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

Forte for Java, Enterprise Edition, 3.0

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

Part No. 816-2848-01
2001 年 10 月 , Revision A

原典： *Forte for Java, Enterprise Edition Tutorial*
Part No: 816-1409-10
Revision A

Please
Recycle

目次

図目次

表目次

はじめに

『Forte™ for Java™, Enterprise Edition チュートリアル』へようこそ。このチュートリアルでは、Forte for Java, Enterprise Edition に導入された次の機能の使用方法について学習します。

- EJB™ 1.1 ビルダー — チュートリアルアプリケーションの Enterprise JavaBeans™ コンポーネントを作成、開発、作成する
- EJB モジュールアセンブリ — EJB コンポーネントを EJB モジュールにアセンブルし、EJB Java Archive (JAR) ファイルにエクスポートする
- テストアプリケーション機能 — J2EE リファレンス実装 (Reference Implementation: RI) をアプリケーションサーバーとして使用し、クライアントを手動で作成せずに Enterprise Beans をテストする
- Web サービスモジュール — 既存の EJB コンポーネントから Web サービスを構築し、Web ブラウザで表示可能な JSP™ ページを生成する
- J2EE リファレンス実装モジュール — チュートリアルアプリケーションを配備し、テストする

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

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

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

```
c:¥>cd MyWorkDir¥MyPackage
```

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

```
% cd MyWorkDir/MyPackage
```

お読みになる前に

このチュートリアルでは、Java 2 Platform, Enterprise Edition 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。Forte for Java Enterprise Edition の機能を使用した、J2EE 準拠のアプリケーションの作成、開発、配備に関する作業を理解するためにこのチュートリアルが役に立ちます。チュートリアルを開始する前に、次の内容をよく理解しておく必要があります。

- Java プログラミング言語
- Enterprise JavaBeans の概念
- Java サーブレット構文
- JDBC™ 対応のドライバ構文
- JavaServer Pages 構文
- HTML 構文
- リレーショナルデータベースの概念 (表やキーについて)
- 特定のデータベースソフトウェアを使用して表を作成する方法
- J2EE アプリケーションのアセンブリの概念
- J2EE アプリケーションの配備の概念

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

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

本書の構成

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

第 1 章では、チュートリアルに必要なソフトウェア要件、Java 2 SDK, Enterprise Edition ソフトウェアの起動方法、Forte for Java 統合開発環境 (IDE) の起動方法、J2EE リファレンス実装を実行するための IDE の設定方法、チュートリアルデータベース表の作成方法について説明します。また、インストールされる Forte for Java ディレクトリの一覧と説明も記載されています。

第 2 章では、チュートリアルアプリケーションの機能とアーキテクチャについて説明します。

第 3 章では、チュートリアルアプリケーションの EJB 層の作成方法を順を追って説明します。ここでは、Forte for Java のテストアプリケーション機能を使用し、各 Bean の配備をテストする方法についても説明します。

第 4 章では、Web サービスモジュールを使用し、アプリケーションのビジネスメソッドに基づいた XML オペレーションを作成し、自動的に Web サービスを生成し、各アプリケーションをクライアントでテストする方法について説明します。

第 5 章では、第 4 章 の Web サービスで生成された出力に基づき、クライアントページがどのように構築されているかについて説明します。

表記上の規則

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

表 P-1　表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** Password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

コード例は次のように表示されます。

- C シェルプロンプト

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

■ Bourne シェルおよび Korn シェルのプロンプト

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

■ スーパーユーザーのプロンプト

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

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

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

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

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

関連マニュアル

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

オンラインで入手可能なマニュアル

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

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

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

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

- リリースノート (HTML 形式)

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

- インストールガイド (PDF 形式)

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

- Forte for Java プログラミングシリーズ (PDF 形式)

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

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

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

  - 『持続プログラミング』 Part No. 816-2850-01

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

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

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

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

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

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

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

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

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

- Forte for Java チュートリアル (PDF 形式)

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

  - 『Forte for Java, Community Edition チュートリアル』 Part No. 816-2847-01

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

  - 『Forte for Java, Enterprise Edition チュートリアル』 Part No. 816-2848-01 (このマニュアル)

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

オンラインヘルプ

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

プログラム例

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

Javadoc

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

Sun のマニュアルのオンラインでの提供

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

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

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

`http://docs.sun.com`

Sun のマニュアルの注文方法

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

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

ご意見の送付先

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

`docfeedback@sun.com`

電子メールのタイトルに、対象マニュアルの Part No. (このマニュアルの場合は 816-2848-01) を明記してください。

第1章

チュートリアルを開始する前に

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

チュートリアルアプリケーションのソフトウェア要件

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

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

Forte for Java IDE の実行に必要なコンポーネント

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

チュートリアルアプリケーションの作成と実行に必要なコンポーネント

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

- 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 ブラウザとしては、Netscape Communicator™ または Microsoft Internet Explorer を使用できます。Netscape Communicator, version 4.7*x* は、Forte for Java のポータルからダウンロードできます (製品 CD-ROM にも含まれています)。

- Web サーバー — Tomcat, version 3.2

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

- データベースソフトウェア — PointBase Network Server, version 3.5

  Forte for Java, Community Edition のインストール時に PointBase をインストールできます。PointBase がインストールされているかどうかを調べるには、Forte for Java がインストールされているディレクトリ下に `pointbase` ディレクトリが存在するかどうかを確認します。PointBase がインストールされていない場合は、インストーラを再度起動してインストールを行うことができます。

- J2EE リファレンス実装 (Reference Implementation: RI) version 1.2.2

チュートリアルの J2EE アプリケーションを配備するには、アプリケーションサーバーが必要です。ここでは、J2EE リファレンス実装の特別なバージョンを使用する必要があります。これは、Java 2 SDK, Enterprise Edition, version 1.2.2 に含まれているもので、Forte for Java の ポータルからダウンロードすることができます (製品 CD-ROM にも含まれています)。

IDE に対して、使用しているアプリケーションサーバーを通知する必要があります。この方法については、12 ページの「リファレンス実装を実行するように IDE を設定する」 に記載されています。

チュートリアルデータベース表の作成

Forte for Java, Enterprise Edition のチュートリアルでは、2 つのデータベース表が必要です。これらの表を作成するには、`rest_pb.sql` という SQL スクリプトを使用します。このスクリプトは、ユーザー設定ディレクトリの `sampledir/tutorials/DiningGuide/SDb` ディレクトリにあります。

注 - `rest_pb.sql` スクリプトは、実際には 2 つのバージョンの EatersDigest 表を作成します。1 つは標準のデータベースで、これを使用して EJB コンポーネントと詳細クラスを作成します。もう 1 つは変更バージョンで、これは J2EE リファレンス実装プロトコルで文字列「EJBTable」を表に追加し、それぞれの表名と列名を引用符で囲む処理をサポートするものです。表の変更バージョンは、EJB 層をテストする際にユーザーのデータベースに定義されているため、アプリケーションを配備するたびに RI がデータベース表を作成および削除しないようにプロパティを設定します。他のアプリケーションサーバーを使用する場合は、表を引用符で囲む必要はありません。

---

rest_pb.sql スクリプトで作成するデータベーススキーマを、表 1-1 に示します。

表 1-1　EatersDigest データベース表

表名	列	主キー	その他
Restaurant	restaurantName	○	
	cuisine		
	neighborhood		
	address		
	phone		
	description		
	rating		
CustomerReview	restaurantName	○	複合主キー (CustomerName が Restaurant 表の restaurantName を参照している)
	customerName	○	
	review		

Restaurant (および "RestaurantEJBTable") 表には、表 1-2 に示すレコードが含まれています。

表 1-2　Restaurant 表のレコード

restaurant-Name	cuisine	neighborhood	address	phone	description	rating
French Lemon	Mediterranean	Rockridge	1200 College Avenue	510 888 8888	Very nice spot.	5
Bay Fox	Mediterranean	Piedmont	1200 Piedmont Avenue	510 888 8888	Excellent.	5

CustomerReview 表には、表 1-3 のレコードが含まれています。

表 1-3　CustomerReview 表のレコード

restaurantName	customerName	comment
French Lemon	Fred	Nice flowers.
French Lemon	Fred	Excellent Service

"CustomerreviewEJBTable" 表には、表 1-4 のレコードが含まれています。

表 1-4　"CustomerreviewEJBTable" 表のレコード

restaurantName	customerName	comment
French Lemon	Ralph	Nice flowers.
French Lemon	Fred	Excellent Service

次の指示に従って、これらの表を PointBase データベースにインストールします。

1. PointBase Network Server を起動します。

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

2. PointBase コンソールを起動します。

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

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

注 - このリリースで提供される Java 2 SDK, Enterprise Edition, version 1.2.2 (リファレンス実装を含む) は、PointBase データベースが、このチュートリアルで使用する `sample` というデータベースを実行できるようにすでに構成されています。他の PointBase データベースを作成する場合は、*J2EE_HOME*`/config/default.properties` ファイルの `jdbc.resources` プロパティに、そのための情報を追加する必要があります。詳細は、『Forte for Java, Enterprise Edition インストールガイド』を参照してください。

---

3. 「OK」をクリックします。

   「PointBase Console」ウィンドウが表示されます。

4. 「File」>「Open」を選択し、ファイル検索用のダイアログを表示します。

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

   作業ディレクトリの下の `sampledir/tutorial/DiningGuide/Db` を確認します。`rest_pb.sql` ファイルの内容が、SQL エントリウィンドウにコピーされます。

   Solaris または Unix システムでは、作業ディレクトリは *forte4j-home*`/ffjuser30` となります。

6. 「SQL」>「Execute All」を選択します。

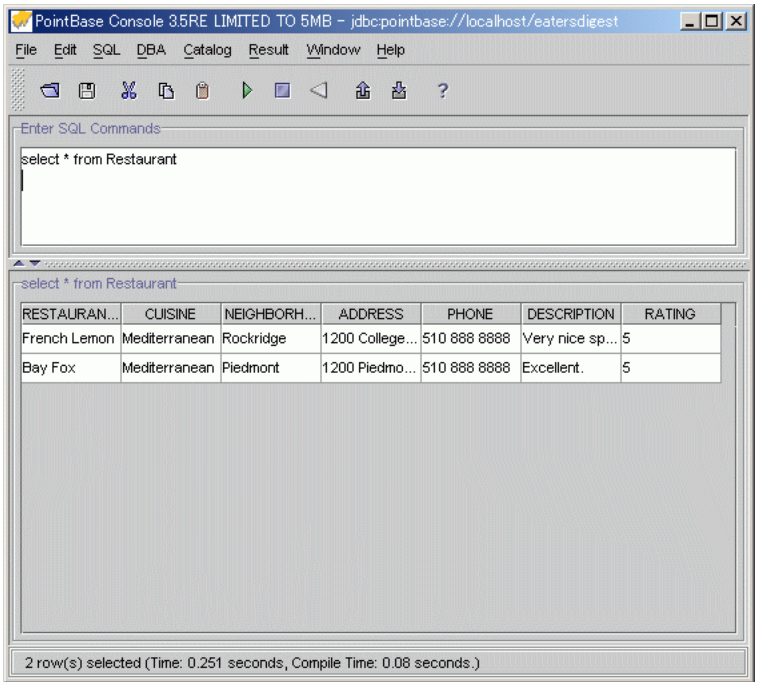
   メッセージが表示され、スクリプトが実行されたことが示されます。このとき、「`Cannot find the table...`」で始まる初期メッセージが表示されますが、無視してください。これらのメッセージが表示されるのは、まだ作成されていない各表に対する DROP 文によるものです。これらの DROP 文は、今後、スクリプトを再実行して表を初期化する場合に役立ちます。

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

```
select * from Restaurant
```

8. 「SQL」>「Execute」を選択します。

   コンソールには、次のような Restaurant 表が表示されます。



9. 「PointBase Console」ウィンドウを閉じます。

   これで、J2EE を起動する準備が完了しました。

Java 2 SDK, Enterprise Edition の起動

Forte for Java IDE を起動する前に、J2EE を起動しておく必要があります。

1. J2EE がインストールされていることを確認します。

   J2EE のバージョン 1.2.2 は、Forte for Java のダウンロードページから無料でダウンロードすることができます。これには、J2EE リファレンス実装も含まれています。

2. `J2EE_HOME` 環境変数を J2EE のルートディレクトリに設定します。

   たとえば、`J2EE_HOME` を `c:¥j2sdkee1.2.2` などに設定します。

3. JAVA_HOME 環境変数を JDK のルートディレクトリに設定します。

   たとえば、JAVA_HOME を c:¥jdk などに設定します。

4. コマンドウィンドウを開きます。

   Microsoft Windows では、「スタート」>「コマンドプロンプト」を選択します。

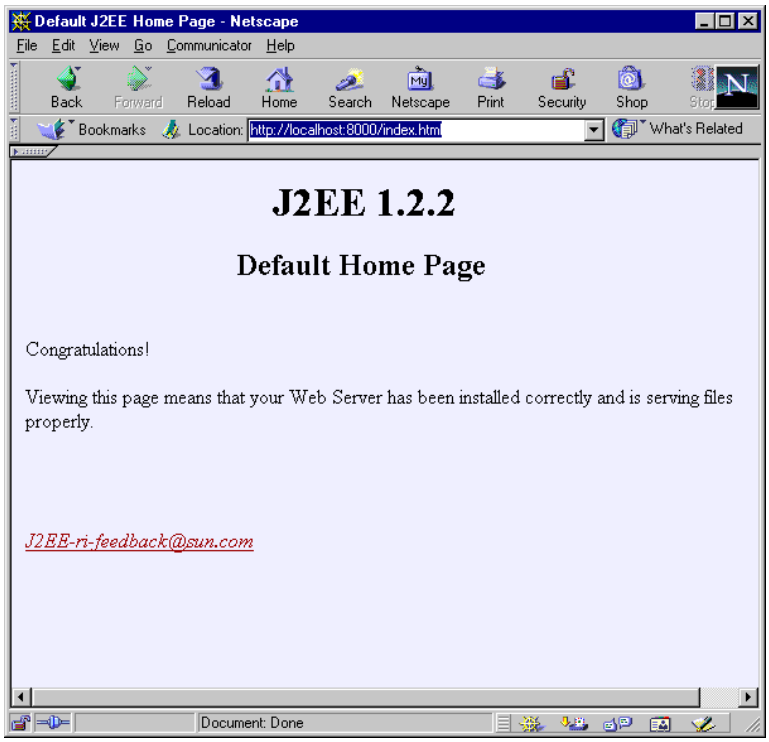
   Solaris、Linux、その他の UNIX ベースのシステムでは、端末ウィンドウを開きます。

5. J2EE_HOME/bin ディレクトリに移動し、j2ee -verbose と入力します。

   進行状況を表すメッセージが表示され、J2EE が起動していることが示されます。最後に、「J2EE server startup complete」というメッセージが表示されます。

6. Web ブラウザを起動して http://localhost:8000 の URL に移動し、J2EE が起動していることを確認します。

   J2EE のホームページは次のように表示されます。



これで、Forte for Java IDE を起動する準備が整いました。後で、アプリケーションをアセンブルして配備する準備ができたとき、または単純にテストするときに、このアプリケーションサーバーを使用していることを IDE に通知する必要があります。詳細は、12 ページの「リファレンス実装を実行するように IDE を設定 する」 を参照してください。

注 - Forte for Java ダウンロードサイトから入手した J2EE 1.2.2 の場合は、`FORTE4J_HOME` 環境変数を設定する必要があります。

Forte for Java IDE の起動

次の節で説明している手順で、Forte for Java IDE を起動します。詳細は『Forte for Java, Enterprise Edition インストールガイド』に記載されています。

注 - Forte for Java IDE を起動する前に、J2EE リファレンス実装をインストールしておく必要があります。

Solaris、UNIX、Linux 環境の場合

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

```
$ sh runide.sh
```

このスクリプトをカスタマイズする方法は、10 ページの「コマンド行オプションを使用したセッションの変更」を参照してください。

Microsoft Windows 環境の場合

インストールが終了すると、「Forte for Java EE」アイコンがデスクトップに表示されます。IDE を起動するには、このアイコンをダブルクリックします。IDE を実行するモードに応じて 2 つのアイコンが用意されています。この 2 つのモードは、次に示す実行可能ファイルとして 、*forte4j-home*¥`bin` ディレクトリに格納されています。

- `runidew.exe` は、コンソールウィンドウを使用しないで IDE を起動します。「スタート」メニューから IDE を起動する場合に、このファイルを使用します。
- `runide.exe` は、コンソールウィンドウを使用して IDE を起動します。コンソールウィンドウには、IDE の標準エラーと標準出力が示されます。コンソール上で Ctrl+Break キーを押すと、実行スレッドのダンプを出力することができます。Ctrl+C キーを押すと、プログラムがすぐに終了します。

これ以外の方法として、「スタート」>「プログラム」>「Forte for Java EE」>「Forte for Java EE」を選択して IDE を起動することもできます。また、次のようにコマンド行から実行可能ファイルのいずれかを実行することもできます。

```
C:¥> runidew.exe [switch]
```

コマンド行オプションを使用したセッションの変更

表 1-5 にIDE の起動方法を変更するためのオプションについて説明します。詳細は『Forte for Java, Enterprise Edition インストールガイド』を参照してください。

- Microsoft Windows システムの場合

コマンド行で IDE を実行する際にオプションを設定できます。

- Solaris、Linux、その他の UNIX マシンの場合

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

表 1-5　`runide` コマンド行オプション

オプション	意味
`-classic`	クラシック JVM を使用する
`-cp:p` *addl-classpath*	指定したクラスパスをクラスパスの先頭に追加する
`-cp:a` *addl-classpath*	指定したクラスパスを Forte for Java クラスパスの末尾に追加する

表 1-5　runide コマンド行オプション (続き)

オプション	意味
-fontsize *size*	IDE のグラフィカルユーザーインタフェースで使用するフォントのサイズを、指定したサイズに設定する
-J*jvm-flags*	指定したフラグを JVM に直接渡す (-J と引数の間にはスペースを入れない)
-jdkhome *jdk-home-dir*	デフォルトの SDK の代わりに、指定した Java 2 SDK を使用する
-h または -help	コマンド行オプションを表示する GUI ダイアログを開く
-hotspot、-client、-server、-classic、-native、 -green	指定したタイプの JVM を使用する
-single	シングルユーザーモードで IDE を実行する。ユーザー設定ディレクトリの代わりに、*forte4j-home* から IDE を起動できるようにする
-ui *UI-class-name*	IDE の見た目と使い心地として所定のクラスを使用して IDE を実行する
-userdir *user-directory*	現在のセッションに指定したユーザー設定ディレクトリを使用する。詳細については次の節を参照

ユーザー設定ディレクトリの指定

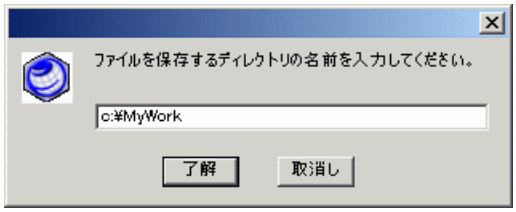
デフォルトでは、Forte for Java ソフトウェアをマルチユーザーモードで実行し、個別のプロジェクト、サンプル、IDE 設定をユーザーが指定した任意のディレクトリに格納します。これにより、個々の開発者がそれぞれの開発作業を同期させながら、個人の作業と設定を維持できます。

- Solaris、UNIX、Linux 環境の場合

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

- Microsoft Windows システムの場合

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



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

リファレンス実装を実行するように IDE を設定する

リファレンス実装 (Reference Implementation:RI) を 使用してアプリケーションをテストするには、Forte for Java IDE のサーバーレジストリに「参照」を作成する必要があります。

1. Forte for Java 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」レジストリノード、およびその下位の「インストールされているサーバー」サブノードを展開します。

   「J2EE Reference Implementation」ノードが表示されます。
3. 「J2EE Reference Implementation」ノードのプロパティを表示します。

   「プロパティ」ウィンドウが表示されている場合は、「J2EE RI」ノードを選択します。「プロパティ」ウィンドウが表示されていない場合は、「J2EE RI」ノードを右クリックし、コンテキストメニューで「プロパティ」を選択します。
4. 「プロパティ」ウィンドウで、「RIHome プロパティ」の値をユーザーの *J2EE_HOME* ディレクトリに変更します。

   たとえば、c:¥j2sdkee1.2.2 などを指定します。

5. 「J2EE Reference Implementation」ノードを右クリックし、「サーバーインスタンスの追加」を選択します。

   次の図に示すように、手順 5 で開始したサーバーインスタンスを表すノードが追加されます。

   これで、J2EE リファレンス実装は、配備されたアプリケーションを使用できるようになります。

Forte for Java のディレクトリ構造

Forte for Java ソフトウェアをインストールすると、インストールディレクトリに、表 1-6 に示すサブディレクトリが作成されます。

表 1-6　Forte for Java のディレクトリ構造

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

Forte for Java ソフトウェアをデフォルト (マルチユーザー) モードで起動する場合は、表 1-7 に示すサブディレクトリがユーザー設定ディレクトリにインストールされます。これらのほとんどが Forte for Java のホームディレクトリのサブディレクトリに対応しており、設定を保持するために使用されます。

表 1-7　ユーザー設定ディレクトリのディレクトリ構造

ディレクトリ	目的
`beans`	IDE にインストールされた JavaBeans コンポーネントのユーザー設定が格納される。
`javadoc`	IDE にインストールされた Javadoc ファイルのユーザー設定が格納される。
`lib`	システム `lib` ファイルのユーザー設定が格納される。
`modules`	アップデートセンターからダウンロードされたモジュールが格納される。
`sampledir`	エクスプローラの「ファイルシステム」区画にデフォルトでマウントされるディレクトリ。IDE 内で作成したオブジェクトが保存される。ただし、別のディレクトリをマウントして代わりに使用する場合は、そのディレクトリに保存される。
`sampledir/examples`	サンプルアプリケーションが格納される。Enterprise Edition 固有のサンプルを Forte for Java ポータルからダウンロードする場合は、サンプルを `~/sampledir/examples` に保存するように指示されます。
`sampledir/tutorial`	このドキュメントに記載されている DiningGuide チュートリアルと、そのデータベーススクリプトなどのチュートリアルアプリケーションが格納される
`system`	システムファイルとディレクトリのユーザー設定が格納される (技術サポートが必要な場合に役立つ `ide.log` など)。

第2章

チュートリアルアプリケーションの概要

ここでは、チュートリアル用のサンプルアプリケーションを作成することで、Forte for Java, Enterprise Edition の機能を使用して簡単な J2EE アプリケーションを作成する方法を学習します。

この章では、まず作成するアプリケーションについて説明し、アプリケーションの要件、およびその要件を満たすアーキテクチャについて説明します。最後の節では、Forte for Java, Enterprise Edition の機能 (EJB ビルダー、テストアプリケーション機能、Web サービスモジュールなど) を使用する方法について説明します。

この章の構成は次のとおりです。

チュートリアルアプリケーションの機能

チュートリアルアプリケーション EatersDigest は、レストランのガイドを題材とした簡単なアプリケーションで、利用できるレストランの一覧とその特徴 (名前、料理の種類、地域、住所、電話番号、簡単な説明、1 ～ 5 のランク) を表示し、選択したレストランの評価を参照したり、自分の評価を追加したりできます。ここでは、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やりとりします。具体的には、次のことを行います。

1. レストランの一覧を表示する。
2. 詳細を知りたいレストランについて、そのレストランに関する顧客の評価一覧を参照する。
3. レストランに対するユーザー自身の評価を評価一覧に追加する。

アプリケーションのシナリオ

EatersDigest でのやりとりは、顧客が Web ブラウザで (レストラン一覧が表示されている) アプリケーションの URL ページを訪れたときから始まり、アプリケーションの Web サイトを終了した時点で終了します。次のシナリオでは、EatersDigest アプリケーションと顧客のやりとりについて示します。このシナリオによって、アプリケーションで行われるやりとりと、アプリケーションで必要となる機能を理解することができます。

1. ブラウザで、アプリケーションの Restaurant Listing ページの URL を指定します。

   EatersDigest の Restaurant Listing ページが表示され、すべてのレストランの一覧、レストランの名前、料理の種類、場所、電話番号、簡単な評価、ランク (1 ～ 5) が示されます。各レストランのレコードには、2 つのボタン (「View Customer Comments」と「Add Your Comment」) が表示されます。

2. 特定のレストランの「View Customer Comments」ボタンをクリックします。

   All Customer Reviews By Restaurant Name ページに、そのレストランについて投稿されているすべての評価が表示されます。このページには、「Return to Restaurant List」ボタンがあります。

3. Customer Review ページの「Return to Restaurant List」ボタンをクリックします。

   Restaurant Listing ページが再表示されます。

4. Restaurant Listing ページの「Add Your Comment」ボタンをクリックします。

   Enter Customer Review ページに、3 つのフィールドと「Submit」ボタンが表示されます。表示されるフィールドは、「Restaurant Name」 (選択されたレストラン) と、2 つのテキスト入力フィールド (「Customer Name」と「Review」) です。

5. Enter Customer Review ページの「Customer Name」と「Review」フィールドに入力して「Submit」ボタンをクリックします。

   CustomerReview データベース表に顧客名と評価が追加され、Restaurant Listing ページが再表示されます。

6. 以前に選択したレストランで「View Customer Comments」ボタンをクリックし、自身の評価を表示します。

   All Customer Reviews By Restaurant Name ページが表示され、追加された新しい評価を含むすべての評価が示されます。

アプリケーション機能の仕様

次に、アプリケーションのユーザーインタフェースの主な機能を示します。アプリケーションのシナリオは、これらの機能によってサポートされています。

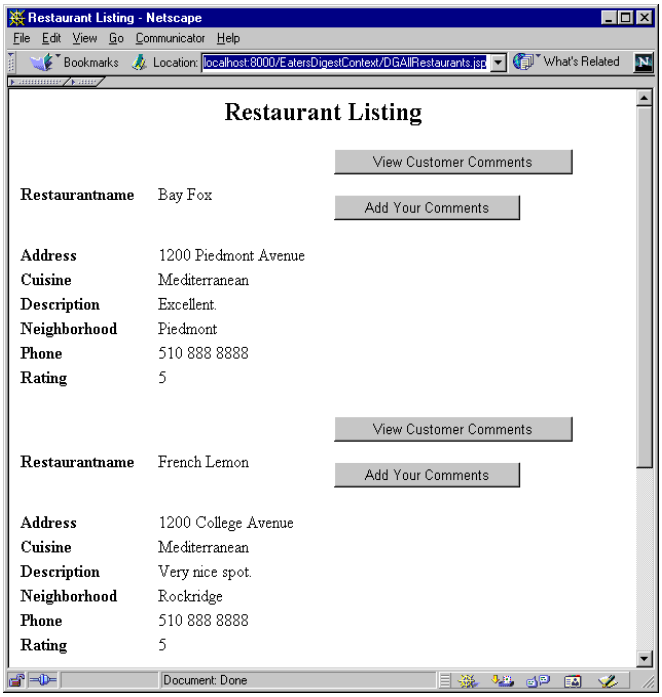
- 表示されている一覧内の全レストランデータのマスタービュー
- マスターレストラン一覧ページ上の、特定のレストランに関する評価をすべて取得するためのボタン
- 特定のレストランに関するすべての評価のマスタービュー
- マスターレストラン一覧ページ上の、新しい評価を追加するためのボタン
- レストランごとの評価ページ上の、レストラン一覧ページに戻るためのボタン
- 表示されているレストラン名に対し、新しい顧客名と顧客の評価を入力するためのテキスト入力ページ
- テキスト入力ページ上の、入力した評価データをデータベースに追加するためのボタン

ユーザーから見たチュートリアルアプリケーション

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

1. Web ブラウザで、アプリケーションの URL を指定します。

   EatersDigest の Restaurant Listing JSP ページが表示され、データベース内のすべてのレストランが示されます。



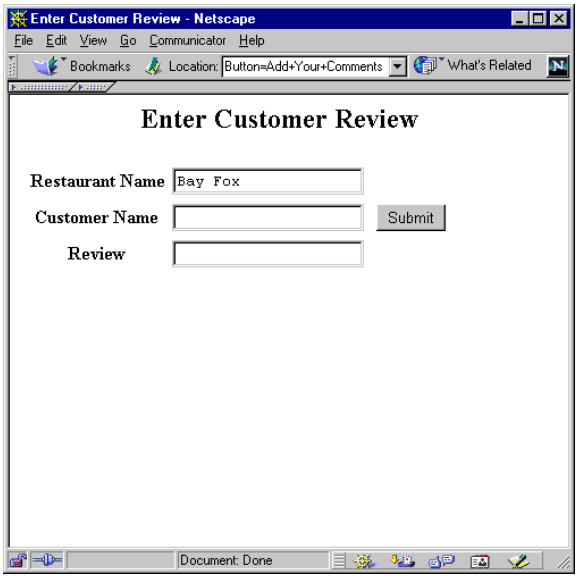
   このページには、3 ページの「チュートリアルデータベース表の作成」 で作成した Restaurant 表のデータが表示されます。

2. 特定のレストランに対する顧客の評価を表示するには、対象のレストランの「View Customer Comments」ボタンをクリックします。

   All Customer Reviews By Restaurant Name ページが表示され、選択したレストランについての評価がすべて示されます。

3. 「View Restaurant List」ボタンをクリックし、Restaurant Listing ページに戻ります。
4. レストランに対して評価を追加するには、「Add Your Comments」ボタンをクリックします。

   Enter Customer Review ページが表示され、選択したレストラン名が示されます。このレストランに対し、自分の名前と評価を入力することができます。

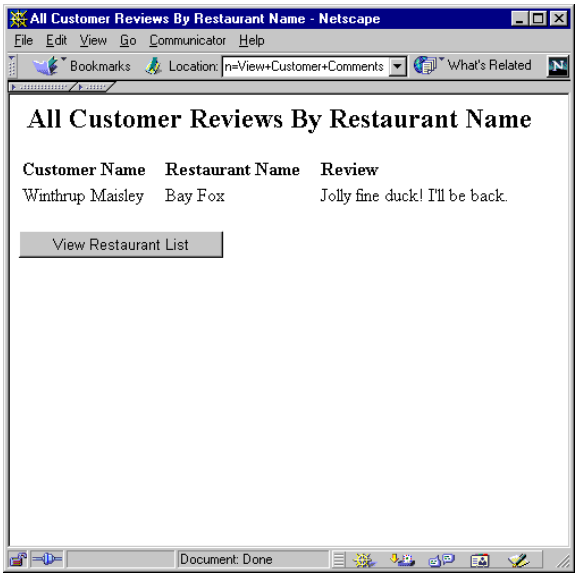


5. 名前と評価を入力し、「Submit」ボタンをクリックします。

   CustomerReview データベース表にテキストが追加され、Restaurant Listing ページが再表示されます。

6. 自分で入力した評価を表示するには、レストランの「View Customer Comments」ボタンをクリックします。

   All Customer Reviews By Restaurant Name ページが表示されます。



   必要な分だけ評価を追加して表示します。

7. 評価の入力が終了したら、ブラウザで別の Web ページを指定するか、ブラウザを終了してアプリケーションを終了します。

チュートリアルアプリケーションの構造

このチュートリアルアプリケーションの中枢は EJB 層で、これにはエンティティタイプの 2 つのEnterprise Beans が含まれています。これは EatersDigest の 2 つのデータベース (Restaurant と CustomerReview)、および 2 つの詳細クラスを表すもので、各フィールドで getter/setter メソッドを使用してエンティティ Bean フィールドと同じ内容を書き込みます。詳細クラスを使用すると、データベースのデータを取得する際に、エンティティ Bean に対するメソッド呼び出しの数を減らすことができます。

図 2-1 は、EatersDigest アプリケーションの構造を表しています。

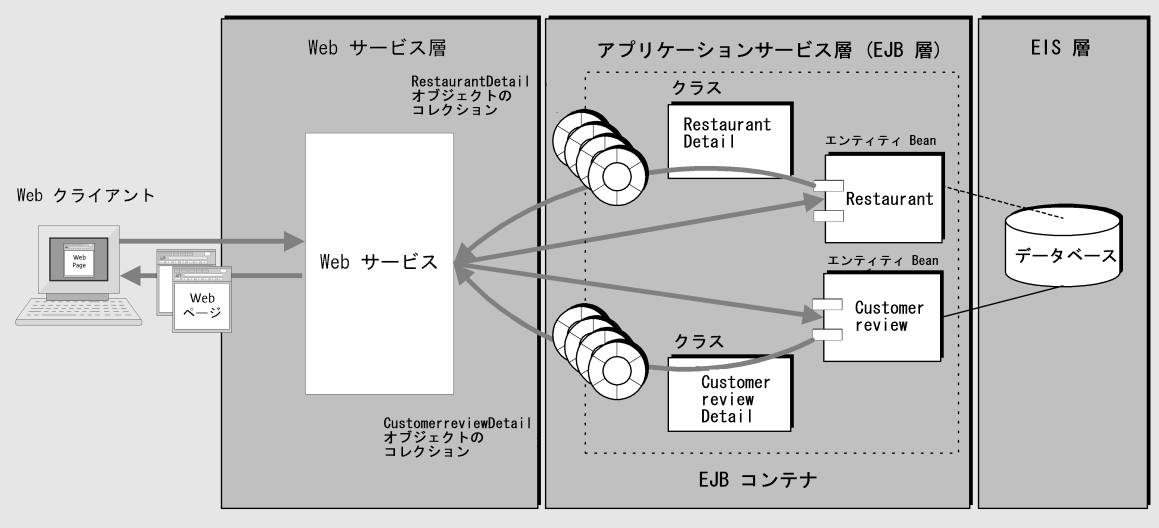


図 2-1　EatersDigest アプリケーションの構造

図 2-1 では、アプリケーションサービスコンポーネントと Web サービスとの接続が、意図的に曖昧に表現されています。これは、このチュートリアルで、クライアントとエンティティ Beans 間のやりとりを管理するために、2 つの方法が示されているためです。1 つめの方法は、第 3 章 (および 図 2-2) に示されているもので、セッション Enterprise Bean を使用してトラフィックマネージャとして機能します。もうひとつの方法は 第 4 章に示されているもので、Web サービスモジュール機能を使用し、重要なアプリケーションビジネスメソッドに基づいて XML オペレーションを作成し、自動的に Web サービスを生成してそれらを管理します。

アプリケーションの要素

図 2-1 に示されている要素について簡単に説明します。

- アプリケーションサービス層 (EJB 層)

  このチュートリアルでは、最初に EJB 層を作成してテストします。EJB 層は、次の要素で構成されています。

  - 2 つのエンティティ Enterprise Beans — コンテナ管理の持続性 (Container-Managed Persistence: CMP) を使用し、アプリケーションの 2 つのデータベース表を表します。
  - 2 つの詳細クラス — 返されたデータベースレコードを保持します。

  第 3 章で使用するものには、次のものが含まれています。

  - ステートレスなセッション Enterprise Bean — クライアントの要求を管理し、クライアントに返されるオブジェクトをフォーマットします。

  第 4 章で使用するものには、次のものが含まれています。

  - クライアント要求を満たすエンティティ Bean メソッドに基づいた XML オペレーション。

- Web サーバー層

  第 3 章 — Forte for Java のテストアプリケーション機能を使用して各 Enterprise Bean をテストする際に、Web モジュールを自動的に生成します。この Web モジュールでは、サーブレットと JSP ページが自動的に生成されます。これによって、各 Bean のメソッドを実行することができます。

  第 4 章 — EJB 層で選択されたメソッドに基づいて XML オペレーションを作成した後で Web サービス層を生成し、これらの XML オペレーションに、Web サービスに対する参照を追加します。クライアント JSP ページは自動的に作成され、これによって、XML オペレーションを実行することができます。

- クライアント

  クライアント構成要素は、アプリケーションページを表示する Web ブラウザです。第 5 章では、第 4 章の Web サービスモジュールによって作成された生成済の出力ページに基づいて、用意されているクライアントページを使用します。

EJB 層の詳細

EatersDigest アプリケーションの EJB 層には、エンティティタイプの 2 つの Enterprise Beans、および 2 つの詳細クラスが含まれています。第 3 章では、その他のコンポーネント (セッション Bean) を使用し、クライアントと Enterprise Beans のやりとりを管理します。

- `Restaurant` CMP EJB 構成要素

  `Restaurant` Bean はエンティティ Bean であり、コンテナ管理の持続性 (CMP) を使用し、Restaurant データベース表のデータを表します。

- `Customerreview` CMP EJB 構成要素

  CMP タイプのエンティティ Bean である `Customerreview` エンティティ Bean も、CustomerReview データベース表のデータを表します。

- `RestaurantDetail` クラス

  この構成要素には、`Restaurant` エンティティ Bean と同じフィールド、および各フィールドに対する getter/setter メソッド (エンティティ Bean のリモート参照からこのデータを取得するためのもの) が定義されています。このコンストラクタは、レストランデータを表すオブジェクトをインスタンス化します。次に、このオブジェクトはクライアントの JSP または HTML ページにフォーマットされ、参照できるようになります。

- `CustomerreviewDetail` クラス

  この構成要素は、`RestaurantDetail` クラスが `Restaurant` エンティティ Bean に対して機能するのと同様に、`Customerreview` エンティティ Bean に対して機能します。

- `DiningGuideManager` セッション EJB 構成要素

  この構成要素はステートレスセッション Bean で、これを使用してクライアントとエンティティ Bean のやりとりを管理します。これは、第 3 章の中で、やりとりを管理するための標準 J2EE として紹介されています。第 4 章で説明しているように、代わりに Web サービスモジュールを使用し、アプリケーションのビジネスメソッドに基づいて XML オペレーションを使用する Web サービスを作成する場合は、このコンポーネントは必要ありません。このセッション Bean では、アプリケーションの構造は、図 2-2 のようになります。

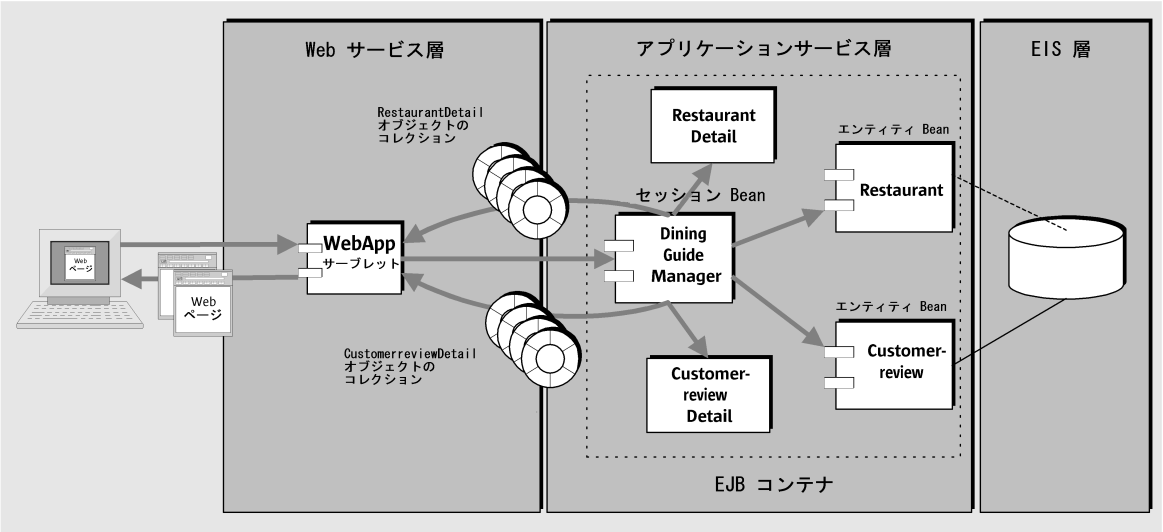


図 2-2　Manager セッション Bean を使用した EatersDigest アプリケーションの構造

チュートリアルアプリケーションの作成に必要な作業

チュートリアルを作成する作業は 3 つの章に分かれています。最初の章 (第 3 章) では、EJB 層を作成し、IDE のテストアプリケーション機能を使用し、作業用のエンティティ Bean をそれぞれテストします。この章では、セッション Bean を作成してトラフィックを管理します。これは、Web サービスとクライアントを手動で作成する場合の一般的なモデルです。

2 番目の章 (第 4 章) では、Web サービスモジュールを使用して Web サービスを作成する場合には、セッション Bean が不要であることを理解します。Web サービスモジュールを使用すると、アプリケーションのビジネスメソッドをベースとして、クライアントとエンティティ間の Bean トラフィックを直接管理することができます。クリティカルなビジネスメソッドに対する XML オペレーションを作成し、Web サービスによって、このオペレーションにアクセスするための Web サービスと基本的なクライアント JSP ページを作成します。

最後の章 (第 5 章) では、用意したクライアントページ (第 4 章で生成したページの改良版) を使用します。このページは、20 ページの「ユーザーから見たチュートリアルアプリケーション」 に記載されているアプリケーションを終了します。

EJB 構成要素の作成

第 3 章では、Forte for Java のさまざまな機能を使用し、以下の処理を行う方法について学習します。

- EJB ビルダーを使用してエンティティ Bean とセッション Bean を簡単に作成する
- データベーススキーマから、(getter/setter メソッドを使用して) クラスを生成する
- テストアプリケーション機能を使用し、Enterprise Bean からテスト J2EE アプリケーションをアセンブルする
- J2EE アプリケーションに EJB 参照を追加する
- J2EE リファレンス実装アプリケーションサーバーにテストアプリケーションを配備する
- テストアプリケーション機能によって作成されたテストクライアントページから、Enterprise Bean メソッドを実行する

EJB ビルダーの使用方法

EJB ビルダーウィザードは、Enterprise Bean を構成するさまざまなコンポーネントを自動的に作成します。そのセッション Bean はステートレスまたはステートフルで、コンテナ管理の持続性 (CMP) または Bean 管理の持続性 (BMP) の特徴を持つエンティティ Bean です。第 3 章 では、既存のデータベース表に基づいて 2 つの CMP エンティティ Bean を作成します。

エンティティ Bean を作成する場合に、作成プロセスの間にデータベースにどのように接続するか、また、フィールドが表の列を表すエンティティ Bean をどのように生成するかについて学習します。Bean の基本的な部分は、Java コードを使用して Forte for Java のエクスプローラに生成されます。この Java コードは、ホームインタフェース、リモートインタフェース、Bean クラス、(適切な場合は) 主キークラスに対し、すでに生成されているものです。ここでは、(Bean を全体的に表す) 論理的な Bean ノードを使用して Bean プロパティを編集、変更する方法を学習します。EJB ビルダーの GUI 機能を使用し、作成メソッド、検索メソッド、ビジネスメソッドを追加する方法についても学習します。

詳細 (Detail) クラスの生成

このチュートリアルでは透過的な持続性 (Transparent Persistence: TP) は使用しませんが、このツールを使用することで、詳細 (Detail) クラスを効率よく作成することができます。詳細クラスには、エンティティ Bean と同じフィールドを定義する必要があります。透過的な持続性の機能を使用すると、データベース表から自動的に Java クラスを生成できます。このクラスには、データベースフィールド、各フィールドに対する getter/setter メソッドが含まれます。この機能を使用し、生成したクラスに持続性の機能を追加せずに、詳細クラスを作成することができます。

テストアプリケーション機能

Forte for Java IDE には、テスト用のクライアントを作成せずに、Enterprise JavaBean 構成要素をテストする機能が用意されています。この機能では、J2EE リファレンス実装をアプリケーションサーバーとして使用し、Enterprise Bean を (Web モジュールとクライアント JSP ページが含まれている) J2EE アプリケーションの一部として配備します。これらの JSP ページは 1 つの HTML ページにまとめられているため、Bean の 1 つのインスタンスを作成し、そのビジネスメソッドを実行することができます。

3 つの Enterprise Bean に対しては、それぞれ個別のテストアプリケーションを作成します。エンティティ Bean に対しては、テストアプリケーションは J2EE アプリケーションを生成します。この J2EE アプリケーションには Web モジュールが含まれており、その Web モジュールには、クライアントが Web ブラウザから使用するために自動的に生成された JSP ページ、およびエンティティ Bean に対する EJB モジュールが含まれています。セッション Bean の EJB モジュールにはエンティティ Bean の EJB モジュールも含まれています。これは、エンティティ Bean のメソッドをコールするためです。IDE でコマンドを使用し、セッション Bean の EJB モジュールに対してエンティティ Bean の参照を追加します。

Web ブラウザでセッション Bean をテストすると、アプリケーションのすべてのビジネスメソッドを実行することができます。第 3 章 の最後には、独自の Web サービスとクライアントを手動で作成する場合に、テストクライアントアプリケーションを使用するためのガイドラインが記載されています。

チュートリアルの Web サービスの作成

第 4 章では、Forte for Java のさまざまな機能を使用し、以下の処理を行う方法について学習します。

- アプリケーションのビジネスメソッドを XML オペレーションとしてラップすることによって、アプリケーションの応答メカニズムを作成する
- XML オペレーションで、どのエンティティのビジネスメソッドがコールされるのかを指定する
- 複数の XML オペレーションを 1 つの Web サービスに組み込む
- Web サービスを含む J2EE アプリケーションを作成する
- Web サービスの実行時クラスとクライアントページを生成する

ビジネスメソッドから XML オペレーションを作成する

Web サービスモジュールの機能を使用し、特定のクライアント要求に応答する XML オペレーションを作成します。クライアントが要求を渡すと、Web サービスがそれに応答して XML オペレーションを実行します。XML オペレーションは、EJB 層のオブジェクト上でメソッドを呼び出すことができます。これは、EatersDigest アプリケーションでは、エンティティ Bean を意味します。このようにして、Web サービスは、第 3 章で作成したセッション Bean の管理機能を置き換えます。

「テンプレートウィザードから新規作成」を使用して Forte for Java のエクスプローラに XML オペレーションを作成し、オペレーションの基本的なプロパティ (名前、パッケージ、メソッド、メソッドが返す内容) を指定します (メソッドがコレクションを返す場合は、クラスにコレクションの構成要素を指定します) 。XML オペレーションを作成した後は、ソースエディタでオペレーションを定義します。ソースエディタには、XML オペレーションのツリービュー、オペレーションの入出力を XML 形式で示すウィンドウが表示されます。

第 4 章では、次の 4 つの XML オペレーションを作成します。

- `AllRestaurantsXMO` — `RestaurantDetail` オブジェクトのコレクション (データベース内のすべてのレストランに対して 1 つのオブジェクト) を返します。
- `AllCustomerreviewsByRestaurantNameXMO` — `restaurantname` パラメータを取得し、指定されたレストランに対する `CustomerreviewDetail` オブジェクトのコレクションを返します。

- AddCustomerReviewXMO — restaurantname、customername、review データの入力を取得し、CustomerreviewDetail オブジェクトを返します。
- EnterCustomerReviewXMO — element1 という文字列 (restaurantname フィールドで使用する文字列) を取得します。

4 番目の XML は、アプリケーションのエンティティ Bean のメソッドをベースとしたものではなく、Web サービスライブラリのメソッドをベースとしています。これは、パラメータを渡す場合に使用します。

Web サービスの作成

Web サービスは、「テンプレートウィザードから新規作成」を使用して名前とパッケージの場所を定義し、エクスプローラに作成した論理的なエンティティです。Web サービスを作成したら、そのサービスに XML オペレーションを追加します。Web サービスにドキュメントファイルを追加することもできます。第 5 章では、変更した JSP ページを追加してクライアントを作成します。

チュートリアル用の J2EE アプリケーションの作成

Web サービスの実行時クラスを生成するためには、J2EE アプリケーションを作成し、それに Web サービスを追加します。このようにすると、Web サービスの WAR ファイルと EJB JAR ファイルが有効になるため、プロパティをカスタマイズすることができます (たとえば「Web コンテキスト」プロパティをカスタマイズします)。第 3 章でテストアプリケーション機能を使用して作成した 2 つのエンティティ Bean からも、Web モジュールを追加します。これで EatersDigest の J2EE アプリケーションが完成し、配備する準備ができます。

チュートリアルアプリケーションの配備とテスト

J2EE リファレンス実装サーバーが実行されていることを確認し、IDE コマンドを使用して EatersDigest アプリケーションを配備します。このようにすると、各 XML オペレーションに対して 1 つの JSP が生成され、ページ上のすべてのオペレーションを示す開始ページが生成されます。また、各 XML オペレーションに対するクラスと、Web サービスに対するクラスが生成されます。

生成された JSP ページには、入力パラメータが必要なときの入力フィールド、オペレーションを実行するための「Submit Query」ボタンが含まれています。これらを使用して各オペレーションをテストします。

Forte for Java Enterprise Service Presentation Toolkit (Forte ESP ツールキット) がアプリケーションの Web サーバー (この場合は J2EE リファレンス実装) にインストールされると、開発者は、Forte for Java IDE から Forte ESP レジストリに XML オペレーションを公開することができます。これによって Web 設計者は、(Forte ESP ツールキットのプラグインがインストールされている) Macromedia® DreamWeaver™ や Adobe® GoLive™ のいずれかを使用し、XML オペレーションの動的データをページレイアウトの要素にマップすることができます。Web サービスが生成するページは、これらのページの開始ポイントとして機能させることもできます。

このチュートリアルでは、Forte ESP の開発は対象外となっており、XML オペレーションの公開は行わないため、Forte ESP ツールキットをインストールする必要はありません (詳細は、32 ページの「最後に」を参照)。また、実行時の Forte ESP タグを利用するために、XML オペレーションを公開したり、Forte ESP ツールキットをインストールする必要はありません。

クライアントページの改善

第 5 章では、用意されている 4 つの JSP ページを使用し、チュートリアルのクライアントを作成します。このクライアントの機能については、20 ページの「ユーザーから見たチュートリアルアプリケーション」 に記載されています。Forte ESP ツールキットには独自のチュートリアルが用意されており、Web サービスモジュールで生成されたページを変更する方法を説明しています。これらの手順は、第 5 章には明記されていません。

EatersDigest のチュートリアルアプリケーションを完成するには、完成したページを `WebService/Documents` ディレクトリにインストールし、EatersDigest アプリケーションを再配備して Web ブラウザで最初のページを指定するようにします。

ここでは、主な修正として HTML フォームが送信されたときにパラメータを渡す方法と顧客の評価を入力する方法を説明しています。

最後に

このチュートリアルアプリケーションは、比較的短期間 (1 日程度) で作成できるようにしたうえで、Forte for Java, Enterprise Edition の主な機能を表すアプリケーションを実行するように設計されています。そのため、次のような制約があります。

- エラー処理がない
- デバッグ手続きがない
- Web サービスの XML オペレーションの公開について説明していない
- Forte ESP ツールキットを使用したクライアントの作成について説明していない

XML オペレーションを公開し、Forte ESP を使用して動的データを取得する方法については、簡単なチュートリアルドキュメントが用意されています。これは、Forte for Java のポータルからダウンロードできる Enterprise Edition 固有のサンプルに含まれている「Hello Suite」の例となります。サンプルファイルを *your-user-dir*`/sampledir/examples` に展開すると、`~/examples/ee/webservices/hellosuite` で「Hello Suite」を確認できます。このサンプルの詳細は、『XML データサービス用 JSP のプログラミング』で説明しています。

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。EatersDigest アプリケーションは、IDE のユーザー設定ディレクトリにある `sampledir/tutorial` ファイルシステムの `DiningGuide` フォルダからアクセスできます。

第3章

EJB 層の作成

この章では、EatersDigest のチュートリアルアプリケーションの EJB 層を作成する方法を順を追って説明します。ここでは、EJB ビルダーを使用してエンティティ Bean とセッション Bean を作成する方法、およびこれらをテストするための IDE のテスト機能について学習します。この章のトピックは次のとおりです。

この章の作業を完了すると、チュートリアルアプリケーションの EJB 層全体を、配備されたテストアプリケーションとして実行できるようになります。

EJB 層を作成した後は、独自の Web サービスとクライアントページを自由に作成することができます。また、第 4 章に進み、Java™ Web Service Designer を使用してアプリケーションの Web サービスを作成する方法を学習し、第 5 章で、Forte ESP ツールキットdを使用してアプリケーションのクライアントページを作成する方法を学習することもできます。

チュートリアルの EJB 層の概要

この章では、チュートリアルアプリケーションの中枢モジュールである EJB 層を作成します。各コンポーネントを作成するたびに、IDE のテストアプリケーション機能を使用してテストを行います。このテストアプリケーションでは、テスト用の Web サービスとクライアントを自動的に作成します。

作成する EJB 層には、次の Bean が定義されます。

- Restaurant エンティティ Bean
- Customerreview エンティティ Bean
- DiningGuideManager セッション Bean
- RestaurantDetail Bean
- CustomerreviewDetail Bean

エンティティ Bean

エンティティ Bean とその作成方法およびプログラミングについては、『Enterprise JavaBeans コンポーネントのプログラミング』を参照してください。

エンティティ Bean には、概念を定義する共有データセットに対し、一貫したインタフェースが用意されています。このチュートリアルでは、2 つの概念 (レストラン、顧客の評価) を使用します。ここで作成する `Restaurant` および `Customerreview` のエンティティ Bean は、第 1 章で作成したデータベース表を表します。

これらのエンティティ Bean には、コンテナ管理の持続性 (CMP) または Bean 管理の持続性 (BMP) のいずれかを定義することができます。BMP のエンティティ Bean では、開発者は、Bean のフィールドをデータベース表の列にマップするためのコードを記述する必要があります。CMP のエンティティ Bean では、EJB の実行環境で持続性の操作が管理されます。このチュートリアルでは、CMP のエンティティ Bean を使用します。IDE の EJB ビルダーウィザードを使用し、データベースに接続してマップする列を指定します。ウィザードでは、データベースにマップされるエンティティ Bean を作成します。

EJB ビルダーでは、CMP のエンティティ Bean 用のフレームワークを作成します。具体的には、エンティティ Bean を体系付けてカスタマイズを容易にするための論理ノード、必要なホームインタフェース、リモートインタフェース、Bean クラスを作成します。

ここでは、インスタンス変数 (CMP フィールド) を初期化するために、エンティティ Bean に対して`create` メソッドを手動で追加する必要があります。`create` メソッドを作成すると、このメソッドは、IDE によってホームインタフェースに自動的に送られます。また、対応する `ejbCreate` メソッドも Bean クラスに送られます。初期化の際に `create` メソッドにコードを追加すると、そのコードは、`ejbCreate` メソッドにも同時に追加されます。また、テストを行うために、各エンティティ Bean にビジネスメソッドを作成します。

セッション Bean

セッション Bean の詳細については、『Enterprise JavaBeans コンポーネントのプログラミング』を参照してください。

エンティティ Bean が共有データを表している場合は、セッション Bean は (概念間をまたがってはいるが共有されていない) データにアクセスします。セッション Bean は、特定のタスクを完了するために必要な手順も管理します。セッション Bean は、ステートフル、ステートレスのどちらに定義することもできます。ステートフルなセッション Bean は、クライアントとの継続する対話状態を保持しながら、クライアントの代わりにタスクを実行します。ステートレスなセッション Bean は、クライアントとの対話状態を保持せず、1 つのクライアントに占有されません。ステートレスな Bean が、あるクライアントに対するメソッドの呼び出しを完了すると、別のクライアントの要求をサービスできるようになります。

EatersDigest アプリケーションでのクライアント要求には、データベース内のすべてのレストランのデータを取得する、特定のレストランについて顧客の評価をすべて検索する、特定のレストランについて評価を投稿する、という処理があります。これらの要求は相互に関連付けられているわけではなく、対話状態を保持する必要はありません。このため、EatersDigest のチュートリアルでは、ステートレスなセッション Bean を使用して各要求に必要な手順を管理します。

セッション Bean は、クライアントの要求を満たすために、レストランと顧客評価のレコードのコレクションを作成する処理を繰り返します。この処理は、エンティティ Bean 上で各フィールドに対して getter/setter メソッドを追加することによって対処することができますが、この方法では、セッション Bean で表内の行を取得するたびに、各フィールドに対してメソッドを呼び出さなければなりません。このチュートリアルでは、メソッド呼び出しの回数を減らすために、特別なヘルパークラス (詳細クラス) を使用して行データを保持します。

詳細 (Detail) クラス

詳細クラスには、対応するエンティティ Bean と同じフィールド、および各フィールドについての getter/setter メソッドを定義します。セッション Bean は、エンティティ Bean を参照する際に対応する詳細クラスを使用し、エンティティ Bean で返されるリモート参照の各インスタンスを作成します。セッション Bean は、単純に詳細クラスのコンストラクタを呼び出すだけで、データの行をインスタンス化して参照することができます。セッション Bean は、このようにして行インスタンスのコレクションを作成することができます。このコレクションは、参照できるように、クライアントの HTML ページにフォーマットすることができます。

図 3-1 は、詳細クラスの機能について表しています。

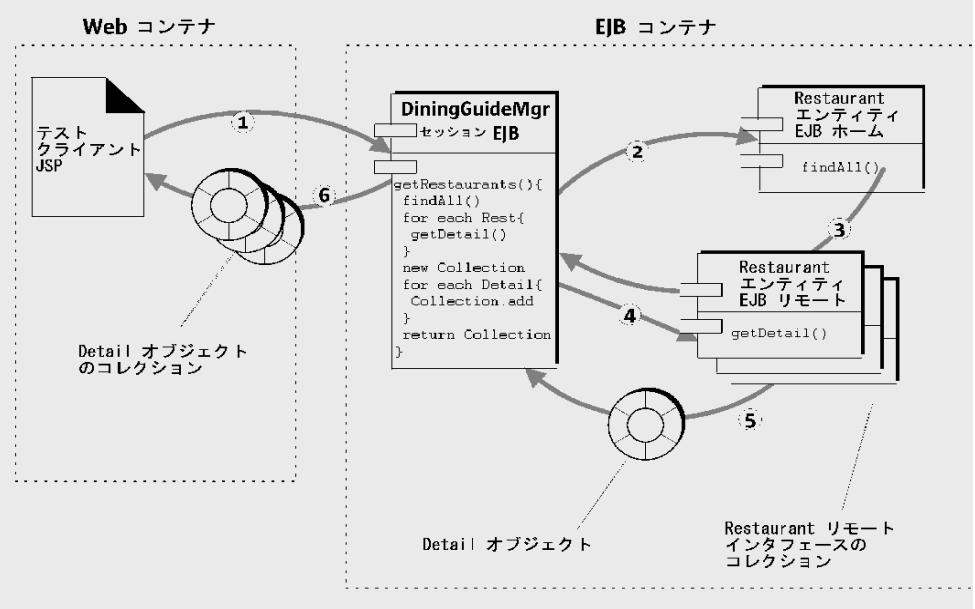


図 3-1　詳細クラスの機能

1. Web コンテナは、すべてのレストランデータに対するクライアントの要求を、`DiningGuideManager` セッション Bean に渡します。
2. セッション Bean は、`Restaurant` エンティティ Bean の `findAll` メソッドを呼び出し、`Restaurant` エンティティ Bean を検索します。

3. findAll メソッドは、エンティティ Bean で使用できるリモート参照をすべて取得します。

4. リモート参照が返されると、セッション Bean は、それぞれについて Restaurant Bean の getRestaurantDetail ビジネスメソッドを呼び出し、RestaurantDetail クラスをフェッチします。

5. getRestaurantDetail メソッドは RestaurantDetail オブジェクトを返します。このオブジェクトはコレクションに追加されます。

6. セッション Bean は、RestaurantDetail のすべてのオブジェクトのコレクションを Web コンテナに返し、クライアントで参照できるように適切にフォーマットします。

手順のまとめ

EJB 層を作成する手順は次のとおりです。

1. エンティティ Bean を作成します。

EJB ビルダーウィザードを使用し、CMP のエンティティ Bean スケルトンを作成します。次に、テストを行うために、create メソッドと finder メソッド、および簡単なビジネスメソッドを追加します。

2. データベーススキーマから詳細クラスを作成します。

データベーススキーマを取得し、Restaurant 表および CustomerReview 表からクラスを生成します。これらのクラスは、各フィールドに対する getter/setter メソッドを持つように自動的に生成されます。

3. エンティティ Bean 上に get メソッドを作成し、詳細クラスを取得します。

4. IDE のテストアプリケーション機能を使用し、エンティティ Bean のメソッドをテストします。

自動的に生成されたテストクライアントを Web ブラウザで表示し、create メソッドを実行して Bean のインスタンスを作成し、ビジネスメソッドを有効にします。次に、Bean のビジネスメソッドを実行します。

5. セッション Bean を作成します。

EJB ビルダーを使用してステートレスなセッション Bean のスケルトンを作成し、その Bean の create メソッドを修正してエンティティ Bean について検索できるようにします。次に、getter メソッドを作成し、各エンティティ Bean に対して (詳細クラスの) 詳細オブジェクトのコレクションを作成します。

6. テストアプリケーション機能を使用し、セッション Bean をテストします。

EJB モジュールのプロパティシートで、CMP のエンティティ Bean への参照を追加します。テストアプリケーションを作成し、テストアプリケーションの EJB モジュールに EJB モジュールを追加します。テストクライアントを使用してセッション Bean のインスタンスを作成し、そのメソッドを実行します。

システムの設定

チュートリアルアプリケーションを使用する前に、次の作業を行う必要があります。詳細は、第 1 章を参照してください。

1. PointBase ネットワークサーバーを起動します。
2. チュートリアルのデータベース表を作成します。
3. システムで Web ブラウザを使用できるようにします。
4. J2EE サーバー (リファレンス実装を含む Java 2 SDK, Enterprise Edition) を起動します。
5. Forte for Java, Enterprise Edition を起動します。
6. IDE のサーバーレジストリに、リファレンス実装のインスタンスを作成します。

EJB ビルダーによるエンティティ Bean の作成

2 つのエンティティ Bean (`Restaurant`、`Customerreview`) を作成し、第 1 章で作成した 2 つのデータベース表を表します。

最初にディレクトリを作成し、アプリケーションを格納するファイルシステムとしてマウントします。次に、EJB 層のパッケージを作成します。最後に、パッケージ内にエンティティ Bean を作成します。

1. ファイルシステム内の任意の場所にディレクトリを作成し、`DiningGuide` という名前を付けます。

   たとえば、`MyWork` というユーザー設定ディレクトリに `DiningGuide` ディレクトリを作成します。

2. Forte for Java IDE で「ファイル」>「ファイルシステムをマウント」を選択し、`DiningGuide` ディレクトリをマウントします。

   `c:¥MyWork¥DiningGuide` ディレクトリが、エクスプローラにマウントされます。

3. 新しくマウントしたファイルシステムを右クリックし、「新規パッケージ」を選択します。

   このパッケージを使用してアプリケーションの EJB 層を保持します。

4. 新しいパッケージに `Data` という名前を付けます。

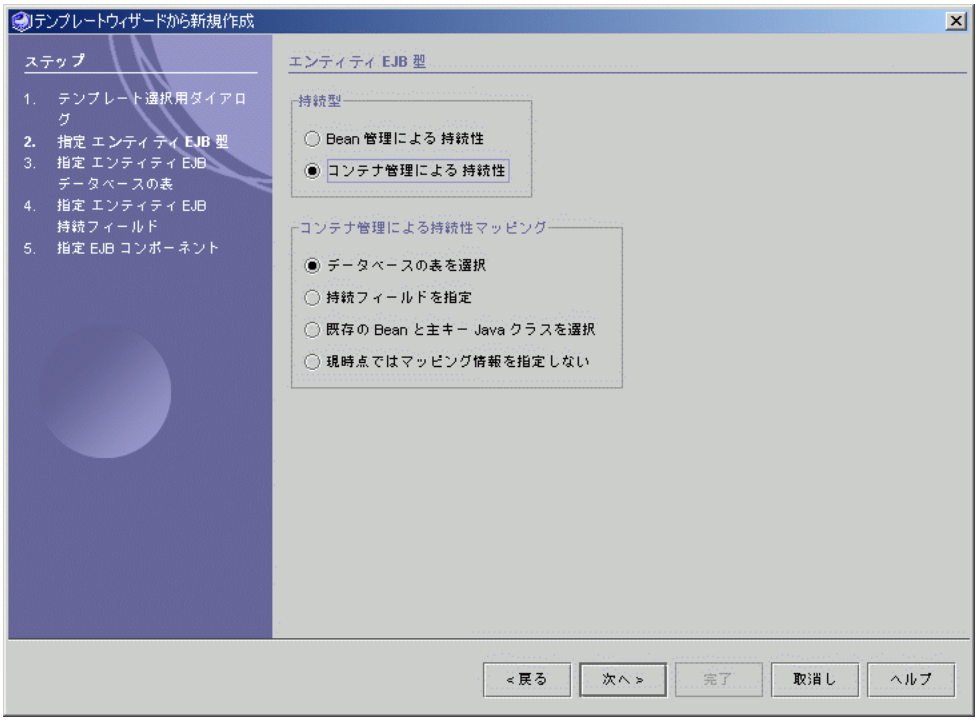
5. `Data` パッケージを右クリックし、「新規」>「EJB」>「Entity Bean」を選択します。

   EJB ビルダーウィザードが表示され、「エンティティ EJB 型」区画が示されます。

6. 「コンテナ管理による持続性」を選択します。

7. 「データベースの表を選択」(デフォルト) を選択します。

   EJB ビルダーは次のように表示されます。



8. 「次へ」をクリックします。

   「エンティティ EJB データベースの表」区画が表示されます。

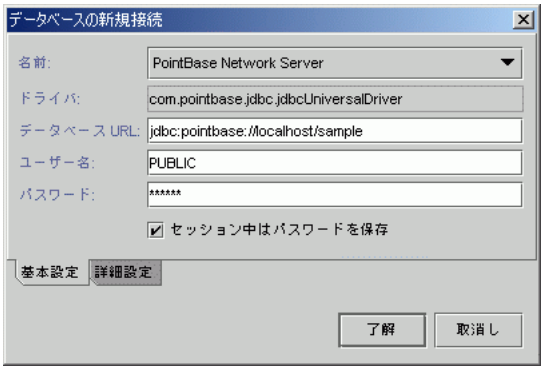
9. 「データベース接続からデータベースの表を選択」を選択し、「接続を追加」をクリックします。

   「データベースの新規接続」ダイアログが表示されます。

10. 次の値を入力します。

    名前: **PointBase Network Server**
    ドライバ : **com.pointbase.jdbc.jdbcUniversalDriver**
    データベース URL : **jdbc:pointbase://localhost/sample**
    ユーザー名 : **PUBLIC**
    パスワード : **PUBLIC** (アスタリスクで表示されます)

ダイアログは次のように表示されます。



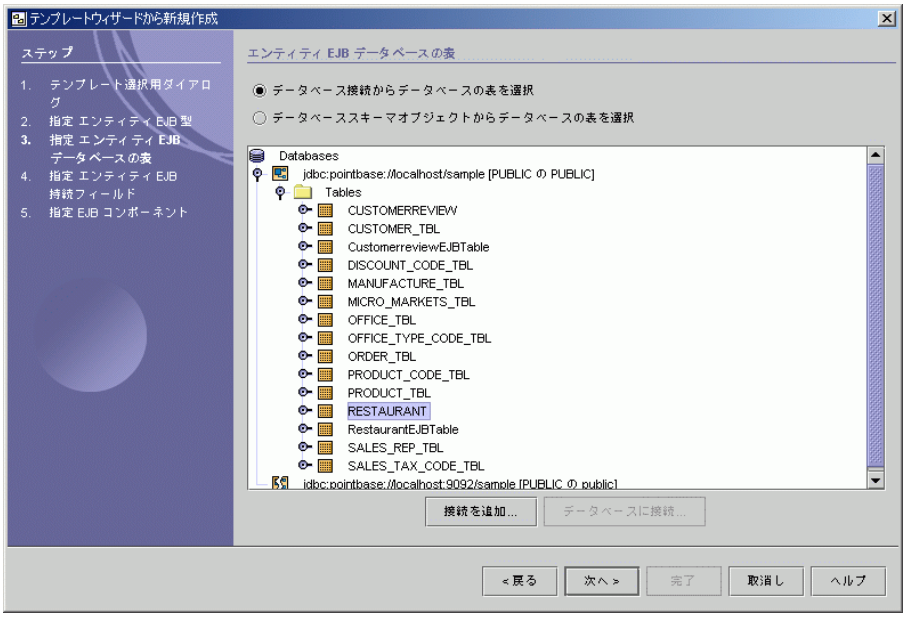
11. 「了解」をクリックしてデータを確定し、ダイアログを閉じます。

「エンティティ EJB データベースの表」区画に新しい接続が表示されます。

注 - 接続が表示されない場合は、PointBase Network Server が起動されているかどうかを確認します。3 ページの「チュートリアルデータベース表の作成」 に記載されている手順 1 を参照してください。

---

12. 新しい接続 `sample` と `Tables` フォルダを開き、RESTAURANT 表を選択します。



13. 「次へ」をクリックします。

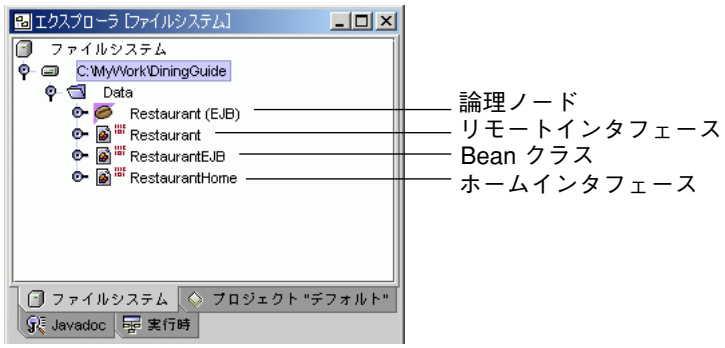
    「エンティティ EJB 持続フィールド」区画が表示されます。ここでは、Restaurant データベース表の列と、対応する持続フィールドが並んで表示されます。持続フィールドは、ウィザードで `Restaurant` エンティティ Bean を作成したときにマップされる列となります。デフォルトのラベルをすべて承認します。

14. もう一度「次へ」をクリックします。

    EJB コンポーネントページが表示され、作成される `Restaurant` Bean のパートが示されます。EJB ビルダーウィザードでは、新しいエンティティ Bean に、データベース表と同じ名前を自動的に定義します。

15. 「完了」をクリックします。

    新しい `Restaurant` エンティティ Bean、およびそのパートがすべて作成され、エクスプローラに表示されます。



    3 つのパートはクラスで、4 つめのパートは論理ノードです。論理ノードを作成してエンティティ Bean のすべての要素をグループ化し、処理を簡単にすることができます。

    ここで、`Customerreview` エンティティ Bean を作成します。

16. Data パッケージを右クリックし、「新規」>「EJB」>「Entity Bean」を選択します。

17. 「コンテナ管理による持続性」を選択します。

18. 「データベースの表を選択」を選択します。

    Restaurant Bean を作成した後で IDE を終了していない場合は、sample データベースに接続されている状態です。これは、  アイコンで確認することができます。次の手順に進みます。

    Restaurant Bean を作成した後で IDE を終了している場合は、sample データベースは一覧に表示されますが、アイコンは壊れた形 () で示されます。接続するには、次のようにします。

    a. データベースの一覧で次の項目を選択します。

       ```
       jdbc:pointbase://localhost/sample[PUBLIC on PUBLIC]
       ```

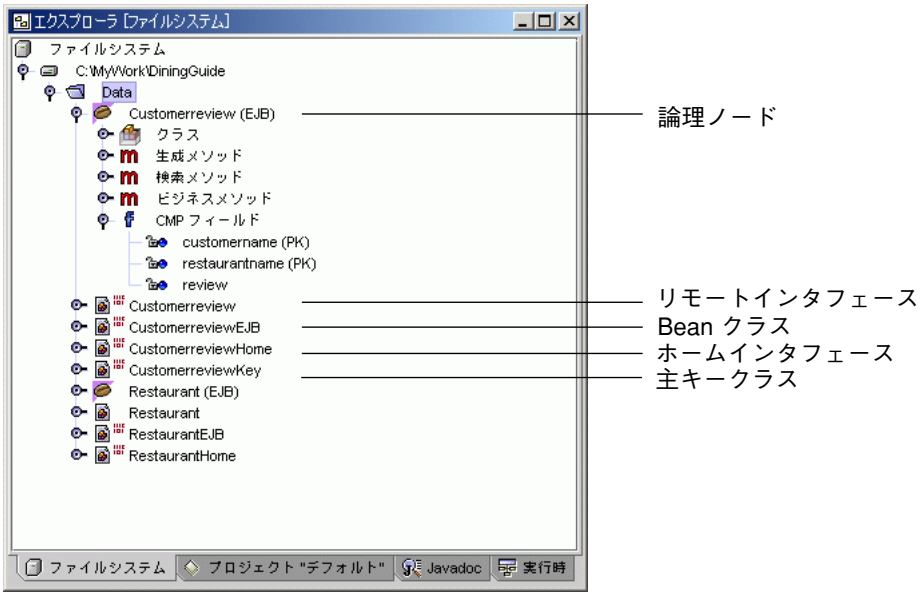
    b. 「データベースに接続」をクリックします。

    c. 「接続」ダイアログで、パスワードに PUBLIC と入力して「了解」をクリックします。

19. データベースノード、および Tables フォルダを開き、CUSTOMERREVIEW ビューを選択して「次へ」をクリックします。

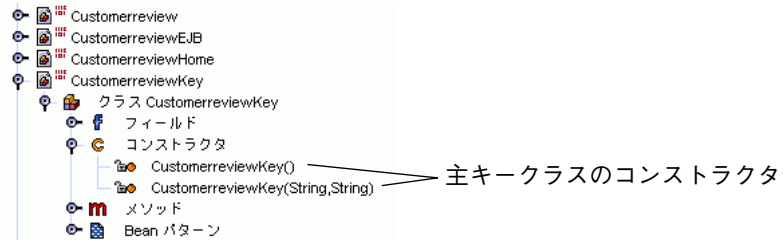
20. 「持続フィールド」区画で「次へ」をクリックし、最後の区画 (「コンポーネント」区画) で「完了」をクリックします。

    Data パッケージの Customerreview エンティティ Bean が、エクスプローラに表示されます。ここで、CustomerreviewKey Bean というコンポーネントも表示されています。エンティティ Bean が複合主キーを持っている場合には、この Bean が自動的に作成されます (この表に複合主キーが定義されているかどうかを確認するには、第 1 章の表 1-1 を参照してください)。



21. `CustomerreviewKey` クラスを開き、「コンストラクタ」ノードを開きます。

    EJB ビルダーでは、2 つのコンストラクタが自動的に作成されます。1 つは引数をとらないコンストラクタで、もう 1 つは多重定義されたコンストラクタ (2 つの複合主キーフィールドを設定するもの) です。



次に、エンティティ Bean メソッドを作成します。

CMP エンティティ Bean に対する生成メソッドの作成

2 つのエンティティ Bean に対して生成メソッドを作成し、パラメータとコードを追加して Bean のインスタンスのフィールドを初期化します。

1. Forte for Java のエクスプローラで、論理ノード `Restaurant` (Bean アイコン ) を右クリックします。
2. コンテキストメニューで「新規生成メソッド」を選択します。

   「新規生成メソッド」ダイアログが表示されます。
3. 「追加」ボタンを使用し、7 つのパラメータ (Restaurant 表の各列に対して 1 つずつ) を次のように作成します。

   `restaurantname` (**`java.lang.String`**)
   `cuisine` (**`java.lang.String`**)
   `neighborhood` (**`java.lang.String`**)
   `address` (**`java.lang.String`**)
   `phone` (**`java.lang.String`**)
   `description` (**`java.lang.String`**)
   `rating` (**`java.lang.Integer`**)

   注 - テストアプリケーション機能を使用して Bean をテストする際には、これらのパラメータを作成する順序が重要になります。パラメータは、ここに記載されている順序で作成してください。

   デフォルトで作成される 2 つの例外は、そのままにしてください。
4. 「了解」をクリックします。

   `RestaurantHome` インタフェースに `create` メソッドが送られ、`Restaurant` Bean クラス (`RestaurantEJB`) に `ejbCreate` メソッドが送られます。Bean クラスには、関連する `ejbPostCreate` メソッドも追加されます。
5. 論理ノード `Restaurant`、および「生成メソッド」フォルダを開き、`create` メソッドをダブルクリックします。

   テキストエディタに Bean の `ejbCreate` メソッドが表示されます。

6. 次のように、Bean インスタンスのフィールドを初期化するためのコードを ejbCreate メソッドの本体に追加します。

```
public String ejbCreate(java.lang.String restaurantname,
java.lang.String cuisine, java.lang.String neighborhood,
java.lang.String address, java.lang.String phone,
java.lang.String description, java.lang.Integer rating) throws
CreateException {
    if (restaurantname == null) {
        throw new CreateException("The restaurant name is
            required.");
    }
    this.restaurantname = restaurantname;
    this.cuisine = cuisine;
    this.neighborhood = neighborhood;
    this.address = address;
    this.phone = phone;
    this.description = description;
    this.rating = rating;

    return null;
}
```

ejbCreate メソッドが呼び出されると、この Bean のコンテナ管理フィールドに基づいて、データベースに新しいレコードが作成されます。

次に、Customerreview の create メソッドを作成します。

7. 論理ノード Customerreview (Bean アイコン ) を右クリックし、「新規生成メソッド」を選択します。

8. 「追加」ボタンを使用し、次の 3 つのパラメータ (CustomerReview 表の各列に対して 1 つずつ) を作成します。

restaurantname (**java.lang.String**)
customername (**java.lang.String**)
review (**java.lang.String**)

注 - 手順 3 で説明したように、これらのパラメータは、上記の順序で作成する必要があります。

---

9. 「了解」をクリックします。

10. 論理ノード Customerreview、および「生成メソッド」フォルダを開き、create メソッドをダブルクリックします。

    テキストエディタに Bean の ejbCreate メソッドが表示されます。

11. 次のように、Bean インスタンスのフィールドを初期化するためのコードを ejbCreate メソッドの本体に追加します。

```
public CustomerreviewKey ejbCreate(java.lang.String
restaurantname, java.lang.String customername, java.lang.String
review) throws CreateException {
    if ((restaurantname == null) || (customername == null)) {
        throw new CreateException("Both the restaurant name and
customer name are required.");
    }
    this.restaurantname = restaurantname;
    this.customername = customername;
    this.review = review;

    return null;
}
```

    ejbCreate メソッドが呼び出されると、この Bean のコンテナ管理フィールドに基づいて、データベースに新しいレコードが作成されます。

    次に、2 つの エンティティ Bean について検索メソッドを作成します。このメソッドは、Bean のすべてのインスタンスまたは特定のインスタンスを検出するものです。

エンティティ Bean の検索メソッドの作成

すべての Restaurant データを検出するには、Restaurant Bean に findAll メソッドを作成します。また、特定のレストランに関する Customerreview データを検出するには、Customerreview Bean に findByRestaurantName を作成します。

1. 論理ノード Restaurant を右クリックし、「新規検索メソッド」を選択します。
2. 「新規検索メソッド」ダイアログで、次の値を入力します。

    名前 : **findAll**
    戻り型 : **java.util.Collection**

    デフォルトの例外は、そのままにします。

3. 「了解」をクリックします。

   Restaurant Bean の Home インタフェースに、新しい findAll メソッドが作成されます。

   次に、Customerreview Bean に検索メソッドを作成します。

4. 論理ノード Customerreview を右クリックし、「新規検索メソッド」を選択します。

5. 「新規検索メソッド」ダイアログで、次の値を入力します。

   名前 : **findByRestaurantName**
   戻り型 : **java.util.Collection**

6. 次の値のパラメータを追加します。

   フィールド名 : **restaurantname**
   型 : **java.lang.String**

   デフォルトの例外は、そのままにします。

7. 「了解」をクリックします。

   Customerreview Bean の Home インタフェースに、新しい findByRestaurantName メソッドが作成されます。

8. Customerreview の検索メソッドにある findByRestaurantName メソッドのプロパティシートを開きます。

   「プロパティ」ウィンドウが表示されていない場合は、メソッドを右クリックし、コンテキストメニューで「プロパティ」を選択します。

9. プロパティシートの「J2EE RI」タブをクリックします。

10. SQL 文の値フィールドをクリックし、ブラウズ「...」ボタンをクリックして SQL 文全体を表示します。

    ここでは、WHERE 句は完成していません。

11. SQL 文を次のように修正します。

```
SELECT "customername", "restaurantname" FROM
"CustomerreviewEJBTable" WHERE "restaurantname" = ?1
```

    使用する数字は、検索メソッドの restaurantname パラメータの位置によって決定されます (この場合は 1 つしかないため、1 となります)。

テスト用ビジネスメソッドの作成

いずれかのパラメータの値を返すエンティティ Bean に対してビジネスメソッドを作成し、後で Bean をテストできるようにします。

1. 論理ノード Restaurant を右クリックし、「新規ビジネスメソッド」を選択します。

   「新規ビジネスメソッド」ダイアログが表示されます。

2. 次の値を入力します。

   名前 : **getRating**
   戻り型 : **java.lang.Integer**

3. 「了解」をクリックします。

   Restaurant リモートインタフェース、および RestaurantEJB クラスに getRating メソッドの宣言が送られます。

4. 論理ノード Restaurant、および「ビジネスメソッド」フォルダを開き、getRating メソッドをダブルクリックします。

5. getRating メソッドの本体に、次のコードを追加します。

```
public java.lang.Integer getRating() {
    return rating;
}
```

   getRating メソッドを使用すると、選択したレストランのレコードのランクを表す列の値が返されます。

6. Data パッケージを右クリックし、コンテキストメニューの「すべてを構築」を選択します。

   パッケージ全体がエラーなしでコンパイルされます。これで、エンティティ Bean を確認する準備ができました。

   次に、Customerreview Bean に対して類似のメソッドを作成します。

7. 論理ノード Customerreview を右クリックし、「新規ビジネスメソッド」を選択します。

8. 「新規ビジネスメソッド」ダイアログで、次の値を入力します。

   名前 : **getReview**
   戻り型 : **java.lang.String**

9. 「了解」をクリックします。

   Customerreview リモートインタフェース、および CustomerreviewEJB クラスに getReview メソッドの宣言が送られます。

10. 論理ノード Customerreview、および「ビジネスメソッド」フォルダを開き、getReview メソッドをダブルクリックします。

11. getReview メソッドの本体に、次のコードを追加します。

```
public java.lang.String getReview() {
    return review;
}
```

エンティティ Bean のコンパイル

次のようにしてエンティティ Bean をコンパイルします。

1. Data パッケージを右クリックして「すべてをコンパイル」を選択するか、論理 Bean を右クリックして「EJB クラスのコンパイル」を選択します。

   2 つの Bean がエラーなしでコンパイルされます。

2. エクスプローラで、未コンパイルを示すマークが消えていることを確認します。

詳細クラスを作成してエンティティ Bean データを表示する

36 ページの「詳細 (Detail) クラス」で説明したように、このチュートリアルでは「詳細 (Detail) クラス」という特別なクラスを使用します。これは、表示用に行データを保持し、エンティティ Bean へのメソッド呼び出しの回数を減らすためのメカニズムです。これらの詳細クラスには、対応する エンティティ Bean、各フィールドのアクセスメソッド、および各フィールドを設定するコンストラクタと同じフィールドを定義する必要があります。詳細クラスは手動で作成することもできますが、Forte for Java IDE には、クラスを作成する処理をサポートする機能も用意されています。

詳細クラスを生成するには、透過的な持続性用の持続可能な Java クラスを生成します。最初にデータベーススキーマをインポートし、持続性を定義する機能を使用せずに、そこから Java クラスを生成します。

データベーススキーマからのクラスの生成

詳細クラスを生成するには、次のようにします。

1. Forte for Java エクスプローラで `Data` パッケージを右クリックし、「新規」> 「Database」> 「Database Schema」を選択します。

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

2. 新しいスキーマに `DiningSchema` という名前を付け、「次へ」をクリックします。

   接続情報を示す区画が表示されます。

3. 「既存の接続」を選択し、リストから `sample` を選択します。

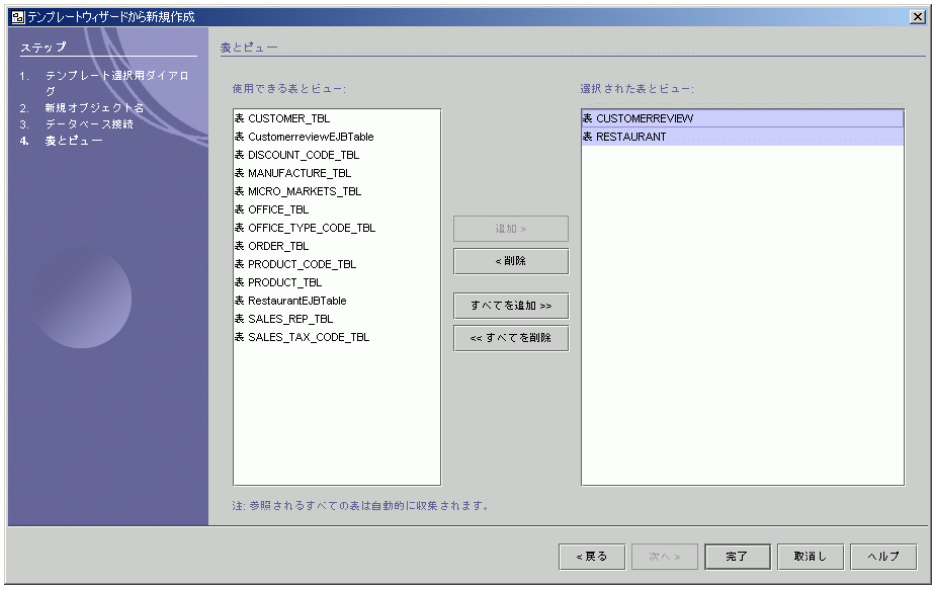
   エンティティ Bean を作成してから IDE を再起動した場合は、接続にログインするためのダイアログが表示されます。`PUBLIC` と入力して「了解」をクリックすると、ダイアログが閉じます。

4. 「次へ」をクリックします。

   取得する表を選択するための区画が表示されます。ここでは、`CUSTOMERREVIEW` と `RESTAURANT` を選択します。

5. 表 CUSTOMERREVIEW と表 RESTAURANT を選択して「追加」ボタンをクリックします (複数の表を選択するには、Ctrl キーを使用します)。

   これらの表が、次のように「選択された表とビュー」ウィンドウに表示されます。



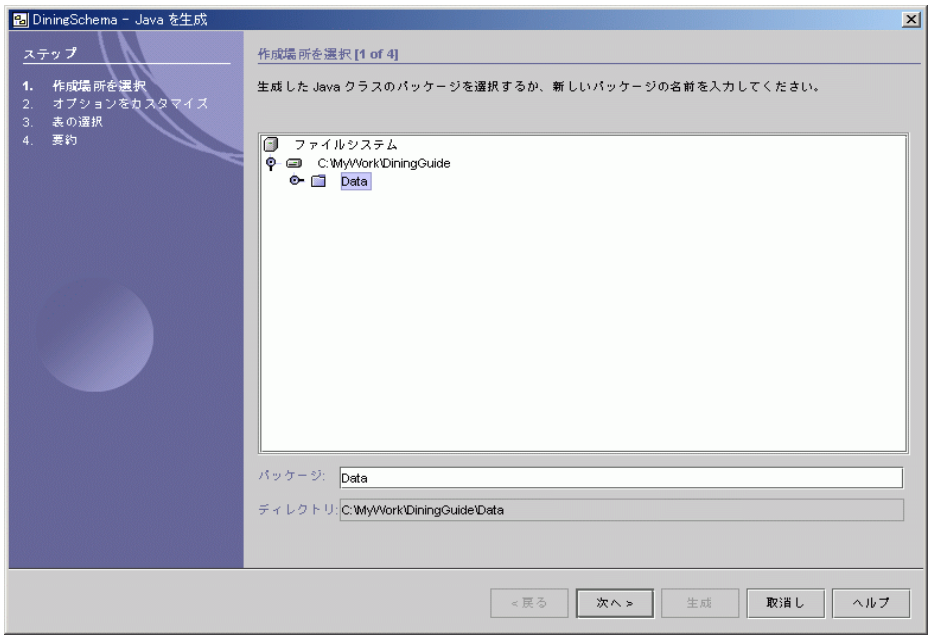
6. 「完了」をクリックします。

   進捗を表すウィンドウが表示されます。このウィンドウが消えると、Forte for Java のエクスプローラに新しいデータベーススキーマが表示されます。

7. `DiningSchema` ノード ( ) を右クリックし、コンテキストメニューの「Java を生成」を選択します。

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

8. Data パッケージを選択します。

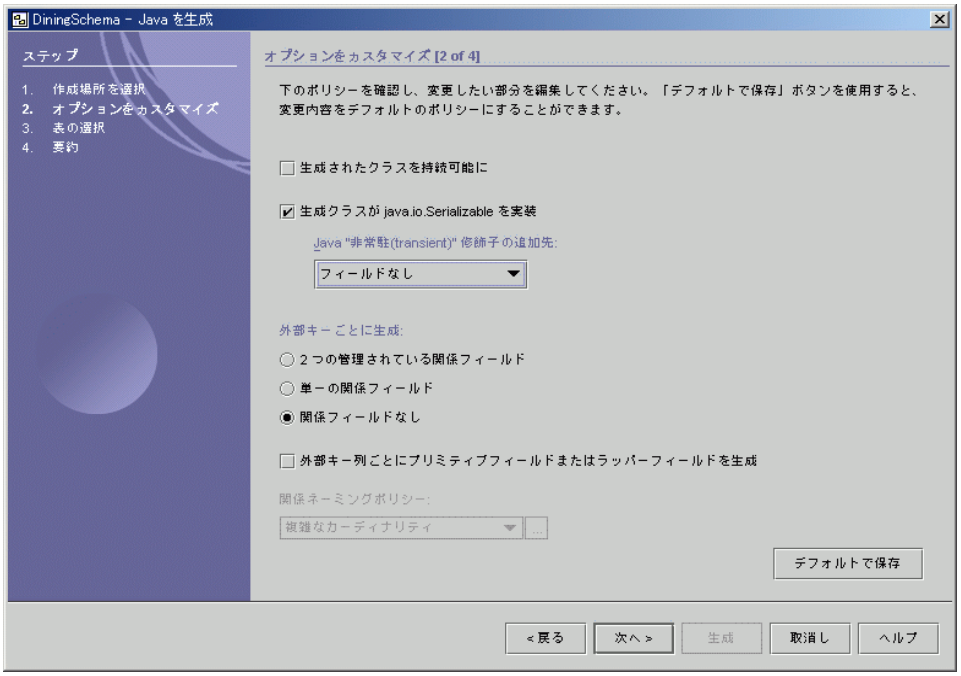


9. 「次へ」をクリックします。

10. 「オプションをカスタマイズ」区画で、次のオプションを設定します。

    生成されたクラスを持続可能に : 選択しない
    生成クラスが java.io.Serializable を実装 : 選択する
    Java 非常駐 (transient) 修飾子の追加先 : フィールドなし
    外部キーごとに生成 : 関係フィールドなし

    ウィザードのページは、次のように表示されます。



11. 「次へ」をクリックします。

    ウィザードに「表の選択」区画が表示されます。

12. 「すべての表を追加」ボタンをクリックします。

    表 RESTAURANT と表 CUSTOMERREVIEW を、ウィンドウの「選択された表とビュー」ウィンドウに移動します。次に、「Java クラス」ラベルを編集し、詳細クラスの名前を作成します。

13. 「Java クラス」の「Restaurant」ラベルをクリックし、名前を「RestaurantDetail」に変更して Enter キーを押します。

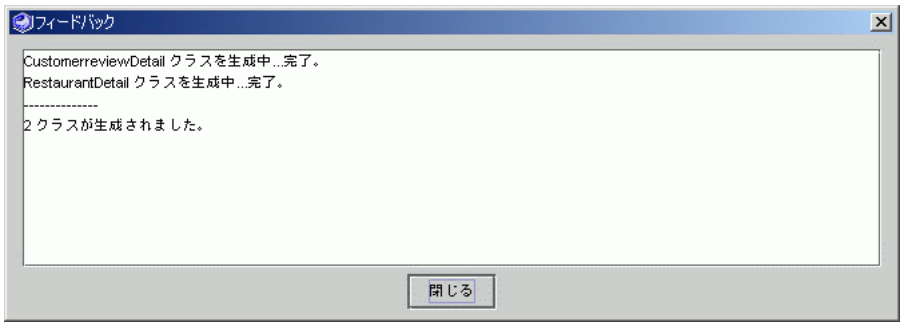
14. 同様に、「Customerreview」を「CustomerreviewDetail」に変更して Enter キーを押します。

15. 「次へ」をクリックします。

    ウィザードに、生成された表がまとめて表示されます。ここでは、CustomerreviewDetail と RestaurantDetail が表示されます。

16. 「生成」をクリックします。

    IDE に「フィードバック」ウィンドウが表示され、生成されたクラスが示されます。



17. ウィンドウを閉じます。

    エクスプローラに、RestaurantDetail クラスと CustomerreviewDetail クラスが表示されます。これで、詳細クラスを編集する準備ができました。

生成されたクラスの編集

ここでは主に、生成されたクラスのコンストラクタを編集し、各フィールドをインスタンス化します。ただし、生成された rating フィールドの型を最初に調整する必要があります。

rating フィールドの型の調整

RestaurantDetail クラスの rating フィールドには、Restaurant エンティティ Bean の同じフィールドとは別の型が定義されています。Restaurant.rating フィールドの型は Integer ですが、透過的な持続性では「このようなケースを考慮しない」という規則によって、RestaurantDetail.rating フィールドの型は Short になります。したがって、Restaurant エンティティ Bean のフィールドと一致するように、手動で型を Integer に変更する必要があります。このフィールドが他のメソッドで使用される場合は、必ず同じ変更を行います。

1. Forte for Java のエクスプローラで、RestaurantDetail クラスの rating フィールドを選択します。

2. rating フィールドのプロパティシートを開きます。

   「プロパティ」ウィンドウが開いている場合は、フィールドを選択します。

   「プロパティ」ウィンドウが開いていない場合は、フィールドを選択してコンテキストメニューで「プロパティ」を選択します。

3. 「種類」に java.lang.Integer と指定します。

   このフィールドの型が、テキストエディタ内で自動的に変更されます。

4. getRating メソッドと setRating メソッドのデータ型を次のように変更します。

```
public java.lang.Integer getRating() {
    return rating;
}

public void setRating(java.lang.Integer rating) {
    this.rating = rating;
}
```

5. 最後に、readObject メソッドのデータ型を次のように変更します。

```
private void readObject(java.io.ObjectInputStream in) throws
java.io.Exception, ClassNotFoundException {
    restaurantname = (String)in.readObject();
    cuisine = (String)in.readObject();
    neighborhood = (String)in.readObject();
    address = (String)in.readObject();
    phone = (String)in.readObject();
    description = (String)in.readObject();
    rating = (java.lang.Integer)in.readObject();
}
```

コンストラクタの作成

クラスフィールドをインスタンス化する詳細クラスのコンストラクタを作成するには、次のようにします。

1. RestaurantDetail Bean を展開して `RestaurantDetail` クラスノードを右クリックし、「新規」>「コンストラクタ」を選択します。

   「新規コンストラクタを編集」ウィザードが表示されます。

2. 次のメソッドパラメータを追加して「了解」をクリックします。

   `java.lang.String restaurantname`
   `java.lang.String cuisine`
   `java.lang.String neighborhood`
   `java.lang.String address`
   `java.lang.String phone`
   `java.lang.String description`
   `java.lang.Integer rating`

3. フィールドを初期化するために、`RestaurantDetail` コンストラクタの本体に次のコードを追加します。

```
public RestaurantDetail(java.lang.String restaurantname,
java.lang.String cuisine, java.lang.String neighborhood,
java.lang.String address, java.lang.String phone,
java.lang.String description, java.lang.Integer rating){
    System.out.println("Creating new RestaurantDetail");
    this.restaurantname = restaurantname;
    this.cuisine = cuisine;
    this.neighborhood = neighborhood;
    this.address = address;
    this.phone = phone;
    this.description = description;
    this.rating = rating;
}
```

4. 同様に、次のパラメータを使用して CustomerreviewDetail クラスにコンストラクタを追加します。

   `java.lang.String restaurantname`
   `java.lang.String customername`
   `java.lang.String review`

5. フィールドを初期化するために、CustomerreviewDetail コンストラクタの本体に次のコードを追加します。

```
public RestaurantDetail(java.lang.String restaurantname,
java.lang.String customername, java.lang.String review){
    System.out.println("Creating new CustomerreviewDetail");
    this.restaurantname = restaurantname;
    this.customername = customername;
    this.review = review;
}
```

6. Data パッケージを右クリックし、「すべてをコンパイル」を選択します。

   パッケージが、エラーなしでコンパイルされます。

   次に、エンティティ Bean に get メソッドを作成し、詳細クラスのインスタンスを取得します。

エンティティ Bean の get メソッドの作成

対応する詳細クラスのインスタンスを返す、各エンティティ Bean にメソッドを作成します。

1. Forte for Java のエクスプローラで、論理ノード Restaurant(EJB) を右クリックし、「新規ビジネスメソッド」を選択します。
2. 「新規ビジネスメソッド」ダイアログの「名前」フィールドに、「getRestaurantDetail」と入力します。
3. 戻り値の型として、「ブラウズ...」ボタンを使用して RestaurantDetail クラスを選択します。

   「戻り型」フィールドに、Data.RestaurantDetail と表示されます。
4. 「了解」をクリックしてメソッドを作成します。
5. テキストエディタでメソッドを開き、次のコードを追加します。

```
public Data.RestaurantDetail getRestaurantDetail() {
    return (new RestaurantDetail(restaurantname, cuisine,
neighborhood, address, phone, description, rating));
}
```

6. 同様に、次の値を使用して、Customerreview エンティティ Bean にビジネスメソッドを追加します。

   名前 : **getCustomerreviewDetail**
   戻り型 : **Data.CustomerreviewDetail**

7. テキストエディタでメソッドを開き、次のコードを追加します。

```
public Data.CustomerreviewDetail getCustomerreviewDetail() {
    return (new CustomerreviewDetail(restaurantname,
customername, review));
}
```

8. Data パッケージを右クリックし、「すべてをコンパイル」を選択します。

   パッケージ全体がエラーなしでコンパイルされます。

   チュートリアルアプリケーションのエンティティ Bean、および詳細クラスヘルパーの作成は完了です。次に、Bean をテストします。

エンティティ Bean のテスト

Forte for Java IDE には、独自のクライアントを作成せずにエンティティ Bean をテストするための機能があります。この機能では、J2EE リファレンス実装 (Reference Implementation: RI) をアプリケーションサーバーとして使用し、JavaServer Pages 技術を使用するアプリケーションの一部としてエンティティ Bean を配備します。また、Bean のインスタンスの作成、create メソッドとビジネスメソッドの実行が可能なページを Web ブラウザで参照できる形式で作成します。

この機能を使用し、Restaurant Bean の create メソッドと getRating メソッドを実行します。

注 - テストクライアント用に生成された SQL 文は、リファレンス実装 (RI) のプロトコルに制約されます。つまり、すべての表名と列名は引用符で囲まれ、表名には EJBTable が追加されます。テストアプリケーション内で Restaurant という名前の表を検索するように Bean インスタンスに要求すると、アプリケーションでは、RI プロトコルによって、Bean インスタンスが RestaurantEJBTable という名前の表を検索します。これは、`rest_pb.sql` が、このような表を作成しているためです (3 ページの「チュートリアルデータベース表の作成」を参照)。

エンティティ Bean のテストクライアントの作成

最初に、Restaurant Bean のリファレンス実装プロパティに PointBase 情報を追加し、テストアプリケーションでデータベースを検出しログインできるようにします。次に、テストクライアントを作成します。

1. **論理ノード `Restaurant` を選択し、「プロパティ」ウィンドウを開きます。**

   「プロパティ」ウィンドウが表示されていない場合は、`Restaurant(EJB)` を右クリックし、コンテキストメニューで「プロパティ」を選択します。「プロパティ」ウィンドウをそのまま表示しておく場合は、「表示」>「プロパティウィンドウ」を選択します。

2. **「プロパティ」ウィンドウの「J2EE RI」タブを選択します。**

注 - 「プロパティ」ウィンドウに「J2EE RI」タブがない場合は、サーバーレジストリのリファレンス実装にインスタンスがないことが考えられます。12 ページの「リファレンス実装を実行するように IDE を設定 する」を参照してください。

---

3. **プロパティに次の値を入力し、データベース接続を指定します。**

   データソースの JNDI 名: **`jdbc/Pointbase`**
   データソースのパスワード: **`PUBLIC`**
   データソースのユーザー名: **`PUBLIC`**

注 - 「Pointbase」の綴りは、必ず先頭を大文字にしてください。これは、*J2EE_HOME*/config/default.properties ファイルで指定されており、このプロパティと正確に一致する必要があるためです。また、 パスワードを入力した後は、 必ず Enter キーを押してください。マウスを使用してカーソルを他のフィールドへ移動させてしまうと、入力したパスワードは設定されません (このことは、「データソースのパスワード」フィールドでのみ適用されます)。

---

4. 次のプロパティに「false」を選択し、表の作成と削除を自動的に行う機能を無効にします。

   配備時の表の作成: **false**
   配備を元に戻すときに表の削除: **false**

   3 ページの「チュートリアルデータベース表の作成」 では、データベーススクリプトを使用して「*Tablename*EJBTable」という表を作成しました。アプリケーションを配備したり元に戻したりすると、RI は別の方法で表を作成および削除します。これらのフィールドを true に設定すると、配備するたびに新しい表データを作成しなければなりません。

   「J2EE RI」タブの他のプロパティについて調べる場合は、このタブがアクティブになっている状態で「Help」キーを押し、状況依存ヘルプを参照します。

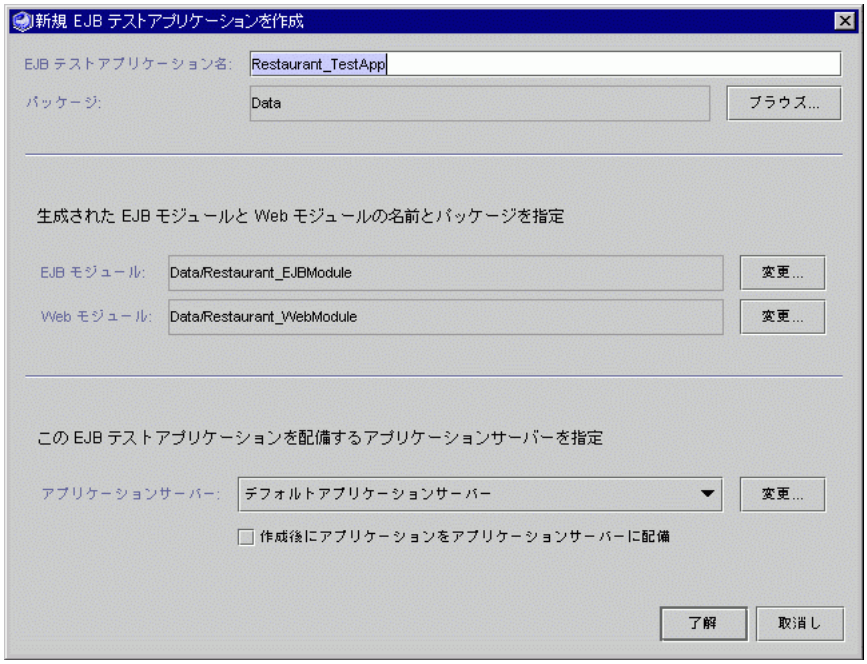
5. 「ファイル」>「すべてを保存」を選択し、作業内容を保存します。

6. 論理ノード Restaurant を右クリックし、「新規 EJB テストアプリケーション」を選択します。

   「新規 EJB テストアプリケーションを作成」ウィザードが表示されます。ウィザードの状況依存ヘルプを参照するには、「Help」キーを押します。

7. デフォルト値を適用します。

   ウィザードのウィンドウは、次のように表示されます。

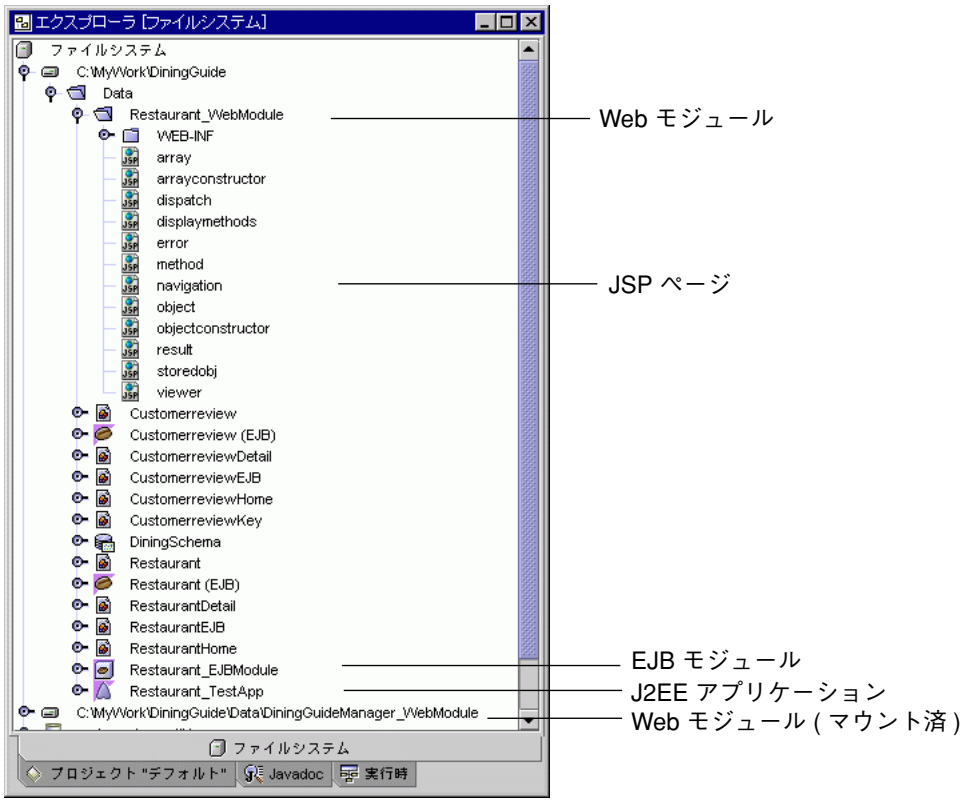


8. 「了解」をクリックします。

   進捗を表すモニターが表示されます。

9. エクスプローラに、テストオブジェクトが表示されます。

   テストアプリケーションによって、(`Restaurant` エンティティ Bean が含まれている) EJB モジュール、(多数の JSP ページとヘルパー Java クラスが含まれている) Web モジュール、(EJB モジュールと Web モジュールが含まれている) J2EE アプリケーションが作成されます。



IDE で作成された J2EE アプリケーションは、Web モジュールと EJB モジュールで構成されます。これは、`Restaurant_TestApp` を開いて確認することができます。

10. エクスプローラで EJBモジュール (`Restaurant_EJBModule`) を選択し、プロパティシートを表示します。

    「プロパティ」ウィンドウが表示されていない場合は、「表示」>「プロパティ」を選択します。

11. 「トランザクション設定」入力フィールドの「...」ボタンをクリックします。

    対象プロパティが下の方にある場合は、プロパティリストを下へスクロールします。

12. 「トランザクション属性」の値が「Required」になっていることを確認し、「了解」をクリックします。

13. `Restaurant_TestApp` J2EE アプリケーションノードを右クリックし、コンテキストメニューで「配備」を選択します。

    IDE の出力ウィンドウの「RI アプリケーションの配備」タブで、配備の進捗を確認することができます。ここで、`Restaurant_TestApp` というWeb クライアントが配備されたことが示されます。これは、`Restaurant` エンティティ Bean をテストするために使用するクライアントです。

    アプリケーションが配備されたことを確認するには、J2EE コマンドウィンドウを開きます。正常に配備された場合は、最後に「`Application Restaurant_TestApp deployed`」という文が表示されます。

参照 - 配備が失敗した場合は、J2EE リファレンス実装を正しく実行できるように IDE が設定されていることを確認します。具体的には、RIHome プロパティが *J2EE_HOME* の値に設定されているかどうかを確認します。12 ページの「リファレンス実装を実行するように IDE を設定 する」 の手順 4 を参照してください。正しく設定した後で、もう一度配備します。

次に、`Restaurant` エンティティ Bean をテストします。

テストクライアントによる エンティティ Bean のテスト

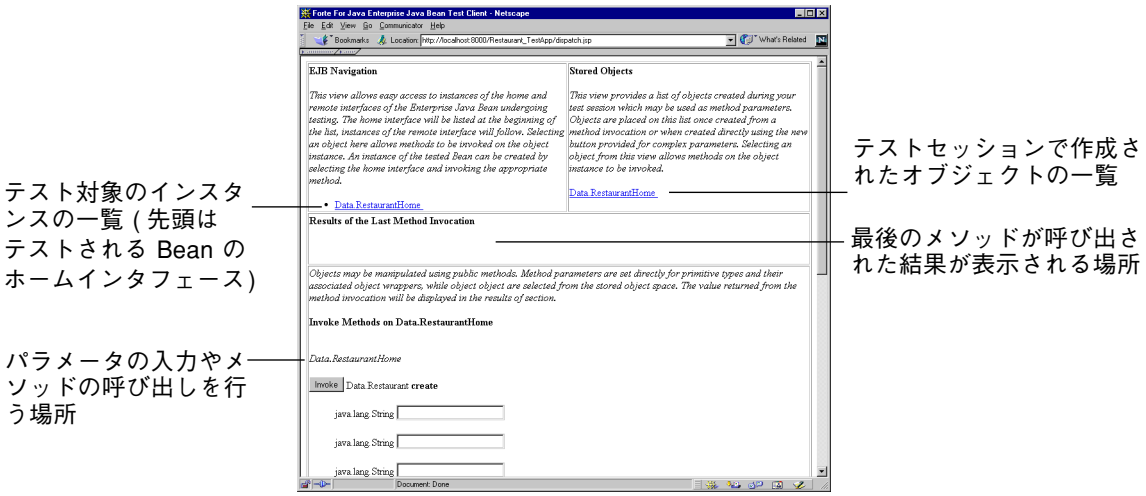
テストクライアントの Web ページで、`Restaurant` Bean のホームインタフェースの `create` メソッドを使用して Bean のインスタンスを作成し、次に、そのインスタンスでビジネスメソッド (この場合は `getRating`) をテストします。

Web ブラウザでテストアプリケーションを指定し、アプリケーションを開始します。

1. オペレーティングシステムで Web ブラウザを開き、次の URL を指定します。

   `http://localhost:8000/Restaurant_TestApp`

   ブラウザにテストクライアントが表示されます。



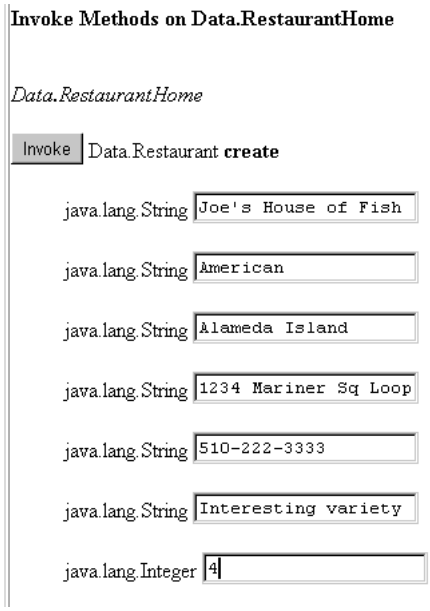
2. `create` メソッドを呼び出して `Restaurant` Bean のインスタンスを作成します。

   「Invoke Methods on Data.RestaurantHome」のタイトルの下には、`create` メソッドと 7 つのフィールドが示されます。これらのフィールドには名前が定義されていませんが、作成した順序で表示されるため、内容を推測することができます (45 ページの「CMP エンティティ Bean に対する生成メソッドの作成」の手順 3 を参照)。

注 - パラメータの内容を簡単に参照するには、テキストエディタで `Restaurant.create` メソッドの定義を検索します。7 つのフィールドは、ここで示される定義と同じ順序になっています。

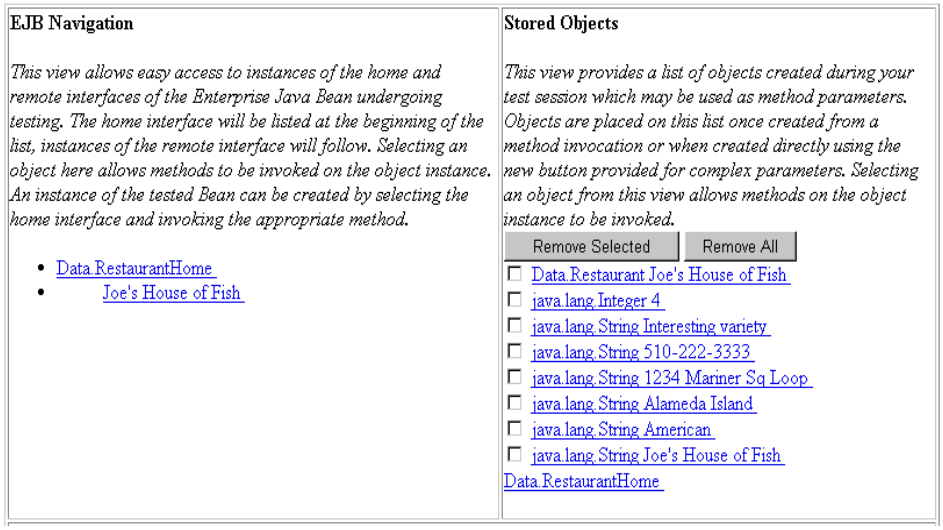
参照 - パラメータを別の順序で表示する場合は、エクスプローラで `Restaurant.create` メソッドノードを右クリックし、「カスタマイズ」を選択します。「カスタマイザ」ダイアログで「上へ」ボタンと「下へ」ボタンをクリックして選択し、パラメータの順序を調整します。その後で、エクスプローラでノードを右クリックして「配備」を選択し、テストアプリケーションを再配備します。

フィールドに任意のデータを入力します (フィールドの順序は異なる場合があります)。

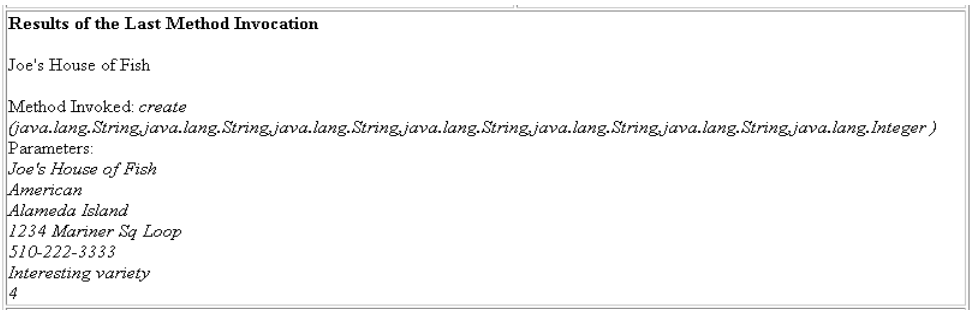


3. `create` メソッドの「Invoke」ボタンをクリックします。

   テストアプリケーションが配備されると、作成したレコードがテストデータベースに追加されます。次のように、新しい `Restaurant` インスタンスの `restaurantname` の値が左上に表示され、データオブジェクトが右上に示されます。



結果は、「Results」領域に表示されます。



4. 「Invoke」ボタンをクリックし、 `Restaurant` Bean の `findAll` メソッドをテストします。

   結果の領域には、次のように表示されます。

   ここでは、3 つの内容が返されます。これは、手順 3 で作成した新しいデータベースレコードが、第 1 章で作成した 2 つの Bean に追加されたことを表しています。

5. `Bay Fox` と入力してメソッドの「Invoke」ボタンをクリックし、`findByPrimaryKey` メソッドをテストします。

   結果の領域には、`Bay Fox` レコードが返されたことが示されます。

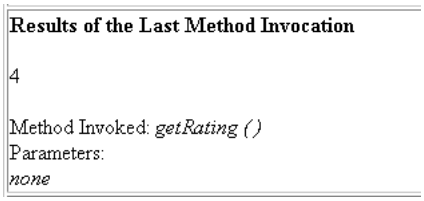
   次に、エンティティ Bean のビジネスメソッドをテストします。

6. (左上の) インスタンス一覧で `Data.RestaurantHome` の下に表示されている「Joe's House of Fish」のインスタンスを選択します。

   「Invoke Methods」領域に、`getRating` メソッドが表示されます。

7. getRating メソッドの「Invoke」ボタンをクリックします。

   この操作の結果は、「Results」領域に次のように表示されます。



   これは、データベースに新しいレコードが作成され、getRating メソッドを使用してフィールドのいずれかの値が取得されたことを表しています。

   作成されたオブジェクトを選択し、そのメソッドを呼び出してテストを続行します。たとえば、Data.RestaurantDetail オブジェクトのいずれかの値を選択すると、getter メソッドを呼び出してデータを表示することも、setter メソッドを呼び出してデータベースに新しいデータを書き込むこともできます。

8. テストが終了した後で、Web ブラウザで別の URL を指定するかブラウザを終了し、テストクライアントを停止することができます。

   Restaurant_TestApp と Customerreview_TestApp は、第 4 章で使用します。

9. Customerreview エンティティ Bean のテストクライアントを作成し、Customerreview Bean について、60 ページの「エンティティ Bean のテストクライアントの作成」から始まるすべての手順を再実行してテストします。

注 - これらの手順では繰り返しが多くなりますが、Customerreview_EJBModule は第 4 章で必要になるため、この手順を省略することはできません。

テストアプリケーションを削除する場合は、最初に Web アプリケーションをマウント解除した後で、Data パッケージの J2EE アプリケーション、Web モジュール、EJB モジュールを削除します。

これでセッション Bean を作成する準備が完了しました。

EJB ビルダーによるセッション Bean の作成

ステートレスなセッション Bean を作成し、クライアントとエンティティ Bean とのやりとりを管理します。

1. Forte for Java エクスプローラで Data パッケージを右クリックし、「新規」>「EJB」>「Session Bean」を選択します。

   ウィザードが表示され、「セッション EJB 型」区画が示されます。

2. 次の値を設定します。

   状態 : ステートレス
   トランザクション型 : コンテナ管理によるトランザクション
   Transparent Persistence と共に使用 : 選択しない

3. 「次へ」をクリックします。

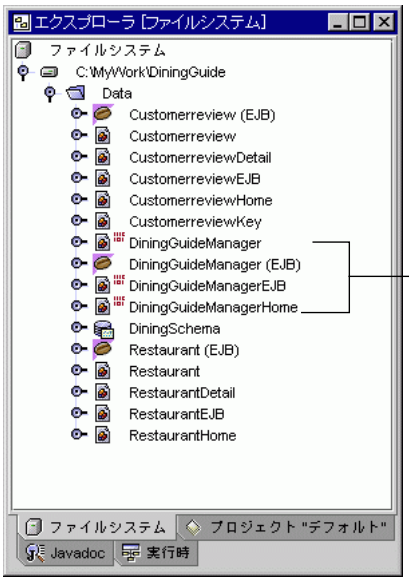
   ウィザードの「EJB コンポーネント」区画が表示され、このセッション Bean に作成されるすべてのコンポーネントが示されます。

4. 「名前」フィールドに `DiningGuideManager` と指定します。

   この名前を入力すると、コンポーネントの名前がすべて変更されます。

5. 「完了」をクリックします。

   エクスプローラに、新しい `DiningGuideManager` セッション Bean が表示されます。



セッション Bean の `create` メソッドの作成

`DiningGuideManager` セッション Bean を作成したときに、`create` メソッドも作成されているため、ここでは変更するだけになります。

セッション Bean は初期化に必要な途中の状態を保持しないため、ステートレスなセッション Bean の `create` メソッドは引数をとりません。このセッション Bean の `create` メソッドは、`InitialContext` を作成することによって開始されます。`create` メソッドは、後で `InitialContext` を使用して必要なリモート参照を取得します。

1. テキストエディタで、`DiningGuideManager` の `create` メソッドを開きます。

2. JNDI で RestaurantHome インタフェースに対するリモート参照を検索するメソッドをコーディングします。

```
public void ejbCreate(){
    System.out.println("Entering DiningGuideManagerEJB.ejbCreate()");
    Context c = null;
    Object result = null;

    if (this.myRestaurantHome == null) {
        try {
            c = new InitialContext();
            result = c.lookup("Restaurant");
            myRestaurantHome
=(RestaurantHome)javax.rmi.PortableRemoteObject.narrow (result,
RestaurantHome.class);
        }
        catch (Exception e) {System.out.println("Error: "+ e); }
    }
```

3. 上記のコードの下に、CustomerreviewHome インタフェースに対する類似の JNDI 検索を追加します。

```
    Context crc = null;
    Object crresult = null;

    if (this.myCustomerreviewHome == null) {
        try {
            crc = new InitialContext();
            crresult = crc.lookup("Customerreview");
            myCustomerreviewHome
=(CustomerreviewHome)javax.rmi.PortableRemoteObject.narrow(crresult,
CustomerreviewHome.class);
        }
        catch (Exception e) {System.out.println("Error: "+ e); }
    }
```

4. 次に、javax.naming パッケージの import 文を追加します。

   この文は、ファイルの先頭にある他の import 文の下に追加します。javax.naming には、使用したlookup メソッドが含まれているため、必ずインポートします。

```
import javax.ejb.CreateException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.ejb.SessionSynchronization;
import javax.naming.*;
```

5. 最後に、myRestaurantHome と myCustomerreviewHome フィールドを宣言します。

   これらの宣言は、前述の ejbCreate メソッドの上の DiningGuideManagerEJB セッション Bean の記述に追加します。

```
public class DiningGuideManagerEJB implements SessionBean {
    private SessionContext context;
    private RestaurantHome myRestaurantHome;
    private CustomerreviewHome myCustomerreviewHome;
```

次に、DiningGuideManager のビジネスメソッドを作成します。

ビジネスメソッドを作成して詳細データを取得する

DiningGuideManager Bean では、「レストランの一覧を参照する」というクライアントの要求を受け取ったときに、Restaurant のすべてのデータを取得するメソッドが必要になります。また、クライアントから顧客評価の一覧が要求された場合には、特定のレストランについて Customerreview データを取得する別のメソッドも必要になります。これらの要求に対処するために、getAllRestaurants メソッドおよび getCustomerreviewsByRestaurant メソッドを作成します。

1. 論理ノード DiningGuideManager を右クリックし、「新規ビジネスメソッド」を選択します。

2. 「新規ビジネスメソッド」ダイアログで、次の値を入力します。

   名前 : **getAllRestaurants**
   戻り型 : **java.util.Vector**

3. 「了解」をクリックします。

   DiningGuideManager セッション Bean のビジネスメソッドにメソッドシェルが作成されます。

4. テキストエディタでメソッドを開き、次のコードを追加します。

```
public java.util.Vector getAllRestaurants() {
    System.out.println("Entering DiningGuideManagerEJB.getAllRestaurants()");
    java.util.Vector restaurantList = new java.util.Vector();
    try {
        java.util.Collection rl = myRestaurantHome.findAll();
        if (rl == null) { restaurantList = null; }
        else {
            RestaurantDetail rd;
            java.util.Iterator rli = rl.iterator();
            while ( rli.hasNext() ) {
                rd =((Restaurant)rli.next()).getRestaurantDetail();
                System.out.println(rd.getRestaurantname());
                System.out.println(rd.getRating());
                restaurantList.addElement(rd);
            }
        }
    } catch (Exception e) {
        System.out.println("Error in DiningGuideManagerEJB.getAllRestaurants():
" + e);
    }
    System.out.println("Leaving DiningGuideManagerEJB.getAllRestaurants()");
    return restaurantList;
}
```

   このコードでは、Restaurant Bean の各リモート参照に対して RestaurantDetail のインスタンスを取得し、それを restaurantList というベクトルに追加し、このベクトルを返します。

   次に、顧客評価の一覧を取得するための類似のメソッドを作成します。

5. 論理ノード DiningGuideManager を右クリックし、「新規ビジネスメソッド」を選択します。

6. 「新規ビジネスメソッド」ダイアログで、次の値を入力します。

   名前 : **getCustomerreviewsByRestaurant**
   戻り型 : **java.util.Vector**

7. メソッドのパラメータを追加します。

   フィールド名 : **restaurantname**
   型 : **java.lang.String**

8. 「了解」をクリックします。

   DiningGuideManager セッション Bean にメソッドシェルが作成されます。

9. テキストエディタでメソッドを検索し、次のコードを追加します。

```
public java.util.Vector getCustomerreviewsByRestaurant(java.lang.String
restaurantname) {
    System.out.println("Entering
DiningGuideManagerEJB.getCustomerreviewsByRestaurant()");
    java.util.Vector reviewList = new java.util.Vector();
    try {
        java.util.Collection rl =
myCustomerreviewHome.findByRestaurantName(restaurantname);
        if (rl == null) { reviewList = null; }
        else {
            CustomerreviewDetail crd;
            java.util.Iterator rli = rl.iterator();
            while ( rli.hasNext() ) {
                crd =((Customerreview)rli.next()).getCustomerreviewDetail();
                System.out.println(crd.getRestaurantname());
                System.out.println(crd.getCustomername());
                System.out.println(crd.getReview());
                reviewList.addElement(crd);
            }
        }
    } catch (Exception e) {
        System.out.println("Error in
DiningGuideManagerEJB.getCustomerreviewsByRestaurant(): " + e);
    }
    System.out.println("Leaving
DiningGuideManagerEJB.getCustomerreviewsByRestaurant()");
    return reviewList;
}
```

getAllRestaurants のコードと同様に、このメソッドも、Customerreview Bean の各リモート参照に対して CustomerreviewDetail のインスタンスを取得し、それを reviewList というベクトルに追加し、そのベクトルを返します。

EJB 参照の追加

セッション Bean を配備する場合には、その Bean のプロパティに、エンティティ Bean (セッション Bean が呼び出すメソッドを持つエンティティ Bean) に対する参照が含まれるようにします。いったん Bean が EJB モジュールにアセンブルされると参照を追加することはできないため、ここでセッション Bean に追加します。

1. Forte for Java のエクスプローラで、`DiningGuideManager` Bean の論理ノードを選択します。
2. Bean のプロパティシートが表示されます。

   「プロパティ」ウィンドウが表示されていない場合は、Bean を右クリックし、コンテキストメニューで「プロパティ」を選択します。
3. 「プロパティ」ウィンドウの「参照」タブをクリックします。
4. 「EJB 参照」フィールドをクリックし、ブラウズ「...」ボタンをクリックします。

   「EJB 参照」ダイアログが表示されます。
5. 「追加」ボタンをクリックします。

   「追加 EJB 参照」ダイアログが表示されます。
6. `Restaurant` エンティティ Bean 用に次の値を指定し、「了解」をクリックします。

   参照名 : **`Restaurant`**
   説明 : (省略可)
   参照される EJB 名 : **`Restaurant`**
   型 : **`Entity`**
   ホームインタフェース : **`Data.RestaurantHome`**
   リモートインタフェース : **`Data.Restaurant`**

参照 - ブラウズ「...」ボタンを使用して「参照される EJB 名」を検索すると、「ホームインタフェース」と「リモートインタフェース」フィールドには、値が自動的に挿入されます。

---

7. 同様に、Customerreview エンティティ Bean に参照を追加します。

「EJB 参照」ダイアログは、次のように表示されます。

8. 「ファイル」>「すべてを保存」を選択します。

これで、チュートリアルアプリケーションの EJB 層が完成し、テストの準備ができました。エンティティ Bean をテストする際には、IDE のテストアプリケーション機能によって、Web 層と JSP ページが作成されます。JSP ページは、クライアントがブラウザで参照することができます。

セッション Bean のテスト

DiningGuideManager セッション Bean をテストするには、IDE のテストアプリケーション機能を使用します。セッション Bean のメソッドは、同じ層の他のすべてのコンポーネント上のメソッドにアクセスできるため、この機能では、EJB 層全体をテストします。

セッション Bean のテストクライアントの作成

DiningGuideManager Bean からテストアプリケーションを作成します。次に、EJB モジュールに 2 つのエンティティ Bean を追加します。

注 - PointBase 情報が、Restaurant および Customerreview Bean の J2EE RI プロパティに追加されていることを確認してください。詳細は、60 ページの「エンティティ Bean のテストクライアントの作成」の手順 1 から 手順 5 までを参照してください。

---

1. 論理ノード DiningGuideManager を右クリックし、「新規 EJB テストアプリケーション」を選択します。

   「新規 EJB テストアプリケーションを作成」ウィザードが表示されます。

2. デフォルト値をすべて適用し、「了解」をクリックします。

   進捗を表すモニターが表示されます。

3. エクスプローラに、テストオブジェクトが表示されます。

   テストアプリケーションによって、DiningGuideManager_EJBModule、DiningGuideManager_WebModule (マウント済)、および DiningGuideManager_TestApp (EJB モジュールと Web モジュールが含まれている J2EE アプリケーション) が作成されます。

   EJB モジュールには DiningGuideManager Bean しか含まれていないため、2 つのエンティティ Bean を追加します。

4. EJB モジュール (DiningGuideManager_EJBModule) を右クリックし、「EJB を追加」を選択します。

   「EJB を EJB モジュールに追加」ウィンドウが表示されます。

5. Restaurant エンティティ Bean を選択し、「了解」をクリックします。

6. Customerreview エンティティ Bean に対し、手順 4 と手順 5 を繰り返します。

   DiningGuideManager_EJBModule は、次のように表示されます。

7. 「ファイル」>「すべてを保存」を選択します。

8. DiningGuideManager_TestApp J2EE アプリケーションノードを右クリックし、コンテキストメニューで「配備」を選択します。

   IDE の出力ウィンドウの「RI アプリケーションの配備」タブで、配備の進捗を確認することができます。ここで、DiningGuideManager_TestApp というWeb クライアントが配備されたことが示されます。

   アプリケーションが配備されたことを確認するには、J2EE コマンドウィンドウを開きます。正常に配備された場合は、最後に「Application DiningGuideManager_TestApp deployed」という文が表示されます。

   次に、DiningGuideManager セッション Bean をテストします。

テストクライアントによるセッション Bean のテスト

テストクライアントの Web ページで create メソッドを実行して DiningGuideManager セッション Bean のインスタンスを作成し、そのインスタンスでビジネスメソッド (getRating) をテストします。

Web ブラウザでテストアプリケーションを指定し、アプリケーションを開始します。

1. オペレーティングシステムで Web ブラウザを開き、次の URL を指定します。

   http://localhost:8000/DiningGuideManager_TestApp

   ブラウザに、テストクライアントとインスタンスリストが表示されます。左上のインスタンスリストには、DiningGuideManagerHome のインスタンスのみが示されています。

2. DiningGuideManagerHome の create メソッドを呼び出し、DiningGuideManager セッション Bean のインスタンスを作成します。

   インスタンスリストに Data.DiningGuideManager[*x*] インスタンスが表示されます。これで、Bean の getter メソッドをテストすることができます。

3. 新しい Data.DiningGuideManager[*x*] インスタンスを選択します。

   getAllRestaurants および getCustomerreviewsByRestaurant メソッドが有効になります。

4. `getAllRestaurants` メソッドの「Invoke」ボタンをクリックします。

   64 ページの「テストクライアントによる エンティティ Bean のテスト」で、データベースに「Joe's House of Fish」を作成した場合は、(右上に示される) 作成されたオブジェクトの一覧にサイズ 3 のベクトルが表示されます。メソッドを呼び出した結果は、次のように表示されます。このレコードを作成していない場合は、結果が多少異なります。

5. `getCustomerreviewsByRestaurant` メソッドのフィールドに「`Bay Fox`」と入力し、「Invoke」ボタンをクリックします。

   データベースには顧客の評価がないため、`CustomerreviewDetail` レコードは返されません (第 1 章の表 1-4 を参照)。次に、「French Lemon」レコードについて処理を行います。

6. 同じフィールドに `French Lemon` と入力し、メソッドを呼び出します。

   2 つの `CustomerreviewDetail` レコードが返されます。

7. テストが終了した後で、Web ブラウザで別の URL を指定するかブラウザを終了し、テストクライアントを停止することができます。

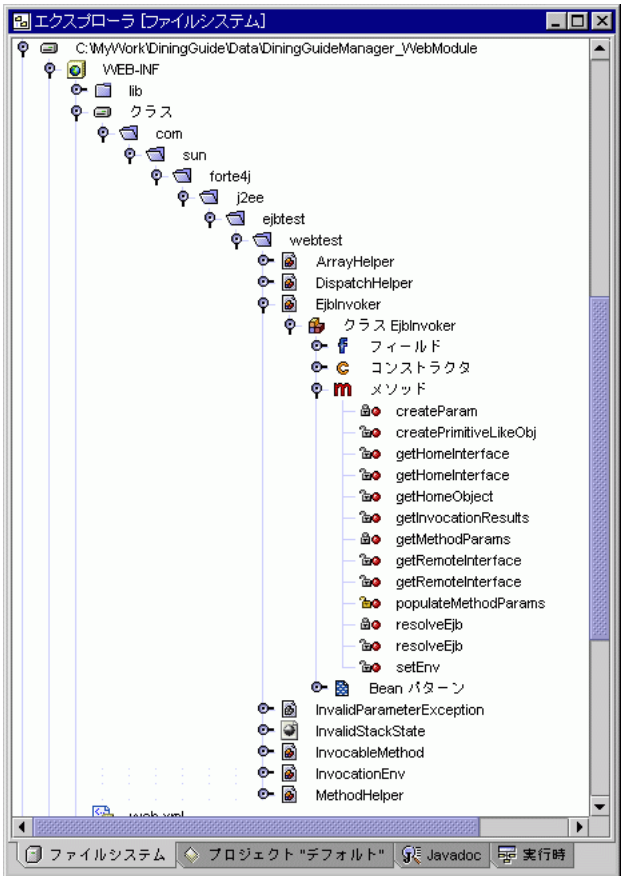
クライアントを作成する際の注意事項

これで、EatersDigest アプリケーションの EJB 層を正しく作成することができました。Forte for Java IDE の Web サービスモジュールによるアプリケーションの Web サービスの作成 (第 4 章)、および、Forte ESP ツールキットによるクライアントページの作成 (第 5 章) を行う準備ができました。

独自の Web サービスとクライアントを作成する場合は、Forte for Java テストアプリケーションにあるガイドラインを参照してください。

`DiningGuideManager` Bean のように、セッション Bean にアクセスする Web サービスには、lookup メソッドを持つサーブレットや JSP ページを定義しておく必要があります。これは、EJB 層で、エンティティ Bean のホームインタフェースとホームオブジェクトを取得するためです。テストアプリケーション機能によって作成された Web モジュールには、必要なコードのサンプルが用意されています。

lookup メソッドの例は、Web モジュールの `EjbInvoker` クラスで参照できます。このクラスの、`WEB-INF/Classes/com/sun/forte4j/j2ee/ejbtest/webtest` ディレクトリを参照してください。



たとえば、次のメソッドに lookup コードの例があります。

- `EjbInvoker.getHomeObject`
- `EjbInvoker.getHomeInterface`
- `EjbInvoker.resolveEjb`

第4章

Web サービスの作成

この章では、Forte for Java の Web サービスモジュール使用し、EatersDigest アプリケーションの Web サービスを作成する方法について学習します。

この章のトピックは、次のとおりです。

チュートリアルの Web サービスの概要

この章では、EatersDigest アプリケーションの EJB 層を Web サービスとして作成します。この手順の一部として、コンポーネントを明示的に作成し、いくつかのものを生成します。

ここでは、1 つの論理 Web サービスと、4 つの XML オペレーションを明示的に作成します。

- `DiningGuideWebService` Web サービス
- `AllRestaurantsXMO` XML オペレーション
- `AllCustomerreviewsByRestaurantNameXMO` XML オペレーション
- `EnterCustomerReviewXMO` XML オペレーション
- `AddCustomerReviewXMO` XML オペレーション

また、実行時クラスとして、各 XML オペレーションに対するクラスと Web サービスに対するクラスを生成します。

- `DiningGuideWebService.class`
- `AllRestaurantsXMO.class`
- `AllCustomerreviewsByRestaurantNameXMO.class`
- `EnterCustomerReviewXMO.class`
- `AddCustomerReviewXMO.class`

また、Web サービスをテストするためのクライアントページとして、4 つの JSP ページ (各 XML オペレーション用) と HTML の開始ページを生成します。

- `AllRestaurantsXMO.jsp`
- `AllCustomerreviewsByRestaurantNameXMO.jsp`
- `EnterCustomerReviewXMO.jsp`
- `AddCustomerReviewXMO.jsp`
- `DiningGuideWebService.html`

また、J2EE アプリケーションも明示的に作成します。このアプリケーションには、これまでの章で作成した Web サービスと EJB モジュールを追加します。

Web サービス

Web サービスおよびその作成方法とプログラミングについては、『Web サービスのプログラミング』を参照してください。

IDE で作成する Web サービスは論理エンティティであり、これを使用して Web サービスをプログラムします。Web サービスの機能を開発するには、XML オペレーションを作成し、それを論理 Web サービスに追加します。また、Web サービスで参照する必要のあるドキュメント (JSP ページ、HTML ファイル、イメージファイルなど) を追加します。

XML オペレーション

XML オペレーションおよびその作成方法とプログラミングについては、『Web コンポーネントのプログラミング』を参照してください。

XML オペレーションは論理エンティティであり、これを使用して Web サービスの応答メカニズムをプログラムします。作成する各 XML オペレーションは、特定のクライアント要求に対する応答を表します。Web サービスがクライアント要求を受け取ると、それに応答して 1 つの XML オペレーションが実行されます。

各 XML オペレーションが、EJB 層のオブジェクトにおいて、いくつかのメソッドを呼び出すようにプログラムします。このチュートリアルの Web サービスでは、XML オペレーションは、アプリケーションのエンティティ Bean においてメソッドを呼び出します。

XML オペレーションは、XML ドキュメントの形式 (XML 入力ドキュメント) のクライアント要求を受け付けます。XML オペレーションは要求を受け取ると、XML 入力ドキュメントの要素をパラメータ (XML オペレーションで呼び出すメソッドのパラメータ) にマップします。次に、XML オペレーションはメソッドを呼び出し、戻り値を XML ドキュメントとしてフォーマットし、ドキュメント (XML 出力ドキュメント) を返します。

ソースエディタを使用すると、XML ドキュメントやメソッドの呼び出しを手動でコーディングすることなく、XML オペレーションをプログラムすることができます。

実行時クラス

Web サービスのプログラミングが終了した後で、実行時クラスを生成します。実行時クラスで直接作業することはありませんが、実行時クラスは、XML オペレーションや論理 Web サービスが含まれているパッケージに生成されます。

クライアントページ

Web サービスの実行時クラスを生成すると、論理 Web サービスの `Documents` ディレクトリに、クライアントページも生成されます。これらのクライアントページは、Web サービスのテストで使用します。また、すべての機能を装備したクライアントを開発する際のひな型や参考として、クライアントページを使用することもできます。

XML オペレーションごとに JSP ページが 1 つ生成されます。Web サービス全体について、1 つの開始 HTML ページも生成されます。

開始ページには、生成された各 JSP ページに対応する HTML フォームが 1 つずつ含まれています。テストの際には、これらの HTML フォームを使用して対応する JSP ページを要求します。HTML フォームには入力フィールドが含まれています。これらの入力フィールドによって、パラメータを必要とする JSP ページに対してパラメータを提供します。この処理の説明は、第 5 章に記載されています。

開始ページから、これらの JSP ページのいずれかを要求すると、次の処理が行われます。

1. JSP ページが、対応する要求を Web サービスに送信します。XML ドキュメントの形式で要求を送信します。この形式は、Forte for Java で提供されている ESP タグライブラリを使用して作成されます。
2. Web サービスが、要求された XML オペレーションを実行します。
3. XML オペレーションは要求を処理し、XML 出力ドキュメントにデータを返します。
4. ESP タグライブラリは、XML 出力ドキュメントからデータを抽出し、JSP ページは、Web ブラウザに表示できるようにそのデータをフォーマットします。

XML オペレーションの作成

Web サービスを作成する前に、XML オペレーションを作成します。ここでは、アプリケーションの 2 つのエンティティ Bean のメソッドをベースとしたオペレーションを作成します。

- `AllRestaurantsXMO` — `Data.RestaurantHome.findAll` メソッドをベースとします。

  `Restaurant.getRestaurantDetail` も呼び出します。`RestaurantDetail` オブジェクトのコレクション (データベース内のすべてのレストランに対して 1 つのオブジェクト) を返します。

- AllCustomerreviewsByRestaurantNameXMO — Data.CustomerreviewHome.findByRestaurantName メソッドをベースとします。

  Customerreview.getCustomerreviewDetail も呼び出します。restaurantname パラメータを取得し、指定されたレストランに対する CustomerreviewDetail オブジェクトのコレクションを返します。

- AddCustomerReviewXMO — Data.CustomerreviewHome.create メソッドをベースとします。

  restaurantname、customername、review の入力データを取得し、CustomerreviewDetail オブジェクトを返します。

ここでは、4 つめの XML オペレーションも作成します。これは、Data パッケージ内のメソッドではなく、Web サービスのデザインユーティリティメソッドをベースとしています。このメソッドによって、アプリケーションで同じパラメータを渡したり、返したりすることができます。アプリケーションは、このオペレーションを使用し、レストランの名前を最初に Customer Review データエントリページに渡し、次にその名前を 3 つのパラメータと共に JSP ページに渡します。JSP ページでは、Customer Review をデータベースに追加します。この処理のための XML オペレーションは、次のとおりです。

- EnterCustomerReviewXMO — com.sun.forte4j.webdesigner.utilities.WebServicesDesignerUtilities.Identity メソッドをベースとします。

  element1 という文字列 (restaurantname フィールドで使用する文字列) を取得します。

エンティティ Bean のメソッドに基づいて XML オペレーションを作成する

DiningGuide ディレクトリに、WebService という名前のパッケージを作成します。次に、Web サービスウィザードを使用し、パッケージ内に 3 つのXML オペレーションを作成します。

WebService パッケージの作成

1. Forte for Java のエクスプローラで、マウントされている DiningGuide ファイルシステムを右クリックし、「新規パッケージ」を選択します。

   「新規パッケージを作成」ダイアログが表示されます。

2. WebService という名前を指定して「了解」をクリックします。

   DiningGuide ディレクトリに、新しい WebService パッケージが表示されます。

AllRestaurantsXMO オペレーションの作成

1. 新しい WebService パッケージを右クリックして、「新規」>「Web Services」>「XML Operation」を選択します。

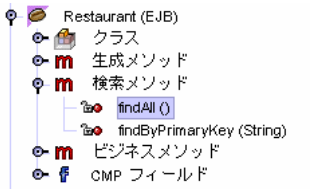
   「テンプレートウィザードから新規作成」の「XML オペレーション」区画が表示されます。

2. 次の値を入力します。

   - 名前 : **AllRestaurantsXMO**
   - パッケージ : **WebService**

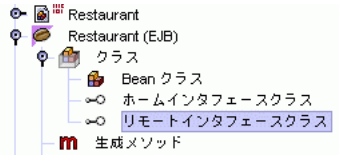
     これはデフォルトですが、この値が指定されていない場合は、「ブラウズ」ボタンを使用してパッケージを検索してください。
   - メソッド : **Data.RestaurantHome.findAll**

     「ブラウズ」ボタンを使用し、以下のように論理 Bean からメソッドを選択します。選択できない場合は、XMO エディタに正しく表示されません。

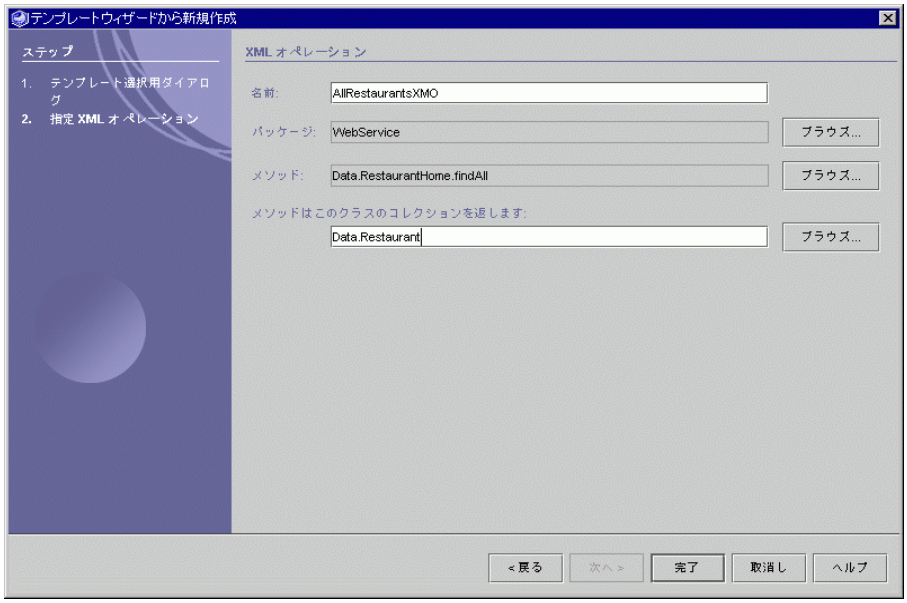


- メソッドはこのクラスのコレクションを返します : **`Data.Restaurant`**

  これは、`Restaurant` エンティティ Bean のリモートインタフェースです。前述のメソッドの場合には、「ブラウズ」ボタンを使用し、以下のように論理 Bean からリモートインタフェースを選択します。



ウィザードには次のように表示されます。

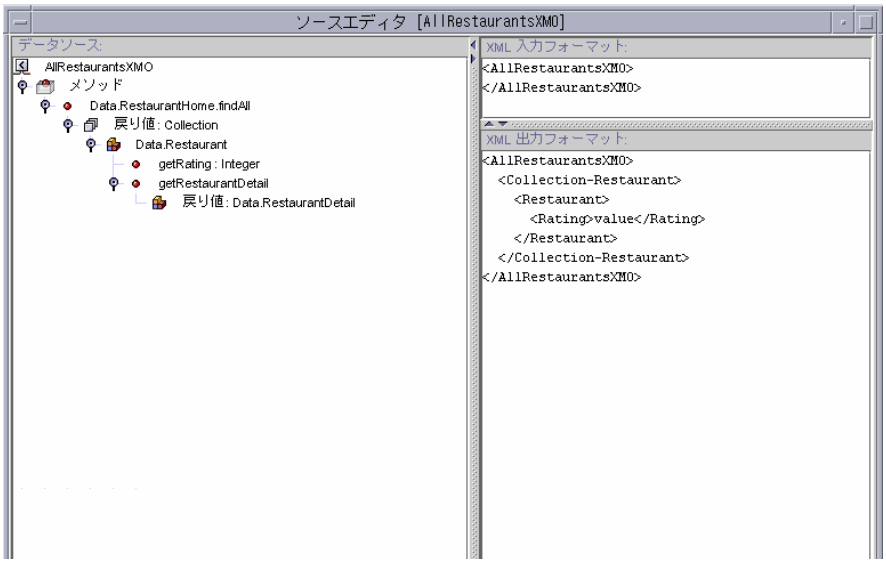


3. 「完了」をクリックします。

   新しい XML オペレーションノード、およびそのコンテンツがエクスプローラに表示され、Web サービスのソースエディタにオペレーションが表示されます。

4. Web サービスのソースエディタの「データソース」区画ですべてのノードを展開し、新しいオペレーションのコンテンツを表示します。

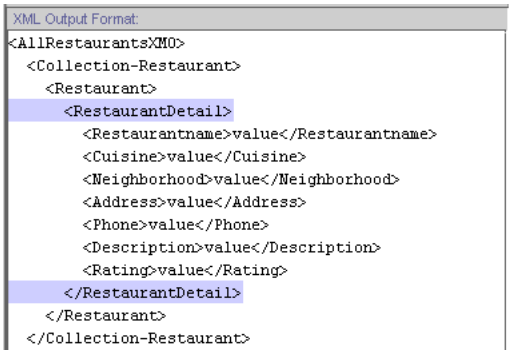
Web サービスのソースエディタには、次のように表示されます。



5. `getRating` メソッドを削除 (メソッドを右クリックして「削除」を選択) します。
6. (`getRestaurantDetail` の下の) 戻り値 : `Data.RestaurantDetail` ノードを右クリックし、「展開」を選択します。

   これによって、`RestaurantDetail` のすべてのフィールドがオペレーションに追加されます。

   ソースエディタの「XML 出力フォーマット」区画にフィールドが表示されます。ここで、フィールドのラベル文字が、Bean ではすべてが小文字だった状態から先頭だけが大文字に変わっていることに注意してください。このことは、後でコードで参照する場合に非常に重要になります。



AllCustomerreviewsByRestaurantNameXMO オペレーションの作成

1. 前の節で説明した手順 1 から手順 3 を使用し、次の値のXML オペレーションを作成します。

   名前: **AllCustomerreviewsByRestaurantNameXMO**
   パッケージ : **WebService**
   メソッド : **Data.CustomerreviewHome.findByRestaurantName**
   メソッドはこのクラスのコレクションを返します : **Data.Customerreview**

2. ソースエディタで、AllCustomerreviewsByRestaurantNameXMO オペレーションを開きます。

   このオペレーションは自動的に開く場合もあります。開かない場合は、エクスプローラでオペレーションをダブルクリックします。

3. ソースエディタの「データソース」区画で、メソッドの下のすべてのノードを展開します。

4. getReview メソッドを削除します。

5. 戻り値 : Data.CustomerreviewDetail を右クリックして「展開」を選択し、Customerreview フィールドをオペレーションに追加します。

AddCustomerReviewXMO オペレーションの作成

このオペレーションはコレクションを返さずに、以下の内容を作成します。

1. 他の手順と同様に、次の値の XML オペレーションを作成します。

   名前 : **AddCustomerReviewXMO**
   パッケージ : **WebService**
   メソッド : **Data.CustomerreviewHome.create**
   メソッドはこのクラスのコレクションを返します : (空白にしておきます)

2. ソースエディタで、AddCustomerReviewXMO オペレーションを開きます。

3. ソースエディタの「データソース」区画で、メソッドの下のノードを展開します。

4. getReview メソッドを削除します。

5. 戻り値 : Data.CustomerreviewDetail ノードは展開しないでください。

   このXML オペレーションではフィールドは必要ありません。

次に、最後の XML オペレーションを作成します。これは、作成したメソッドをベースとしていません。

他のメソッドをベースとして XML オペレーションを作成する

このチュートリアルでは、Web サービスライブラリのメソッドを使用して同じパラメータをやりとりします。最初に、必要なメソッドを持つ Bean が含まれているライブラリ JAR ファイルをマウントします。次に、前述の場合と同様に XML オペレーションを作成します。

Web サービスモジュールの JAR ファイルをマウントする

Web サービスモジュールから `webservices-util.jar` ファイルをマウントします。

1. Forte for Java のエクスプローラで「ファイルシステム」を右クリックし、「JAR アーカイブをマウント」を選択します。

   「JAR ファイルをマウント」ダイアログが表示されます。

2. *FORTE4J_HOME*`/modules` に移動します。

3. `webservices-util.jar` を選択します。

4. 「マウント」をクリックします。

   `webservices-util.jar` ファイルがエクスプローラにマウントされます。

`EnterCustomerReviewXMO` オペレーションの作成

`Identity` メソッドをベースとして XML オペレーションを作成します。

1. `WebService` パッケージを右クリックし、「新規」>「Web Services」>「XML Operation」を選択します。

2. ウィザードで、次の値を入力します。

   名前 : **`EnterCustomerReviewXMO`**
   パッケージ : **`WebService`**
   メソッド :
   `com.sun.forte4j.webdesigner.utilities.WebServicesDesignerUtilities.Identity`
   メソッドはこのクラスのコレクションを返します : (空白にしておきます)

Identity メソッドを選択すると、「メソッドの選択」ウィンドウが次のように表示されます。

3. 「完了」をクリックします。

   エクスプローラに、新しい EnterCustomerReviewXMO オペレーションが表示されます。

4. ソースエディタで、EnterCustomerReviewXMO オペレーションを開きます。

   このオペレーションは自動的に開く場合もあります。開かない場合は、エクスプローラでオペレーションをダブルクリックします。

5. ソースエディタの「データソース」区画で「メソッド」ノードを展開します。

6. Identity メソッドのプロパティを開きます。

   「プロパティ」ウィンドウが表示されていない場合は、メソッドを右クリックして「プロパティ」を選択します。

7. 「タグ名」の値を restaurantname に変更します。

   ソースエディタの「XML 出力フォーマット」区画で、フィールドの名前が変わっていることを確認します。

   これで XML オペレーションの作成は完了です。次に、Web サービスを作成します。

Web サービスの作成

DiningGuideWebService Web サービスを作成し、Web サービスに対する参照を、作成した XML オペレーションに追加します。

1. IDE のエクスプローラで WebService パッケージを右クリックし、「新規」>「Web Services」>「Web Service」を選択します。

   「Web サービス」区画が表示されます。

2. 次の値を入力します。

   名前 : **DiningGuideWebService**
   パッケージ : **WebService**

3. 「完了」をクリックします。

   エクスプローラの WebService パッケージの下に新しい DiningGuideWebService Web サービスが表示されます。

4. DiningGuideWebService ノードを右クリックし、「参照を追加」を選択します。

   ブラウズウィンドウに、マウントされたディレクトリが表示されます。
   DiningGuide/WebService ノードを開きます。

5. 作成した 4 つの XML オペレーションを選択します。

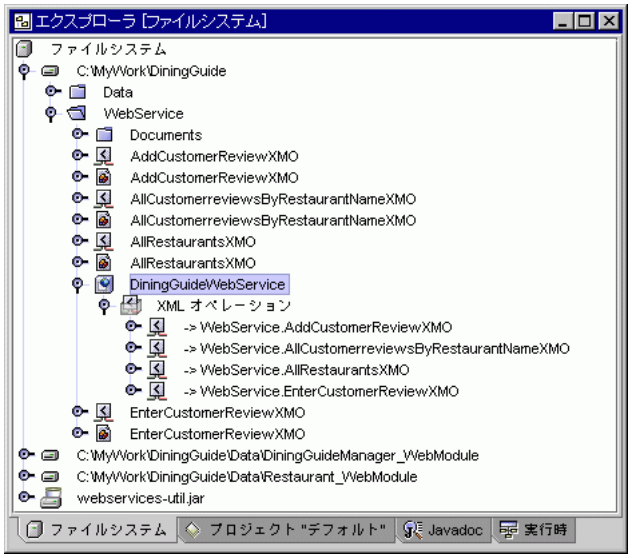
   Ctrl キーを押し、複数の項目を選択します。

6. 「了解」をクリックします。

   エクスプローラの DiningGuideWebService/XML オペレーションに 4 つのオペレーションが表示されます。ここで、WebService ファイルをコンパイルします。

7. DiningGuideWebService ノードを右クリックし、「Java ファイルの生成 / コンパイル」ファイルを選択します。

   オペレーションが完了すると、IDE の出力ウィンドウに「完了」と表示され、WebService ノードの Documents フォルダに、新しい項目が作成されます。



   これで Web サービスが作成され、J2EE アプリケーションを作成する準備が完了しました。

Web サービスの J2EE アプリケーションの作成

次に、EJB モジュールと Web サービスのモジュールが含まれる J2EE アプリケーションを作成します。アプリケーションに EatersDigestApp という名前を定義し、Data パッケージから DiningGuideWebService モジュール、Restaurant_EJBModule、Customerreview_EJBModule をアプリケーションに追加します。次に、アプリケーションに Web コンテキストを指定します。

1. IDE のエクスプローラで WebService パッケージを右クリックし、「新規」>「J2EE」>「Application」を選択します。

   ウィザードが表示されます。

2. 名前に EatersDigestApp と入力し、「完了」をクリックします。

   WebService ディレクトリに、新しい J2EE アプリケーションが表示されます。

3. EatersDigestApp アプリケーションを右クリックし、「モジュールを追加」を選択します。

   「アプリケーションにモジュールを追加」ウィンドウが表示されます。

4. DiningGuideWebService ノードを選択し、「了解」をクリックします。

5. EatersDigestApp アプリケーションを開きます。

   DiningGuideWebService のWAR ファイルと EJB JAR ファイルがアプリケーションに追加されています。

6. EatersDigestApp アプリケーションをもう一度右クリックし、「モジュールを追加」を選択します。

7. Data パッケージを開いて次のモジュールを選択し、「了解」をクリックします。

   Restaurant_EJBModule
   Customerreview_EJBModule
   (複数の項目を選択する場合は、Ctrl キーを押しながら項目をクリックします。)

注 - これらのモジュールが両方とも Data パッケージに存在しない場合は、モジュールを作成して追加する必要があります。60 ページの「エンティティ Bean のテストクライアントの作成」を参照してください。

次に、新しい J2EE アプリケーションに Web コンテキストを定義します。

8. EatersDigestApp アプリケーションの DiningGuideWebService_War ファイルのプロパティシートを開きます。

   「プロパティ」ウィンドウが表示されていない場合は、ファイルを右クリックして「プロパティ」を選択します。

9. 「Web コンテキスト」フィールドに EatersDigestContext と入力します。

10. 「ファイル」>「すべてを保存」を選択します。

    EatersDigestApp アプリケーションの作成が終了し、配備する準備ができました。

チュートリアルアプリケーションの配備

チュートリアルアプリケーションを配備する前に、IDE で J2EE リファレンス実装を実行できるように設定しておきます。詳細は、12 ページの「リファレンス実装を実行するように IDE を設定 する」を参照してください。

アプリケーションの配備では、テストアプリケーション機能によって配備した場合よりも長い時間がかかります (76 ページの「セッション Bean のテスト」を参照)。

- **IDE のエクスプローラで `EatersDigestApp` アプリケーションを右クリックし、「配備」を選択します。**

  進捗状況を示すモニターが表示され、オペレーションの進捗が示されます。

  IDE の出力ウィンドウの「RI アプリケーションの配備」タブで、配備の進捗を確認することができます。最後に、`EatersDigestApp` の配備が完了したことを表すメッセージが示されます。

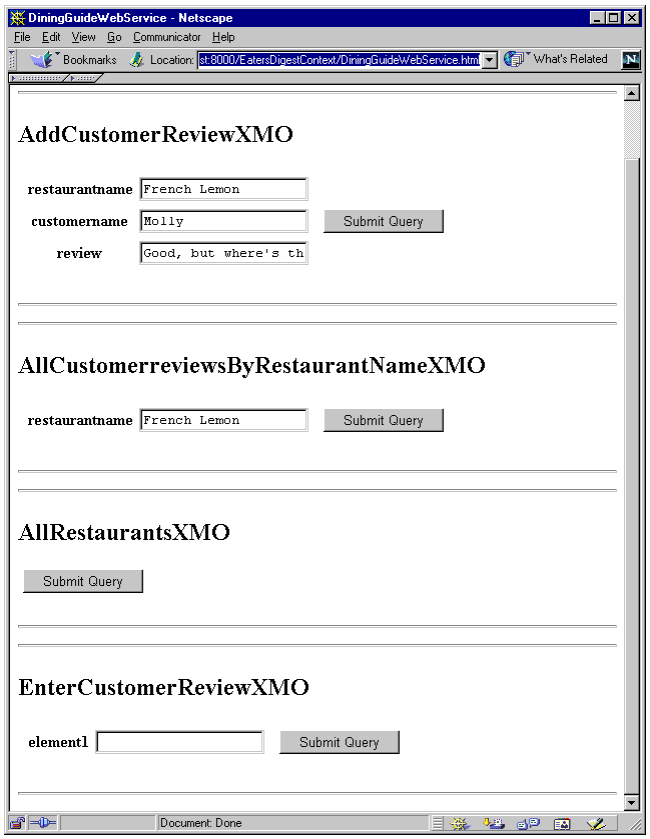
  アプリケーションが配備されたことを確認するには、J2EE コマンドウィンドウを開きます。正常に配備された場合は、最後に「`Application Restaurant_TestApp deployed`」という文が表示されます。

生成されたクライアントページのテスト

1. **オペレーティングシステムで Web ブラウザを開き、次の URL を指定します。**

   ```
   http://localhost:8000/EatersDigestContext
   ```

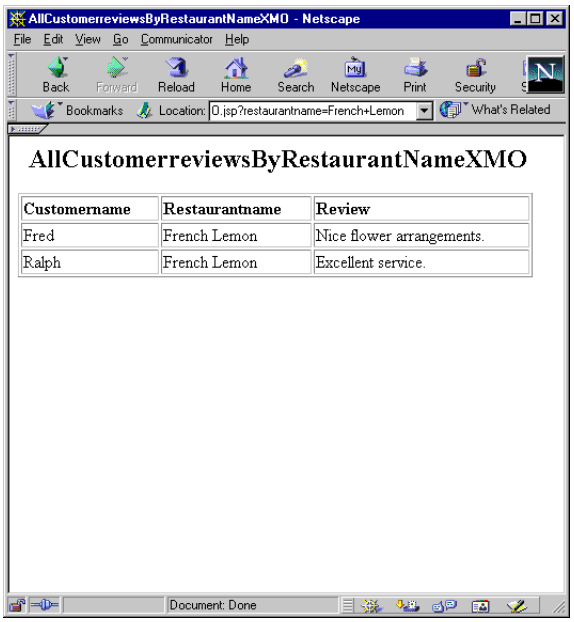
   ブラウザに、生成された開始ページが表示されます (エクスプローラでは、このページは、`Documents` フォルダの `DiningGuideWebService` HTML ページとなります)。85 ページの「クライアントページ」で説明したように、Web サービスのモジュールは、XML オペレーションに対してそれぞれ 1 つの JSP ページと開始ページを生成します。このページには HTML フォームが含まれており、これを使用して生成された各 JSP ページをテストします。EatersDigest チュートリアルの開始ページは、次のように表示されます。



このページで、各オペレーションが定義どおりに機能するかどうかテストすることができます。

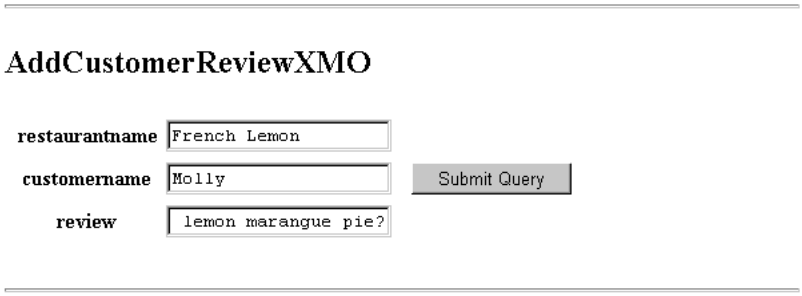
2. テキストフィールドに `French Lemon` と入力して「Submit Query」ボタンをクリックし、`AllCustomerreviewsByRestaurantNameXMO` をテストします。

アプリケーションは、データベースからデータを返し、このオペレーション用に生成された JSP ページにデータを次のように表示します。



3. 開始ページに戻るには、ブラウザの「戻る」ボタンを使用します。
4. `restaurantname` フィールドに `French Lemon` と入力し、他の 2 つのフィールドには任意の内容を入力します。`AddCustomerReviewXMO` オペレーションをテストします。

   たとえば、次のように入力します。

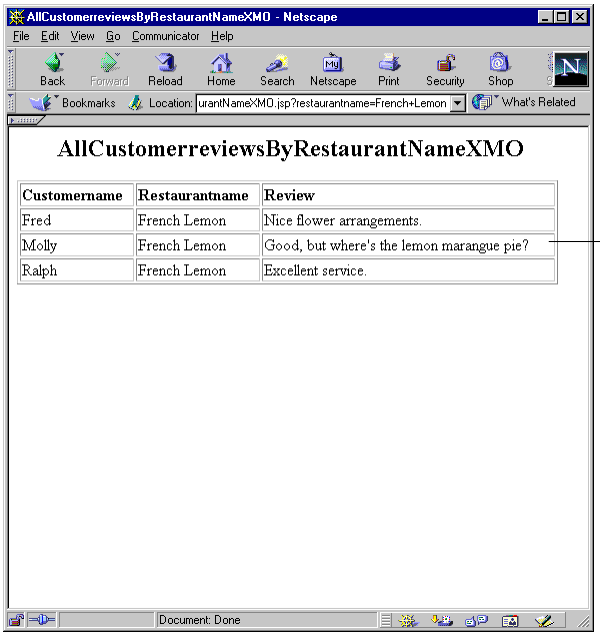


5. 「Submit Query」ボタンをクリックします。

   このオペレーションに対して生成されたページが、1 つのオペレーションラベル付きで表示されます (これについては、111 ページの「顧客評価の追加」で変更します)。データベースにレコードが書き込まれたことを確認するには、次の手順に進みます。

6. 開始ページに戻り、手順 2 を繰り返します (レストラン「French Lemon」についての顧客評価をすべて取得します)。

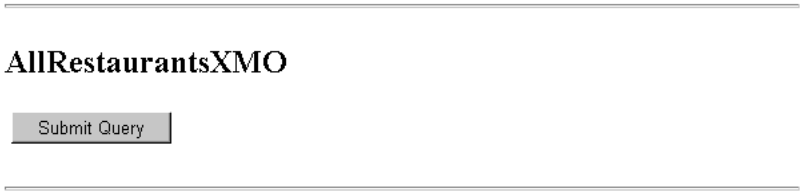
   古いデータに新しいデータが追加されて再表示されます。



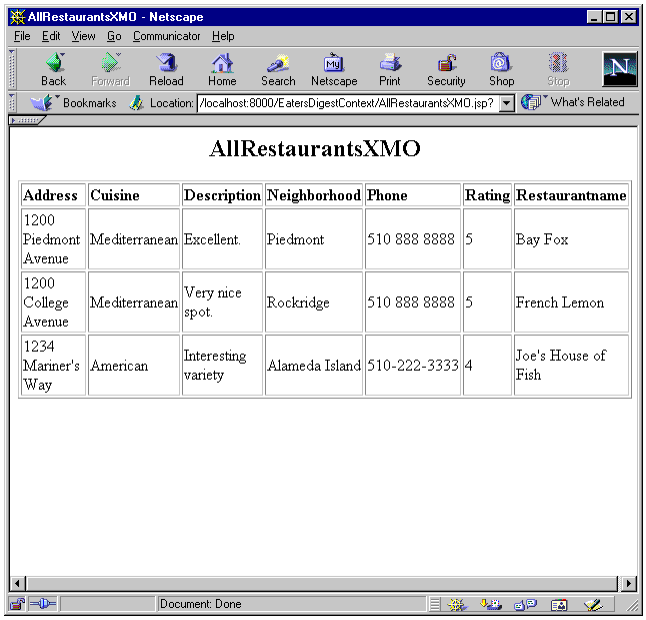
   次に、他のオペレーションをテストします。

7. 開始ページに戻るには、ブラウザの「戻る」ボタンを使用します。

8. 開始ページの「Submit Query」ボタンをクリックして `AllRestaurantsXMO` オペレーションをテストします。



この処理の結果として、次のようなページが表示されます。



もう一度開始ページに戻って、最後のオペレーションをテストします。

最後のオペレーションの `EnterCustomerReviewXMO` はヘルパーオペレーションで、第 5 章でも使用します。このオペレーションは、新しい評価を追加して表示されるページに `Restaurantname` の値を渡す際に使用します。詳細は、111 ページの「顧客評価の追加」を参照してください。

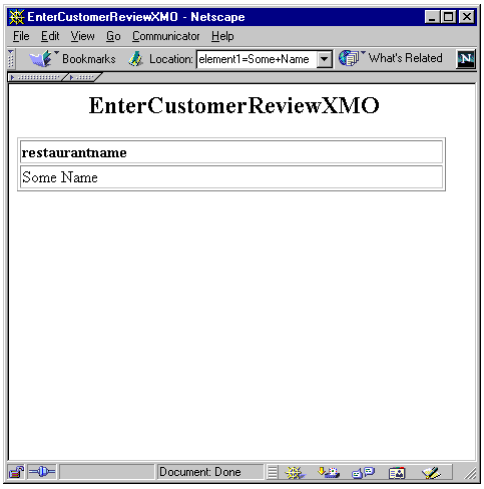
`EnterCustomerReviewXMO` オペレーションをテストすると、入力した内容が返されるかどうか、その内容が適切であるかどうかを確認することができます。

9. **開始ページに戻り、`EnterCustomerReviewXMO` オペレーションのページで任意の文字列を入力します。**

   たとえば、次のように入力します。

10. 「Submit Query」をクリックします。

結果のページに、入力した文字と同じものが表示されます。たとえば、次のようになります。



これは一見したところは、あまり意味のないページですが、JSP ページとパラメータをやりとりする場合に重要になります。

これで、EatersDigest チュートリアルの Web サービスの作成は完了です。第 5 章では、用意されている HTML ページを使用し、完全なクライアントを作成します。この HTML ページは、Web サービス用に生成されている JSP ページを使用します。

第5章

クライアントの作成

この章では、EatersDigest アプリケーションの実行方法について説明します。ここで使用する JSP ページは、第 4 章で生成した JSP ページを基にして、外観や使い勝手を変更しています。変更した JSPページは、`~/sampledir/tutorial/DiningGuide/CustomizedPages` ディレクトリに格納されています。

この章のトピックは、次のとおりです。

チュートリアルのクライアントを開発する

通常の開発プロセスでは、アプリケーション開発者がアプリケーションのバックエンド (EJB 層と Web サービス) を開発し、Web 設計者は、これとは別にアプリケーションのフロントエンド (クライアントの Web ページ) を開発します。この際、メソッド呼び出しと動的データの参照部分はプレースホルダを使用し、アプリケーション開発に依存せずに作業します。ある時点で Web 設計者は、実際のコールおよびクライアントページに対する参照を追加する必要があります。

Forte for Java ツールを使用した開発と Web デザインの統合

Forte for Java には、Web サービスモジュールと Forte for Java Enterprise Service Presentation Toolkit (Forte ESP ツールキット) が用意されており、これを使用してアプリケーションのバックエンドとフロントエンドの統合をサポートします。

開発者は Web サービスを使用し、アプリケーションサービス層からメソッドを呼び出す XML オペレーションを作成します。Web サービスの GUI を使用し、各オペレーションごとにテストクライアント JSP ページを作成できます。この JSP ページには、オペレーションを呼び出すための Forte ESP タグが含まれています。これについては、第 4 章で確認しました。

Forte ESP ツールキットがアプリケーションの Web サーバーにインストールされている場合は、開発者は、これらの XML オペレーションを Forte for Java IDE から Forte ESP レジストリへ公開できます。

アプリケーションの XML オペレーションが Forte ESP レジストリに公開されると、Web 設計者は、Forte ESP の拡張機能がインストールされたデザインツールを使用して、オペレーションの動的データをページレイアウトの要素へグラフィカルにマップすることができます。実行時には、データに動的にアクセスできるように JSP ページにカスタムタグが挿入されます。

Forte ESP ツールキットの拡張機能は、Macromedia® Dreamweaver™ と Adobe® GoLive™ 用に用意されています。Forte ESP ツールキット、および Dreamweaver と GoLive の無料トライアルバージョンは、Forte for Java のポータルからダウンロードすることができます。

EatersDigest の Forte ESP ツールキットの手順の省略

Forte ESP ツールキットには、レストランを題材にした独自のチュートリアルがあります。Forte ESP チュートリアルの設定ファイルは Forte ESP ツールキットに含まれており、動的データを挿入する方法 (Forte ESP タグと XML および XSLT コーディングが自動的に生成される仕組みを詳細に表すサンプル) は、『XML データサービス用 JSP のプログラミング』に記載されています。XML オペレーションの公開、および XML オペレーションを使用する Dreamweaver クライアントの開発のプロセスを説明した簡単なチュートリアルもあります。これについては、Forte for Java のポータルからダウンロード可能な Enterprise Edition 固有のサンプルに含まれている、「Hello Suite」を参照してください。

EatersDigest のチュートリアルでは、Web 設計者が Dreamweaver や GoLive といったアプリケーションを使用し、Forte ESP チュートリアルに記載されている方法と類似したものを使用してクライアント Web ページを開発していることを想定しています。また、Forte ESP ツールキットがインストールされており、第 4 章で Forte ESP レジストリに作成した 4 つの XML オペレーションが公開されている状態で、Web 設計者がこれらのものを Web ページで使用できると想定しています (ESP レジストリに公開する方法は、『Web サービスのプログラミング』を参照してください)。

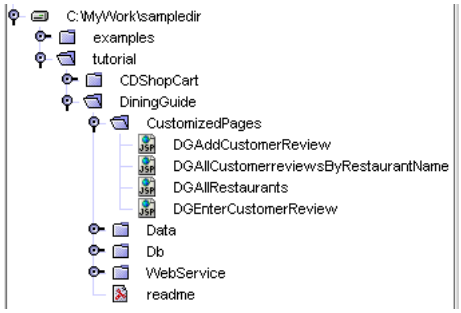
EatersDigest のチュートリアルアプリケーションを完了するには、完成したページを `WebService/Documents` ディレクトリにインストールし、EatersDigest アプリケーションを再配備し、Web ブラウザで最初のページを指定するようにします。

変更した JSP ページをチュートリアルアプリケーションに追加する

変更した 4 つのクライアントページは、ユーザー設定ディレクトリの下の `~/sampledir/tutorial/DiningGuide` ディレクトリにある `CustomizedPages` サブディレクトリに保存されています。Forte for Java IDE で、`sampledir` ディレクトリをマウントし、Web サービスの `Documents` ディレクトリにファイルをコピーし、アプリケーションを再配備します。

1. Forte for Java のエクスプローラで、ユーザー設定ディレクトリの下の `sampledir` ディレクトリをマウントします。
2. `~/sampledir/tutorial/DiningGuide/CustomizedPages` ディレクトリを展開します。

   4 つの JSP ページが表示されます。



3. 4 つの JSP ページを選択し、「編集」>「コピー」を選択します。
4. `DiningGuide/WebService` の `Documents` ノードを右クリックし、「ペースト」>「コピー」を選択します。
5. `EatersDigestApp` ノードを右クリックし、「配備」を選択します。

   配備プロセスが正常に終了したら、106 ページの「チュートリアルアプリケーションの実行」に進みます。

チュートリアルアプリケーションの実行

ブラウザで、配備したアプリケーションを指定して EatersDigest アプリケーションを実行します。

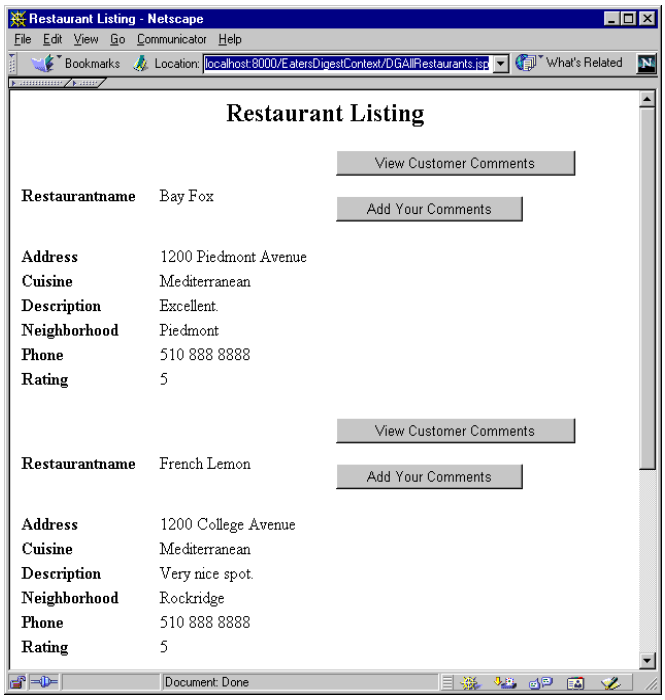
1. J2EE が実行されていることを確認します。

   詳細は、7 ページの「Java 2 SDK, Enterprise Edition の起動」を参照してください。
2. ブラウザで、`EatersDigestContext` ディレクトリの `DGAllRestaurants.jsp` を指定します。

   たとえば、次のように指定します。

   ```
   http://localhost:8000/EatersDigestContext/DGAllRestaurants.jsp
   ```

   Restaurant Listing ページが表示されます。



3. 20 ページの「ユーザーから見たチュートリアルアプリケーション」の説明に従い、実際に機能を使用してみます。
4. ブラウザを終了するか別の URL を指定し、アプリケーションを終了します。

変更内容について

カスタマイズした JSP ページと、Web サービスで生成された JSP ページを比較すると、動的データを処理する際に表現とフローを作成するために使用した技術について理解することができます。この節では、EatersDigest アプリケーションで、変更がどのように行われたかを説明します。詳細は、『XML データサービス用 JSP のプログラミング』に記載されている Forte ESP のチュートリアルを参照してください。

カスタマイズされたページはすべて、第 4 章で説明した Web サービスで生成されたファイルをベースとしています。ファイルの両方のバージョンについて、アプリケーションのメソッド対応の XML オペレーションを、Forte ESP タグのコードブロック `<forte:esp>` *body* `</forte:esp>` で参照することができます。ブロック内には、

他の Forte ESP タグが定義されることもあります。実行時には、Forte ESP タグライブラリ内の Java コードが呼び出され、タグおよびタグの本体が処理されます。これらのタグは JSP の実行時システムで処理されるため、タグを処理するために Forte ESP ツールキットをインストールする必要はありません。

すべてのページにおいて、次の項目が標準的な HTML に変更されています。

- タイトル、表の列見出しの改善
- 表の罫線の消去
- レストラン表の一覧のフォーマットの変更

2 つのレストラン一覧ページを比較してください。

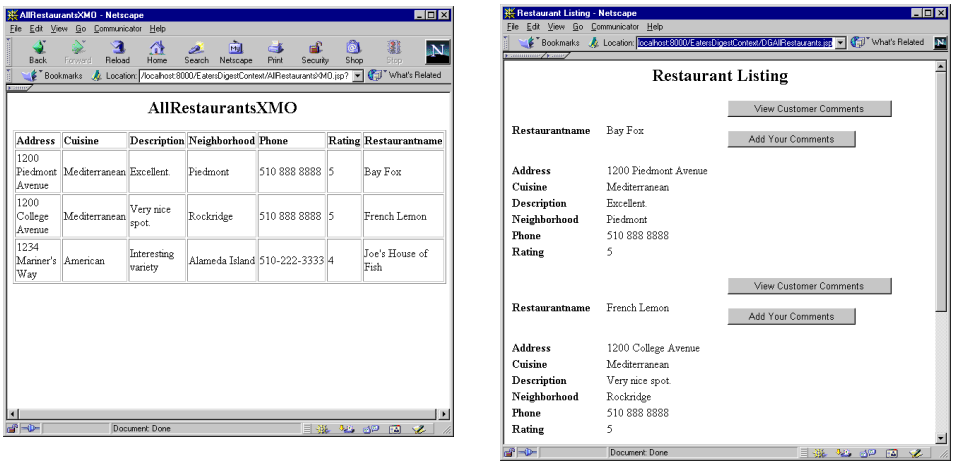


図 5-1　生成されたレストラン一覧のページと `DGAllRestaurants` ページ

顧客の評価を表示するページには、他にも違いがあります。

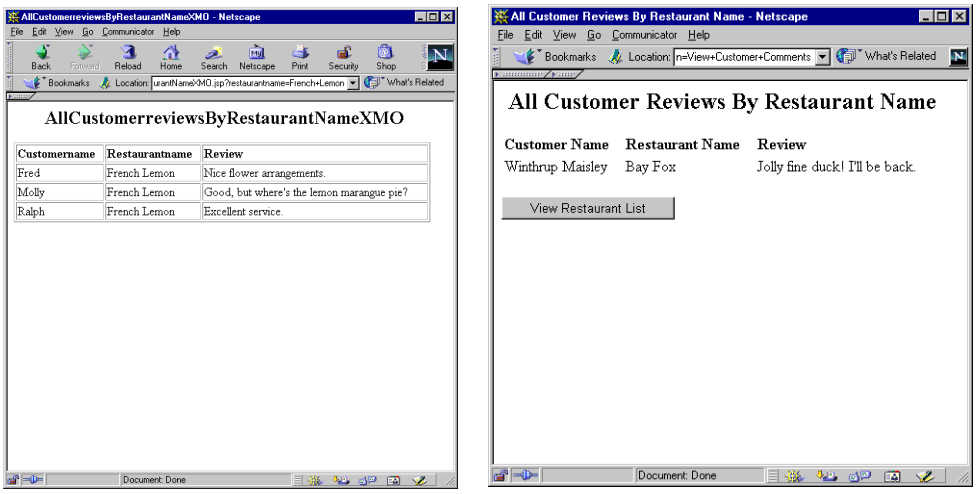


図 5-2　生成された View Customer Review ページと DGAllCustomerreviewsByRestaurantName ページ

「HTML フォームでパラメータを渡す」、「顧客の評価を追加する」という 2 つのカスタマイズについて考えてみます。

HTML フォームでパラメータを渡す

Restaurant Listing ページ (DGAllRestaurants) では、HTML フォームが発行されたときに、XML 結果セットのパラメータが渡される仕組みについて説明しています。これは、ページに表示されている 2 つのボタン (「View Customer Comments」と「Add Your Comments」) によって実行されます。この処理は、すべて次のコードセグメントで表現されます。

コードセグメントは Forte ESP タグで始まります。このタグは、レストランの詳細オブジェクトのコレクションを取得するXML オペレーション用に、Web サービスで生成されます。次に、返されたレストランの各オブジェクトを表示するための XSL 文が定義されています。2 つのボタンに対してフォームがそれぞれ定義されています。

最初のフォームは、「View Customer Comments」ボタンを定義します。このフォームでは AllCustomerreviewsByRestaurantNameXMO をベースとした DGAllCustomerreviewsByRestaurantName ページを呼び出します。コードセグメント内で最初に強調表示 (太字で表示) されている領域では、Forte ESP タグ <forte:inputtext> を使用し、AllRestaurantsXMO の XML 結果セットから restaurantname パラメータを渡し、顧客評価の一覧を取得します。

2 番目のフォームは、「Add Your Comments」ボタンを定義します。このフォームでは、Identity メソッドをベースとして element1 パラメータを渡す DGEnterCustomerReview ページを呼び出します (92 ページの「他のメソッドをベースとして XML オペレーションを作成する」を参照)。XML オペレーションは、アプリケーションで同じパラメータをやりとりできるように生成されています。

このフォームの強調表示 (太字で表示) されている領域では、<forte:inputtext> タグを使用して element1 パラメータを渡します。

```
<forte:esp name="TestAllRestaurantsXMO"
sourceURL="local::AllRestaurantsXMO:"resultSet="AllRestaurantsXMO/Collection-R
estaurant/Restaurant/RestaurantDetail:Address, Cuisine, Description,
Neighborhood, Phone, Rating,Restaurantname">
<table width="99%" border="0">
    <xsl:for-each select="AllRestaurantsXMO/Collection-
Restaurant/Restaurant/RestaurantDetail">
    <tr>
        <td><b>Restaurantname</b></td>
        <td><forte:expand value="{Restaurantname}">Restaurantname
</forte:expand>
        </td>
        <td>
        <form method="get" action="DGAllCustomerreviewsByRestaurantName.jsp">
            <forte:inputtext value="{Restaurantname}">
                <input type="hidden" name="restaurantname"
value="restaurantHolder" />
            </forte:inputtext>

            <input type="submit" name="ViewCommentsButton" value="View Customer
Comments">
            </input>
        </form>
        <form method="get" action="DGEnterCustomerReview.jsp">
            <forte:inputtext value="{Restaurantname}">
                <input type="hidden" name="element1" value="restaurantHolder"/>
            </forte:inputtext>

            <input type="submit" name="AddCommentsButton" value="Add Your
Comments">
            </input>
        </form>
```

顧客評価の追加

このアプリケーションでは、顧客の評価を追加するために、EnterCustomerReviewXMO ページをベースとして DGEnterCustomerReview ページを使用し、データを収集します。DGEnterCustomerReview は DGAddCustomerreview を呼び出します。これは、 AddCustomerReviewXMO オペレーションを使用してデータベースへ実際の挿入を行います。

AddCustomerReviewXMO オペレーション用に表示する開始ページの入力フォームについて考えます。

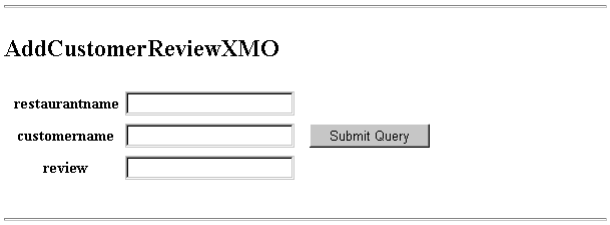


図 5-3　AddCustomerReviewXMO 用に生成された入力フォーム

顧客評価を入力するための、カスタマイズされたページ (DGEnterCustomerReview) については、AddCustomerReviewXMO 入力フォームが使用されています。

図 5-4　DGEnterCustomerReview ページ

ここで、EnterCustomerReviewXMO オペレーションによって、選択されたレストランの restaurantname パラメータが「Restaurant Name」フィールドに挿入されていることがわかります。

DGEnterCustomerReview ページのコードを参照すると、この処理がどのように行われているかがわかります。

```
<forte:esp name="EnterCustomerReviewXMO" inline="true"
sourceURL="local::EnterCustomerReviewXMO:?element1={?element1}"
resultSets="EnterCustomerReviewXMO:restaurantname">

<form method="get" action="DGAddCustomerReview.jsp">
  <TABLE>
    <TR>
      <TD>
        <TABLE>
          <TR>
            <TH>Restaurant Name</TH>
            <TD>
            <forte:inputtext value="{EnterCustomerReviewXMO/restaurantname}">
           <input type="text" name="restaurantname" value="restaurantHolder" />
             </forte:inputtext>
            </TD>
          </TR>
          <TR>
            <TH>Customer Name</TH>
            <TD>
              <input type="text" size="20" name="customername">
              </input>
            </TD>
          </TR>
          <TR>
            <TH>Review</TH>
            <TD>
              <input type="text" size="20" name="review">
              </input>
            </TD>
          </TR>
        </TABLE>
      </TD>
      <TD>
        <input type="submit" value="Submit">
        </input>
      </TD>
    </TR>
  </TABLE>
</form>


</forte:esp>
```

最後に、DGAddCustomerReview ページで AddCustomerReviewXMO を呼び出してデータベースにデータを挿入し、すべてのレストランを表示します。

```
<forte:esp name="AddCustomerReviewXMO" inline="true"
sourceURL="local::AddCustomerReviewXMO:?restaurantname={?restaur
antname}?customername={?customername}?review={?review}"
resultSets="AddCustomerReviewXMO:Customerreview">
    <table width="99%" border="1">
    <tr>
        <td><b>Customerreview</b></td>
    </tr>
    <xsl:for-each select="AddCustomerReviewXMO">
    <tr>
        <td><forte:expand value=
            "{Customerreview}">Customerreview</forte:expand>
        </td>
    </tr>
    </xsl:for-each>
    </table>
</forte:esp>
```

索引

L

M

N

P

R

S

T

W

X

え

か

け

こ

り