。

共有ハンドルを削除する API は、今後アプリケーション・モジュールを通じて提供される予定です。

今後のリリースでは、自動的に共有ハンドルを作成し、古いハンドルをデータベースから削除する機能がサポートされ、エンド・ユーザーの作業が不要になる予定です。

カスケード削除の動作

Association ウィザードの「**コンポジット Association**」チェックボックスは、データベース表定義の外部キーに対する `ON DELETE CASCADE` 制約に対応しています。ただし実行時にエンティティで `remove()` がコールされた場合に、Business Components for Java では“キャッシュされコンポジット化された子”のすべてが自動的に削除されるわけではありません。デフォルトでは、ユーザー（またはアプリケーション）が親を削除する前に子を削除していない場合、`remove()` により例外がスローされます。

実行時のカスケード削除機能は、プログラム的にインプリメント可能です。`remove()` のコール時にアプリケーションで“メモリー内カスケード削除”を実行する場合、次のようにして、コンポジット Association の親である `EntityImpl` サブクラスのコードをオーバーライドできます。

親エンティティの remove() のオーバーライド

コンポジット Association に存在するすべてのインスタンスを再帰的に削除するように、remove() メソッドをインプリメントします。デフォルトのインプリメンテーションでは、this インスタンスのみ削除されます。

```
public void remove()
{
  // 各コンポジット Association について、子を削除
  RowIterator r1 = getComposedEntity1();
  {
    while (r1.hasNext())
    {
      r1.next().remove();
    }
  }
  RowIterator r2 = getComposedEntity2();
  {
    while (r2.hasNext())
    {
      r2.next().remove();
    }
  }
  // this インスタンスを削除
  super.remove();
}
```

親エンティティの postChanges() のオーバーライド

postChanges() のデフォルトのインプリメンテーションでは、すべてのコンポジットの使用済の子メンバーに postChanges() を（再帰的に）コールすることにより、ポストが必要なすべてのインスタンスを走査し、次に独自の doDML メソッドをコールして DELETE 文を生成します。

子の行に対する明示的な DELETE 文の生成を回避するには、postChanges() をオーバーライドする必要があります。コンポジットの一致するデータベース外部キー関係に ON DELETE CASCADE がある場合は、親に対する DELETE 文のみで削除できます。

```
/**
 *カスタム DML の UPDATE/INSERT/DELETE ロジックをここに記述
 */
public void postChanges(TransactionEvent e)
{
  if (getPostState() == STATUS_DELETED)
```

```
  {
  // DELETED メソッドに対する super.postChanges(e)がすべての子に
  // postChanges をコール。データベースに ON DELETE CASCADE がある場合、
  // この行に対する DELETE 文のみ実行
  // そのため super.postChanges をバイパス

    doDML(DML_DELETE, e);
  }
  else
  {
    super.postChanges(e);
  }
}
```

今後のリリースでは、実行時のカスケード削除機能は Oracle Business Components for Java によりサポートされる予定です。

JDeveloper のビルトイン・サーブレット/JSP WebServer を使用したサーブレット初期化パラメータの設定

サーブレットの初期化パラメータは、構成ファイルで設定可能です。これは webtogo.ora という名前のファイルで、次の場所にあります。

```
C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥lib¥webtogo.ora
```

このファイルの[SERVLET_PARAMETERS]という領域に、サーブレット・パラメータを追加できます。たとえば（webtogo.ora ファイルで）次のようにします。

```
[SERVLET_PARAMETERS]
My_Init_Parameter=My_Value
.
.
.
```

この値を取得するには、サーブレットで次のようなコードを使用します。

```
// 値を取得するグローバル文字列
String my_init_val;

// getInitParameter コールでオーバーライドされる初期メソッド
public void init(ServletConfig config) throws ServletException
{
  super.init(config);
  my_init_val = getInitParameter("My_Init_Parameter");
}
```

注意: これらのパラメータは、システム共通です。また、この機能は JDeveloper の今後のリリースで変更される場合があります。

JDeveloper 3.1 と JDK 1.3

JDeveloper 3.1 には、JDK の切換え機能があります。この機能により、Java Development Kit の新しいバージョンを IDE に追加することが可能です。JDeveloper リリース 3.1 には、JDK 1.2.2（デフォルト）および JDK 1.1.8 が含まれています。現在 JavaSoft では JDK 1.3 がリリースされています。JDK 1.3 は JDeveloper には含まれませんが、http://java.sun.com/products/jdk/1.3/ja/から無料でダウンロード可能です。

注意: JDK 1.3 には JavaDoc が含まれません。このドキュメントは http://java.sun.com/j2se/1.3/ja/docs.html から無料でダウンロードできます。JDK 1.3 のホーム・ディレクトリに JavaDoc を解凍することをお薦めします。Java2 SDK ドキュメントの詳細は、JDK 1.3 のリリース・ノートを参照してください。

JDK 1.3 のインストール

1. JDK 1.3 を、`C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥jdk1.3` のディレクトリにインストールします。
2. `C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥java1.2¥jre¥bin` ディレクトリから、`ojvm` ディレクトリと、`fdebug.exe` および `fdebug_g.exe` の 2 ファイルを、`C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥jdk1.3¥jre¥bin` にコピーします。
3. `C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥jdk1.3¥jre¥lib¥jvm.cfg` ファイルを編集します。このファイルには、このリリースでサポートされる Java 仮想マシン（JVM）がリストされています。JDK 1.3 で Oracle JVM を実行させるには、次の行を追加します。

   ```
   -ojvm
   ```

   注意: `jvm.cfg` ファイルでの各 JVM の順序は重要です。リストの 1 番目の JVM がデフォルトの JVM になります。

ステップ 2 および 3 を自動的に実行するために、`InstallOJVM.bat` というバッチ・ファイルが用意されています。これは、`C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥bin` ディレクトリ内にあります。

JDK 1.3 を使用するための JDeveloper 3.1 の設定

1. JDeveloper を起動し、「**ツール**」メニューから「**デフォルトのプロジェクト・プロパティ**」を選択します。「**ターゲットの JDK バージョン**」プルダウン・メニュー・ボックスの隣にある「**定義**」ボタンをクリックします。
2. 「**使用可能な JDK のバージョン**」ダイアログが表示されます。「**新規**」ボタンをクリックし、次に「**...**」ボタンをクリックして、JDK 1.3 の `java.exe` を検索するか、または JDK 1.3 のディレクトリ内の `java.exe` までのパスを入力します。
3. `java.exe` へのパスを選択すると、「**クラス・パス**」および「**ソース・パス**」テキストボックスのパスは自動的に決定されます。JavaDoc をダウンロードし、インストールした場合、「**ドキュメント・パス**」も自動的に生成されます。「使用可能な JDK のバージョン」リストに JDK 1.3 が追加されます。
4. 「**OK**」をクリックし、ダイアログを閉じます。
5. 「**ターゲットの JDK バージョン**」プルダウン・メニュー・ボックスから JDK 1.3 を選択して「**OK**」をクリックし、変更した「**デフォルトのプロジェクト・プロパティ**」を確定します。

これで、JDeveloper 3.1 を使用して Java アプリケーションを開発する際に JDK 1.3 が使用可能になります。新規プロジェクトにデフォルトで JDK 1.3 を使用する場合は、「**デフォルトのプロジェクト・プロパティ**」ダイアログの「**ターゲットの JDK バージョン**」プルダウン・メニュー・ボックスで、それが選択されていることを確認してください。既存プロジェクトの場合、JDK 1.3 を使用する際に「**ターゲットの JDK バージョン**」を変更する必要があります。

JDeveloper 3.1 用 Java プラットフォームとしての JDK 1.3 の使用方法

JDeveloper 3.1 の一部は Java でプログラムされているため、それ自体に新しい JDK 1.3 を利用することも可能です。これには次の 3 つの方法があります。

- Windows NT のコマンド・プロンプトを使用し、次のコマンドで JDeveloper 3.1 を起動します。

  ```
  C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥bin¥jdeveloper -jdk=1.3.0
  ```

- 「**スタート**」メニューに、`jdeveloper.exe` への新規ショートカットを作成します。ショートカットを右クリックし、「**プロパティ**」ダイアログの「ショートカット」タブで、「**リンク先:**」テキストボックスに次のように入力します。

```
"C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥bin¥jdeveloper" "-
jdk=1.3.0"
```

- JDeveloper の初期化ファイルを、次のようにして変更します。

  1. JDeveloper 3.1 を終了します。
  2. `C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥bin¥jdeveloper.ini` ファイルを開きます。
  3. `[Java_2]`のセクションまでスクロールします。`JDK` のパラメータを、次のように置き換えます。

     変更前: `JDK = java version "JDK 1.2.2_JDeveloper"`
     変更後: `JDK = java version "1.3.0"`
  4. ファイルを保存します。
  5. JDeveloper の次の起動時には、Java プラットフォームとして JDK 1.3 が使用されるようになります。

変更が有効かどうかを確認するには、JDeveloper 3.1 を起動して「**ヘルプ**」→「**バージョン情報**」を選択します。アイコンの下に、次のように表示されます。

```
1.3.0
OJVM
```

国際化（I18N）の問題

ビューアで HTML および JSP ファイルを表示した場合、'□'が現われる（1325251）

JDeveloper 3.1 内のビューアで HTML ファイルおよび JSP ファイルを表示させると、HTML タグ内の日本語が'□'で表示されます。単体で作成した場合の HTML ファイル、JSP ファイル、プロジェクト・ウィザードで「プロジェクトの HTML ファイルを生成」チェックボックスを選択した場合に生成される HTML ファイル、ビジネス・コンポーネント JSP アプリケーション・ウィザードで生成される `description.html`、`main.jsp` などのファイルのデフォルトの内容でこの現象が確認されています。これらのファイルを Web ブラウザで表示させた場合は、問題ありません。

JSP アプリケーションのランタイムで、ボタンおよびいくつかの文字列が翻訳されていない（1326494）

ビジネス・コンポーネント JSP アプリケーション・ウィザードを使用して作成した JSP アプリケーションを実行した際、表示されるボタン・ラベルやリンク文字列が翻訳されていません。これは、サード・パーティの JSUI ベース・コンポーネントを使用しているためです。今後のリリースでは JSUI ベース・コンポーネントは使用されず、この問題は修正される予定です。

英語以外のクラス名、JSP 名およびサーブレット名

JDK 1.1.X を使用する場合、英語以外のクラス名、JSP 名およびサーブレット名は `ClassNotFound` 例外の原因となります。JDK 1.2.X を使用する場合は、main メソッドを含むクラス名、JSP 名およびサーブレット名が英語以外の場合のみこの問題が発生します。

IE 5.0 を使用する場合、マルチバイト文字を使用したタグは JSP Runtime でサポートされない（1046818）

Microsoft Internet Explorer 5.0 を使用する場合、ハイパーリンクにマルチバイト文字が含まれていると JSP Web アプリケーションが中断します。

Oracle Java Education について

「**ヘルプ**」メニューから「**Oracle Java Education**」を選択した場合に表示される「Java Training for Oracle Professionals」ページの内容は、米国オラクル社の研修プログラムです。日本における研修プログラムとは異なりますので、あらかじめご了承ください。

リリース 3.1 での既知の問題

エンティティ属性の「新規の間」更新可能の設定（1254735）

エンティティ属性の「新規の間」更新可能の設定は、3 層モードでの実行時には正常に機能しません。現在、3 層モードでの実行時には属性は更新不可能です。

このサポートは、今後のリリースで予定されています。

回避方法:　エンティティ属性に「新規の間」を使用せずに、「常に」を使用してください。

アプリケーションの main メソッドが public であることが必要

JDeveloper 3.1 では、アプリケーションの main メソッドが public かつ static void でなければならないという規則を新たに設けました。これは Java の公式仕様です。JDeveloper 2.0 ではアクセス・フラグが無視されるため、アクセス修飾子とは無関係に `static void main(String[])` メソッドを含むクラスの実行が可能でした。

これと同じように作成されたコードは、JDeveloper 3.1 では実行できません。

Web Object Manager により、サブクラス化された既存の Bean コードが上書きされる（1252827）

Web Object Manager を使用して既存の Web Bean をサブクラス化する場合、`<web_bean>SubClass.java` というデフォルトの Java ファイルが作成され、プロジェクトに追加されます。（生成される前にこのファイル名は編集できますが、デフォルトは `<web_bean>SubClass.java` です。）

その後同じ Web Bean を再度サブクラス化した場合、Web Object Manager は、もとのファイルでの変更の有無にかかわらず、最初に作成された`<web_bean>SubClass.java` ファイルをオーバーライドします。

> **注意:** 同じ Web Bean を 2 回以上サブクラス化する場合には、生成される Java ファイルの名前がプロジェクト内の既存ファイルと異なっていることを必ず確認してください。

拡張アプリケーション・モジュール（1252123）

現在、拡張アプリケーション・モジュールは、リリース 8.1.6 のデータベースに EJB として配布できません。このサポートは、今後のリリースのデータベースで予定されています。

回避方法:　拡張アプリケーション・モジュールを作成せず、オリジナルとして同じビュー・オブジェクトを持つ新規のアプリケーション・モジュールを作成してください。

JSP プロパティ・ファイル:　ConnectionName プロパティに VisiBroker に関するコメントが必要（1232958）

JSP プロパティ・ファイル `app.properties` の `ConnectionMode` に関するコメントには、パラメータになり得る値が `8i`、`EJB` または `local` となっています。ビジネス・コンポーネント JSP アプリケーション・ウィザードを使用して作成した JSP ファイル群を VisiBroker

または Oracle Application Server に配布済のアプリケーション・モジュールにバインドする場合、次の文字列を ConnectionMode のパラメータに使用する必要があります。

- VisiBroker の場合、VB_BINDING、VB_COLOCATE または VB_NAMING
- Oracle Application Server の場合、OAS_EJB

JSP リモート・デバッグ

ローカル・デバッグに使用したプロジェクトで JSP に対するリモート・デバッグを実行する場合、リモート配布用のパッケージはローカルで使用したパッケージと一致する必要があります。つまり、JDeveloper の myhtml ディレクトリ（デフォルトでは、MyProject4_html などのディレクトリ名が普通）にあるサブディレクトリと同名のサブディレクトリを、Web サーバーの html ディレクトリに作成し、JSP ファイルはこのサブディレクトリにコピーする必要があります。この場合 JSP の URL は、http://host/MyProject4_html/test.jsp などとなります。

ローカル・パッケージと一致しないディレクトリに JSP ファイルを配布する場合は、リモート・デバッグ用に新規のプロジェクトを作成する必要があります。

JSP のデバッグ（1229083）

JSP を（ローカルまたはリモートで）デバッグする場合、式の評価を使用するデバッグ機能はすべて使用できません。これには条件ブレークポイント、ブレークポイント発生の式のロギングおよび「**実行**」メニューからの「**評価/変更**」ダイアログも含まれます。

JServer で実行中のユーザーに対する Java2 権限の付与

BC4J コンポーネントを実行するユーザーには、特定の Java2 スタイルの権限を付与する必要があります。SCOTT などのユーザーにこれらの権限を付与するために、次の SQL スクリプトを実行できます。

```
SET VERIFY OFF
EXEC DBMS_JAVA.GRANT_PERMISSION('SCOTT',
  'SYS:java.util.PropertyPermission', '*', 'write');
EXEC DBMS_JAVA.GRANT_PERMISSION('SCOTT',
  'SYS:java.util.PropertyPermission', '*', 'read');
EXEC DBMS_JAVA.GRANT_PERMISSION('SCOTT',
  'SYS:java.lang.RuntimePermission', 'createClassLoader', null);
EXEC DBMS_JAVA.GRANT_PERMISSION('SCOTT',
  'SYS:java.lang.RuntimePermission', 'setContextClassLoader', null);
COMMIT;
```

```
SET VERIFY ON
EXIT;
```

注意: DBMS_JAVA.GRANT_PERMISIION プロシージャの第 1 パラメータでユーザー名を指定する場合、必ずすべて大文字を使用してください。

ビジネス・コンポーネント JSP アプリケーションの実行時におけるイメージ・ファイルの不足（1217046）

まず「**default**」テンプレートを生成し、それから「**Oracle**」テンプレートを生成する場合、ImageBase が webapp¥cabo¥images に変更されるため、一部のイメージ・ファイルは cabo¥images ディレクトリからなくなります。これは予期される動作です。両方のテンプレートを同時に使用するアプリケーション・モジュールの実行は、サポートされません。これは、同じテーマ・イメージ・ディレクトリを使用するとみなされるためです。有効なテーマは、最後に選択されたものです。この動作は、プロパティ・ファイルに格納されているテーマ・イメージ・ディレクトリにより発生します。

Java Web Server にデータベース・サーブレットを配布する際の 404 エラー（1221312）

次のようなエラーが発生する場合があります。

```
404 error : "The file that you requested could not be found on this server.
If you provided the URL, please check to ensure that it is correct. If
you followed a hypermedia link, please notify the administrator of that
server of this error. "
```

この場合、initializeDbServlet() メソッド中のサーブレット名を確認してください。URL は、servlets/MyMDServlet ではなく、servlet/MyMDServlet である必要があります。JWS では/servlet/*servlet name* を使用します。

Oracle スタイルのバインド変数

Business Components for Java に影響のある JDBC の問題点は、次のとおりです。WHERE 句パラメータは、JDBC の PreparedStatement に対し（インデックスを 1 ずつ増分して）setObject または setNull をコールします。次のプログラム例は、Oracle JDBC ドライバが数の一致ではなく位置の一致によってこれらの setObject コールを受け取ることを示すものです。

```
// プログラム例
// ======================================

import java.sql.*;
import java.io.*;
import oracle.sql.*;

public class bind
{
  Connection conn;

  public static void main(String argv[])
  {
    bind ex = new bind();

    try
    {
      DriverManager.registerDriver(new
          oracle.jdbc.driver.OracleDriver());

      ex.conn = DriverManager.getConnection("jdbc:oracle:thin:scott/
          tiger@localhost:1521:ORCL");

      ex.work();
      ex.work2();
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
  }

  public void work() throws Exception
  {
    PreparedStatement stat;

    stat = conn.prepareStatement("select * from emp e where ENAME
        = :2 or JOB = :1");

    stat.setObject(1, "KING");
    stat.setObject(2, "PRESIDENT");

    ResultSet rslt = stat.executeQuery();

    if (rslt.next())
    {
      System.out.println("name:" + rslt.getString("ENAME"));
    }
    else
    {
      System.out.println("no row found");
```

```
    }
  }

  public void work2() throws Exception
  {
    PreparedStatement stat;

    stat = conn.prepareStatement("select * from emp e where ENAME
        = :2 or JOB = :1");

    stat.setObject(1, "PRESIDENT");
    stat.setObject(2, "KING");

    ResultSet rslt = stat.executeQuery();

    if (rslt.next())
    {
      System.out.println("name:" + rslt.getString("ENAME"));
    }
    else
    {
      System.out.println("no row found");
    }
  }
}
```

このプログラムを実行すると、まず work が実行され、次に work2 が実行されます。work() により :2 に setObject index 1 がバインドされ、:1 に setObject index 2 がバインドされます。work2() は、このバインドを逆転します。

実行の結果は次のようになります。

```
name:KING
no row found
```

work2() で行が検出されなかったことから、このバインドは位置により実行されたことがわかります。

8.0 を使用する場合、BLOB 列が JBO でサポートされない（804754）

JBO で BLOB 列をサポートさせる場合、JDBC ドライバ 8.1.X および Net8 8.1.X が必要です。

JDBC Thin ドライバと Net8 名前/値リストを使用する場合、Oracle8*i* への配布が不完全（1250825）

Connection Manager では、接続方法として Oracle JDBC Thin ドライバおよび Net8 名前/値ペアを使用した Oracle8*i* への接続を作成することが可能です。この接続はデータベース・ブラウザで表示できますが、Oracle JDBC Thin ドライバを使用する場合には loadjava ユーティリティで"`@host:lport:SID`"という形式のデータベース文字列が必要であるため、配布が不完全になります。

この問題を回避するには、次の 2 つの方法があります。

- Connection Manager でこの接続タイプを作成する際、Net8 の「名前/値ペア」ではなく「名前付きホスト」接続を使用します。
- Oracle JDBC Thin ドライバではなく、Oracle JDBC OCI-8 ドライバを使用します。JDeveloper で Oracle JDBC OCI-8 ドライバを使用する場合、次の「JDK 1.2/1.3 とネイティブ・コード」を参照してください。

JDK 1.2/1.3 とネイティブ・コード

JDeveloper で Oracle JDBC OCI ドライバを使用する場合（JDK は 1.2 または 1.3）、`bin¥jdeveloper.ini` ファイルで、`java.library.path` および `sun.boot.library.path` に`%ORACLE_HOME%¥bin` ディレクトリを追加してください。これは次のようになります。

```
[Java_2]
JDK=java version "JDK1.2.2_JDeveloper"
Java2VM=OJVM
JLP=-Djava.library.path=.;%JAVA_ROOT%¥bin;%JAVA_ROOT%¥jre¥bin;
    C:¥ORANT¥BIN
SLP=-Dsun.boot.library.path=.;%JAVA_ROOT%¥bin;%JAVA_ROOT%¥jre¥bin;
    C:¥ORANT¥BIN
```

JDK 1.2/1.3 の JDeveloper で、ネイティブ・ライブラリ（DLL ファイル）をコールする際に JNI を使用するアプリケーションを実行する場合、プロジェクト・プロパティを編集する必要があります。「実行/デバッグ」タブの「Java VM パラメータ」に次のような設定を追加してください。

```
-Djava.library.path="<Path where DLL files are located>"
```

JBCL の再統合（990533）

JBCL の再統合に必要なステップは、「**Deprecated Features**」→「**Building Data Forms Using JBCL**」→「**Restoring JBCL Functionality**」の項に記載されています。

この項に記載されていない、次の追加のステップが必要です。

JDeveloper を終了し、lib¥jdeveloper.properties ファイルを開き、次の行からコメント文字（#）を削除します。

```
#jdeveloper.addin.14=borland.jbuilder.dmdesigner.DMdesignerAddin
```

Netscape で JSP Web アプリケーションを実行した場合に発生するエラー（1037092）

Netscape で JSP Web アプリケーションを実行する際、次のようなエラーが発生することがあります。

```
Can't find the file "C:¥Program Files¥Oracle¥JDeveloper 3.1.1.2¥
   myprojects¥WebAppRunner.html" (or one of its component).
```

これは Windows NT エクスプローラの問題です。この問題を改善するには、Windows NT エクスプローラ「**表示**」→「**オプション**」を開きます。「**ファイルタイプ**」タブをクリックします。「登録されているファイルタイプ」で「Netscape Hypertext Document」（または htm/html 拡張子が関連付けられているアプリケーション）を選択し、「**編集...**」ボタンをクリックします。「**ファイルタイプの編集**」ダイアログで、「**アクション:**」フィールドの「**open**」を選択し、「**編集**」ボタンをクリックします。「**アクションの編集**」ダイアログで、「**DDE を使う**」チェック・ボックスのチェックを外します。

すべてのダイアログで「**OK**」をクリックし、ダイアログを閉じます。これで問題が解決されます。

jdeveloper.ini ファイルは、論理ドライブ情報ではなく物理ドライブ情報を保存（829112）

インストールした JDeveloper へのデスクトップ・ショートカットを代替ドライブ上に作成した場合、このショートカットは JDeveloper の起動時に Windows NT により変更されます。これは Windows NT の標準的な動作です。このためハードコードされたドライブ名ではなく、物理リンクに関連付けられた JDeveloper 初期化ファイルの一部に影響が生じることがあります。

ビジネス・コンポーネント Java フォームの問題

複数の接続タイプ

Java フォームについて定義する接続タイプは、アプリケーションで指定するすべての SessionInfo コンポーネントの接続タイプと同じである必要があります。

たとえばアプリケーションでは、2 つのインスタンスで Oracle8*i* と VisiBroker に別々に接続することはできません。データベース対応アプリケーションは常に、パスに最初に出現するクライアント・ライブラリを検索し、すべての SessionInfo コンポーネントが同じライブラリを使用すると仮定するためです。複数の接続タイプをサポートするために複数のクライアント・ライブラリを導入する方法は、このリリースではサポートされません。

InfoSwing の TreeControl に関する制限

InfoSwing の TreeControl は、主キー属性を持つビュー・オブジェクトに基づいている必要があります。TreeControl は、そのデータ・ソースとして使用されるビュー・オブジェクト間の関係を決定する際、ビジネス・コンポーネント・アプリケーション・モジュールに依存します。

「現行プロジェクト」オプションを使用して Java フォーム用のビジネス・コンポーネント・アプリケーション・モジュールを定義した場合、そのプロジェクトでは InfoSwing の TreeControl は使用できません。ビジネス・コンポーネント・データ・フォーム・ウィザードでは、データ・フォーム自体と同じプロジェクト内にビジネス・コンポーネントを作成する場合には、エンティティ・オブジェクトおよび Association のみを定義します。ビュー・オブジェクトおよびアプリケーション・モジュールは実行時に動的に作成されるため、設計時には TreeControl の基になるビュー・オブジェクトがありません。

また、InfoSwing の TreeControl には既知のバグ（1054195）があります。第 1 レベルでバインドされた内部結合ビュー・オブジェクトでは、ルート・ノードのセットからの行の移動または削除（外部キー属性の設定を null にまたは null から変更する）が、正常に機能しません。

「Java の生成」オプションにより、配布済の Java ストアド・プロシージャに発生するエラー（1100083）

JPublisher ウィザードでは、配布済 Java ストアド・プロシージャの Java ファイルが生成されません。

表型のオブジェクトに対し、Java の生成が不可能（1161068）

表型のオブジェクトに対する Java コードを生成しようとした場合、次のエラーが返されます。

```
WARNING:You requested SQLData classes that may not be portable.
```

EJB トランザクションは 60 秒でタイムアウト（1248616）

EJB トランザクションは 60 秒でタイムアウトします。

BC4J を EJB として配布する際、常に JBO ランタイムの配布を促す（1347800）

Business Components for Java アプリケーション内の.jpx ファイルを右クリックし、「ビジネス・コンポーネントの配布...」を選択して EJB オプションで配布する際、設定が正しいにもかかわらず、常に「Business Components for Java ランタイム・フレームワークが配布されていません。ただちに配布しますか?」というメッセージが表示されます。