# Sun™ ONE Studio 5
# Web アプリケーション
# チュートリアル

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No. 817-3297-10
2003 年 7 月, Revision A

**原典：** *Sun ONE Studio 5 Web Application Tutorial*
Part No: 817-2320-10
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™ ONE Studio 5 Web アプリケーションに導入されている次の機能の使用方法を学びます。

- Java™ サーブレットおよび 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
- チュートリアルアプリケーションのテストのための Sun ONE Application Server への配備と実行

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/software/sundev/jde/documentation/`

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。ほとんどの手順で Sun ONE Studio 5 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 の場合は、次のように入力します。

```
% cd MyWorkSpace/MyPackage
```

# お読みになる前に

このチュートリアルでは、Java 2 Platform, Enterprise Edition (J2EE™) Blueprints リソースに記載されているアーキテクチャに準拠したアプリケーションを作成します。J2EE 準拠のアプリケーションを作成、開発、配備するために Sun ONE Studio 5, Standard Edition の機能を使用するには、このチュートリアルが役に立ちます。

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、CDShopCart アプリケーションに必要なソフトウェア、Sun ONE Studio 5 統合開発環境 (IDE) の起動方法、Sun ONE Application Server Web サーバーの起動方法を説明しています。また、チュートリアルのデータベース表を作成する方法も説明しています。

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、CDShopCart 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 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。

付録 C では、Oracle データベースを使用してこのチュートリアルアプリケーションを作成し、実行する方法を説明しています。

# 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。<br>DIR を使用してすべてのファイルを表示します。<br>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**<br>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。<br>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。<br>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越える場合、バックスラッシュは、継続を示します。 | machinename% grep '^#define \<br>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 5 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 開発者リソースポータルのドキュメントサイト (http://sun.co.jp/software/sundev/jde/documentation/) および `docs.sun.com`™ (http://docs.sun.com) から入手できます。

`docs.sun.com` Web 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 5 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。

  - 『Sun ONE Studio 5, Standard Edition リリースノート』

- 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 5 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、アプリケーションサーバーの情報、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 5, Standard Edition インストールガイド』
  - 『Sun ONE Studio 4, Mobile Edition インストールガイド』

- Sun ONE Studio 5 プログラミングシリーズ (PDF 形式)

  Sun ONE Studio 5 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』

    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』

    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』

    Sun ONE Studio 5 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』

    Sun ONE Studio 5 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』

    Sun ONE Studio 5 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 5 チュートリアル (PDF 形式)

  Sun ONE Studio 5, Standard Edition の主な機能の活用方法を紹介しています。

  - 『Sun ONE Studio 5 Web アプリケーションチュートリアル』

    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 5 J2EE アプリケーションチュートリアル』

    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』

    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 5 IDE から参照できます。ヘルプを開くには、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 5 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 開発者リソース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 5 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。

# 技術サポートへの問い合わせ

製品についての技術的なご質問がございましたら、以下のサイトからお問い合わせください (このマニュアルで回答されていないものに限ります)。

`http://sun.co.jp/service/contacting`

# 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 5 Web アプリケーション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。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。アプリケーションファイルとは、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 5 Developer Resources ポータル (`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の入手とインストール

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。

- Sun ONE Studio 5, Standard Edition ソフトウェア
  - Sun ONE Studio 5, Standard Edition 統合開発環境 (IDE)
  - Sun ONE Application Server 7 ソフトウェア

Sun ONE Studio 5, Standard Edition のインストーラでは、両方の製品がインストールされます。ただし、Sun ONE Application Server 7 のサポートされているバージョンが見つかった場合は、インストールされません。たとえば、Solaris™ 9 Update 2 オペレーティング環境には、Sun ONE Application Server 7 が含まれます。

Sun ONE Studio 5, Standard Edition ソフトウェアは、次の場所から入手できます。

- Sun ONE Studio 5, Standard Edition の CD
- Sun ONE ポータル (`http://wwws.sun.com/software/sundev/jde/`)
- Sun ONE Developer Resources ポータル (`http://forte.sun.com/ffj/`)

- Java™ 2 Software Development Kit (J2SE™ SDK)、バージョン 1.4.1_02 以降

  システムに J2SE SDK がなかった場合、Sun ONE Studio 5, Standard Edition のインストーラは実行されません。システムに J2SE SDK がある場合は、インストーラが起動し、使用している J2SE SDK のバージョンが、プラットフォーム用の IDE および Sun ONE Application Server 7 に必要なバージョンかどうかが確認されます。必要なバージョンではなかった場合は、正しいバージョンをインストールする必要があることを示すメッセージが表示され、インストーラが終了します。J2SE SDK は、IDE と同じ場所から入手できます。

- PointBase ネットワークサーバーデータベースソフトウェア

  このチュートリアルでは、PointBase データベースを使用します。PointBase は、Sun ONE Studio 5, Standard Edition ソフトウェアと同時に、Sun ONE Application Server 7 ソフトウェアを含むサブディレクトリにインストールされます。IDE とアプリケーションサーバーを別々にインストールした場合は、アプリケーションサーバーに PointBase ソフトウェアが含まれない場合があります。PointBase ソフトウェアが含まれない場合は、PointBase ソフトウェアをダウンロードし、手動でインストールする必要があります。この方法については、『Sun ONE Application Server 7 入門ガイド』を参照してください。また、付録 C では、Oracle データベースでアプリケーションを作成する方法を説明しています。

- チュートリアルの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 スクリプトについては、付録 B で説明しています。SQL スクリプトのファイルは、CDShopCart チュートリアルのアプリケーションファイルに含まれます。アプリケーションファイルは Sun ONE Studio 5 Developer Resources ポータルから入手できます。10 ページの「チュートリアルのデータベース表の作成」では、PointBase データベースにチュートリアルのデータベース表をインストールする方法を説明しています。付録 C では、Oracle データベースにチュートリアルの表をインストールする方法を説明しています。

- Web サーバー

  チュートリアルは 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、Sun ONE Application Server 7 の Web サーバーコンポーネントを使用します。

- 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。Web ブラウザには、Netscape Communicator™ または Microsoft Internet Explorer を使用できます。

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 5 開発者リソースポータルのドキュメントサイト (`http://sun.co.jp/software/sundev/jde/documentation/`) をご覧ください。

# ソフトウェアの起動

この節では、ソフトウェアのインストール後に Sun ONE Studio 5 IDE と Sun ONE Application Server 7 を起動する方法を説明します。

## IDE の起動

Sun ONE Studio 5 IDE は、複数の方法で起動できます。ここでは、1 つの方法だけを示します。ほかの方法については、『Sun ONE Studio 5, Standard Edition インストールガイド』を参照してください。

IDE を起動するには、次のようにします。

- **Sun ONE Studio 5 IDE を起動するには、プログラムの実行可能ファイルを実行します。**
  - Microsoft Windows では、「スタート」->「プログラム」->「Sun Microsystems」->「Sun ONE Studio 5 SE」->「Sun ONE Studio 5 SE」を選択します。
  - Solaris、UNIX、および Linux の各環境では、次のように端末ウィンドウで `runide.sh` スクリプトを実行します。

```
$ sh s1studio-install-directory/bin/runide.sh
```

ここで *s1studio-install-directory* 変数は、IDE のホームディレクトリです。デフォルトでは、`/opt/studio5_se` (UNIX のスーパーユーザー) です。

# アプリケーションサーバーの起動

ここに示す手順を実行する前に、アプリケーションサーバーのドメインへの書き込み権が必要です。デフォルトのドメインはインストール時に作成され、アクセスにはスーパーユーザーの権限が必要です。スーパーユーザーの権限とは、Microsoft Windows システムでは administrator の権限、Solaris または Linux の環境では root 権限です。使用しているユーザー ID にスーパーユーザーの権限がある場合は、以下に示す手順で、デフォルト設定を使用してアプリケーションサーバーを起動できます。スーパーユーザーの権限を持たない標準ユーザーは、5 ページの「管理サーバーの起動 (標準ユーザー)」に示す手順に従う必要があります。

## 管理サーバーの起動 (スーパーユーザーの場合)

以前に管理サーバーを起動した場合は、実行中かどうかを確認します。詳細は、9 ページの「Sun ONE Application Server 7 がデフォルトのサーバーであることの確認」を参照してください。管理サーバーを初めて起動する場合は、ここから始めます。

管理サーバーを起動するには、次のようにします。

1. **IDE で、エクスプローラの「実行時」タブを選択します。**

   エクスプローラの「実行時」タブに「サーバーレジストリ」ノードが表示されます。このノードの下には、インストールされているすべての Web サーバーとアプリケーションサーバーのサブノードがあります。また、デフォルトのサーバーを示すノードがあります。

2. **「サーバーレジストリ」ノードを選択します。**

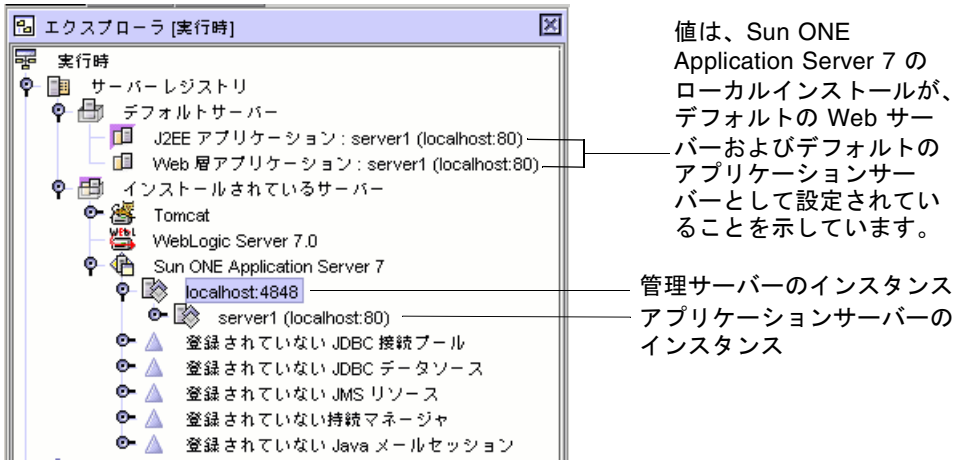
   管理サーバーを起動するかどうかを確認するウィンドウが開きます。この管理サーバーは、デフォルトドメインの管理サーバーを指し、権限を持つユーザーだけが実行できます。

3. **「了解」をクリックしてデフォルトの管理サーバーを起動します。**

   IDE によってデフォルトの管理サーバーが起動し、Sun ONE Application Server 7 が IDE のデフォルトのアプリケーションサーバーに設定されます。

4. **「サーバーレジストリ」ノード、「インストールされているサーバー」ノード、「Sun ONE Application Server 7」ノードの順に展開します。**

   エクスプローラの「サーバーレジストリ」ノードは次のようになっています。




8 ページの「アプリケーションサーバーのインスタンスの起動」に示す手順に従って、サーバーのインスタンスを起動します。

## 管理サーバーの起動 (標準ユーザー)

ユーザー ID にスーパーユーザーの権限がない場合は、ここに示す手順を実行する前に、スーパーユーザーにドメインを作成してもらう必要があります。作成方法は、『Sun ONE Studio 5, Standard Edition インストールガイド』で説明しています。

以前に IDE を起動し、管理サーバーを作成して起動した場合は、9 ページの「Sun ONE Application Server 7 がデフォルトのサーバーであることの確認」の手順に従って、管理サーバーが実行中かどうかを確認します。管理サーバーを初めて起動する場合は、ここから始めます。

この手順を実行する前に、ドメインに関するプロパティの値をいくつか確認する必要があります。これらの値は、管理者に確認します。プロパティの値を確認したら、次の表に記入してください。

**表 1-1** 管理サーバーのプロパティの値

| 管理サーバーのプロパティ | 値 |
|---|---|
| 管理サーバーホスト | |
| 管理サーバーポート | |
| ユーザー名 | |
| ユーザーパスワード | |
| ドメイン | |

管理サーバーを起動するには、次のようにします。

1. IDE で、エクスプローラの「実行時」タブを選択します。

   エクスプローラの「実行時」タブに「サーバーレジストリ」ノードが表示されます。このノードの下には、インストールされているすべての Web サーバーとアプリケーションサーバーのサブノードがあります。また、デフォルトのサーバーを示すノードがあります。

2. 「サーバーレジストリ」ノードを選択します。

   管理サーバーを起動するかどうかを確認するウィンドウが開きます。この管理サーバーは、デフォルトドメインの管理サーバーを指し、権限を持つユーザーだけが実行できます。

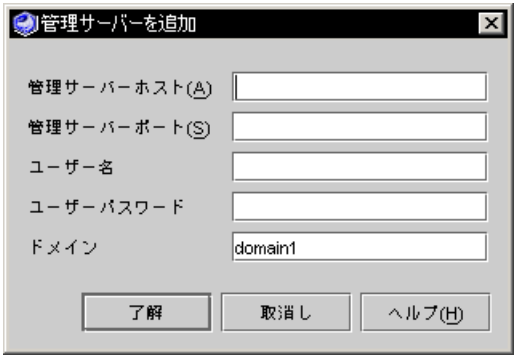
   「了解」をクリックすると、使用できない管理サーバーが作成され、起動されてしまいます。「取消し」をクリックします。

3. 管理サーバーを IDE に追加します。

   a. 「サーバーレジストリ」ノード、「インストールされているサーバー」ノードを展開します。

   b. 「Sun ONE Application Server 7」ノードを右クリックして、「管理サーバーを追加」を選択します。

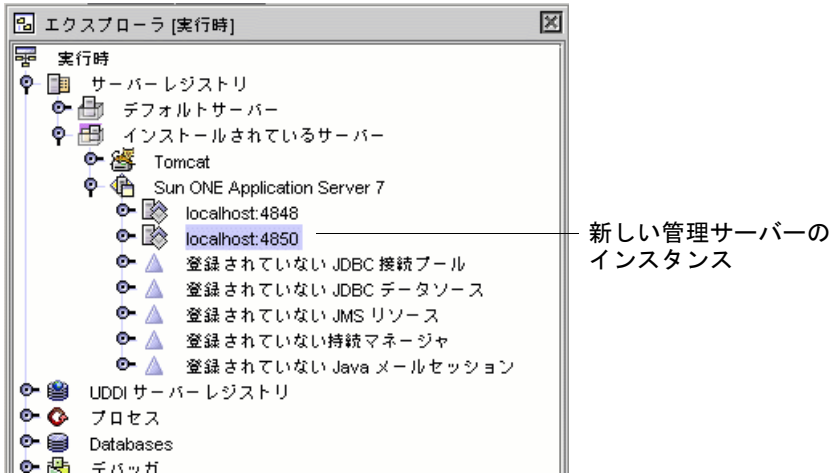
      「管理サーバーを追加」ダイアログが表示されます。




   c. 表 1-1 の値を入力し、「了解」をクリックします。

      ここで、「管理サーバーに接続できません。管理サーバーがローカルの場合、サーバーを起動します。」というメッセージが表示された場合、「了解」をクリックして、エラーウィンドウを閉じます。管理サーバープロセスの起動状況を示す進捗ウィンドウが表示されます。

      エクスプローラに、新しい管理サーバーノードが作成されます。次の画面イメージでは、新しい管理サーバーのホストは localhost で、ポート番号は 4850 です。




4. アプリケーションサーバーのインスタンスを作成します。

   a. 新しい管理サーバーのノードを右クリックして、「サーバーインスタンスを作成」を選択します。

      「サーバーインスタンスの値を入力」ダイアログが表示されます。

   b. 名前とポート番号を入力します。

      たとえば、「`MyServer`」および「`4855`」と入力します。

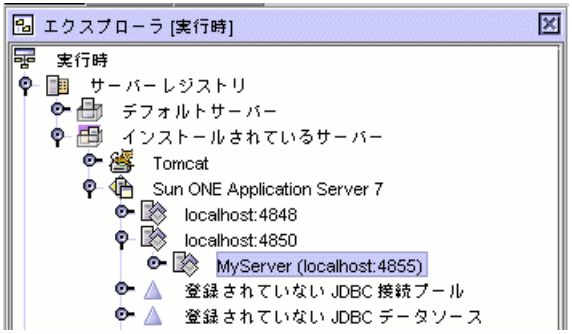
---

**注 –** Solaris システムと Linux システムでは、1024 未満のポート番号は予約されています。1024 以上のポート番号を使用してください。すべてのシステムで、デフォルトのアプリケーションサーバーで使用されるポート番号 80 は使用しないでください。

---

   c. 「了解」をクリックします。

      管理サーバーが起動し、出力ウィンドウとステータスバーにメッセージが表示されます。IDE に新しいサーバーインスタンスが作成されます。




5. 新しいサーバーインスタンスを右クリックし、「デフォルトとして設定」を選択して、デフォルトのアプリケーションサーバーと Web サーバーを設定します。

6. 「デフォルトサーバー」ノードを展開して、この処理が正常に実行されたことを確認します。

   J2EE アプリケーションと Web 層アプリケーションのデフォルトサーバーに、新しいサーバーがデフォルトとして表示されます。


次の項に示す手順に従って、サーバーのインスタンスを起動します。

## アプリケーションサーバーのインスタンスの起動

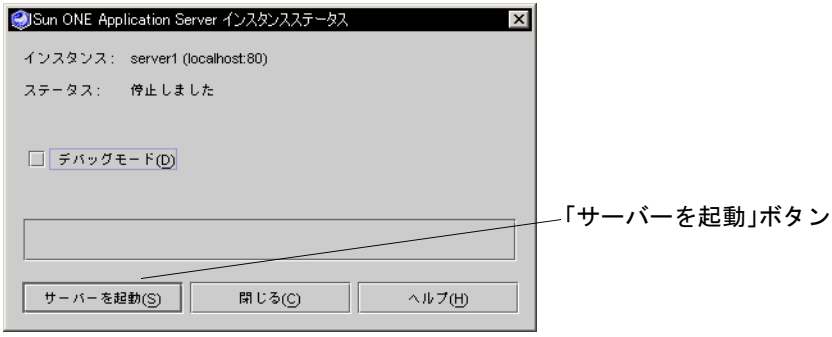
開発中にアプリケーションのテスト配備および配備を行ったとき、管理サーバーが実行中の場合は、IDE によってアプリケーションサーバーのインスタンスが自動的に起動されます。ここでは、この後に示す操作を行うために、アプリケーションサーバーのインスタンスを手動で起動します。

サーバーのインスタンスを起動するには、次のようにします (すべてのユーザー)。

1. アプリケーションサーバーのノードを右クリックして、「ステータス」を選択します。

**注 –** このノードが表示されない場合は、管理サーバーのインスタンスのノードを選択し、「再表示」を選択します。

次に示す「Sun ONE Application Server インスタンスステータス」ダイアログが表示されます (インスタンス名は異なる場合があります)。




2. **「サーバーを起動」ボタンをクリックします。**

   ダイアログに「サーバーを停止」ボタンがある場合は、サーバーはすでに実行中です。

   Microsoft Windows システムでは、進捗メッセージを示すコマンドウィンドウが表示されます。

   サーバーが起動すると、「Sun ONE Application Server インスタンスステータス」ダイアログに「ステータス: 実行中」と表示されます。

3. **「閉じる」をクリックします。**

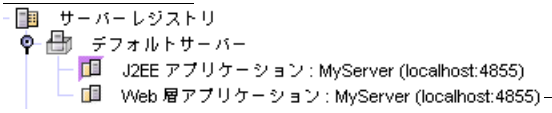
   10 ページの「チュートリアルのデータベース表の作成」に進みます。

## Sun ONE Application Server 7 がデフォルトのサーバーであることの確認

Sun ONE Application Server 7 を以前に起動した場合は、次の手順で、このサーバーがまだデフォルトのサーバーであることを確認します。

1. **IDE で、エクスプローラの「実行時」タブを選択します。**
2. **「サーバーレジストリ」ノード、「デフォルトサーバー」サブノードを展開します。**

   「Web 層アプリケーション」ノードのラベルが「*server-instance* (*server-hostname*:*server-port-number*)」の場合は、Sun ONE Application Server 7 がデフォルトの Web サーバーになっています。次の節に進みます。それ以外の場合は、以下の手順に進みます。




値は、Sun ONE Application Server 7 のローカルインストールが、デフォルトの Web サーバーとして設定されているところを示しています。

3. 「インストールされているサーバー」ノードの下で Web サーバーのインスタンスを右クリックし、「デフォルトとして設定」を選択します。

   サーバーが、J2EE と Web 層アプリケーションのデフォルトのサーバーとして設定されます。

# チュートリアルのデータベース表の作成

CDShopCart チュートリアルに入る前に、PointBase ネットワークサーバー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。データベース表の作成には、付録 B にある SQL スクリプトを使用します。CDShopCart チュートリアルの `cdshop.zip` ファイルに含まれる `CDCatalog_pb.sql` というスクリプトファイルも使用できます。この zip ファイルは、次のサイトから入手できます。

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

---

**注 –** Sun ONE Studio 5 IDE と Sun ONE Application Server 7 を同時にインストールしなかった場合は、『Sun ONE Studio 5, Standard Edition インストールガイド』に示す手順に従って、IDE とアプリケーションサーバーを PointBase データベースソフトウェアに接続する必要があります。この操作は、次の手順を実行する前に行う必要があります。

---

PointBase データベースにチュートリアルの表をインストールするには、次のようにします。

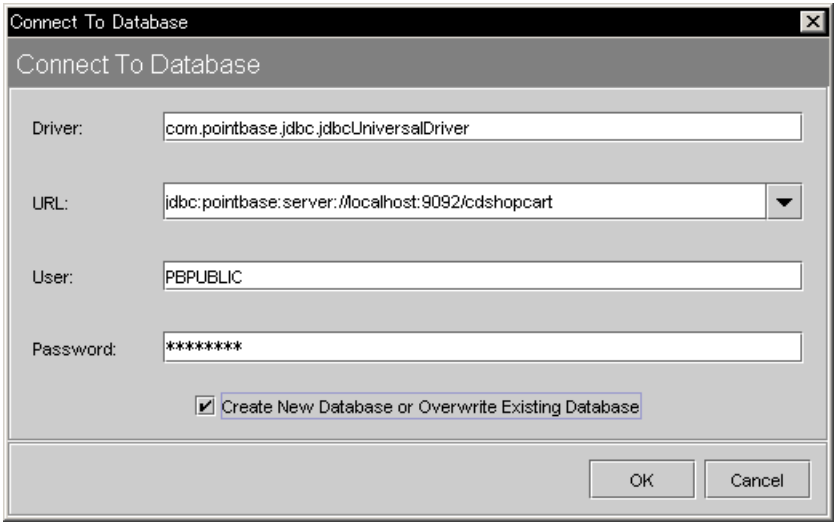
1. 「ツール」->「PointBase ネットワークサーバー」->「サーバーを起動」を選択して、IDE から PointBase サーバーを起動します。

   「PointBase ネットワークサーバー」ウィンドウが表示されます。ウィンドウを最小化します。

2. 「ツール」->「PointBase ネットワークサーバー」->「コンソールを起動」を選択して、IDE から PointBase コンソールを起動します。

   「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 `sample` 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. 87 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。

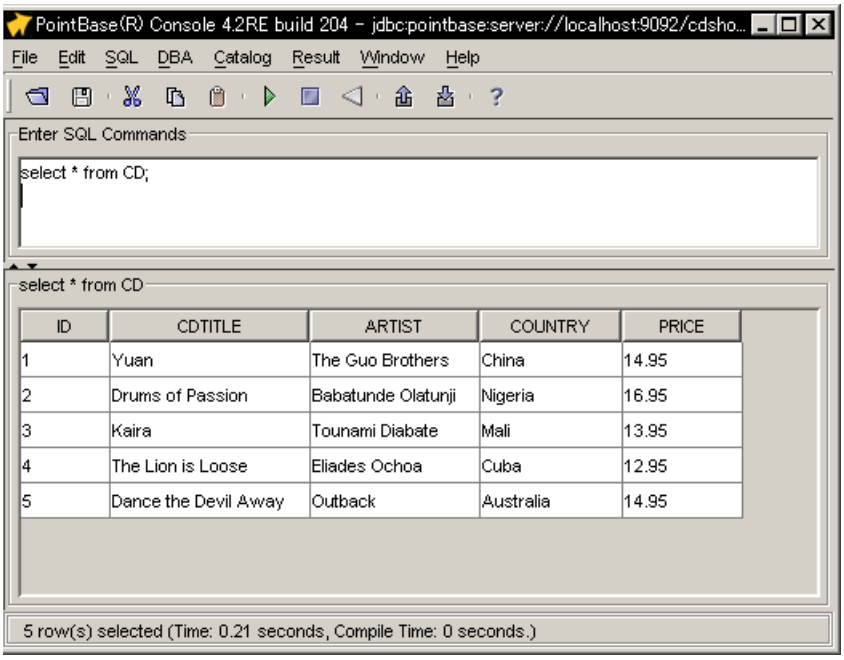
6. 「SQL」->「Execute All」を選択します。

   スクリプトが実行され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、まだ作成されていない表に対する DROP 文があるためです。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。

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、表が作成されたことを確認します。

```
select * from CD;
```

8. 「SQL」->「Execute」を選択します。
   コンソールに、CD 表が表示されます。




---

**注 –** 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。

CDCatalog スクリプトでは、表 1-2 に示すデータベーススキーマが作成されます。

表 1-2　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 1-3 に示すレコードが格納されます。

**表 1-3** 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 |

これで、チュートリアルに取り組む準備ができました。作成するアプリケーションの概要については、第 2 章を参照してください。アプリケーションの作成をすぐに始めるには、第 3 章に進んでください。

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 5, Standard Edition の機能を使用して Web アプリケーションの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 Apache のタグライブラリなどの Sun ONE Studio 5 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 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と、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、CD カタログページが再表示され、CD を追加選択できます。利用者はこの手順を何度でも繰り返すことができ、また同じ 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、Shopping Cart 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、最後の 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。

## アプリケーションの機能仕様

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。

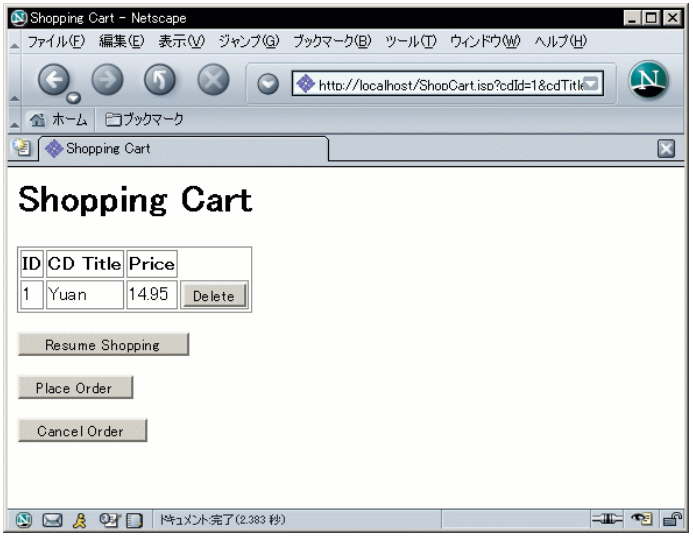
CDShopCart アプリケーションを実行するには、次のようにします。

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 ページによって作成されます。


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。

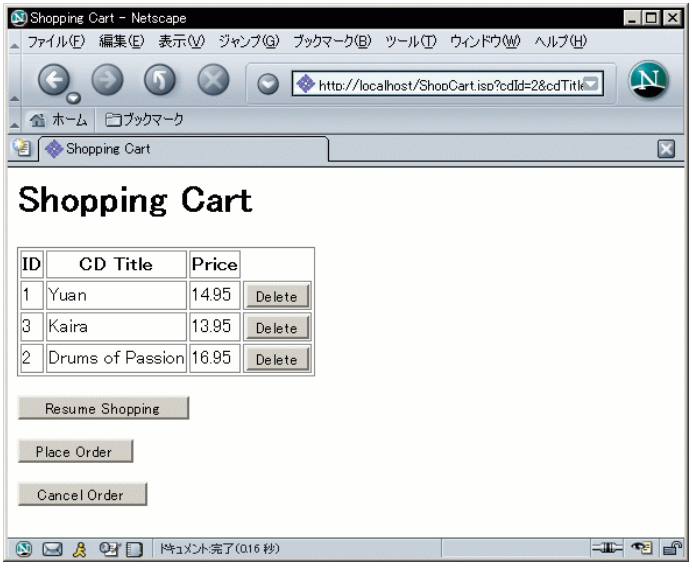
   この操作によって、Shopping Cart ページが表示され、選択した CD が表示されます。このページは、ShopCart JSP ページによって作成されます。



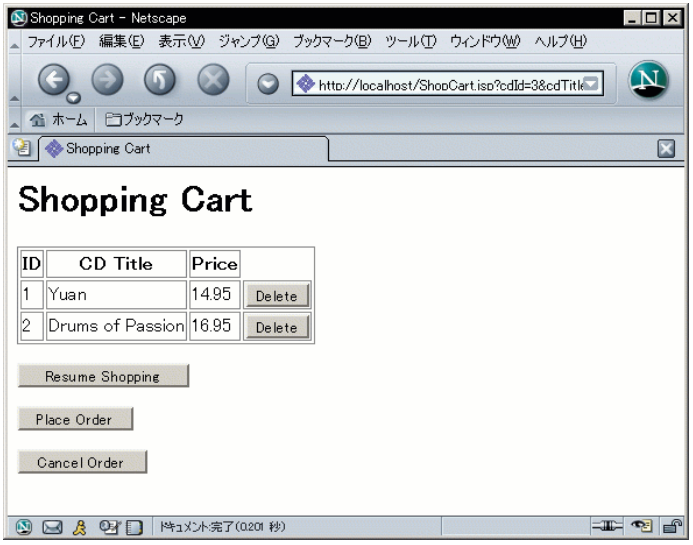

3. 別の CD を追加するには、「Resume Shopping」をクリックします。再び CD Catalog List ページが表示されます。

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。

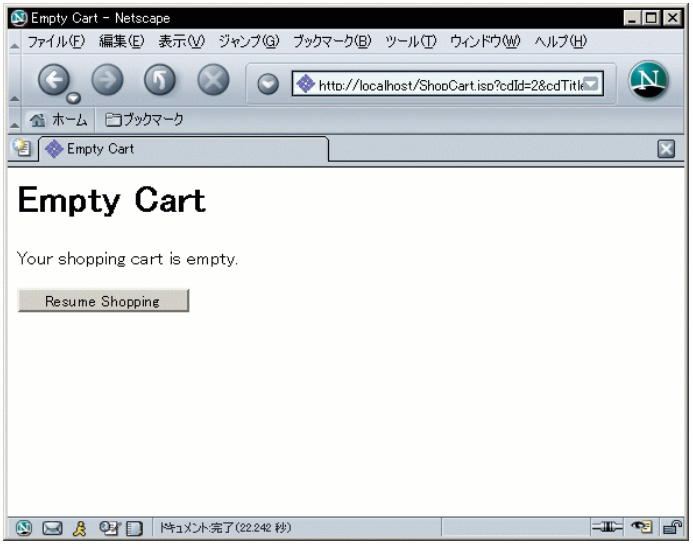
5. 購入する CD をすべて選択するまで、手順 2 と手順 3 を繰り返します。
   Shopping Cart ページに選択した CD がすべて表示されます。同じ CD を何度か選択した場合は、それぞれ別々の行に表示されます。




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。

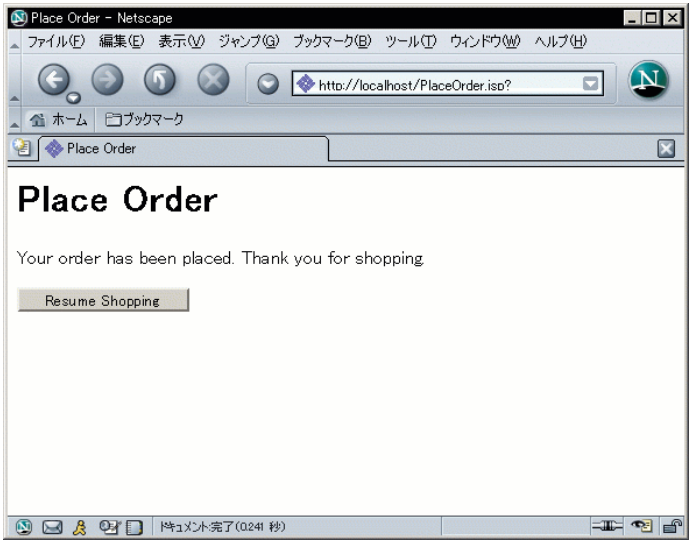



表内の最後の CD を取り除くと、Empty Cart JSP ページが表示されます。




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。

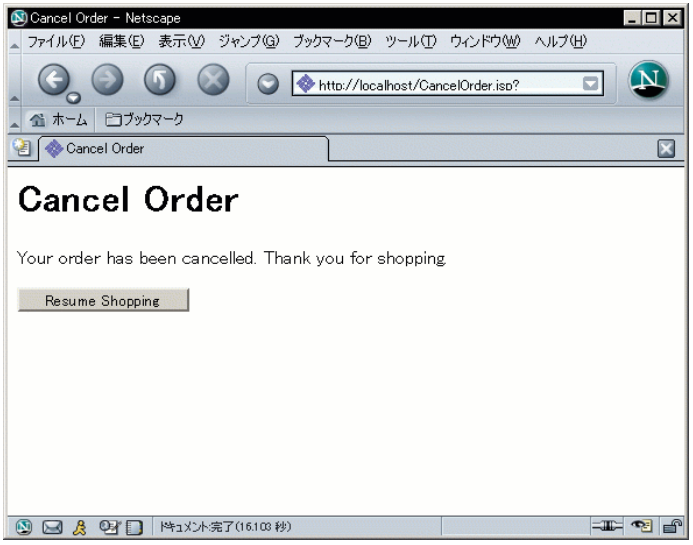
   Place Order ページが表示されます。このページは、PlaceOrder JSP ページによって作成されます。




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配備および使用可能な Web 資源の最小単位です。『Java Servlet Specification Version 2.3』で導入され、Sun ONE Studio 5 IDE 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、必要なデータオブジェクトのデフォルトのバージョン、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。

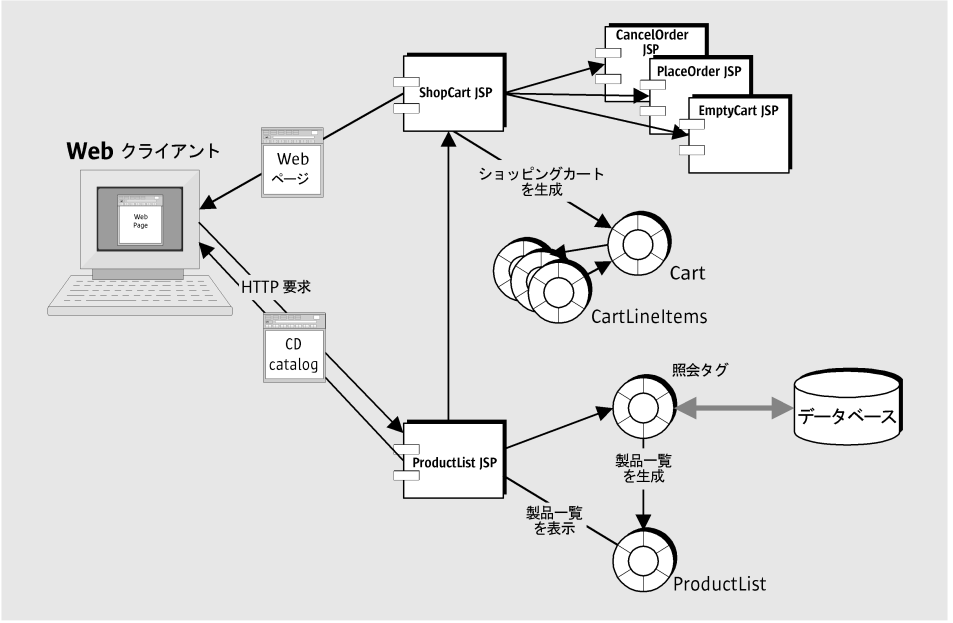



図 2-1　CDShopCart アプリケーションの構造

# 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、アプリケーションの各ページを表示する Web ブラウザです。

- サービスコンポーネント (Web モジュール)。次の要素が含まれます。
  - 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 - 購入のために選択された 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最後の CD が削除され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 1 品目 (CD) を表します。
- CartLineItemBeanInfo- Bean Info コンポーネント。多重定義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 4 つの JSP ページで構成される 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します。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) のうち、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、コアタグを使用して表に CD を表示します。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 品目のデータがこの 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 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。カートにあった最後の CD が取り除かれると、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。また、「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 5 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があります。

第 3 章では、Sun ONE Studio 5, Standard Edition の次の機能の使用方法を学びます。

- 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 5 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。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 モジュールの開発の詳細については説明していません。29 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 CD カタログのデータをフェッチし、そのデータを CD Catalog List Web ページに表示する ProductList JSP ページを作成します。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。このとき、データアクセスとデータ表示の機能に JSTL タグを利用します。

34 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 のデータをフェッチする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 ページに HTML 形式で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。56 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。

## サポート要素の作成

ShopCart JSP ページのサポート要素には、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。

46 ページの「CartLineItem Bean の作成」では、品目 (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 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。また、52 ページの「Cart Bean の作成」では、選択された品目 (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、カート内の品目を追加・削除するためのメソッドも含まれています。

62 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。64 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 65 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たびに、IDE によって、その Web モジュールが内部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

これらの手順は、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、ソースファイルを確認したり、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したりすると便利です。CDShopCart アプリケーションは、Sun ONE Studio 5 Developer Resources ポータル (`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`) から入手できます。

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 5 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。

この章の構成は次のとおりです。



各コンポーネントは作成するたびにテストをします。この章の作業を終えると、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。

**参考 –** この章で説明している JSP ページや JavaBeans コンポーネントの全ソースコードは、付録 A に掲載しています。

## 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、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、Sun ONE Studio 5, Standard 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 5 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。

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 5 IDE の Web モジュール機能は、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配置する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 5 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

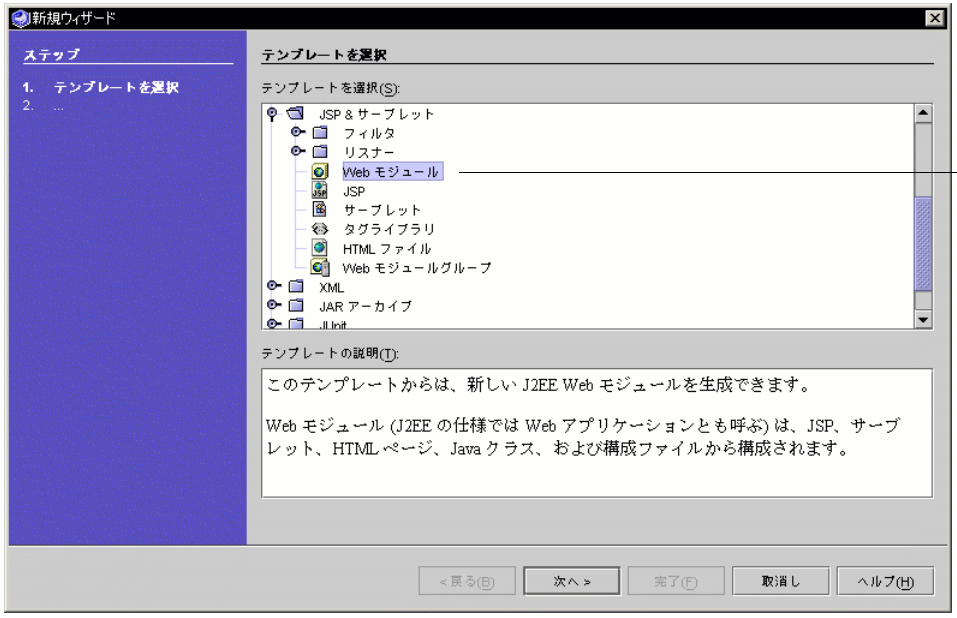
この節では、CDShopCart アプリケーションの Web モジュールを作成します。Sun ONE Studio 5 の Web モジュール機能を使用すると、Web モジュールディレクトリの新規作成と、既存のディレクトリを変換した Web モジュールの作成ができます。

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」タブで、デフォルト (または任意の) ファイルシステムを選択し、「ファイル」->「新規」を選択して新規ウィザードを表示します。**

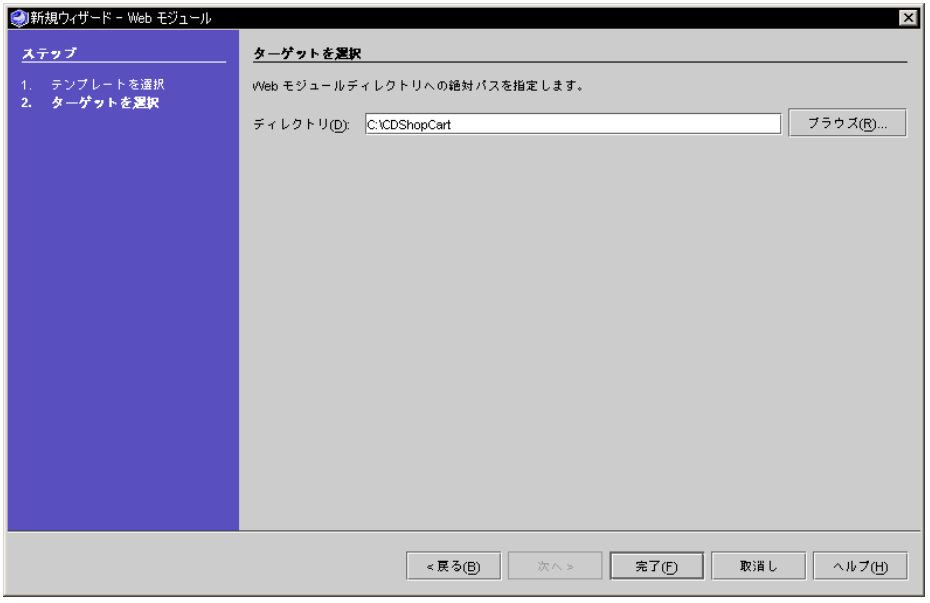
   このウィザードが提供するテンプレートを利用して、さまざまなオブジェクトを作成することができます。

2. **「JSP & サーブレット」ノードを開き、「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ページを表示します。

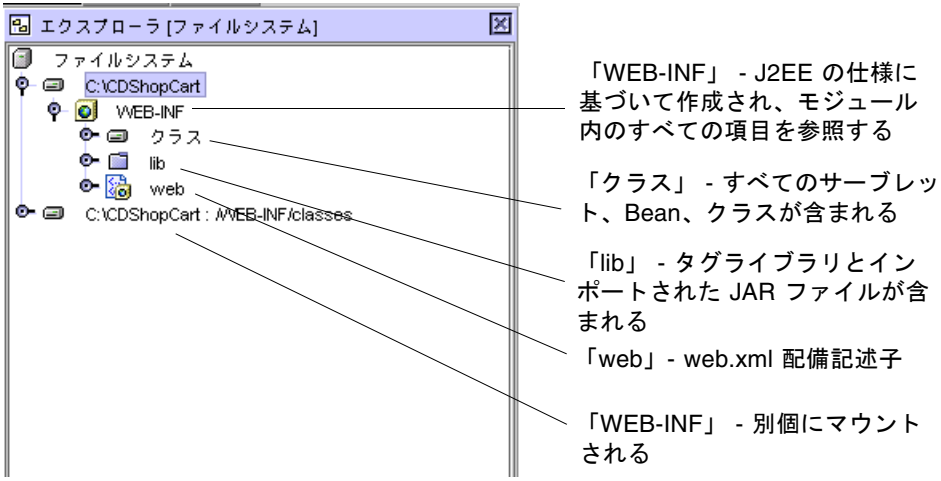
4. 「ディレクトリ」フィールドにデフォルトで表示されるディレクトリ指定を消去して、*your-directory*\CDShopCart と入力します。




5. 「完了」をクリックします。

   新しい Web モジュール、CDShopCart がエクスプローラに作成されます。「デフォルトプロジェクト」ウィンドウに Web モジュールの代替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。




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。

## Web アプリケーションのコンテキストルートの設定

CDShopCart Web アプリケーションを配備するには、Web サーバーのコンテキストルートを指定する必要があります。コンテキストルートは、WEB-INF フォルダのプロパティとして、次の手順で設定します。

1. WEB-INF フォルダのプロパティを表示します。

   プロパティのウィンドウは、エクスプローラの下に表示されます。これは静的なプロパティウィンドウで、選択したノードのプロパティが表示されます。WEB-INF ファイルを選択してそのプロパティを表示します。WEB-INF フォルダを右クリックし、「プロパティ」を選択して、プロパティを表示することもできます。この場合は、動的なプロパティウィンドウが開きます。このウィンドウには、選択するノードに関係なく、WEB-INF フォルダのプロパティだけが表示されます。

2. 「コンテキストルート」プロパティの値フィールドをクリックします。

3. 「**`/CDShopCart`**」と入力し、Return または Enter を押します。

# 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、データを表示用に書式設定するには、JSTL (JSP Standard Tag Library) タグを利用します。JSTL は、Apache Foundation のプロジェクトの一つである Jakarta プロジェクトで生まれたタグライブラリです。Sun ONE Studio 5 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 種類のコードを含めることができます。

- 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 - 次の 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 ページをセッションに組み込むかなど、JSP ページに関するグローバルな情報を宣言するときに使用する要素です。
  - 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 オブジェクトを使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 クラスが、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 5 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` - コアタグや 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次のとおりです。

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

たとえば、`core` タグライブラリは次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` タグの構文は次のとおりです。

```
<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 タグの構文は次のとおりです。

```
<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。

## Web モジュールへの JSTL タグライブラリの追加

JSTL タグライブラリの JAR ファイルである `standard.jar` と、サポートする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、「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。**

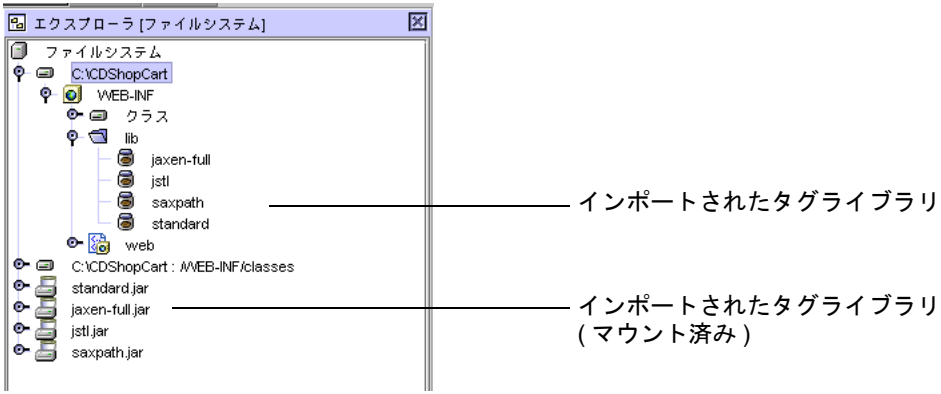
   JSP タグライブラリ・リポジトリブラウザが表示されます。


2. standard ファイルが選択されていることを確認して、「了解」をクリックします。

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

3. 「lib」ノードの下に、ファイルが表示されることを確認します。

   このチュートリアルで使用するタグを格納している jstl.jar および standard.jar ファイルが「lib」ノードの下に表示されます。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。




---

**参考** – 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 5 の クラスパス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。

---

# CD Catalog List ページの作成

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します。この仕組みは、次の要素で構成されます。

- すべてのコードを含む JSP ページ (ProductList)
- コードで参照するタグライブラリのタグライブラリ宣言
- データベースに接続するための setDataSource タグ
- CD データをフェッチするための query タグ
- CD データを表示するための反復子タグ
- CD の行ごとの「Add」ボタン

図 3-1 のようなページを作成します。

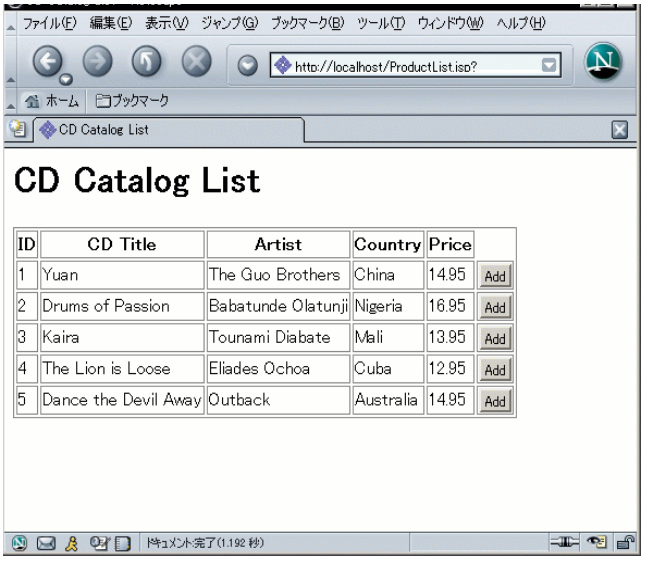



図 3-1　　CD Catalog List ページ

## ProductList JSP ページの作成

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します。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」->「すべてのテンプレート」を選択します。**

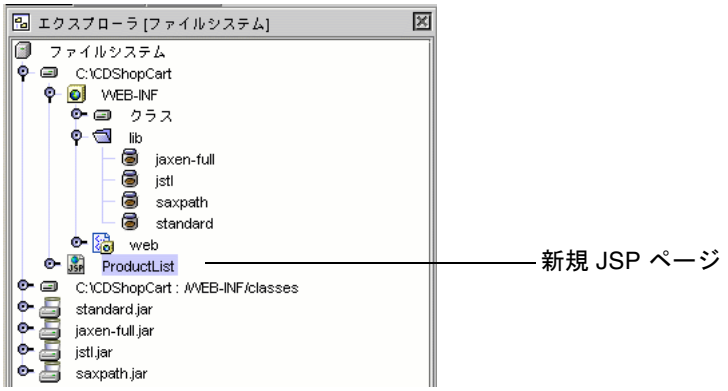
   新規ウィザードが表示されます。

2. JSP とサーブレットのノードを展開し、その下の JSP ノードを選択し、「次へ」をクリックします。

   新規ウィザードの「新規オブジェクト名」ページが表示されます。

3. 「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、35 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。次の手順では、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. この行の下で、次のようにコアおよび 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。

```
<%@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。<body> タグの下に、ページの見出し (<H1> タグを使用) を追加してから、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" />
```

Oracle のデフォルトのポート番号は 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。

## 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。

query タグは driver タグのすぐ後に挿入します。

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 を使用します)。

```
<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

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 の構文は次のとおりです。

```
<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、表にデータを格納します。

```
<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>
```

次の手順で、さらにコードを追加します。

## 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。利用者が 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。このコードは前の手順で作成したコードの直後に記述します。

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. 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。

---

**参考** – 70 ページの「ProductList.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。

---

## ProductList JSP ページのテスト

ProductList JSP ページをコンパイルし、Web アプリケーションを配備し、ProductList ページを実行することで作業内容をテストします。IDE によってデフォルトのブラウザが自動的に起動され、ページが表示されます。

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。このウィンドウは最小化します。

2. **Sun ONE Application Server 7 管理サーバーが実行中であり、アプリケーションサーバーがデフォルトの Web サーバーであることを確認します。**

   詳細は、9 ページの「Sun ONE Application Server 7 がデフォルトのサーバーであることの確認」を参照してください。

3. **Web モジュールの「コンテキストルート」プロパティが `/CDShopCart` に設定されていることを確認します。**

   詳細は、33 ページの「Web アプリケーションのコンテキストルートの設定」を参照してください。

4. **エクスプローラの「ファイルシステム」タブで CDShopCart Web モジュールを右クリックし、コンテキストメニューから「すべてを構築」を選択します。**

   出力ウィンドウのメッセージ領域でステータスメッセージを確認します。問題がなければ、最後に「完了」と表示されます。出力ウィンドウにエラーが表示されている場合は、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。70 ページの「ProductList.jsp のソース」に、ソースコードを掲載しています。

5. **CDShopCart Web モジュールを右クリックし、「配備」を選択します。**

   アプリケーションが配備されたら、進捗ウィンドウが閉じます。

6. **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。

   Sun ONE Application Server 7 が実行中ではない場合は、IDE によって起動されます。Microsoft Windows システムでは、コマンドウィンドウが開き、ログメッセージが表示されますが、このウィンドウは最小化してもかまいません。

   サーブレットが実行されると、IDE によって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。

   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ことを選択された CD を表示する仕組みを作成します。この仕組みは、次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。多重定義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 ごとに用意する
- 「Cancel Order」「Resume Shopping」「Place Order」ボタン - それぞれに対応する機能を実装する

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。

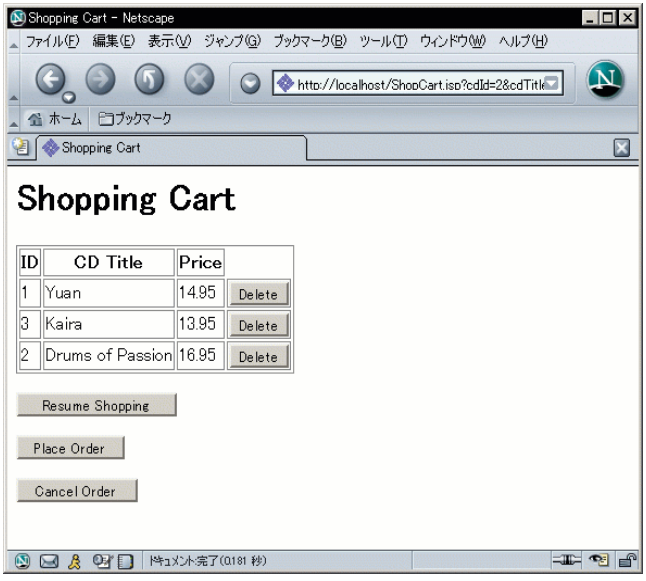



図 3-2　Shopping Cart ページ

# 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ありません。したがって、クラスサブディレクトリの下にクラス (Bean) を保持する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。

---

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. パッケージ名として **`ShopCart`** と入力して、「完了」をクリックします。

   新しい ShopCart パッケージがエクスプローラに表示されます。
3. ShopCart パッケージを右クリックし、「新規」->「すべてのテンプレート」を選択します。

   新規ウィザードが表示されます。
4. Java Bean ノードを展開し、「`Bean`」を選択し、「次へ」をクリックします。
5. 「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。
6. Bean とそのクラスを展開して、内容を表示します。
7. 「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。
8. 「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。

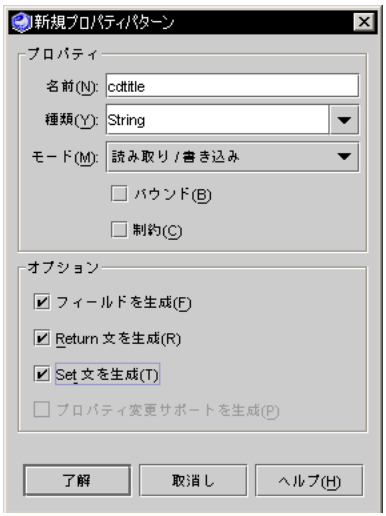
   a. 「名前」フィールドに **`cdtitle`** と入力します。

   b. 「種類」に `String` を選択します。

   c. 次のオプションを有効にします。

   - 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。



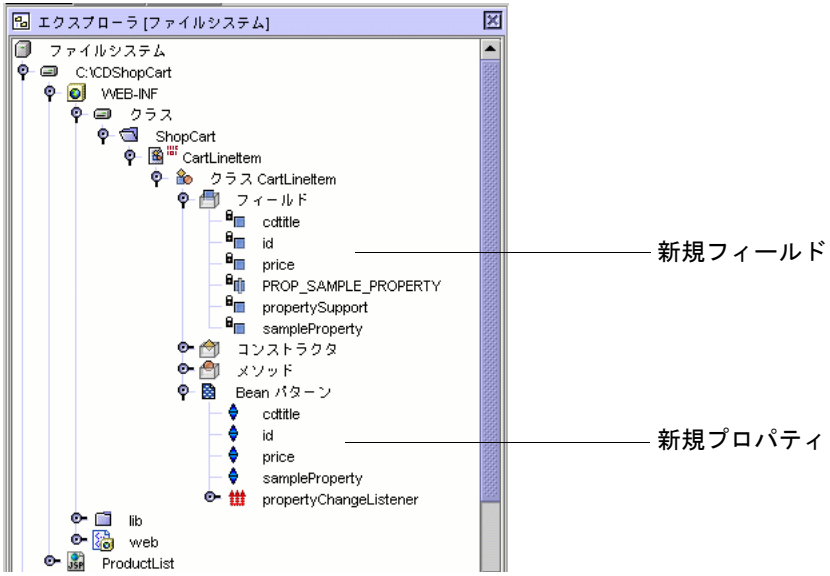

9. 「了解」をクリックして、ダイアログを閉じます。
10. 同様に、次の設定で id プロパティを作成します。

| プロパティ | 値 |
|---|---|
| 名前 | id |
| 種類 | int |
| 有効なオプション | フィールドを生成<br>Return 文を生成<br>Set 文を生成 |

11. また、次の設定で `price` を作成します。

| プロパティ | 値 |
|---|---|
| 名前 | price |
| 種類 | double |
| 有効なオプション | フィールドを生成<br>Return 文を生成<br>Set 文を生成 |

12. 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。




13. 新規フィールドの一つ (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。
14. 「メソッド」ノードを展開し、各フィールドの新しい get メソッドと set メソッドを表示します。

## 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 プロパティと 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多重定義して、適切なコードを追加します。

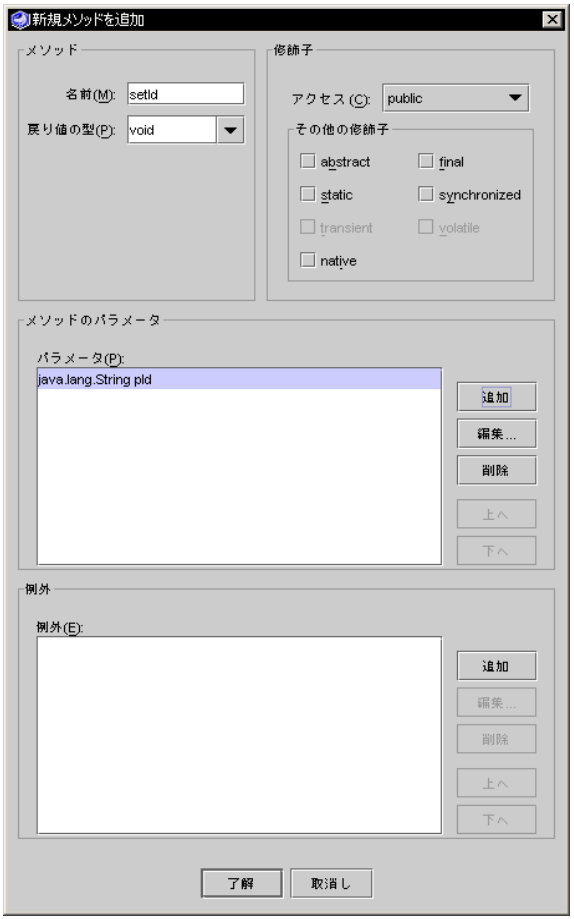
setId メソッドおよび setPrice メソッドを多重定義するには、次のようにします。

1. CartLineItem Bean の「メソッド」ノードを右クリックし、「メソッド...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. エクスプローラで新しいメソッドをダブルクリックして、そのコードをソースエディタに表示します。

8. 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);
}
```

**参考 –** 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。

同じ方法で `setPrice` メソッドを作成します。

9. setPrice メソッドを次のように定義します。

| プロパティ | 値 |
|---|---|
| 名前 | setPrice |
| 戻り値の型 | void |
| パラメータの型 | java.lang.String |
| パラメータ名 | pPrice |

10. 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);
}
```

11. クラス ( ) ではなく、CartLineItem Bean ( ) を選択し、「構築」->「コンパイル」を選択するか、F9 を押します。

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。

**参考 –** 72 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 のプロパティが、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 プロパティおよび price プロパティの set メソッドを多重定義したため、もはや Introspector はそれらのプロパティを Bean の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 の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。

id と price が CartLineItem Bean のプロパティであることを指示するには、次のようにします。

1. **ShopCart パッケージを右クリックし、「新規」->「すべてのテンプレート」を選択します。**

   新規ウィザードが表示されます。

2. **Java Bean ノードを展開し、「アイコンなし BeanInfo」を選択し、「次へ」をクリックします。**

3. **この BeanInfo に `CartLineItemBeanInfo` という名前を付け、「完了」をクリックします。**

   新しい CartLineItemBeanInfo ノードがエクスプローラに表示されます。

4. **BeanInfo のノードとそのメソッドノードを展開します。**

5. **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。

6. 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;
    }
}
```

---

**参考 –** ソースエディタにペーストまたは入力したコードは、Ctrl+Shift F を押して書式を整えます。

---

7. CartLineItemBeanInfo ノードを選択し、F9 を押して Bean をコンパイルします。

   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 に基づいて作成します。

Cart Bean を作成するには、次のようにします。

1. ShopCart パッケージを右クリックし、「新規」->「Bean」を選択します。

   この項目は、コンテキストメニューに表示されます。このショートカットは、「新規」メニューから項目を選択するときに役立ちます。

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て、「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、次の設定で新しい lineItems Bean パターンを作成します。

| プロパティ | 値 |
|---|---|
| 名前 | lineItems |
| 種類 | java.util.Vector |
| 選択オプション | フィールドを生成<br>Return 文を生成<br>Set 文を生成 |

種類 java.util.Vector は、ドロップリストにないので、キーボードから入力します。

lineItems 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。

5. 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。

6. lineItems のプロパティを表示し、「修飾子」の値フィールドをクリックします。

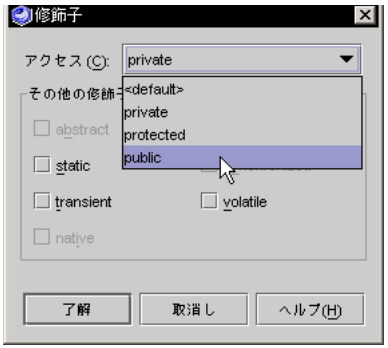
---

**注 –** 「エクスプローラ」ウィンドウの下に「プロパティ」ウィンドウが表示されている場合は、項目を選択するとそのプロパティがウィンドウに表示されます。lineItems を右クリックし、コンテキストメニューから「プロパティ」を選択して、プロパティを表示することもできます。

---

7. 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。

8. 「アクセス」リストから「public」を選択します。




9. 「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、選択された品目番号を返すメソッド、そしてカートから品目を削除する別のメソッドを追加します。

10. 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。

11. 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();
}
```

12. Cart の「メソッド」ノードを右クリックし、「メソッド...追加」を選択して、次のように findLineItem メソッドを定義します。

| プロパティ | 値 |
|---|---|
| 名前 | findLineItem |
| 戻り値の型 | int |
| パラメータの型 | int |
| パラメータ名 | pID |

13. 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;
}
```

14. 手順 12 を繰り返し、removeLineItem メソッドを次のように定義します。

| プロパティ | 値 |
|---|---|
| 名前 | removeLineItem |
| 戻り値の型 | void |
| パラメータの型 | int |
| パラメータ名 | pID |

15. 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()");
}
```

16. **クラス ( ) ではなく、Cart Bean ( ) を選択し、F9 を押して Cart Bean をコンパイルします。**

    Bean がコンパイルされます。次に、ShopCart JSP ページを作成します。

---

**参考 –** 79 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. **CDShopCart Web モジュールを右クリックし、「新規」->「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。

## 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 (CartLineItem はこのライブラリの 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ます。「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、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、Shopping Cart 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 Cart 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フェッチするコードを作成します。これは、41 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、表にデータを格納します。

```
<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. 60 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。

- ShopCart JSP ページに次のコードを追加します。

```
<p>
<!-- 3 つのボタンを作成 -->
<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>
<! ページの終了>
</body>
</html>
```

---

**参考** – 82 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。ここでは、CDShopCart Web アプリケーションの再配備が必要です。なぜなら、現在のアプリケーションには、以前に配備したバージョンには存在しないコンポーネントが含まれているからです。

---

**注 –** Sun ONE Application Server 7 が IDE のデフォルト Web サーバーになっていることを確認します。詳細は、9 ページの「Sun ONE Application Server 7 がデフォルトのサーバーであることの確認」を参照してください。

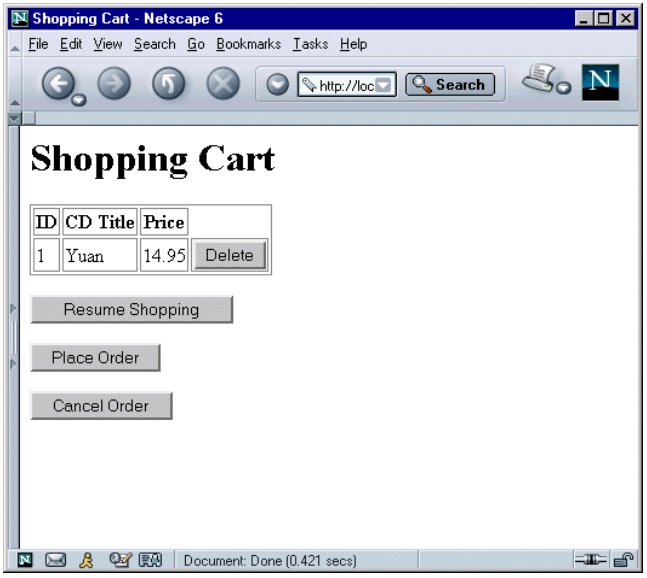
---

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 モジュールを右クリックし、「すべてを構築」を選択します。
3. CDShopCart Web モジュールを再度右クリックし、「配備」を選択します。

   IDE によってアプリケーションが再配備されます。配備処理の進捗を示す進捗ウィンドウが表示され、処理が完了すると自動的に閉じます。
4. 「ProductList JSP ページ」を右クリックし、「実行」を選択します。

   IDE によってデフォルトのブラウザが起動し、CD Catalog List ページが表示されます。
5. 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。




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

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。

## 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 (56 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、ProductList 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」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一例にすぎません。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 5 の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」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」に変更して、次のような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。

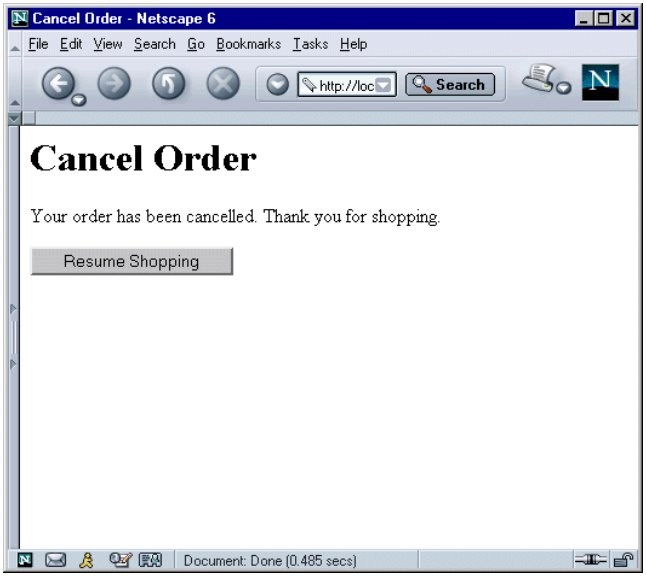



図 3-5　Cancel Order ページ

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」に変更して、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。ここでも、CDShopCart Web アプリケーションの再配備が必要です。現在のアプリケーションには、以前に配備したバージョンには存在しないコンポーネントが含まれているからです。

---

**注 –** Sun ONE Application Server 7 が IDE のデフォルト Web サーバーになっていることを確認します。詳細は、9 ページの「Sun ONE Application Server 7 がデフォルトのサーバーであることの確認」を参照してください。

---

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 モジュールを右クリックし、「すべてを構築」を選択します。
3. CDShopCart Web モジュールを再度右クリックし、「配備」を選択します。

   IDE によってアプリケーションが再配備されます。配備処理の進捗を示す進捗ウィンドウが表示され、処理が完了すると自動的に閉じます。

4. 「ProductList JSP ページ」を右クリックし、「実行」を選択します。

   IDE によってデフォルトのブラウザが起動し、CD Catalog List ページが表示されます。

5. 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。

6. 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。

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

8. カートに CD を追加してみます。

9. 「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。

10. 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。

11. 別の CD をカートに追加してみます。

12. Shopping Cart ページが表示されたら、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が終了したため (手順 9)、このカートには CD が 1 枚しかないはずです。

13. 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。

14. 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。

15. 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。

16. アプリケーションを終了するには、ブラウザで別の URL を指定します。

    これで、CDShopCart アプリケーションが完成しました。

# 付録 A

# CDShopCart のソースファイル

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



このコードは、次の Sun ONE Studio 5 Developer Resources ポータルからダウンロードできる CDShopCart アプリケーションの zip ファイルにもソースファイルとして含まれています。

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

---

**参考 –** ソースファイルの内容を Sun ONE Studio 5 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。

---

Solaris と Linux を使用している場合は、これらのファイルをコピーしないことをお勧めします。ソースエディタで、行末のキャリッジリターンが認識されません。ソースファイルを表示するには、CDShopCart のソース zip ファイルを解凍し、解凍したディレクトリを IDE にマウントします。

# 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" password="PBPUBLIC" />

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

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

<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

```
/*
 * CartLineItem.java
 *
 * 2003 年 4 月 11 日午後 2 時 24 分作成
 */

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 = value;
        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);
        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 //GEN-FIRST:BeanDescriptor
    private static BeanDescriptor beanDescriptor = null;
    private static BeanDescriptor getBdescriptor(){
        //GEN-HEADEREND:BeanDescriptor

        // 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;     } //GEN-LAST:BeanDescriptor


    // プロパティ情報はイントロスペクションから取得 //GEN-FIRST:Properties
    private static PropertyDescriptor[] properties = null;
    private static PropertyDescriptor[]
getPdescriptor(){//GEN-HEADEREND:Properties

        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;     } //GEN-LAST: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 //GEN-FIRST:Events
    private static EventSetDescriptor[] eventSets = null;
    private static EventSetDescriptor[] getEdescriptor(){//GEN-HEADEREND:Events

        // 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;     } //GEN-LAST:Events

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

        // 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;     } //GEN-LAST:Methods


    private static int defaultPropertyIndex = -1; //GEN-BEGIN:Idx
    private static int defaultEventIndex = -1; //GEN-END:Idx


    //GEN-FIRST:Superclass

    // 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

    //GEN-LAST:Superclass

    /**
     * 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
     * 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
     * PropertyDescriptor の配列。自動解析で情報を
     * 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
     * EventSetDescriptor の配列。自動解析で情報を
     * 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
     * MethodDescriptor 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
     * 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、その利用者がもっと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

```
/*
 * Cart.java
 *
 * 2003 年 4 月 11 日午後 2 時 36 分作成
 */

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;
}

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;
    }

    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);
}
// 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
<%
}
%>
```

```
<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.id}"/>">
<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>
<!-- 3 つのボタンを作成 -->
<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>
<! ページの終了>
</body>
</html>
```

# EmptyCart.jsp のソース

```
<%@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>
```

# PlaceOrder.jsp のソース

```
<%@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>
```

## 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 チュートリアル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 5 Developer Resources ポータルからダウンロードできる CDShopCart アプリケーションの zip ファイルにもスクリプトファイルとして含まれています。

`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)
```

# 付録C

# Oracle データベースを使用したチュートリアルの作成

この付録では、Oracle データベースで CDShopCart チュートリアルを作成し、実行する手順を示します。この付録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。アプリケーションファイルとは、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、および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は、Sun ONE Studio 5 Developer Resources ポータル (`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`) から、ダウンロードすることができます。

## IDE の Oracle データベースへの接続

Sun ONE Studio 5 ソフトウェアを Oracle データベースに接続するように設定するには、次の操作が必要です。

- Oracle JDBC ドライバを有効にする
- IDE を Oracle サーバーに接続する

# Oracle Type 4 JDBC ドライバの有効化

JDBC ドライバを有効にするには、ドライバライブラリを Sun ONE Studio 5 と Sun ONE Application Server 7 のクラスパスに追加します。クラスパスに追加するには、Oracle Type 4 JDBC ドライバライブラリ (`classes12.zip` ファイル) が必要です。このドライバは、Oracle のポータルからダウンロードできます。

Oracle Type 4 JDBC ドライバを有効にするには、次のようにします。

1. **Oracle Type 4 ドライバライブラリを *s1studio-install-directory*`/lib/ext` ディレクトリにコピーします。**

   たとえば、`classes12.zip` ファイルを `c:\Sun\studio5_se\lib\ext` にコピーします。Solaris と Linux では、コピー先ディレクトリは異なります。

   ---

   **注 –** Sun ONE Studio 5 のホームディレクトリに書き込むには、root または administrator の権限が必要です。

   ---

2. **IDE を再起動します。**
3. **エクスプローラの「実行時」タブで、アプリケーションサーバーのインスタンスを選択します。**

   ラベルは *app-server-name* (*app-server-host*:*app-server-port*) という形式になっています。たとえば、デフォルトのサーバーは「server1 (localhost:4848)」です。また、標準ユーザーのサーバーは、「MyServer (localhost:4855)」のようになります。

4. **アプリケーションサーバーのインスタンスのプロパティを表示します。**

   プロパティのウィンドウは、通常はエクスプローラの下に表示されます。ノードを選択すると、ウィンドウにプロパティが表示されます。サーバーインスタンスのノードを右クリックし、「プロパティ」を選択して、プロパティを表示することもできます。

5. **「クラスパス接頭辞」プロパティのプロパティエディタを開きます。**

   このプロパティの値フィールドをクリックし、省略符号ボタン (...) をクリックします。クラスパス接頭辞のエディタウィンドウが表示されます。

6. **「JAR/ZIP を追加」ボタンをクリックします。**

   「JAR ファイルを追加」で `classes12.zip` ファイルを検索します。

7. **`classes12.zip` ファイルを選択して「了解」をクリックします。**
8. **「了解」をクリックしてプロパティエディタウィンドウを閉じます。**

# IDE の Oracle サーバーへの接続

チュートリアルのさまざまな操作を行うには、IDE を Oracle データベースに接続する必要があります。接続は、コンポーネントの作成前または作成中に行うことができます。コンポーネントを作成する前にデータベースに接続するには、次のようにします。

1. Oracle サーバーが実行中であることを確認します。
2. エクスプローラの「実行時」タブで、「Databases」ノードとその「Drivers」サブノードを展開します。

   「Oracle thin」というノードが表示されます。

   このノードに赤い取り消し線がある場合は、Oracle JDBC ドライバが有効になっていません。92 ページの「Oracle Type 4 JDBC ドライバの有効化」の手順に従って、有効にしてください。
3. このノードを右クリックし、「接続」を選択します。

   「データベースの新規接続」ダイアログが表示されます。
4. 「名前」フィールドで「Oracle thin」が選択されていることを確認します。
5. 「データベース URL」、「ユーザー名」、および「パスワード」の各プロパティの値を入力します。

   たとえば、ローカルにインストールされた Oracle データベースの SID が「extut」、デフォルトの Oracle ログインのユーザー名が「scott」、パスワードが「tiger」の場合は、次の値を使用します (1521 は、Oracle の標準ポート番号です)。

| 名前 | 値 |
|---|---|
| データベース URL | jdbc:oracle:thin:@localhost:1521:extut |
| ユーザー名 | scott |
| パスワード | tiger |

6. 「セッション中はパスワードを保存」オプションを有効にします。

   「データベースの新規接続」ダイアログは次のようになります。


7. 「了解」をクリックします。
8. 「ドライバ」ノードを閉じます。

   labeled jdbc:oracle:thin:@*hostname*:1521:*sid* [*Username* on *Password*] という、新しい Oracle thin ドライバノードが表示されます。

# データベース表の作成

CDShopCart チュートリアルでは、データベース表を 1 つ使用します。この表を Oracle Server データベースに作成する必要があります。この節では、用意されている SQL スクリプトを使用して表を作成する方法を示します。Microsoft Windows では、付録 B にある SQL スクリプトをコピー & ペーストして、表を作成できます。Solaris と Linux では、CDShopCart のアプリケーションファイルに含まれる `CDCatalog_ora.sql` というスクリプトファイルを使用できます。

Microsoft Windows システムで Oracle データベースにチュートリアルの表をインストールするには、次のようにします。

1. 「スタート」->「プログラム」->「Oracle (バージョン)」->「Application Development」->「SQL Plus」を選択して、Oracle コンソールを開きます。
2. ユーザー名とパスワードを使用して SQL Plus にログインします。
3. SQL プロンプトが表示されたら、付録 B のスクリプトをコピーし、プロンプトの横にペーストします。

---

**参考** – 最初の DROP 文が、まだ作成されていない表を参照するので、スクリプトの実行時にエラーが発生しますが、このエラーは無視できます。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。

---

Solaris と Linux の各環境でチュートリアルの表をインストールするには、次のようにします。

1. Developer Resources ポータルからダウンロードした `CDShopCart.zip` ファイルを解凍します。

   たとえば、`/MyZipFiles` ディレクトリに解凍します。

2. コマンドプロンプトで、次のように入力します。

```
$ cd your-unzip-dir/CDShopCart/SQLscripts
$ sqlplus db-userid/db-password@db-servicename @CDCatalog_ora.sql
```

   たとえば、`/MyZipFiles` ディレクトリの場合は次のようになります。

```
$ cd /MyZipFiles/CDShopCart/SQLscripts
$ sqlplus scott/tiger@MyDB @CDCatalog_ora.sql
```

   DROP 文のエラーが発生しますが、このエラーは無視できます。

## IDE でのデータベース表の表示

この手順を実行する前に、93 ページの「IDE の Oracle サーバーへの接続」に従って IDE をデータベースに接続します。

IDE に接続したデータベースに CD 表が作成されていることを確認するには、次のようにします。

1. エクスプローラの「実行時」タブで、「Databases」ノード、Oracle の接続ノード、およびその「表」ノードを展開します。

   CD 表が表示されない場合は、「表」ノードを右クリックし、コンテキストメニューから「再表示」を選択します。

2. CD 表を右クリックし、コンテキストメニューから「データを表示」を選択します。

   コマンドエディタに表のデータが表示されます。


# Oracle データベースを使用した CDShopCart アプリケーションの作成

Oracle データベースを使用してアプリケーションを作成するには、第 3 章の手順を 1 箇所だけ変更します。

40 ページの「setDataSource タグを使用したデータベースへの接続」の手順 2 で、PointBase ではなく Oracle 用の setDataSource 文を使用します。各変数に、接続に対応する値を指定します。たとえば、ローカルシステムにある Oracle データベースの名前が「MyOracleDB」、ユーザー IDが「scott」、パスワードが「tiger」の場合は、次のように指定します。

```
<sql:setDataSource var="productDS"
url="jdbc:oracle:thin:@localhost:1521:MyOracleDB"
driver="oracle.jdbc.driver.OracleDriver"
user="scott" password="tiger" />
```

これ以外の手順は、PointBase の場合と同じです。

# 索引

## Q



## R



## S



## T



## V



## W



## あ