Sun™ ONE Studio 4, Community Edition チュートリアル

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 817-0842-10
2002 年 9 月 Revision A

原典： *Sun ONE Studio 4, Community Edition Tutorial*
Part No: 816-7868-10
Revision A

目次

図目次

表目次

はじめに

このチュートリアルでは、Sun™ Open Net Environment (Sun ONE) Studio 4, Community Edition に導入されている以下の機能の使用方法を学びます。

- Java™ Servlet および JavaServer Pages™ (JSP™) 技術を使用した Web アプリケーションのサポート
- Jakarta プロジェクトの JSTL (JSP Standard Tag Library) 基準実装を使用したデータベースアクセス

このマニュアルで説明しているプログラム例は、実際に作成することができます。作業環境については、以下の Web サイトにあるリリースノートを参照してください。

`http://sun.co.jp/forte/ffj/documentation/index.html`

使用するプラットフォームによっては、このマニュアルに掲載している画面イメージと異なることがあります。その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 Sun ONE Studio 4 ソフトウェア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、Microsoft Windows の「コマンドプロンプトウィンドウ」で次の構文を入力します。

```
c:\>cd MyWorkSpace\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 MyWorkSpace/MyPackage
```

お読みになる前に

このチュートリアルでは、データベースと対話して、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。その設計とアーキテクチャは、J2EE (Java 2™ Platform, Enterprise Edition) Blueprints に準拠しています。このチュートリアルは、Sun ONE Studio 4, Community Edition の機能を利用して 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。

このチュートリアルを学ぶにあたっては、次の知識があることが前提になります。

- 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
- Jakarta プロジェクトの JSTL (JSP Standard Tag Library)
- 表やキーなどの、リレーショナルデータベースの概念
- 利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また、Web アプリケーションをはじめとする Java サーブレットや JavaServer Pages に関する知識も必要です。これらの概念は、次の資料に定義されています。

- Java Servlet Specification Version 2.3
  `http://java.sun.com/products/servlet/download.html#specs`
- JavaServer Pages Specification Version 1.2
  `http://java.sun.com/products/jsp/download.html#specs`

JSTL チュートリアルおよびその他の有用な情報のリンクについては、以下をご覧ください。
`http://jakarta.apache.org/taglibs/tutorial.html`

注 - Sun では、本マニュアルに掲載した第三者の Web サイトのご利用に関しましては責任はなく、保証するものでもありません。また、これらのサイトあるいはリソースに関する、あるいはこれらのサイト、リソースから利用可能であるコンテンツ、広告、製品、あるいは資料に関して一切の責任を負いません。Sun は、これらのサイトあるいはリソースに関する、あるいはこれらのサイトから利用可能であるコンテンツ、製品、サービスのご利用あるいは信頼によって、あるいはそれに関連して発生するいかなる損害、損失、申し立てに対する一切の責任を負いません。

内容の紹介

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、このチュートリアルの CDShopCart アプリケーションに必要なソフトウェア、チュートリアルデータベース表のインストール方法、Sun ONE Studio 4 統合開発環境 (IDE) の起動方法、IDE が使用している Web サーバーの確認方法を説明しています。またこの章では、インストールした Sun ONE Studio 4 のディレクトリを表にまとめています。

第 2 章 では、このチュートリアルの CDShopCart アプリケーションのアーキテクチャについて説明しています。

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ています。 このアプリケーションは、音楽用 CD をオンラインで購入するための単純なショッピングカートアプリケーションです。

付録 A では、このチュートリアルアプリケーションのソースファイルの全内容をまとめています。

付録 B では、このチュートリアルアプリケーションのデータベーススクリプトファイルの全内容をまとめています。

書体と記号について

書体または記号	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、コーディング例。	.cvspass ファイルを編集します。 DIR を使用してすべてのファイルを表示します。 Search is complete.
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表わします。	> **login** Password:
AaBbCc123 または ゴシック	コマンド行の可変部分。実際の名前または実際の値と置き換えてください。	削除するには **DEL** *filename* と入力します。 rm **ファイル名** と入します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照してください。 これらは、「クラス」オプションと呼ばれます。
\	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、バックスラッシュは、継続を示します。	machinename% grep '^#define \ XV_VERSION_STRING'
➤	階層メニューのサブメニューを選択することを示します。	作成:「返信」➤「送信者へ」

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	`machine_name%`
UNIX の Bourne シェルと Korn シェル	`machine_name$`
スーパーユーザー (シェルの種類を問わない)	#

関連マニュアル

Sun ONE Studio 4 のマニュアルは、Acrobat Reader (PDF) ファイル、リリースノート、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

以下に紹介するマニュアルは、Sun ONE Studio 4 のドキュメントサイト (http://sun.co.jp/forte/ffj/documentation/index.html) および `docs.sun.com`™ (http://docs.sun.com) から入手できます。

`docs.sun.com` ウェブサイトでは、サンのマニュアルをインターネットを通じて閲覧、印刷、購入することができます。サイト内でマニュアルを見つけられない場合には、製品と一緒にローカルシステムまたはローカルネットワークにインストールされているマニュアルインデックスを参照してください。

- リリースノート (HTML 形式)

  Sun ONE Studio 4 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)

  対応プラットフォームへの Sun ONE Studio 4 統合開発環境 (IDE) 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。

  - 『Sun ONE Studio 4, Community Edition インストールガイド』
    - Part No. 817-0845-10
  - 『Sun ONE Studio 4, Enterprise Edition for Java インストールガイド』
    - Part No. 817-0844-10
  - 『Sun ONE Studio 4, Mobile Edition インストールガイド』
    - Part No. 817-0846-10

- Sun ONE Studio 4 プログラミングシリーズ (PDF 形式)

  Sun ONE Studio 4 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. 817-0837-10

    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。

  - 『J2EE アプリケーションのプログラミング』 - Part No. 817-0839-10

    EJB モジュールや Web モジュールを J2EE にアセンブルする方法を説明しています。また、J2EE アプリケーションの配備や実行についても説明しています。

  - 『Enterprise JavaBeans コンポーネントのプログラミング』
    - Part No. 817-0838-10

    Sun ONE Studio 4 の EJB ビルダーウィザードや、他の IDE コンポーネントを使用し、EJB 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、メッセージ駆動型 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. 817-0816-10

  Sun ONE Studio 4 IDE を使用して Web サービスを構築したり、UDDI レジストリを経由して第三者に Web サービスを利用させたり、また、ローカル Web サービスや UDDI レジストリから Web サービスクライアントを生成する方法などを説明しています。

- 『Java DataBase Connectivity の使用』 - Part No. 817-0840-10

  Sun ONE Studio 4 IDE の JDBC 生産性向上ツールを使用し、JDBC アプリケーションを作成する方法について説明しています。

- Sun ONE Studio 4 チュートリアル (PDF 形式)

  Sun ONE Studio 4 の Edition ごとに用意されており、主な機能の活用方法を紹介しています。

  - 『Sun ONE Studio 4, Community Edition チュートリアル』
    - Part No. 817-0842-10

    簡単な J2EE Web アプリケーションの構築方法を順を追って解説します。

  - 『Sun ONE Studio 4, Enterprise Edition for Java チュートリアル』
    - Part No. 817-0841-10

    EJB コンポーネントと Web サービス技術を使用したアプリケーションの構築方法を順を追って解説します。

  - 『Sun ONE Studio 4, Mobile Edition チュートリアル』
    - Part No. 817-0843-10

    携帯やPDA 端末などの無線機器を対象とした簡単なアプリケーションの構築方法を順を追って解説します。このアプリケーションは Java 2 Platform, Micro Edition (J2ME™ プラットフォーム) に準拠し、Mobile Information Device Profile (MIDP) と Connected, Limited Device Configuration (CLDC) を満たすものです。

  チュートリアルアプリケーションは、以下のサイトからもアクセスできます。
  `http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

オンラインヘルプ

オンラインヘルプは、Sun ONE Studio 4 IDE から参照できます。ヘルプを起動するには、ヘルプキー (Windows および Linux 環境では F1 キー、Solaris オペレーティング環境では Help 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

Sun ONE Studio 4 の機能を紹介したプログラム例とチュートリアルアプリケーションを、以下の Sun ONE Studio Developer Resource のポータルサイトからダウンロードすることができます。

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

このチュートリアルで使用するアプリケーションも上記サイトに収録されています。

Javadoc

Javadoc 形式のマニュアルは、Sun ONE Studio 4 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。

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、マニュアルの Part No. (817-0842-10) を明記してください。

第1章

チュートリアルを学ぶにあたって

この章では、Sun ONE Studio 4, Community Edition のチュートリアルを学ぶにあたって準備しておく必要がある事柄をまとめています。マニュアルなどを参照しなくてもよいよう、このチュートリアルには、『Sun ONE Studio 4, Community Edition インストールガイド』と重複するインストール情報が含まれています。この章の内容は次のとおりです。

注 - このマニュアルには、「CDShopCart アプリケーションファイル」の名前を参照している箇所が出てきます。それらのファイルは、完成したチュートリアルアプリケーション、そのアプリケーションの実行方法を説明した readme ファイル、必要なデータベース表を作成するための SQL スクリプトファイル (複数のバージョンがある) などのことです。これらのファイルを 1 つの ZIP 形式のファイルにまとめたものが Sun ONE Studio 4 Developer Resources ポータルサイト (`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`) にあり、ダウンロードすることができます。

必要なソフトウェア

この節では、このチュートリアルを学ぶにあたって用意しておく必要があるソフトウェアをまとめています。Sun ONE Studio 4 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をすべて揃えておく必要があります。

全般的なシステム要件については、リリースノートまたは Sun ONE Studio 4 Developer Resources ポータルサイトの Documentation ページ (`http://forte.sun.com/ffj/documentation/`) をご覧ください。

Sun ONE Studio 4 IDE のシステム要件

Sun ONE Studio 4 IDE を使用するには、Java™ 2 Software Development Kit (J2SE SDK) が必要です。IDE のインストールを開始すると、システムに J2SE SDK ソフトウェアを検索し、適切なバージョンがシステムに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。Java 2 SDK の適切なバージョンは、Java Developer ポータルサイト (`http://java.sun.com/j2se/`) 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ソフトウェア

チュートリアルの作成と実行には、次のソフトウェアが必要です。その一部は Sun ONE Studio 4, Community Edition のデフォルトインストールに含まれています。ソフトウェアのバージョンについては、次の Web サイトで提供しているリリースノートをご覧ください。

`http://forte.sun.co.jp/forte/ffj/documentation/index.html`

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

    Sun ONE Studio 4, Community Edition をインストール時に、PointBase をインストールすることができます。PointBase がインストールされているかどうかは、Sun ONE Studio 4 がインストールされているディレクトリに `pointbase`

ディレクトリが存在するかどうかで確認できます。pointbase ディレクトリが存在しない場合は、インストーラを再度実行してインストールすることができます。

- 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

- チュートリアルデータベース表を作成する SQL スクリプト

  チュートリアルの SQL スクリプトは、付録 B で提供しています。これらのスクリプトは Sun ONE Studio 4 Developer Resources ポータルサイトから入手することもできる CDShopCart アプリケーションファイルに含まれています。チュートリアルデータベース表のインストールについては、9 ページの「チュートリアルデータベース表の作成」を参照してください。

- Web サーバー

  チュートリアルアプリケーションは Web サーバーを必要とする Web アプリケーションです。このチュートリアルでは、IDE にバンドルされている Tomcat (バージョン 4.0) 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。

- 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。Netscape Communicator™ または Microsoft Internet Explorer が使用できます。Netscape Communicator は、Sun ONE Studio Developer Resources ポータルサイトからダウンロードできます (製品 CD にも含まれています) 。

Sun ONE Studio 4 IDE の起動

次の節で説明している手順で Sun ONE Studio 4 IDE を起動します。詳細は『Sun ONE Studio 4, Community Edition インストールガイド』を参照してください。

Microsoft Windows における起動

インストールが完了したら、以下のいずれかの方法で IDE を起動してください。

- デスクトップ上の「Sun ONE Studio 4 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。この実行可能ファイルは、*forte4j-home*¥`bin` ディレクトリにあります。このディレクトリには実行可能な `runide.exe` もあります。この `runide.exe` アイコンを使用した場合、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」->「Sun Microsystems」->「Sun ONE Studio 4 CE」->「Sun ONE Studio」を選択する。

- コマンド行から実行可能ファイルを実行する。

```
C:¥> runide.exe [switch]
```

スイッチについては、5 ページの「コマンド行スイッチによるセッションの変更」を参照してください。

Solaris、Linux などの UNIX 環境における起動

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。このスクリプトは端末ウィンドウで以下のコマンドを入力して起動することができます。

```
$ sh runide.sh
```

このスクリプトのカスタマイズ方法については、5 ページの「コマンド行スイッチによるセッションの変更」を参照してください。

コマンド行スイッチによるセッションの変更

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『Sun ONE Studio 4, 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

  コマンド行から IDE を実行する際にオプションを設定できます。

- Solaris、UNIX、Linux 環境の場合

  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 `ide.sh` ファイルを変更するか、オプションを指定して `ide.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` コマンド行スイッチ

スイッチ	意味
`-classic`	クラシック JVM を使用する
`-cp:p` *addl-classpath*	指定されたクラスパスを Sun ONE Studio 4 のクラスパスの先頭に追加する
`-cp:a` *addl-classpath*	指定されたクラスパスを Sun ONE StudioSun ONE Studio 4 の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、指定されたサイズに設定する
`-locale` *language* [:*country*[:*variant*]]	このセッションで、デフォルトのロケールではなく指定されたロケールを使用する
`-J`*jvm-flags*	指定されたフラグを 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`	GUI ダイアログを開いて、コマンド行オプション一覧表示する
`-hotspot`、`-client`、`-server`、`-classic`、`-native`、`-green`	-hotspot、-client、-server、-classic 、-native、-green から指定したタイプの JVM を使用する

表 1-1　runide コマンド行スイッチ (続き)

スイッチ	意味
`-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

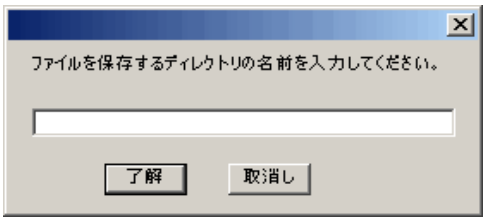
Sun ONE Studio 4 IDE は、各ユーザー専用のディレクトリにユーザーのプロジェクトやサンプル、IDE 設定を保存します。これにより、個々の開発者がそれぞれの開発作業を同期させながら、個人の作業と設定を維持できます。

- Solaris、Linux などの UNIX 環境の場合

  `-userdir` コマンド行スイッチ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かった場合、デフォルトでは、ユーザー設定は *user-home*`/ffjuser40ce` ディレクトリに格納されます。

- Microsoft Windows システムの場合

  Sun ONE Studio 4 IDE を初めて起動すると、ユーザー設定ディレクトリを指定するよう求められます。その場合は、`C:¥MySettings` のようにフルパスでディレクトリを指定します。



  この値は、後で使用できるようレジストリに記録されます。IDE の起動時に `-userdir` コマンド行スイッチを使用して、特定のセッションでは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。

Sun ONE Studio 4 のディレクトリ構成

Sun ONE Studio 4 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　Sun ONE Studio 4 のディレクトリ構成

ディレクトリ	用途
`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`	Sun ONE Studio 4 の起動ツール (Microsoft Windows の場合は `ide.cfg` ファイルも) が格納される。 (Solaris のみ) xemacs、gvim、xdesigner、forte_fcc など Solaris ツールと一緒に配布されるスタンドアロンアプリケーションの起動スクリプトも格納される
`docs`	Sun ONE Studio 4 のヘルプファイルと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)。
`emacs`	(Solaris のみ) emacs ファイルが格納される
`examples`	Sun ONE Studio 4, Enterprise Edition for Java の主な機能を例証するためのソースファイルが格納される
`j2sdkee1.3.1`	Java 2 Platform, Enterprise Edition (J2EE™ SDK) v 1.3.1 が格納される。(Sun ONE Studio 4 のインストール時に指定した場合)
`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
`man`	(Solaris のみ) Solaris Developer Module のマニュアルページが格納される（インストールした場合）
`modules`	Sun ONE Studio 4 モジュールの JAR ファイルが格納される
`pointbase`	PointBase Server 4.2 Restricted Edition データベースの実行可能ファイルとクラス、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)

表 1-2　Sun ONE Studio 4 のディレクトリ構成 (続き)

ディレクトリ	用途
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と一緒に再配布できる
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。(技術サポートを依頼するときに役立つ ide.log など)
tomcat401	Tomcat 4.01 Web サーバーのソースが格納される

Sun ONE Studio 4 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、Sun ONE Studio 4 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成

ディレクトリ	用途
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	Update Center 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」タブにデフォルトでマウントされるディレクトリ。このディレクトリには、IDE 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した場合は、そのマウントされたディレクトリに保存される
sampledir/examples	複数の NetBeans サンプ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。
tomcat401_base	JSP ページの操作のユーザー設定が格納される

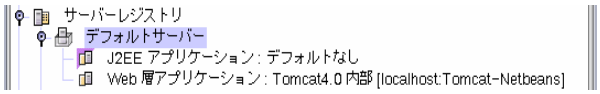
デフォルト Web サーバーの確認

CDShopCart チュートリアルでは、Tomcat 4.0 Web サーバーを使用します。 Sun ONE Studio 4 インストーラは Tomcat をデフォルトの Web サーバーとしてインストールしますが、CDShopCart アプリケーションをテストまたは実行する前に、Tomcat がデフォルトのサーバーになっていることを確認してください。

以下の手順に従って、Tomcat Web サーバーがデフォルトのサーバーになっていることを確認します。

1. IDE で、エクスプローラの「実行時」タブをクリックします。
2. 「サーバーレジストリ」ノードを展開し、さらにそのサブノードの「デフォルトサーバー」を展開します。

   - IDE が正しいサーバーを使用している場合、「デフォルトサーバー」ノードは以下のように表示されます。



   - Tomcat 4.0 以外のものが表示された場合は、次の操作を行ってください。

   a. デフォルト Web サーバーノードを右クリックして、「デフォルトサーバーを設定」を選択します。

      「デフォルト Web サーバーを選択」ダイアログが表示されます。

   b. 適切なサーバーを選択して「了解」をクリックします。

チュートリアルデータベース表の作成

チュートリアルに入る前に、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。この節では、PointBase データベースに表をインストールする手順を説明します。これとは別のデータベースプログラムを使う場合には、付録 B のチュートリアル用 SQL スクリプトを参照してください。

これらの SQL スクリプトファイルは、チュートリアルアプリケーションのソースコードをまとめた zip ファイルにも入っています。この zip ファイルは、Sun ONE Studio Developer Resources ポータルサイトからダウンロードできます。

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

表 1-4 に示すように、4 種類の SQL スクリプトファイルが用意されています。

表 1-4　チュートリアルの表を作成するための SQL スクリプト

スクリプト名	説明
`CDCatalog_pb.sql`	チュートリアルで使用する表を PointBase の SQL 形式で作成し、データを格納する。(スクリプトの内容は、90 ページの「PointBase データベース用のスクリプト」参照)
`CDCatalog_ora.sql`	チュートリアルで使用する表を Oracle の SQL 形式で作成し、データを格納する。(スクリプトの内容は、91 ページの「Oracle データベース用のスクリプト」参照)
`CDCatalog_ms.sql`	チュートリアルで使用する表を Microsoft SQLServer の SQL 形式で作成し、データを格納する。 (スクリプトの内容は、92 ページの「Microsoft SQLServer データベース用のスクリプト」参照)
`CDCatalog_db2.sql`	チュートリアルで使用する表を IBM DB2 の SQL 形式で作成し、データを格納する。(スクリプトの内容は、93 ページの「IBM DB2 データベース用のスクリプト」参照)

`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`		

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

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)。

注 - すでに Sun ONE Studio 4 IDE を起動している場合は、データベース表の作成中も起動したままでも、終了してもかまいません。

チュートリアルデータベース表を作成するには、次の操作を行ってください。

1. PointBase サーバーを起動します。

   - Solaris または Linux 環境の場合
     forte4j-home/pointbase/server ディレクトリの Server ファイルを実行します。

   - Microsoft Windows の場合
     「スタート」->「プログラム」->「Sun Microsystems」->「Sun ONE Studio 4 CE」->「PointBase」->「Server」を選択するか、
     forte4j-home/pointbase/server ディレクトリの server.bat ファイルをダブルクリックします。

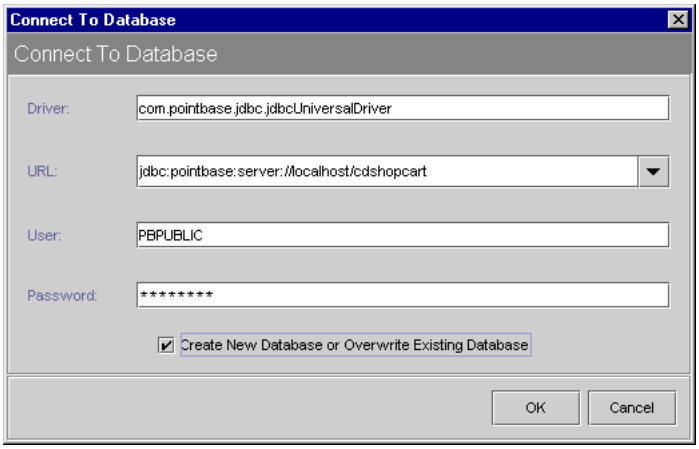
2. PointBase コンソールを起動します。

   - Solaris または Linux 環境の場合
     forte4j-home/pointbase/client ディレクトリの Console ファイルを実行します。

- Microsoft Windows の場合
  「スタート」->「プログラム」->「Sun Microsystems」->「Sun ONE Studio 4 CE」->「PointBase」->「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 ドライバとデフォルトのサンプルデータベースの値が示されます。

3. 以下に示すように、「URL」フィールド末尾の sample という単語を cdshopcart に変更します。



4. 「Create New Database」チェックボックスを選択して、「OK」をクリックします。

   PointBase コンソールが表示されます。ステータスメッセージとして「Ready」が表示されたら、次の手順に進みます。

5. 90 ページの「PointBase データベース用のスクリプト」の PointBase スクリプトをコンソールの SQL エントリウィンドウに入力します。

   チュートリアルソースの zip ファイルにある CDCatalog_pb.sql ファイルを使用する場合は、次の操作を行ってください。

   a. 「File」->「Open」を選択して、ファイル選択のダイアログを表示します。

   b. CDCatalog_pb.sql ファイルを選択して「Open」をクリックします。

6. 「SQL」->「Execute All」を選択します。

   スクリプトを実行したことを示すメッセージが表示されます。「Cannot find the table...」で始まる初期メッセージは無視してください。このメッセージが表示されるのは、表に対する DROP 文があるためで、表はまだ作成されていないからです。この DROP 文は、後で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。

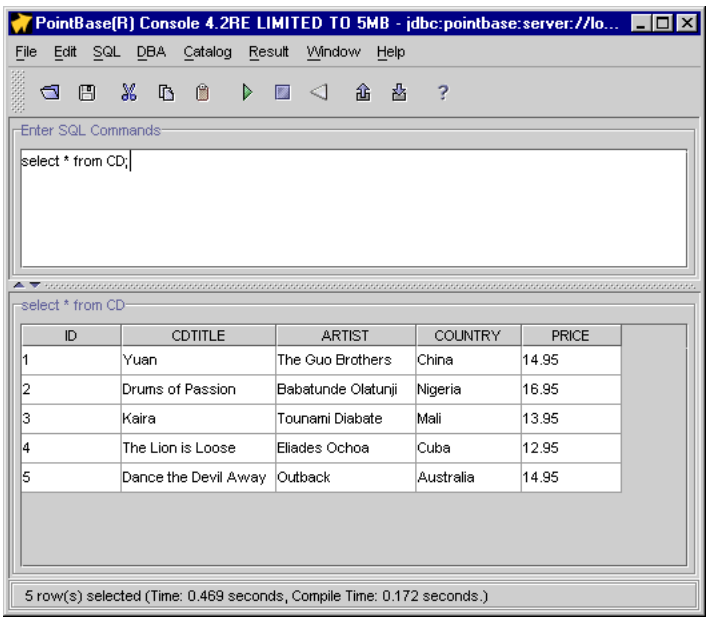
   forte4j-home/pointbase/databases ディレクトリにはデータベースファイルが 2 つ作成されます。

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

```
SELECT * FROM CD;
```

8. 「SQL」->「Execute」を選択します。

   コンソールに、次のような CD 表が表示されます。



注 - CD 表の表示が上の図と異なる場合、「Window」->「Windows」を選択して表示形式を変更します。

---

9. 「PointBase Console」ウィンドウを閉じます。

これで、チュートリアルに取り組む準備ができました。

第2章

CDShopCart の紹介

実際にアプリケーションを作成することで、Sun ONE Studio 4, Community Edition の機能を使用した Web アプリケーションコンポーネントを作成する方法を学習します。

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。最初にこのアプリケーションに求められる条件を明確にし、続いて、それらの条件を満たすアーキテクチャを取り上げます。最後の節では、Web モジュール構造や Sun ONE Studio 4 のタグライブラリなどの Sun ONE Studio 4, 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を説明します。

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働き

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。このアプリケーションの利用者は、Web ブラウザを使用してアプリケーションのインタフェースと次のようなやりとりをします。

1. 利用者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. 利用者が CD の購入を確定すると、アプリケーションがその注文に対して「Thank You」というメッセージを表示し、セッションを終了します。

4. この時点で、利用者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利用者のやりとりは、利用者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、利用者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利用者とのやりとりを示しています。このシナリオを見ていくと、このアプリケーションに求められる条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. 利用者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します。

   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 の一覧と関連情報 (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. 利用者は、CD に関連付けられている「Add」ボタンを押して、購入する CD を選択します。

   この操作によって、Shopping Cart ページが表示されます。このページには、選択された CD のタイトル、ID 番号、価格が表示されます。

3. 利用者は、さらに購入する CD を追加できます。

   利用者が Shopping Cart ページにある「Resume Shopping」ボタンを押すと、アプリケーションが CD カタログページを再表示して、利用者が CD を追加選択できるようにします。利用者はこの手順を何度でも繰り返し、同じ CD を複数回追加することもできます。この場合は同じ CD について、複数の行がカートに追加されます。各 CD の枚数 (Amount) を表す列は表示されません。

4. 利用者は、ショッピングカートページ上の各 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. 利用者はページ上の「Resume Shopping」ボタンを押して CD Catalog List ページに戻ることも、「Cancel Order」ボタンを押してセッションを終了することもできます (Cancel Order ページについては 手順 7 を参照)。

6. 利用者は、Shopping Cart ページ上の「Place Order」ボタンを押して購入を決定します。

   この操作によって「Thank You」というメッセージが表示され、セッションが終了します。このとき利用者は、このメッセージページ上の「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 を指定してこのアプリケーションを終了することもできます。

7. 利用者は、Shopping Cart ページの「Cancel Order」ボタンを押すと、いつでも注文を取り消すことができます。

   「Cancel Order」ボタンを押すと、「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

上記のようなシナリオで CDShopCart アプリケーションが使用されると仮定した場合、そのやりとり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としては、以下が挙げられます。

- ページ間を移動するための一群のリンク
- サイトで販売する CD のマスタービュー (リスト表示)
- 購入のために選択された 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
- チェックアウトのための、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
- ホームページに戻って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

利用者から見たチュートリアルアプリケーション

ここでは、15 ページの「チュートリアルアプリケーションの働き」で説明したシナリオと機能仕様が、利用者から見た場合にどのように実現されるかを説明します。

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。再び CD Catalog List ページが表示されます。

4. 同じまたは異なる CD の「Add」をクリックします。

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

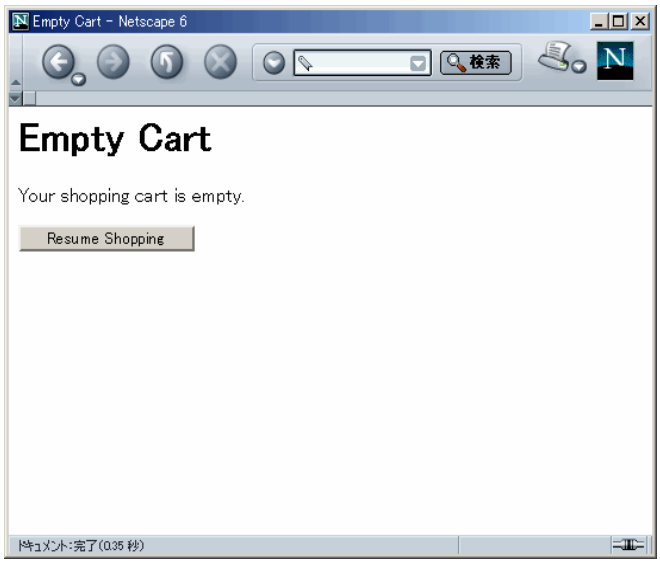
5. 購入する CD をすべて選択するまで、手順 2 と手順 3 を繰り返します。

   Shopping Cart ページに選択した CD がすべて表示されます。同じ CD を何度か選択した場合は、それぞれ別々の行に表示されます。

6. CD を取り除くには、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。取り除かれた CD は表示されません。

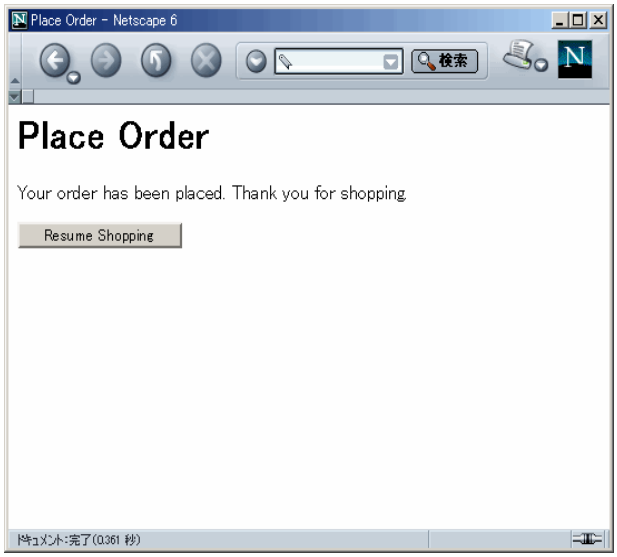
   表内のすべての CD を取り除くと、Empty Cart JSP ページが表示されます。



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

8. 注文を確定するには、Shopping Cart ページの「Place Order」ボタンをクリックします。

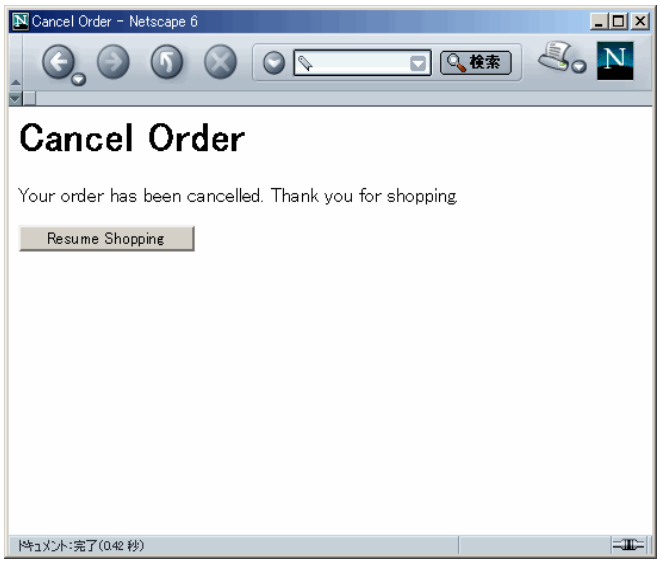
   Place Order ページが表示されます。このページは、PlaceOrder JSP ページによって作成されます。



   この時点で利用者は、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 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、その Web アプリケーションから結果を受信する Web 主体のアプリケーションです。Web アプリケーションは、Web コンポーネントとそれら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 ページなどの、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 モジュールで構成されています。Web モジュールとは、J2EE アプリケーション内にある Web 資源の最小単位であると同時に、配備および使用可能な最小単位です。『Java Servlet Specification Version 2.3』で導入され、Sun ONE Studio 4, Community Edition に実装された機能の一つに 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、必要なデフォルトのバージョンのデータオブジェクト、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についての詳細は、『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、Web モジュール構造に固有の情報については、オンラインヘルプにある 「JSP/ サーブレットモジュールの概要」の「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構成要素とその相互関係を示します。

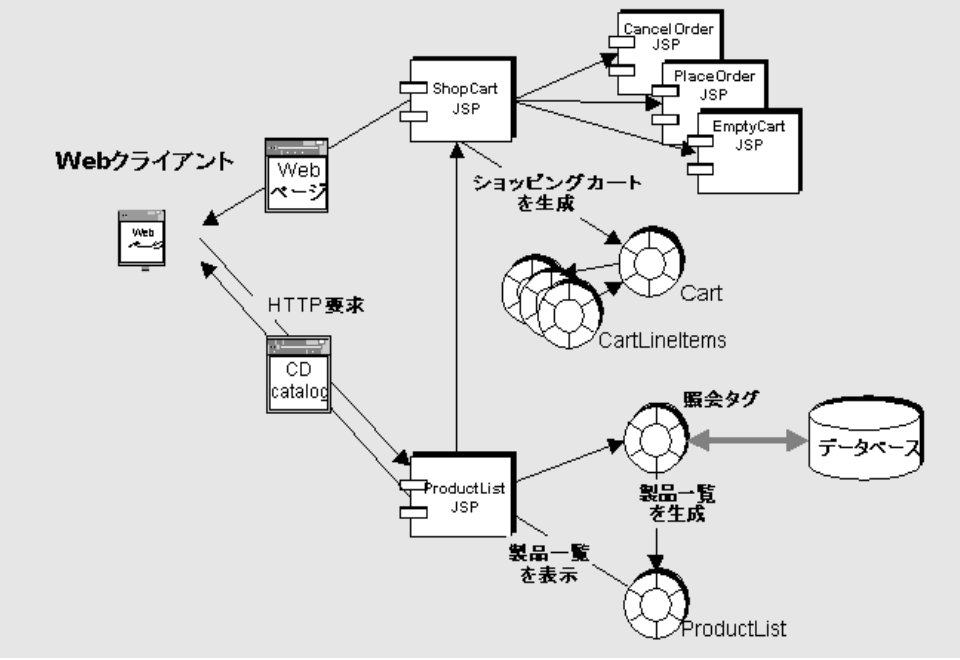


図 2-1　CDShopCart アプリケーションの構造

アプリケーションの構成要素

以下に、図 2-1 に示したアプリケーションの各要素を簡単に説明します。

- クライアントコンポーネント

  クライアントコンポーネントは、アプリケーションページを表示する Web ブラウザです。

- サービスコンポーネント (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 ページの表にそのデータを表示します。また、このページには「Add」ボタンがあり、利用者はこのボタンを使って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 - 購入のために選択した CD を Shopping Cart Web ページの表に表示します。 このページには、「Delete」「Place Order」「Cancel Order」「Resume Shopping」ボタンがあります。
  - `EmptyCart` JSP ページ - ショッピングカートから CD を削除したときにメッセージを表示します。このページには「Resume Shopping」ボタンがあります。
  - `CancelOrder` JSP ページ - 注文が取り消されたことを示すメッセージを表示します。このページには、ProductList JSP ページに戻るための「Resume Shopping」ボタンがあります。
  - `PlaceOrder` JSP ページ - 注文が行われたことを示すメッセージを表示します。このページには、ProductList JSP ページに戻るための「Resume Shopping」ボタンがあります。
  - `Cart` Bean - 購入のために選択された CD を表します。
  - `CartLineItem` Bean - カート内の明細 (CD) を表します。

サービスコンポーネントの詳細

CDShopCart アプリケーションのサービスコンポーネントは Web モジュールであり、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する、 4 つの JSP ページで構成されます。この 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

  現在の利用者のセッションを検索し、セッションが存在しない場合はそれを作成する JSP ページです。このページでは、Apache Foundation の JSTL (JSP Standard Tag Library) の SQL タグを使用してデータベースの CD リストにアクセスしたり、コアタグを使用して表に CD を表示したりします。また、このページには、表示する CD 品目ごとに「Add」ボタンを表示します。

- `ShopCart` JSP ページ

ProductList ページの「Add」ボタンがクリックされると、該当する CD の明細データがこの JSP ページに渡されます。ShopCart ページは、CartLineItem オブジェクトで構成される Cart オブジェクトをインスタンス化し、JSTL タグライブラリのコアタグを使用して、表にそれらオブジェクトを表示します。このページには、カート内の CD の明細ごとに「Delete」ボタンを表示します。このボタンがクリックされると、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表を再表示します。カートにあった CD がすべて取り除かれると、アプリケーションは EmptyCart ページに進みます。ShopCart ページには、「Resume Shopping」「Cancel Order」「Place Order」ボタンがあります。これらのボタンがクリックされると、それぞれ ProductList ページ、CancelOrder ページ、PlaceOrder ページに進みます。

- Cart JavaBeans コンポーネント

  lineItem 属性を持ち、CartLineItem オブジェクトを取得および削除するためのメソッドが含まれる Bean です。この Bean は ShopCart ページによってインポートされます。

- CartLineItem JavaBeans コンポーネント

  CD 関連の属性を持ち、Cart 品目 (CD) の属性 (ID、タイトル、アーティスト、国籍、価格) を取得および設定するためのメソッドが含まれる Bean です。

- CancelOrder JSP ページ

  ShopCart ページで「Cancel Order」ボタンがクリックされたときに呼び出される JSP ページです。このページは、注文が取り消されたことを示すメッセージを表示してセッションを無効にします。また、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

  ShopCart ページでカートの最後の CD が削除されたときに呼び出される JSP ページです。このページは、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

  カートに CD が含まれている ShopCart ページで「Place Order」ボタンがクリックされたときに呼び出される JSP ページです。このページは、注文が確定されたことを示すメッセージを表示してセッションを無効にします。「Resume Shopping」ボタンを表示します。

チュートリアルアプリケーションの作成に必要な作業の概要

チュートリアルの中には、基本的なアプリケーションを実際に作成してみる章があります。チュートリアルアプリケーションを作成するには、第 1 章で説明している手順に従って Sun ONE Studio 4 ソフトウェアをインストールし、必要な設定を行っておくことと、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もあります。

この節では、次の第 3 章で学ぶ Sun ONE Studio 4, Community Edition の機能の使用方法の概要をまとめておきます。

- Web モジュール (作成、開発、テスト実行)
- データベースとの接続とやりとりに使用する埋め込み JSTL タグ
- 取り込んだデータを反復処理したり表示したりする埋め込み JSTL タグ

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

Web モジュールの作成

Sun ONE Studio 4 IDE 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。この構造そのものが Web モジュールです。一つの 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 モジュールの開発の詳細については説明していません。27 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ます。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』とオンラインヘルプを参照してください。

JSTL タグライブラリの使用方法

CDShopCart Web モジュールでは、まず 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、購入予定として選択された CD を表示する ShopCart JSP ページを作成します。こうしたデータアクセスおよびデータ表現機能に JSTL タグを利用します。

35 ページの「JSP タグによるデータベースデータのフェッチと表示」では、SQL タグを使用してデータベースに JDBC 接続し、CD のデータを取り込む方法を説明しています。また、コア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ています。この反復処理によって、ProductList ページは、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 のデータを表示します。60 ページの「Shopping Cart 表の品目を追加または削除するコードの追加」では、コア JSTL 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特定する方法を紹介しています。この処理によって、カート表の適切な列にフィールド値を表示することができます。

サポート構成要素の作成

ShopCart JSP ページのサポート構成要素には、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。

49 ページの「CartLineItem Bean の作成」では、CD の「Add」ボタンがクリックされたときに、ProductList から ShopCart に渡される明細 (CD) のパラメータを保持するオブジェクトを持つ Bean を作成する方法を紹介します。さらに、55 ページの「Cart Bean の作成」では、選択された明細 (CD) を蓄積するオブジェクトを持つ Bean の作成方法を学びます。Cart Bean には、カートの品目を追加・削除するためのメソッドも含まれています。

66 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、注文に対するお礼の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。また 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

チュートリアルでは、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を実行する度に、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。

最後に

このチュートリアルアプリケーションは、比較的短時間 (1 日程度) で作成できるように簡潔に設計されています。このため、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、以下の Sun ONE Studio 4 Developer Resources ポータルサイトから入手できます。
`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

第3章

CDShopCart アプリケーションの作成

この章では、CDShopCart アプリケーションの作成方法を手順に従って説明します。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 Sun ONE Studio 4 ソフトウェアをインストールし、必要な設定をしておくことと、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、次の節で構成されています。

各コンポーネントは作成するたびにテストをします。この章の後半では、第 2 章で説明しているとおりに基本アプリケーションを実行できるようになります。

参照 - この章で説明している JSP ページや JavaBeans コンポーネントのソースコードは、すべて付録 A に掲載されています。

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、一つの 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、Sun ONE Studio 4, Community Edition IDE を使用し、Web モジュールにショッピングカート機能を実装する方法を説明します。

Sun ONE Studio 4 の Web モジュールとは

『Java Servlet Specification, version 2.3』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ドキュメントルートと呼ばれ、このルートに、Web アプリケーションに含まれるすべてのファイルが格納されます。この階層には、`WEB-INF` ディレクトリという非公開の特殊なサブディレクトリもあります。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ます。具体的には、Web 配備記述子 (`web.xml` ファイル) やサーブレット、クラスを読み込む際に Web アプリケーションローダが使用するユーティリティクラスなどが含まれます。

しかし、アプリケーションのファイルを 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 コンテナに配備するには、それらのファイルが最終的に Web モジュール構造の構成要素になっている必要があります。Sun ONE Studio 4 IDE の Web モジュール構造機能は、必要な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 モジュール開発についての詳細は、『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、Web モジュールについての詳細は、Sun ONE Studio 4 のオンラインヘルプを参照してください。

`CDShopCart` Web モジュールの作成

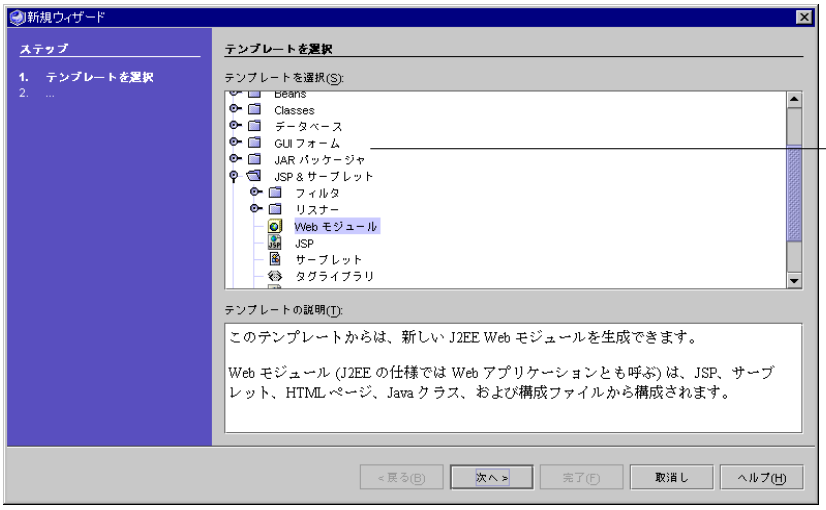
この節では、CDShopCart アプリケーションの Web モジュールを作成します。Sun ONE Studio 4 の Web モジュール機能を使用すると、何もないところから Web モジュールディレクトリを作成できます (また、既存のディレクトリを変換して Web モジュールを作成することもできる)。

`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」タブから「ファイル」->「新規」を選択し、新規ウィザードを表示します。

   このウィザードが提供するテンプレートを利用して、さまざまなオブジェクトを作成することができます。

2. 「JSP & サーブレット」ノードを展開し、「Web モジュール」を選択します。



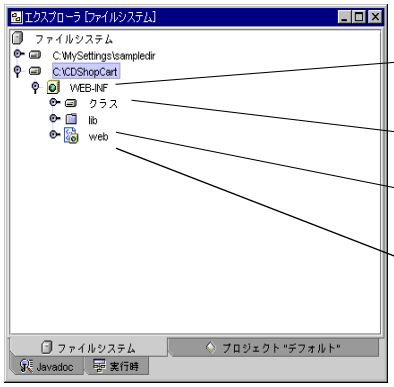
3. 「次へ」をクリックして、Web モジュールのディレクトリ名を指定するための区画を表示します。

4. 「ディレクトリ」フィールドにデフォルトで表示されるディレクトリ指定を消去して、`c:\CDShopCart` と入力します。

5. 「完了」をクリックします。

   エクスプローラに新しい Web モジュール、`CDShopCart` が作成されます。「デフォルトプロジェクト」ウィンドウに Web モジュールのもう一つのビューがインストールされたことを示すダイアログが表示されます。「了解」をクリックして、ダイアログを閉じます。

6. Web モジュール内のノードを展開し、自動的に作成された項目を確認します。



「WEB-INF」 - J2EE の仕様に基づいて作成され、モジュール内のすべての項目を参照する

「クラス」 - すべてのサーブレット、Bean、クラスが含まれる

「lib」 - タグライブラリの lib ディレクトリとインポートされた JAR ファイルが含まれる

「web」 - web.xml 配備記述子

これで、アプリケーションの最初のコンポーネントである、ProductList JSP ページの作成を開始できます。

JSP タグによるデータベースデータのフェッチと表示

この節では、CD 製品データをフェッチして表示する ProductList という JSP ページを作成します。データベースに接続して表データを取り込み、表示するデータの書式設定をするには、JSTL (JSP Standard Tag Library) タグを利用します。JSTL は、Apache Foundation のプロジェクトの 1 つである Jakarta プロジェクトで生まれたタグライブラリです。Sun ONE Studio 4 には JSTL が埋め込まれています。

JSTL に関する説明や例、チュートリアルについては、Jakarta プロジェクトの Web サイト (http://jakarta.apache.org/taglibs/index.html) を参照してください。

JSP タグとは

JSP ファイルの本文には、固定テンプレートデータと要素という 2 種類のコードを含むことができます。

- 固定テンプレートデータ - JSP コンテナが認識しないコードで、変更が加えられずにそのまま HTTP 応答に渡されます。

固定テンプレートデータとしては、たとえば XML コードや HTML コードがあります。CDShopCart アプリケーションでは、HTML コードを使用して見出しやタイトル、表、ボタンを作成します。

- 要素タイプ - 以下の 3 種類の要素があります。
  - 命令 - 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ときに使用される要素です。
  - スクリプト要素 - JSP ファイルに Java コードを埋め込むことができます。
  - JSP タグ - Java コードを記述することなく Java オブジェクトを併用できる方法として、 XML 形式のタグを使用します。

    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。

標準タグとカスタムタグ

JSP 仕様書に定義されている標準タグは、あらゆる JSP コンテナで使用できます。カスタムタグは、「タグライブラリ」と呼ばれる XML ドキュメントで定義されます。カスタムタグライブラリは、命令要素で宣言すると JSP ページから利用できるようになります。

JSTL タグ

JSTL (JSP Standard Tag Library) は、Jakarta プロジェクトによって作成されたカスタムタグライブラリです。Sun ONE Studio 4 IDE には、JSTL タグとそれらのタグをサポートするクラス、インタフェースを含む次の JAR ファイルが埋め込まれています

- `standard.jar` - コアや SQL タグを含む複数の JSTL タグライブラリのファイル
- `jstl.jar` - CDShopCart が使用する `standard.jar` の JSTL ライブラリをサポートするインタフェースとクラスのファイル

これらの JAR ファイルで使用するタグライブラリ記述子 (TLD) ファイルとして、`sql.tld` (データベースアクション用) と `core.tld` (他のすべてのアクション用) の 2 つがあります。

JSTL タグの使用方法

JSTL タグを使用するには、タグライブラリを宣言し、定められているタグ構文に従ってタグを使用する必要があります。

JSTL 構文規則についての詳細は、Jakarta プロジェクトの Web サイト (`http://jakarta.apache.org/taglibs/index.html`) にあるJSTL の資料を参照してください。

`taglib` 命令の使用方法

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

`taglib` 命令では、特定の URI (Uniform Resource Identifier) 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。URI および JSTL タグの接頭辞は、Apache JSTL 規則で定義されています。

taglib 命令の一般的な構文は以下のとおりです。

```
<%@ taglib prefix="prefix" uri="http://java.sun.com/jstl/taglibname
```

たとえば、コアタグライブラリは以下のように宣言します。

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core
```

タグ構文の使用方法

JSTL タグは XML 構文に基づいており、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、本体 (任意)、対応する終了タグ
- 空のタグと指定可能な属性

CDShopCart では、開始タグとして `query` タグを使用します。このタグは、データソースからデータをフェッチします。`query` タグの構文は以下のとおりです。

```
<sql:query var="stored_query" dataSource="${dataSource}" >
    body
</sql:query>
```

空のタグとしては、sql ライブラリの setDataSource タグを使用します。このタグは、データベースに対する JDBC 接続を作成します。setDataSource タグの構文は以下のとおりです。

```
<sql:setDataSource var="dataSource"
    url="driver_url"
    driver="driver_string"
    user="user_id" password="pwd" />
```

JSTL タグ規則では、タグに関する情報をエクスポートするタグ属性として " var" を使用することになっています。この "var" が選ばれた理由は、"id" 規則を使用するスクリプト変数ではなく、JSP 変数をタグ属性としていることを強調するためです。

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仕組み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. 「Web モジュールへの JSTL タグライブラリの追加」 - 39 ページ
2. 「ProductList JSP ページの作成」 - 40 ページ
3. 「タグライブラリの宣言」 - 41 ページ

4. 「setDataSource タグを使用したデータベースへの接続」 - 42 ページ
5. 「query タグを使用した CD データのフェッチ」 - 43 ページ
6. 「反復子タグを使用したデータの表示」 - 43 ページ
7. 「各 CD 行の「Add」ボタンの作成」 - 45 ページ

Web モジュールへの JSTL タグライブラリの追加

この節では、JSTL タグライブラリの JAR ファイルである `standard.jar` と、サポートクラスおよびインタフェースの JAR ファイルである `jstl.jar` を `CDShopCart` Web モジュールにインポートします。これらのファイルは、アクションを実装するために必要です。

Web モジュールに JSTL タグライブラリを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` Web 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。

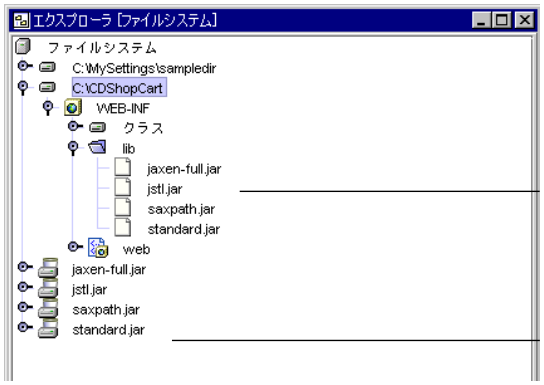
   JSP タグライブラリ・リポジトリブラウザが表示されます。

2. `standard` ファイルを選択して「了解」をクリックします。

   エクスプローラで、「`WEB-INF`」ノードにある「`lib`」ノードを開きます。

3. 「lib」ノードにあるファイルを確認します。

   lib ノードの下に表示される jstl.jar および standard.jar ファイルは、このチュートリアルで使用するタグを格納しています。この他に 2 つの jar ファイルが表示されます。これらのファイルは XML タグに必要な API を格納していますが、このチュートリアルでは使用しません。これら 4 つのファイルは個別にマウントされ、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、Sun ONE Studio 4 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。正しく追加されていると、リストに 2 つの JAR ファイルがあります。

ProductList JSP ページの作成

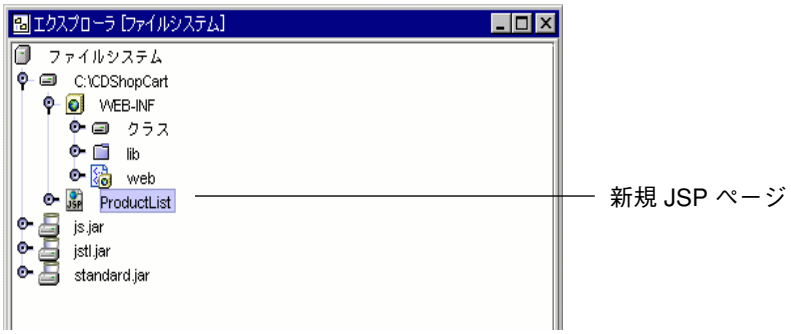
これで、タグを使用してデータベースから CD データを取り込み、表に表示するページを作成することができます。このページのタイトルは CD Catalog List で、ProductList JSP ページによって作成されます。

1. エクスプローラで CDShopCart Web モジュールを右クリックし、コンテキストメニューから「新規」->「JSP & サーブレット」->「JSP」を選択します。

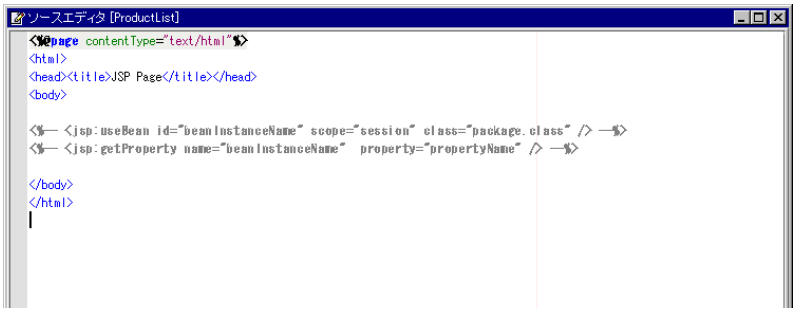
   新規ウィザードによって「新規オブジェクト名」区画が表示されます。

2. 「名前」フィールドに ProductList と入力して、「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 JSP ページのスケルトンが表示されます。



タグライブラリの宣言

最初に、 37 ページの「taglib 命令の使用方法」の説明に従って、タグライブラリを宣言する必要があります。この命令は、ページタイトル直後の JSP ページ本文の前に挿入してください。以下の手順では、ページのタイトルを変更し、2 つのタグライブラリ用の命令を追加します。

1. ProductList ページの本文で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. コアおよび SQL アクションのタグライブラリをインポートする命令を追加します。以下のように入力します。

```
<%@taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
```

setDataSource タグを使用したデータベースへの接続

最初に使用するタグは sql ライブラリの setDataSource タグです。このタグは、データベースに対する JDBC 接続を作成します。ページの見出し (<H1> タグを使用) を追加してから、<body> タグの下に、データベースに接続するための setDataSource タグを挿入します。

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. ヘッダーの下に JDBC 接続を作成します。

   - PointBase ドライバを使用している場合

```
<sql:setDataSource var="productDS"
    url="jdbc:pointbase:server//localhost/cdshopcart"
    driver="com.pointbase.jdbc.jdbcUniversalDriver"
    user="PBPUBLIC" password="PBPUBLIC" />
```

   - Oracle データベース (thin ドライバ) を使用している場合

```
<sql:setDataSource var="productDS"
    url="jdbc:oracle:thin:@hostname:port#:SID"
    driver="oracle.jdbc.driver.OracleDriver"
    user="userid" password="password" />
```

デフォルトの Oracle ポート番号は 1521 です。

■ Microsoft SQLServer データベース (Weblogic ドライバ) を使用している場合

```
<sql:setDataSource var="productDS"
    url="jdbc:weblogic:mssqlserver4:database@hostname:port#"
    driver="weblogic.jdbc.mssqlserver4.Driver"
    user="userid" password="password" />
```

SQLServer のデフォルトのポート番号は 1433 です。

参照 - 入力したコードの書式を自動的に整えるには、ソースエディタ上にカーソルを移動し、Ctrl+Shift F を押します。

---

query タグを使用した CD データのフェッチ

ここでは、sql タグの query を使用して、データベースを照会しデータ行を含む結果セットを 1 つ取得して、その結果セットを productQuery 結果セットに書き込みます。そして、この JSP 変数を反復子タグに渡して、結果を表示します。query タグは、標準の SQL 文の SELECT をサポートしています。

driverタグのすぐ後に query タグを挿入してください。

データベースにすべての CD データの照会を行うには、次のようにします。

- ProductList ページの driver タグの直後に、データベースからすべての CD データを選択する照会を作成します。

```
// 上で作成した productDS dataSource を使用 :
<sql:query var="productQuery" dataSource="${productDS}">
    SELECT * FROM CD
</sql:query>
```

反復子タグを使用したデータの表示

この時点で表を作成し、表のセルにデータを格納します。まず HTML タグで表を作成し、forEach タグを使用してフェッチしたデータを反復処理するコードを作成します。このとき各行のフィールドのデータのフェッチには、out タグを使用します。

forEach の構文は以下のとおりです。

```
<c:forEach var="$Current_collection_item" items="${Collection}" >
```

items 変数は反復処理対象の現在のコレクションを、また、var 変数はそのコレクションの現在の項目をそれぞれ保持します。コレクションが結果セットの場合、現在の項目は現在の行位置にある結果セットオブジェクトです。たとえば myResultSet という名前の結果セットにある行を反復処理するには、次のコードを使用します。

```
<c:forEach var="row" items="${myResultSet.rows}" >
    <TR>
        <TD><c:out value="${row.col1}"/></TD>>
        <TD><c:out value="${row.col2}" /></TD>
        <TD><c:out value="${row.col3}" /></TD>
    </TR>
</c:forEach>
```

表を作成して、そのセルにデータを取り込むには、次のようにします。

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. forEach タグと out タグを使用して、表にデータを格納するコードを作成します。

```
// 次の表に、結果の現在行にある
// 各フィールドの値を取り込む
<c:forEach var="row" items="productQuery.rows" >
    <TR>
        <TD><c:out value="${row.ID}"/></TD>
        <TD><c:out value="${row.CDTITLE}"/></TD>
        <TD><c:out value="${row.ARTIST}"/></TD>
        <TD><c:out value="${row.COUNTRY}"/></TD>
        <TD><c:out value="${row.PRICE}"/></TD>
// 次節で、さらにここにデータを追加する
```

各 CD 行の「Add」ボタンの作成

CD 表では、行ごとに 1 つの CD のデータが保持されます。CD を購入するには、その CD 行の「Add」ボタンをクリックします。HTML フォームを作成して利用者からの入力 (ボタンのクリック) を受け付ける領域を定義し、この領域内に Shopping Cart JSP ページに渡す情報を埋め込みます。このコードは前のコードの直後に記述します。

「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。

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

2. 埋め込み情報 (製品 ID、タイトル、価格) を指定します。

```
    <input type=hidden name=cdId value="<c:out value=
"${row.ID}"/>">
    <input type=hidden name=cdTitle value="<c:out value=
"${row.CDTITLE}"/>">
    <input type=hidden name=cdPrice value="<c:out value=
"${row.PRICE}"/>">
```

3. 直前のコードのすぐ下に「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. 反復処理と表を終了します。

```
</c:forEach>
</TABLE>
```

6. 「ファイル」->「保存」を選択して、作業内容を保存します。

参照 - 74 ページの「ProductList.jsp のソース」に、ProductList JSP ページの全ソースコードを掲載しています。

ProductList JSP ページのテスト

ここでは、これまでの作業内容に問題がなかったかどうかを確認します。ProductList JSP ページをコンパイルした後、統合された Sun ONE Studio 4 実行時システムとブラウザを使用して、ページを実行してみましょう。

ProductList ページをテストするには、次のようにします。

1. PointBase ネットワークサーバーをまだ起動していない場合は、「ツール」->「PointBase ネットワークサーバー」->「サーバーを起動」を選択して起動します。

   PointBase サーバーのウィンドウが表示されます。

2. Tomcat がデフォルト Web サーバーになっていることを確認します。

   エクスプローラの「実行時」タブを選択し、「サーバーレジストリ」->「デフォルトサーバー」の順に展開します。Tomcat 4.0 が Web 層アプリケーションに指定されていない場合、「インストールされているサーバー」を展開し、「Tomcat 4.0」ノードの下にあるすべてのサブノードを展開します。「localhost:8081」を右クリックして、「デフォルトとして設定」を選択します。

3. エクスプローラの「ファイルシステム」タブで 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ステータスメッセージが表示されます。問題がなければ、最後に「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。問題を解決して、「完了」というメッセージが表示されるまで、手順を繰り返してください。

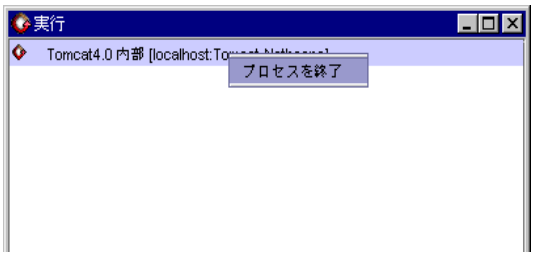
4. ProductList JSP ページを選択し、ツールバーの「実行」ボタン () をクリックしてページを実行します。

   「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」を選択して、実行することもできます。

   組み込み Tomcat サーバーが起動されて、「実行」作業スペースに切り替わり、「実行」ウィンドウが開きます。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 に示すような CD Catalog List ページが表示されます。

5. プロセスの実行を終了するには、「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。

   Tomcat Web サーバープロセスの実行が終了します。



6. 「編集」タブをクリックして「編集」作業スペースに戻ります。

   以上で、データベースとの接続、データの取り込みおよび表示に JSTL タグを使った JSP ページが完了しました。Shopping Cart ページの作成に進んでください。

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された CD を表示する仕組みを作成します。この仕組みは、以下の要素で構成されます。

- `CartLineItem` Bean - `ProductList` ページからパラメータとして渡された CD 行 (利用者が選択した CD の行) の属性を保持する
- `CartLineItemBeanInfo` - Bean Info コンポーネント。オーバーロードされた設定メソッドがあるかどうかに関係なく、`id` および `price` が `CartLineItem` のプロパティであることを示す
- `Cart` Bean - `CartLineItem` オブジェクトを保持する
- `ShopCart` JSP ページ - `Cart` オブジェクトを受け取り、表に 1 行で表示する
- 「Delete」ボタン - `Shopping Cart` ページ上に表示される CD ごとに用意される
- 「Cancel Order」「Resume Shopping」「Place Order」ボタン - それぞれに対応する機能が実装される

複数の CD が選択されている場合、`Shopping Cart` ページは図 3-2 のようになります。

図 3-2　Shopping Cart ページ

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

1. 「CartLineItem Bean の作成」 - 49 ページ
2. 「id プロパティおよび price プロパティを文字列へ変換」 - 51 ページ
3. 「CartLineItemBeanInfo コンポーネントの作成」 - 54 ページ
4. 「Cart Bean の作成」 - 55 ページ
5. 「Shopping Cart ページの作成」 - 59 ページ
6. 「Shopping Cart ページのテスト」 - 64 ページ

CartLineItem Bean の作成

ここでは、CD Catalog List (ProductList) ページから Shopping Cart ページに渡すパラメータを保持するオブジェクトを持つカート品目 Bean を作成します。このためには、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。

注 - J2SE 1.4 では、すべてのクラスがパッケージに含まれている必要があり、このチュートリアルは J2SE 1.4 に準拠しています。WEB-INF ディレクトリの **クラス** サブディレクトリはパッケージではないため、クラス (Bean) を保持するためには、**クラス**サブディレクトリにパッケージを作成する必要があります。このチュートリアルでは、このパッケージの名前は ShopCart です。この変更は、J2SE 1.3.1 を使用している場合でも同様に機能します。

---

1. CDShopCart Web モジュールの「WEB-INF」ノードを展開し、「クラス」ノードを右クリックして「新規」->「Java パッケージ」を選択します。
2. パッケージに ShopCart という名前を付けます。
3. ShopCart パッケージを右クリックして、「新規」->「Beans」->「Java Bean」を選択します。

   新規ウィザードによって「新規オブジェクト名」区画が表示されます。
4. 「名前」フィールドに CartLineItem と入力して「完了」をクリックします。

   エクスプローラに新しい CartLineItem Bean が、ソースエディタにそのコードがそれぞれ表示されます。Bean に付いている赤い印 ( ) は、その Bean を「コンパイルする必要がある」ことを示します。コンパイルは後で行うので、ここではこの印は無視してください。
5. Bean とそのクラスを展開して、内容を表示します。
6. 「Bean パターン」ノードを右クリックして、「追加」->「プロパティ」を選択します。
7. 「新規プロパティパターン」ダイアログで cdtitle プロパティを定義します。

   a. 「名前」フィールドに cdtitle と入力します。

   b. 「種類」は String を選択します。

   c. 次のオプションを設定します。

   - フィールドを生成

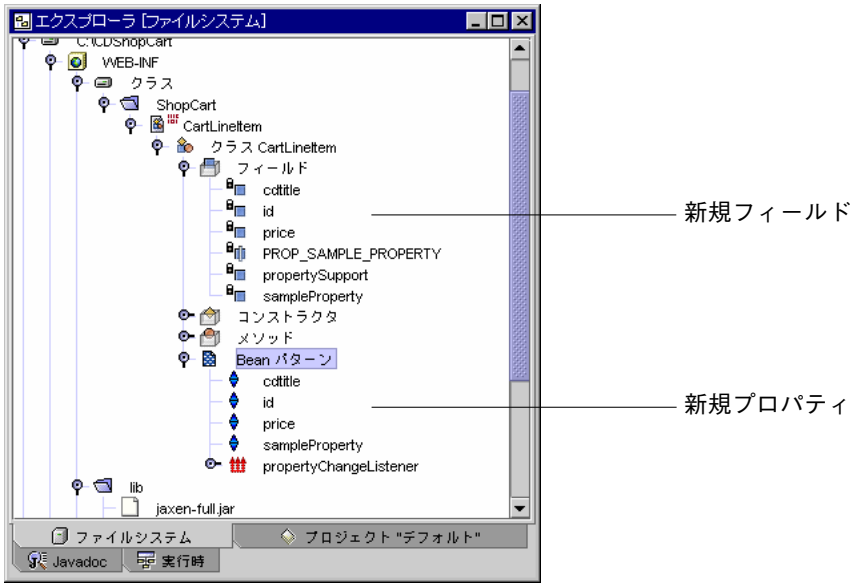
- return 文を生成
- set 文を生成

定義を終えたダイアログは次のようになります。

8. 「了解」をクリックして、ダイアログを閉じます。
9. 手順 6 から 手順 8 を繰り返すことによって、同様に `id` プロパティを設定します。
   a. 「名前」フィールドに `id` と入力します。
   b. 「種類」に `int` を選択します。
   c. 次のオプションを設定します。
      - フィールドを生成
      - return 文を生成
      - set 文を生成
10. 再度手順 6 から 手順 8 を繰り返すことによって、同様に `price` プロパティを設定します。
    a. 「名前」フィールドに `price` と入力します。
    b. 「種類」に `double` を選択します。
    c. 次のオプションを設定します。
       - フィールドを生成

- return 文を生成
- set 文を生成

11. `CartLineItem` Bean クラスの「フィールド」ノードを展開して、作成した新規フィールドを表示します。



12. 「メソッド」ノードを展開し、各フィールドに新しく作成した `get` および `set` メソッドを表示します。

13. 新規フィールドの 1 つ (たとえば `cdtitle`) をダブルクリックして、ソースエディタにコードを表示します。

`id` プロパティおよび `price` プロパティを文字列へ変換

`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` プロパティは文字列ではないため、これらのプロパティ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オーバーロードして、適切なコードを追加します。

`setId` メソッドおよび `setPrice` メソッドをオーバーロードするには、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、「メソッドを追加」を選択します。

2. 「新規メソッドを追加」ダイアログで setId メソッドを定義します。

   a. 「名前」フィールドに `setId` と入力します。

   b. 「戻り値の型」に `void` を選択します。

3. 「メソッドのパラメータ」ボックスで「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。

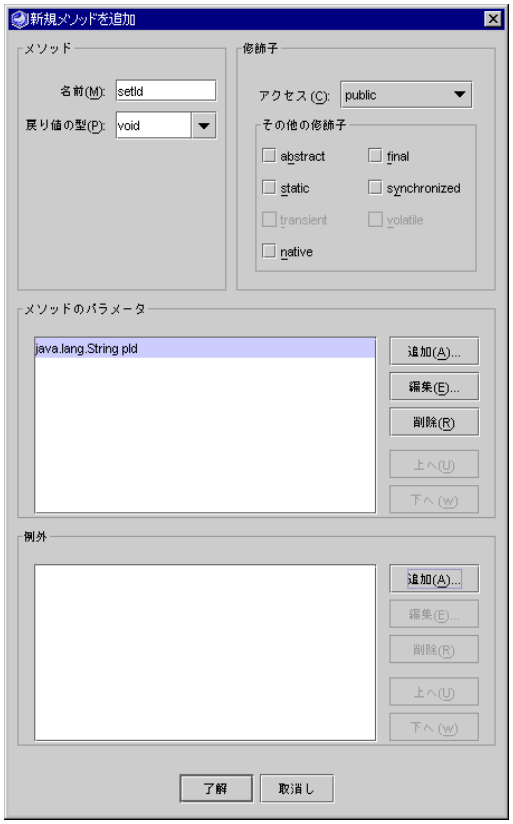
4. `pId` パラメータを定義します。

   a. 「型」に `java.lang.String` を選択します。

   b. 「名前」に `pId` を入力します。

5. 「了解」をクリックします。

   定義を終えた「新規メソッドを追加」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、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);
}
```

参照 - コピー & ペーストでソースエディタにコードを入力する場合、ソースエディタ内にカーソルを移動し、Ctrl+Shift F を押すことによって、自動的に書式を整えることができます。

`setPrice` メソッドにも同様の作業を行います。

8. 手順 1 から 手順 6 を繰り返すことによって `setPrice` メソッドを定義します。

   メソッドの定義

   a. 「名前」フィールドに `setPrice` と入力します。

   b. 「戻り値の型」に `void` を選択します。

   パラメータの定義

   a. 「型」に `java.lang.String` を選択します。

   b. 「名前」に `pPrice` を入力します。

9. この新しいメソッドに次のコードを追加します。

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

10. 「`CartLineItem` Bean ( ) (クラスではない) 」を選択し、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、「コンパイルする必要がある」ことを示す赤い印が Bean のノードから消えて、`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。

参照 - 76 ページの「CartLineItem Bean のソース」に、この Bean の全ソースコードを掲載しています。

CartLineItemBeanInfo コンポーネントの作成

Bean プロパティが、2 つで 1 組の公開アクセスメソッド (get と set) によってアクセス可能な非公開属性であることは、Introspector クラスに組み込まれている標準の Java 規則です。しかし、前節で id および price プロパティに set メソッドをオーバーロードしたため、もはや Introspector はそれらのプロパティを Bean プロパティと認識しません。

JSTL 式言語では、Java の規則に従っている場合にのみ Bean プロパティを使用可能にするため、それらのフィールドをプロパティとして認識されるようにする必要があります。たとえば次のコードの場合、"." 演算子の後の項目はプロパティである必要があります。

```
<TD><c:out value="${row.id}"/></TD>
<TD><c:out value="${row.cdtitle}"/></TD>
```

この問題を回避するには、BeanInfo コンポーネントを使って Bean のプロパティを直接指定します。CartLineItem Bean に対して BeanInfo コンポーネントを作成し、id と price がプロパティであることを指示するコードを記述します。

以下の手順で、id と price が CartLineItem Bean のプロパティであることを指示します。

1. 「ShopCart パッケージ」を右クリックして、「新規」->「Beans」->「アイコンなし BeanInfo」を選択します。
2. この BeanInfo に CartLineItemBeanInfo という名前を付け、「完了」をクリックします。

   エクスプローラに新しい CartLineItemBeanInfo ノードが作成されます。
3. BeanInfo のノードとそのメソッドノードを展開します。
4. getPdescriptor メソッドをダブルクリックします。

   CartLineItemBeanInfo Bean のコードの中の getPdescriptor メソッドが定義されている箇所がソースエディタに表示されます。

5. getPdescriptor メソッドの本文に次のコードを入力します。

```
if (properties == null) {
    try {
        PropertyDescriptor[] props = {
            new PropertyDescriptor("cdtitle", CartLineItem.class),
            new PropertyDescriptor("id", CartLineItem.class),
            new PropertyDescriptor("price", CartLineItem.class)};
            properties = props;
    } catch (IntrospectionException ex) {
        return null;
    }
}
```

参照 - コピー & ペーストでソースエディタにコードを入力する場合、ソースエディタ内にカーソルを移動し、Ctrl+Shift F を押すことによって、自動的に書式を整えることができます。

---

6. CartLineItemBeanInfo ノードを右クリックして、「コンパイル」を選択します。

BeanInfo Bean がコンパイルされます。

Cart Bean の作成

ProductList JSP ページで「Add」ボタンが押されると、Shopping Cart JSP ページは、その「Add」ボタンに対応する Cart オブジェクトを特定し、対応する Cart オブジェクトがない場合は、インスタンス化します。この Cart オブジェクトには、ProductList JSP ページから渡される CD 品目オブジェクトが保持されます。Cart オブジェクトは、Cart Bean に基づいて作成されます。

Cart Bean を作成するには、

1. ShopCart パッケージを右クリックして、「新規」->「Beans」->「JavaBean」を選択します。
2. この Bean に Cart という名前を付けて、「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、「追加」->「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で新しい `lineItems` Bean パターンを作成します。

   a. 「名前」フィールドに `lineItems` と入力します。

   b. 「種類」に `java.util.Vector` と入力します。

   c. 次のオプションを設定します。

      - フィールドを生成
      - return 文を生成
      - set 文を生成

   d. 「了解」をクリックして Bean パターンを作成し、ダイアログを閉じます。

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て、新しく作成されたコードをソースエディタで表示します。

   フィールドのアクセス権を、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」ノードを展開し、「`lineItems`」フィールドを選択します。

7. `lineItems` の「プロパティ」ウィンドウにある「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る) 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」を選択してください。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」->「編集」->「プロパティウィンドウ」を選択します。

---

8. 省略符号ボタン (...) をクリックして修飾子ダイアログを表示します。

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。

    続いて、カート品目 (CD) オブジェクトをインスタンス化するコード、選択された品目数を返すメソッド、そしてカートから品目を削除するもう 1 つのメソッドを追加します。

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

    ソースエディタに `Cart()` コンストラクタが表示されます。

12. 新しい `lineItems` (CD) オブジェクトをインスタンス化する次のコード (太字部分) を `Cart` Bean のコンストラクタに追加します。

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

13. 「Cart メソッド」ノードを右クリックし、「メソッドを追加」を選択して、`findLineItem` メソッドを定義します。

    メソッドの定義

    a. 「名前」フィールドに `findLineItem` と入力します。

    b. 「戻り値の型」に `int` を選択します。

    パラメータの定義

    a. 「型」に `int` を選択します。

b. 「名前」に pId を入力します。

14. ソースエディタを使用し、findLineItem メソッドに次のコード (太字部分) を追加します。

```
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 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. 手順 13 を繰り返し、 removeLineItem メソッドを定義します。

メソッドの定義

a. 「名前」フィールドに removeLineItem と入力します。

b. 「戻り値の型」に void を選択します。

パラメータの定義

a. 「型」に int を選択します。

b. 「名前」に pId を入力します。

16. ソースエディタ内で次のコード (太字部分) 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. 「`Cart` Bean ( ) (クラスではない) 」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 がエラーなくコンパイルされたら、`ShopCart` JSP ページの作成に進みます。

参照 - 82 ページの「Cart Bean のソース」に、この Bean の全ソースコードを掲載しています。

Shopping Cart ページの作成

ここでは、CD Catalog List ページからパラメータを受け取り、その一部 (ID、タイトル、価格) を表に 1 行で表示するページを作成します。このページには、表から品目を削除したり、Catalog List ページに戻ったり、注文を確定したりするための仕組みも提供します。このページのタイトルは「Shopping Cart」で、`ShopCart` JSP ページによって作成されます。

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」を選択して JSP ページを作成します。
2. この JSP ページに `ShopCart` という名前を付け、「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に `ShopCart` JSP が表示されます。

   このページを作成するには、次の作業を行います。

   1. 「Shopping Cart 表の品目を追加または削除するコードの追加」 - 60 ページ
   2. 「反復子タグを使用したデータの表示」 - 43 ページ
   3. 「ページへのボタンの追加」 - 63 ページ

Shopping Cart 表の品目を追加または削除するコードの追加

この節では、ショッピングカートに入れられた品目 (CD) の表を作成する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するには、Cart Bean と java.util ライブラリ (このライブラリの CartLineItem は Vector タイプ)、CartLineItem をインポートする命令を使用します。ProductList JSP ページで使用したものと同じコアタグを使用するため、これらのタグをインポートする命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作成された CD を追加するコードを追加してください。「Delete」ボタンが押されたときに品目を削除するコード、さらには、表が空の場合に別の JSP ページ (後に EmptyCart JSP ページという名前で作成) に移動するためのコードも追加します。

ProductList JSP ページ同様、ここでは反復子タグを使用して、表のデータを整理します。そして、フォームを使用して表を作成し、表の各行に「Delete」ボタンを追加します。

1. java.util ライブラリと ShopCart パッケージをインポートする Page 命令を追加します。

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

2. ページのタイトルを Shopping Cart に変更し、core.jar ライブラリをインポートする命令を追加します。

```
<head><title>Shopping Cart</title></head>
<%@taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
```

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 タグを挿入し、Cart Bean を使用するよう JSP ページに指示します。

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

このコードは、Cart オブジェクトをインスタンス化し、そのインスタンスをセッションに追加します。

続いて、セッションで追加操作が行われたときの処理、すなわち、ProductList ページで「Add」ボタンがクリックされたときの処理を指定します。このためには、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するコードを追加します。

5. 現在の操作を取得して、「Add」ボタンがクリックされたときの処理を定義するコードを作成します。

```
<%
    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);
    }
```

続いて、セッションで削除操作が行われたときの処理、すなわち、ShoppingCart ページで「Delete」ボタンがクリックされたときの処理を指定します。

6. 次のコードを使用して、「Delete」ボタンがクリックされ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」のクリックで Cart CD 表の最後の行が削除されたときの処理を指定します。このためには、JSP の forward タグを使用して、EmptyCart JSP ページに移動するコードを作成します。 (EmptyCart ページ は、この後すぐに作成します) 以下が、手順 5 で作成し始めたスクリプトの最後のコードです。このコードでは、forward タグのためにスクリプトレットの実行をいったん中断しますが、最終的にスクリプトレットを終了するには、スクリプトレットに戻って再実行する必要があります。

7. 次のコードを使用します。

```
    // カートから最後の品目が削除された場合
    if (((Vector)session.getAttribute("myLineItems")).size()
        == 0) {
        // スクリプトレットを一時的に終了して、JSP の forward タグを
        // 使用して EmptyCart ページに移動できるようにする
        %>
        <jsp:forward page="EmptyCart.jsp" />
        // スクリプトレットの実行を再開する
        <%
    }
%>
```

反復子タグを使用したカート表へのデータの取り込み

ここでは、forEach タグと out タグを使用して、渡されたデータを反復処理し、各行のフィールドデータを取り込むコードを作成します。これは、43 ページの「反復子タグを使用したデータの表示」で説明した手順によく似ています。

1. 購入予定のデータを取り込む表の作成の最初のステップとして、表の見出しを作成します。

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

2. forEach タグと out タグを使用して、表にデータを取り込むためのコードを作成します。

```
<c:forEach var="item" items="$myLineItems">
    <TR>
        <TD><c:out value="${item.id}"/></TD>
        <TD><c:out value="${item.cdtitle}"/></TD>
        <TD><c:out value="${item.price}"/></TD>
```

上記コード内の out タグは、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. 63 ページの「ページへのボタンの追加」の手順に従って、各行に「Delete」ボタンを作成します。

```
<TD>
<form method=get action="ShopCart.jsp">
    <input type=hidden name=cdId value="<c:out value=
"${item.id}"/>">
    <input type=hidden name=cdTitle value="<c:out value=
"${item.cdtitle}"/>">
    <input type=hidden name=cdPrice value="<c:out value=
"${item.price}"/>">
    <input type=submit name=operation value="Delete">
</form>
</TD>
    </TR>
</c:forEach>
</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。

1. 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
<form method=get action="ProductList.jsp">
    <input type=submit value="Resume Shopping">
</form>
// 「Place Order」ボタンを作成
<form method=get action="PlaceOrder.jsp">
    <input type=submit value="Place Order">
</form>
// 「Cancel Order」ボタンを作成
<form method=get action="CancelOrder.jsp">
    <input type=submit value="Cancel Order">
</form>
</p>
// ページの終了
</body>
</html>
```

参照 - 85 ページの「ShopCart.jsp のソース」に、このページの全ソースコードを掲載しています。

Shopping Cart ページのテスト

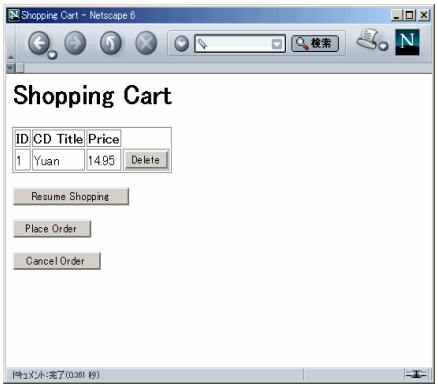
ShopCart ページは直接テストすることができません。ProductList ページをテスト実行し、「Add」ボタンを使用して Shopping Cart ページへ移動します。

1. PointBase ネットワークサーバーをまだ起動していない場合は、「ツール」->「PointBase ネットワークサーバー」->「サーバーを起動」を選択することによって起動します。
2. 「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。
3. ProductList JSP ページを右クリックし、「実行 (強制再読み込み)」を選択します。

   数秒で CD Catalog List ページが表示されます。

4. いずれかの「Add」ボタンをクリックして、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。



5. 「Resume Shopping」ボタンを使用して、CD Catalog List ページに戻ります。
6. 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」を選択して実行を終了します。
7. メインウィンドウの「編集」タブをクリックして、編集作業スペースに戻ります。

   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

ここでは、カートが空になった場合に表示する JSP ページと、「Place Order」ボタンまたは「Cancel Order」ボタンがクリックされたときに表示する、それぞれのボタンに対応する JSP ページを作成します。

この節では、次の作業を行います。

- 「Empty Cart ページの作成」 - 66 ページ
- 「Place Order ページの作成」 - 67 ページ
- 「Cancel Order ページの作成」 - 69 ページ

Empty Cart ページの作成

空のベクトルを検出した場合、反復子 (iterator) タグは空の表を作成するのではなく例外をスローします。この例外の対処方法としてチュートリアルでは、例外が発生したかどうかを検査し (60 ページの「Shopping Cart 表の品目を追加または削除するコードの追加」の手順 7)、空の場合は Empty Cart ページを表示する処理を行います。この Empty Cart ページには、Product List ページに戻るための「Resume Shopping」ボタンが含まれます。

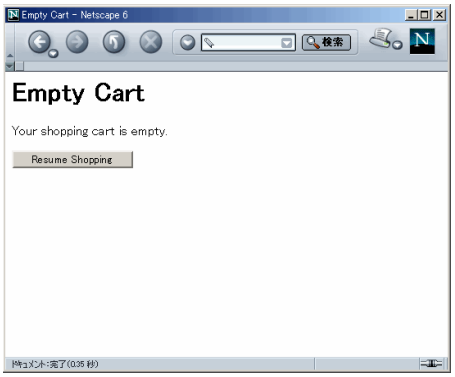


図 3-3　Empty Cart ページ

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 と入力して、「完了」をクリックします。

   エクスプローラに EmptyCart JSP ページが、ソースエディタにそのソースコードがそれぞれ表示されます。

3. ページのタイトルを Empty Cart に変更して、次のような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>
    // メッセージを表示
    Your shopping cart is empty.
    <P>
    // 「Resume Shopping」ボタンを追加
    <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

Place Order ページと Cancel Order ページは非常に単純なページです。注文の確定や取り消しというアクションを実装する方法は多数あり、ここで紹介するページは 1 つの例にすぎません。これらのページのプログラミングでは、これまでに見てきた Sun ONE Studio 4 機能のほんの一部を使用するにすぎないので、このチュートリアルでは、あえてもっとも簡単な実装方法を紹介することにします。

Place Order ページは、Shopping Cart ページで「Place Order」ボタンがクリックされたときに表示されます。このページが表示されると、セッションが終了します。

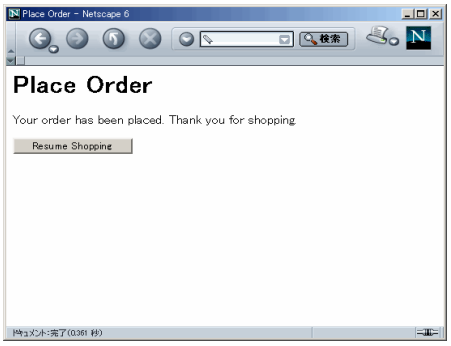


図 3-4　Place Order ページ

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、「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>
    <H1> Place Order </H1>
    // セッションを無効にする
    <%
    session.invalidate();
    %>
    Your order has been placed. 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 PlaceOrder ページを保存します。

Cancel Order ページの作成

Cancel Order ページは、Shopping Cart ページで「Cancel Order」ボタンがクリックされたときに表示されます。このページが表示されると、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。

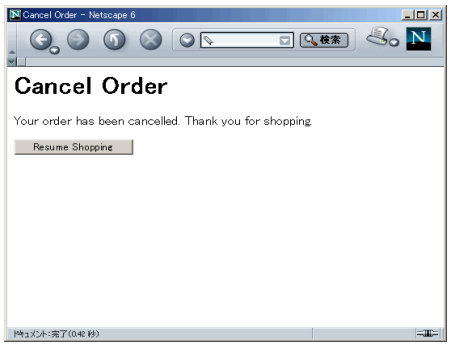


図 3-5　Cancel Order ページ

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 ページに似た次のような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

Shopping Cart ページ同様、メッセージページは、`ProductList` JSP を実行することによってテストします。Shopping Cart に CD 品目を追加して、適切なアクションを実行することによって、それぞれのメッセージページを表示してみます。

1. PointBase ネットワークサーバーをまだ起動していない場合は、「ツール」->「PointBase ネットワークサーバー」->「サーバーを起動」を選択して起動します。
2. 「`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。

   JSP ページがすべてコンパイルされます。
3. 「`ProductList` JSP ページ」を右クリックし、「実行 (強制再読み込み)」を選択します。

   「`WEB-INF`」ノードを右クリックして、「実行 (強制再読み込み)」を選択する方法もあります。

   数秒で CD Catalog List ページが表示されます。
4. いずれかの「Add」ボタンをクリックして、Shopping Cart ページに移動します。
5. Empty Cart ページをテストするには、カートに入れた品目の「Delete」ボタンをクリックします。

   Empty Cart ページが表示されます。
6. 「Resume Shopping」ボタンをクリックして、CD Catalog List ページに戻ります。
7. カートに CD を追加してみます。
8. 「Cancel Order」ボタンをクリックすることによって、Cancel Order ページをテストします。
9. 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。
10. 別の CD をカートに追加します。

11. Shopping Cart ページが表示されたら、追加した カートに含まれている CD が追加した CD だけであることを確認します。

    「Cancel Order」操作によって直前のセッションを終了したため (手順 8)、このカートには CD が 1 枚しかないはずです。

12. 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。

13. 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。

14. 別の CD をカートに追加してみます。

    Place Order ページでセッションは終了しているため、カートには CD が 1 枚しかないはずです。

15. 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。

付録 A

CDShopCart のソースファイル

この付録では、以下の CDShopCart コンポーネントのコードをまとめています。

これらのソースファイルは、次の Sun ONE Studio 4 Developer Resources ポータルサイトからダウンロードできる CDShopCart アプリケーションの zip ファイルにも含まれています。

```
http://forte.sun.com/ffj/documentation/tutorialsandexamples.html
```

参照 – これらのファイルの内容を Sun ONE Studio 4 のソースエディタにカット & ペーストすると、すべての書式設定が失われます。ソースエディタでコードの書式を整えるには、ソースエディタ内にカーソルを移動し、Ctrl + Shift F を押してください。

ProductList.jsp のソース

```
<%@page contentType="text/html"%>
<html>
<head><title>CD Catalog List</title></head>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
<body>
<h1> CD Catalog List </h1>

<sql:setDataSource var="productDS"
    url="jdbc:pointbase:server://localhost/cdshopcart"
    driver="com.pointbase.jdbc.jdbcUniversalDriver"
    user="PBPUBLIC" />

<%--<sql:setDataSource var="productDS"
    url="jdbc:oracle:thin:@<hostname>:<port#>:<SID>"
    driver="oracle.jdbc.driver.OracleDriver"
    user="userid" /> --%>

<%--<sql:setDataSource var="productDS"
    url="jdbc:weblogic:mssqlserver4:<database>@<hostname>:<port#>"
    driver="weblogic.jdbc.mssqlserver4.Driver"
    user="userid" /> --%>

<sql:query var="productQuery"  dataSource="${productDS}"  >
    SELECT * FROM CD
</sql:query>

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

```
<c:forEach var="row" items="${productQuery.rows}">
    <TR>
        <TD><c:out value="${row.ID}"/></TD>
        <TD><c:out value="${row.CDTITLE}"/></TD>
        <TD><c:out value="${row.ARTIST}"/></TD>
        <TD><c:out value="${row.COUNTRY}"/></TD>
        <TD><c:out value="${row.PRICE}"/></TD>
        <TD>
        <form method=get action="ShopCart.jsp">
            <input type=hidden name=cdId value="<c:out value="${row.ID}"/>">
            <input type=hidden name=cdTitle value="<c:out
value="${row.CDTITLE}"/>">
            <input type=hidden name=cdPrice value="<c:out
value="${row.PRICE}"/>">
            <input type=submit name=operation value=Add>
        </form>
        </TD>
    </TR>
</c:forEach>
</TABLE>
</body>
</html>
```

CartLineItem Bean のソース

```
package ShopCart;

import java.beans.*;

public class CartLineItem extends Object implements java.io.Serializable {

    private static final String PROP_SAMPLE_PROPERTY = "SampleProperty";

    private String sampleProperty;

    private PropertyChangeSupport propertySupport;

    /** cdtitle プロパティの値を保持 */
    private String cdtitle;

    /** id プロパティの値を保持 */
    private int id;

    /** price プロパティの値を保持 */
    private double price;

    /** CartLineItem を新規作成 */
    public CartLineItem() {
        propertySupport = new PropertyChangeSupport ( this );
    }

    public String getSampleProperty() {
        return sampleProperty;
    }

    public void setSampleProperty(String value) {
        String oldValue = sampleProperty;
        sampleProperty = valie;
        propertySupport.firePropertyChange(PROP_SAMPLE_PROPERTY, oldValue,
sampleProperty);
    }
```

```
    public void addPropertyChangeListener (PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener (listener);
    }
    public void removePropertyChangeListener (PropertyChangeListener listener){
        propertySupport.removePropertyChangeListener (listener);
    }

    /** cdtitle プロパティ用の取得メソッド
    * @return: cdtitle プロパティの値
    */
    public String getCdtitle() {
        return this.cdtitle;
    }

    /** cdtitle プロパティの設定メソッド
    * @param cdtitle: cdtitle プロパティの新しい値
    */
    public void setCdtitle(String cdtitle) {
        this.cdtitle = cdtitle;
    }

    /** id プロパティの取得メソッド
    * @return: id プロパティの値
    */
    public int getId() {
        return this.id;
    }

    /** id プロパティの設定メソッド
    * @param id: id プロパティの新しい値
    */
    public void setId(int id) {
        this.id = id;
    }

    /** price プロパティの取得メソッド
    * @return: price プロパティの値
    */
    public double getPrice() {
        return this.price;
    }

    /** price プロパティの設定メソッド
    * @param price: price プロパティの新しい値
    */
```

```
    public void setPrice(double price) {
        this.price = price;
    }

    public void setId(java.lang.String pId) {
        int val = Integer.parseInt(pId);    // ここではエラーチェックなし
        this.setId(val);
    }

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

CartLineItemBeanInfo のソース

```
package ShopCart;

import java.beans.*;

public class CartLineItemBeanInfo extends SimpleBeanInfo {

    // Bean 記述子情報はイントロスペクションから取得
    private static BeanDescriptor beanDescriptor = null;
    private static BeanDescriptor getBdescriptor(){

        // ここに BeanDescriptor をカスタマイズするためのコードを追加できる

        return beanDescriptor;     }

    // プロパティ情報はイントロスペクションから取得
    private static PropertyDescriptor[] properties = null;
    private static PropertyDescriptor[] getPdescriptor(){
        if (properties == null) {
            try {
                PropertyDescriptor[] props = {
                    new PropertyDescriptor("cdtitle", CartLineItem.class),
                    new PropertyDescriptor("id", CartLineItem.class),
                    new PropertyDescriptor("price", CartLineItem.class)};
                    properties = props;
            } catch (IntrospectionException ex) {
                return null;
            }
        }
        return properties;     }

    // イベントセット情報はイントロスペクションから取得
    private static EventSetDescriptor[] eventSets = null;
    private static EventSetDescriptor[] getEdescriptor(){

        // ここにイベントセット配列をカスタマイズするためのコードを追加できる

        return eventSets;     }
```

```
    // メソッド情報はイントロスペクションから取得
    private static MethodDescriptor[] methods = null;
    private static MethodDescriptor[] getMdescriptor(){

        // ここにメソッド配列をカスタマイズするためのコードを追加できる

        return methods;     }

    private static int defaultPropertyIndex = -1;
    private static int defaultEventIndex = -1;


// ここに Superclass BeanInfo をカスタマイズするためのコードを追加できる

    /**
    * bean の <code>BeanDescriptor</code> を取得
    *
    * @return: このBean の編集可能なプロパティを表す
    * BeanDescriptor。自動解析で情報を
    * を取得する場合は null を返せる
    */
    public BeanDescriptor getBeanDescriptor() {
        return getBdescriptor();
    }

    /**
    * Bean の <code>PropertyDescriptor</code> を取得
    *
    * @return: この Bean がサポートする編集可能なプロパティを表す
    * PropertyDescriptors の配列。自動解析で情報を
    * を取得する場合は null を返せる
    * <p>
    * プロパティが添え字付きの場合、結果配列内のそのエントリは
    * PropertyDescriptor の IndexedPropertyDescriptor サブクラスに属する。
    * getPropertyDescriptors のクライアントは "instanceof" を使用して、指定された
    * PropertyDescriptor が IndexedPropertyDescriptor であるかどうかを調べる
    */
    public PropertyDescriptor[] getPropertyDescriptors() {
        return getPdescriptor();
    }

    /**
    * Bean の <code>EventSetDescriptor</code> を取得
    *
    * @return: この Bean が発生させるイベントの種類を表す
```

```
    * EventSetDescriptors の配列。自動解析で情報を
    * を取得する場合は null を返せる
    */
    public EventSetDescriptor[] getEventSetDescriptors() {
        return getEdescriptor();
    }

    /**
    * Bean の <code>MethodDescriptor</code> を取得
    *
    * @return: この bean が実装するメソッドを表す
    * MethodDescriptors の配列。自動解析で情報を
    * 取得する場合は null を返せる
    */
    public MethodDescriptor[] getMethodDescriptors() {
        return getMdescriptor();
    }

    /**
    * Bean は、そのカスタマイズを行う開発者が更新対象として
    * よく初期選択すると考えられるプロパティを、「デフォルト」のプロパティとして
    * を持つことができる
    * @return: getPropertyDescriptors によって返された
    *   PropertyDescriptor 配列内のデフォルトのプロパティの添え字
    * <P>デフォルトのプロパティが存在しない場合は -1 を返す。
    */
    public int getDefaultPropertyIndex() {
        return defaultPropertyIndex;
    }

    /**
    * Bean は、その利用者が最もよく使用すると考えられる
    * イベントを「デフォルト」のイベントとして持つことができる
    * @return:  getEventSetDescriptors によって返された
    *       EventSetDescriptor 配列内のデフォルトのイベントの添え字
    * <P>   デフォルトのイベントが存在しない場合は -1 を返す
    */
    public int getDefaultEventIndex() {
        return defaultEventIndex;
    }
}
```

Cart Bean のソース

```
package ShopCart;

import java.beans.*;

public class Cart extends Object implements java.io.Serializable {

    private static final String PROP_SAMPLE_PROPERTY = "SampleProperty";

    private String sampleProperty;

    private PropertyChangeSupport propertySupport;

    /** lineItems プロパティの値を保持 */
    public java.util.Vector lineItems;

    /** Cart を新規作成 */
    public Cart() {
        propertySupport = new PropertyChangeSupport ( this );
        lineItems = new java.util.Vector();
    }

    public String getSampleProperty () {
        return sampleProperty;
    }

    public void setSampleProperty (String value) {
        String oldValue = sampleProperty;
        sampleProperty = value;
        propertySupport.firePropertyChange (PROP_SAMPLE_PROPERTY, oldValue,
sampleProperty);
    }

    public void addPropertyChangeListener (PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener (listener);
    }
```

```
    public void removePropertyChangeListener (PropertyChangeListener listener){
        propertySupport.removePropertyChangeListener (listener);
    }

    /** lineItems プロパティの取得メソッド
    * @return: lineItems プロパティの値
    */
    public java.util.Vector getLineItems() {
        return this.lineItems;
    }

    /** lineItems プロパティの設定メソッド
    * @param lineItems lineItems プロパティの新しい値
    */
    public void setLineItems(java.util.Vector lineItems) {
        this.lineItems = lineItems;
    }

    /**
    * 渡された ID で指定された cartItems 内の品目の
    * 要素番号を返す
    */
    public int findLineItem(int pId) {
        System.out.println("Entering Cart.findLineItem()");
        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 ;
    }

    /** cartItems リストから cartItem を削除
    * @param: 削除する CartLineItem の pId ID
    */
```

```
    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()");
    }
}
```

ShopCart.jsp のソース

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

<html>
<head><title>Shopping Cart</title></head>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<body>
<h1> Shopping Cart </h1>

<jsp:useBean id="myCart" scope="session" class="ShopCart.Cart" />
<%
    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);
    }
    if (myOperation.equals("Delete")) {
        String s = request.getParameter("cdId");
        System.out.println(s);
        int idVal = Integer.parseInt(s);
        myCart.removeLineItem(idVal);
    }
    if (((Vector)session.getAttribute("myLineItems")).size() == 0) {
        %>
        <jsp:forward page="EmptyCart.jsp" />
        <%
    }
%>
```

```
<TABLE border=1>
<TR>
    <TH>ID</TH>
    <TH>CD Title</TH>
    <TH>Price</TH>
</TR>
<c:forEach var="item" items="${myLineItems}">
<TR>
    <TD><c:out value="${item.id}"/></TD>
    <TD><c:out value="${item.cdtitle}"/></TD>
    <TD><c:out value="${item.price}"/></TD>

    <TD>
        <form method=get action="ShopCart.jsp">
        <input type=hidden name=cdId value="<c:out value="${item.id}"/>">
        <input type=hidden name=cdTitle value="<c:out
value="${item.cdtitle}"/>">
        <input type=hidden name=cdPrice value="<c:out value="${item.price}"/>">
        <input type=submit name=operation value=Delete>
        </form>
    </TD>
</TR>
</c:forEach>

</TABLE>
<p>
<form method=get action="ProductList.jsp">
    <input type=submit value="Resume Shopping">
</form>
<form method=get action="PlaceOrder.jsp">
    <input type=submit value="Place Order">
</form>
<form method=get action="CancelOrder.jsp">
    <input type=submit value="Cancel Order">
</form>
</p>

</body>
</html>
```

EmptyCart.jsp のソース

```
<%@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>
```

PlaceOrder.jsp のソース

```
<%@page contentType="text/html"%>
<html>
<head><title>Place Order</title></head>
<body>
    <h1> Place Order </h1>
    <%
    session.invalidate();
    %>
    Your order has been placed.  Thank you for shopping.
    <p>
    <form method=get action="ProductList.jsp">
        <input type=submit value="Resume Shopping">
    </form>
    </p>
</body>
</html>
```

CancelOrder.jsp のソース

```
<%@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>
```

付録 B

CDShopCart 用のデータベーススクリプト

この付録では、CDShopCart アプリケーション用のデータベーススクリプトをまとめています。

これらのスクリプトは、次の Sun ONE Studio 4 Developer Resources ポータルサイトからダウンロードできる CDShopCart アプリケーションの zip ファイルにも含まれています。

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

PointBase データベース用のスクリプト

PointBase データベースに対しては、次の SQL スクリプトを使用してください。

```
drop table CD;

create table CD(
    id  int,
    cdtitle char(20),
    artistchar(20),
    countrychar(20),
    pricenumber(8,2),
    primary key(id));

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

commit;
```

Oracle データベース用のスクリプト

Oracle データベースに対しては、次の SQL スクリプトを使用してください。

```
//* sqlplus tutorial/tutorial@dbname @scriptname で実行 */
drop table CD;

create table CD(
    id  int,
    cdtitlechar(20),
    artistchar(20),
    countrychar(20),
    pricenumber(8,2),
    primary key(id));
grant all on CD to public;

insert into CD (id, cdtitle, artist, country, price)
  values (1, 'Yuan','The Guo Brothers', 'China',14.95);
insert into CD (id, cdtitle, artist, country, price)
  values (2, 'Drums of Passion','Babatunde Olatunji',
'Nigeria',16.95);
insert into CD (id, cdtitle, artist, country, price)
  values (3, 'Kaira','Tounami Diabate', 'Mali',13.95);
insert into CD (id, cdtitle, artist, country, price)
  values (4, 'The Lion is Loose','Eliades Ochoa', 'Cuba',12.95);
insert into CD (id, cdtitle, artist, country, price)
  values (5, 'Dance the Devil Away','Outback', 'Australia',14.95);
commit;
```

Microsoft SQLServer データベース用のスクリプト

Microsoft SQLServer データベースに対しては、次の SQL スクリプトを使用してください。

```
use cdcat
go
drop table CD
go

create table CD(
    id  int,
    cdtitlevarchar(20) null,
    artistvarchar(20) null,
    countryvarchar(20) null,
    pricemoney null,
PRIMARY KEY(id))
go
grant all on CD to public
go

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

IBM DB2 データベース用のスクリプト

IBM DB2 データベースに対しては、次の SQL スクリプトを使用してください。

```
drop table CD

create table CD(
    id  int not null primary key ,
    cdtitlechar(20),
    artistchar(20),
    countrychar(20),
    pricenum(8,2))

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

索引

F

H

I

J

L

M

N

O

P

Q

R

S

T

V

W

あ

く

こ

さ

し

て

ふ

め

ゆ