Forte™ for Java™, Community Edition チュートリアル

Forte for Java, Community Edition, 3.0

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 U.S.A.
650-960-1300

Part No. 816-2847-01
2001 年 10 月 , Revision A

原典： *Forte for Java, Community Edition Tutorial*
Part No: 816-1408-10
Revision A

Please
Recycle

Adobe PostScript

目次

図目次

表目次

はじめに

『Forte™ for Java™, Community Edition チュートリアル』へようこそ。このチュートリアルでは、Forte for Java, Community Edition に導入された機能の使用方法について説明します。Community Edition に導入された機能とは、Java サーブレットと JavaServer Pages™ 技術を使用する Web アプリケーションと、Forte for Java のカスタムタグライブラリおよび透過的な持続性 (Transparent Persistence: TP) を使用したデータベースアクセスをサポートする機能です。

このマニュアルで説明する作業は、次のプラットフォームおよびオペレーティングシステムで実行できます。

- Solaris™ 8 (SPARC™ プラットフォーム版)
- Microsoft Windows 2000, SP2
- Microsoft Windows NT 4.0, SP6
- Red Hat Linux 6.2

このマニュアルに掲載している画面イメージは、すべて Windows 版の Forte for Java ソフトウェアのものです。他のプラットフォームを使用する場合でも、表示上の違いはわずかであるため、内容を理解するのには問題ありません。ほとんどの手順で Forte for Java のユーザーインタフェースを使用しますが、場合によってはコマンド行にコマンドを入力する必要があります。その場合は、次のように、Microsoft Windows の「コマンドプロンプトウィンドウ」でのプロンプトと構文が例として示されています。

```
C:¥>cd MyWorkDir¥MyPackage
```

UNIX® や Linux 環境では、次のようなプロンプトとなり、¥ マーク (またはバックスラッシュ) ではなくスラッシュを使用します。

```
% cd MyWorkDir/MyPackage
```

お読みになる前に

このチュートリアルでは、データベースとやりとりし、生成したコンテンツを動的に表示する単純な Web アプリケーションを作成します。このアプリケーションの設計と構造は、Java 2 Platform, Enterprise Edition Blueprints リソースに準拠しています。このチュートリアルは、Forte for Java, Community Edition 機能の使用方法や、Web アプリケーションのコンポーネントの構築方法を学ぶ場合に役に立ちます。次の知識があることを前提としています。

- Java プログラミング言語
- Java サーブレット構文
- JDBC™ 対応のドライバ構文
- JavaServer Pages 構文
- HTML 構文
- リレーショナルデータベースの概念 (表やキーについて)
- 使用するデータベースの使用方法

このチュートリアルでは、次に示すような J2EE の概念に関する一般的な知識が必要です。

- *Java 2 Platform, Enterprise Edition Blueprints*
  `www.java.sun.com/j2ee/blueprints`
- *Java 2 Platform Enterprise Edition Specification*
  `www.java.sun.com/products`
- *Java 2 Enterprise Edition Developer's Guide*
  `www.java.sun.com/j2ee/j2sdkee/devguide1_2_1.pdf`
- *Java Servlet Specification, v2.2*
  `www.java.sun.com/products/servlet/index.html`
- *JavaServer Pages Specification*, v1.1
  `www.java.sun.com/products/jsp/index.html`

本書の構成

このマニュアルは、初めから順を追って読むことを前提に作成されています。チュートリアルの各章は、前の章で作成したコードに基づいて構成されています。

第 1 章では、チュートリアルアプリケーションのソフトウェア要件、チュートリアルデータベース表のインストール方法、Forte for Java 開発環境の起動方法について説明します。また、インストールされる Forte for Java ディレクトリの一覧と説明も記載されています。

第 2 章 では、チュートリアルアプリケーションの構造について説明します。

第 3 章 では、チュートリアルアプリケーションの作成方法を順を追って説明します。このチュートリアルアプリケーションは、ショッピングカートを使用して音楽用 CD をオンラインで購入する単純なアプリケーションです。

第 4 章 では、データベースに顧客の注文データの書き込みができるようにチュートリアルアプリケーションを拡張する方法を説明します。データベースへの書き込みには透過的な持続性 (Transparent Persistence: TP) 技術を使用します。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P-1　表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`system% su` `Password:`
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには `rm` *filename* と入力します。

表 P-1　表記上の規則 (続き)

字体または記号	意味	例
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `  XV_VERSION_STRING'`

コード例は次のように表示されます。

■ C シェルプロンプト

```
system% command y|n [filename]
```

■ Bourne シェルおよび Korn シェルのプロンプト

```
system$ command y|n [filename]
```

■ スーパーユーザーのプロンプト

```
system# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しtています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

プラス記号 (+) は 2 つのキーを同時に押すことを示します。たとえば Ctrl+D は Control キーを押したまま D キーを押すことを意味します。

関連マニュアル

Forte for Java のマニュアルは、Acrobat Reader (PDF) ファイル、オンラインヘルプ、サンプルアプリケーションの `Readme` ファイル、Javadoc™ 文書の形式で提供しています。

オンラインで入手可能なマニュアル

次のマニュアルは、Forte for Java のポータルサイト、`docs.sun.com` の Web サイト、およびインターネットオンラインブックストアの Fatbrain.com から入手することができます。

示すマニュアルは、Forte for Java ポータル、`docs.sun.com` の Web サイト、およびインターネット上の書店 Fatbrain.com から入手できます。

次のマニュアルは、Forte for Java のポータルサイト、`docs.sun.com` の Web サイト、およびインターネットオンラインブックストアの Fatbrain.com から入手することができます。

Forte for Java ポータルサイトでのマニュアルの入手先は、`http://www.sun.co.jp/forte/ffj/documentation/index.html` です。`docs.sun.com` の URL は、`http://docs.sun.com` です。Fatbrain.com の URL は、`http://www.fatbrain.com/documentation/sun` です。

- リリースノート (HTML 形式)

  Forte for Java の Edition ごとに用意されています。このリリースでの変更情報と技術上の注意事項を説明しています。

- インストールガイド (PDF 形式)

  Forte for Java の Edition ごとに用意されています。対応プラットフォームへの Forte for Java のインストール手順を説明しています。さらに、システム要件、アップグレード方法、Web サーバーやアプリケーションサーバーのインストール、コマンド行での操作、インストールされるサブディレクトリ、Javadoc の設定、データベースの統合、アップデートセンターの使用方法などが含まれます。

- Forte for Java プログラミングシリーズ (PDF 形式)

  Forte for Java の各機能を使用して優れた J2EE アプリケーションを開発するための方法を詳細に説明しています。

  - 『Web コンポーネントのプログラミング』 Part No. 816-2849-01

    JSP ページ、サーブレット、タグライブラリを使用し、クラスやファイルをサポートする Web アプリケーションを J2EE Web モジュールとして構築する方法を説明しています。

  - 『持続プログラミング』 Part No. 816-2850-01

    Forte for Java が提供するさまざまな持続性プログラミングモデルのサポート機能について説明しています。特に、JDBC と透過的な持続性についてを詳細に説明しています。

  - 『Enterprise JavaBeans コンポーネントのプログラミング』 Part No. 816-2845-01

    Forte for Java の EJB ビルダーウィザードや、その他のグラフィカルユーザーインタフェースを使用し、Enterprise JavaBeans コンポーネント (コンテナ管理や Bean 管理の持続性の機能を持つセッション Bean やエンティティ Bean) を作成する方法を説明しています。

  - 『Web サービスのプログラミング』 Part No. 816-2844-01

    Web サービスモジュールが提供するツールを使用して Web サービスを構築する方法を説明しています。Web サービスは、XML (Extensible Markup Language) 文書の形式で提供されるアプリケーションビジネスサービスであり、HTTP を介して配信されます。

  - 『XML データサービス用 JSP のプログラミング』 Part No. 816-2843-01

    Forte for Java Enterprise Service Presentation Toolkit (Forte ESP ツールキット) を使用し、HTML に動的 XML データを組み込む方法を説明しています。

  - 『J2EE モジュールおよびアプリケーションのアセンブルと実行』 Part No. 816-2846-01

    EJB モジュールと Web モジュールを組み合わせて J2EE アプリケーションを作成する方法と、J2EE アプリケーションを配備して実行する方法を説明しています。

- Forte for Java チュートリアル (PDF 形式)

  チュートリアルアプリケーションは、ユーザー設定ディレクトリの下の `sampledir/tutorial` ディレクトリにあります。

  - 『Forte for Java, Community Edition チュートリアル』 Part No. 816-2847-01 (このマニュアル)

    Forte for Java, Community Edition のツールを使用し、簡単な J2EE Web アプリケーションを作成する方法を順を追って説明しています。

  - 『Forte for Java, Enterprise Edition チュートリアル』 Part No. 816-2848-01

    Enterprise JavaBeans コンポーネント、アプリケーションテスト機能、Forte for Java Web サービス技術を使用し、アプリケーションを作成する方法を順を追って説明しています。

オンラインヘルプ

オンラインヘルプは、Forte for Java 開発環境内から参照できます。ヘルプキー (Solaris オペレーティング環境では Help キー、Windows および Linux 環境では F1 キー) を押すか、「ヘルプ」 > 「内容」を選択します。ヘルプの項目と検索機能が表示されます。

プログラム例

Forte for Java の機能を紹介したプログラム例が、関連する `Readme` ファイルとともに、ユーザー設定ディレクトリの `sampledir/examples` ディレクトリに置かれています。また、Forte for Java のポータルサイトから、Enterprise Edition に固有のサンプルファイルをダウンロードし、それらを `sampledir/examples` ディレクトリに置くこともできます。チュートリアルアプリケーション (本書および『Forte for Java, Enterprise Edition チュートリアル』で説明されているアプリケーションを含む) はすべて、`sampledir/tutorial` ディレクトリに置かれています。

Javadoc

Javadoc 形式のマニュアルは、Forte for Java の多くのモジュールに用意されており、IDE の中で参照できます。このマニュアルの使用方法については、リリースノートを参照してください。IDE を起動すると、エクスプローラの Javadoc タブで Javadoc マニュアルを参照できます。

Sun のマニュアルのオンラインでの提供

Sun の各種システムのマニュアルを、次の Web サイトで提供しています。

`http://www.sun.com/products-n-solutions/hardware/docs`

Solaris のマニュアルセットとその他の多くのマニュアルを、次の Web サイトで提供しています。

`http://docs.sun.com`

Sun のマニュアルの注文方法

Sun の製品マニュアルは、Fatbrain.com インターネットブックストアを通じて米国 Sun Microsystems, Inc. に直接注文できます。Fatbrain.com の Sun Documentation Center へは次の URL でアクセスできます。

`http://www.fatbrain.com/documentation/sun`

ご意見の送付先

Sun のマニュアルについてのご意見やご要望をお寄せください。今後のマニュアル作成の参考にさせていただきます。次のアドレスまで電子メールをお送りください。

`docfeedback@sun.com`

電子メールのタイトルに、対象マニュアルの Part No. (このマニュアルの場合は 816-2847-01) を明記してください。

第1章

チュートリアルを開始する前に

この章では、Forte for Java, Community Edition のチュートリアルを開始する前に必要な作業について説明します。インストールの詳細については、『Forte for Java, Community Edition インストールガイド』で説明していますが、ここでもその一部をとりあげています。この章の内容は次のとおりです。

チュートリアルアプリケーションのソフトウェア要件

この節では、Forte for Java, Community Edition のチュートリアルを開始する前にシステムを準備する方法について説明します。Forte for Java 統合開発環境 (IDE) の実行に必要なコンポーネントだけでなく、チュートリアルアプリケーションの作成と実行に必要なコンポーネントがすべてそろっているかどうかを確認します。

一般的なシステム要件については、リリースノートまたは Forte for Java のポータル `http://www.sun.co.jp/forte/ffj/index.html` から参照できます。

Forte for Java IDE の実行に必要なコンポーネント

Forte for Java IDE では、Java Development Kit (JDK™) が必要です。IDE をインストールすると、インストーラがユーザーのシステム内で JDK ソフトウェアを検索し、システム上に正しいバージョンがインストールされていない場合は、その旨をユーザーに通知してインストールを停止します。JDK の適切なバージョンは、Forte for Java のポータルからダウンロードできます。

チュートリアルアプリケーションの作成と実行に必要なコンポーネント

チュートリアルの作成と実行には次のものが必要です。これらの一部は、Forte for Java, Community Edition のデフォルトインストールに含まれています。

- データベースソフトウェア (次のいずれか)
  - PointBase Network Server, version 3.5

    Forte for Java, Community Edition のインストール時に PointBase をインストールできます。PointBase がインストールされているかどうかを調べるには、Forte for Java がインストールされているディレクトリ下に `pointbase` ディレクトリが存在するかどうかを確認します。PointBase がインストールされていない場合は、インストーラを再度起動してインストールを行うことができます。
  - Oracle 8.1.7, JDBC Thin Driver 8.1.7
  - Microsoft SQL Server 2000, WebLogic JDBC driver 5.1.0 または JDBC-ODBC ブリッジ (SQL Server 2000 ODBC ドライバ)
  - IBM DB2 7.1, JDBC Thin Driver for DB2 7.1
- Web サーバー (次のいずれか)
  - Tomcat, version 3.2

    チュートリアルでは、Web アプリケーションを使用するため Web サーバーが必要です。このチュートリアルでは、IDE にバンドルされている Tomcat の組み込みバージョンを使用します。これには、テストを目的とした Web サーバーの機能があります。
  - iPlanet Web Server, version 6.0

■ Web ブラウザ

チュートリアルアプリケーションのページを表示するには Web ブラウザが必要です。Web ブラウザとしては、Netscape Communicator™ または Microsoft Internet Explorer を使用できます。Netscape Communicator, version 4.7*x* は、Forte for Java のポータルからダウンロードできます (製品 CD-ROM にも含まれています)。

Forte for Java IDE の起動

次の節で説明している手順で Forte for Java IDE を起動します。詳細は『Forte for Java, Community Edition インストールガイド』を参照してください。

Solaris、UNIX、Linux 環境の場合

インストールが完了すると、*forte4j-home*/bin ディレクトリに runide.sh スクリプトが格納されます。このスクリプトは、次のようにして起動することができます。

```
$ sh runide.sh
```

このスクリプトをカスタマイズする方法については、4 ページの「コマンド行オプションを使用したセッションの変更」を参照してください。

Microsoft Windows 環境の場合

次の 3 種類の起動方法があります。

■ デスクトップ上の「Forte for Java CE」アイコンをダブルクリックする。

runidew.exe 実行可能プログラムが実行され、これによりコンソールウィンドウを使用せずに IDE が起動されます。このアイコンは、*forte4j-home*¥bin ディレクトリにあります。ここには、代替アイコンの runide.exe もあり、このアイコンを使用した場合は、コンソールウィンドウに IDE からの標準エラーと標準出力が表示されます。コンソール上で Ctrl+Break キーを押すと、実行中のスレッドのリストを取得できます。Ctrl+C キーを押すと、プログラムを直ちに終了することができます。

- 「スタート」>「プログラム」>「Forte for Java CE」>「Forte for Java CE」を選択する。
- コマンド行から任意の実行可能プログラムを実行する。

```
C:¥> runide.exe [switch]
```

コマンド行オプションを使用したセッションの変更

表 1-1 に、IDE の起動方法を変更するためのオプションについて説明します。詳細は『Forte for Java, Community Edition インストールガイド』を参照してください。

- Microsoft Windows システムの場合

コマンド行で IDE を実行する際にオプションを設定できます。

- Solaris、Linux、その他の UNIX マシンの場合

インストールディレクトリの `bin` サブディレクトリにある `runide.sh` ファイルを変更するか、オプションを指定して `runide.sh` を起動する独自のシェルスクリプトを作成します。

表 1-1　`runide` コマンド行オプション

オプション	意味
`-classic`	クラシック JVM を使用する
`-cp:p` *addl-classpath*	指定したクラスパスをクラスパスの先頭に追加する
`-cp:a` *addl-classpath*	指定したクラスパスを Forte for Java クラスパスの末尾に追加する
`-fontsize` *size*	IDE のグラフィカルユーザーインタフェースで使用するフォントのサイズを、指定したサイズに設定する
`-J`*jvm-flags*	指定したフラグを JVM に直接渡す (`-J` と引数の間にはスペースを入れない)
`-jdkhome` *jdk-home-dir*	デフォルトの SDK の代わりに、指定した Java 2 SDK を使用する
`-h` または `-help`	コマンド行オプションを表示する GUI ダイアログを開く

表 1-1　runide コマンド行オプション (続き)

オプション	意味
-hotspot、-client、-server、-classic、-native、 -green	指定したタイプの JVM を使用する
-single	シングルユーザーモードで IDE を実行する。ユーザー設定ディレクトリの代わりに、*forte4j-home* から IDE を起動できるようにする
-ui *UI-class-name*	IDE の見た目と使い心地として所定のクラスを使用して IDE を実行する
-userdir *user-directory*	現在のセッションに指定したユーザー設定ディレクトリを使用する。詳細については次の節を参照

ユーザー設定ディレクトリの指定

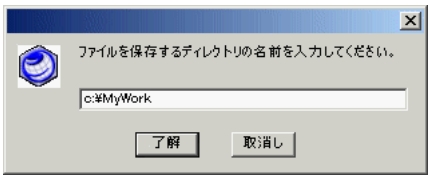
デフォルトでは、Forte for Java ソフトウェアをマルチユーザーモードで実行し、個別のプロジェクト、サンプル、IDE 設定をユーザーが指定した任意のディレクトリに格納します。これにより、個々の開発者がそれぞれの開発作業を同期させながら、個人の作業と設定を維持できます。

- Solaris、UNIX、Linux 環境の場合

-userdir コマンド行オプションを使用してユーザー設定ディレクトリを明示的に指定しない場合は、ユーザー設定はデフォルトで *user-home*/ffjuser30 ディレクトリに格納されます。

- Microsoft Windows システムの場合

Forte for Java IDE を初めて起動した場合は、このディレクトリを指定するダイアログが表示されます。C:¥MyWork のようにフルパスで指定します。



IDE の起動時に -userdir コマンド行オプションを使用し、別のユーザー設定ディレクトリを指定することもできます。

Forte for Java のディレクトリ構造

Forte for Java ソフトウェアをインストールすると、インストールディレクトリに、表 1-2 に示すサブディレクトリが作成されます。

表 1-2　Forte for Java のディレクトリ構造

ディレクトリ	目的
`beans`	IDE にインストールされた JavaBeans™ コンポーネントが格納される。
`bin`	Forte for Java 起動ツール (Microsoft Windows インストールの場合は `ide.cfg` ファイルも) が格納される
`docs`	Forte for Java のヘルプファイルおよびその他の各種マニュアルが格納される (リリースノートは *forte4j-home* を参照)。
`iPlanet`	iPlanet プラグインで使用するファイルが格納される。
`javadoc`	デフォルト設定で IDE の Javadoc リポジトリにマウントされるディレクトリ。IDE で提供される Javadoc およびユーザーが作成した Javadoc の両方がこのディレクトリに格納される。
`lib`	IDE 実装の中核となる JAR ファイルとオープン API が格納される。
`modules`	Forte for Java モジュールの JAR ファイルが格納される。
`pointbase`	PointBase Network Server データベース用の実行可能プログラム、クラス、データベース、マニュアルが格納される (インストールした場合)。
`sources`	ライブラリのソースが格納される。これらのソースは、ユーザーアプリケーションで再配布できる。
`system`	IDE が特別な目的に使用するファイルとディレクトリが格納される (技術サポートが必要な場合に役立つ `ide.log` など)。

Forte for Java ソフトウェアをデフォルト (マルチユーザー) モードで起動する場合は、表 1-3 に示すサブディレクトリがユーザー設定ディレクトリにインストールされます。これらのほとんどが、Forte for Java のホームディレクトリのサブディレクトリに対応しており、設定を保持するために使用されます。

表 1-3　ユーザー設定ディレクトリのディレクトリ構造

ディレクトリ	目的
`beans`	IDE にインストールされた JavaBeans コンポーネントのユーザー設定が格納される。
`javadoc`	IDE にインストールされた Javadoc ファイルのユーザー設定が格納される。
`lib`	システム `lib` ファイルのユーザー設定が格納される。
`modules`	アップデートセンターからダウンロードされたモジュールが格納される。
`sampledir`	エクスプローラの「ファイルシステム」区画にデフォルトでマウントされるディレクトリ。IDE 内で作成したオブジェクトが保存される。ただし、別のディレクトリをマウントして代わりに使用する場合は、そのディレクトリに保存される。
`sampledir/examples`	サンプルアプリケーションが格納される。
`sampledir/tutorials`	このマニュアルで説明する CDShopCart チュートリアルアプリケーションと、そのデータベーススクリプトなどの複数のチュートリアルアプリケーションが格納される。
`system`	システムファイルおよびディレクトリのユーザー設定が格納される。

チュートリアルデータベース表の作成

Forte for Java, Community Edition のチュートリアルを開始する前に、任意のデータベースにデータベース表を作成してインストールしておく必要があります。この表を作成するために、各 SQL に対応した SQL スクリプトが用意されています。

表 1-4 に示すスクリプトは、ユーザー設定ディレクトリの `sampledir/tutorial/CDShopCart/SQLscripts` サブディレクトリに格納されています。

表 1-4 チュートリアルの表作成に使用する SQL スクリプト

スクリプト名	説明
`CDCatalog_pb.sql`	チュートリアルアプリケーションが使用する表を PointBase SQL 形式で作成し、データを格納する。
`CDCatalog_ora.sql`	チュートリアルアプリケーションが使用する表を Oracle SQL 形式で作成し、データを格納する。
`CDCatalog_ms.sql`	チュートリアルアプリケーションが使用する表を SQLServer SQL 形式で作成し、データを格納する。

`CDCatalog` スクリプトは、表 1-5 に示すデータベーススキーマを作成します。

表 1-5 `CDCatalog` データベース表

表名	列	主キー	その他
CD	id	○	
	cdtitle		
	artist		
	country		
	price		
SEQUENCE	tableName	○	
	nextPK		
CDORDER	id	○	
	orderDate		
ORDERITEM	orderID	○	
	lineItemID		外部キーは、CDORDER (id) を参照する。
	productID		外部キーは、CD (id) を参照する。

CD 表には、表 1-6 に示すレコードが格納されます。

表 1-6　CD 表のレコード

id	cdtitle	artist	country	price
1	Yuan	The Guo Brothers	China	14.95
2	Drums of Passion	Babatunde Olatunji	Nigeria	16.95
3	Kaira	Tounami Diabate	Mali	13.95
4	The Lion is Loose	Eliades Ochoa	Cuba	12.95
5	Dance the Devil Away	Outback	Australia	14.95

Sequence 表には、表 1-7 に示すレコードが格納されます。

表 1-7　Sequence 表のレコード

tableName	nextPK
CdOrder	1

PointBase での表のインストール

最初にチュートリアルデータベースを作成し、そのデータベースに表を読み込みます (表は、任意の PointBase データベースにインストールできます)。

1. PointBase Network Server を起動します。

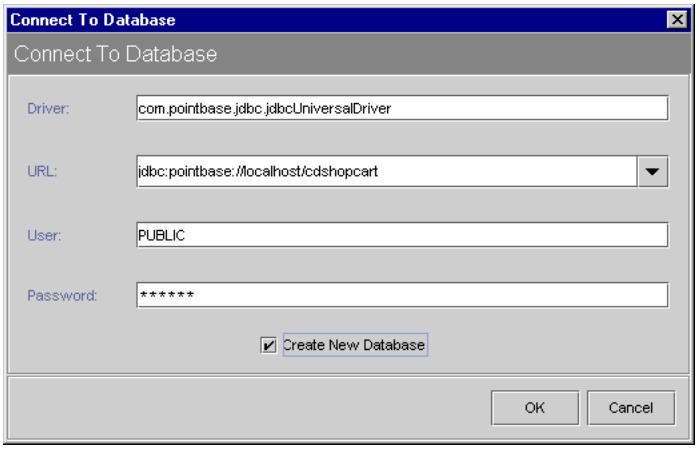
   - Solaris または Linux 環境の場合
     forte4j-home/pointbase/network ディレクトリの netserver.sh ファイルを実行します。
   - Microsoft Windows の場合
     「スタート」>「プログラム」>「Forte for Java CE」>「PointBase」>「Network Server」>「Server」を選択するか、*forte4j-home*/pointbase/network ディレクトリの netserver.bat ファイルをダブルクリックします。

2. PointBase コンソールを起動します。

   - Solaris または Linux 環境の場合
     forte4j-home/pointbase/client ディレクトリの console.sh ファイルを実行します。
   - Microsoft Windows の場合
     「スタート」>「プログラム」>「Forte for Java CE」>「PointBase」>「Client Tools」>「Console」を選択するか、*forte4j-home*/pointbase/client ディレクトリの console.bat ファイルをダブルクリックします。

   「Connect to Database」ダイアログが表示され、PointBase ドライバには、デフォルトのサンプルデータベースの値が表示されています。

3. 以下に示すように、「URL」フィールドの末尾にある sample という文字を cdshopcart に変更します。



4. 「Create New Database」チェックボックスをオンにして「OK」をクリックします。

   「PointBase Console」ウィンドウが表示されます。ステータスメッセージの最後に「Ready」が表示されたら、次の作業に進みます。

5. 「File」>「Open」を選択し、ファイル選択用のダイアログを表示します。

6. CDCatalog_pb.sql ファイルを指定し、「Open」をクリックします。

   ユーザー設定ディレクトリの sampledir/tutorial/CDShopCart/SQLscripts にある CDCatalog_pb.sql を選択します。CDCatalog_pb.sql ファイルの内容が SQL エントリウィンドウにコピーされます。

7. 「SQL」>「Execute All」を選択します。

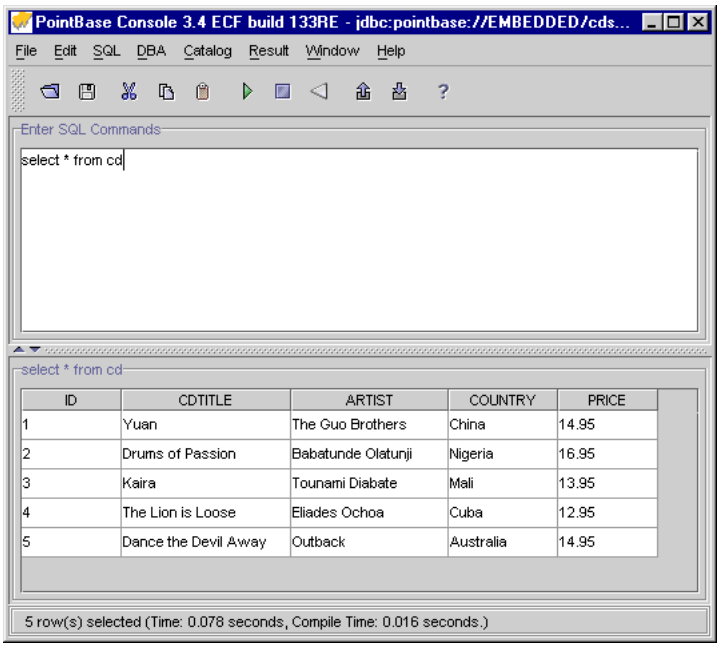
   メッセージが表示され、スクリプトが実行されたことが示されます。このとき、「Cannot find the table...」で始まる初期メッセージが表示されますが、無視してください。これらのメッセージが表示されるのは、まだ作成されていない各表に対する DROP 文によるものです。これらの DROP 文は、今後、スクリプトを再実行して表を初期化する場合に役立ちます。

8. 「Window」>「Clear Input」を選択して SQL エントリウィンドウをクリアし、次を入力して作成した表をテストします。

```
SELECT * FROM CD
```

9. 「SQL」>「Execute」を選択します。

   コンソールには、次のような CD 表が表示されます。



10. 「PointBase Console」ウィンドウを閉じます。

    3 ページの「Forte for Java IDE の起動」 に進みます。

他のデータベースへの表のインストール

PointBase 以外のデータベースを使用する場合は、そのデータベースが IDE で使用できることを確認してから、表をインストールします。

PointBase 以外のデータベースの設定

Forte for Java で PointBase 以外のデータベースを使用する場合は、次のようにします。

1. ユーザーのシステム上にデータベースソフトウェアをインストールします。

   データベースソフトウェアのサーバーバージョンまたはクライアントバージョンを使用できます。クライアントバージョンをインストールする場合は、対応するデータベースサーバーにアクセスできる必要があります。

2. 適切な JDBC 対応ドライバを *forte4j-home*/lib/ext ディレクトリに手動でコピーします。

   たとえば、Oracle の場合は、classes12.zip ファイルをこのディレクトリにコピーします。

3. アプリケーションコードで JDBC 接続文字列を作成するときに、データベース (およびドライバ) に適切な値を指定します。

   詳細は、42 ページの「JDBC の connection タグを使用したデータベースへの接続」を参照してください。

チュートリアル表のインストール

Oracle または Microsoft SQLServer データベースにチュートリアルデータベースをインストールする手順を示します。

Oracle を使用する場合は、次のようにします。

- コマンド行で CDCatalog_ora.sql スクリプトを読み込みます。

  次の例では、tutorial/tutorial というユーザー名/パスワードを使用し、TUTORIAL というサービス名のデータベースに表が読み込まれます。

```
C:¥> cd your_user_dir¥sampledir¥tutorial¥CDShopCart¥SQLscripts
C:¥> sqlplus tutorial¥tutorial@TUTORIAL @CDCatalog_ora.sql
```

Microsoft SQLServer を使用する場合は、次のようにします。

1. データベースを作成するか、既存のデータベースを指定します。
2. CDCatalog_ms.sql ファイルの第 1 行目を次のように変更します。

```
use CDShopCart
```

使用するデータベース名が CDShopCart に置き換わります。

3. CDCatalog_ms.sql スクリプトを読み込みます。

次の例では、tutorial/tutorial というユーザー名/パスワードを使用し、MY_MSSQL というサーバーのデータベースに表が読み込まれます。

```
c:¥> cd your_user_dir¥sampledir¥tutorial¥CDShopCart¥SQLscripts
c:¥> isql -SMY_MSSQL -Ututorial -Ptutorial < cdcatalog_ms.sql
```

第2章

チュートリアルアプリケーションの概要

実際にアプリケーションを作成することで、Forte for Java, Community Edition の機能を使用した Web コンポーネントの作成方法を学習します。

この章では、作成するアプリケーションについて説明します。最初に要件を詳述し、次にこれらの要件を満たす構造について説明します。最後の節では、Web モジュール構造、Forte for Java タグライブラリ、透過的な持続性 (TP) など、Forte for Java, Community Edition の機能を使用してアプリケーションを作成する方法について説明します。

この章の構成は次のとおりです。

チュートリアルアプリケーションの機能

サンプルのチュートリアルアプリケーションである CDShopCart は、音楽用 CD を購入するための簡単なオンラインショッピングカートアプリケーションです。顧客は、Web ブラウザを使用し、アプリケーションのインタフェースと次のようにやりとりします。

1. 顧客はカタログページから CD を選択し、ショッピングカートに追加します。
2. ショッピングカートにさらに CD を追加することも、ショッピングカートに入れた CD を取り除くこともできます。

3. 顧客が CD を購入する際に、アプリケーションはデータベースに注文を書き込み、「Thank You」というメッセージと注文番号を表示し、セッションを終了します。

4. この時点で、顧客はアプリケーションを終了することも、注文ページに戻って新しいショッピングセッションを開始することもできます。

アプリケーションのシナリオ

CDShopCart アプリケーションとのやりとりは、顧客がアプリケーションのカタログページを訪れるときから始まり、顧客が注文を済ませるかサイトから出た時点で終了します。以下に示すシナリオは、CDShopCart アプリケーションとのユーザーのやりとりを示します。これらのシナリオを見ていくと、このアプリケーションの要件と、アプリケーション内で行われるやりとりを理解できます。

1. 顧客が、アプリケーションのホームページの URL をブラウザに指定することによって、アプリケーションが起動されます。

   ホームページは、CD Catalog List ページです。このページには、販売されている音楽 CD の一覧とそれに関連する情報 (CD のタイトル、CD の ID 番号、アーティストの名前、アーティストの国籍、価格) が表示されます。

2. 顧客は、CD に関連付けられている「Add」ボタンをクリックし、購入する CD を選択します。

   これによって、Shopping Cart ページが表示されます。このページには、選択した CD のタイトル、ID 番号、価格が表示されます。

3. 顧客はさらに購入する CD を選択できます。

   顧客が Shopping Cart ページにある「Resume Shopping」ボタンを押すと、顧客が別の CD を購入できるようにアプリケーションが CD カタログページを再表示します。顧客はこの手順を何度でも繰り返すことができます。同じ CD を複数回追加することも可能です (この場合は同じ CD の行がカートに追加されます。各 CD の数量 (Amount) を表す列は表示されません)。

4. 顧客は、ショッピングカートページ上にある CD に関連付けられている「Delete」ボタンを押して、ショッピングカートから CD を取り除きます。

   このボタンを押すと、CD が取り除かれた状態のショッピングカートが再度表示されます。なお、すべての CD を削除した場合は、ショッピングカートは表示されず、カートが空であることを示す別のページが表示されます。

5. 顧客は、ページ上にある「Resume Shopping」ボタンをクリックして CD Catalog List ページに戻ることも、「Cancel Order」ボタンをクリックしてセッションを終了させることもできます (Cancel Order ページについては、手順 7 で説明しています)。

6. 顧客は、Shopping Cart ページ上の「Place Order」ボタンをクリックして購入を決定します。

   これによって注文がデータベースに書き込まれ、「Thank You」というメッセージと注文番号が表示されてから、このセッションが終了します。このとき、顧客は、このメッセージページ上にある「Resume Shopping」リンクをクリックして新しいセッションを開始することも、ブラウザを閉じるか別の URL を指定することによって、このアプリケーションを終了することもできます。

7. 顧客が、Shopping Cart ページで「Cancel Order」ボタンを押すと、注文をいつでも取り消すことができます。

   これによって、「Cancel Order」というメッセージを示すページが表示され、セッションが終了します。Cancel Order ページには、「Resume Shopping」ボタンがあり、このボタンを使用して新しいセッションを開始することもできます。

アプリケーション機能の仕様

以下に、ショッピングの手順をサポートするア プリケーションのユーザーインタフェースの主な機能を示します。

- ページ間を移動するための一連のリンク
- サイトで提供する機能の一覧
- 購入のために選択した CD のビュー
- 購入する CD を追加するためのカタログページ上のボタン
- CD を削除するためのショッピングカートページ上のボタン
- チェックアウトを起動するためのショッピングカートページ上のボタン
- 注文を取り消すためのショッピングカートページ上のボタン
- ホームページに戻り、新しい注文を行うための、チェックアウトページ上のボタン
- ホームページに戻るための、空のカートページ上のボタン
- 注文を取り消すための、空のカートページ上のボタン

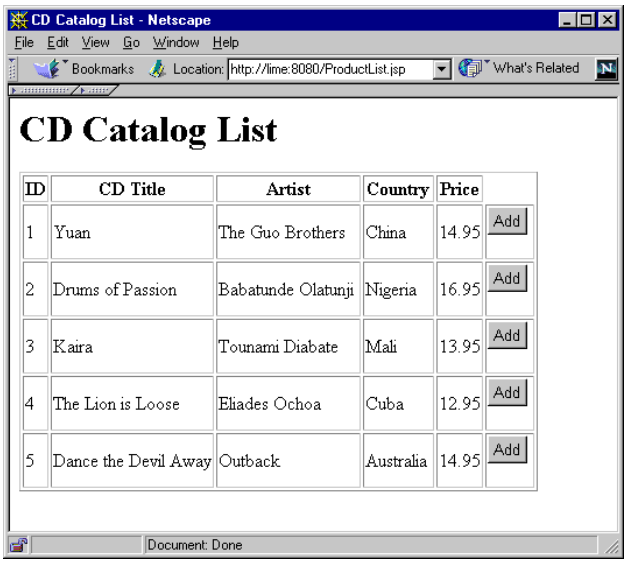
ユーザーから見たチュートリアルアプリケーション

15 ページの「チュートリアルアプリケーションの機能」で説明したシナリオと機能仕様が、ユーザーから見た場合にどのように実現されるかを示します。

CDShopCart アプリケーションを実行するには、次のようにします。

1. アプリケーションを起動すると、最初に CD のタイトル一覧を表示する CD Catalog List ページが表示されます。

   このページは、ProductList JSP ページで作成されます。



2. CD をショッピングカートに追加するには、その CD の行にある「Add」ボタンをクリックします。

   これによって、Shopping Cart ページが表示され、選択した CD が表示されます。このページは、ShopCart JSP ページで作成されます。

3. 別の CD を追加するには、「Resume Shopping」をクリックします。これによって、CD Catalog List ページに戻ります。

4. 同じまたは異なる CD について「Add」をクリックします。

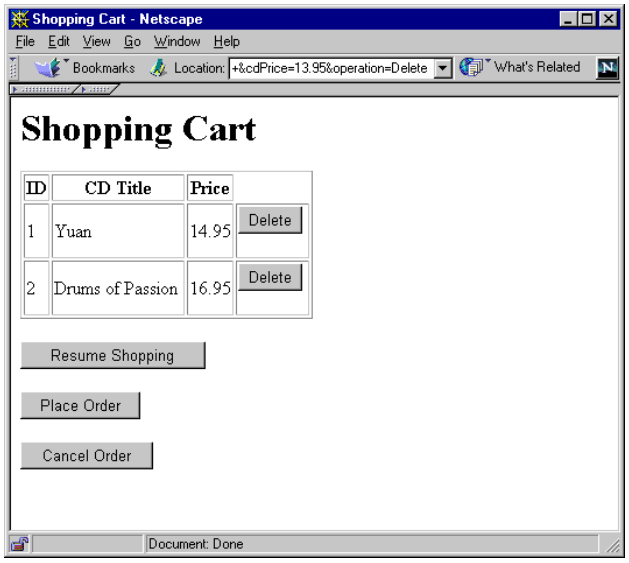
   Shopping Cart ページが表示され、追加した CD も表示されます。

5. 購入する CD をすべて選択するまで、手順 2 と手順 3 を繰り返します。

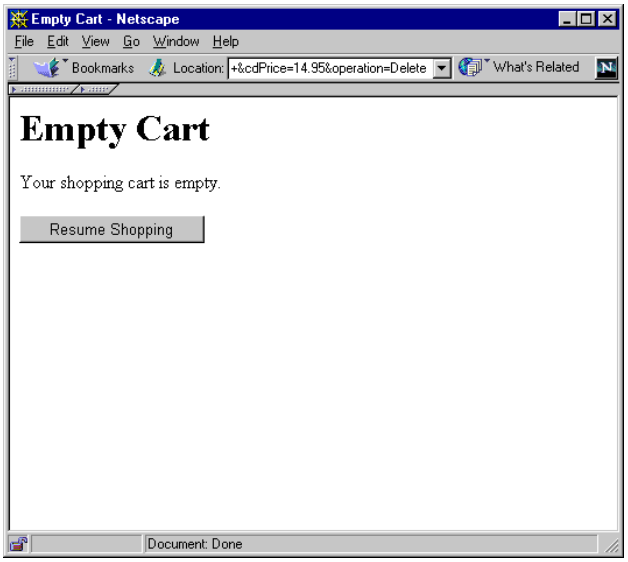
   Shopping Cart ページに選択した CD が表示されます。同じ CD を複数選択した場合でも、別の行に表示されます。

6. CD を削除するには、その CD の「Delete」ボタンをクリックします。

   表が再表示されます。削除した CD は表示されません。

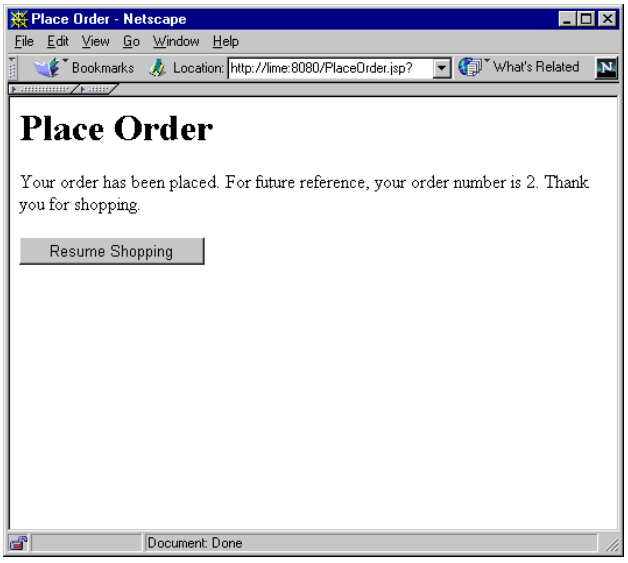


表内のすべての CD を削除すると、Empty Cart ページが表示されます。



7. 「Resume Shopping」ボタンをクリックし、CD Catalog List ページに戻ります。
8. 注文を行うには、Shopping Cart ページの「Place Order」ボタンをクリックします。

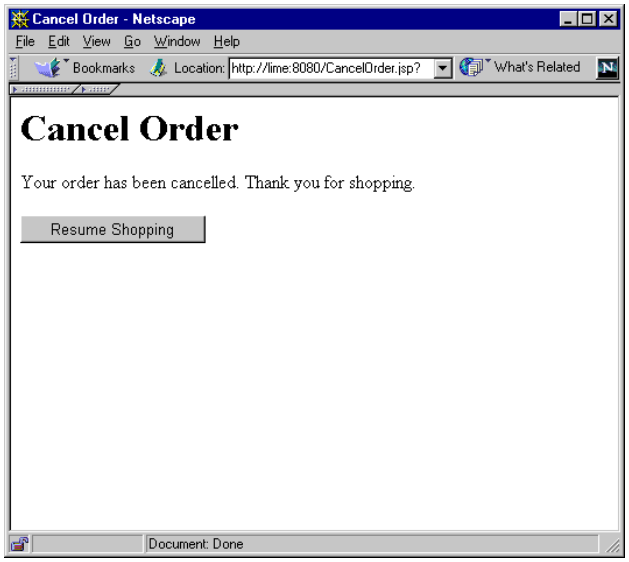
   Place Order ページが表示されます。このページは、PlaceOrder JSP ページによって作成されます。



この時点で、ブラウザに別の URL を指定してこのアプリケーションを終了することも、「Resume Shopping」ボタンをクリックして新しいセッションを開始することもできます。

9. 注文を取り消す場合は、Shopping Cart ページで「Cancel Order」ボタンをクリックします。

   Cancel Order ページが表示されます。このページは、CanelOrder JSP ページによって作成されます。



新しいセッションを開始するには、「Resume Shopping」ボタンをクリックします。

チュートリアルアプリケーションの構造

CDShopCart アプリケーションは、Web 主体アプリケーションであり、Web クライアントを使用して Web アプリケーションに要求を送信し、Web アプリケーションから結果を受信します。Web アプリケーションは、Web コンポーネントと、それをサポートするクラス、Bean、ファイルの集まりです。Web コンポーネントとは、サーブレットや JSP ページなど、サーバー側の J2EE コンポーネントです。

CDShopCart アプリケーションは、単一の Web モジュールから構成されています。Web モジュールとは、J2EE アプリケーション内にある Web 資源の最小の配備単位であると同時に、使用可能な最小単位です。Forte for Java, Community Edition で紹介する機能の 1 つに Web モジュール構造があります。Web モジュール構造は、必要なディレクトリ構造、必要なデータオブジェクトのデフォルトバージョン、および Web モジュールが必要とするその他の特殊なサービスを自動的に作成します。

Web モジュールと関連する概念の詳細については、『Web コンポーネントのプログラミング』を参照してください。Web モジュール構造については、「JSP/サーブレットモジュールの概要」のオンラインヘルプにある「Web モジュールの開発」も参照してください。

図 2-1 に、CDShopCart アプリケーションの各要素とその相互関係を示します。

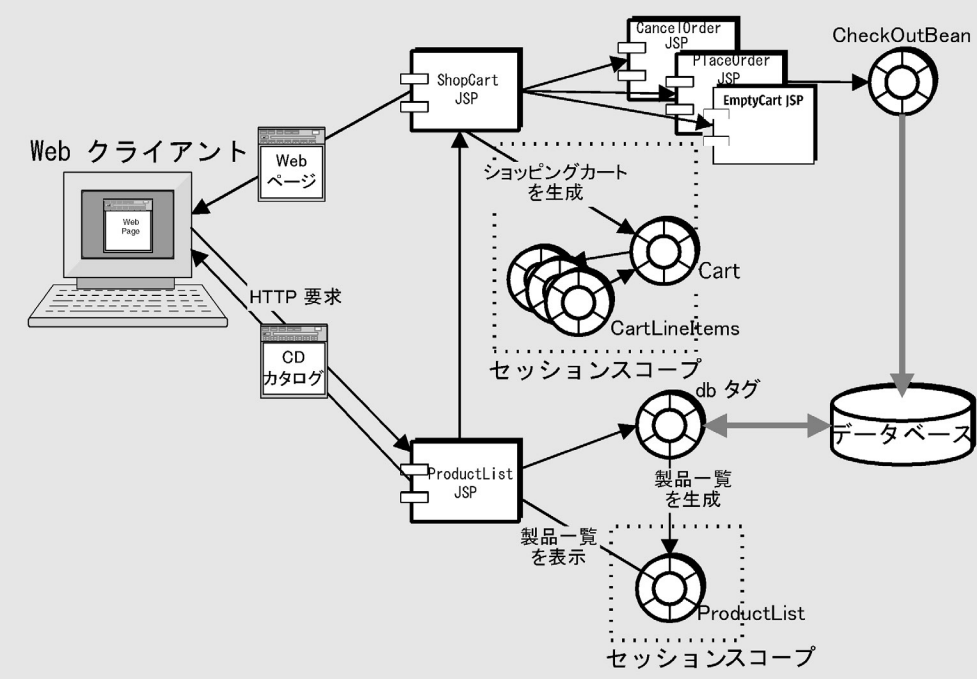


図 2-1　CDShopCart アプリケーションの構造

アプリケーションの要素

以下に、図 2-1 で示した各要素について説明します。

- クライアント構成要素

クライアント構成要素は、アプリケーションページを表示する Web ブラウザです。

- サービス構成要素 (Web モジュール) には、次のものがあります。
  - ProductList JSP ページ — データベースから CD データを取り込み、CD Catalog List Web ページ上の表にそのデータを表示します。また、このページでは、「Add」ボタンを表示します。ユーザーはこのボタンを使用し、CD をショッピングカートに追加することができます。
  - ShopCart JSP ページ — 購入するために選択した CD を Shopping Cart Web ページの表に表示します。このページでは、「Delete」「Place Order」「Cancel Order」「Resume Shopping」の各ボタンを表示します。
  - EmptyCart JSP ページ — 顧客がショッピングカートから最後の CD を削除したときにメッセージを表示します。このページは「Resume Shopping」ボタンを表示します。
  - CancelOrder JSP ページ — 注文が取り消されたことを示すメッセージと、ProductList JSP ページに戻るための「Resume Shopping」ボタンを表示します。
  - PlaceOrder JSP ページ — `CheckOutBean` を使用し、注文をデータベースに保存して注文番号を生成します。このページは、注文が行われたことを示すメッセージと注文番号、および ProductList JSP ページに戻るための「Resume Shopping」ボタンを表示します。
  - `Cart` Bean — 購入のために選択した CD を表します。
  - `CartLineItem` Bean — カート明細 (CD) を表します。
  - `CheckOutBean` Bean — 透過的な持続性を使用し、注文データでデータベースを更新し、一意の注文番号を生成します。

サービス構成要素の詳細

CDShopCart アプリケーションのサービス構成要素は、Web モジュールであり、この Web モジュールには 4 つの JSP ページが含まれます。これらの JSP ページは、クライアントからの入力を受けて、アプリケーションの動作を調整します。また、Web モジュールのサポート要素には、JavaBeans 要素と HTML ページファイルがあります。

■ `ProductList` JSP ページ

このページは、現在のユーザーのセッションを検索し、存在しない場合は、それを作成します。`ProductList` ページは、Forte for Java データベースのタグライブラリに含まれているタグを使用し、データベースに格納されている CD の一覧にアクセスしたり、Forte for Java 表現ライブラリに含まれているタグを使用し、CD を表に表示したりします。`ProductList` ページは、表示するカート明細 (CD) ごとに「Add」ボタンを表示します。

■ `ShopCart` JSP ページ

ユーザーが `ProductList` ページ上の「Add」ボタンをクリックしたときに、CD 明細データがこの JSP ページに渡されます。ShopCart JSP ページは、`Cart` オブジェクト (`CartLineItem` オブジェクトから構成される) をインスタンス化し、Forte for Java 表現タグライブラリのタグを使用し、Cart オブジェクトを表内に表示します。`ShopCart` ページでは、各カート明細 (CD) に「Delete」ボタンを表示します。このボタンをクリックすると、このページは、スクリプトレットを使用してその CD を削除し、表データを更新して再度表を表示します。カート内の最後の CD が削除されたときは、`EmptyCart` ページに移動します。`ShopCart` ページは、「Resume Shopping」「Cancel Order」「Place Order」の各ボタンを表示します。これらのボタンによって、それぞれ、`ProductList` ページ、`CancelOrder` ページ、`PlaceOrder` ページに進みます。

■ `Cart` JavaBean

この Bean は、`lineItem` 属性だけでなく、`CartLineItem` オブジェクトを取得および削除するためのメソッドを保持しています。`DShopCart` ページによってインポートされます。

■ `CartLineItem` JavaBean

この Bean は、CD 関連の属性だけでなく、`Cart` 明細 (CD) の属性 (ID、タイトル、アーティスト、国籍、価格) の取得と設定を行うメソッドを保持しています。

■ `CancelOrder` JSP ページ

この JSP ページが呼び出されるのは、ユーザーが `ShopCart` ページで「Cancel Order」ボタンをクリックしたときです。このページは、セッションを終了し、注文が取り消されたことを示すメッセージを表示し、`ProductList` ページに戻るための「Resume Shopping」ボタンを表示します。

- `EmptyCart` JSP ページ

この JSP ページが呼び出されるのは、ユーザーが `ShopCart` ページで最後の CD を削除したときです。`EmptyCart` ページは、カートが空であることを示すメッセージと、「Resume Shopping」ボタンを表示します。

- `PlaceOrder` JSP ページ

この JSP ページが呼び出されるのは、カート内に CD が存在するときにユーザーが `ShopCart` ページの「Place Order」ボタンをクリックした場合です。`PlaceOrder` ページでは、`CheckOutBean` を使用して注文をデータベースに保存してから、このセッションを終了し、「Thank you」というメッセージを表示します。`PlaceOrder` ページは、「Resume Shopping」ボタンを表示します。

- `CheckOutBean` JavaBean

この Bean は、透過的な持続性ツールを使用してデータベースに接続し、顧客の注文に一意の注文番号を割り当て、新しい注文データでデータベースを更新します。

チュートリアルアプリケーションの作成に必要な作業

実際に行う作業は、2 つの章 (第 3 章と第 4 章) に分かれています。第 3 章では、基本アプリケーションを作成しますが、注文をデータベースに保存しません。第 4 章では、透過的な持続性 (TP) を使用し、注文をデータベースに保存する方法を学びます。

チュートリアルアプリケーションを作成するには、あらかじめ、第 1 章で説明している手順に従って Forte for Java ソフトウェアをインストールして設定し、起動しておくことと、チュートリアルのデータベース表をインストールしておく必要があります。

基本アプリケーションの作成

第 3 章では、Forte for Java, Community Edition の次に挙げる機能の使用方法を学習します。

- Web モジュール (作成、開発、テスト実行)
- データベースタグ (データベースへの接続とデータベースとのやりとりに使用する)
- 表現タグ (取り込まれたデータを反復処理する)

この他、サポート要素であるいくつかの Bean や HTML ページも作成します。

Web モジュールを作成する

Forte for Java には、Web アプリケーションの階層的なディレクトリ構造を自動的に作成するためのツールが用意されています。この構造は、Web モジュールです。透過的な持続性に関連した要素を除いて、1 つの Web モジュール構造内で CDShopCart アプリケーション全体を開発します。

このチュートリアルでは、Web モジュール開発の詳細については説明していません。31 ページの「Web モジュールの作成」では、Web モジュールの作成に関する導入的な内容を紹介し、Web モジュール構造の基本要素と Web モジュールを作成するための簡単な方法の概略を示しているだけです。Web モジュール開発の詳細については、『Web コンポーネントのプログラミング』を参照してください。

Forte for Java のタグライブラリの使用方法

CDShopCart Web モジュールでは、ProductList JSP ページを作成し、CD カタログのデータを取り込み、そのデータを CD Product List Web ページに表示します。次に、ShopCart JSP ページを作成し、購入のためにユーザーが選択した CD を表示します。このような処理を行うために、Forte for Java のカスタム JSP タグライブラリにあるデータベースアクセス機能とデータ表現機能を利用します。

36 ページの「Forte for Java のカスタムタグの使用方法」では、カスタムデータベースタグを使用してデータベースに JDBC 接続し、CD データを取り込む方法について説明します。次に、表現タグを使用して結果データを反復処理する方法についても説明します。これによって、ProductList ページでは、Web ページ上の HTML フォームに結果データを表示することができます。

ShopCart JSP ページは、ProductList JSP ページから渡された CD データを表示します。59 ページの「CD の追加や削除を行うコードの追加」では、表現タグを使用して渡された値を反復処理し、個々のフィールド値を見つける方法を示しています。これによって、渡された値をカート表内の該当する列に表示することができます。

サポート要素の作成

ShopCart JSP ページで使用するサポート要素には、2 つの Bean (Cart Bean と CartLineItem Bean) と、3 つの JSP ページ (CancelOrder、PlaceOrder、EmptyCart) があります。

50 ページの「CartLineItem 構成要素 (JavaBeans) の作成」では、ユーザーが CD の「Add」ボタンをクリックしたときに、ProductList から ShopCart に渡されるカート明細 (CD) のパラメータを保持するオブジェクトを持つ Bean を作成します。次に、55 ページの「Cart 構成要素 (JavaBeans) の作成」では、選択された複数のカート明細 (CD) を保持するオブジェクトを持つ Bean の作成方法を学習します。Cart Bean には、カート明細に行を追加および削除するためのメソッドもあります。

65 ページの「Empty Cart ページの作成」では、カートが空になったことを示すメッセージを表示する JSP ページを作成します。これは、Shopping Cart ページで空のフォームの表示を防止するためです。

この他に 2 つの JSP ページを作成しますが、ProductList や ShopCart ほどの重要なロジックは含まれていません。67 ページの「Place Order ページの作成」では、注文を受けたメッセージを表示し、セッションを終了する JSP ページを作成します。第 4 章では、このページに機能拡張を行い、透過的な持続性を使用して注文をデータベースに書き込みます。69 ページの「Cancel Order ページの作成」では、注文が取り消されたことを示し、セッションを終了する同様のページを作成します。

アプリケーションのテスト実行

この章では、各要素を作成した直後にテストを行います。Web モジュールのコンポーネントのどれかを実行すると、IDE によって、その Web モジュールが内部的なコンテナに自動的に配備されます。

透過的な持続性の追加

第 4 章では、透過的な持続性を使用し、 第 1 章で作成した CdOrder 表に注文を書き込む方法を学習します。

持続可能クラスの作成

3 つのチュートリアルデータベース表 (CDORDER、ORDERITEM、SEQUENCE) を使用し、顧客の注文とその注文の明細に一意の番号を生成します。これによって、個々の注文を一意に識別して格納することができます。これらのデータベース表を Java クラスとして操作できるようにするには、データベーススキーマを取り込み、それらのスキーマをもとに持続可能クラスを生成します。これらのクラスに持続性をもたせる最後の手順として、クラスの「機能拡張」を行います。つまり、これらのクラスを JAR ファイルにパッケージ化します。次に、この JAR ファイルをアプリケーションの

Web モジュール階層に組み込みます。これによって、JAR ファイルが Web モジュールの WAR (Web ARchive) ファイルに格納されるようになります。これらの手順については、78 ページの「持続可能クラスの作成」を参照してください。

データベースへの注文の保存

1 つの `CheckOutBean` は、ユーザーが「Place Order」ボタンをクリックしたときに、注文データをデータベースに書き込むためのあらゆる透過的な持続性の手続きをカプセル化します。この Bean を作成するための手続きについては、92 ページの「持続性認識 Bean の作成」で説明しています。また、この手続きでは、持続性マネージャファクトリの作成や持続性マネージャの作成も行います。前者では、データベース接続のためのテンプレートを作成し、後者では、トランザクション操作および照会操作を管理します。また、照会インタフェースのメソッドを使用し、必要なデータベース機能を実行します。この照会インタフェースのメソッドを使用しない場合は、SQL またはデータストアに固有の、別の言語を使用する必要があります。

結果を使用して注文を行う

101 ページの「`CheckOutBean` を呼び出すように PlaceOrder ページを変更する」では、`CheckOutBean` メソッドからの結果を使用するために PlaceOrder JSP ページを修正し、注文をデータベースに書き込み、返された注文番号を表示します。

アプリケーション全体のテスト実行

CDShopCart アプリケーションを実行し、いくつかの CD をショッピングカートに追加し、注文を行います。このテストでは、生成された注文番号を表示し、透過的な持続性の能力を証明します。また、データベース表内のデータの変化を確認し、生成される注文番号がどのように変化するかを調べることもできます。

最後に

このチュートリアルアプリケーションは、比較的短期間 (1 日程度) で作成できるように非常に簡潔に設計されています。そのため、次のような制約があります。

- エラー処理がない
- デバッグ手続きがない
- 配備する WAR ファイルの作成方法に関する説明がない

これらの手続きは、今後のリリースで追加される予定です。

チュートリアルアプリケーションは、すぐに完成できるように単純なアプリケーションとして設計されていますが、アプリケーション全体をインポートし、ソースファイルを表示し、そのソースファイル内にあるメソッドコードを、作成するメソッドにコピーすると便利です。CDShopCart アプリケーションは、IDE のユーザー設定ディレクトリにある `sampledir/tutorial` ファイルシステムからアクセスできます。

第3章

基本チュートリアルアプリケーションの作成

この章では、CDShopCart チュートリアルアプリケーションの作成方法を順を追って説明します (注文データのデータベースへの書き込みは除きます)。チュートリアルアプリケーションを作成するには、あらかじめ、第 1 章で説明する手順に従って Forte for Java ソフトウェアをインストールして設定し、起動しておくことと、チュートリアルデータベース表をインストールしておく必要があります。

この章では、次のトピックを扱います。

各コンポーネントは作成するたびにテストします。この章では、第 2 章で説明したように、基本的なチュートリアルアプリケーションを実行できるようになります。ただし、手順 9 で注文番号は表示されません。注文番号の表示は、第 4 章で行います。

Web モジュールの作成

CDShopCart アプリケーションは、Web アプリケーションです。Web アプリケーションは、Web モジュールから構成されます。CDShopCart アプリケーションは、1 つの Web モジュールだけを含む非常に単純なアプリケーションです。

この節では、Forte for Java, Community Edition を使用し、Web モジュール内でショッピングカート機能を実装する方法を説明します。

Web モジュールとは

『Java Servlet Specification, v2.2』によれば、 Web アプリケーションは、構造化されたディレクトリ階層として存在します。この階層のルートは、`document root` と呼ばれ、Web アプリケーションに組み込まれているすべてのファイルがここに格納されます。この階層には、`WEB-INF` ディレクトリという非共用サブディレクトリも含まれています。このサブディレクトリには、Web アプリケーションに関係するが、クライアントに直接サービスを提供しない要素が含まれています。具体的には、Web 配備記述子 (`web.xml` ファイル)、サーブレットおよびユーティリティクラスが含まれます。なお、ユーティリティクラスは、Web アプリケーションローダがクラスの読み込み元として使用します。

J2EE 配備構造を使用しても、必ずしもこのような特殊なディレクトリ構造内で Web アプリケーションを開発する必要はありません。しかし、アプリケーションのファイルが最終的に Web モジュール構造の一部になっていない場合は、それらのファイルを WAR ファイル (Web ARchive 形式のファイル) としてパッケージ化し、Web コンテナに組み込むことができません。Forte for Java の Web モジュール構造は、必須のディレクトリ階層を作成し、そのディレクトリ階層にいくつかのオブジェクトのデフォルトバージョンを埋め込むプロセスの大部分を自動化しています。

注 - このチュートリアルでは、Web モジュール開発の詳細については説明していません。Web モジュールの開発作業については『Web コンポーネントのプログラミング』を参照してください。また、「JSP/サーブレットモジュールの概要」のオンラインヘルプにある「Web モジュールの開発」も参照してください。

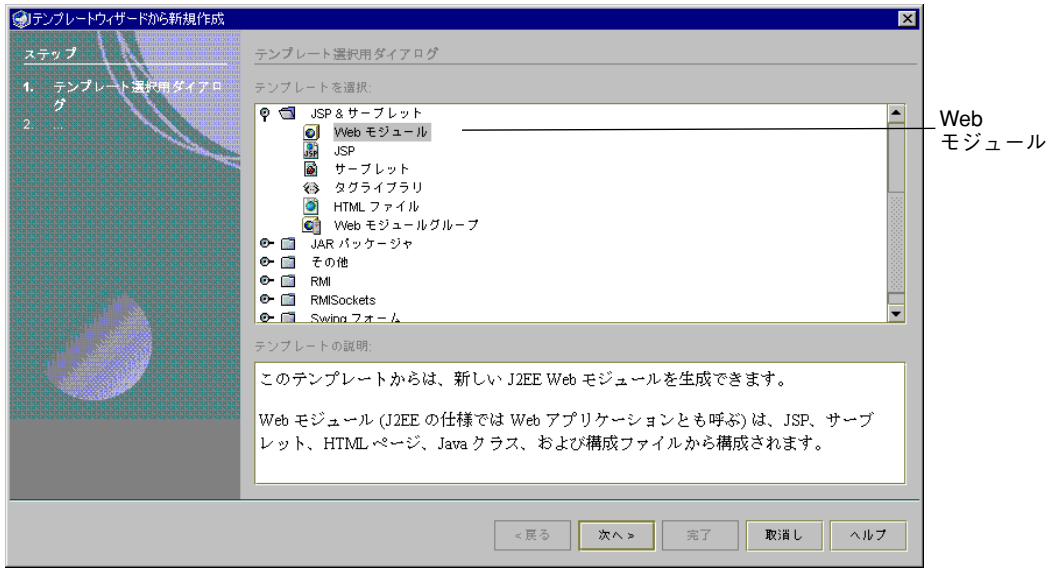
`CDShopCart` Web モジュールの作成

この節では、CDShopCart アプリケーションの Web モジュールを作成します。この Web モジュールは、ディレクトリです。Forte for Java の Web モジュール機能を使用すると、既存のディレクトリを Web モジュールに変換することができますが、現在そのようなディレクトリは存在しないため、最初から作成する必要があります。

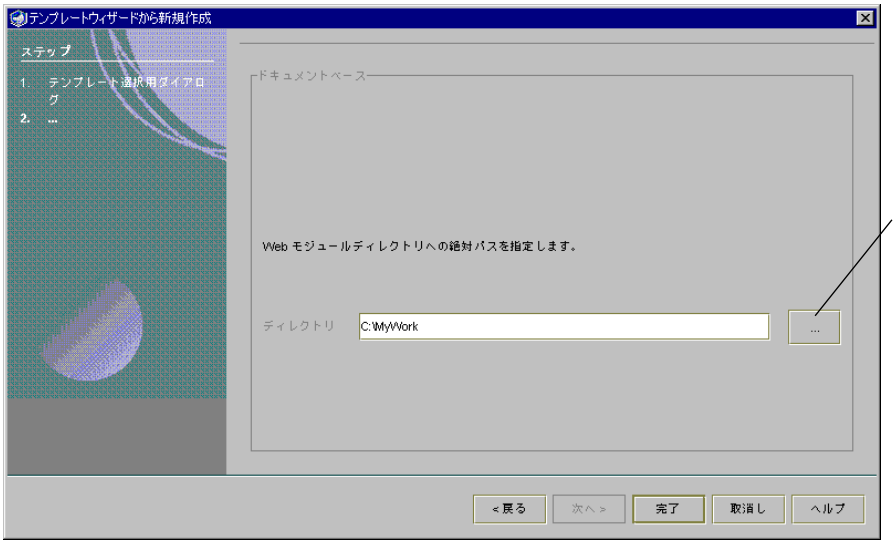
参照 - 作業を進める前に、デフォルトで提供されている sampledir ディレクトリを含むファイルシステムのマウントを解除することもできます。Forte for Java ソフトウェアで提供されている CDShopCart Web モジュールは、sampledir/tutorial ディレクトリにあります。CDShopCart Web モジュールが 2 つあると識別が困難になります。

CDShopCart Web モジュールを作成するには、次のようにします。

1. 「エクスプローラ」ウィンドウの「ファイルシステム」区画をアクティブにし、「ファイル」>「新規」を選択して「テンプレートウィザードから新規作成」を表示します。
2. 「JSP & サーブレット」ノードを開いて、「Web モジュール」を選択します。

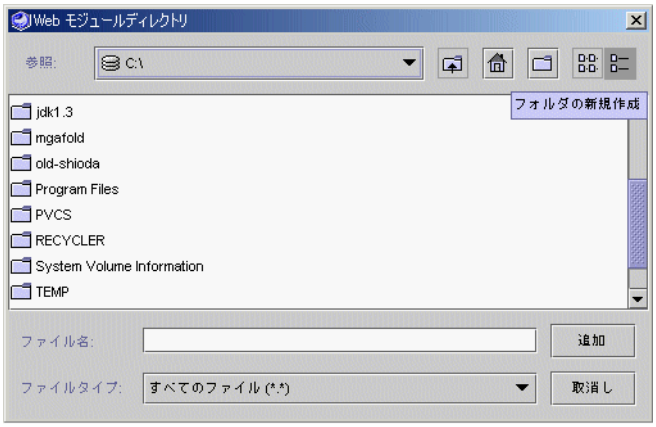


3. 「次へ」をクリックして、Web モジュールのディレクトリパスを指定するための区画を開きます。



4. 「...」ボタンをクリックして、新規ディレクトリを追加します。

5. 表示される「Web モジュールディレクトリ」ダイアログから、作成する場所を選択し、「フォルダの新規作成」ボタンをクリックします。

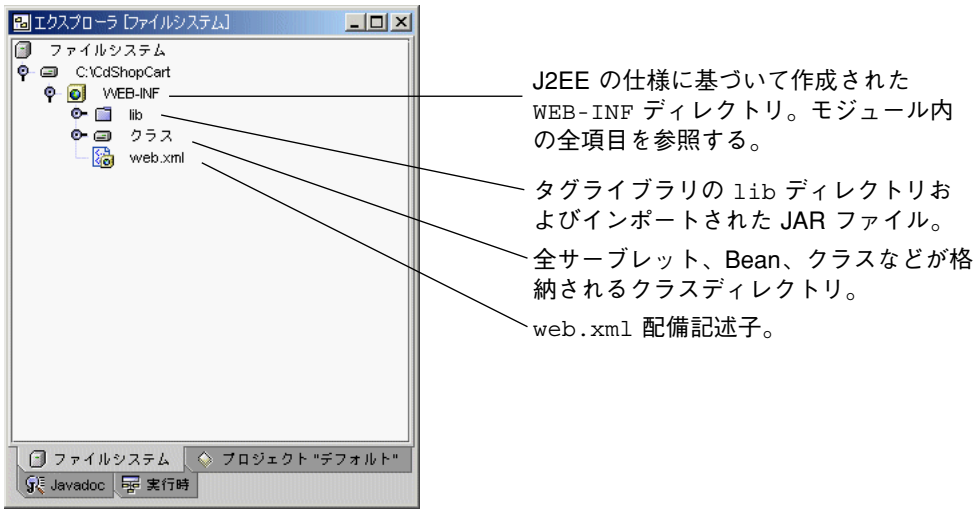


6. 「新しいフォルダ」(または「名称未設定フォルダ」) というフォルダが見つかるまで、画面をスクロールします。

7. 「新しいフォルダ」(または「名称未設定フォルダ」) を選択し、再度これをクリックして名前を編集できるようにします。
8. 名前を `CDShopCart` に変更し、Enter キーを押します。
9. フォルダリストから、`CDShopCart` フォルダを選択し、「追加」をクリックします。

   `CDShopCart` ディレクトリが「テンプレートウィザードから新規作成」(手順 3 を参照) に表示されます。
10. 「完了」をクリックします。

    以上でエクスプローラに新しい Web モジュールが作成されました。Web モジュールのビューを「プロジェクト」タブで変更するように要求されますが、「了解」をクリックして、このメッセージを消去します。
11. Web モジュール内でノードを開き、自動的に作成された項目を確認します。



これで、アプリケーションの最初の構成要素である、`ProductList` JSP ページの作成を開始できます。

Forte for Java のカスタムタグの使用方法

この節では、`ProductList` JSP ページを作成します。このページは、CD 製品データを取り込んで表示します。このチュートリアルでは、Forte for Java カスタムデータベースタグとそのタグライブラリを使用してデータベース機能を実行し、カスタム表現ライブラリを使用して取り込んだデータを反復します。

JSP タグとは

JSP ページで使用するタグは、JSP ファイルの本文で使用できるコードの 1 つの種類にすぎません。次に、使用できるコードの種類とコードにタグを組み込む方法についてまとめます。

JSP ファイルの本文では、固定テンプレートデータと要素という二種類のコードを使用できます。

- 固定テンプレートデータ

これは、JSP コンテナが認識しないタイプであり、HTTP 応答に変更を加えずそのまま渡します。固定テンプレートデータの例には、XML コードや HTML コードがあります。これらのコードは、CDShopCart で使用して、見出し、タイトル、表、ボタンなど、共通の HTML 要素を作成します。

- 要素タイプ
  - 指令

    どのパッケージをインポートするか、JSP をセッションに組み込むかなど、JSP に関するグローバルな情報を宣言するために使用します。
  - スクリプト要素

    Java コードを JSP ファイル内に埋め込むことができます。
  - タグ

    アクション要素は、Java コードを記述することなく Java オブジェクトと併用できる方法として、XML 方式のタグを使用します。たとえば、アクションを使用すると、Java オブジェクトの配置やインスタンス化、オブジェクトプロパティの取得や設定を行うことができます。各タグに関連付けられた Java クラスは、タグの機能を実装します。

タグ (アクション要素)

アクション要素には、どの JSP コンテナでも使用できる標準的なものがいくつかあります。これらの要素ついては、JSP 仕様ドキュメントで定義されています。また、カスタムアクション要素も作成することができます。カスタムアクション要素は、タグライブラリと呼ばれる XML ドキュメントで定義されています。タグライブラリは、命令要素内で宣言することによって、個々の JSP ページで使用できるようになります。データベース操作および表現のために CDShopCart アプリケーションで使用するタグは、Forte for Java で提供されているカスタムタグです。

Forte for Java タグライブラリ

IDE では、3 つのカスタムタグライブラリを提供しています。これらのタグライブラリは、互いに連動して行ベースの動的データの視覚的表現を生み出し、JDBC ResultSet、EJB コンポーネント、透過的な持続性クラス (JDO)、ベクトル、その他 JavaBeans のコレクションなど、さまざまなデータソースと併用することができます。タグライブラリには、次のものがあります。

- `ietags.jar` (表現および条件に関するタグ)
- `dbtags.jar` (データベースタグ)
- `tptags.jar` (透過的な持続性に関するタグ)

IDE では、タグライブラリのソースファイルが用意されています。これらのソースファイルは、タグ機能を実装するクラスおよびタグ構文を説明する XML 定義で使用されます。このファイルは、*forte4j-home*`/sources` ディレクトリ内の `jsptaglibs_src.jar` ファイルに存在します。

詳細

Forte for Java のタグライブラリについては、「JSP/サーブレットモジュールの概要」のオンラインヘルプにある「カスタムタグライブラリの作成」を参照してください。ここでは、構文に関する詳細な説明の他に、簡単な例が示されています。より本格的な例は、*your_user_dir*`/sampledir/examples/TagLibDemo` ディレクトリに用意されています。なお、このデモの実行方法については、このディレクトリに含まれている `readme.txt` ファイルを参照してください。

独自のカスタムタグの作成方法をはじめとした一般的なタグライブラリについては、『Web コンポーネントのプログラミング』を参照してください。

以降の節では、データベースタグおよび表現タグの使用方法について学習します。

CD Catalog List ページの作成

この節では、第 1 章でインストールしたデータベースからデータを取り込み、それを表に表示する機構を作成する方法について説明します。図 3-1 のようなページを作成します。

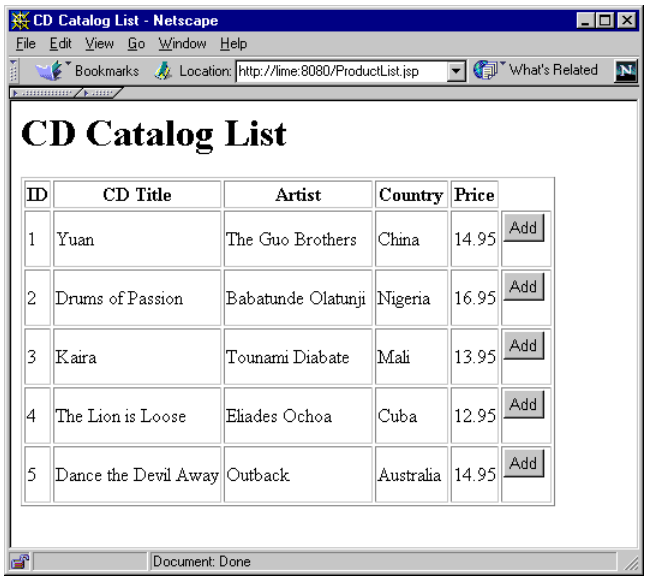


図 3-1　CD Catalog List ページ

このページを作成するには、次の作業を行います。

1. 39 ページの「Forte for Java タグライブラリの Web モジュールへの追加」
2. 40 ページの「ProductList JSP ページの作成」
3. 41 ページの「タグライブラリの宣言」
4. 42 ページの「JDBC の connection タグを使用したデータベースへの接続」
5. 44 ページの「JDBC の query タグを使用して CD データを取り込む」
6. 44 ページの「表現タグを使用してデータを反復処理する」
7. 46 ページの「各 CD 行の「Add」ボタンの作成」
8. 47 ページの「JDBC の cleanup タグを使用したクリーンアップ」

Forte for Java タグライブラリの Web モジュールへの追加

この節では、2 つの Forte for Java タグライブラリを CDShopCart モジュールにインポートし、これらのタグで実装されるアクションを使用します。1 つはデータベースアクションを実装するための dbtags.jar であり、もう 1 つは、表現アクションを実装するための ietags.jar です。

Forte for Java タグライブラリを Web モジュールにインポートするには、次のようにします。

1. エクスプローラで、CDShopCart モジュールを右クリックし、「ツール」>「JSP タグライブラリを追加」>「タグライブラリリポジトリ内を検索」を選択します。

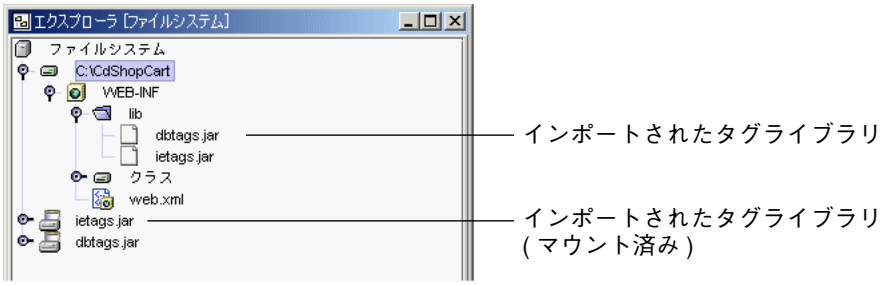
   「JSP タグライブラリ・リポジトリブラウザ」が表示されます。

2. dbtags と ietags の各ライブラリを選択 (Shift キーまたは Ctrl キーを押しながらクリック) して「了解」をクリックし、これらのライブラリを Web モジュールに追加します。

   エクスプローラで、「WEB-INF」ノード下の「lib」ノードを展開します。

3. 両方のタグライブラリが表示されていることを確認します。

   タグライブラリも別途マウントされます。エクスプローラでは、次のように表示されます。



参照 - トップレベルの「ファイルシステム」ノードを右クリックし、「カスタマイズ」を選択します。開いたウィンドウに、Forte for Java の CLASSPATH に現在マウントされているすべてのファイルが表示されます。タグライブラリが、ここに表示されていることを確認します。

`ProductList` JSP ページの作成

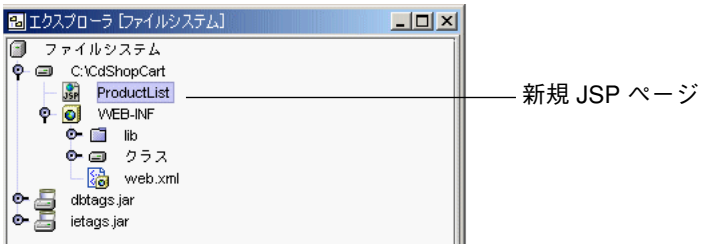
これで、タグを使用してデータベースから CD データを取り込み、表内に表示するページを作成することができます。このページのタイトルは、「CD Catalog List」です。このページを作成する機構は `ProductList` JSP ページにあります。

1. エクスプローラで、`CDShopCart` Web モジュールを右クリックし、「新規」>「JSP & サーブレット」>「JSP」を選択します。

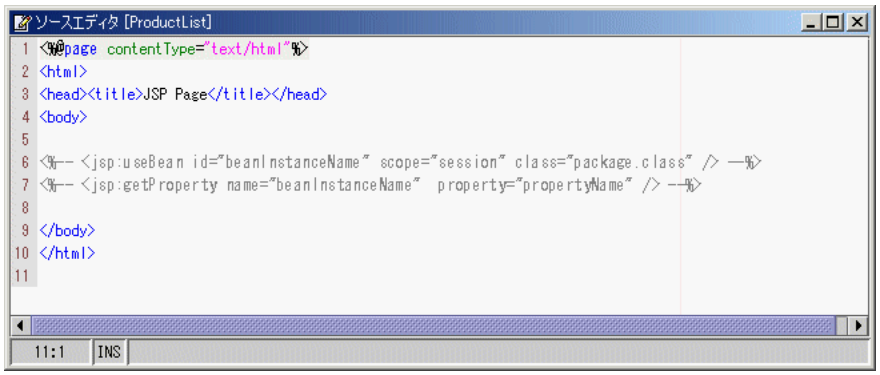
   「テンプレートウィザードから新規作成」に「新規オブジェクト名」区画が表示されます。

2. 「名前」フィールドに `ProductList` と入力し、「完了」をクリックします。

   Web モジュール内に `ProductList` JSP ページが表示されます。



ソースエディタには、JSP ページのスケルトンが表示されます。



ここでは、ソースエディタ のカラーコーディング機能とコード補完機能に注目してください。

タグライブラリの宣言

JSP ページでタグを使用するには、まず、`taglib` 指令を使用してタグライブラリを宣言する必要があります。

`taglib` 指令では、指定の URI にあるタグライブラリを JSP ページで使用することを宣言し、ライブラリ内のアクションの呼び出しに使用するタグ接頭辞を指定します。両方のタグライブラリに指定する URI は、Web モジュール内でのそれらの格納場所 (`/WEB-INF/lib`) を示します。`dbtags.jar` 内のタグの接頭辞は、`jdbc` にし、`ietags.jar` 内の表現タグの接頭辞は、`pr` にします。

これらの指令は、ページタイトルの直後でかつ JSP の本文の上に記述しなければなりません。以下の手続きでは、ページのタイトルの変更方法と、2 つのタグライブラリの指令の追加方法を示しています。

1. `ProductList` ページの本文では、ドキュメントのタイトルを「`CD Catalog List`」に変更します。

```
<head><title>CD Catalog List</title></head>
```

2. dbtags.jar および ietags.jar パッケージをインポートする命令を追加します。

```
//データベースタグライブラリ指令を追加する
<%@taglib uri="/WEB-INF/lib/dbtags.jar" prefix="jdbc" %>
//表現タグライブラリ命令を追加する
<%@taglib uri="/WEB-INF/lib/ietags.jar" prefix="pr" %>
```

JDBC の connection タグを使用したデータベースへの接続

JSP タグは、XML 構文に従って、次のいずれかの形式で記述します。

- 開始タグ (要素名) と指定可能な属性/属性値の組み合わせ、任意指定の本文、および対応する終了タグ
- 空のタグと指定可能な属性

このチュートリアルでは、両方のタイプの構文を使用します。まずは、JDBC の connection タグから使用します。これは、多くの属性をもつ空のタイプのタグです。このタグは、データベースに対し JDBC 接続を行います。これらの接続には、application、session、request、page のいずれかのスコープ (scope) を設定することができます。デフォルトのスコープは、application です。この JSP では、接続をアプリケーション全体でグローバルにするので、デフォルトのスコープ (application) を使用します。

connection タグは多くの属性を持ちますが、ここでは、次のコードに示す id、driver、url、user 属性だけを使用します。

```
<jdbc:connection id="connection_id"
    driver="driver_string"
    url="driver_url"
    user="user_id" password="pwd" />
```

詳細は、「JSP/サーブレットモジュールの概要」のオンラインヘルプの「カスタムタグライブラリの作成」にある、JDBC データアクセスタグに関するヘルプを参照してください。

connection タグを使用してデータベースに接続するには、最初に HTML の body タグの下に h1 タグでページの見出しを追加し、次に connection タグをその下に追加します。

1. ProductList ページの body タグの下に、ページのタイトルを作成します。

```
<body>
<h1> CD Catalog List </h1>
```

2. タイトルの下に、JDBC の connection タグを作成します。

   ■ 次のタグは、PointBase ドライバ用です。

```
<jdbc:connection id="jdbcConn"
    driver="com.pointbase.jdbc.jdbcUniversalDriver"
    url="jdbc:pointbase://localhost/cdshopcart"
    user="PUBLIC" password="PUBLIC" />
```

   ■ (thin driver を使用して) Oracle データベースを使用する場合は、以下のコードを使用します。

```
<jdbc:connection id="jdbcConn"
    driver="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@hostname:port#:SID"
    user="userid" password="password"
```

   デフォルトの Oracle ポート番号は、1521 です。

   ■ Weblogic ドライバを使用して Microsoft SQLServer データベースを使用する場合は、以下のコードを使用します。

```
<jdbc:connection id="jdbcConn"
    driver="weblogic.jdbc.mssqlserver4.Driver"
    url="jdbc:weblogic:mssqlserver4:database@hostname:port#"
    user="userid" password="password"
```

   SQLServer のデフォルトのポート番号は、1433 です。

JDBC の query タグを使用して CD データを取り込む

次に、query という別の JDBC タグを使用してデータベースを照会します。query タグは、データベースを照会し、結果を取得します。照会を実行したのち、生成された結果セットが pageContext 内の resultsID に格納されます。この ID を反復子タグに渡すと、結果を表示することができます。query タグでは、標準 SQL 文の Select をサポートしています。属性の代わりに SQL 文が本文に指定されるため、JSP スクリプト機能を使用して、query の作成方法を制御することができます。

query タグの構文は複雑です。次のように、開始タグと終了タグの間に、属性/属性値の組み合わせと任意指定の本文が挿入されます。

```
<jdbc:query id="stored_query_id" connection="connection_id"
        resultsId="results_id" resultsScope="scope" >
    body
</jdbc:query>
```

query タグは、connection タグの直後に記述します。

全 CD データを検索するためのデータベースの照会を行うには、次のようにします。

- ProductList ページで、connection タグの直後に、データベースから全 CD データを選択する照会を作成します。

```
//上記で作成した「jdbcConn」接続を使用して
//それをセッションに含める
<jdbc:query id="productQuery" connection="jdbcConn"
resultsId="productDS" resultsScope="session" >
    SELECT * FROM CD
</jdbc:query>
```

表現タグを使用してデータを反復処理する

この時点で、表を作成し、表のセルにデータを格納します。ここでは、iterator と field という 2 つの表現タグを使用し、取り込んだデータを反復処理します。

- iterator — results というデータソース内の行を反復処理します。iterator は、results というインタフェース、および JDBCResultSet, java.util.Vector、java.util.Collection といった特殊なデータソースタイプを理解します。

構文は、次のとおりです。

```
<pr:iterator results="results_id" >
    body
</pr:iterator>
```

- field — 結果の現在行にある値または名前を取得します。結果は、閉じている行またはフィールドの反復子から取得するのが普通ですが、直接指定することもできます。このチュートリアルでは、行の反復子を使用します。

表示されるフィールドは、name 属性または index 属性によって特定されます。name 属性が存在する場合は、タグによって名前の付いた列が結果から取り出されます。一方、index 属性が指定されていると、タグによってインデックス付きの列が取り出されます。両方とも存在しない場合は、field タグを fieldIterator タグで囲む必要があります。この場合は、fieldIterator の結果の現在行の現在列を取り出します。

このチュートリアルでは、name 属性を使用します。この構文は次のとおりです。

```
<pr:field name="field_name"/>
```

まず、HTMLタグを使用して表を作成します。次に、上述した query タグで指定した結果を iterator タグに渡します。最後に、HTML 構文内の field タグに結果の特定の列を取り込みます。

1. CD データを格納する表を作成します。

```
<TABLE border=1>
    <TR>
        <TH>ID</TH>
        <TH>CD Title</TH>
        <TH>Artist</TH>
        <TH>Country</TH>
        <TH>Price</TH>
    </TR>
```

2. 次に、iterator タグと field タグを使用して表にデータを格納します。

```
//次の表に、結果の現在行にある
//各フィールドから値を取り込む。
<pr:iterator results="productDS" >
    <TR>
        <TD><pr:field name="id"/></TD>
        <TD><pr:field name="cdtitle"/></TD>
        <TD><pr:field name="artist"/></TD>
        <TD><pr:field name="country"/></TD>
        <TD><pr:field name="price"/></TD>
//次の節で、ここにさらにデータを追加します。
</pr:iterator>
```

各 CD 行の「Add」ボタンの作成

CD 表の各行は、CD のデータを保持しています。ユーザーが CD を購入するには、各行の「Add」ボタンをクリックします。ここでは、HTML フォームを作成して、ユーザーが入力する (ボタンをクリックする) 領域を定義し、この領域内に Shopping Cart JSP ページに渡す情報を埋め込みます。このコードは、上記の終了 iterator タグ (最終行) の上に記述します。

「Add」ボタンを作成するには、次のようにします。

1. セル内に表のフォームを作成します (上記で作成した </pr:iterator> タグの上に作成します)。

```
<TD>
<form method=get action="ShopCart.jsp">
```

2. 埋め込む情報 (製品の ID、タイトル、価格) を指定します。

```
<input type=hidden name=cdId value="<pr:field name="id"/>">
<input type=hidden name=cdTitle value="<pr:field
name="cdtitle"/>">
<input type=hidden name=cdPrice value="<pr:field name="price"/>">
```

3. 上記コードのすぐ下に「Add」ボタンを指定します。

```
<input type=submit name=operation value=Add>
```

4. フォーム、セル、行を終了します。

```
</form>
</TD>
    </TR>
```

5. iterator 終了タグの下で表を終了します。

```
</pr:iterator>
</TABLE>
```

JDBC の cleanup タグを使用したクリーンアップ

cleanup タグを追加し、JSP ページの本文を終了します。JDBC の cleanup タグは、JSP ページ内にある他のタグが使用している資源を解放します。cleanup タグの構文は、次のとおりです。

```
<jdbc:cleanup scope="scope" status="ok|error" />
```

このタグの詳細については、オンラインヘルプを参照してください。

資源のクリーンアップを行ってページを終了するには、次のようにします。

1. 次のコードを入力します。

```
<jdbc:cleanup scope="session" status="ok" />
</body>
</html>
```

クリーンアップタグのスコープ (scope) は、クリーンアップ機構を使用するその他のタグのスコープセットに関係しています。この場合、query タグはクリーンアップ機構を使用し、照会により取得した結果セットを閉じます。結果セットのスコープは query タグのセッションに設定されているため (44 ページの「JDBC の query タグを使用して CD データを取り込む」 を参照)、cleanup タグのスコープもセッションに設定されます。

2. 「ファイル」>「保存」を選択し、作業内容を保存します。

ProductList JSP ページのテスト実行

作業の妥当性検査を行います。ProductList JSP ページをコンパイルし、統合化された Forte for Java 実行時システムとブラウザを使用し、ページを実行します。

ProductList ページをテストするには、次のようにします。

1. エクスプローラで、CDShopCart Web モジュールを選択し、「構築」>「すべてを構築」を選択します。

   ツールバーの下部にあるメッセージ領域に表示されるステータスメッセージを観察します。すべてが正常に終了すると、「完了」と表示されます。問題がある場合は、出力ウィンドウにエラーメッセージが表示され、問題の行が指摘されます。問題を修正し、「完了」というメッセージが表示されるまで、この手順を繰り返します。

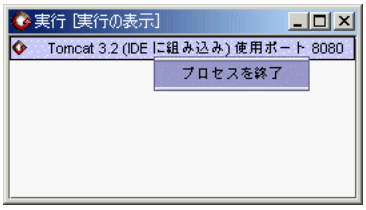
2. ProductList JSP を選択し、ツールバーの「実行」ボタン ( ) をクリックして ProductList JSP を実行します。

   代わりに、「構築」>「実行」を選択するか、ProductList を右クリックしてコンテキストメニューから「実行」を選択することもできます。

   IDE は組み込み Tomcat サーバーを起動し、次に実行作業領域に切り替え、「実行」ウィンドウを開きます。サーブレットが実行されると、「実行」ウィンドウにメッセージが表示され、ブラウザが開きます。数秒すると、図 3-1 のように CD Catalog List ページが表示されます。

3. 実行を中止する場合は、「実行」ウィンドウ内のプロセスを右クリックし、「プロセスを終了」を選択します。

   中止可能なプロセスは、Tomcat Web サーバーのプロセスです。



注 - セッション内で再度プログラムを実行するときは、「実行 (再読み込み強制)」を選択する必要があります。Java クラスまたはサーブレットを変更した場合は、この選択は必須です。JSP ページだけを変更した場合は必要ありません。

---

4. 「編集」タブをクリックして編集作業領域に戻ります。

以上で、JSP ページの作成、Forte for Java のカスタムタグを使用したデータベースへの接続、データベースからのデータの取り込みと表示が完了しました。次に、Shopping Cart ページを作成します。

Shopping Cart ページとサポート要素の作成

次に、CD Catalog List ページから購入する選択した CD を表示する機構を作成します。具体的には、次のような構成要素が必要です。

- ProductList ページからパラメータとして渡された選択済み CD 行の属性を保持する Bean (`CartLineItem`)
- `CartLineItem` オブジェクトを保持する別の Bean (`Cart`)
- `Cart` オブジェクトを受け取り、それらを表内の行として表示する `ShopCart` JSP ページ
- `ShopCart` ページ上に表示される CD ごとに用意する「Delete」ボタン
- それぞれの機能を実装した「Cancel Order」「Resume Shopping」「Place Order」ボタン

数枚の CD を選択した場合、作成されるページは図 3-2 のようになります。

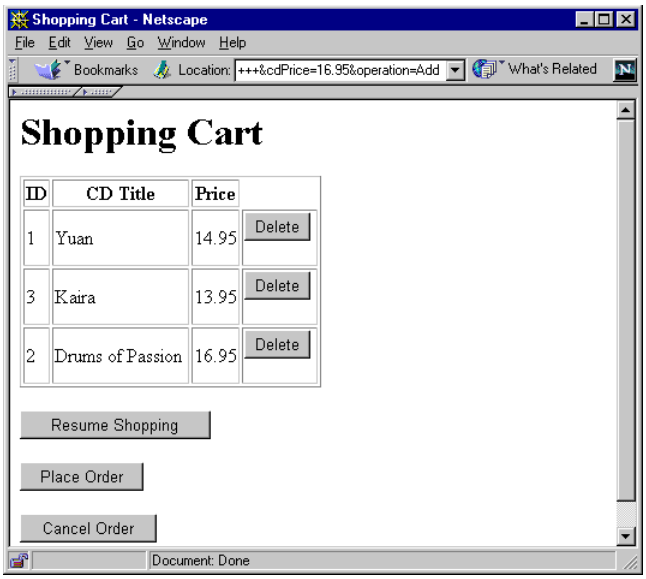


図 3-2　Shopping Cart ページ

次の手順を参照し、`ShopCart` ページとその Bean を作成します。

1. 50 ページの「CartLineItem 構成要素 (JavaBeans) の作成」
2. 53 ページの「id および price プロパティの文字列への変換」
3. 55 ページの「Cart 構成要素 (JavaBeans) の作成」
4. 59 ページの「Shopping Cart ページの作成」
5. 64 ページの「Shopping Cart ページのテスト実行」

`CartLineItem` 構成要素 (JavaBeans) の作成

CD Catalog List (`ProductList`) ページから Shopping Cart ページに渡されるパラメータを保持するオブジェクトを持つ、カート明細 (CD) Bean を作成します。このために、この Bean に 3 つのプロパティとアクセスメソッドを作成します。

1. `CDShopCart` Web モジュールの「WEB-INF」ノードを開き、「クラス」ノードを右クリックして「新規」>「Beans」>「Bean」を選択します。

   「テンプレートウィザードから新規作成」の「新規オブジェクト名」区画が表示されます。

2. 「名前」フィールドに `CartLineItem` と入力して「完了」をクリックします。

   エクスプローラ内に新しい `CartLineItem` が、ソースエディタ内にそのコードが表示されます。Bean に付いている赤い印 ( ) は、その Bean が「コンパイルが必要である」ことを表します。コンパイルは後で行うため、この印を特に気にすることはありません。

3. Bean とそのクラスを開き、その内容を表示します。

   `cdtitle` プロパティを作成します。

4. 「Bean パターン」ノードを右クリックし、「新規」>「プロパティ」を選択します。

5. 「新規プロパティパターン」ダイアログに、次の値を設定します。

   名前 : **`cdtitle`**
   種類 : **`String`**
   フィールドを生成 : チェックマークを付ける
   Return 文を生成 : チェックマークを付ける
   Set 文を生成 : チェックマークを付ける

   ダイアログは次のようになります。

6. 「了解」をクリックして、このダイアログを閉じます。

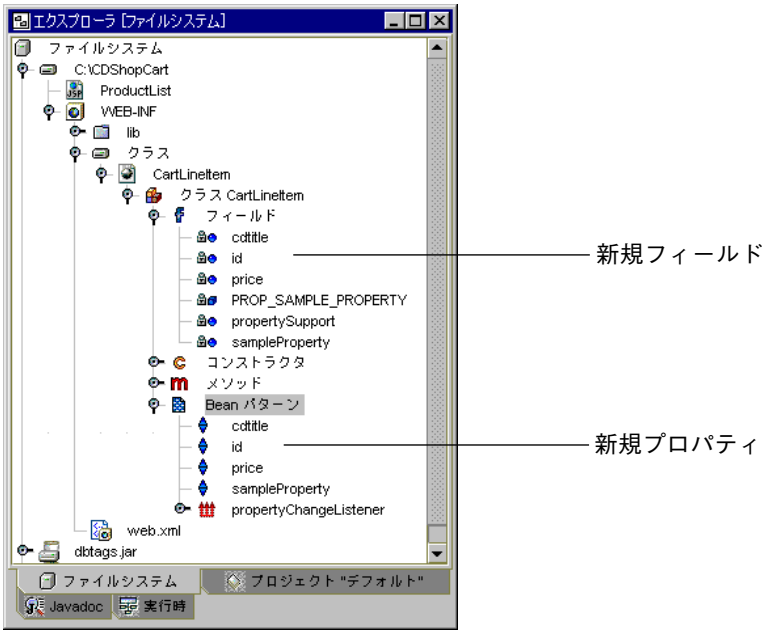
7. 同様に、`id` プロパティを作成します。フィールドには次の値を設定します。

   名前 : **`id`**
   種類 : **`int`**
   フィールドを生成 : チェックマークを付ける
   Return 文を生成 : チェックマークを付ける
   Set 文を生成 : チェックマークを付ける

8. 最後に、`price` プロパティを作成します。フィールドには次の値を設定します。

   名前 : **`price`**
   種類 : **`double`**
   フィールドを生成 : チェックマークを付ける
   Return 文を生成 : チェックマークを付ける
   Set 文を生成 : チェックマークを付ける

9. `CartLineItem` Bean クラスの「フィールド」ノードを開き、作成した新フィールドを表示します。



10. 「メソッド」ノードを開き、各フィールドに作成された新しい `get` および `set` のメソッドを表示します。

11. 新しいフィールド (または新しいメソッド) をダブルクリックし、作成された新しいコードをソースエディタで表示します。

id および price プロパティの文字列への変換

ProductList JSP ページから渡されるパラメータは、すべて文字列として渡されます。ただし、id と price の各プロパティは文字列ではないため、これらを文字列に変換する必要があります。この変換を効率よく行うには、プロパティの設定メソッドを無効にし、適切なコードを追加します。

setId メソッドおよび setPrice メソッドを無効にするには、次のようにします。

1. CartLineItem Bean の「メソッド」ノードを右クリックし、「新規メソッド...」を選択します。
2. 「新規メソッドを編集」ダイアログに、次の値を入力します。

   名前 : **setId**
   戻り値 : **void**
3. 「メソッドのパラメータ」区画で「追加」ボタンをクリックし、「メソッドパラメータを入力」ダイアログを表示します。
4. 次の値を入力します。

   型 : **java.lang.String**
   名前 : **pId**
5. 「了解」をクリックします。

   「新規メソッドを編集」ダイアログは、次のようになります。

6. 「了解」をクリックしてメソッドを作成し、このダイアログを閉じます。
7. この新しいメソッドに次のコードを追加します。

```
public void setId(java.lang.String pId) {
    int val = Integer.parseInt(pId);
    this.setId(val);
}
```

`setPrice` メソッドについても、同様の作業を行います。

8. 次の値を指定し、新しい `setPrice` メソッドを作成します。

   「メソッド」
   名前 : **`setPrice`**
   戻り値 : **`void`**

   「メソッドのパラメータ」
   型 : **`java.lang.String`**
   名前 : **`pPrice`**

9. 次のコードをこの新しいメソッドに追加します。

```
public void setPrice(java.lang.String pprice) {
    double val = Double.parseDouble(pprice);
    this.setPrice(val);
}
```

10. (クラス  ではなく) `CartLineItem` Bean () を右クリックし、メニューから「構築」>「コンパイル」を選択します。

    Bean のコンパイルでエラーが発生しなかった場合は、「コンパイルが必要である」ことを示す赤い印が Bean のノードから削除され、`Cart` Bean を作成できるようになります。エラーが発生した場合は、入力内容を確認してコンパイルをやり直します。

`Cart` 構成要素 (JavaBeans) の作成

Shopping Cart JSP ページは、ユーザーが「Add」ボタンを押したときに `ProductList` JSP ページによって渡された CD 明細オブジェクトを保持するカートオブジェクトをインスタンス化します。すでにそのようなカートオブジェクトが存在する場合は、それを検索します。`Cart` オブジェクトは、`Cart` Bean をベースにしています。

1. Web モジュールの「WEB-INF」ノードを開き、「クラス」ノードを右クリックして「新規」>「Beans」>「Bean」を選択します。
2. 「名前」フィールドに `Cart` と入力し、「完了」をクリックします。
3. `Cart` Bean の「Bean パターン」ノードを右クリックし、「新規」>「プロパティ」を選択します。

4. 「新規プロパティパターン」ダイアログに、次の値を設定します。

   名前： **lineItems**
   種類 : **java.util.Vector**
   フィールドを生成 : チェックマークを付ける
   Return 文を生成 : チェックマークを付ける
   Set 文を生成 : チェックマークを付ける

5. 新しい lineItems フィールド (または新しいアクセスメソッド) をダブルクリックし、作成した新しいコードをソースエディタで表示します。

   フィールドへのアクセスを、private から public に変更する必要があります。

6. Cart クラスの「フィールド」ノードを開き、「lineItems」フィールドを選択します。

7. 「プロパティ」ウィンドウの「修飾子」フィールドをクリックします。

注 - 「プロパティ」ウィンドウ (通常は「エクスプローラ」ウィンドウの下に表示されます) が開いていない場合は、「lineItems」を右クリックして、コンテキストメニューから「プロパティ」を選択します。「プロパティ」ウィンドウを開いたままにする場合は、「ウィンドウ」>「ウィンドウ」>「ブラウズ」>「プロパティウィンドウ」を選択します。

---

8. 「...」ボタンをクリックして修飾子のプロパティエディタを表示します。

9. 「アクセス」リストから「public」を選択します。

10. 「了解」をクリックして変更を適用します。

ここで、カート明細 (CD) オブジェクトをインスタンス化するコード、選択されている CD の数を返すメソッド、およびカートからカート明細 (CD) を削除する別のメソッドを追加する必要があります。

11. Cart Bean の「コンストラクタ」ノードを開き、「Cart()」をダブルクリックします。

    これにより、ソースエディタに `Cart()` コンストラクタが表示されます。

12. `Cart` Bean のコンストラクタにコードを追加し、次のように、新しい `lineItems` (CD) オブジェクトをンスタンス化します。

```
public Cart() {
    propertySupport = new PropertyChangeSupport ( this );
    lineItems = new java.util.Vector();
}
```

13. Cart の 「メソッド」ノードを右クリックして「新規メソッド...」を選択し、このメソッドに次の値を入力します。

    「メソッド」
    名前 : **findLineItem**
    戻り値 : **int**

    「メソッドのパラメータ」
    型 : **int**
    名前 : **pID**

14. ソースエディタで次のコードを findLineItem メソッドに追加します。

```
public int findLineItem(int pID) {
    System.out.println("Entering Cart.findLineItem()");
    //渡された ID で指定された cartItems 内の
    //CD の要素番号を返す
    int cartSize = (lineItems == null) ? 0 : lineItems.size();
    int i ;
    for (i = 0; i < cartSize; i++ ) {
        if  ( pID ==
            ((CartLineItem)lineItems.elementAt(i)).getId() )
            break ;
    }
    if (i >= cartSize) {
        System.out.println("Couldn't find line item for ID: " +
            pID);
        return -1 ;
    }
    else
        return i ;
}
```

15. removeLineItem メソッドを作成し、次の値を入力します。

「メソッド」
名前 : **removeLineItem**
戻り値 : **void**

「メソッドのパラメータ」
型 : **int**
名前 : **pID**

16. ソースエディタ内で次のコードを removeLineItem メソッドに追加します。

```
public void removeLineItem(int pID) {
    System.out.println("Entering cart.removeLineItem()");
    int i = findLineItem(pID);
    if (i != -1) lineItems.remove(i);
    System.out.println("Leaving cart.removeLineItem()");
}
```

17. (クラス  ではなく) Cart Bean () を選択し、「コンパイル」ボタンをクリックして Cart Bean をコンパイルします。

Bean はエラーなくコンパイルされます。次に、ShopCart JSP ページを作成します。

Shopping Cart ページの作成

次に、CD Catalog List ページから渡されたパラメータを受け取り、それらの一部 (ID、タイトル、価格) を表内の行として表示するページを作成します。このページでは、表から CD を削除し、Catalog List ページに戻り、注文を行うための機構も提供します。このページのタイトルは「Shopping Cart」で、このページを作成する機構は、`ShopCart` JSP ページにあります。

1. エクスプローラで `CDShopCart` Web モジュールを右クリックし、「新規」>「JSP & サーブレット」>「JSP」を選択します。
2. 「名前」フィールドに `ShopCart` と入力し、「完了」をクリックします。

   エクスプローラとソースエディタの両方に `ShopCart` JSP が表示されます。

   このページを開発するには、次の作業を行います。

   1. 59 ページの「CD の追加や削除を行うコードの追加」
   2. 62 ページの「表現タグを使用したカート表へのデータの格納」
   3. 63 ページの「ページへのボタンの追加」

CD の追加や削除を行うコードの追加

この節では、ショッピングカート内の CD 表を作成するためのコードを追加します。`Cart` オブジェクトおよび `CartLineItem` オブジェクトをインスタンス化します。そのため、`Cart` Bean、`java.util` ライブラリ (このライブラリの `CartLineItem` は Vector タイプ)、および `CartLineItem` をインポートするための指令を使用します。`ProductList` JSP ページで使用したものと同じ表現タグを使用するたね、これらのタグをインポートする指令も追加する必要があります。

スクリプトレットコードを使用してカートを作成し、`ProductList` JSP ページから渡されたパラメータをもとに、カート明細 (CD) の追加や削除 (ユーザーが「Delete」ボタンを押した場合) を行います。表が空の場合は、`EmptyCart` JSP ページに進みます。`EmptyCart` JSP ページは後で作成します。

`ProductList` JSP ページと同様に、反復子タグを使用して表データを編成してから、フォームを使用して表を作成し、各行に「Delete」ボタンを追加します。

1. page 指令を使用し、java.util ライブラリ、Cart Bean、CartLineItem Bean をインポートします。

```
<%@page contentType="text/html" %>
<%@page import="java.util.*, Cart, CartLineItem" %>
```

2. ページのタイトルを「Shopping Cart」に変更し、ietags.jar ライブラリをインポートするための指令を追加します。

```
<head><title>Shopping Cart</title></head>
<%@taglib uri="/WEB-INF/lib/ietags.jar" prefix="pr" %>
```

3. body タグの下に、このページの「Shopping Cart」という見出しを作成します。

```
<body>
<h1> Shopping Cart </h1>
```

4. 見出しの下に、usebean タグを使用して Cart Bean の使用を JSP ページに指示します。

```
<jsp:useBean id="myCart" scope="session" class="Cart" />
```

このコードは、Cart オブジェクトをインスタンス化し、それをセッション内に置きます。

ここでは、セッションで CD の「追加」を行っている場合の処理を指定する必要があります。すなわち、ユーザーが ProductList ページで「Add」ボタンをクリックしたときの処理です。cdID、cdTitle、cdPrice の各オブジェクトを取得し、これらを myCart オブジェクトに追加します。

5. 現在のオペレーションを取得し、ユーザーが「Add」ボタンをクリックしたときの処理を定義するコードで開始します。

```
<%
    String myOperation = request.getParameter("operation");
    session.setAttribute("myLineItems",myCart.getLineItems());

    if (myOperation.equals("Add")) {
        CartLineItem lineItem = new CartLineItem();
        lineItem.setId(request.getParameter("cdId"));
        lineItem.setCdtitle(request.getParameter ("cdTitle"));
        lineItem.setPrice(request.getParameter("cdPrice"));
        myCart.lineItems.addElement(lineItem);
        }
```

次に、セッションで CD の削除を行う場合の処理を指定する必要があります。これは、ユーザーが `ShopCart` ページで「Delete」ボタンをクリックしたときの処理です。

6. 次のコードを使用し、ユーザーが「Delete」ボタンをクリックしたときの処理を指定します。

```
    if (myOperation.equals("Delete")) {
        String s = request.getParameter("cdId");
        System.out.println(s);
        int idVal = Integer.parseInt(s);
        myCart.removeLineItem(idVal);
        }
```

最後に、「Delete」をクリックすることによって、`Cart` CD 表内の最後の行が削除されたときの処理を指定する必要があります。JSP の `forward` タグを使用して `EmptyCart` JSP ページ に移動します。`EmptyCart` ページ は、この後に作成します。最後のコードによって、手順 5 で開始したスクリプトが終了します。`forward` タグは、スクリプトレットを終了してから記述する必要がありますが、最終的に、このスクリプトレットを終了する前にスクリプトレットを再実行しなければなりません。

7. 次のコードを使用します。

```
    //カートから最後の CD が削除されたとき
    if (((Vector)session.getAttribute("myLineItems")).size()
        == 0) {
        //スクリプトレットを一時的に終了する。これにより、
        //JSP の forward タグを使用して EmptyCart ページに移動できる
        %>
        <jsp:forward page="EmptyCart.jsp" />
        //スクリプトレットを再実行する
        <%
        }
        %>
```

表現タグを使用したカート表へのデータの格納

次に、iterator タグと field タグを使用し、渡されたデータを反復処理しながら表にデータを格納します。この手続きは、44 ページの「表現タグを使用してデータを反復処理する」とほぼ同じです。

1. 表の見出しを作成し、購入予定のデータを格納する表の作成を開始します。

```
<TABLE border=1>
    <TR>
        <TH>ID</TH>
        <TH>CD Title</TH>
        <TH>Price</TH>
    </TR>
```

2. iterator タグと field タグを使用し、表にデータを格納します。

```
<pr:iterator results="myLineItems" >
    <TR>
        <TD><pr:field name="id"/></TD>
        <TD><pr:field name="cdtitle"/></TD>
        <TD><pr:field name="price"/></TD>
```

上記コード内の field タグが、結果の現在行にある指定された名前のフィールドから値を取り込みます。

3. 各行に「Delete」ボタンを作成します。これは、63 ページの「ページへのボタンの追加」と同じです。

```
<TD>
<form method=get action="ShopCart.jsp">
<input type=hidden name=cdId value="<pr:field name="id"/>">
<input type=hidden name=cdTitle value="<pr:field
    name="cdtitle"/>">
<input type=hidden name=cdPrice value="<pr:field
name="price"/>">
<input type=submit name=operation value="Delete">
</form>
</TD>
    </TR>
</pr:iterator>
</TABLE>
```

ページへのボタンの追加

最後に、「Resume Shopping」「Place Order」「Cancel Order」の各ボタンをページに追加します。

- ShopCart JSP ページに次のコードを追加します。

```
<p>
//「Resume Shopping」ボタンを作成する
<form method=get action="ProductList.jsp">
<input type=submit value="Resume Shopping">
</form>
//「Place Order」ボタンを作成する
<form method=get action="PlaceOrder.jsp">
<input type=submit value="Place Order">
</form>
//「Cancel Order」ボタンを作成する
<form method=get action="CancelOrder.jsp">
<input type=submit value="Cancel Order">
</form>
</p>
//ページを終了する
</body>
</html>
```

Shopping Cart ページのテスト実行

Shopping Cart ページは、直接テストすることができません。`ProductList` ページをテスト実行し、「Add」ボタンを使用して Shopping Cart ページへ移動します。

1. `CDShopCart` Web モジュールを選択し、メニューから「構築」>「すべてを構築」を選択します。

   すべてのコンパイルが正常に完了するはずです。

2. `ProductList` JSP ページを右クリックし、「実行 (再読み込み強制)」を選択します。

   数秒すると、CD Catalog List ページが表示されます。

3. いずれかの「Add」ボタンをクリックし、Shopping Cart ページに移動します。

   次のような Shopping Cart ページが表示されます。

4. 「Resume Shopping」ボタンを使用し、CD Catalog List ページに戻ります。

5. 実行を終了します (「実行」ウィンドウ内のプロセスを右クリックして「プロセスを終了」を選択します)。

6. メインウィンドウの「編集」タブをクリックし、編集作業領域に戻ります。

   これで、CDShopCart アプリケーションがほぼ完成しました。後は、EmptyCart ページと PlaceOrder ページを作成するだけです。

3 つのメッセージページの作成

次に、カートが空になった場合に表示される JSP ページと、ユーザーが「Place Order」ボタンや「Cancel Order」ボタンをクリックしたときに表示される 2 つの JSP ページを作成します。

この節では、次の作業を行います。

- 65 ページの「Empty Cart ページの作成」
- 67 ページの「Place Order ページの作成」
- 69 ページの「Cancel Order ページの作成」

Empty Cart ページの作成

反復子 (iterator) タグが空のベクトルを見つけると、空の表を作成するのではなく例外を出力します。ここでは、ベクトルが空であるかどうか (59 ページの「CD の追加や削除を行うコードの追加」の手順 7) を評価し、空の場合には、Empty Cart ページを表示する処理を行います。このページの「Resume Shopping」ボタンを使用すると、アプリケーションに戻ることができます。

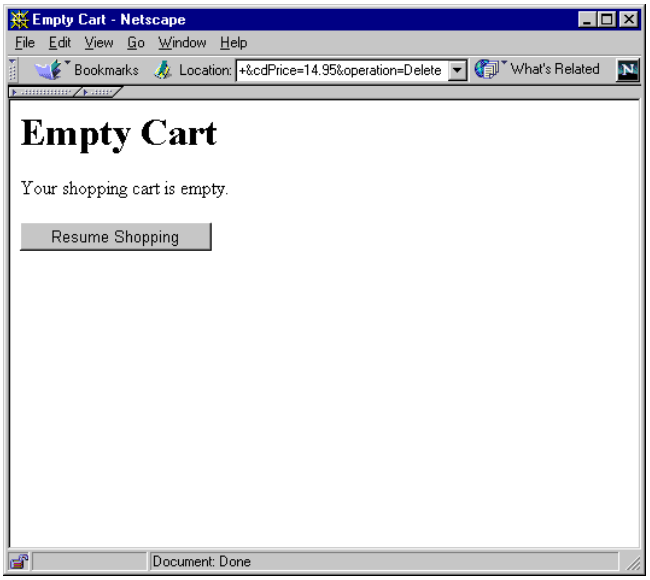


図 3-3　Empty Cart ページ

Empty Cart ページを作成するには、次のようにします。

1. CDShopCart Web モジュールを右クリックし、「新規」>「JSP & サーブレット」>「JSP」を選択します。
2. 「名前」フィールドに EmptyCart と入力し、「完了」をクリックします。

   エクスプローラに EmptyCart JSP ページが表示され、ソースエディタにそのソースコードが表示ます。
3. ページのタイトルを Empty Cart に変更し、次のようにコードを追加します。

```
<%@page contentType="text/html"%>
<html>
<head><title>Empty Cart</title></head>
<body>
    <H1> Empty Cart </H1>
    Your shopping cart is empty.
    <P>
    <form method=get action="ProductList.jsp">
    <input type=submit value="Resume Shopping">
    </FORM>
    </P>
    </body>
</html>
```

4. 「ファイル」>「保存」を選択して EmptyCart ページを保存します。

Place Order ページの作成

図 3-4　Place Order ページ

Place Order ページと Cancel Order ページは、非常に単純なページです。注文やキャンセルを行う方法は数多くありますが、そのうちの 1 つを示したものです。これらのページのプログラミングは、今までに紹介した Forte for Java 機能よりも若干高度な機能になるため、もっとも簡単な実装方法を紹介することにします。

Place Order ページが表示されるのは、Shopping Cart ページで「Place Order」ボタンを押したときです。このページを表示するとセッションが終了します。

Place Order ページを作成するには、次のようにします。

1. `CDShopCart` Web モジュールを右クリックして、「新規」>「JSP & サーブレット」>「JSP」を選択します。
2. 「名前」フィールドに `PlaceOrder` と入力し、「完了」をクリックします。

3. ページのタイトルを Place Order に変更し、次のようにコードを追加します。

```
<%@page contentType="text/html"%>
<html>
<head><title>Place Order</title></head>
<body>
    //見出しを追加する
    <H1> Place Order </H1>
    //セッションを無効にする
    <%
    session.invalidate();
    %>
    //メッセージを追加する
    Your order has been placed. Thank you for shopping.
    //スペースを追加する
    <P>
    //「Add」ボタンを追加する
    <FORM method=get action="ProductList.jsp">
    <INPUT type=submit value="Resume Shopping">
    </FORM>
    </P>
</body>
</html>
```

4. 「ファイル」>「保存」を選択して PlaceOrder ページを保存します。

Cancel Order ページの作成

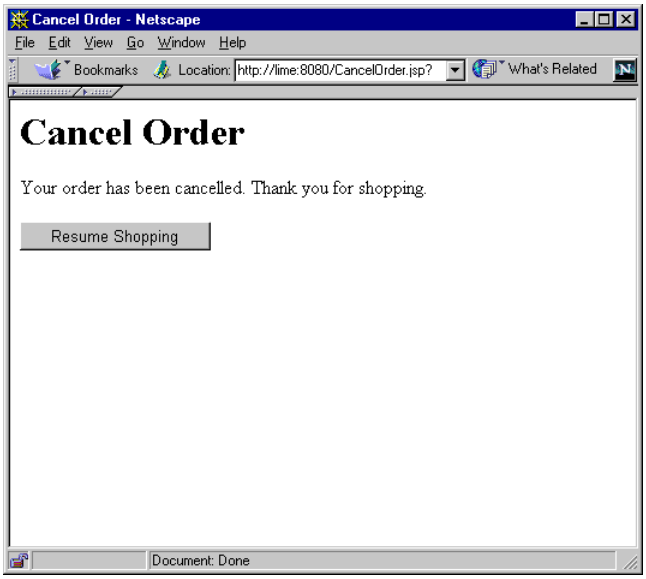


図 3-5　Cancel Order ページ

Cancel Order ページは、Shopping Cart ページ上の「Cancel Order」ボタンをクリックしたときに表示されます。このページを表示するとセッションが終了します。このページは、図 3-5 のようになります。

Cancel Order ページを作成するには、次のようにします。

1. `CDShopCart` Web モジュールを右クリックし、「新規」>「JSP & サーブレット」>「JSP」を選択します。
2. 「名前」フィールドに `CancelOrder` と入力し、「完了」をクリックします。

3. ページのタイトルを Cancel Order に変更し、Place Order ページと同様に、次のようにコードを追加します。

```
<%@page contentType="text/html"%>
<html>
<head><title>Cancel Order</title></head>
<body>
    <H1> Cancel Order </H1>
    <%
    session.invalidate();
    %>
    Your order has been cancelled. Thank you for shopping.
    <P>
    <FORM method=get action="ProductList.jsp">
    <INPUT type=submit value="Resume Shopping">
    </FORM>
    </P>
</body>
</html>
```

4. 「ファイル」>「保存」を選択して CancelOrder ページを保存します。

3 つのメッセージページのテスト実行

Shopping Cart ページと同様に、メッセージページをテストするには、ProduceList ページを実行し、ショッピングカートに CD を追加し、各メッセージページを表示する適切なアクションを行います。

1. CDShopCart Web モジュールを選択し、メニューから「構築」>「すべてを構築」を選択します。

   すべてのコンパイルが正常に完了するはずです。

2. ProductList JSP ページを右クリックし、「実行 (再読み込み強制)」を選択します。

   代わりに、「WEB-INF」ノードを右クリックし、「実行 (再読み込み強制)」を選択することもできます。

   数秒すると、CD Catalog List ページが表示されます。

3. いずれかの「Add」ボタンをクリックし、Shopping Cart ページに移動します。

4. Empty Cart ページをテストするには、今カートに入れた CD の「Delete」ボタンをクリックします。

   Empty Cart ページが表示されます。

5. 「Resume Shopping」ボタンをクリックし、CD Catalog List ページに戻ります。

6. 数枚の CD をカートに追加します。

7. 「Cancel Order」ボタンをクリックし、Cancel Order ページをテストします。

8. ページが表示されたら、「Resume Shopping」ボタンをクリックして CD Catalog List ページに戻ります。

9. 別の CD を 1 枚カートに追加します。

10. Shopping Cart ページが表示されたら、今追加した CD だけがカートに入っていることを確認します。

    「Cancel Order」によって直前のセッションが終了しているため、このカートには CD が 1 枚しか入っていないはずです。

11. 必要な枚数だけ CD をカートに追加し、「Place Order」ボタンを押します。

12. Place Order ページが表示されたら、「Resume Shopping」ボタンをクリックして CD Catalog List ページに戻ります。

13. 別の CD を 1 枚カートに追加します。

    手順 10 と同様、Place Order ページでいったんセッションが終了しているので、カートにはこの 1 枚の CD しか入っていないはずです。

14. アプリケーションを停止するには、「実行」ウィンドウ内のメッセージを右クリックし、「プロセスを終了」を選択します。

    Tomcat Web サーバーのプロセスが終了します。

第4章

チュートリアルアプリケーションでの透過的な持続性 (TP) の使用方法

この章では、(透過的な持続性 (Transparent Persistence: TP) を使用してデータベースとやりとりするための基本的な手法を、顧客の注文を保存することを例にして紹介します。この章を開始する前に、第 3 章 で説明した基本的な CDShopCart アプリケーションを作成しておく必要があります。

この章の内容は次のとおりです。

透過的な持続性の概要

Forte for Java の透過的な持続性は、『Java Data Objects Specification』の 持続技術のプレビュー版です。『Java Data Objects Specification』は公開レビュー中であり、`http://java.sun.com/aboutJava/communityprocess/review/jsr012/index.html` で参照することができます。『持続プログラミング』では、Forte for Java, Community Edition で提供されるこの技術を詳細に説明しています。

透過的な持続性の目的は、データストアに格納されている情報に Java オブジェクトとしてアクセスし、それによって、SQL やその他のデータストア固有の言語ではなく、Java プログラミング言語を使用して、データを操作できるようにすることです。透過的な持続性では、データベーススキーマ内の表を Java クラスに対応付け、Java クラスを持続可能クラスに機能拡張することによって、これを実現しています。

データベースへのアクセスは、標準的な Java 言語による操作と、これらの持続可能クラスに対する透過的な持続性メソッドの呼び出しを使用して行います。アプリケーションをコンパイルして実行するとき、透過的な持続性の実行時システムは、自動的に持続オペレーションを実行し管理します。

データベーススキーマ内の表から持続可能クラスを作成する方法には、2 つあります。1 つは、このスキーマ内にある特定の表から、自動的にクラス定義を作成する方法と、既存のクラスをスキーマ内にある特定の表にカスタマイズした状態で対応付ける方法です。

持続可能クラスでは、持続フィールドと一時フィールドの両方を使用することができます。持続フィールドは、持続データを表し、透過的な持続性の実行時環境によって管理されます。実行時環境は、現在のトランザクションのステータスおよび並行性管理戦略に従って、フィールド値とデータストアとの同期をとったり、クラス値をデータストアに書き出したりします。

一方、一時フィールドは、通常の Java 言語構造を使用しており、アプリケーションロジックによって管理され、透過的な持続性機構には関与しません。アプリケーションがこれらの一時フィールドを使用するのは、持続的なフィールドから別の値を派生させる場合や、データストアに保存する必要のない値をトランザクションで使用する場合などです。

詳細

透過的な持続性の詳細については、『持続プログラミング』とオンラインヘルプで説明しています。また、Forte for Java エクスプローラの「Javadoc」区画では、透過的な持続性メソッドに関する Javadoc のドキュメントを参照することができます。

透過的な持続性の使用方法

アプリケーション開発で 透過的な持続性を使用する場合は、2 段階に分かれます。第 1 段階では、データベーススキーマから持続可能クラスを作成し、第 2 段階では、透過的な持続性クラスのメソッドと実行時環境オブジェクトを使用してデータを操作します。

透過的な持続性メソッドを呼び出すための基本的な手順は、次のとおりです。

1. 持続マネージャファクトリを作成または取得します。

   持続マネージャファクトリは、データベース操作を管理する持続性マネージャを作成する場所です。持続マネージャファクトリは、その持続性マネージャが使用するデータベース接続テンプレートを作成します。

2. 持続性マネージャを作成します。

   持続性マネージャを作成する場合は、あらかじめ 持続マネージャファクトリを作成しておかなければなりません。通常、セッションごとに固有の持続性マネージャが作成されます。持続性マネージャは、持続マネージャファクトリによって定義されたデータベース接続を使用します。ただし、この接続は、持続性マネージャを無効にすることもできます。持続性マネージャは、トランザクションオブジェクトおよび照会オブジェクトの作成に使用するファクトリです。`query` クラスは、一連のデータベースの照会メソッドを提供します。

3. (オプション) 接続ファクトリを作成します。

   接続ファクトリが必要となるのは、接続プールを実装する場合だけです。詳細については、『持続プログラミング』を参照してください。

4. `Transaction` クラスのインスタンスを生成します。

   持続性マネージャのメソッドを使用して Transaction オブジェクトを作成し、Transaction メソッドを使用して持続性マネージャが管理するインスタンスに対してすべてのトランザクション操作を行います。

5. 透過的な持続性 API を使用してデータベースに接続し、トランザクションの起動と終了を行います。

   すべてのアプリケーションのビジネスロジック (データベースに対する照会と更新) は、トランザクションとトランザクションが起動されたデータベース内のコンテキスト内で行われます。

6. アプリケーションのビジネスロジックを起動します。

   アプリケーションは、データベースの照会、レコードの変更、新しいレコードの追加を行う際に、必要なデータを表す一連の持続インスタンスを生成します。持続性マネージャは、このようなインスタンスについて行われるデータベースとのやりとりをすべて管理します。

7. トランザクションをコミットするか強制的に終了します。

   データベースに更新を保存する場合は、トランザクションをコミットします。データベースをトランザクション開始前の状態にロールバックする場合は、トランザクションを強制的に終了します。コミットした後、削除したオブジェクトは一時オブジェクトになります。

8. さらにトランザクションを実行します。

   同じトランザクションオブジェクトを使用し、さらにロジックを実行したり、今実行したロジックを繰り返したりできます。また、別のトランザクションロジックを作成して使用することもできます。

9. データベースを閉じ、アプリケーションを終了します。

CDShopCart アプリケーションでの 透過的な持続性の使用

第 3 章で作成したアプリケーションでは、「Place Order」ボタンを押すと、注文を作成することはできますが、受注したことを示すメッセージだけが表示されるだけで、実際にその注文がデータベースに保存されるわけではありません。そこで、この章では、注文をデータベースに保存するコンポーネントを追加します。

Place Order 機能を実行する主な手順は、次のとおりです。

1. 持続可能クラスを作成し、シーケンス番号の生成と追跡を行い、注文された CD にそれらのシーケンス番号を割り当て、注文とその識別番号を格納します。

   78 ページの「持続可能クラスの作成」では、第 1 章でデータベースにインストールした表を使用します。

2. 持続可能クラスの機能拡張を行います。

   機能拡張とは、透過的な持続性が自動的にこのクラスに必要な JDBC 文を生成するプロセスを指します。クラスの機能拡張を行うには、IDE 内でクラスを実行するか、クラスをパッケージ化します。88 ページの「持続可能クラスの機能拡張」では、`CDShopCart` Web モジュールの `WEB-INF/lib` サブディレクトリ内で、JAR ファイルにそれらのクラスをパッケージ化します。

3. Bean コンポーネントを作成し、注文をデータベースに書き込むためのすべての透過的な持続性技術をこのコンポーネントにカプセル化します。

   92 ページの「持続性認識 Bean の作成」 では、次のアクションを実行する `CheckOutBean` Bean を作成します。

   a. 持続マネージャファクトリを作成します。

   b. 持続性マネージャを作成します。

      通常、持続マネージャファクトリは、アプリケーションレベルで作成し、持続性マネージャは、セッションレベルで作成します。ただし、このような小さなアプリケーションの場合は、簡単にするために、両方を同じコンポーネントで作成します。

   c. データベース接続を開きます。

   d. `Transaction` クラスのインスタンスを生成します。

   e. 識別番号を各注文と CD に割り当てるロジックを起動します。

   f. トランザクションをコミットします。これによって、更新がデータベースに反映されます。

   g. PlaceOrder JSP ページに注文番号を返します。

4. `CheckOutBean` を使用して注文を行い、注文番号を取得するように PlaceOrder ページを変更します。なお、この注文番号は、注文が行われたことを示す新しいメッセージの一部として表示します。

   この作業は、101 ページの「CheckOutBean を呼び出すように PlaceOrder ページを変更する」で行います。

5. アプリケーション全体をテストします。

   この作業は、102 ページの「新規 CDShopCart アプリケーションのテスト実行」で行います。

持続可能クラスの作成

Java クラスをデータベーススキーマに対応付けるには、スキーマを取り込む必要があります。スキーマを取り込むと、ファイルシステムに作業コピーが作成されるので、実際にデータベースに接続しなくても、この作業コピーを使用することができます。このスキーマ内にある表のうち、CDShopCart アプリケーション用に作成した 3 つの表をもとに、持続可能クラスを生成します。次に、このパッケージをコンパイルし、その機能拡張を行います。

このセクションでは、次の作業を行います。

1. 78 ページの「データベーススキーマの収集」
2. 83 ページの「持続可能クラスの生成」
3. 88 ページの「持続可能クラスの機能拡張」

データベーススキーマの収集

取り込んだスキーマは、パッケージ内に格納しなければなりません。このパッケージを、マウントしたファイルシステム内の `CDShopCart` Web モジュールの外で作成します。後でこのクラスを `CDShopCart` Web モジュール内で作成した JAR ファイルにパッケージ化することによって、クラスの機能拡張を行います。

チュートリアルデータベースのスキーマを取り込むには、次のようにします。

1. 「ファイル」>「ファイルシステムをマウント」を選択します。

   「ファイルシステムをマウント」ダイアログが表示されます。

2. ファイルシステムを作成する場所に移動します。

   この例では、ディスク C: 直下のディレクトリを使用しますが、どの場所でも使用できます。

3. 「フォルダの新規作成」ボタンをクリックします。

   このボタンをクリックすると、「新しいフォルダ」(または「名称未設定フォルダ」)という名前のフォルダが作成されます。

4. このフォルダの名前を `forSchema` に変更します。

5. forSchema を選択し、「了解」をクリックします。

   「ファイルシステムをマウント」ダイアログは、次のように表示されます。

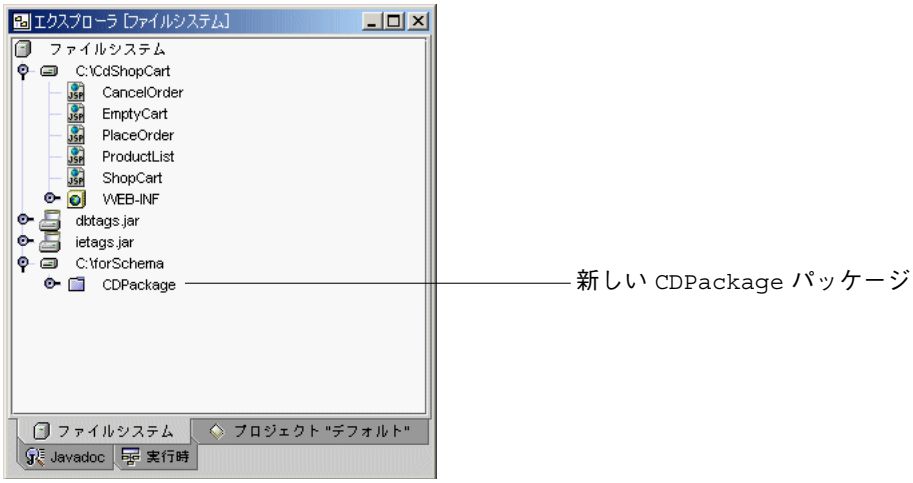
6. 「了解」をクリックします。

   エクスプローラに forSchema フォルダが表示されます。

7. 「forSchema」ノードをマウスの右ボタンでクリックし、コンテキストメニューから「新規パッケージ」を選択します。

8. 「名前」フィールドに CDPackage と入力し、「了解」をクリックします。

   エクスプローラに新しいパッケージが表示されます。



9. CDPackage パッケージを右クリックし、「新規」>「Databases」>「Database Schema」を選択します。

   「テンプレートウィザードから新規作成」の「新規オブジェクト名」区画が表示されます。

10. 「名前」フィールドに cdschema と入力し「次へ」をクリックします。

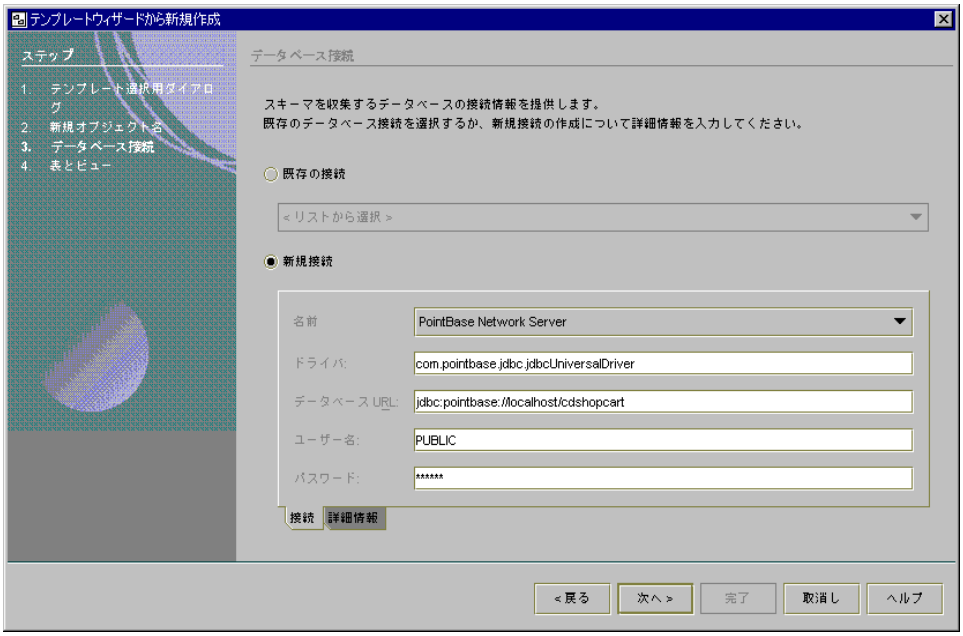
    「データベース接続」区画が表示されます。

11. 「新規接続」オプションを選択し、次の値を入力します。

    名前 : **PointBase Network Server** (ドロップダウンリストから選択します)
    ドライバ : **com.pointbase.jdbc.jdbcUniversalDriver**
    データベース URL : **jdbc:pointbase://localhost/cdshopcart**
    ユーザー名 : **PUBLIC**
    パスワード : **PUBLIC** (アスタリスクで表示されます)

    Oracle または Microsoft SQLServer を使用する場合は、42 ページの「JDBC の connection タグを使用したデータベースへの接続」で使用したものと同じデータを入力します。

ウィザードは、次のようになります。



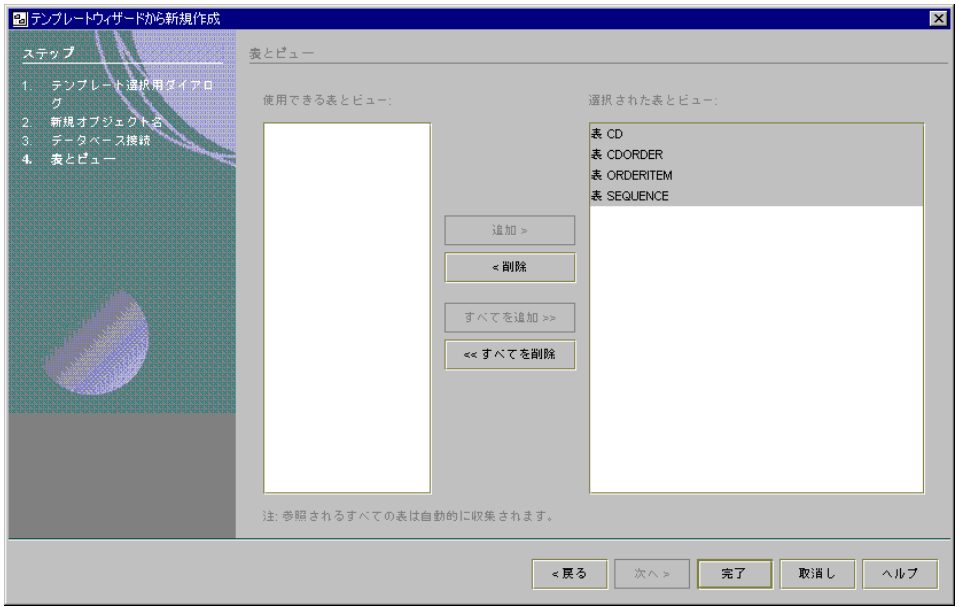
12. 「次へ」をクリックします。

「表とビュー」区画で、どの表を取り込むかを選択します。cdshopcart スキーマにすべての表を取り込みます。

13. 「すべてを追加」ボタンをクリックし、すべての表を選択します。

注 - CD、CDORDER、ORDERITEM、SEQUENCE 以外の表が表示されている場合は、「追加」ボタンを使用してこれらの 4 つの表だけを追加します。

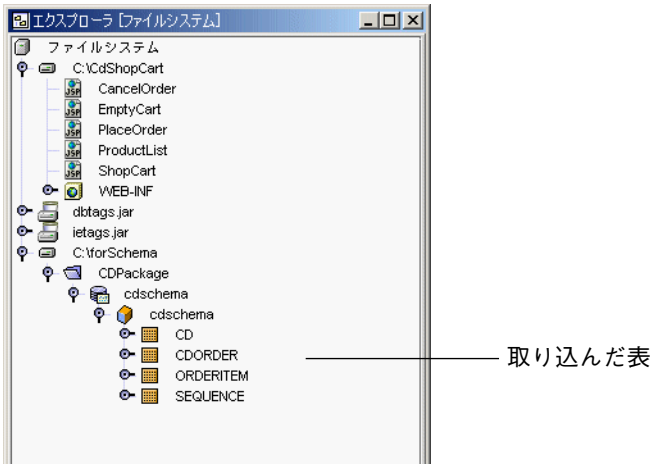
ウィザードは、次のようになります。



14. 「完了」をクリックします。

    進捗状況を示すウィンドウが表示されます。このウィンドウが閉じると、新しいデータベーススキーマが CDPackage パッケージに表示されます。

15. このパッケージとスキーマを開き、以下に示すように、取り込んだ表を表示します。



持続可能クラスを生成します。

持続可能クラスの生成

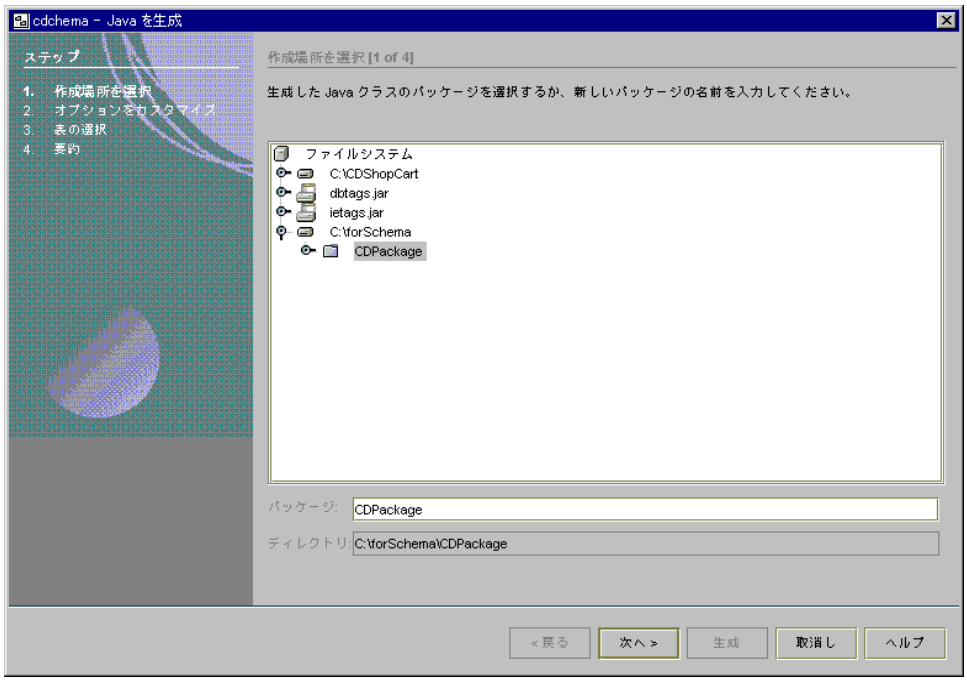
先ほど取り込んだデータベーススキーマ内のいくつかの表をもとに、持続可能クラスを生成します。

1. CDPackage パッケージの下にある「cdschema」ノード ( ) を右クリックし、コンテキストメニューから「Java を生成」を選択します。

   「Java を生成」ウィザードの「作成場所を選択」区画が表示されます。

2. forSchema ファイルシステムを展開し、CDPackage パッケージを選択します。

   ウィザードは、次のようになります。



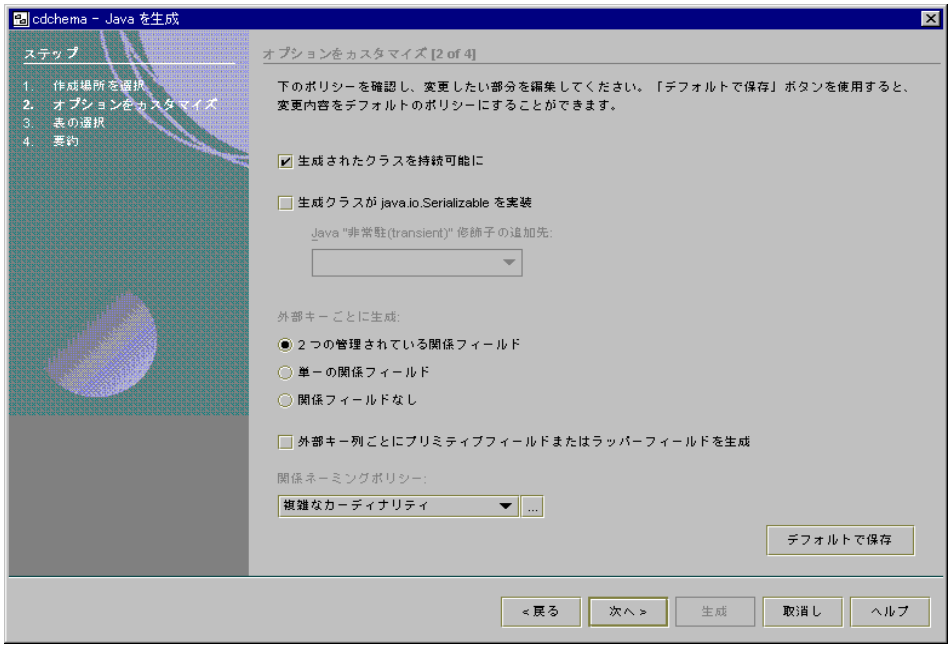
3. 「次へ」をクリックします。

   「Java を生成」ウィザードの「オプションをカスタマイズ」区画が表示されます。

4. 次のオプションを設定します。

   生成されたクラスを接続可能に : 有効
   生成されたクラスが java.io.Serializable を実装 : 無効
   外部キーごとに生成 :「2 つの管理されている関係フィールド」を選択
   外部キー列ごとにプリミティブフィールドまたはラッパーフィールドを生成 : 無効
   関係ネーミングポリシー :「複雑なカーディナリティ」を選択

   ウィザードは、次のようになります。



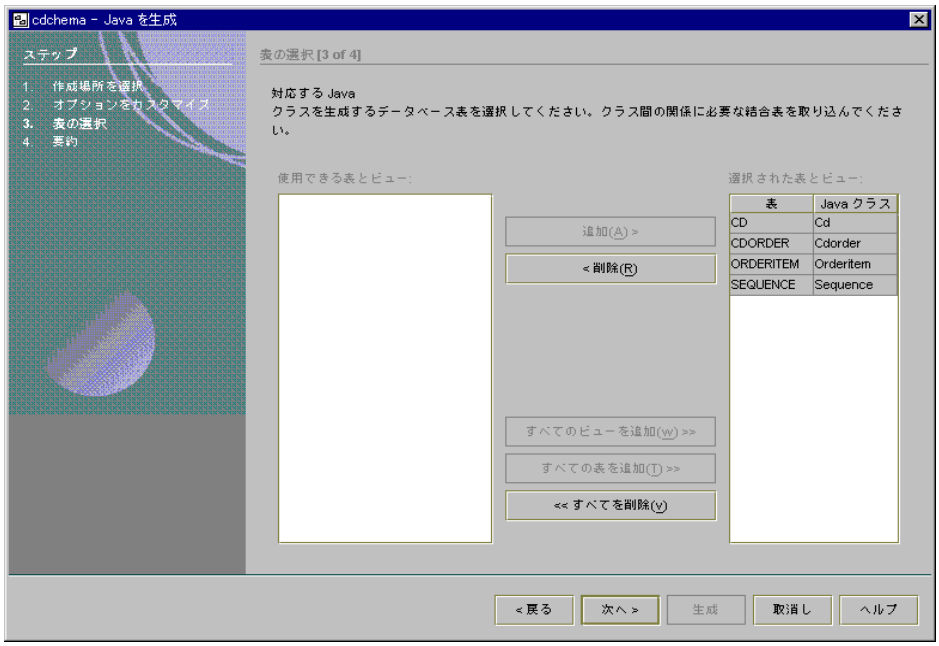
5. 「次へ」をクリックします。

   「Java を生成」ウィザードの「表の選択」区画が表示されます。

6. 「すべての表を追加」をクリックし、CD、CDORDER、ORDERITEM、SEQUENCE をダイアログの右の区画に追加します。

注 - CD、CDORDER、ORDERITEM、SEQUENCE 以外の表が表示されている場合は、「追加」ボタンを使用してこれらの 4 つの表だけを追加します。

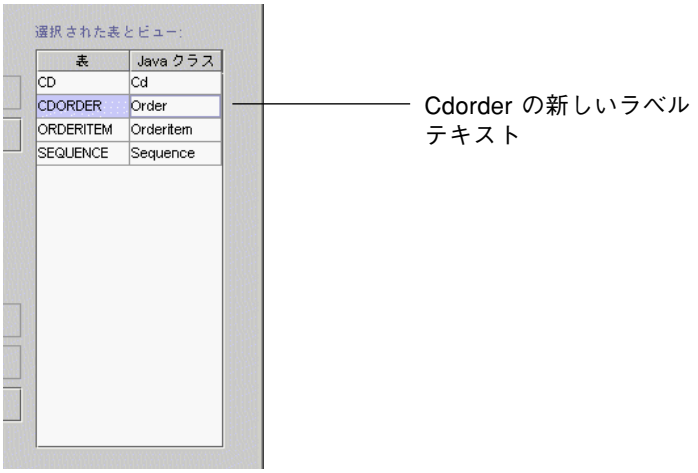
ウィザードは、次のようになります。



7. 「Java クラス」にある `Cdorder` ラベルをクリックして名前を `Order` に変更し、Enter キーを押します。

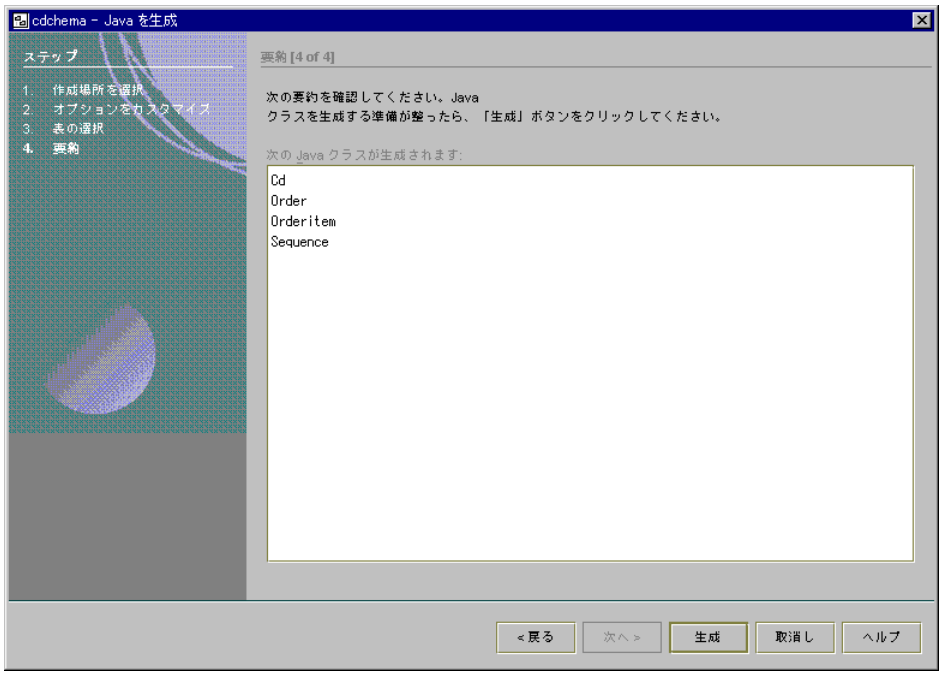
   この操作は、表との対応付けを維持したまま、クラス名を変更できることを示しています。

   「選択された表とビュー」は次のようになります。



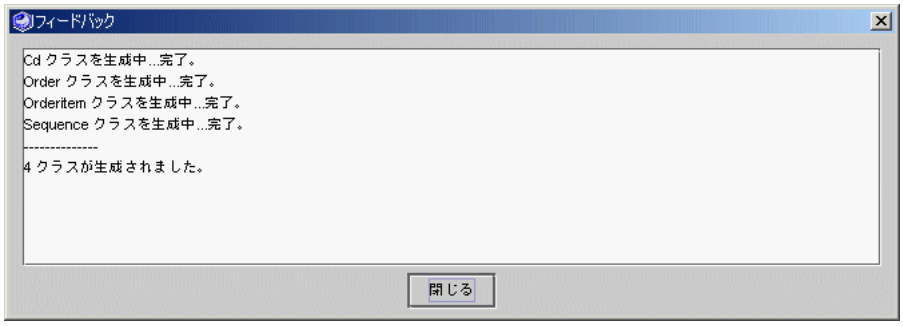
8. 「次へ」をクリックします。

   生成するクラスの要約が表示されます。



9. 「生成」をクリックします。

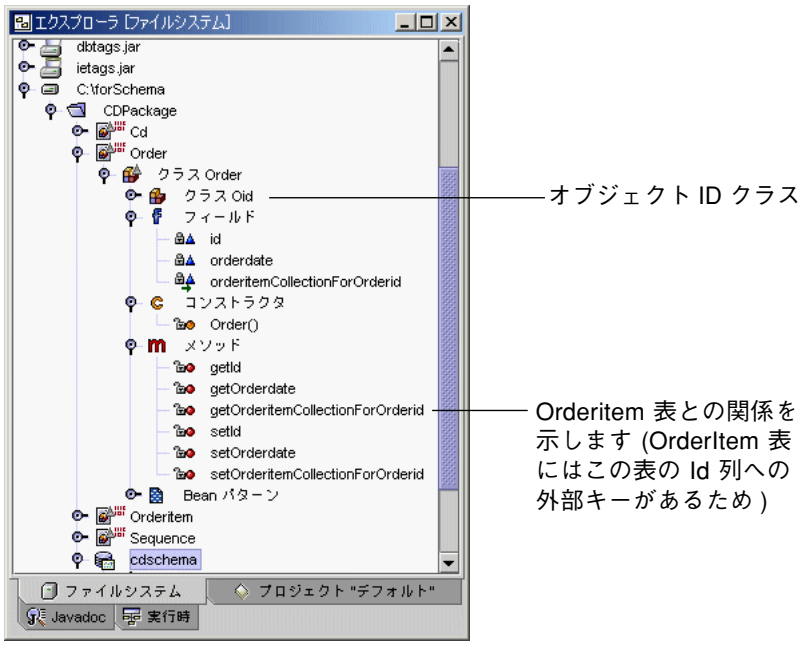
   「フィードバック」ウィンドウが表示され、クラスが生成されたことを示します。



10. 「閉じる」をクリックしてウィンドウを終了します。

11. エクスプローラで、新しく生成されたクラスを 1 つを開きます。

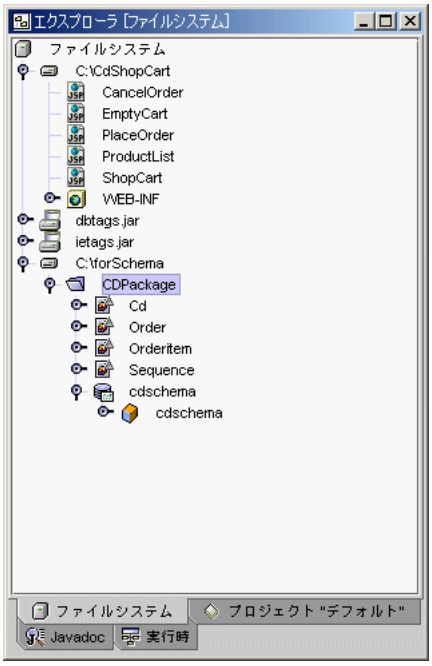
フィールドが表の列から生成されたことと、各フィールドに get および set の各メソッドが存在することを確認します。



生成された各クラスには、Oid (オブジェクト ID) クラスが関連付けられています。オブジェクト ID は、持続可能クラスのインスタンスを一意に識別するのに使用します。持続可能クラスの各インスタンスは、その識別子を保持している Oid クラスのインスタンスに関連付けられています。

12. CDPackage を右クリックし、「コンパイル」を選択してコンパイルします。

持続可能クラスに、クラスがまだコンパイルされていないことを表す赤い印がなくなっていることを確認します。



次に、持続可能クラスの機能拡張を行います。

持続可能クラスの機能拡張

クラスで透過的な持続性を使用するには、com.sun.forte4j.persistence.PersistenceCapable インタフェースを実装する必要があります。このインタフェースを使用すると、持続可能クラスが実行時環境と対話し、これらのクラスから生成されたインスタンスのステータスの確認とリセットを行える一連のメソッドを宣言することができます。クラスの機能拡張を行うプロセスによって、この持続可能 (PersistenceCapable) インタフェースを実装するコードが生成されます。

持続可能クラスを機能拡張するには、JAR ファイルにそれらのクラスをパッケージ化するか、Forte for Java IDE でそれらのクラスを実行します。CDShopCart アプリケーションでは、CDPackage パッケージを CDShopCart Web モジュール内にある JAR ファイルにパッケージ化することにします。

注 - Web モジュール内で持続可能クラスをパッケージ化する方法には 2 つあります。どちらの方法を使用するかは、持続可能クラスが頻繁に変更されるかどうかによって決定します。詳細はリリースノートを参照してください。

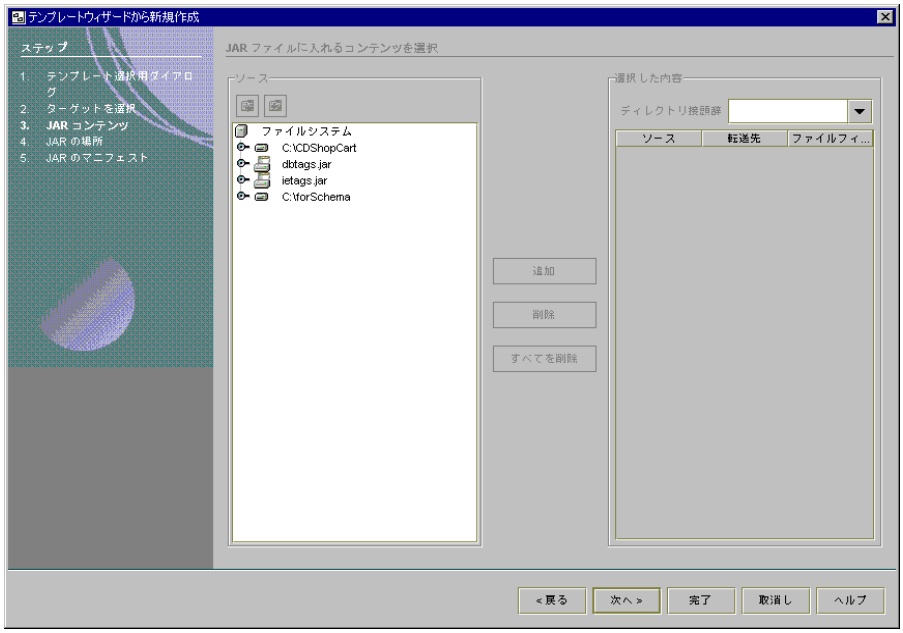
持続可能クラスの機能を拡張するには、次のようにします。

1. CDShopCart の「`WEB-INF`」ノード下の「`lib`」ノードを右クリックし、「新規」>「JAR パッケージャ」>「JAR コンテンツ」を選択します。

   「テンプレートウィザードから新規作成」の「新規オブジェクト名」区画が表示されます。

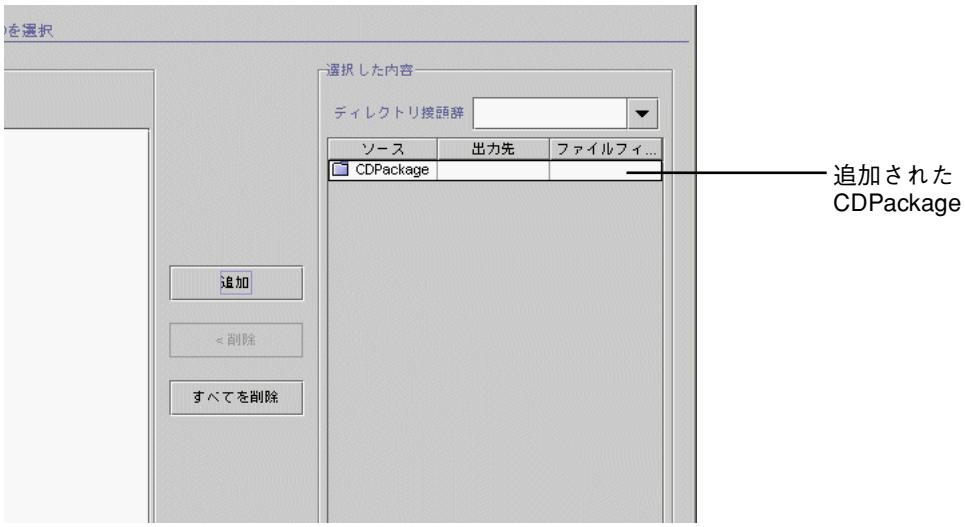
2. 「名前」フィールドに `CDclasses` と入力し、「次へ」をクリックします。

   JAR コンテンツを表す区画が表示されます。この区画では、次のように、新しい JAR ファイルのコンテンツを選択したり、既存の JAR ファイルのコンテンツを変更したりできます。



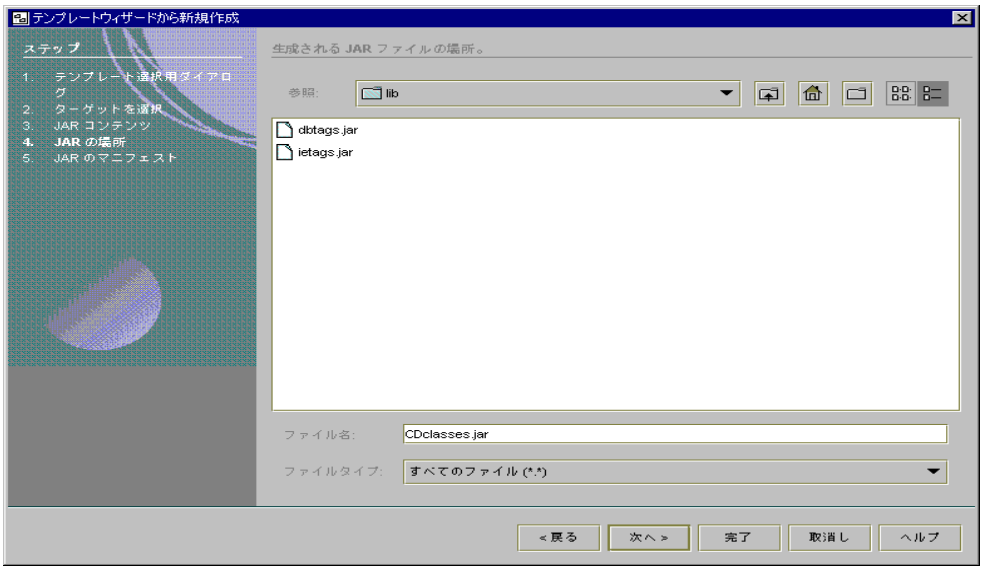
3. CDPackage パッケージを選択し、「追加」ボタンをクリックします。

   「選択した内容」区画の「ソース」に CDPackage が表示されます。



4. 「次へ」をクリックします。

   JAR ファイルの格納場所を選択するためのウィザードが表示されます。次のように、lib フォルダが「参照」フィールドに、CDclasses.jar が「ファイル名」フィールドに表示されます。

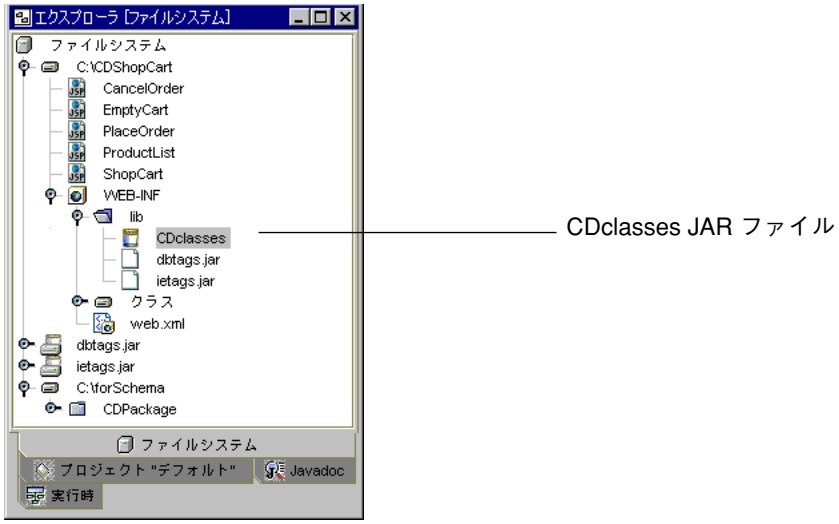


5. 「次へ」をクリックします。

   JAR マニフェストを作成する「manifest Tab」区画が表示されます。

6. 「完了」をクリックして JAR パッケージを生成します。

   次のように、CDPackage の JAR ファイルがエクスプローラ内の WEB-INF/lib フォルダに表示されます。



CLASSPATH でのファイルシステムの移動

CDPackage 内の古い持続可能クラス (機能を拡張されていないということ) が、CLASSPATH で CDShopCart ファイルシステムより下位のファイルシステムであることを確認します。この確認は必ず行う必要があります。これにより、機能が拡張されていないクラスの代わりに、機能が拡張されたクラスをコンパイラが使用するようになります。

1. エクスプローラで「ファイルシステム」を右クリックし、「カスタマイズ」を選択します。

   「カスタマイザダイアログ」が表示されます。リストから CDShopCart と forSchema ファイルシステムを見つけます。

2. forSchema ファイルシステムが、CDShopCart ファイルシステムより上位にある場合は、forSchema ファイルシステムを右クリックして「下へ移動」を選択します。

   forSchema ファイルシステムが CDShopCartファイルシステム (および CDShopCart:/WEB-INF/classes ファイルシステム) より下位にある場合は、何もする必要はありません。

3. 「カスタマイザダイアログ」を閉じます。

次に、透過的な持続性をカプセル化する CheckOutBean Bean を作成します。

持続性認識 Bean の作成

次に、透過的な持続性機能をカプセル化するコンポーネントである CheckOutBean Bean を作成します。CheckOutBean Bean は、持続マネージャファクトリを作成し、データベース接続に使用するプロパティを指定します。次に、持続性マネージャを作成します。ユーザーは、持続性マネージャのメソッドを使用し、transaction オブジェクトと query オブジェクトを作成します。また、query クラスのメソッドを使用し、データベースの照会機能を実行します。

1. CDShopCart Web モジュールの「WEB-INF」フォルダを開きます。
2. 「クラス」フォルダを右クリックし、「新規」>「Beans」>「Bean」を選択します。
3. 「名前」フィールドに CheckOutBean と入力し、「完了」をクリックします。

   エクスプローラに新しい CheckOutBean Bean が表示され、ソースエディタにそのコードが表示されます。

   CheckOutBean Bean のコードに、次のような変更を行います。

   1. 持続マネージャファクトリと持続性マネージャを初期化します。
   2. ID をもとに CD を取り込みます。
   3. カート内の CD ごとに注文とその注文の明細を追加します。
   4. 次の注文のシーケンス番号を取得します。

持続マネージャファクトリと持続性マネージャの初期化

大きなアプリケーションでは、アプリケーションレベルで 持続マネージャファクトリを作成し、セッションごとに持続性マネージャを作成しますが、今回のような小さいアプリケーションの場合は、同じセッション内で 持続マネージャファクトリと持続性マネージャの両方を作成します。

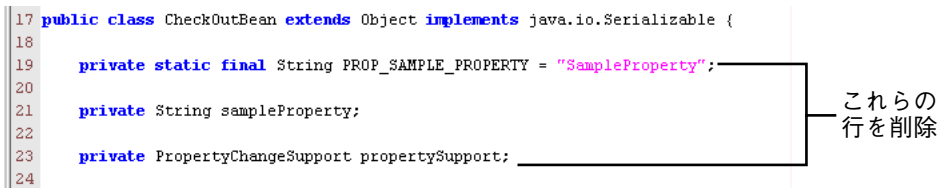
持続マネージャファクトリと持続性マネージャを初期化するには、次のようにします。

1. エディタで、次のように 3 つの新しいインポート文を追加します。

```
import java.beans.*;
import com.sun.forte4j.persistence.*;
import java.util.*;
import CDPackage.*;
```

com.sun.forte4j.persistence ライブラリは、透過的な持続性を使用するために必要です。java.util ライブラリは、コレクションを使用するのに必要です。CDPackage は、先に作成した持続可能クラスで使用するのに必要です。

2. 次のように、CheckOutBean 宣言から最初の 3 つの文を削除します。



3. これらの行を次のコードに置き換えます。

```
private PersistenceManager pm;
```

このコードは、持続性マネージャオブジェクトを宣言します。

4. CheckOutBean のコンストラクタのコードを次のコードに置き換え、持続マネージャファクトリの初期化と持続性マネージャの作成を行います。

```
public CheckOutBean() {
    // 持続マネージャファクトリをインスタンス化する
    PersistenceManagerFactory pmf = new
            PersistenceManagerFactoryImpl();
    // 持続性マネージャが使用する
    // データベース接続を指定する
    pmf.setConnectionUserName("PUBLIC");
    pmf.setConnectionPassword("PUBLIC");
    pmf.setConnectionDriverName(
        "com.pointbase.jdbc.jdbcUniversalDriver");
    pmf.setConnectionURL("jdbc:pointbase://localhost/cdshopcart");
    // すべての持続性マネージャトランザクションについて
    // デフォルトの並行性制御を指定する
    pmf.setOptimistic(false);
    //持続性マネージャを作成する
    this.pm = pmf.getPersistenceManager();
}
```

注 - 別のデータベースを使用している場合は、上記の値を使用しているデータベースの標準的な JDBC 接続文字列の値に置き換えます。

---

5. 「ファイル」>「保存」を選択して作業を保存します。

詳細

持続マネージャファクトリおよび持続性マネージャの作成とメソッドについては、『持続プログラミング』と、透過的な持続性 (Transparent Persistence) のオンラインヘルプを参照してください。

ID をもとにした CD の取り込み

次に、getCd メソッドを作成します。このメソッドでは、Query オブジェクトをインスタンス化し、そのメソッドを使用して PlaceOrder ページから渡された Id をもとに CD を取り込みます。

1. CheckOutBean 内にある残りのデフォルトコードをすべて削除します。
2. エクスプローラで、「CheckOutBean」の「メソッド」ノードを右クリックし、「新規メソッド」を選択します。
3. 「新規メソッドを編集」ダイアログに、次の値を入力します。

   「メソッド」
   名前 : **getCd**
   戻り値 : **Cd**
   アクセス : **public**

   「メソッドのパラメータ」
   型 : **long**
   名前 : **id**
4. 「了解」をクリックしてメソッドを作成します。

5. エディタで、次のコードをメソッド本体に入力します。

```
public Cd getCd(long id) {
    Query q = this.pm.newQuery();
    //結果のクラス (Cd) を照会インスタンスにバインドする
    q.setClass(Cd.class);
    // 照会の入力コレクションを定義する
    q.setCandidates(pm.getExtent(Cd.class, false));
    // 照会フィルタを照会インスタンスにバインドする
    q.setFilter("id == CDid");
    //プレースホルダのパラメータを定義する
    String param = "Long CDid";
    // パラメータを照会文にバインドする
    q.declareParameters(param);
    // 照会を実行してフィルタリングされたコレクションを返す
    Collection result = (Collection)q.execute(new Long(id));
    // 結果の中を反復して指定の CD を見つける
    Iterator i = result.iterator();
    Cd theCd = null;
    if (i.hasNext()) {
        theCd = (Cd)i.next();
    }
    return theCd;
}
```

6. 作業を保存します。

このコードの照会には、どの照会にも必要な次の 3 つの文が含まれます。

- 結果を格納するクラス (setClass メソッドによって設定される)
- 候補となるインスタンスのコレクション (setCandidates メソッドによって設定される)

これは、java.util.Collection またはデータストア内のインスタンスのエクステントです。

- 照会フィルタ (setFilter メソッドによって設定される)

id は、持続可能クラス CD 内にあるフィールドであり、CDid は、照会パラメータ名を示します。

照会メソッドの機能は、SQL 文 の「SELECT * FROM CD WHERE ID = ?」とほぼ同じものです。

詳細

照会インタフェースのメソッドについては、『持続プログラミング』を参照してください。簡単な説明については、エクスプローラの「Javadoc」タブにある「Persistence.zip」フォルダの「Query」インタフェースを参照してください。

カート内の CD ごとに注文と明細を追加する

次の作業を実行する checkout メソッドを作成します。

1. トランザクションを起動します。
2. 注文シーケンス番号と関連するデータを CDORDER 表から取得します。
3. カート明細、各カート明細に対応する製品番号、checkout メソッドに渡されたカート順に対応する製品を生成します。
4. データベースに対し、このデータをコミットします。
5. 注文シーケンス番号を返します。

checkout メソッドを作成するには、次のようにします。

1. 次の値を使用して CheckOutBean の新規メソッドを作成します。

   「メソッド」
   名前 : **checkout**
   戻り値 : **int**
   アクセス : **public**

   「メソッドのパラメータ」
   型 : **Cart**
   名前 : **myCart**

2. checkout メソッドの本文に次のコードを入力します。

```
Transaction tx = pm.currentTransaction();
tx.begin();
```

このコードでは、持続性マネージャのメソッドを使用してトランザクションを作成してから、トランザクションを起動します。

3. 前述のコードの後に、次のコードを追加して注文を作成します。

```
//次の注文シーケンス番号を取得する
int ordNum = this.getSequenceNumber("CDORDER", 1);
//新しい注文を作成する
Order ord = new Order();
//注文の主キーを設定する
ord.setId(ordNum);
//現在の日付を設定する
ord.setOrderdate(new Date());
//データベース更新のマークを付けるように持続性マネージャに通知する
pm.makePersistent(ord);
```

4. 前述のコードの下に、カート内の CD ごとに、カート明細を注文に追加するコードを追加します。

```
//明細番号を初期化する
int itemNum = 1;
//新しいハッシュリストを作成して全カート明細（CD）を格納する
HashSet itemList = new HashSet();
Iterator i = myCart.lineItems.iterator();
while (i.hasNext()) {
    //カート内の次の CD
    CartLineItem c = (CartLineItem)i.next();
    //新規明細を作成する
    Orderitem item = new Orderitem();
    //主キーを設定する
    item.setOrderid(ordNum);
    item.setLineitemid(itemNum++);
    item.setCdOfProductid(getCd(c.getId()));
    //コレクションに追加する
    itemList.add(item);
    //データベース更新のマークを付けるように持続性マネージャに通知する
    pm.makePersistent(item);
}
```

5. このコードの後に、以下のコードを入力してこのトランザクションをコミットし、注文番号を返します。

```
tx.commit();
return ordNum;
```

6. 作業を保存します。

次の注文のシーケンス番号の取得

この節では、getSequenceNumber メソッドを作成します。これは、CheckOutBean の最後のメソッドです。

getSequenceNumber メソッドを作成するには、次のようにします。

1. 次の値を使用して CheckOutBean の新規メソッドを作成します。

   「メソッド」
   名前 : **getSequenceNumber**
   戻り値 : **int**
   アクセス : **public**

   「メソッドのパラメータ」
   型 : **java.lang.String**
   名前 : **tableName**

   「メソッドのパラメータ」
   型 : **int**
   名前 : **amount**

2. エディタで、getSequenceNumber メソッドに次のコードを追加し、キーを初期化します。

```
int key = 0;
```

3. 次に、以下のコードを入力し、データベースを照会してデータベースに格納されているカート順を表すシーケンス番号を取り込みます。

   getSequence() メソッド内の sequenceQuery は、指定した表の新しい主キーを生成します。これは、CDShopCart アプリケーションにとって一意のキーです。

```
//新しい照会を作成する
Query sequenceQuery = pm.newQuery();
// 結果クラス (Sequence) を照会インスタンスにバインドする
sequenceQuery.setClass(Sequence.class);
// 入力コレクションを定義する。フィルタ tablename はフィールドであり、
// name はパラメータ名である
sequenceQuery.setCandidates(pm.getExtent(Sequence.class, false));
sequenceQuery.setFilter("tablename == name");
// パラメータを定義する
String param = "String name";
// パラメータを照会文にバインドする
sequenceQuery.declareParameters(param);
// 照会を実行してフィルタリングされたコレクションを返す
Collection result = (Collection)sequenceQuery.execute(tableName);
Iterator i = result.iterator();
Sequence s = (Sequence)i.next();
// 持続性マネージャが次のキーを更新できるようにする
key = s.getNextpk().intValue();
s.setNextpk(new Integer(key + amount));
// キーを呼び出し側に戻す
return key;
```

注 - Oracle データベースを使用している場合は、最後から 2 番目の行で Integer の代わりに Long を使用します。

---

4. CheckOutBean Bean を右クリックし、「コンパイル」を選択して Bean をコンパイルします。

CheckOutBean を呼び出すように PlaceOrder ページを変更する

後は、PlaceOrder JSP ページを修正して、CheckOutBean Bean を呼び出し、CheckOutBean Bean から返された注文番号を表示するだけです。この処理を行うために、次の要素を追加します。

1. page 指令を追加し、Cart Bean と CheckOutBean Bean をインポートします。
2. usebean タグを追加し、ページで Cart Bean および CheckOutBean Bean を使用できるようにします。
3. 注文番号を宣言します。
4. 注文番号を CheckOutBean が返した番号に設定します。
5. 注文番号が表示されるようにメッセージを修正します。

PlaceOrder JSP ページを修正するには、次のようにします。

1. page 指令を使用し、java.util ライブラリ、Cart Bean、CartLineItem Bean をインポートします。

```
<%@page contentType="text/html" %>
<%@page import="java.util.*, Cart, CartLineItem" %>
```

2. 見出し (h1 タグ) の下に useBean タグを追加します。

```
//Cart Bean を使用する
<jsp:useBean id="myCart" scope="session" class="Cart" />
//CheckOutBean Bean を使用する
<jsp:useBean id="checker" scope="session" class="CheckOutBean" />
```

3. これらの文の下に、次のコードを追加します。

```
<%! int ordNum; %>

<%
ordNum = checker.checkout(myCart);
session.invalidate();
%>
```

2 番目の文は、ordNum を checkout メソッドから返された注文番号に設定します。

4. 最後に、次のように、表示されるメッセージを ordNum を使用するように変更します。

```
Your order has been placed. For future reference, your order number
is <%=ordNum%>.

Thank you for shopping.
```

5. PlaceOrder JSP ページを右クリックして「コンパイル」を選択します。

新規 CDShopCart アプリケーションのテスト実行

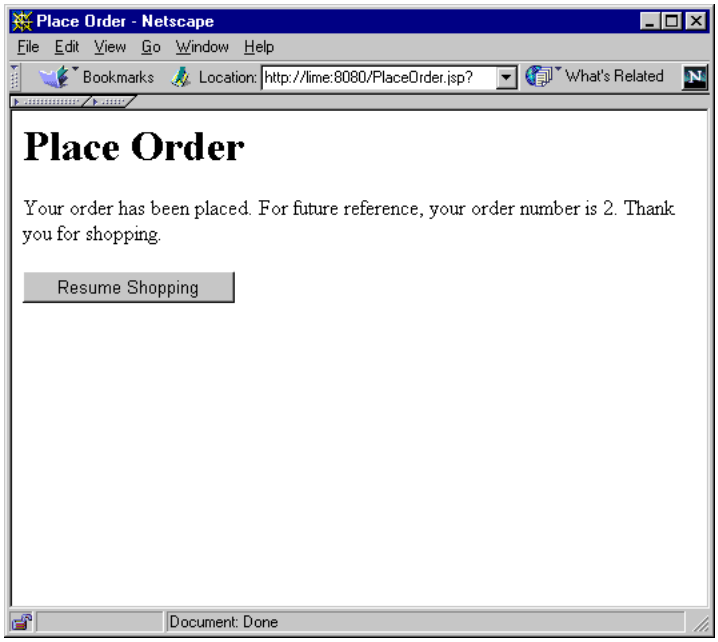
基本的な CDShopCart アプリケーションのテスト (70 ページの「3 つのメッセージページのテスト実行」を参照) と同様に、このアプリケーションもテストします。ProductList ページを実行し、CD をショッピングカートに追加してから、各種のアクションを実行します。これらのアクションによって、さまざまなページが表示されます。ここでは、注文を行うアクションを選択します。注文を行うことによって、ここで説明した CheckOutBean が呼び出され、その結果、生成された注文番号が PlaceOrder ページのメッセージに表示されます。これは、CheckOutBean が正しく動作したことを示します。

メッセージページをテストするには、次のようにします。

1. CDShopCart Web モジュールを選択し、メニューから「構築」>「すべてを構築」を選択します。

すべてのコンパイルが正常に完了するはずです。

2. ProduceList JSP ページを右クリックし、「実行 (再読み込み強制)」を選択します。
   数秒すると、CD Catalog List ページが表示されます。
3. いずれかの「Add」ボタンをクリックし、Shopping Cart ページに移動します。
4. 「Resume Shopping」ボタンをクリックし、CD Catalog List ページに戻ります。
5. 任意の枚数の CD をカートに追加し、「Place Order」ボタンをテストします。
6. Place Order ページは、次のように表示されます。



CDShopCart チュートリアルアプリケーションの作成は、これで完了です。

索引

D

E

F

H

I

J

L

M

N

O

P

Q

R

S

T

W

と

ひ

ふ

ゆ