

Oracle9i Lite for Web-to-Go

開発者ガイド

リリース 5.0.2.1.0

2003 年 4 月

部品番号 : J07297-01

ORACLE®

Oracle9i Lite for Web-to-Go 開発者ガイド, リリース 5.0.2.1.0

部品番号 : J07297-01

原本名 : Oracle9i Lite Developer's Guide, Release 5.0.2.1.0 for Web-to-Go

原本部品番号 : B10343-01

Copyright © 2001, 2002, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	xi
1 概要	
1.1 Web-to-Go とは	1-2
1.2 概念と用語	1-2
1.3 Web-to-Go アプリケーションの開発	1-2
1.4 Web-to-Go のチュートリアル	1-2
2 概念	
2.1 Web-to-Go	2-2
2.2 Web-to-Go 環境	2-2
2.2.1 Web-to-Go アプリケーション	2-3
2.2.2 Web-to-Go 用 Mobile クライアント	2-3
2.2.3 Mobile サーバー	2-4
2.2.4 データベース・サーバー	2-4
2.2.5 ワークスペース	2-5
2.2.6 Web-to-Go の管理	2-5
2.3 同期の概念	2-6
2.3.1 データのレプリケーション	2-6
2.3.2 オフライン・モード	2-6
2.3.3 オンライン・モード	2-7
2.3.4 データおよびアプリケーションの同期	2-7
2.3.5 クライアント同期モード	2-7
2.4 Web-to-Go での開発	2-8

2.4.1	Mobile Development Kit (Web-to-Go 用)	2-8
2.4.2	パッケージ・ウィザード	2-9
2.5	アクセス制御の管理	2-9
2.5.1	Mobile サーバー・コントロール・センター	2-9

3 Web-to-Go アプリケーションの開発

3.1	概要	3-2
3.2	Web-to-Go アプリケーションの作成	3-2
3.2.1	静的コンポーネント	3-3
3.2.2	動的コンポーネント	3-3
3.2.3	データベース・コンポーネント	3-4
3.2.4	データベース接続	3-5
3.3	アプリケーション・ロール	3-6
3.4	JSP の開発	3-6
3.4.1	Mobile クライアント Web サーバー	3-6
3.4.2	Mobile サーバーまたは Web-to-Go 用 Mobile クライアント	3-7
3.5	Web-to-Go 用 Java サブレットの開発	3-7
3.5.1	制限事項	3-7
3.5.2	Mobile Development Kit (Web-to-Go 用) 上のアプリケーションへのアクセス	3-8
3.5.3	サブレットの作成	3-8
3.5.4	サブレットの実行	3-12
3.5.5	サブレットのデバッグ	3-18
3.5.6	Oracle Lite 内のスキーマの直接アクセス	3-18
3.6	Web-to-Go アプレットの使用	3-19
3.6.1	Web-to-Go アプレットの作成	3-19
3.6.2	アプレット用の HTML ページの作成	3-20
3.7	アプレット-JDBC 通信の開発	3-21
3.7.1	getConnection()	3-21
3.7.2	設計上の課題	3-22
3.8	アプレット-サブレット通信の開発	3-22
3.8.1	Web-to-Go サブレットの作成	3-23
3.9	Web-to-Go アプリケーションのデバッグ	3-24
3.9.1	Oracle9i JDeveloper を使用したサンプル 1 の実行	3-25
3.10	ワークスペース・アプリケーションのカスタマイズ	3-29
3.10.1	Webtogo.ora パラメータ	3-30

3.10.2	サンプル・ワークスペース	3-31
3.11	Mobile Server Admin API の使用	3-32

4 Web-to-Go のチュートリアル

4.1	概要	4-3
4.1.1	作業の準備	4-3
4.2	アプリケーションの開発	4-4
4.2.1	ステップ 1: Oracle Lite へのデータベース・オブジェクトの作成	4-4
4.2.2	ステップ 2: アプリケーション・コードの作成	4-6
4.2.3	ステップ 3: アプリケーションのコンパイル	4-6
4.2.4	ステップ 4: アプリケーションの定義およびサプレットの登録	4-7
4.2.5	ステップ 5: アプリケーションの実行	4-15
4.3	アプリケーションのパッケージ化	4-18
4.3.1	ステップ 1: パッケージ・ウィザードを使用したアプリケーションの定義	4-19
4.3.2	ステップ 2: Oracle データベースへのアプリケーション接続の定義	4-22
4.3.3	ステップ 3: スナップショットの定義	4-24
4.3.4	ステップ 4: シーケンスの定義	4-27
4.3.5	ステップ 5: アプリケーションの SQL ファイルの作成	4-29
4.3.6	ステップ 6: アプリケーションのパッケージ化	4-30
4.4	アプリケーションのパブリッシュ	4-31
4.4.1	ステップ 1: 表の所有者アカウントの作成	4-31
4.4.2	ステップ 2: Oracle データベースへのデータベース・オブジェクトの作成	4-31
4.4.3	ステップ 3: Mobile サーバーの起動	4-31
4.4.4	ステップ 4: Mobile サーバーへのログオンと Mobile サーバー・コントロール・センターの起動	4-32
4.4.5	ステップ 5: アプリケーションのアップロード	4-33
4.5	アプリケーションの管理	4-35
4.5.1	ステップ 1: Mobile サーバー・コントロール・センターの起動	4-35
4.5.2	ステップ 2: コントロール・センターを使用した新規ユーザーの作成	4-37
4.5.3	ステップ 3: アプリケーション・プロパティの設定	4-38
4.5.4	ステップ 4: アプリケーションへのアクセス権の付与	4-40
4.5.5	ステップ 5: ユーザーのスナップショット・テンプレート値の定義	4-41
4.5.6	ステップ 6: Message Generator and Processor (MGP) の起動	4-43
4.6	Web-to-Go 用 Mobile クライアントでのアプリケーションの実行	4-44
4.6.1	ステップ 1: Web-to-Go 用 Mobile クライアントのインストール	4-44

4.6.2	ステップ 2: Web-to-Go 用 Mobile クライアントへのログイン	4-46
4.6.3	ステップ 3: Web-to-Go 用 Mobile クライアントの同期	4-52

5 Web-to-Go アプリケーションの定義

5.1	概要	5-2
5.2	接続が切断されているクライアントのためのシーケンス・サポート	5-2
5.2.1	Web-to-Go シーケンス	5-2
5.2.2	WINDOW シーケンス	5-2
5.3	パッケージ・ウィザードの使用	5-7
5.3.1	パッケージ・ウィザードの起動	5-7
5.3.2	プラットフォームの選択	5-8
5.3.3	開発モードでのパッケージ・ウィザードの起動	5-9
5.3.4	新規アプリケーションの命名	5-10
5.3.5	アプリケーション・ファイルの表示	5-15
5.3.6	サブレットの追加	5-18
5.3.7	データベース情報の入力	5-19
5.3.8	アプリケーション・ロールの定義	5-21
5.3.9	レプリケーション用スナップショットの定義	5-22
5.3.10	レプリケーション用のシーケンスの定義	5-30
5.3.11	アプリケーションの DDL の定義	5-35
5.3.12	レジストリでの名前と値のペアの定義	5-37
5.3.13	アプリケーションの完了	5-38
5.3.14	アプリケーションの編集	5-40
5.4	スキーマ展開	5-40
5.4.1	パッケージ・ウィザードの起動	5-41
5.4.2	スナップショットの編集	5-41
5.4.3	アプリケーションの完了	5-42
5.5	web.xml 形式のサポート	5-43
5.5.1	web.xml ファイルの場所	5-43
5.5.2	web.xml でサポートされるタグ	5-43

6 BC4J のチュートリアル

6.1	概要	6-2
6.1.1	作業の準備	6-2
6.2	アプリケーションの開発	6-3

6.2.1	データベース接続の作成	6-3
6.2.2	BC4J コンポーネントの作成	6-7
6.2.3	WTGJdbc 接続を使用するための BC4J コンポーネントの構成	6-10
6.2.4	シンプル・アーカイブとしての BC4J コンポーネントの構築および配布	6-11
6.2.5	BC4J コンポーネントにアクセスするための JSP アプリケーションの作成	6-11
6.2.6	シンプル・アーカイブとしての JSP アプリケーションの配布	6-15
6.3	JSP アプリケーションのパッケージ化	6-15
6.4	コントロール・センターからの JSP アプリケーションのパブリッシュおよび構成	6-18
6.5	BC4J アプリケーションのテスト	6-18
6.6	Web-to-Go 用 Mobile クライアント上での BC4J アプリケーションの実行	6-19
6.7	サンプル・アプリケーションの配布	6-19

A Web-to-Go サンプル・アプリケーション

A.1	概要	A-2
A.1.1	Mobile サーバー	A-2
A.1.2	Mobile Development Kit (Web-to-Go 用)	A-2
A.1.3	Mobile Development Kit (Web-to-Go 用) からのサンプル・プログラムのアクセス	A-2
A.1.4	Mobile サーバーからのサンプル・プログラムのアクセス	A-3
A.2	Sample 1 - Hello World	A-3
A.2.1	ソース・コードの場所	A-3
A.2.2	アプリケーション・ファイル	A-4
A.3	Sample 3 - Recording Tracker	A-4
A.3.1	Sample 3 の使用方法	A-4
A.3.2	Sample 3 のデータベース表	A-4
A.3.3	Sample 3 のサーブレット	A-5
A.3.4	Sample 3 のリソース・バンドル	A-5
A.3.5	ソース・コードの場所	A-5
A.3.6	アプリケーション・ファイル	A-5
A.4	Sample 4 - Hello Applet	A-6
A.4.1	Sample 4 のサーブレット	A-7
A.4.2	ソース・コードの場所	A-7
A.4.3	アプリケーション・ファイル	A-7
A.5	Sample 6 - Image Gallery	A-7
A.5.1	ソース・コードの場所	A-8
A.5.2	アプリケーション・ファイル	A-8

A.6	Sample 7 - Employee Data Applet	A-9
A.6.1	ソース・コードの場所	A-9
A.6.2	アプリケーション・ファイル	A-9

B Web-to-Go Java パッケージ

B.1	oracle.html パッケージの使用方法	B-2
B.1.1	HtmlPage オブジェクトの作成	B-2
B.1.2	<HEAD> 領域へのタグの追加	B-2
B.1.3	<BODY> タグへのコンテンツの追加	B-2
B.1.4	HTML ページの <BODY> セクションへのタグの追加	B-3
B.1.5	クラスを使用した HTML 生成の例	B-3
B.1.6	HTML 要素の Java クラスへのマッピング	B-7
B.1.7	oracle.html パッケージのクラス階層	B-11
B.1.8	IHtmlItem インタフェース	B-11
B.1.9	Item 抽象クラス	B-12
B.1.10	CompoundItem クラスと Container クラス	B-12
B.1.11	HTML ページとオブジェクトの作成	B-12
B.1.12	oracle.html パッケージの継承	B-13
B.2	oracle.lite.web.html パッケージの使用方法	B-16
B.2.1	TemplateParser クラスを使用した HTML テンプレートの処理	B-16
B.2.2	emp.html	B-20
B.2.3	DeleteRecords サブレットを使用したレコードの削除	B-21
B.2.4	マスター・ディテール・フォームの作成と処理	B-22

用語集

索引

図目次

2-1	Web-to-Go 環境	2-2
2-2	ワークスペース	2-5
3-1	開発アーキテクチャ	3-2
3-2	「アプリケーションの選択」ダイアログ・ボックス	3-13
3-3	「アプリケーション」パネル	3-14
3-4	新規プロジェクトの作成	3-25
3-5	プロジェクト設定—入力パス	3-26
3-6	「新規ライブラリの作成」ダイアログ・ボックス	3-28
4-1	「アプリケーションの選択」ダイアログ・ボックス	4-9
4-2	プラットフォーム選択用のダイアログ・ボックス	4-10
4-3	「アプリケーション」パネル	4-11
4-4	すべてのアプリケーション・ファイルのアップロード	4-12
4-5	JSP のコンパイル完了の成功メッセージ	4-13
4-6	新しく生成されたファイルは自動的に追加される	4-14
4-7	「サブレット」パネル	4-15
4-8	アプリケーションのリスト	4-17
4-9	「プラットフォーム」パネル	4-20
4-10	「アプリケーション」パネル	4-21
4-11	「データベース」パネル	4-22
4-12	「スナップショット」パネル	4-24
4-13	「表」ダイアログ・ボックス	4-25
4-14	「スナップショットの編集」ダイアログ・ボックス	4-26
4-15	「スナップショットの編集」ダイアログ・ボックスの「Win32」パネル	4-27
4-16	「シーケンス」パネル	4-28
4-17	「シーケンス」ウィンドウ	4-28
4-18	「アプリケーションの定義の完了」ダイアログ・ボックス	4-29
4-19	「アプリケーションの保存」ダイアログ・ボックス	4-30
4-20	コントロール・センターの起動	4-33
4-21	「新規アプリケーション」ページ	4-34
4-22	コントロール・センターの起動	4-36
4-23	「ユーザーのプロパティ」パネル	4-38
4-24	アプリケーション・プロパティの設定	4-39
4-25	アプリケーションへのユーザー・アクセス権の付与	4-41
4-26	データ・サブセッティング・パラメータ	4-42
4-27	MGP の開始	4-43
4-28	Web-to-Go 用 Mobile クライアントのインストール	4-45
4-29	Web-to-Go ログオン・ページ	4-46
4-30	Web-to-Go 用 Mobile クライアントの初期化	4-48
4-31	「Web-to-Go の同期」画面	4-49
4-32	「Web-to-Go 用 Mobile クライアントを再起動」画面	4-50
4-33	同期プロセスの完了: アプリケーションのダウンロード終了	4-51
4-34	「To Do List」アプリケーション	4-52
4-35	「Web-to-Go の同期」ページ	4-53

5-1	「アプリケーションの選択」ダイアログ・ボックス	5-8
5-2	プラットフォーム選択用のダイアログ・ボックス	5-9
5-3	「アプリケーション」パネル	5-14
5-4	「ファイル」パネル	5-16
5-5	「JSP のコンパイル」ダイアログ・ボックス	5-17
5-6	「サーブレット」パネル	5-19
5-7	「データベース」パネル	5-20
5-8	「ロール」パネル	5-21
5-9	「スナップショット」パネル	5-24
5-10	「新規スナップショット」 - 「サーバー」パネル	5-25
5-11	「新規スナップショット」 - 「Win32」パネル	5-26
5-12	索引の作成	5-27
5-13	「スナップショットの編集」ダイアログ・ボックス (クライアント)	5-29
5-14	「シーケンス」パネル	5-32
5-15	「DDL」パネル	5-35
5-16	「レジストリ」パネル	5-37
5-17	「アプリケーションのパブリッシュ」ダイアログ・ボックス	5-39
5-18	「アプリケーションの選択」ダイアログ・ボックス	5-41
5-19	「スナップショットの編集」ダイアログ・ボックス (クライアント)	5-42
6-1	Connection Manager ウィザード	6-4
6-2	「Connection」ダイアログ・ボックス	6-5
6-3	「ビジネス・コンポーネント・パッケージウィザード: 初期画面」ダイアログ・ボックス	6-7
6-4	「ビジネス・コンポーネント・パッケージウィザード: ステップ 1/3: パッケージ名」	6-8
6-5	「ビジネス・コンポーネント・パッケージウィザード: ステップ 2/3: 接続」	6-9
6-6	「ビジネス・コンポーネント・パッケージウィザード: ステップ 3/3: ビジネス・コンポーネント」ダイアログ・ボックス	6-10
6-7	「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ようこそ」	6-12
6-8	「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」	6-12
6-9	「BC4」クライアント・データ・モデル定義ウィザード - ようこそ」	6-13
6-10	「BC4」クライアント・データ・モデル定義ウィザード - ステップ 1/2: 定義」	6-13
6-11	「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」	6-14

表目次

4-1	チュートリアル概要	4-1
4-2	開発用コンピュータの要件	4-3
4-3	「To Do List」アプリケーションのコンポーネント	4-4
4-4	TODO_ITEMS 表	4-5
4-5	「To Do List」アプリケーションの設定	4-11
4-6	システムが現在認識しているアプリケーションのリスト	4-17
4-7	指定されている値	4-21
4-8	必須アクションの値	4-23
5-1	プラットフォームに応じた必須サブディレクトリ名のリスト	5-13
5-2	サポートされる web.xml タグ	5-43
6-1	開発用コンピュータの要件	6-2
6-2	「tutorialConn」接続要件	6-6
6-3	「WTGJdbc」接続要件	6-6
6-4	「ビジネス・コンポーネント・パッケージウィザード: ステップ 2/3: 接続」の値	6-9
6-5	パッケージ・ウィザードの入力内容	6-16
6-6	サーブレット名およびサーブレットのクラス	6-16
6-7	データベース値	6-17
6-8	「データベースの接続」ウィンドウの説明	6-17

はじめに

このマニュアルでは、Web-to-Go アプリケーションの開発方法に関する情報を提供します。このマニュアルは、Web-to-Go の概要を紹介し、サンプル・アプリケーションを提供します。

このマニュアルの内容は、次のとおりです。

第 1 章「概要」	Web-to-Go の概要およびこのマニュアルの使用方法を説明します。
第 2 章「概念」	Web-to-Go の機能と用語を理解するための概念的なフレームワークを提供します。
第 3 章「Web-to-Go アプリケーションの開発」	Web-to-Go アプリケーションの開発方法に関する情報を提供します。
第 4 章「Web-to-Go のチュートリアル」	チュートリアル方式を使用して Web-to-Go の実装方法を説明します。
第 5 章「Web-to-Go アプリケーションの定義」	Web-to-Go アプリケーションの定義とパッケージ化の手順を説明します。
第 6 章「BC4J のチュートリアル」	BC4J のチュートリアルについて説明します。
付録 A「Web-to-Go サンプル・アプリケーション」	Web-to-Go アプリケーションのサンプルについて説明します。
付録 B「Web-to-Go Java パッケージ」	Web-to-Go Java パッケージのサンプルについて説明します。

この章では、Oracle Web-to-Go の概要を説明します。内容は次のとおりです。

- 1.1 項「Web-to-Go とは」
- 1.2 項「概念と用語」
- 1.3 項「Web-to-Go アプリケーションの開発」
- 1.4 項「Web-to-Go のチュートリアル」

1.1 Web-to-Go とは

Web-to-Go は、Web アプリケーションのための実装プラットフォームです。これは Mobile サーバーの一部であり、イントラネットおよびインターネットの Mobile アプリケーションの作成、配置、管理および操作を簡単にする機能が含まれています。Web-to-Go を使用すると、サーバーおよびラップトップ上で稼働する、ブラウザ・ベースのスケーラブルな Web アプリケーションを開発できます。また、アプリケーションの実行が必要になった場合、選択したアプリケーションやデータを Oracle データベースからクライアント・デバイス上の Oracle Lite データベースにダウンロードできます。ユーザーは、クライアント・デバイス上の Oracle Lite に接続して、オフラインで Web-to-Go アプリケーションを実行します。ユーザーは、選択したアプリケーションでの作業を終了した後、Web-to-Go を使用して Oracle Lite から Oracle データベースにデータをレプリケートして、アプリケーションとデータを同期させます。

1.2 概念と用語

Web-to-Go を実装する前に、Web-to-Go の概念と用語を理解しておく必要があります。[第 2 章「概念」](#)には Web-to-Go の概要が説明されており、[用語集](#)には Web-to-Go の用語と定義の完全なリストが掲載されています。作業を開始する前に、この 2 つの章をよく読んでください。

1.3 Web-to-Go アプリケーションの開発

[第 3 章「Web-to-Go アプリケーションの開発」](#)の説明では、Web-to-Go アプリケーションの開発方法の概要を提供します。アプリケーション開発の詳細に関して、本書では、サンプル・アプリケーションとトラブルシューティング情報も提供しています。

開発したアプリケーションのパブリッシュ、管理および配置の各処理は、『Oracle9i Lite 管理者およびデプロイ・ガイド』に説明されています。

1.4 Web-to-Go のチュートリアル

Web-to-Go の概念と用語を理解して Web-to-Go アプリケーション開発の概要を体系的に把握すると、Web-to-Go の開発および実装過程に関する具体的な詳細を理解できます。[第 4 章「Web-to-Go のチュートリアル」](#)では、単純な Web-to-Go アプリケーションの開発と実行の過程を順に説明します。

この章では、Web-to-Go の機能と用語を理解するための概念的なフレームワークを提供します。内容は次のとおりです。

- 2.1 項「Web-to-Go」
- 2.2 項「Web-to-Go 環境」
- 2.3 項「同期の概念」
- 2.4 項「Web-to-Go での開発」
- 2.5 項「アクセス制御の管理」

2.1 Web-to-Go

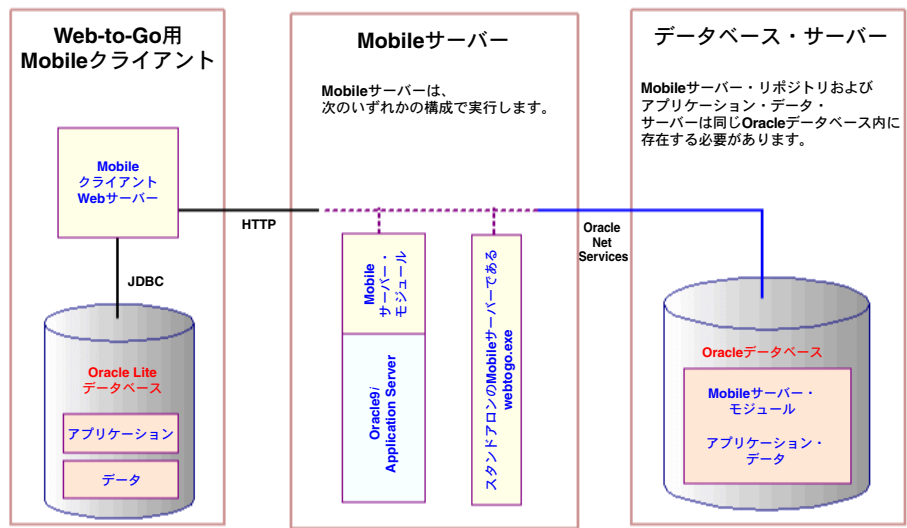
Web-to-Go は、Web ベースの Mobile データベース・アプリケーションを作成および配置するためのフレームワークで、Mobile サーバーの一部を構成します。Web-to-Go モデルは、常にオフラインの状態にあり、同期機能を使用して Oracle データベースとデータ変更を同期するユーザーをサポートします。

ユーザーは、Web ブラウザを使用して Web-to-Go 用 Mobile クライアントから Web-to-Go アプリケーションを実行します。Web-to-Go 用 Mobile クライアントのインストールと使用に必要なものは、HTML ブラウザと Mobile サーバーへのアクセスのみです。

2.2 Web-to-Go 環境

Web-to-Go 環境は、クライアント、アプリケーション・サーバーおよびデータベース・サーバーからなる 3 層の Web モデルです。Web-to-Go ユーザーは、Web-to-Go 用 Mobile クライアントがネットワークに接続されているかどうかにかかわらず、アプリケーションとデータにアクセスできます。各層は、別のマシン上に格納できます。

図 2-1 Web-to-Go 環境



2.2.1 Web-to-Go アプリケーション

Web-to-Go アプリケーションは Mobile データベース・アプリケーションで、オンライン・モードとオフライン・モードでシームレスに稼働します。開発者は、オンライン・モードとオフライン・モード用に別々の Web-to-Go アプリケーションを作成する必要はありません。Web-to-Go のランタイム環境ではオンラインとオフラインの違いが隠されているため、この 2 つの違いはユーザーにとっては透過的です。

ユーザーは、Web ブラウザを介して Web-to-Go アプリケーションにアクセスします。通常、これらのアプリケーションは Oracle データベース・データ・サーバーに格納されているデータの変更に使用されます。Web-to-Go アプリケーションは、静的コンポーネント (HTML ページ、イメージ・ファイルなど)、動的コンポーネント (Java サブレットなど)、およびデータベース・オブジェクト (表、シーケンスなど) で構成できます。Web-to-Go では、これらのコンポーネントをすべてオンラインおよびオフラインの両方のモードで管理し、コンポーネントに加えられたすべての変更を自動的に Web-to-Go システム全体に伝播します。

Web-to-Go は、Java サブレットを使用して動的 Web ページの作成を簡略化します。Java サブレットはプラットフォームから独立したサーバー側モジュールで、Web サーバーの機能を継承するために使用できます。これらのモジュールは Java で作成されていて、Web サーバーにロードされます。

Web-to-Go は、スナップショットおよびシーケンスの 2 つのタイプのデータベース・コンポーネントをサポートします。各コンポーネントは、パッケージ・ウィザードを使用してアプリケーションに登録する必要があります。これによって、Web-to-Go はローカル・クライアント上のスナップショットやシーケンスに必要なオフライン・サポートを作成できます。Web-to-Go は、クライアントが初めてオフライン・モードになった場合、スナップショットやシーケンスに対するこのサポートを作成し、それ以降、スナップショットおよびシーケンスが動的であるため、これらの同期を保ち続けます。

さらに、Web-to-Go はカスタム DDL (データ定義言語) 文を実行し、クライアント上でのビューや索引などのデータベース・オブジェクトの作成を可能にします。このイベントは、クライアントが初めてオフラインになった場合に 1 度のみ発生します。

Web-to-Go アプリケーションのコンポーネントの詳細は、[第 3 章「Web-to-Go アプリケーションの開発」](#)を参照してください。

2.2.2 Web-to-Go 用 Mobile クライアント

クライアント層は、Mobile クライアント Web サーバーおよび Oracle Lite データベースで構成されます。この層のインストールと使用に必要なものは、Mobile サーバーに対するアクセス権と Web ブラウザのみです。クライアント層をインストールすると、ユーザーは、Web ブラウザを介して Mobile クライアント Web サーバーにアクセスして Web-to-Go アプリケーションを実行できます。

2.2.2.1 サイト

Web-to-Go は、Web-to-Go 用 Mobile クライアント上の各ユーザーに対して複数のデータベースを作成します。たとえば、John および Jane がユーザーである場合、Web-to-Go はそれぞれに対して次のようにデータベースを作成します。

```
oldb40¥john¥orders.odt  
oldb40¥john¥entry.odt
```

John 用のこれら 2 つのデータベースは、ともにサイトと呼ばれます。Jane に対しても、Web-to-Go は次のようなデータベースを作成します。

```
oldb40¥jane¥orders.odt  
oldb40¥jane¥entry.odt
```

Jane 用のこれら 2 つのデータベースも、ともにサイトと呼ばれます。Web-to-Go 用 Mobile クライアントには、ユーザー 1 人当たり 1 つのサイトが含まれます。ただし、ユーザーは複数の異なる Web-to-Go 用 Mobile クライアント上にサイトを所有できます。管理者は、Mobile サーバー・コントロール・センターを使用してサイトを追跡し管理します。詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

2.2.3 Mobile サーバー

アプリケーション・サーバーの層には Mobile サーバーが含まれます。このサーバーは、Web-to-Go 用 Mobile クライアントからの要求を処理してデータベース・サーバー内のデータを変更します。Mobile サーバーは、次のいずれかの Web アプリケーション・サーバーとともに稼働するハンドラです。

- Oracle9i Application Server (Oracle9iAS)
- Oracle HTTP Server
- Apache Server
- スタンドアロンの Mobile サーバーである **webtogo.exe**

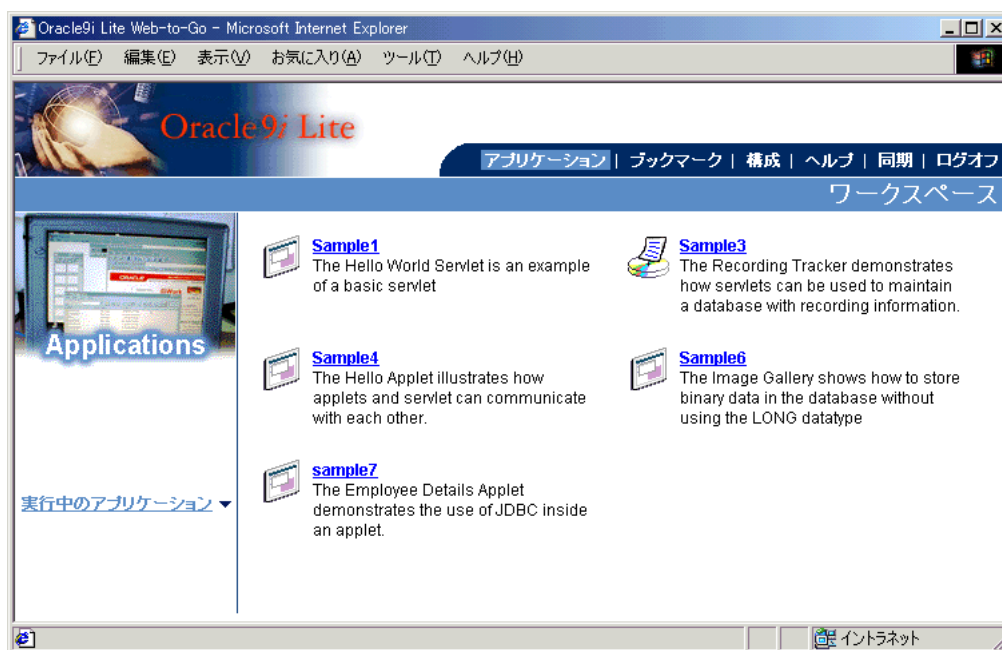
2.2.4 データベース・サーバー

データベース・サーバーの層は、アプリケーション・データと Web-to-Go ファイルを格納します。Web-to-Go ファイルは、Mobile サーバー・リポジトリ (Oracle データベース上に常駐する仮想ファイル・システム) に格納されます。このリポジトリは、すべてのアプリケーション・ファイルとアプリケーション定義を含む永続リソース・リポジトリです。

2.2.5 ワークスペース

ユーザーは、ワークスペースと呼ばれる Web ページから Web-to-Go アプリケーションにアクセスします。Web-to-Go は、ユーザーが Web-to-Go にログインした後に、ユーザーのブラウザ内にワークスペースを生成します。ワークスペースは、ユーザーが使用できるすべてのアプリケーションのアイコン、リンクおよび説明を表示します。ユーザーがアプリケーションを使用できるようになるのは、管理者がアプリケーションを Mobile サーバーにパブリッシュし、ユーザーに対してアクセス権を付与した後です。

図 2-2 ワークスペース



2.2.6 Web-to-Go の管理

Web-to-Go は、Mobile サーバーから一元的に管理されます。Mobile サーバーのパッケージ・ウィザードと Mobile サーバー・コントロール・センターにより、管理者は、アプリケーションをパブリッシュおよび管理できます。パッケージ・ウィザードを使用して、開発者は管理者がパブリッシュする予定のアプリケーションを定義し、管理者は定義されたアプリケーションを Mobile サーバーにパブリッシュします。アプリケーションは Mobile サーバー・コントロール・センターで管理できます。管理者はここでユーザーの作成や権限の割当てができます。

2.3 同期の概念

Web-to-Go 用 Mobile クライアントは Mobile Sync を使用して、ローカル Oracle Lite データベースと Oracle データベース・サーバーの間でデータ変更をレプリケートします。

2.3.1 データのレプリケーション

データの同期中、Web-to-Go 用 Mobile クライアントは、データ変更を Oracle データベースのインキューにアップロードします。次に、クライアントは、アウトキューから新しい更新データをダウンロードし、ローカルの Oracle Lite データベースに適用します。

Consolidator Message Generator and Processor (MGP) は、キューから Oracle データベースに対して、保留中のトランザクションを設定された間隔で適用する Java バックグラウンド・プロセスです。また、Web-to-Go 用 Mobile クライアントがダウンロードする新しいデータ更新も生成します。このような更新は、アウトキューに格納され、次の同期時に Web-to-Go 用 Mobile クライアントにより取り出されます。Web-to-Go 用 Mobile クライアントによってアップロードされたデータの変更は、MGP がインキューを処理してその変更を適用するまで、Oracle データベースの表には反映されません。同様に、MGP 実行後に行われた Oracle データベースの表に対する変更はアウトキューには追加されず、Web-to-Go 用 Mobile クライアントによる同期時にもダウンロードされません。

MGP が実行中でない場合でも、変更をインキューにアップロードしたり、更新をアウトキューからダウンロードできます。ただし、インキューは処理されず、アウトキューは新しい変更を受信しません。MGP は、Mobile サーバー・コントロール・センターから起動および停止できます。MGP の起動の詳細は、『Oracle9i Lite 管理者およびデブレイ・ガイド』を参照してください。

MGP を実行する時間を制御することで、アプリケーションは多くの異なる同期ポリシーを実装できます。たとえば企業では、すべての Mobile アプリケーション・ユーザーに、業務終了後および翌日の業務開始前に各 1 回の合計 2 回、同期を義務付けている場合があります。MGP は、これら 2 回の同期期間の間に実行できます。これにより、全員が毎日すべての更新内容を取得することが保証されます。

2.3.2 オフライン・モード

ユーザーは、Web-to-Go アプリケーションをオフラインで実行します。このモードでは、ネットワークに接続せずに Web-to-Go アプリケーションを実行できます。Mobile クライアント Web サーバーは、ユーザーのアプリケーションへのリンクを持つワークスペース・ページを生成します。これらのリンクは Mobile クライアント Web サーバーを指しているため、リンクの 1 つをクリックするとクライアント層でアプリケーションが起動します。これらのアプリケーションは、Mobile クライアント Web サーバーによりクライアント上で実行され、Oracle Lite データベースに格納されているデータにアクセスします。

ユーザーは、アプリケーションでの作業を終了すると、Oracle データベース・データ・サーバーとの同期を開始して、Oracle Lite データベースと Mobile サーバー・リポジトリの間でユーザーのアプリケーションおよびデータを同期させます。

2.3.3 オンライン・モード

ユーザーは、オンラインでもアプリケーションを実行できます。この場合、クライアントはネットワークに接続する必要があります。ユーザーがアプリケーションをオンライン・モードで実行する場合、Web-to-Go 用 Mobile クライアントはユーザーのログイン要求を Oracle9i Application Server (Oracle9iAS) 上の Mobile サーバー・モジュールにリダイレクトします。その後、Mobile サーバー・モジュールは、ワークスペース・ページを生成し、このページを Web ブラウザに返します。ワークスペースには、Mobile サーバー上のユーザーの Web-to-Go アプリケーションへのリンクが含まれています。アプリケーション・リンクをクリックすると、アプリケーションが起動します。アプリケーションは中間層で実行され、Oracle データベース・データ・サーバーに格納されているデータにアクセスします。

2.3.4 データおよびアプリケーションの同期

データおよびアプリケーションの同期は、次のいずれかの方法で開始できます。

- ユーザーによるオフライン・モードでの明示的同期
- オンライン・モードとオフライン・モードの切替えによる暗黙的同期

ユーザーが同期をとると、Web-to-Go はローカルに加えられたデータ変更を Oracle データベース・データ・サーバーにレプリケートします。また、Oracle データベース・データ・サーバーに対するデータ変更は、すべて Web-to-Go 用 Mobile クライアントの Oracle Lite DBMS 内のデータに適用されます。また、すべてのアプリケーションの変更が Web-to-Go 用 Mobile クライアントにダウンロードされます。

2.3.5 クライアント同期モード

Web-to-Go 用 Mobile クライアントは、次の 2 つのモードで同期できます。

- 常にオフライン
- オンライン / オフライン

モードは、Web-to-Go ワークスペースに移動して「構成」タブをクリックすることで選択できます。クライアント・モードに対するオプションを表示したパネルが表示されます。

「常にオフライン」モードを選択すると、Web-to-Go アプリケーションはネットワークから切断している間にクライアント上で実行されます。データはオフライン中に変更できます。アプリケーションおよびデータを同期させる場合は、ネットワークに接続して、Web-to-Go ワークスペースの「同期」タブをクリックします。クライアントと Mobile サーバーの間でデータおよびアプリケーションが同期されます。

「オンライン / オフライン」モードを選択してネットワークに接続している場合は、Web-to-Go ワークスペースの「オンライン」タブをクリックします。これでオンライン・モードになり、同期プロセスが自動的に開始します。この結果、Mobile サーバー上でオンライン・モードで作業することになります。アプリケーションは Mobile サーバー上で実行されているため、データのみが同期されます。このモードでは、「オンライン」タブの表示が「オフライン」に変わり、いつでも希望するときにオフラインにできるオプションが提供されます。

どちらのモードでも、同期は、[2.3.4 項「データおよびアプリケーションの同期」](#)の説明どおり行われます。

2.4 Web-to-Go での開発

Web-to-Go の Java API を使用すると、開発者はデータベース表のレプリケーションなどの機能をコーディングする必要がないため、Mobile アプリケーションの開発が簡単になります。詳細は、[第 3 章「Web-to-Go アプリケーションの開発」](#)の「[Web-to-Go アプリケーションの作成](#)」を参照してください。

2.4.1 Mobile Development Kit (Web-to-Go 用)

Mobile Development Kit (Web-to-Go 用) を使用して、Web-to-Go アプリケーションを開発およびデバッグできます。開発キットには、チュートリアル、サンプル・プログラム、カスタム・ワークスペース、Web-to-Go API のドキュメントが含まれています。開発キットを使用すると、Java デバッガ内部で Web-to-Go アプリケーションを実行できるため、Java サブプレットの開発が簡単になります。詳細は、[第 3 章「Web-to-Go アプリケーションの開発」](#)の「[Web-to-Go 用 Java サブプレットの開発](#)」を参照してください。

Mobile Development Kit (Web-to-Go 用) に含まれている Java ライブラリには多くの機能が提供されているため、Mobile アプリケーションの開発を簡単かつ効率的に行えます。これらの機能の例としては、接続プーリングや Java ベースの HTML ライブラリがあります。

2.4.1.1 接続プーリング

接続プーリングは、データベースへのデータ・アクセスを必要とするサブプレットを実行するための高速かつスケーラブルなソリューションです。新規 HTTP 要求ごとに新規データベース接続を作成することは、比較的高価で時間のかかる操作です。プーリング接続では、これらの接続を作成する必要はありません。Web-to-Go がデータベース接続のプールを管理し、アプリケーションは接続プールからデータベース接続を要求するのみです。Web-to-Go のモードがオフラインの場合は Oracle Lite に接続され、オンラインの場合は Oracle データベースに接続されます。接続プールによって、アプリケーションからの複雑な接続管理が必要なくなります。

2.4.1.2 Java ベースの HTML ライブラリ

Java ベースの HTML ライブラリを使用すると、開発者は Java サーブレット・コード内に HTML オブジェクトを含めることができます。このライブラリは、Mobile サーバー・リポジトリに格納されている静的 HTML ページに基づいて動的 HTML コンテンツを作成するときにも使用できます。

アプリケーションの完成後は、パッケージ・ウィザードを使用してアプリケーションを Mobile サーバーに配置できます。

2.4.2 パッケージ・ウィザード

パッケージ・ウィザードを使用すると、Web-to-Go アプリケーションの定義を作成または変更して、Mobile サーバー・リポジトリにパブリッシュできます。アプリケーション定義は、アプリケーションのプロパティ（アプリケーションの名前、説明、データベース接続情報など）と同様に、HTML ファイル、データベース・オブジェクトおよび Java サーブレットを指定します。アプリケーション定義は、通常はアプリケーションの開発者が作成します。

システム管理者は、Mobile サーバー・コントロール・センターを使用してアプリケーションを Mobile サーバーの本番システムにインストールします。アプリケーションをパブリッシュした後、管理者は Mobile サーバー・コントロール・センターを使用してアプリケーションのアクセス権をユーザーに割り当てることができます。詳細は、『Oracle9i Lite 管理者およびデブロイ・ガイド』を参照してください。

2.5 アクセス制御の管理

Mobile サーバーは、すべてのユーザーとアプリケーションに対するサーバー側管理を提供します。管理者は、Mobile サーバー・コントロール・センターを使用して、Web-to-Go アプリケーションを管理します。

2.5.1 Mobile サーバー・コントロール・センター

管理者は、Mobile サーバー・コントロール・センターを使用して本番システムにアプリケーションをインストールできます。アプリケーションをパブリッシュすると、管理者は Mobile サーバー・コントロール・センターを使用して、アプリケーションのアクセス権をユーザーに割り当てることができます。

管理者は、Mobile サーバー・コントロール・センターを使用して個別のユーザーまたはグループにアクセス権を付与したり取り消すことによって、アプリケーションに対するアクセス制御を作成および変更できます。また、Mobile サーバー・コントロール・センターを使用して、次のような管理作業を実行できます。

- サーバーのステータス表示
- 非同期レプリケーション・エンジンである MGP の起動、停止およびそのステータスの検証

- サイト情報の表示

また、**Mobile** サーバー・コントロール・センターを使用して、アプリケーションのプロパティを変更できます。

- アプリケーションが **Web-to-Go** 接続プールで管理する接続数の変更
- **Oracle** データベースのログイン・ユーザー名およびパスワードの変更
- クライアントにダウンロードされるデータ・サブセットの決定

詳細は、『**Oracle9i Lite** 管理者およびデプロイ・ガイド』を参照してください。

Web-to-Go アプリケーションの開発

この章では、Web-to-Go アプリケーションの開発方法に関する情報を提供します。内容は次のとおりです。

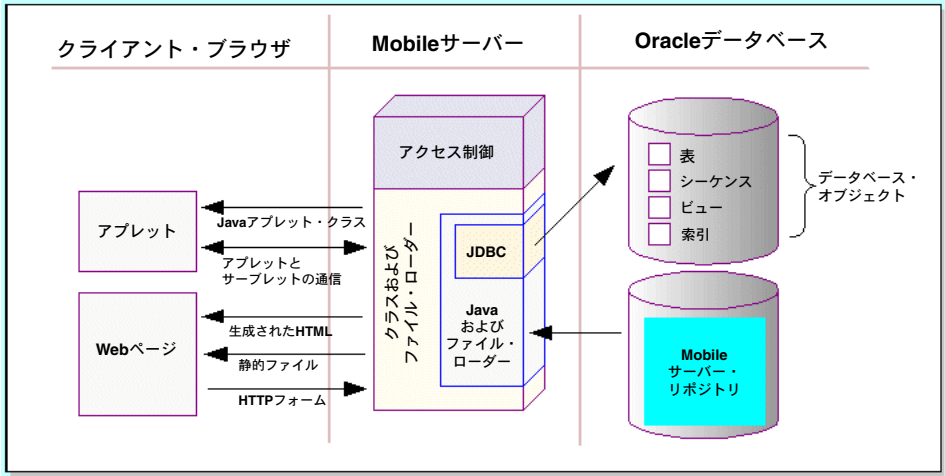
- 3.1 項「概要」
- 3.2 項「Web-to-Go アプリケーションの作成」
- 3.3 項「アプリケーション・ロール」
- 3.4 項「JSP の開発」
- 3.5 項「Web-to-Go 用 Java サブレットの開発」
- 3.6 項「Web-to-Go アプレットの使用」
- 3.7 項「アプレット-JDBC 通信の開発」
- 3.8 項「アプレット-サブレット通信の開発」
- 3.9 項「Web-to-Go アプリケーションのデバッグ」
- 3.10 項「ワークスペース・アプリケーションのカスタマイズ」
- 3.11 項「Mobile Server Admin API の使用」

3.1 概要

Web-to-Go は、Mobile アプリケーションの開発者に使いやすい機能を提供する高水準の Java API を提供します。この API を使用すると、データベース表のレプリケーション、オンラインおよびオフラインのデータベース接続、セキュリティ、ディレクトリ位置、クライアント・デバイスへのアプリケーションの配置などの機能をコーディングする必要がなくなります。

さらに、開発者は Mobile Development Kit (Web-to-Go 用) を使用すると、Java アプレット、Java サーブレットおよび JavaServer Pages (JSP) を含む Web-to-Go アプリケーションの開発とデバッグを実行できます。

図 3-1 開発アーキテクチャ



3.2 Web-to-Go アプリケーションの作成

Web-to-Go アプリケーションは Web 標準に準拠し、ブラウザを使用してフロントエンドの Graphical User Interface (GUI) 要素を表示します。通常、Web-to-Go アプリケーションは、データベース内に格納されているデータにアクセスし、操作を行います。これらのアプリケーションには、静的コンポーネント、動的コンポーネントおよびデータベース・コンポーネントが含まれます。静的および動的コンポーネントは開発ツールを使用して作成し、パッケージ・ウィザードを使用して Mobile サーバー・リポジトリに格納します。アプリケーションのデータベース・コンポーネントは、オブジェクト・リレーショナル・データベース (Oracle Lite または Oracle) に作成して格納します。各コンポーネント・タイプの例を次に示します。

コンポーネント・タイプ	例
静的	HTML ファイル、イメージ・ファイル (GIF や JPG など)、HTML テンプレート
動的	Java サーブレット、Java アプレット、JSP
データベース	表、スナップショット、シーケンス

3.2.1 静的コンポーネント

静的コンポーネントは、グラフィック要素 (GIF ファイルと JPG ファイル) やテキスト要素 (HTML ファイルとテンプレート) などのように変更されることのない HTML ファイルです。

3.2.2 動的コンポーネント

Java アプレット、Java サーブレットおよび JSP は、動的 Web ページを作成する動的コンポーネントです。Java アプレットは豊富な GUI を作成し、Java サーブレットと JSP はサーバー側の機能を継承します。

3.2.2.1 Java アプレット

Java アプレットはブラウザで実行される小規模なアプリケーションで、動的コンテンツを追加することにより HTML ページの機能を継承します。オフライン・モードでは、Java アプレットは Web-to-Go 用 Mobile クライアントを介して Oracle Lite データベースに接続します。オンライン・モードでは、Java アプレットは JDBC Thin ドライバを介して Oracle データベースに直接接続できます。または、Mobile サーバーを介して Oracle データベースに接続します。

3.2.2.2 Java サーブレット

Web-to-Go は、Java サーブレットを使用して動的 Web ページの作成を簡略化します。Java サーブレットはプラットフォームから独立したサーバー側モジュールで、Web サーバーの機能を継承するために使用できます。これらのモジュールは Java で作成されていて、Web サーバーにロードされます。

3.2.2.3 JSP

Web-to-Go は、Java サーブレット・クラス API の拡張である JSP テクノロジーを使用して動的 Web ページを生成します。開発者は、JSP を使用して、基になるコンテンツを変更せずにページのレイアウトを変更します。HTML と Java コードを使用する JSP は、JavaBeans のアクセスを介してプレゼンテーション・コンテンツ (HTML および従来のエディタ) とビジネス・ロジックを融合させます。

3.2.3 データベース・コンポーネント

スナップショットとシーケンスは、Web-to-Go がサポートする 2 つのデータベース・コンポーネントです。スナップショット定義には、スナップショットをとった Mobile サーバー上の表に関する情報が含まれています。さらに、Web-to-Go はカスタム DDL（データ定義言語）文を実行し、ビューや索引などのデータベース・オブジェクトの作成を可能にします。

3.2.3.1 スナップショット

Web-to-Go 用 Mobile クライアントでは、スナップショットをサポートすることによって、接続が切断された状態で表がサポートされます。クライアント側には、各データベース表のスナップショットが作成されます。これらのスナップショットは、クライアントが Mobile サーバーと同期するたびに更新されます。Web-to-Go によってデータの変更が自動的に伝播されます。

スナップショット定義には、クライアントにレプリケートされるデータ量を制限する WHERE 句を含めることができます。これらの定義は、スナップショット・テンプレートと呼ばれます。スナップショット・テンプレートの登録方法およびスナップショット・テンプレートと副問合せスナップショットを使用したアプリケーションの拡張方法の詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

表のスナップショット定義の SQL 文では、CHAR 列を述語として使用しないことをお勧めします。CHAR 列ではなく、VARCHAR2 を使用してください。VARCHAR2 を使用しない場合、データを正常に移行させたり、同期させることができません。バインド変数の CHAR 列が SQL 問合せに使用されているレガシー・データを移行するには、環境を移行する前に、これらの問合せを変更し、これらの問合せに基づいて変更したスナップショット定義をパブリッシュする必要があります。移行の問題のトラブルシューティングの詳細は、『Oracle9i Lite for Windows NT/2000/XP インストールेशनおよび構成ガイド』を参照してください。

3.2.3.2 シーケンス

（多くのデータベース・アプリケーションで使用されている）シーケンスを使用すると、一意識別子を生成できます。また、データベース表への新規レコードの挿入時に、一意の主キー値を生成できます。切断モードのクライアントで表に新しい行が挿入される場合、これが非常に重要になります。レプリケーションでは、主キー値を使用してレコードを識別します。主キーが重複していると、レプリケーションの競合が発生します。このような競合は、管理者が手動で解決する必要があります。このような競合の回避は重要であるため、Web-to-Go には、切断モードで生成されるシーケンス値が必ず一意（有効）になる方式が提供されています。このために、Web-to-Go は切断されている各クライアントに対して、シーケンスごとに一意のシーケンス値ウィンドウを割り当てます。

シーケンス値は、Oracle Lite では -2,147,483,648 ～ 2,147,483,647、Oracle では $-10^{26} \sim 10^{27}$ です。これにより、範囲が大きいシーケンス値を作成できます。たとえば、シーケンス値範囲が 10,000 のユーザーが 10,000 人いる場合でも、100,000,000 個のシーケンス値のみが使用されます。これは、Oracle Lite で使用可能な正数のシーケンス値の約 5% です。詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

3.2.3.3 DDL

Web-to-Go では、表とシーケンスのレプリケーション・サポートに加えて、カスタム DDL (データ定義言語) 文をクライアント上で実行できます。DDL は、ビューやシノニムなどを含むすべてのデータベース・オブジェクトの作成に使用できます。DDL は Web-to-Go アプリケーションの一部で、パッケージ・ウィザードを使用して定義されます。Web-to-Go は、クライアントの最初の同期時に、各クライアントに対して 1 回のみ DDL を実行します。

3.2.3.4 データベース・コンポーネントのアクセス

データベース・コンポーネントには、JDBC 接続を介して Java サーブレットからアクセスできます。Web-to-Go では、JDBC 接続プールを作成してメンテナンスするため、サーブレット・コード内に接続を作成する必要はありません。Web-to-Go をオンライン・モードで実行する場合、接続先は Oracle になります。Web-to-Go を切断モードで実行する場合、接続先は Oracle Lite になります。Web-to-Go は、Java サーブレットの `doPost()` メソッドまたは `doGet()` メソッドを使用して、HTTP 要求を処理する前に接続オブジェクトを `HttpServletRequest` オブジェクトに割り当てます。HTTP 要求が完了すると、接続は自動的に接続プールに戻されます。それ以降の HTTP 要求に同じ接続オブジェクトが割り当てられない場合があるため、サーブレットでは、メソッド `doPost()` または `doGet()` の完了時にトランザクションをコミットするか異常終了させる必要があります。Web-to-Go は、接続オブジェクトを接続プールに戻す前に、保留中のトランザクションをすべて自動的にロールバックします。

3.2.4 データベース接続

データベース接続は、アプリケーション・ベースでもセッション・ベースでもあります。1 つのセッションで、Web-to-Go はアプリケーションごとに 1 つの接続をメンテナンスします。アプリケーションが同時に複数のサーブレットを実行する場合は、複数のサーブレットが同じ接続オブジェクトを使用します。アプリケーションが複数のフレームを使用する場合、または 2 つの異なるブラウザ・ウィンドウを持つアプリケーションにユーザーがアクセスした場合に、これが発生する可能性があります。さらに、同一ブラウザ内の複数のウィンドウは、同一のセッションを共有します。これは同一ブラウザ内の複数インスタンスにもあてはまります。たとえば、同一マシン上で Netscape を 2 回起動しても、セッションは 1 つのみ作成されます。ただし、ユーザーが Netscape Navigator と Internet Explorer の両方を実行するときは、セッションが 2 つ作成されます。

3.3 アプリケーション・ロール

アプリケーションでは、それを実行しているユーザーによって異なる機能を表示するのが一般的です。たとえば、アプリケーションでは、実行者が製造管理者か出荷担当者かによって、異なるメニュー項目を表示できます。

これは、Web-to-Go でアプリケーション・ロールを定義することで実現できます。アプリケーションの動作は、ユーザーが特定のロールを持っているかどうかによって変わります。

前述の例では、アプリケーション・ロール MANAGER を定義できます。メニューを生成するアプリケーション・コード内で、ユーザーがロール MANAGER を持っているかどうかを確認して、正しいメニュー項目を表示する必要があります。

Web-to-Go でアプリケーション・ロールを定義するには、パッケージ・ウィザードを使用します。Mobile サーバー・コントロール・センターを使用して、ユーザーやグループにロールを割り当てます。ただし、ユーザーが特定のロールを持つ場合のアプリケーションの動作を決定し、実装するのは、アプリケーション開発者の責任です。

Web-to-Go ユーザー・コンテキストを問い合わせると、ユーザーのロールのリストを取得できます。

3.4 JSP の開発

Web-to-Go では、JSP のための HTTP 要求を 2 つの方法で処理します。

- [Mobile クライアント Web サーバー](#)
- [Mobile サーバーまたは Web-to-Go 用 Mobile クライアント](#)

3.4.1 Mobile クライアント Web サーバー

Mobile クライアント Web サーバーは JSP 用の HTTP 要求を受け取ると、対応する JSP のソース・ファイルおよびクラス・ファイルが両方とも存在するかどうかを確認します。クラス・ファイルが存在し、JSP ソース・ファイルよりも新しい場合、Mobile クライアント Web サーバーは Java クラスをロードしてサーブレットを実行します。

クラス・ファイルが存在しないか、JSP ソース・ファイルよりも古い場合は、Mobile クライアント Web サーバーは JSP ソース・ファイルを自動的に Java ソース・ファイルに変換し、これを Java クラスとして `APP_HOME/_pages` にコンパイルします。JSP が変換されコンパイルされた後、Mobile クライアント Web サーバーは Java クラスをロードしてサーブレットを実行します。

3.4.2 Mobile サーバーまたは Web-to-Go 用 Mobile クライアント

Mobile サーバーまたは Web-to-Go 用 Mobile クライアントが JSP 用の HTTP 要求を受け取ると、対応する Java クラスが `APP_HOME/_pages` ディレクトリからロードされ実行されます。Web-to-Go 用 Mobile クライアントも Mobile サーバーも、対応するクラス・ファイルが存在していると想定するため、JSP ソース・ファイルをクラス・ファイルに変換する必要があります。さらに、パッケージ・ウィザードを使用してアプリケーションを配置する場合は、JSP ソース・ファイルと対応するクラス・ファイルの両方を含める必要があります。クラス・ファイルは、パッケージ・ウィザード・ツールを使用して作成するか、Oracle JSP (OJSP) のコマンドライン・トランスレータを使用して手動で作成します。

パッケージ・ウィザードの「ファイル」パネルに JSP ファイルを表示します。「ファイル」パネルの「コンパイル」ボタンをクリックします。表示した JSP ファイルは、パッケージ・ウィザードによりすべて検索され、自動的にコンパイルされます。このコンパイル・クラスは、パッケージ・ウィザードによってアプリケーション・パッケージに追加されます。

3.5 Web-to-Go 用 Java サブレットの開発

Web-to-Go Java サブレットは、Mobile Development Kit (Web-to-Go 用) を使用して開発します。Mobile Development Kit (Web-to-Go 用) を使用すると、Web-to-Go サブレットの開発プロセスが簡単になります。Mobile Development Kit (Web-to-Go 用) を使用するには、まずこれを開発クライアントにインストールする必要があります。Mobile Development Kit (Web-to-Go 用) には、Java サブレットを実行する、Mobile クライアント Web サーバーと呼ばれる Web サーバーが含まれています。この Mobile クライアント Web サーバーを使用して、Java サブレットを実行およびデバッグできます。

3.5.1 制限事項

Mobile Development Kit (Web-to-Go 用) Web サーバーは、Mobile サーバーの縮小版で、次の制限事項があります。

- アプリケーション・リポジトリが含まれていません。このため、Mobile Development Kit (Web-to-Go 用) Web サーバーでは、すべてのファイルおよびクラスをファイル・システムから直接ロードします。
- セキュリティおよびアクセス制御は使用できません。
- Mobile Development Kit (Web-to-Go 用) Web サーバーに接続するクライアントは、オフラインに切り替えられません。
- Oracle Lite に対してのみ接続管理を提供します。また、ユーザーを Oracle Lite データベース **webtogo** のスキーマ SYSTEM に接続します。

3.5.2 Mobile Development Kit (Web-to-Go 用) 上のアプリケーションへのアクセス

Mobile Development Kit (Web-to-Go 用) Web サーバー上のアプリケーションには、次の手順でアクセスできます。

Mobile Development Kit (Web-to-Go 用) Web サーバーを起動するには、DOS プロンプトで次のように入力します。

1. `cd <Oracle_Home>%mobile%sdk%bin`
2. `wtgdebug.exe`
3. ブラウザを使用して、Mobile Development Kit (Web-to-Go 用) Web サーバーに接続します。URL は `http://machine_name:7070/` です。アイコンの含まれたページが表示されます。各アイコンは、Mobile クライアント Web サーバーにあるアプリケーションを表します。ポート 7070 は、Web-to-Go のデバッグ用のデフォルト・ポートです。詳細は、`<Oracle_Home>%mobile%sdk%bin%webtogo.ora` にある **webtogo.ora** ファイルを参照してください。
4. アクセスするアプリケーションのアイコンをクリックします。

3.5.3 サープレットの作成

Web-to-Go では、サープレットを使用して HTTP クライアント要求を処理します。サープレットは、次のいずれかを実行して HTTP クライアント要求を処理します。

- 動的 HTML コンテンツを作成し、これをブラウザに返します。
- HTTP POST 要求を使用して HTML フォームを処理しサブミットします。

サープレットでは、Java サープレット API に定義されている `HttpServlet` 抽象クラスを継承する必要があります。サープレットの例を次に示します。

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloWorld extends HttpServlet
{
    /**
     * Process the HTTP POST method
     */

    public void doPost (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        writeOutput("doPost", request, response);
    }
}
```

```
/**
 * Process the HTTP GET method
 */
public void doGet (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    writeOutput("doGet", request, response);
}

/**
 * Write the actual output
 */

public void writeOutput (String method, HttpServletRequest request,
                        HttpServletResponse response)
throws ServletException, IOException
{
    PrintWriter out;

    // set content type
    response.setContentType("text/html");

    // Write the response
    out = response.getWriter();

    out.println("<HTML><HEAD><TITLE>");
    out.println("Hello World");
    out.println("</TITLE></HEAD><BODY>");
    out.println("<P>This is output from HelloWorld "+method+"().");
    out.println("</BODY></HTML>");
    out.close();
}
}
```

3.5.3.1 パッケージ

Web-to-Go では、4 つの Java パッケージを提供します。oracle.html パッケージおよび oracle.lite.html パッケージについては、いずれも [付録 B 「Web-to-Go Java パッケージ」](#) で説明します。oracle.lite.web.servlet パッケージおよび oracle.lite.web.applet パッケージについては、いずれも「[Web-to-Go API Specification](#)」で説明します。パッケージ内に提供されているクラスを使用すると、Java サーブレットの開発が簡単になります。

`oracle.html`: このパッケージには、すべての HTML 要素 (HTML 表とフォーム・ボタン) のクラスが抽象化されて含まれているため、HTML オブジェクトを Java コード内で簡単に作成し操作できます。詳細は、付録 B.1 項「[oracle.html パッケージの使用法](#)」を参照してください。

`oracle-lite.web.html`: このパッケージには、カスタマイズされた Java サブレットを作成するために継承可能なベース・クラスが含まれています。これらのベース・クラスを使用すると、データベース表にリンクされた HTML フォームを簡単に作成できます。データはデータベースから自動的に HTML 形式でロードされます。フォームをサブミットすると、フォーム内のデータの変更がデータベースに自動的に反映されます。詳細は、「Web-to-Go API Specification」の「Using the `oracle-lite.web.html` Package in the Mobile Server API」を参照してください。付録 B.1 項「[oracle.html パッケージの使用法](#)」も参照してください。

`oracle-lite.web.servlet`: サブレットでこのパッケージのクラスを使用すると、ユーザー・プロファイル情報を取得できます。このパッケージは、Java サブレット API の `HttpServletRequest` インタフェースを実装する `OraHttpServletRequest` クラスを定義しています。

`oracle-lite.web.applet`: このパッケージには、Web-to-Go アプレットとともに使用されるクラスが含まれています。このパッケージには、`AppletProxy` クラスが含まれています。このクラスを Web-to-Go アプレットのプロキシとして使用して、JDBC 接続を行うか、Mobile サーバー上のサブレットと通信します。`AppletProxy` クラスが Mobile サーバーとの通信のために使用するクラスもいくつか含まれています。詳細は、「Web-to-Go API Specification」の「`oracle-lite.web.applet`」を参照してください。

3.5.3.2 Web-to-Go のユーザー・コンテキスト

Web-to-Go は、Web-to-Go にログインしているユーザーごとにユーザー・コンテキスト (ユーザー・プロファイル) を作成します。Web アプリケーションは、常にユーザーの特定のコンテキスト内で実行されます。サブレットは、常にアプリケーションの一部で、アプリケーションが実行されるユーザー・コンテキストを使用して、Web-to-Go によって提供されるサービスにアクセスします。ユーザー・コンテキストを使用すると、次の情報を取得できます。

- ユーザーの名前
- ユーザーの実行モード (オンラインまたはオフライン)
- ユーザーがアクセスしているアプリケーション
- データベース接続
- ユーザーがアプリケーションに対して持っているロール
- レジストリでユーザー用に格納されている名前と値のペア

サブレットは、`javax.servlet.http.HttpServletRequest` クラスの `getUserPrincipal` メソッドを介して取得される標準の `java.security.Principal` を使用して、ユーザー・プロファイルにアクセスできます。

このオブジェクトは、JavaSoft 社の定義する

`javax.servlet.http.HttpServletRequest` のサブクラスである `oracle.lite.web.servlet.OraHttpServletRequest` から取得できます。サーブレットでは、要求パラメータの型を `OraHttpServletRequest` オブジェクトにキャストし、`getUserProfile` メソッドをコールしてユーザー・プロフィール・オブジェクトを取得します。たとえば、次のとおりです。

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    // Retrieve the database connection from the User Profile,
    // which can be accessed from the HttpRequest
    OraUserProfile oraUserProfile = ((OraHttpServletRequest)ora_
request).getUserProfile();
    .
    .
    .
}
```

注意： `OraUserProfile` は Web-to-Go の次のリリースでは使用されなくなるため、`java.security.principal` の使用をお勧めします。

3.5.3.3 Java コード内でのデータベース接続

サーブレットは、`UserPrincipal` オブジェクトから、Oracle データベースへの接続を取得できます。

```
Connection conn = ((OraUserProfile) request.getUserPrincipal()).getConnection();
```

3.5.3.4 Mobile サーバー・リポジトリのアクセス

サーブレットは、アプリケーション・リポジトリ内のファイルをオープンしたり、新規ファイルを作成できます。Mobile サーバー・リポジトリへのアクセスは、サーブレット・コンテキストを介して提供されます。サーブレット・コンテキストは、サーブレットの中から `getServletContext()` をコールして取得します。たとえば、次のとおりです。

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    // Retrieve the servlet context
    ServletContext ctxt = getServletContext();

    // Open an input stream to the file input.html in the Mobile Server Repository
    // All file names are relative to the application's repository directory
    InputStream in = ctxt.getResourceAsStream("input.html");
```

```
// Open an output stream to the file output.html in the Mobile Server Repository
// All file names are relative to the application's repository directory
URL          url = ctxt.getResource ("output.html");
URLConnection conn = url.openConnection();
OutputStream out = conn.getOutputStream();
.
.
.
}
```

3.5.4 サブレットの実行

Web-to-Go サブレットを作成した後、そのサブレットを実行します。

3.5.4.1 wtgpack.exe を使用したサブレットの登録

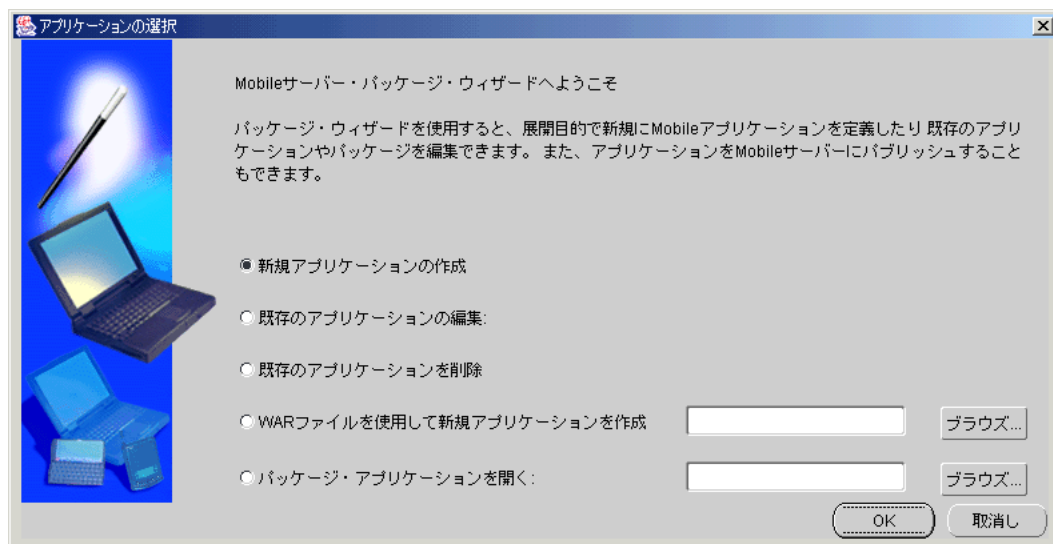
ブラウザからサブレットにアクセスするには、事前にサブレットを Mobile クライアント Web サーバーに登録する必要があります。サブレットに登録するには、まずアプリケーションに登録し、次にこのアプリケーションにサブレットを追加します (Web-to-Go では複数のアプリケーションに登録できます)。ブラウザから Mobile クライアント Web サーバーに接続すると、登録済の全アプリケーションのリストが表示されます。

Mobile Development Kit (Web-to-Go 用) には、パッケージ・ウィザードが含まれています。これはアプリケーションおよびサブレットの登録用ツールです。パッケージ・ウィザードは、コマンドラインで次のように入力して起動します。

```
c:\¥> wtgpack -d
```

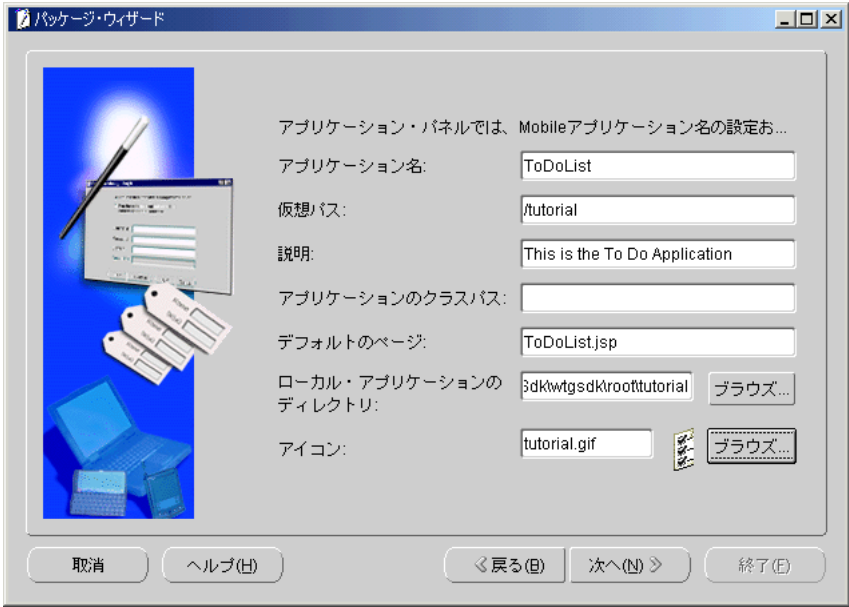
まず、新規アプリケーションを作成するか、既存アプリケーションに対する作業を続行するかを選択します。

図 3-2 「アプリケーションの選択」ダイアログ・ボックス



「OK」をクリックすると、「アプリケーション」パネルが表示されます。

図 3-3 「アプリケーション」 パネル



パッケージ・ウィザードの使用方法の詳細は、第 4 章「Web-to-Go のチュートリアル」を参照してください。

3.5.4.2 webtogo.ora ファイル

Web サーバーとパッケージ・ウィザードの構成情報は、webtogo.ora ファイルに格納されます。

このファイルは、<Oracle_Home>%mobile%sdk%bin ディレクトリにあります。

webtogo.ora ファイルに含まれるパラメータは次のとおりです。

パラメータ	定義
ROOT_DIR	Mobile サーバーは、すべてのファイル・パスをサーバーのルート・ディレクトリに相対的に拡張します。ルート・ディレクトリは、webtogo.ora ファイルにある値 ROOT_DIR を変更すれば変更できます。デフォルト値は、<Oracle_Home>%mobile%sdk%wtgdsdk%root です。
PORT	Web サーバーがリスニングするポートです。デフォルト値は 80 です。Mobile クライアント Web サーバー用のデフォルト値は 7070 です。

パラメータ	定義
XMLFILE	アプリケーション情報を含む XML ファイルです。パッケージ・ウィザードが XML ファイルを作成しメンテナンスします。XML ファイルはパッケージ・ウィザードを使用して変更できます。

詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』の初期化パラメータの説明を参照してください。

3.5.4.3 wtgdebug.exe の使用

次の手順を実行して wtgdebug.exe を起動します。

1. DOS プロンプトで次のように入力します。

```
wtgdebug
```

2. ブラウザを使用して、次の URL にある Mobile クライアント Web サーバーに接続します。

```
http://machine_name:port
```

これで、Mobile クライアント Web サーバーが現在認識しているアプリケーションのリストが表示されます。Mobile クライアント Web サーバーは、このリストを XML ファイルから取り出します。このリストには、デフォルトでは Servlet Runner と Sample というサンプル・アプリケーションが含まれています。

3. デバッグするアプリケーションを選択します。新しいブラウザ・ウィンドウが起動されます。このウィンドウは、アプリケーションをステップ実行するために使用します。

注意： サブレットを変更して再コンパイルする場合は、Web サーバーを再起動する必要があります。Web サーバーは [Ctrl] キーを押しながら [C] キーを押して停止できます。

3.5.4.4 WebToGoServer.class の使用方法

Mobile クライアント Web サーバーは Java で作成されているため、wtgdebug.exe を実行するかわりにサーバーを Java Virtual Machine (JVM) 内で実行することもできます。Mobile クライアント Web サーバーを JVM 内で実行すると、Java デバッガ内で Mobile クライアント Web サーバーを実行して Web-to-Go アプリケーションをデバッグできます。Mobile クライアント Web サーバーの Java 版の起動には、`oracle.lite.web.server.WebToGoServer` クラスを使用できます。

Java 版の Mobile クライアント Web サーバーを使用するには、その前に CLASSPATH に次のファイルを追加する必要があります。

```
<Oracle_Home>%mobile%sdk%bin%webtogo.jar
<Oracle_Home>%mobile%classes%olite40.jar
<Oracle_Home>%mobile%classes%xmlparser.jar
<Oracle_Home>%mobile%classes%classgen.jar
<Oracle_Home>%mobile%classes%ojsp.jar
<Oracle_Home>%mobile%classes%jssl-1_2.jar
<Oracle_Home>%mobile%classes%javax-ssl-1_2.jar
<Oracle_Home>%mobile%classes%consolidator.jar
```

CLASSPATH には、<Oracle_Home>%mobile%sdk%wtgsdk%root などのアプリケーション・クラスの位置を追加する必要があります。第 4 章「[Web-to-Go のチュートリアル](#)」の「[ステップ 5: アプリケーションの実行](#)」を参照してください。

ファイル **RunWebServer.java** は、クラス `oracle.lite.web.server.WebToGoServer` を使用して Mobile クライアント Web サーバーを起動し制御する方法を示します。このファイルは、次のディレクトリにあります。

```
<Oracle_Home>%mobile%sdk%wtgsdk%src
```

Mobile クライアント Web サーバーを起動するには、次の手順を実行します。

1. 次のコマンドを使用して Java ファイルをコンパイルします。

```
javac RunWebServer.java
```
2. 次のコマンドを使用して Mobile クライアント Web サーバーを実行します。

```
java RunWebServer
```

3.5.4.5 Web サーバーのプロパティの制御

Mobile クライアント Web サーバーの様々なプロパティを、`WebToGoServer.setProperty()` メソッドを使用して動的に設定することができます。これらの値は、**webtogo.ora** の値を上書きします。制御可能なプロパティは、次のとおりです。

プロパティ	定義
config_file	使用する構成ファイルです。「webtogo.ora ファイル」を参照。
port	Mobile クライアント Web サーバーがリスニングするポートです。
debug	デバッグを使用可能にします。デバッグ・メッセージを表示する場合は、値を「0」に設定します。
log_file	デバッグ用のログ・ファイルです。指定すると、デバッグ・メッセージがこのファイルに送信されます。指定しないと、メッセージが画面に表示されます。
root_dir	ルート・ディレクトリです。webtogo.ora にある ROOT_DIR をオーバーライドします。ROOT_DIR を参照してください。

たとえば、次のとおりです。

```
WebToGoServer.setProperty ("config_file",
                             "d:¥¥orant¥¥mobile¥¥server¥¥bin¥¥webtogo.ora");
WebToGoServer.setProperty ("debug", "0");
WebToGoServer.setProperty ("port", "80");
```

3.5.4.6 サブレットの登録

サブレットは、パッケージ・ウィザードを使用しないで動的に追加できます。サブレット・クラス HelloWorld を Mobile クライアント Web サーバーに登録するには、次の Java コードを使用します。

```
WebToGoServer.addServlet("HelloWorld", "/Hello");
```

デフォルトのアプリケーション「Servlet Runner」にこのサブレットを追加し、次の URL を入力するとサブレットにアクセスできます。

http://machine_name/servlets/Hello

このサブレットは次の HTML を返します。

```
<HTML><HEAD><TITLE>
Hello World
</TITLE></HEAD><BODY>
<P>This is output from HelloWorld doGet().
</BODY></HTML>
```

3.5.4.7 MIME タイプの登録

特定のファイル拡張子を持つファイルに対する HTTP 要求をすべて処理する、独自のサブレットを作成できます。たとえば、「asp」で終わる要求をすべて処理する ASPHandler というサブレットを作成できます。

このハンドラは、メソッド `WebToGoServer.addMIMEHandler()` を使用して Mobile クライアント Web サーバーに登録できます。たとえば次のように指定します。

```
WebToGoServer.addMIMEHandler("text/asp", "asp", "ASPHandler")
```

3.5.4.8 名前と値のペアの登録

Web-to-Go は、オブジェクトを永続的に格納するために使用可能なレジストリを提供します。このようなオブジェクトは、メソッド `OraUserProfile.getValue()` を使用してサブレットのコード内で取得できます。Mobile サーバーは、レジストリ・オブジェクトをアクセス制御システム内に格納します。Mobile クライアント Web サーバーの場合は、メソッド `WebToGoServer.addRegistryEntry()` を使用してこれらのレジストリ・オブジェクトを動的に設定できます。これによって、レジストリを使用するアプリケーションをテストできます。たとえば、次のとおりです。

```
// Add a registry name/value pair to the default application "servletRunner"
WebToGoServer.addRegistryEntry ("usercode", "1111");
// Add a registry name/value pair to the specified application.
WebToGoServer.addRegistryEntry ("TESTAPPLICATION", "code", "112");
```

3.5.5 サブレットのデバッグ

ソフトウェア開発では、デバッガを使用して、コードを調べ、不具合を修正する場合があります。Web-to-Go では、Java サブレットを含むアプリケーションのテストにデバッガを使用できます。Java デバッガ内で Java サブレットを実行することによって、Java コード内にブレークポイントを設定し、コードを表示し、スレッドを調べ、オブジェクトを評価できます。デバッガ内で `WebToGoServer` クラスを使用することによって、Web-to-Go アプリケーションをデバッグできます。詳細は、「[Oracle9i JDeveloper を使用したサンプル 1 の実行](#)」を参照してください。

3.5.6 Oracle Lite 内のスキーマの直接アクセス

Mobile Development Kit (Web-to-Go 用) は、Oracle Lite へのデータベース接続を自動的に作成します。このデータベース接続により、データベース・スキーマ SYSTEM に接続されます。サブレット・コード内で、HTTP 要求からこの接続を取得できます。詳細は、「[データベース・コンポーネントのアクセス](#)」を参照してください。Oracle Lite データベースへは、ODBC を使用して直接接続することもできます。ODBC を使用して Oracle Lite データベースに直接接続すると、次の作業の実行に役立ちます。

- 表、ビューおよびシーケンスなどのスキーマ・オブジェクトの作成
- 手動による表の内容の検査

Oracle Lite に接続するには、DOS プロンプトで次の構文を入力して `msql` を起動します。

```
msql system/x@jdbc:polite:webtogo
```

3.6 Web-to-Go アプレットの使用

Web-to-Go は Java アプレットをサポートします。セキュリティ上の理由から、Web-to-Go アプレットでは Mobile サーバーまたは Oracle データベースとの接続にプロキシ・クラスを使用する必要があります。AppletProxy クラスが Web-to-Go アプレットのプロキシとして機能し、Web-to-Go サブレットとの通信または JDBC との接続に必要なメソッドをアプレットに対して提供します。AppletProxy のインスタンスは、アプレットのインスタンス化中に作成されます。AppletProxy クラスのインスタンスが作成されると、AppletProxy オブジェクトが Mobile サーバーと通信して、サーバーとの接続または Oracle データベースとの JDBC 接続の確立に必要なすべての情報を導出します。

3.6.1 Web-to-Go アプレットの作成

Web-to-Go アプレットは、`java.applet.Applet` を継承したものです。`init()` メソッドで Web-to-Go アプレットを初期化する場合は、アプレット参照をパラメータとして渡すことによって、AppletProxy クラスのインスタンスを作成します。AppletProxy クラスのインスタンスが生成されると、AppletProxy クラスの別のメソッドを使用して、サブレットと通信したり Oracle データベースとの JDBC 接続を確立できます。たとえば、次のとおりです。

```
import oracle.lite.web.applet.*;
public class AppApplet extends Applet
{
    public void init()
    {
        ..
        ..
        // Create Instance and pass Reference of applet as parameter
        proxy = new AppletProxy(this);
    }
    AppletProxy proxy;
}
```

アプレットでは、サブレットとの通信に次のメソッドを使用できます。各メソッドには、AppletProxy クラスのインスタンスが必要です。

- [getResultObject\(\)](#)
- [setSessionId\(\)](#)
- [showDocument\(\)](#)

アプレットでは、データベースとの JDBC 接続の確立に `getConnection()` メソッドを使用できます。

3.6.2 アプレット用の HTML ページの作成

Web-to-Go アプレットは、次のタグを含む HTML ページから起動されます。

```
<html>
<body>
<applet ARCHIVE="/webtogo/wtgapplet.jar" CODE="MyApplet.class" WIDTH=200 HEIGHT=100>
<PARAM NAME="ORACLE_LITE_WEB_SESSION_ID" VALUE="123">
</applet>
</body>
</html>
```

AppletProxy クラスは、ORACLE_LITE_WEB_SESSION_ID パラメータの値を使用して、Mobile サーバーからセッション ID を取得します。その後、アプレットからサブレットへのすべての要求にこのセッション ID が追加されます。HTML コードは静的 HTML ページ内に作成することも、サブレットから生成することもできます。

3.6.2.1 静的 HTML ページ

Web-to-Go は、APPLET タグを含む静的ページに対して、自動的にパラメータを追加できます。このオプションを実行する場合は、次の構文に示されているように、HTML ページの拡張子を `.ahtml` に変更する必要があります。

`page_name.ahtml`

クライアントがこの HTML ページにアクセスすると、Web-to-Go システム・サブレットによって ORACLE_LITE_WEB_SESSION_ID パラメータに必要な `<PARAM>` タグが HTML に追加されます。たとえば、次のとおりです。

```
<PARAM NAME="ORACLE_LITE_WEB_SESSION_ID" VALUE="123">
```

Web-to-Go システム・サブレットは、VALUE 属性を Web-to-Go セッション ID に設定します。

3.6.2.2 サブレットから生成される HTML ページ

`<APPLET>` タグを含む HTML ページは、動的に生成することもできます。HTML ページを動的に生成する場合は、セッション ID パラメータを手動で追加する必要があります。セッション ID 情報は、次のとおり入力して `oraUserProfile` から取得できます。

```
import oracle.lite.web.html.*;
import oracle.lite.web.servlet.*;

public class AppServlet extends HttpServlet
{
```

```

public void doGet(HttpServletRequest req, HttpServletResponse resp)
{
    PrintWriter out = new PrintWriter(resp.getOutputStream());
    out.println("<HTML>");
    out.println("<BODY>");
    out.println("<APPLET ARCHIVE=\"/webtogo/wtgapplet.jar"
                CODE='MyApplet.class' WIDTH=200 HEIGHT=100>");
    // Add these lines to add one more PARAM tag in html page
    // This code should be added in-between <APPLET> and </APPLET> tag
    OraHttpServletRequest ora_request = (OraHttpServletRequest) req;
    OraUserProfile         oraUserProfile = ora_request.getUserProfile();
    out.println(" <PARAM NAME=¥"ORACLE_LITE_WEB_SESSION_ID¥" VALUE=¥"
                +oraUserProfile.getAppletSessionId(req)+"¥"> ");
    out.println("</APPLET>");
    out.println("</BODY>");
    out.println("</HTML>");
    out.close();
}
}

```

3.7 アプレットー JDBC 通信の開発

データベースにアクセスする Java アプレットは、JDBC 接続を使用して開発できます。AppletProxy クラスのインスタンスが生成されると、AppletProxy クラスの getConnection() メソッドを使用して、JDBC 接続を取得する必要があります。getConnection() メソッドは、JDBC 接続オブジェクトを返します。

注意： AppletProxy クラスの説明は、[「Web-to-Go アプレットの作成」](#)にあります。

3.7.1 getConnection()

getConnection() メソッドは、JDBC 接続の取得に使用できます。getConnection() メソッドは接続モードがオンラインかオフラインかを判断し、正しいデータベース接続（オンライン・モードの場合は Oracle データベース、オフライン・モードの場合は Oracle Lite）をユーザーに提供します。たとえば、次のとおりです。

```

import oracle.lite.web.applet.*;
public class AppApplet extends Applet
{
    public void init()
    {
        ..
        ..
    }
}

```

```
// Create Instance and pass Reference of applet as parameter
proxy = new AppletProxy(this);
}
public java.sql.Connection getDataBaseConnection()
{
    java.sql.Connection dBConnection = proxy.getConnection();
    return dBConnection;
}
AppletProxy proxy;
}
```

3.7.2 設計上の課題

Web-to-Go アプレットは、ユーザーが Web-to-Go を終了した後もデータベース接続を保持します。このアプレットは、ユーザーがブラウザの「アドレス」ウィンドウに新しい URL を入力するか、「戻る」ボタンをクリックした後も接続を保持します。Web-to-Go アプリケーションの設計者は、作成するアプリケーションが、ユーザーが Web-to-Go を終了したときにデータベース接続を明示的にクローズするように保証する必要があります。たとえば、前述のコード・サンプルでは、参照される `dBConnection.close()` メソッドをコールして接続をクローズできます。

`dBConnection.close()` method referenced in the above code sample.

3.8 アプレット-サーバレット通信の開発

Web-to-Go 環境では、Java サーバレットと通信する Java アプレットを開発できます。クライアントが初めて Mobile サーバーに接続する場合、サーバーはセッション ID を生成し、これをクライアントに送り返します。これ以降のサーバーへのクライアント要求にはこのセッション ID が含まれます。Mobile サーバーは、クライアントの要求を実行する前にセッション ID を認証します。アプレットが Web-to-Go サーバレットと通信する場合は、各アプレット要求にこのセッション ID が含まれている必要があります。各アプレット要求にセッション ID を追加するには、`AppletProxy` クラスの `SetSessionID` メソッドを使用できます。`AppletProxy` クラスには、アプレットとサーバレット間の通信を提供するその他のメソッドも含まれています。

注意： Java サーバレットと通信するには、`getResultObject()` メソッドおよび `showDocument()` メソッドを使用できます。独自の URL 接続オブジェクトを作成する場合は、`setSessionID` メソッドを使用します。

3.8.1 Web-to-Go サーブレットの作成

サーブレットでは、Java サーブレット API に定義されている `HttpServlet` 抽象クラスを継承する必要があります。次の例では、`HttpServlet` クラスを継承する `HelloWorld` というサーブレットを作成します。このサーブレットは、文字列をオブジェクトとしてコールするアプレットに対して「Hello World」という文字列を送信します。

```
public class HelloWorld extends HttpServlet
{
    public void doGet (HttpServletRequest request, HttpServletResponse response)
    {
        ObjectOutputStream out = new ObjectOutputStream (resp.getOutputStream());
        Object obj = (Object) "Hello World" ;
        out.writeObject(obj);
        out.close();
    }
}
```

3.8.1.1 getResultObject()

Web-to-Go アプレットは、`getResultObject` メソッドを使用して Web-to-Go サーブレットと通信します。このとき、サーブレットの URL および `ServletParameter` オブジェクトをパラメータとして渡します。サーブレットは、テキスト文字列を使用してアプレット要求に応答します。`ServletParameter` オブジェクトは、シリアル化可能なオブジェクト、または名前と値のペアを含む文字列です。サーブレットがパラメータを受け入れた場合、`getResultObject` メソッドをコールしてサーブレット・パラメータを引数の 1 つとして渡すことができます。たとえば、次のとおりです。

```
public Object getResult()
{
    java.net.URL url = new URL("http://www.foo.com/EmpServlet");
    String ServletParameter = "empname=John";
    Object resultObject = proxy.getResultObject(url, ServletParameter);
    return resultObject;
}
```

3.8.1.2 setSessionId()

`setSessionId` メソッドは、既存の `URLConnection` オブジェクトにセッション ID を追加するために使用できます。アプレット - サーブレット通信メカニズムを作成する場合は、メソッドの最後に `setSessionID (URLConnection)` をコールします。このメソッドは、渡された `URLConnection` オブジェクトにセッション ID を追加してから、`URLConnection` オブジェクトを返します。たとえば、次のとおりです。

```
public void YourMethod()
{
    java.net.URL url = new URL("http://www.foo.com/MyServlet");
    java.net.URLConnection con = url.openConnection();
```

```
..
..
..
// pass the URLConnection to the method setSessionId
con = proxy.setSessionID(con);
// Do whatever you want to do with this URLConnection object
ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream());
out.writeObject(obj);
out.flush();
out.close();
}
```

3.8.1.3 showDocument()

showDocument メソッドは、静的ドキュメント（拡張子が **.html**、**.doc**、**.xls**、またはその他のユーザー定義拡張子を持つドキュメントを含む）を表示します。showDocument メソッドは、これらのドキュメントを Mobile サーバーから取り出して、クライアント・ブラウザに表示します。ドキュメントを表示するには、ユーザーがドキュメントのアクセス権を所有し、Mobile サーバーに正しい MIME タイプが設定されている必要があります。

showDocument(String relativeDocUrl, String winName) メソッドは、winName パラメータで渡されるウィンドウ名を持つ別のブラウザ・ウィンドウにドキュメントを表示します。次のメソッドは、サーバーからヘルプ・ファイルを起動し、「helpWin」という名前のブラウザ・ウィンドウに表示します。

```
public void showHelp()
{
    String relativeDocUrl = "Help/HelpIndex.html";
    proxy.showDocument (url, helpWin);
}
```

アプレットが使用するブラウザ・ウィンドウにドキュメントを表示するには、次のように showDocument(url) をコールします。

```
public void showHelp()
{
    String relativeDocUrl = "Help/HelpIndex.html";
    proxy.showDocument (url);
}
```

3.9 Web-to-Go アプリケーションのデバッグ

Mobile Development Kit (Web-to-Go 用) および Java デバッガ (Oracle9i JDeveloper、Borland 社の JBuilder、Visual J++ など) をすでにインストールしてある場合は、Java デバッガ内で Web-to-Go アプリケーションを実行できます。この項の例では、Oracle9i JDeveloper を想定していますが、ほとんどの情報は他のデバッガにも当てはまります。

3.9.1 Oracle9i JDeveloper を使用したサンプル 1 の実行

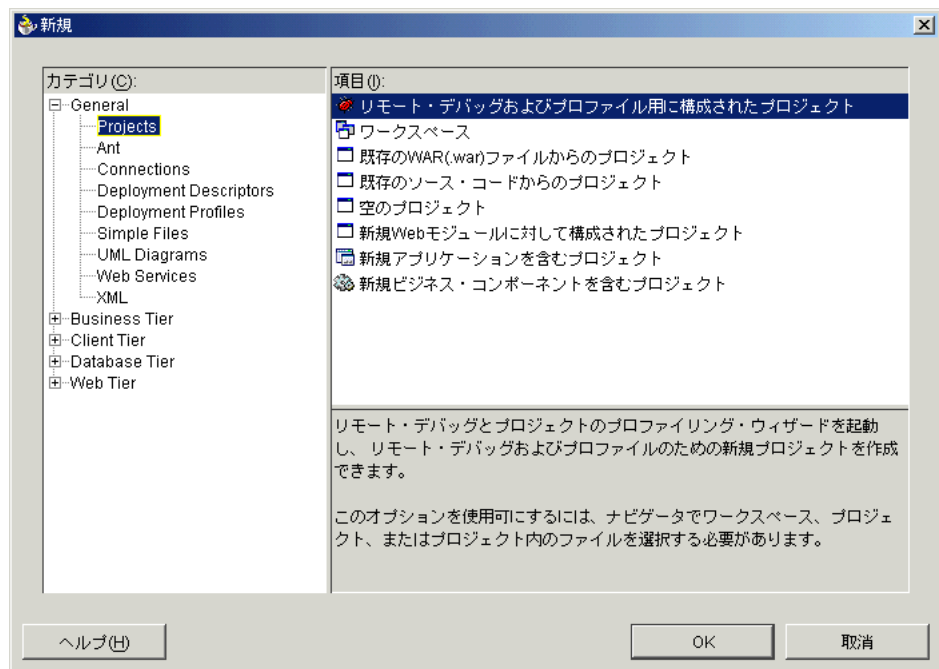
ここでは、Mobile Development Kit (Web-to-Go 用) に付属する Sample1 アプリケーションを実行できるように Oracle9i JDeveloper を構成する方法を説明します。Oracle9i JDeveloper の使用方法の詳細および使用方法が説明されているドキュメントは、Oracle9i JDeveloper のオンライン・ヘルプおよび Oracle9i JDeveloper のドキュメントを参照してください。

3.9.1.1 デバッグ・プロジェクトの作成

Oracle9i JDeveloper で新規デバッグ・プロジェクトを作成するには、次の手順を実行します。

1. Oracle9i JDeveloper を起動します。
2. Oracle9i JDeveloper で新規プロジェクトを作成するために、「File」>「New」を選択します（ただし、ワークスペースが Oracle9i JDeveloper で定義済であることを前提とします）。
3. 左側のパネルの「Directories」メニューから、「Projects」を選択してから（図 3-4）、「空のプロジェクト」を選択します。

図 3-4 新規プロジェクトの作成

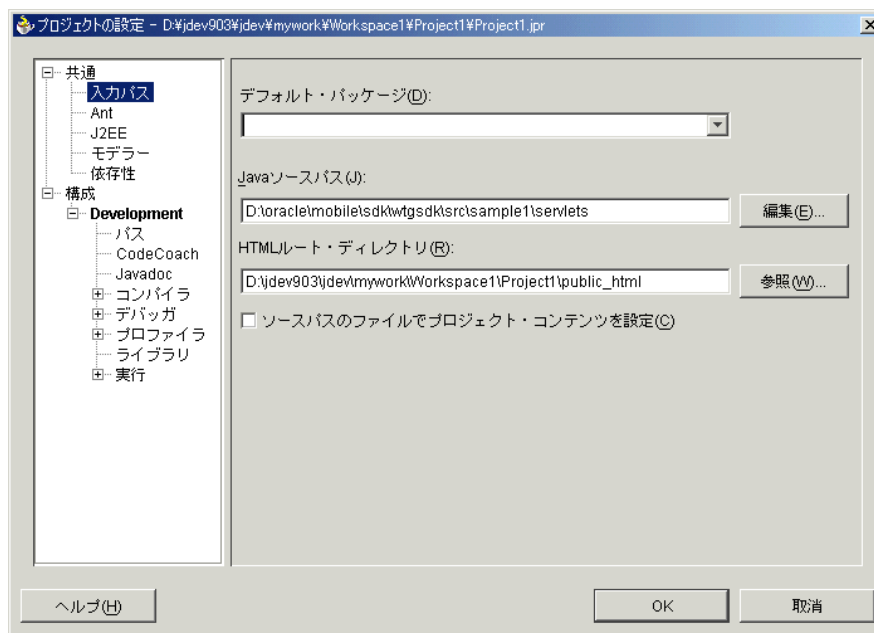


- 新規プロジェクトに対応するようにプロジェクト設定を設定します。「Projects」を右クリックして、プロジェクト設定を取得します。「プロジェクトの設定」ダイアログ・ボックスの左側のパネルで、「共通」を開いて「入力パス」を選択します。図 3-5 に示されているように、右側のパネルの「Java ソース・パス」フィールドに次の情報を入力します。

```
<ORACLE_HOME>%mobile%sdk%wtg%src%sample1%servlets
```

「デフォルト・パッケージ」フィールドは空白のまま残します。デフォルトの HTML ルート・ディレクトリは変更しないでください。

図 3-5 プロジェクト設定—入力パス



- 左側のパネルで「構成」を開いてから、「Development」を開きます。左側のパネルの「Development」の下に表示されている「パス」を選択します。右側のパネルの「Output Directory」フィールドに次の情報を入力します。

```
<ORACLE_HOME>%mobile%sdk%wtg%root%sample1%servlets
```

3.9.1.2 ライブラリの作成

Oracle9i JDeveloper では、CLASSPATH を設定するかわりにライブラリを使用します。これによって、一連の .jar ファイルの管理が簡単になります。

WTGSDK ライブラリ用ファイル

次の .jar ファイルを持つ WTGSDK ライブラリを作成し、このライブラリをプロジェクトに追加します。

<Oracle_Home>%mobile%classes%ojsp.jar

<Oracle_Home>%mobile%classes%olite40.jar

<Oracle_Home>%mobile%sdk%bin%webtogo.jar

<Oracle_Home>%mobile%classes%servlet.jar

<Oracle_Home>%mobile%classes\$xmlparser.jar

<Oracle_Home>%mobile%classes%classgen.jar

<Oracle_Home>%mobile%classes%wtgpack.jar

<Oracle_Home>%mobile%classes%jssl-1_2.jar

<Oracle_Home>%mobile%classes%javax-ssl-1_2.jar

WTGSDK ライブラリの作成方法

WTGSDK ライブラリを作成するには、次の手順を実行します。


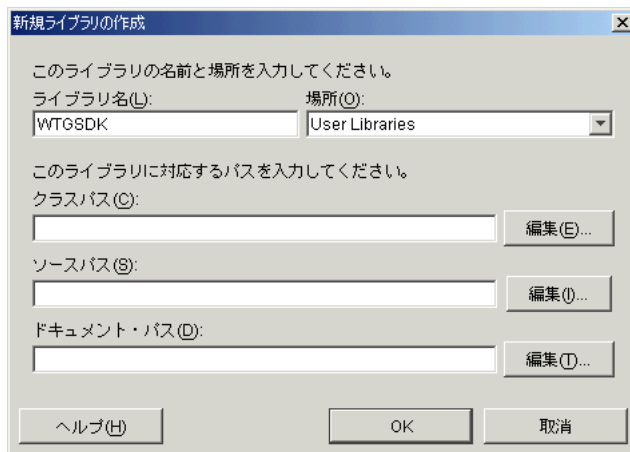
1. 左側のパネルで「ライブラリ」を選択してから、右側のパネルの「新規」をクリックします。
2.  図 3-6 に示すように、「新規ライブラリの作成」ダイアログ・ボックスが表示されます。「ライブラリ名」フィールドに、WTGSDK と入力します。

図 3-6 「新規ライブラリの作成」ダイアログ・ボックス



3. フィールド「クラスパス」フィールドの隣の「編集 ...」ボタンをクリックします。「File」ダイアログ・ボックスが表示されます。
4. 適切なディレクトリから、前述の 9 つの **.jar** ファイルを選択します。
5. 「OK」をクリックしてファイルを追加します。
6. 「OK」をクリックしてライブラリを作成します。

3.9.1.3 プロジェクトへのファイルの追加

Sample1 のファイルをプロジェクトに追加するには、次の手順を実行します。

1. Oracle9i JDeveloper システムのナビゲータで緑色の正符号 (+) をクリックして、Java ソースをプロジェクトに追加します。「File」ダイアログ・ボックスが表示されます。
2. ディレクトリ `<Oracle_Home>%mobile%sdk%wtgsrc%sample1%servlets` にある Java ソース・ファイル **HelloWorld.java** を選択し、「Open」をクリックします。
3. ディレクトリ `<Oracle_Home>%mobile%sdk%wtgsrc` にあるファイル **RunWebServer.java** もプロジェクトに追加します。
4. プロジェクト・ソースのパスを更新するかどうかを尋ねるダイアログ・ボックスが表示されるため、「No」をクリックします。

3.9.1.4 実行とデバッグ

コード内のブレークする文をマウスの右ボタンでクリックして、ブレークポイントを1つ以上設定します。「Toggle breakpoint」を選択します。文の背景が赤になり、ブレークポイントが設定されていることを示します。

1. システム・ナビゲータ・ウィンドウで、ファイル **RunWebServer.java** を選択します。
2. 選択したファイルを右クリックして「Debug」を選択し、デバッガ内で Mobile サーバーを起動します。

これで Mobile サーバーを使用する準備ができました。Web ブラウザから次の URL にアクセスして、Web-to-Go サーバーにアクセスできます。

`http://<machine_name>/`

ここで、<machine_name> は、Oracle9i JDeveloper を実行しているコンピュータのホスト名です。

3.9.1.5 トラブルシューティング

Mobile サーバーを Java デバッガ内で実行し、Web ブラウザを使用してアクセスする場合は、パフォーマンスが低下する可能性があります。次の操作を実行すると、パフォーマンスを向上できます。

- Web ブラウザを別マシン上で実行します。
- (Web ブラウザを起動した後) タスク・マネージャを使用して Web ブラウザ・プロセスの優先順位を「低」に設定します。

3.10 ワークスペース・アプリケーションのカスタマイズ

Mobile Development Kit (Web-to-Go 用) には、基本的な Web-to-Go ワークスペース・アプリケーションを構成する一連の API が含まれています。開発者はこれらの API を使用して、標準の Web-to-Go ワークスペース・アプリケーションを、カスタマイズしたものに置き換えることができます。これらの API が提供する機能には、次のものがあります。

- ログイン
- ログオフ
- 同期
- ユーザー・アプリケーションのリスト
- ユーザーのパスワードの変更

カスタマイズした Web-to-Go ワークスペース・アプリケーションの作成に使用する API の詳細は、次のディレクトリにある「Web-to-Go API Specification」を参照してください。

```
<Oracle_Home>%mobile%doc%javadoc%wtg
```

カスタマイズした Web-to-Go ワークスペース・アプリケーションを開発した後、開発者は、**webtogo** と呼ばれる Oracle Lite データベースを作成して、新しく作成した Web-to-Go ワークスペース・アプリケーションをこのデータベースにロードする必要があります。このデータベースは、Web-to-Go 用 Mobile クライアントの Mobile サーバー・リポジトリとして機能します。詳細は、サンプル Web-to-Go ワークスペース・アプリケーションに付属のファイル **crclient.bat** を参照してください。

次に、開発者は Web-to-Go 用 Mobile クライアント用の **webtogo.ora** ファイルを作成する必要があります。このファイルは、カスタマイズした Web-to-Go ワークスペース・アプリケーションを使用するように Mobile サーバーに指示します（**webtogo.ora** ファイルの正しいパラメータ設定値は、「[Webtogo.ora パラメータ](#)」を参照してください）。

次に、開発者は、Web-to-Go 用 Mobile クライアントによって作成された **webtogo.odbc** ファイル、Web-to-Go 用 Mobile クライアント用の **webtogo.ora** ファイルおよび Web-to-Go ワークスペース自体を Mobile サーバー・リポジトリにロードする必要があります。詳細は、サンプル Web-to-Go ワークスペース・アプリケーションに付属のファイル **crserver.bat** を参照してください。

次に、管理者は、サーバー上の **webtogo.ora** ファイルを変更して、Mobile サーバーに新しい Web-to-Go ワークスペース・アプリケーションを使用するよう指示する必要があります（**webtogo.ora** ファイルの正しいパラメータ設定値は、「[Webtogo.ora パラメータ](#)」を参照してください）。

3.10.1 Webtogo.ora パラメータ

カスタマイズした Web-to-Go ワークスペース・アプリケーションを使用するように Web-to-Go に指示するには、**webtogo.ora** ファイルの [WEBTOGO] セクションに次のパラメータを設定する必要があります。

パラメータ	設定
CUSTOM_WORKSPACE	YES
CUSTOM_DIRECTORY	Web-to-Go ワークスペース・アプリケーションのリポジトリ・ディレクトリ。たとえば、次のとおりです。 /myworkspace
DEFAULT_PAGE	Web-to-Go ワークスペース・アプリケーションのエントリ・ポイント。たとえば、次のとおりです。myfirstpage.html
CUSTOM_FIRSTSERVLET	カスタマイズしたワークスペースで使用するサーブレットの名前。たとえば、次のとおりです。 CUSTOM_FIRSTSERVLET= HelloWorld;/hello

注意： Web-to-Go のサポートするワークスペース・アプリケーションは、Mobile サーバー当たり 1 つのみです。

3.10.2 サンプル・ワークスペース

Mobile Development Kit (Web-to-Go 用) には、Web-to-Go Workspace API の使用方法を示すサンプル Web-to-Go ワークスペース・アプリケーションが含まれています。開発者は、自分の Web-to-Go ワークスペース・アプリケーションを開発する場合、出発点としてこのサンプル・アプリケーションを使用できます。サンプル Web-to-Go ワークスペース・アプリケーションは、JSP および **.html** ファイルを使用して作成されています。JSP ファイルは、Mobile Development Kit (Web-to-Go 用) の **myworkspace/src** ディレクトリにあります。これらのファイルは、クラス・ファイルにコンパイルされ、**myworkspace/out** ディレクトリにコピーされます。このディレクトリには、サンプル Web-to-Go ワークスペース・アプリケーションによって使用されるすべての **.html** ファイルおよびイメージ・ファイルも含まれています。

Mobile Development Kit (Web-to-Go 用) には、JSP ファイルのコンパイル、Web-to-Go 用 Mobile クライアント用 Oracle Lite データベース **webtogo** の作成、Mobile サーバー・リポジトリへのすべての必要なファイルのロードに使用する次のスクリプトが含まれています。

スクリプト名	説明
compile.bat	.jsp ファイルをコンパイルし、クラス・ファイルを myworkspace/out ディレクトリにコピーします。
crclient.bat	myworkspace/out ディレクトリのすべてのファイルを webtogo.odb ファイルにコピーします。
crserver.bat	myworkspace/webtogo ディレクトリのすべてのファイルを、Mobile サーバー・リポジトリにコピーします (webtogo.odb ファイルおよび webtogo.ora ファイルを含む)。

3.11 Mobile Server Admin API の使用

Mobile Server Admin API を使用すると、管理者はアプリケーション・リソースをプログラムで管理できます。管理者は Mobile Server Admin API セットを使用して、カスタマイズされた独自のコントロール・センター・アプリケーションを作成し、次のような操作を実行することも可能です。

- ユーザーおよびユーザー・グループの作成および変更
- ユーザーおよびグループに対するアプリケーション・アクセス権の付与および取消し
- ユーザーに対するアプリケーション・ロールの付与および取消し
- アプリケーションに対するグループ・レベルのアクセス権へのユーザーの挿入および除外
- ユーザーへのスナップショット変数の割当て
- アプリケーションの一時停止および再開
- 事前にパッケージ化された Web-to-Go アプリケーションのパブリッシュ
- アプリケーションの基盤となるデータベース接続のカスタマイズ

この API を使用したコントロール・センターの作成方法の詳細は、次のディレクトリにある「Web-to-Go API Specification」を参照してください。

<Oracle_Home>\mobile\doc\javadoc\wtg

注意： 管理者は、スナップショット定義やサーブレットなどのアプリケーションの基本的なプロパティを変更する目的でオープン API セットを使用することはできません。これは、パッケージ・ウィザードを介してのみ可能です。詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

Web-to-Go のチュートリアル

この章では、Web-to-Go アプリケーションの実装フェーズを順番に説明します。内容は次のとおりです。

- [4.1 項「概要」](#)
- [4.2 項「アプリケーションの開発」](#)
- [4.3 項「アプリケーションのパッケージ化」](#)
- [4.4 項「アプリケーションのパブリッシュ」](#)
- [4.5 項「アプリケーションの管理」](#)
- [4.6 項「Web-to-Go 用 Mobile クライアントでのアプリケーションの実行」](#)

このチュートリアルは、概要および5つの項で構成されていて、それぞれに項目が含まれています。各項は、「To Do List」アプリケーションのライフ・サイクルにおける各フェーズを表しています。各項の完了後、その内容を復習するか、関連ドキュメントを参照するか、または次に進むことができます。このチュートリアルの内容は、次のとおりです。

表 4-1 チュートリアルの概要

項	説明
概要	このチュートリアルの目的および使用方法を説明します。
アプリケーションの開発	Mobile Development Kit（Web-to-Go 用）を使用して「To Do List」アプリケーションのコンポーネントを作成しテストする方法を説明します。
アプリケーションのパッケージ化	「To Do List」アプリケーションのコンポーネントをパッケージ化する方法を説明します。
アプリケーションのパブリッシュ	アプリケーションをパッケージ化した後で Mobile サーバーにパブリッシュする方法を説明します。

表 4-1 チュートリアル概要（続き）

項	説明
アプリケーションの管理	「To Do List」アプリケーションの管理方法を説明します。
Web-to-Go 用 Mobile クライアントでのアプリケーションの実行	Web-to-Go 用 Mobile クライアントをインストールして「To Do List」アプリケーションを使用する方法を説明します。

4.1 概要

このチュートリアルでは、簡単な「To Do List」アプリケーションの作成、配置および管理方法を示すことによって、Web-to-Go アプリケーションの実装プロセスをガイドします。「To Do List」アプリケーションの項目は、すべてリレーショナル・データベースに格納されます。項目が完了したかどうかを示すステータスが項目ごとにメンテナンスされます。複数のユーザーが「To Do List」アプリケーションを使用できますが、ユーザーが表示できるのはそのユーザー専用の項目のみです。

4.1.1 作業の準備

このチュートリアルでは、Mobile Development Kit (Web-to-Go 用) および Mobile サーバーが同一コンピュータ上にインストールされ、構成済であることが前提です。このチュートリアルを開始する前に、開発用コンピュータおよびクライアント・コンピュータが次に指定されている要件を満たしていることを確認する必要があります。

4.1.1.1 開発用コンピュータの要件

開発用コンピュータは、次の要件を満たしている必要があります。

表 4-2 開発用コンピュータの要件

要件	説明
Windows NT、2000 または XP でのユーザー・ログイン:	■ 開発用コンピュータ上の Windows NT、2000 または XP ログイン・ユーザーには、ADMINISTRATOR 権限が必要です。
インストール済の Java コンポーネント:	■ Java Development Kit 1.3.1 以上。
インストール済の Oracle コンポーネント:	■ Mobile サーバー (Oracle9i Lite CD-ROM)。 ■ Mobile Development Kit (Web-to-Go 用) (Oracle9i Lite CD-ROM)。

4.1.1.2 クライアント・コンピュータの要件

クライアント・コンピュータには、ネットワークを介して Mobile サーバーに接続するブラウザが必要です。このコンピュータを使用して、Web-to-Go アプリケーションをオンライン・モードおよびオフライン・モードでテストします。

4.2 アプリケーションの開発

この項では、Mobile Development Kit（Web-to-Go 用）を使用して「To Do List」アプリケーションを開発しテストする方法を説明します。「To Do List」アプリケーションには、次のコンポーネントが含まれています。

表 4-3 「To Do List」アプリケーションのコンポーネント

コンポーネント	機能
Java サーブレット	データベースにアクセスして「To Do」項目を挿入します。
JSP	「To Do List」アプリケーションのユーザー・インタフェースを HTML 形式で提供します。
JavaBean	JSP へのデータベース・アクセスを提供します。

この項では、次の操作を実行します。

- [ステップ 1: Oracle Lite へのデータベース・オブジェクトの作成](#)
- [ステップ 2: アプリケーション・コードの作成](#)
- [ステップ 3: アプリケーションのコンパイル](#)
- [ステップ 4: アプリケーションの定義およびサーブレットの登録](#)
- [ステップ 5: アプリケーションの実行](#)

Mobile Development Kit（Web-to-Go 用）は、常に Oracle Lite データベースを開発データベースとして使用します。このデータベースは、CREATEDB 文を使用して各自で作成できます。

Mobile Development Kit（Web-to-Go 用）は、Mobile クライアント Web サーバーと呼ばれる Web サーバーも使用します。

4.2.1 ステップ 1: Oracle Lite へのデータベース・オブジェクトの作成

このステップでは、「To Do List」アプリケーションのデータベース・オブジェクトを Oracle Lite データベースに作成します。

開発フェーズ中に、「To Do List」アプリケーションのサーブレットが「To Do」項目を Oracle Lite データベースに格納します。後の配置フェーズで、Oracle Lite から Oracle データベースにデータベース・オブジェクトをコピーします。

4.2.1.1 「To Do List」アプリケーションのデータベース・オブジェクト

「To Do List」アプリケーションは、次のデータベース・オブジェクトを使用します。

1. TODO_ITEMS 表。このアプリケーションは、TODO_ITEMS というデータベース表に「To Do」項目を格納します。この表には次の列が含まれています。

表 4-4 TODO_ITEMS 表

列	機能
ID	主キー。
TODO_ITEM	「To Do」項目を説明するテキスト。
USERNAME	「To Do」項目の所有者。
DONE	「To Do」項目が完了したかどうかを示します。

2. TODO_SEQ シーケンス。ユーザーが TODO_ITEMS 表に新規レコードを挿入するたびに、TODO_SEQ シーケンスが新規レコード用の主キー値を生成します。

4.2.1.2 必須アクション

Oracle Lite データベースにデータベース・オブジェクトを作成します。データベース・オブジェクトを作成するには、SQL スクリプト **tutorial.sql** を実行します。DOS プロンプトで次のとおり入力します。

1. createdb webtogo webtogo
2. cd <Oracle_Home>%mobile%sdk%wtgSDK%src\tutorial
3. msql system/x@jdbc:polite:webtogo @tutorial.sql

注意：「webtogo」と「@tutorial.sql」の間にはスペースが必要です。

msql は、Oracle Lite データベースに対して SQL 文を実行できるインタラクティブ・ツールです。これは、SQL*Plus に類似しています。

4.2.2 ステップ2: アプリケーション・コードの作成

「To Do List」アプリケーションの Java コードは、すでにこのチュートリアルで提供されています。次の Java コードの説明をよく読み、コードを再確認してください。

4.2.2.1 JSP

「To Do List」の JSP は、次のことを実行します。

- HTML ページを生成します。
- 未完了の「To Do」項目のリストを HTML ページに表示します。
- HTML ページの完了「To Do」項目にフラグを設定します。

「To Do List」の JSP には、次の場所からアクセスできます。

<Oracle_Home>%mobile%sdk%wtg%src%tutorial%ToDoList.jsp

4.2.2.2 JavaBean

「To Do List」の JSP は、JavaBean を使用して Oracle データベースに対する操作を実行します。「To Do List」の JavaBean には、次の場所からアクセスできます。

<Oracle_Home>%mobile%sdk%wtg%src%tutorial%ToDoBean.java

4.2.2.3 Java サーブレット

「To Do List」の Java サーブレットは、新規の「To Do」項目を Oracle データベースに挿入し、「To Do List」の JSP を使用して HTML ページを再生成します。「To Do List」の Java サーブレットには、次の場所からアクセスできます。

<Oracle_Home>%mobile%sdk%wtg%src%tutorial%InsertToDo.java

4.2.2.4 必須アクション

次の場所の Java アプリケーション・コードを表示します。

<Oracle_Home>%mobile%sdk%wtg%src%tutorial

4.2.3 ステップ3: アプリケーションのコンパイル

このステップでは、次の操作を実行してアプリケーションをコンパイルします。

1. 必要なライブラリを含めるための CLASSPATH の設定
2. Java サーブレットおよび JavaBean のコンパイル
3. JSP のインストール

4.2.3.1 必須アクション

1. 必要なライブラリ（Java Servlet Development Kit、Web-to-Go ライブラリなど）を含めるために CLASSPATH を設定します。DOS プロンプトから CLASSPATH を設定するために、**setenv.bat** という名前のスクリプトが提供されています。次のように入力します。

```
cd <Oracle_Home>%mobile%sdk%wtgSDK%bin
setenv.bat
```

2. アプリケーションをコンパイルします。アプリケーションは、手動でコンパイルするか、**compile.bat** スクリプトを実行してコンパイルできます。スクリプトを実行するには、DOS プロンプトで次のように入力します。

```
<Oracle_Home>%mobile%sdk%wtgSDK%src%tutorial
compile.bat
```

アプリケーションを手動でコンパイルするには、次の手順を実行します。

- a. DOS プロンプトで次のように入力して、Java サブレットをコンパイルします。

```
<Oracle_Home>%mobile%sdk%wtgSDK%src%tutorial
javac- d ../%root%tutorial InsertToDo.java
```

これで、次のサブレット・クラス・ファイルが作成されます。

```
<Oracle_Home>%mobile%sdk%wtgSDK%root%tutorial%InsertToDo.class
```

- b. DOS プロンプトで次のように入力して、JavaBean をコンパイルします。

```
javac -d ../%root%tutorial%WEB-INF%classes ToDoBean.java
```

- c. DOS プロンプトで次のように入力して、JSP をインストールします。

```
copy ToDoList.jsp <Oracle_Home>%mobile%sdk%wtgSDK%root%
%tutorial%ToDoList.jsp
```

4.2.4 ステップ 4: アプリケーションの定義およびサブレットの登録

このステップでは、パッケージ・ウィザードを使用して、「To Do List」アプリケーションの作成、アプリケーション・ファイルの追加、および Mobile クライアント Web サーバーへのアプリケーションのサブレットの登録を行います。開発環境では、アプリケーションとその関連サブレットをすべて Mobile クライアント Web サーバーに登録する必要があります。「To Do List」の JSP または JavaBean を登録する必要はありません。

4.2.4.1 パッケージ・ウィザード

開発者はパッケージ・ウィザードを使用して、Web-to-Go アプリケーションを作成または変更します。このチュートリアルでは、最初に開発モードでパッケージ・ウィザードを実行し、次に通常のモードでパッケージ・ウィザードを実行します。開発モードでは、パッケージ・ウィザードを使用して次の機能を実行します。

- Web-to-Go アプリケーションの定義
- サブレットの登録
- ファイルの追加
- JSP ファイルのコンパイル
- レジストリ・エントリの追加

パッケージ・ウィザードを開発モードで実行すると、配置時にのみ使用されるパネルは使用禁止になります。ここではアプリケーションをローカル・マシンにパブリッシュするため、パッケージ・ウィザードにアプリケーションの接続情報やデータベース情報を入力する必要があります。

詳細は、[第 5 章「Web-to-Go アプリケーションの定義」](#)を参照してください。

4.2.4.2 必須アクション

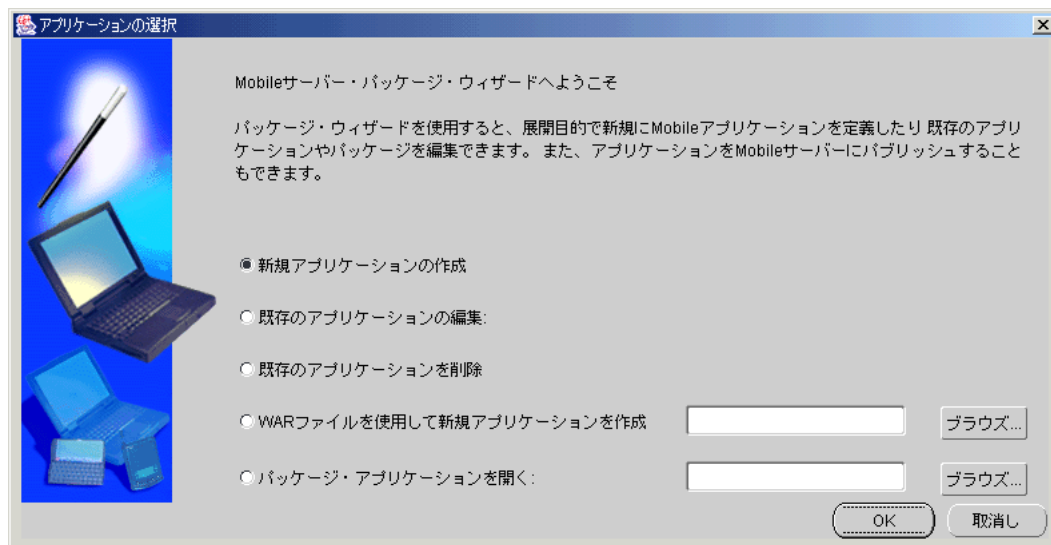
次の手順を実行して、「To Do List」アプリケーションを定義し、そのサブレットを登録します。

1. パッケージ・ウィザードをデバッグ・モードで起動します。DOS プロンプトで次のとおり入力します。

- a. `cd <Oracle_Home>%mobile%sdk%bin`
- b. `wtgpack -d`

[図 4-1](#) に示されているように、パッケージ・ウィザードが表示され、「新規アプリケーションの作成」、「既存のアプリケーションの編集」、「既存のアプリケーションを削除」、「パッケージ・アプリケーションを開く」の各オプションが提供されます。オプションの横のラジオ・ボタンを選択します。既存のアプリケーションを削除しても、XML ファイルからアプリケーションが削除されるのみで、ファイル・システムからは削除されないことに注意してください。

図 4-1 「アプリケーションの選択」ダイアログ・ボックス



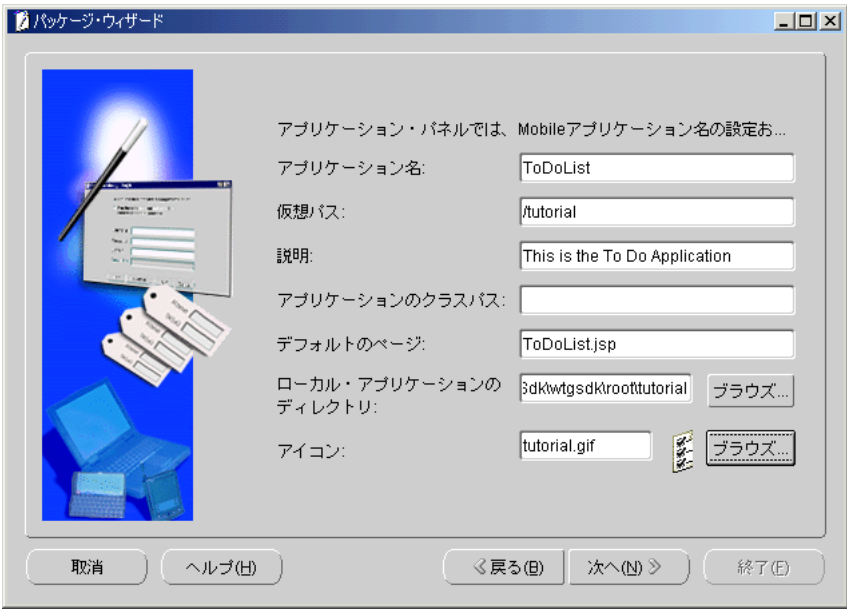
2. 「新規アプリケーションの作成」ラジオ・ボタンを選択し「OK」をクリックします。
3. プラットフォーム選択用のダイアログ・ボックスが表示されます。図 4-2 に示すように、このダイアログ・ボックスではアプリケーションのプラットフォームを指定できます。「使用可能プラットフォーム」リストから Web-to-Go を選択し、下矢印をクリックします。「選択済プラットフォーム」フィールドに Web-to-Go プラットフォームが表示されます。「次へ」をクリックします。「使用可能プラットフォーム」リスト（ダイアログ・ボックスの上半分）では、アプリケーションに対してサポートされる他のプラットフォーム、たとえば、Win32 ネイティブや Palm などを選択できます。このチュートリアルで選択するプラットフォームは、Web-to-Go です。上矢印と下矢印により、選択したプラットフォームをダイアログ・ボックスの下半分の「選択済プラットフォーム」フィールドに移動したり、プラットフォームの選択を変更する場合に「使用可能プラットフォーム」リストに戻すことができます。

図 4-2 プラットフォーム選択用のダイアログ・ボックス



4. 図 4-3 に示すような「アプリケーション」パネルが表示されます。

図 4-3 「アプリケーション」 パネル



5. 「アプリケーション」 パネルを使用して「To Do List」アプリケーション設定を変更します。次の各フィールドに指定された値を入力します。

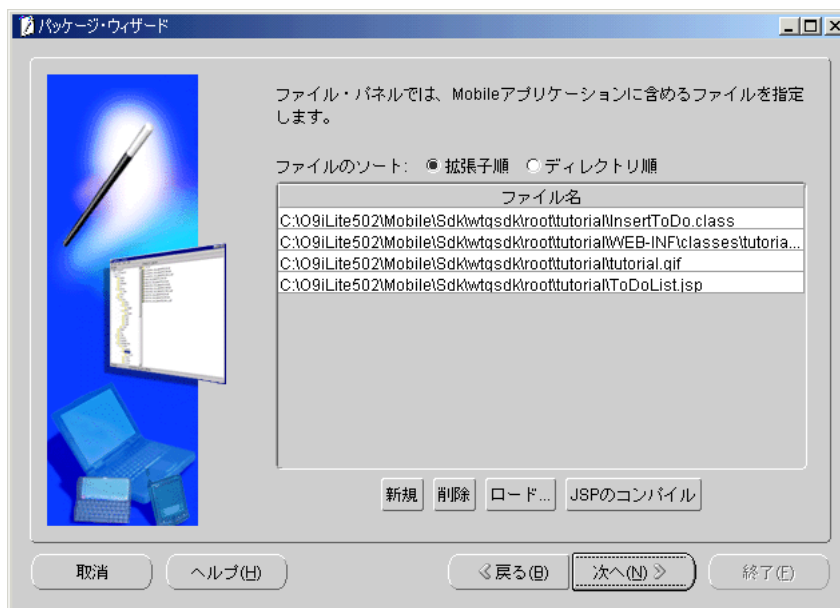
表 4-5 「To Do List」アプリケーションの設定

フィールド	値
アプリケーション名	ToDoList
仮想パス	/tutorial
説明	This is the To Do Application
デフォルトのページ	ToDoList.jsp (大 / 小文字を区別する)
ローカル・アプリケーションのディレクトリ	<Oracle_Home>%mobile%sdk%wtgSDKroot\tutorial
アイコン	tutorial.gif

6. 「次へ」をクリックします。「ファイル」パネルが表示されます。「ファイル」パネルを使用して、アプリケーションの一部となるファイルを選択します。アプリケーションのルート・ディレクトリの下のファイルは、自動的に含められます。

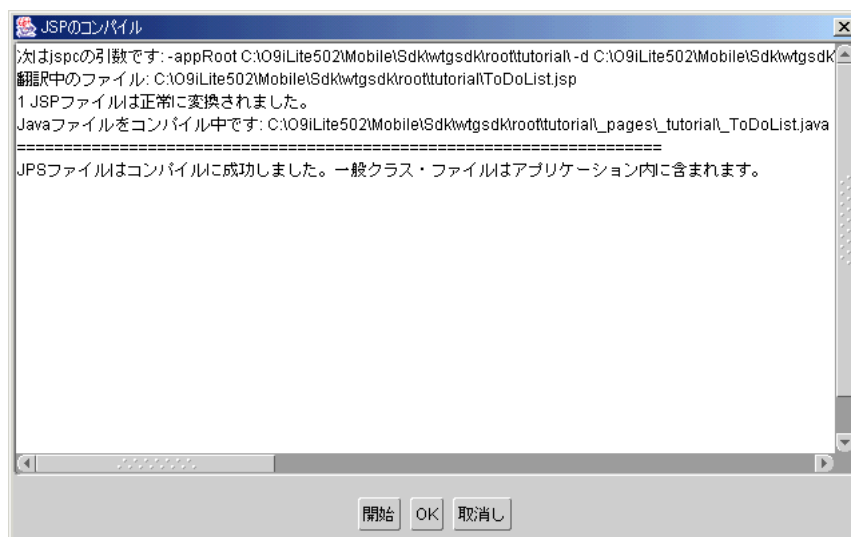
「ファイル」パネルは、パッケージ・ウィザードによってローカル・アプリケーションのディレクトリから Mobile サーバー上のアプリケーション・リポジトリにアップロードするファイルを識別します。

図 4-4 すべてのアプリケーション・ファイルのアップロード



7. JSP ファイルをコンパイルします。「JSP のコンパイル」ボタンをクリックします。
すべての JSP ファイルが、Java サーブレット・クラスにコンパイルされます。コンパイルが完了すると、次の成功メッセージが表示されます。

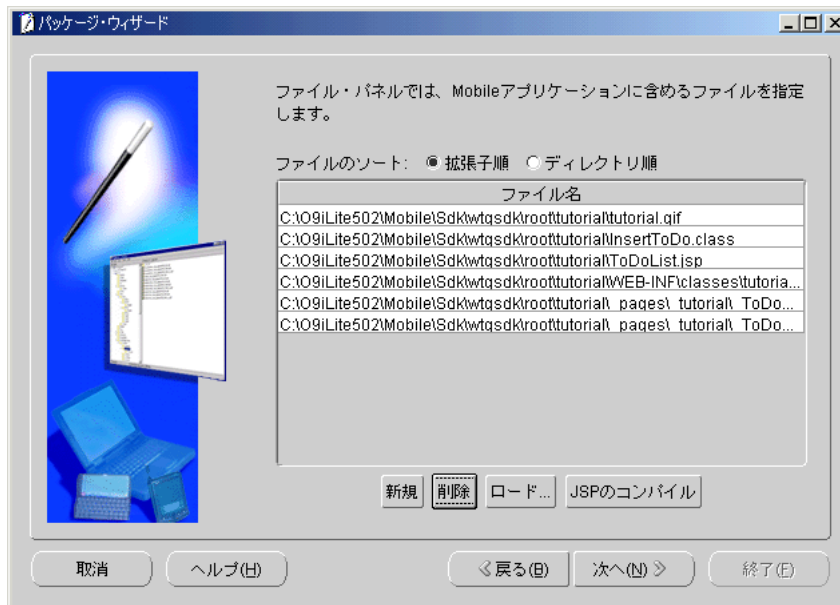
図 4-5 JSP のコンパイル完了の成功メッセージ



新規に生成されたファイルは、自動的にアプリケーション・ファイルのリストに追加されます。

8. 「OK」をクリックします。新しく生成されたファイルがアプリケーション・ファイルのリストに追加され、次のようなパネルが表示されます。

図 4-6 新しく生成されたファイルは自動的に追加される



9. 「To Do List」アプリケーション・サーブレットを表示するには、「次へ」をクリックします。パッケージ・ウィザードは、ローカル・アプリケーション・ディレクトリにあるサーブレットを自動的に検出して選択し、これらを Mobile クライアント Web サーバーに登録します。「To Do List」アプリケーションのサーブレットが「サーブレット」パネルに表示されます。「To Do List」アプリケーションにはサーブレットが1つのみ含まれるため、「サーブレット」パネルに表示される行は1行のみです。

図 4-7 「サーブレット」 パネル



「サーブレット」パネルを使用すると、仮想パス（サーブレット名）を Java クラス（サーブレットのクラス）にマップできます。

パッケージ・ウィザードの使用の詳細は、[第 5 章「Web-to-Go アプリケーションの定義」](#)を参照してください。

- フィールドを選択して、サーブレットの名前を「insert」に変更します。選択すると表示が白に変わります。サーブレット名は大 / 小文字が区別され、すべて小文字にする必要があります。サーブレット名は、必ず「次へ」をクリックする前に変更してください。
- 「次へ」をクリックします。「レジストリ」パネルが表示されます。入力の必要なレジストリ設定はありません。「終了」をクリックします。これでアプリケーション定義が保存され、アプリケーションを実行する準備ができました。

4.2.5 ステップ 5: アプリケーションの実行

このステップでは、Mobile クライアント Web サーバーを開発用コンピュータ上で起動して、「To Do List」アプリケーションを実行します。次に、Web ブラウザを起動し、ブラウザを「To Do List」アプリケーションの URL に接続して、このアプリケーションにアクセスします。

4.2.5.1 Mobile Development Kit (Web-to-Go 用) Web サーバー

Mobile クライアント Web サーバーは、パッケージ・ウィザードで指定された「To Do List」アプリケーション情報と Java サブレットをロードします。Mobile クライアント Web サーバーを起動すると、このサーバーが常駐するコンピュータの URL を指定して、任意の Web ブラウザからアクセスできます。Mobile クライアント Web サーバー用のデフォルトのポートは 7070 です。Mobile クライアント Web サーバーで使用するポートは、**webtogo.ora** ファイルのポート・エントリを変更して構成できます。フルパスは、次のとおりです。

```
<Oracle_Home>%mobile%sdk%bin%webtogo.ora
```

webtogo.ora ファイルのパラメータ構成に関する追加情報は、『Oracle9i Lite 管理者およびデプロイ・ガイド』、および第 3 章「Web-to-Go アプリケーションの開発」の「[webtogo.ora ファイル](#)」を参照してください。

4.2.5.2 必須アクション

次の手順を実行して、「To Do List」アプリケーションを実行します。

1. Mobile クライアント Web サーバーを起動します。DOS プロンプトで次のように入力します。

- a. `cd <Oracle_Home>%mobile%sdk%bin`
- b. `wtgdebug.exe`

Mobile クライアント Web サーバーが起動され、どのサブレットがロードされたかが示されます。サブレットに `System.out.println()` 文が含まれている場合は、このウィンドウにメッセージが表示されます。

2. Web ブラウザを起動し、次の URL に接続します。

`http://your_machine:7070/`

Web-to-Go システムが現在認識しているアプリケーションのリストが次のように表示されます。

図 4-8 アプリケーションのリスト

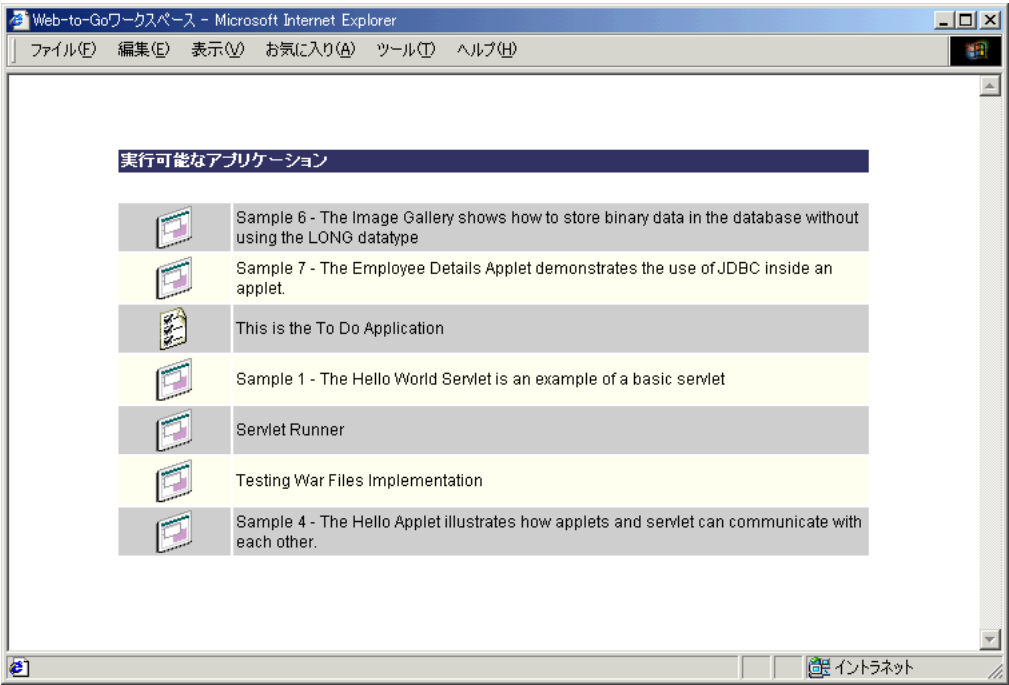


表 4-6 システムが現在認識しているアプリケーションのリスト

アプリケーション	説明
「To Do List」アプリケーション	ステップ 4 で Mobile クライアント Web サーバーに追加したアプリケーションです。
ServletRunner	アプリケーションに割り当てられていないパブリッシュ済の全サーブレットを含むデフォルト・アプリケーションです。
Sample 1	Hello World サブレットは、基本的なサブレットの例です。このアプリケーションは次の場所にあります。 <Oracle_Home>%mobile%sdk\wtg\root\sample1
Sample 3	Recording Tracker は、サブレットを使用してレコード情報でデータベースをメンテナンスする方法を示します。このアプリケーションは次の場所にあります。 <Oracle_Home>%mobile%sdk\wtg\root\sample3

表 4-6 システムが現在認識しているアプリケーションのリスト（続き）

アプリケーション	説明
Sample 4	Hello Applet は、アプレットとサーブレットが相互に通信する方法を示します。このアプリケーションは次の場所にあります。 <Oracle_Home>%mobile%sdk%wtgSDK%root%sample4
Sample 7	Employee Data Applet は、アプレット内部での JDBC の使用方法を示します。このアプリケーションは次の場所にあります。 <Oracle_Home>%mobile%sdk%wtgSDK%root%sample7
Sample 6	Image Gallery は、LONG データ型を使用せずにバイナリ・データをデータベースに格納する方法を示します。このアプリケーションは次の場所にあります。 <Oracle_Home>%mobile%sdk%wtgSDK%root%sample6

3. 「To Do List」アプリケーションをクリックします。次の情報を含むブラウザの新しいウィンドウが表示されます。
- 未完了の「To Do」項目のリスト

■ 新規「To Do」項目の作成に使用する簡単な HTML フォーム
- 未完了の「To Do」項目には、項目の前に文字「X」が付きます。「X」をクリックすると、「To Do List」アプリケーションはその項目が完了したものとしてフラグを設定し、その項目をリストから削除します。

4.3 アプリケーションのパッケージ化

この項目では、アプリケーションをパッケージ化して、Mobile サーバーにパブリッシュする準備を行う方法を説明します。ここでは次の作業を実行します。

- ステップ 1: パッケージ・ウィザードを使用したアプリケーションの定義
- ステップ 2: Oracle データベースへのアプリケーション接続の定義
- ステップ 3: スナップショットの定義
- ステップ 4: シーケンスの定義
- ステップ 5: アプリケーションの SQL ファイルの作成

4.3.1 ステップ 1: パッケージ・ウィザードを使用したアプリケーションの定義

このステップでは、パッケージ・ウィザードを使用して「To Do List」アプリケーションを選択して説明文を追加します。

4.3.1.1 パッケージ・ウィザード

パッケージ・ウィザードを使用すると、Web-to-Go アプリケーションを作成または変更して、Mobile サーバーにパブリッシュできます。このチュートリアルでは、開発フェーズのステップ 4～8 でパッケージ・ウィザードを使用します。

4.3.1.2 必須アクション

「To Do List」アプリケーションを選択し記述するには、パッケージ・ウィザードを通常の方法で起動します。

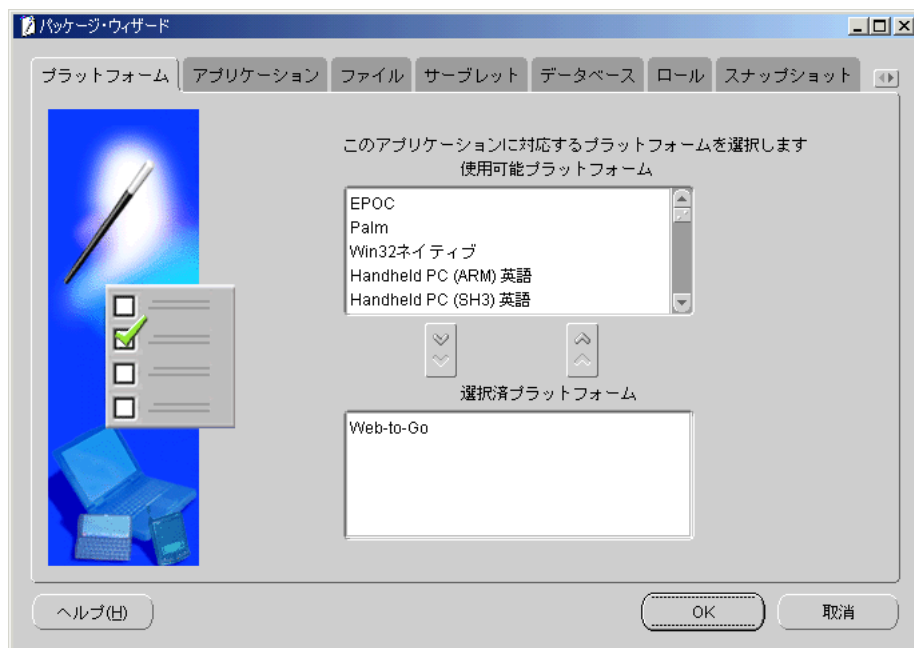
1. DOS プロンプトで次のとおり入力します。

- a. `cd <Oracle_Home>%mobile%sdk%bin`
- b. `wtgpack`

パッケージ・ウィザードが表示され、すべてのパネルが使用可能になります。パッケージ・ウィザードの起動時、デフォルトでは「ようこそ」パネルが表示されます。

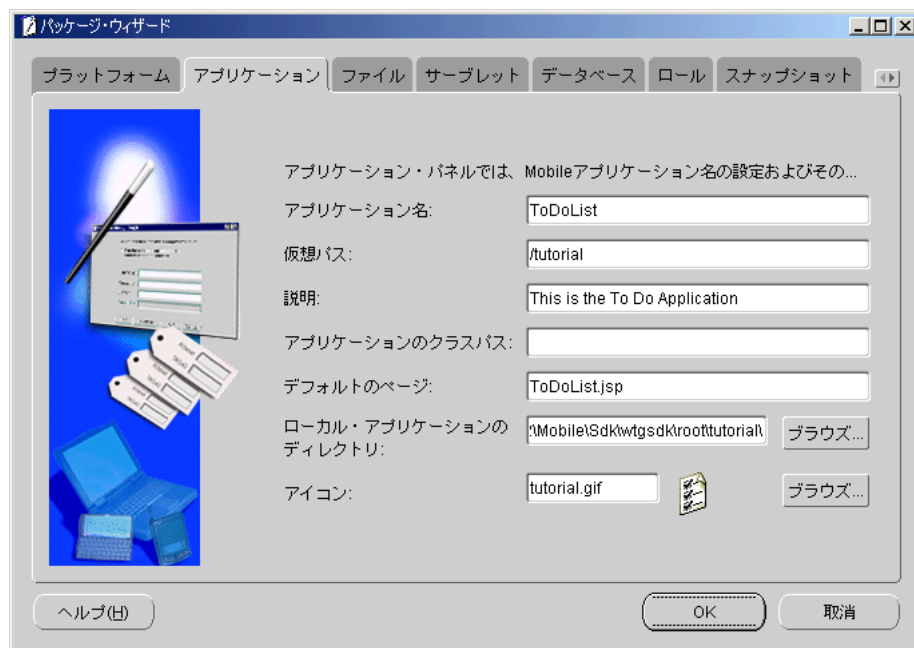
2. 「既存のアプリケーションの編集」を選択し、ドロップダウン・リストから「To Do List」アプリケーションを選択します。
3. 「OK」をクリックします。「プラットフォーム」パネルが表示されます。[図 4-9](#) に示すように、「プラットフォーム」パネルには、[4.2.4 項「ステップ 4: アプリケーションの定義およびサブレットの登録」](#) で入力した情報と同じ情報が含まれています。

図 4-9 「プラットフォーム」パネル



4. 「アプリケーション」タブをクリックします。「アプリケーション」パネルが表示されます。「アプリケーション」パネルには、[4.2.4 項「ステップ 4: アプリケーションの定義およびサブレットの登録」](#)で入力した情報と同じ情報が含まれています。

図 4-10 「アプリケーション」パネル



5. 次の手順を実行して、「To Do List」アプリケーションに説明文を追加します。

a. 次の各フィールドに指定されている値が正しいことを確認します。

表 4-7 指定されている値

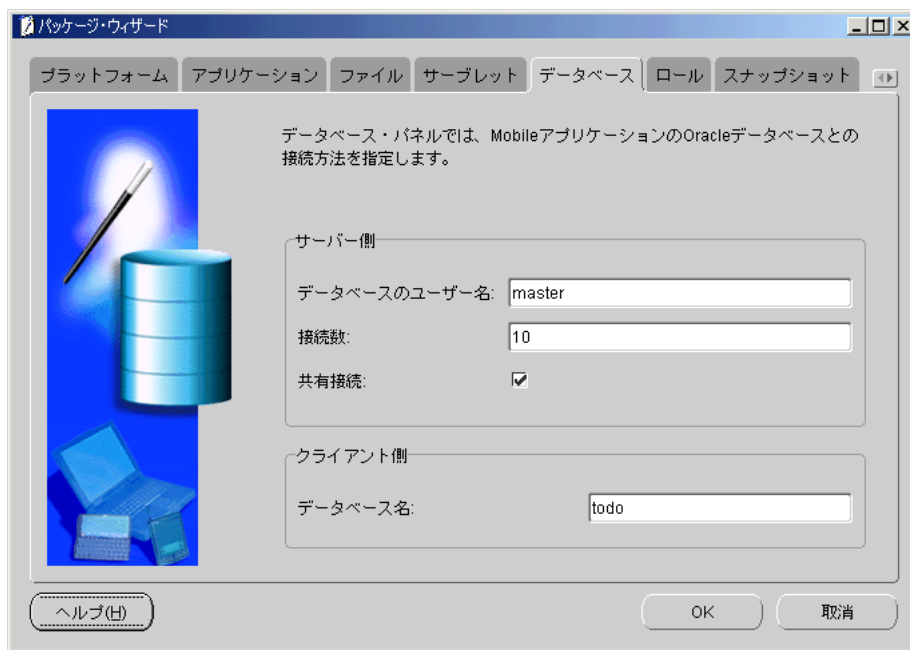
フィールド	値
アプリケーション名	ToDoList
仮想パス	/tutorial
説明	This is the To Do Application
アプリケーションのクラスパス	
デフォルトのページ	ToDoList.jsp (大 / 小文字を区別する)
ローカル・アプリケーションのディレクトリ	<Oracle_Home>%mobile%sdk\wtg\root\tutorial
アイコン	tutorial.gif

- b. 「ファイル」タブをクリックします。「ファイル」パネルが表示されます。ファイルはすべて、すでにアプリケーションに追加されています。
- c. 「サーブレット」タブをクリックします。「サーブレット」パネルが表示されます。
- d. 「データベース」タブをクリックします。「データベース」パネルが表示されます。

4.3.2 ステップ 2: Oracle データベースへのアプリケーション接続の定義

このステップでは、パッケージ・ウィザードの「データベース」パネルを使用して、「To Do List」アプリケーションの Oracle データベースへの接続を定義します。「データベース」パネルで、Oracle データベース・サーバー上のレプリケーション・マスター・グループに対する Web-to-Go アプリケーション・ユーザーの接続を定義します。この場合、「To Do List」アプリケーションのデータベース・オブジェクトを含む Oracle データベース・サーバーの接続情報を定義します。

図 4-11 「データベース」パネル



接続設定は、「To Do List」アプリケーション用に最大 10 個の接続を作成するように Web-to-Go に指示します。接続は、接続プールを使用してユーザー間で共有されます。ユーザーに接続が不要になるとその接続はプールに戻され、別のユーザーが使用できます。詳細は、[第 3 章「Web-to-Go アプリケーションの開発」](#)の「データベース・コンポーネント」を参照してください。

データベース名は、このアプリケーション用に Web-to-Go 用 Mobile クライアント上に作成されるデータベース・ファイルおよび対応する DSN を参照します。

4.3.2.1 必須アクション

各フィールドに次の値を入力します。

表 4-8 必須アクションの値

フィールド	値
データベースのユーザー名	master
接続数	10
共有接続	チェック
データベース名	todo

「スナップショット」タブをクリックします。「スナップショット」パネルが表示されます。

注意： このチュートリアル・アプリケーションでは特別なロールを使用しないため、「ロール」はスキップします。

「ロール」パネルを使用して、開発者は Web-to-Go アプリケーション用のロールを定義できます。ただし通常は、アプリケーション・ロールは開発者が Web-to-Go アプリケーションに組み込む必要があります。自動的に組み込まれません。アプリケーション・ロールの作成方法の詳細は、[第 3 章「Web-to-Go アプリケーションの開発」](#)の 3.3 項「アプリケーション・ロール」を参照してください。

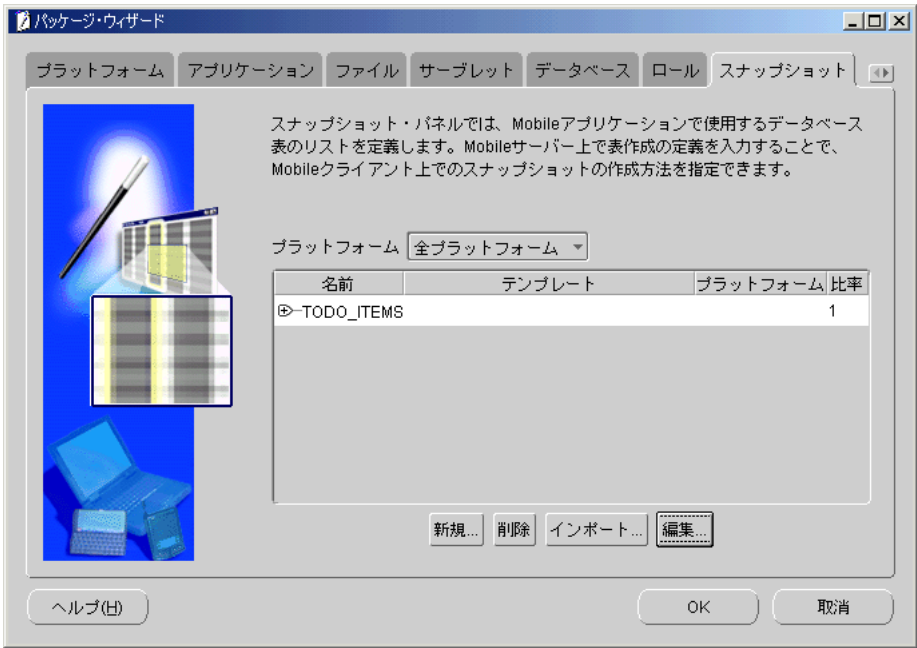
4.3.3 ステップ 3: スナップショットの定義

このステップでは、パッケージ・ウィザードを使用して「To Do List」アプリケーションのデータベース・スキーマ・オブジェクトを配置します。

4.3.3.1 「スナップショット」パネル

「スナップショット」パネルには、スナップショットを作成する対象のデータベース表を定義します。表定義は、パッケージ・ウィザードを使用して開発データベースからインポートできます。これらの定義は、Mobile アプリケーション用のスナップショットを定義するために使用できます。

図 4-12 「スナップショット」パネル



4.3.3.2 必須アクション

「スナップショット」パネルで次のステップを実行して、開発データベースから表定義をインポートします。

1. 「インポート」をクリックします。「接続」ウィンドウが表示されます。指定されたフィールドに次の情報を入力します。

フィールド	値
ユーザー名	system
パスワード	manager
URL	jdbc:polite:webtogo

2. 「OK」をクリックします。「表」ウィンドウが表示され、使用可能な表のリストが表示されます。

図 4-13 「表」ダイアログ・ボックス



3. TODO_ITEMS 表を選択し、「追加」をクリックしてから「閉じる」をクリックします。TODO_ITEMS スナップショットとスナップショット・テンプレート用の SQL 文が、「表」ウィンドウの右のフレームに表示されます。
4. TODO_ITEMS 表を選択し、「編集」をクリックします。「スナップショットの編集」ダイアログ・ボックスが表示されます。

図 4-14 「スナップショットの編集」ダイアログ・ボックス

スナップショットの編集

サーバー Win32

スナップショット名 TODO_ITEMS

比率 1

所有者 MASTER

☒ SQLの生成

SQL:

```
CREATE TABLE &WTG_SCHEMA.TODO_ITEMS
(ID NUMBER(28,0) NOT NULL
, TO_DO VARCHAR2(100) NOT NULL
, USERNAME VARCHAR2(100) NOT NULL
, DONE NUMBER(1,0)
, PRIMARY KEY (ID))
```

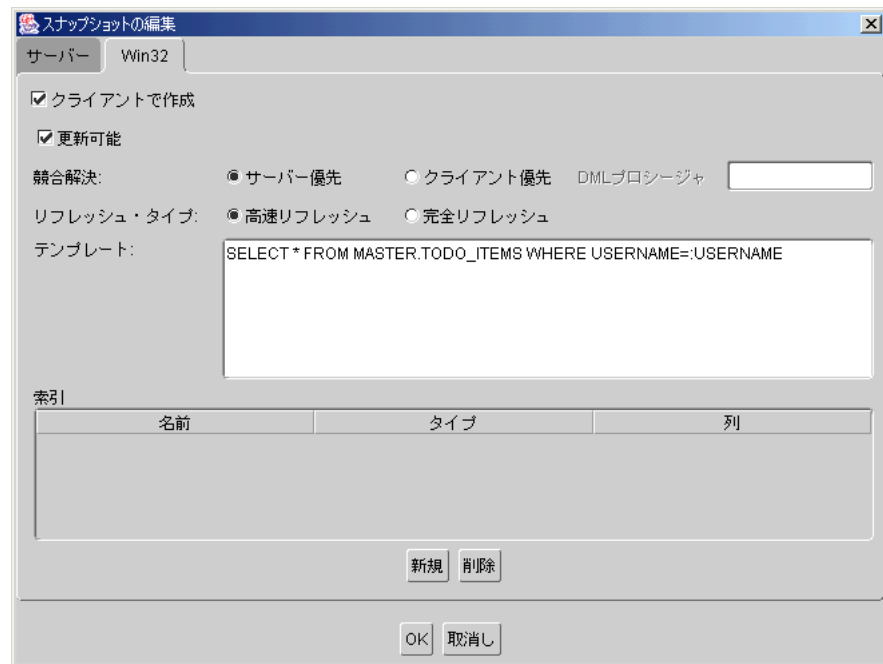
OK 取消し

5. 「比率」を1に変更します。これにより、クライアント上でスナップショットが参照される順序が制御されます。
6. 「Win32」タブをクリックします。図 4-15 に示すように、Win32 クライアント用のパネルが表示されます。「クライアントで作成」チェックボックスを選択します。「テンプレート」フィールドの SQL 文を次のように変更して「OK」をクリックします。

```
SELECT * FROM MASTER.TODO_ITEMS WHERE USERNAME = :USERNAME
```

次の図には、ここで説明した「テンプレート」フィールドの SQL 文が示されています。

図 4-15 「スナップショットの編集」ダイアログ・ボックスの「Win32」パネル



7. 「OK」をクリックします。

表のスナップショット定義の SQL 文では、CHAR 列を述語としては使用しないことをお勧めします。CHAR 列ではなく、VARCHAR2 を使用してください。VARCHAR2 を使用しないと、データを正常に移行させたり、またはデータを正常に同期させることができません。

4.3.4 ステップ 4: シーケンスの定義

「シーケンス」パネルには、オフライン・モードにあるクライアント・アプリケーション用に Web-to-Go が作成するシーケンスを定義します。このステップでは、「To Do List」アプリケーションがオフライン・モードで使用する TODO_SEQ シーケンスの新しい定義を作成します。後で、Oracle データベースに実際のシーケンスを作成します。同期中に、Web-to-Go は TODO_SEQ シーケンスのローカル・コピーをクライアント上に自動的に作成します。

1. 「シーケンス」タブをクリックします。図 4-16 に示すような「シーケンス」パネルが表示されます。パネルには次のような説明が表示されます。「シーケンス・パネルでは、Web-to-Go アプリケーションで使用するデータベース・シーケンスのリストを定義します。Web-to-Go サーバー上でシーケンス作成の定義を入力することで、Web-to-Go 用 Mobile クライアント上でのシーケンスの作成方法を指定できます。」シーケンスは、このステップを実行する前にデータベースに存在している必要があります。

図 4-16 「シーケンス」 パネル



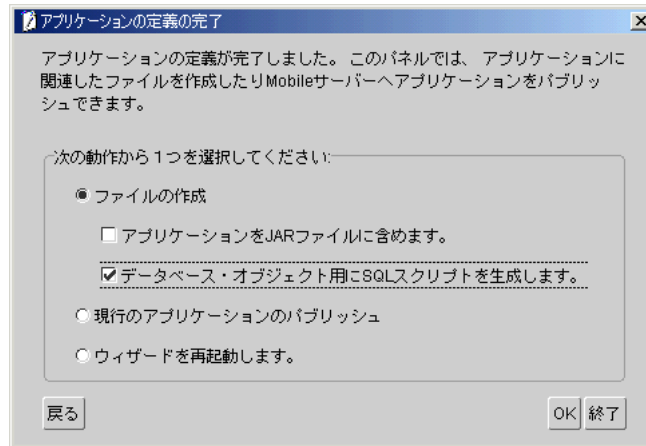
2. 「インポート」をクリックします。図 4-17 に示すような、使用可能なシーケンスのリストを示した「シーケンス」ウィンドウが表示されます。

図 4-17 「シーケンス」 ウィンドウ



3. TODO_SEQ シーケンスを選択し「追加」をクリックします。
4. 「閉じる」をクリックします。
5. 「OK」をクリックします。図 4-18 に示すように、「アプリケーションの定義の完了」ダイアログ・ボックスが表示されます。

図 4-18 「アプリケーションの定義の完了」ダイアログ・ボックス



注意： このチュートリアル・アプリケーションでは、DDL やレジストリは使用しません。したがって、これらはスキップします。

4.3.5 ステップ 5: アプリケーションの SQL ファイルの作成

「To Do List」アプリケーションに SQL ファイルを作成するには、「アプリケーションの定義の完了」ダイアログ・ボックスを使用します。

4.3.5.1 必須アクション

「ファイルの作成」ラジオ・ボタンを選択してから「データベース・オブジェクト用に SQL スクリプトを生成します。」チェックボックスをクリックします。「OK」をクリックします。

これで、データベース・オブジェクト用の SQL スクリプトが生成されます。

パッケージ・ウィザードが、指定されたファイルを次のディレクトリに挿入します。

```
<Oracle_Home>%mobile%sdk%wtgSDK%root%\tutorial\sql
```

ファイル	説明
ToDoList.sql	他の SQL ファイルをコールするマスター・スクリプトです。
tables.sql	すべての SQL 表を作成するスクリプトです。
Sequences.sql	シーケンスを作成する SQL スクリプトです。
DDLs.sql	DDL が定義されていないため、このファイルは空です。

4.3.6 ステップ 6: アプリケーションのパッケージ化

「アプリケーションの定義の完了」ダイアログ・ボックスを使用して「To Do List」アプリケーションを jar ファイルにパッケージ化します。

4.3.6.1 必須アクション

「ファイルの作成」ラジオ・ボタンを選択してから「アプリケーションを JAR ファイルに含めます。」チェックボックスをクリックします。「データベース・オブジェクト用に SQL スクリプトを生成します。」チェックボックスはチェックしないようにしてください。

jar ファイルの名前を入力するよう要求されます。デフォルトの場所は、次のとおりです。

```
<Oracle_Home>\%Mobile\Sdk\wtg-sdk\root\ToDoList.jar
```

図 4-19 に示すように、「アプリケーションの保存」ダイアログ・ボックスが表示され、「JAR ファイルを保存する場所を指定してください。」という要求と「ブラウズ」ボタンが表示されます。

図 4-19 「アプリケーションの保存」ダイアログ・ボックス



「OK」をクリックします。

アプリケーション・ファイルと定義の含まれた jar ファイルが作成されます。

これで、アプリケーションをパッケージ化するために必要な開発作業をすべて完了しました。アプリケーションは、パッケージ化されています。

4.4 アプリケーションのパブリッシュ

アプリケーションをパッケージ化すると、いつでもこれをパブリッシュできます。次の項では、アプリケーションをパブリッシュする手順を説明します。

4.4.1 ステップ 1: 表の所有者アカウントの作成

このステップでは、Oracle データベースで「To Do List」アプリケーションのオブジェクトを所有するデータベース・ユーザーを作成します。DOS プロンプトで次のとおり入力します。

1. `sqlplus system/manager@webtogo.world`
2. `create user master identified by master;`
3. `grant connect, resource to master;`

4.4.2 ステップ 2: Oracle データベースへのデータベース・オブジェクトの作成

このステップでは、「To Do List」アプリケーションのデータベース・オブジェクトを Oracle データベースに作成します。

4.4.2.1 必須アクション

DOS プロンプトで次のように入力して、SQL マスター・スクリプトを実行します。

```
cd <Oracle_Home>%mobile%sdk%wtg%root%tutorial%sql  
sqlplus master/master@webtogo.world @ToDoList.sql
```

このスクリプトは、Oracle データベースに対して次の操作を実行します。

- `TODO_ITEMS` 表を作成します。
- シーケンスを作成します。

4.4.3 ステップ 3: Mobile サーバーの起動

このステップでは、Mobile サーバーを起動します。

4.4.3.1 必須アクション

Mobile サーバーを起動するには、DOS のコマンドラインで次のように入力します。

```
webtogo -d0 mobileadmin/manager@webtogo.world
```

この例のパスワードは、`manager` です。独自のパスワードを使用してください。

4.4.4 ステップ 4: Mobile サーバーへのログオンと Mobile サーバー・コントロール・センターの起動

このステップでは、Mobile サーバーにログオンして、Mobile サーバー・コントロール・センターを起動します。

4.4.4.1 必須アクション

Mobile サーバー・コントロール・センターを起動するには、次の手順を実行します。

1. Web ブラウザを起動し、次の URL を入力して Mobile サーバーに接続します。

`http://server/webtogo`

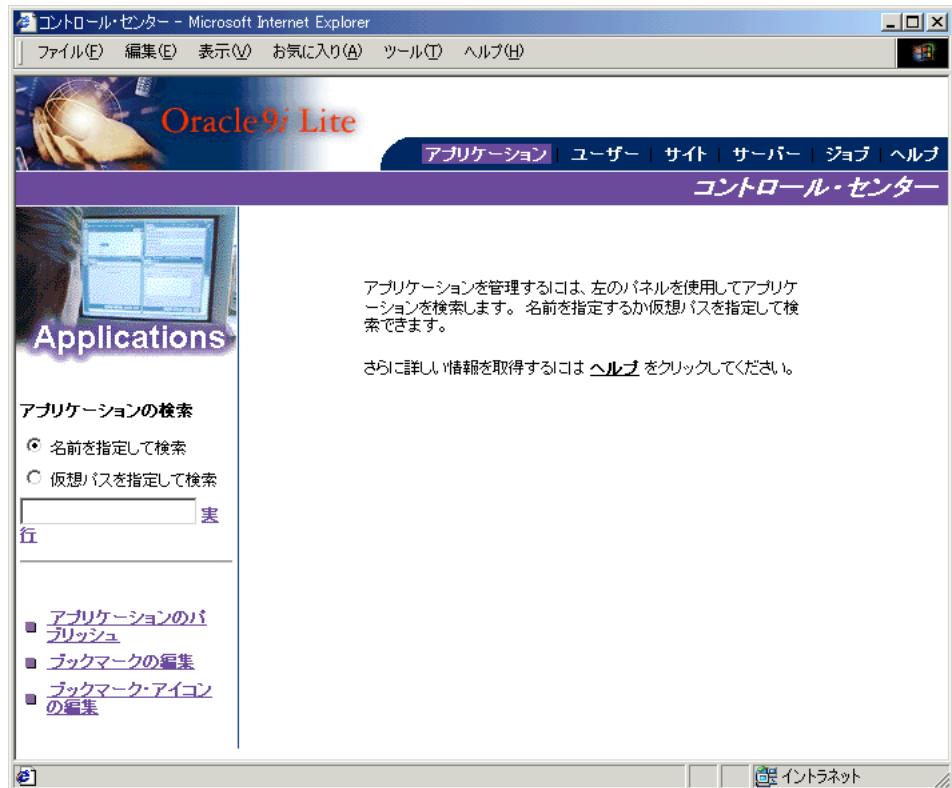
注意： `server` 変数は、使用する Mobile サーバーのホスト名に置き換えてください。

2. 次のアカウント情報を入力し、Mobile サーバー管理者としてログオンします。

フィールド	値
ユーザー名	administrator
パスワード	admin

3. ワークスペース内の「コントロール・センター」リンクをクリックしてコントロール・センターを起動します。図 4-20 に示すように、新しいブラウザ・ウィンドウに Mobile サーバー・コントロール・センター・アプリケーションが表示されます。図は、「アプリケーション」ページの表示されたブラウザ・ウィンドウを示しています。右側のパネルでは、左側のパネルを使用してパブリッシュするアプリケーションを検索するよう示しています。アプリケーションは、ラジオ・ボタンを使用して、名前または仮想パスで検索できます。

図 4-20 コントロール・センターの起動



4.4.5 ステップ 5: アプリケーションのアップロード

このステップでは、「To Do List」アプリケーションの含まれている jar ファイルをアップロードします。

4.4.5.1 必須アクション

アプリケーションを Mobile サーバーにアップロードするには、次の手順を実行します。

1. 右上にある「アプリケーション」タブをクリックします。「アプリケーション」メニューが左側のフレームに表示されます。
2. 左側のフレームの「アプリケーションのパブリッシュ」をクリックします。図 4-21 に示すように、ワークスペースに「新規アプリケーション」画面が表示されます。

- リポジトリのディレクトリに次の値を入力します。

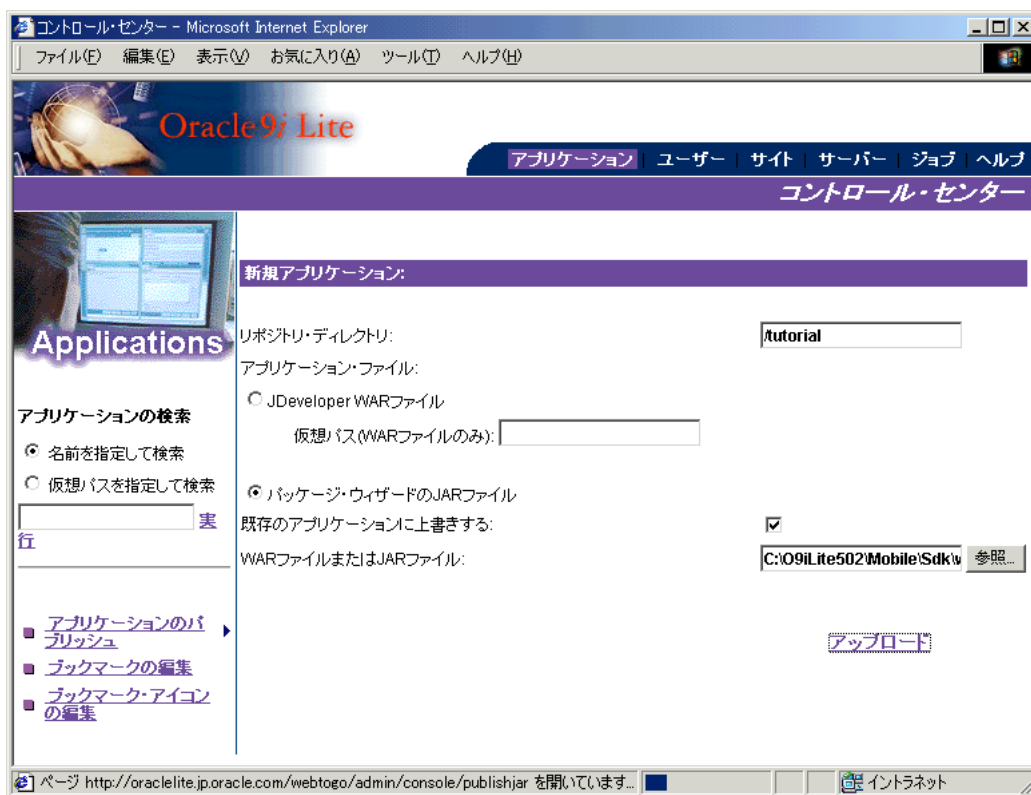
/tutorial

- 「参照 ...」 ボタンを使用して、4.3.6 項「ステップ 6: アプリケーションのパッケージ化」で作成した jar ファイルを選択します。jar ファイルのデフォルトの場所は、次のとおりです。

OracleHome¥Mobile¥sdk¥wtg¥root¥ToDoList.jar

- 「アップロード」をクリックして、アプリケーションをパブリッシュします。

図 4-21 「新規アプリケーション」 ページ



アプリケーションがパブリッシュされます。

注意： アプリケーションのプロパティは、この後の **4.5.3 項「ステップ 3: アプリケーション・プロパティの設定」** のステップで設定します。

4.5 アプリケーションの管理

ここでは次の作業を実行します。

- コントロール・センターを起動します。
- コントロール・センターを使用して新規ユーザーを作成します。
- アプリケーション・プロパティを設定します。
- アプリケーションにユーザー・アクセス権を付与します。
- ユーザーのスナップショット・テンプレート変数にスナップショット・テンプレート値を定義します。

このチュートリアルで説明されているコントロール・センターでの作業の詳細は、『Oracle9i Lite 管理者およびデプлой・ガイド』を参照してください。

4.5.1 ステップ 1: Mobile サーバー・コントロール・センターの起動

このステップでは、Mobile サーバー・コントロール・センターを起動します。この Web ベースのアプリケーションはブラウザで実行され、Mobile サーバー・アプリケーションを簡単に管理できます。

4.5.1.1 必須アクション

Mobile サーバー・コントロール・センターを起動するには、次の手順を実行します。

1. Web ブラウザを起動し、次の URL を入力して Mobile サーバーに接続します。

`http://server/webtogo`

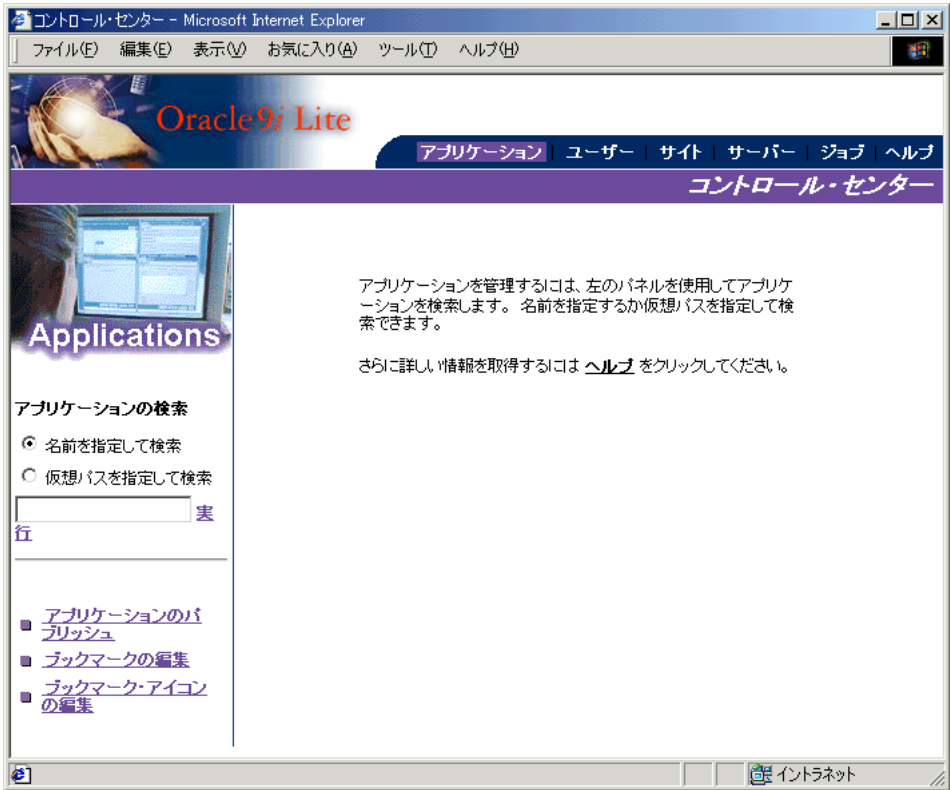
注意： `server` 変数は、使用する Mobile サーバーのホスト名に置き換えてください。

2. 次のアカウント情報を入力し、Mobile サーバー管理者としてログインします。

フィールド	値
ユーザー名	administrator
パスワード	admin

3. ワークスペース内の「コントロール・センター」アイコンをクリックしてコントロール・センターを起動します。コントロール・センター・アプリケーションがブラウザの新しいウィンドウに表示されます。

図 4-22 コントロール・センターの起動



4.5.2 ステップ 2: コントロール・センターを使用した新規ユーザーの作成

このステップでは、新規ユーザーを作成します。

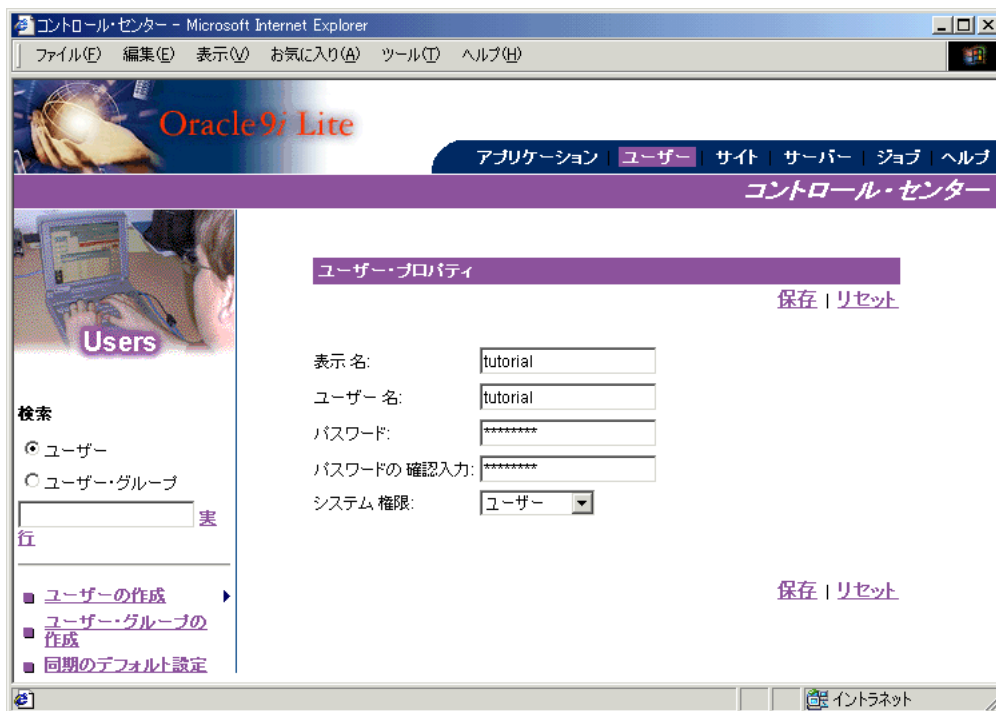
4.5.2.1 必須アクション

Mobile サーバーの新規ユーザーを作成するには、次の手順を実行します。

1. 右上にある「ユーザー」タブをクリックします。ユーザー・メニューが左側のフレームに表示されます。
2. 「ユーザーの作成」をクリックします。新規ユーザー情報がワークスペースに表示されます。
3. 次のユーザー情報を入力してから「保存」をクリックします。

フィールド	値
表示名 :	tutorial
ユーザー名 :	tutorial
パスワード :	tutorial
パスワードの確認入力 :	tutorial
システム権限 :	ユーザー

図 4-23 「ユーザーのプロパティ」パネル



4.5.3 ステップ 3: アプリケーション・プロパティの設定

このステップでは、「To Do List」アプリケーションのプロパティを設定します。

4.5.3.1 必須アクション

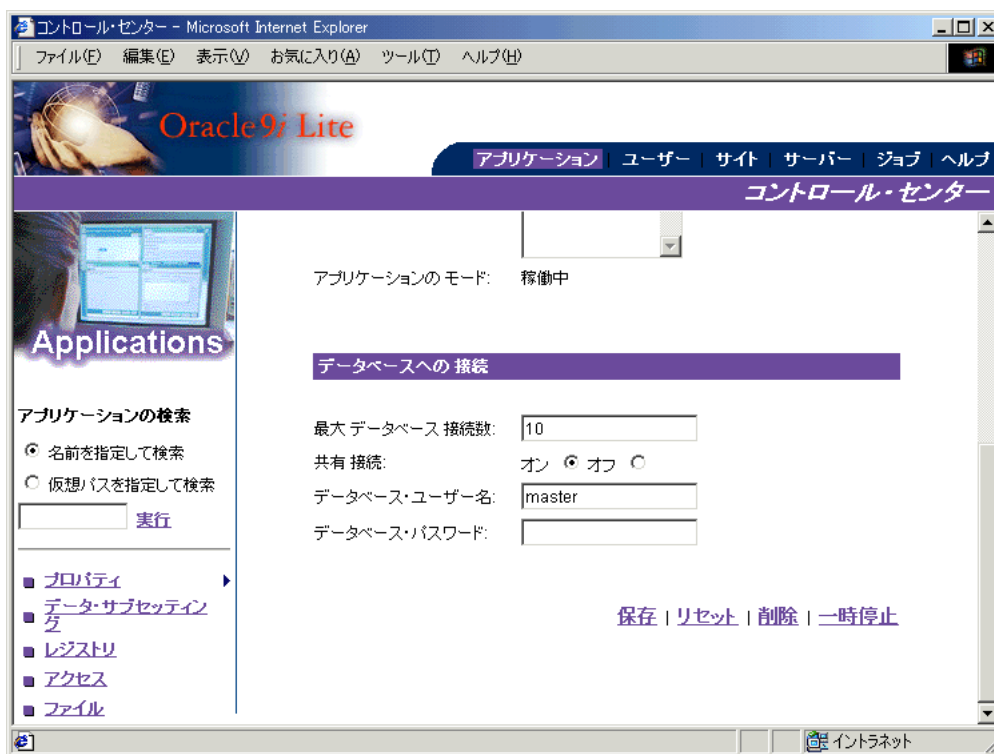
「To Do List」アプリケーションのプロパティを設定するには、次の手順を実行します。

1. 右上にある「アプリケーション」タブをクリックします。「アプリケーション」メニューが左側のフレームに表示されます。
2. 「To Do List」アプリケーションを検索する条件を入力します。「名前を指定して検索」オプションを選択し、検索フィールドに「ToDo」と入力して「実行」をクリックします。ワークスペースに「To Do List」アプリケーションが表示されます。

注意：「検索」フィールドを空白のままにして「実行」をクリックすることもできます。こうすると、使用可能な Mobile サーバー・アプリケーションがすべてワークスペースにリストされます。

3. 「To Do List」アプリケーションをクリックし、左側のフレームにある「プロパティ」をクリックします。
4. 「データベース・パスワード」フィールドに「master」と入力します。これは、Web-to-Go デモ・スキーマのデフォルトのパスワードです。「保存」をクリックします。

図 4-24 アプリケーション・プロパティの設定



4.5.4 ステップ 4: アプリケーションへのアクセス権の付与

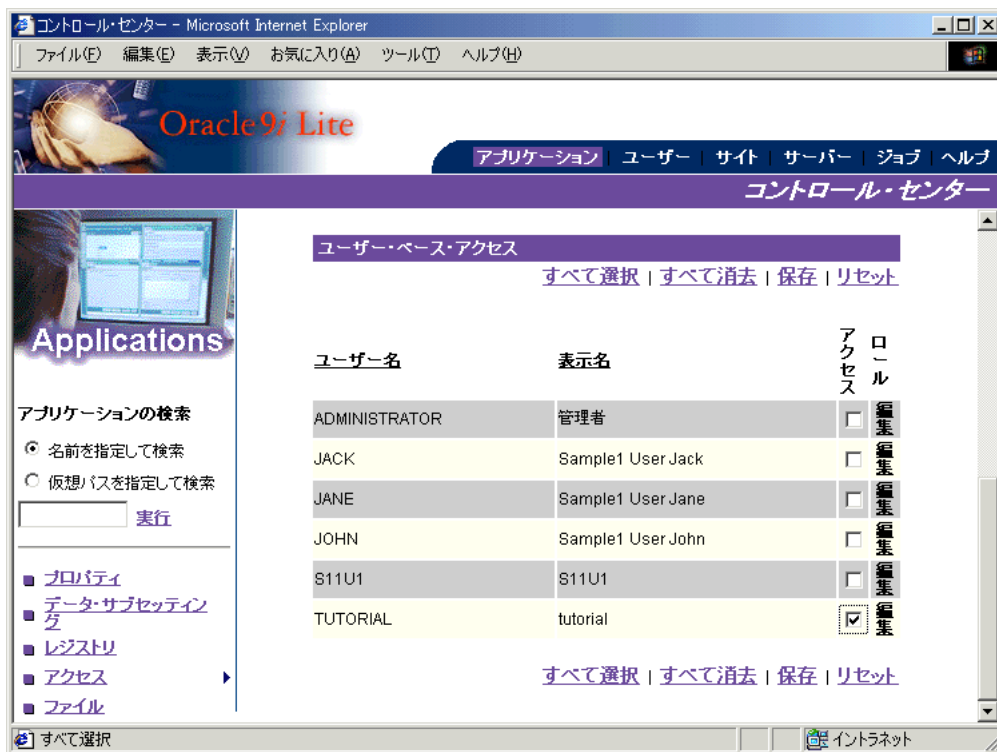
このステップでは、ユーザー TUTORIAL に対して「To Do List」アプリケーションへのアクセス権を付与します。

4.5.4.1 必須アクション

ユーザー TUTORIAL に対して「To Do List」アプリケーションへのアクセス権を付与するには、次の手順を実行します。

1. 左側のフレームで「アクセス」をクリックします。コントロール・センターが 2 つのリストを表示します。1 つのリストにはすべてのアプリケーション・ユーザーが含まれ、もう 1 つのリストにはすべてのアプリケーション・グループが含まれています。ユーザーまたはグループのアプリケーションに対するアクセス権の有無は、チェックボックスに示されています。
2. 「ユーザー・ベース・アクセス」リストでユーザー TUTORIAL を見つけます。次に、そのユーザー TUTORIAL のチェックボックスを選択します。
3. 「保存」をクリックします。これで、ユーザー TUTORIAL に「To Do List」アプリケーションへのアクセス権が付与されました。

図 4-25 アプリケーションへのユーザー・アクセス権の付与



4.5.5 ステップ 5: ユーザーのスナップショット・テンプレート値の定義

このステップでは、ユーザー TUTORIAL 用にスナップショット・テンプレート変数を定義します。各 Web-to-Go 用 Mobile クライアントは、同期時に同一のアプリケーション・データをダウンロードします。場合によっては、アプリケーションがダウンロードするデータを各ユーザーごとに指定することがあります。ユーザーのスナップショット・テンプレート変数を変更すると、これが可能になります。

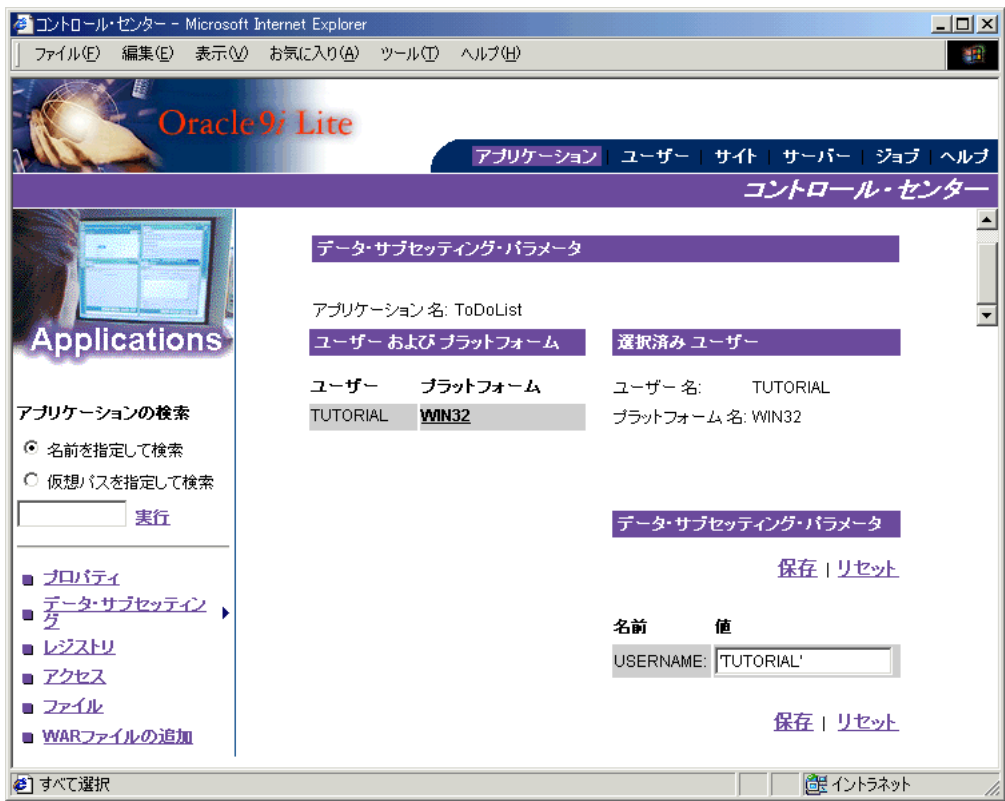
4.5.5.1 必須アクション

ユーザーのデータ・サブセッティング・パラメータを変更するには、次の手順を実行します。

1. 左側のフレームで「データ・サブセッティング」をクリックします。
2. ユーザーが TUTORIAL の場合は、プラットフォーム Win32 を選択します。

- 3. 右側にデータ・サブセッティング・パラメータが表示されます。
- 4. 引用符付きで 'TUTORIAL' と入力し、「保存」を選択します。

図 4-26 データ・サブセッティング・パラメータ



スナップショットの詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

これで「To Do List」アプリケーションの管理が正常に終了しました。

4.5.6 ステップ 6: Message Generator and Processor (MGP) の起動

このステップでは、Web-to-Go コントロール・センターから Consolidator Message Generator and Processor (MGP) を起動します。Web-to-Go 用 Mobile クライアントは同期時に、データ変更を Mobile サーバーにアップロードします。MGP は、Oracle データベースにデータ変更を適用する別プロセスです。

4.5.6.1 必須アクション

MGP を起動するには、次の手順を実行します。

1. Mobile サーバー・コントロール・センターで「サーバー」タブをクリックします。
2. 「サーバー」パネルの左側のフレームにある「MGP コントロール」をクリックします。
3. 「開始」ボタンをクリックします。右側のフレームには現在のモードとして「MGP を稼働中」が表示され、「ステータス」ボタンが表示されます。

図 4-27 MGP の開始



4.6 Web-to-Go 用 Mobile クライアントでのアプリケーションの実行

ここでは次の作業を実行します。

- Web-to-Go 用 Mobile クライアントのインストール
- Web-to-Go 用 Mobile クライアントへのログイン
- Web-to-Go 用 Mobile クライアントの同期

注意： アプリケーションは Mobile サーバー以外のマシン上にインストールしてテストする必要があります。

4.6.1 ステップ 1: Web-to-Go 用 Mobile クライアントのインストール

この項では、Mobile クライアント・セットアップ・プログラムを使用して Web-to-Go 用 Mobile クライアントをインストールします。このセットアップ・プログラムは、Web ブラウザを使用してダウンロードし実行する実行可能ファイルです。

4.6.1.1 必須アクション

Web-to-Go 用 Mobile クライアントをインストールするには、次の手順を実行します。

1. Web ブラウザを起動し、次の URL を入力して Mobile サーバーに接続します。

`http://server/webtogo/setup`

注意： `server` は、使用する Mobile サーバーのホスト名に置き換えてください。

次のテキストが含まれた Web ページが表示されます。

- Web-to-Go 用 Mobile クライアント
ダウンロードするにはここをクリックしてください。
- Win32 用 Mobile クライアント
ダウンロードするにはここをクリックしてください。
- ブランチ・オフィス
ダウンロードするにはここをクリックしてください。

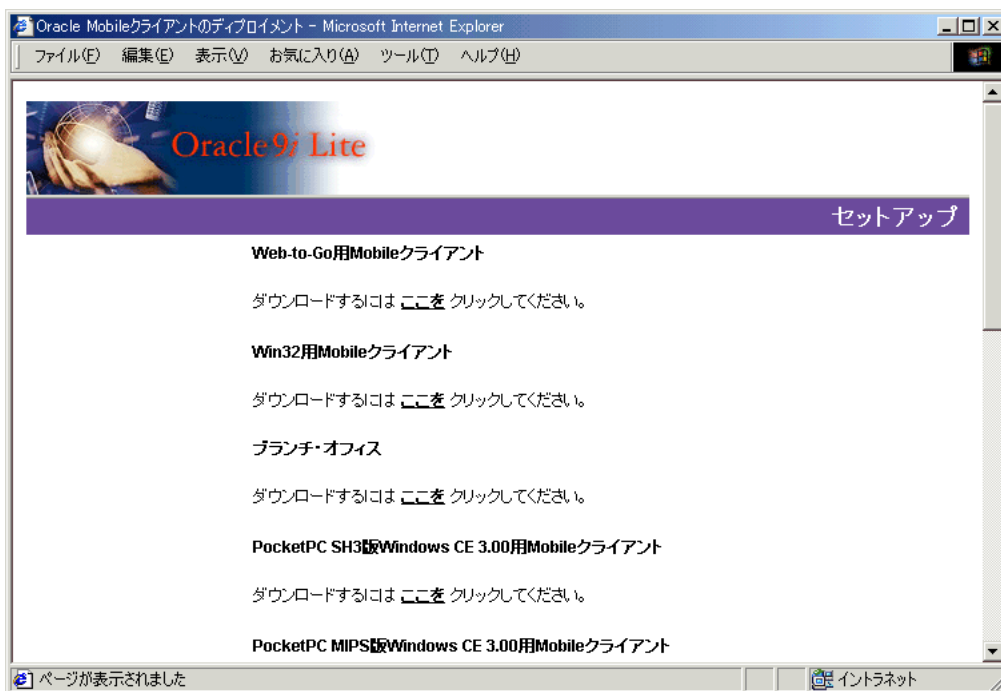
- 全プログラム

すべてのプログラムを表示するには、ここをクリックしてください。

(下線の付いた単語「ここを」は常にハイパーリンクです。)

2. 最初のハイパーリンク「ここを」をクリックして Web-to-Go 用 Mobile クライアントのセットアップ・プログラムにアクセスします。

図 4-28 Web-to-Go 用 Mobile クライアントのインストール



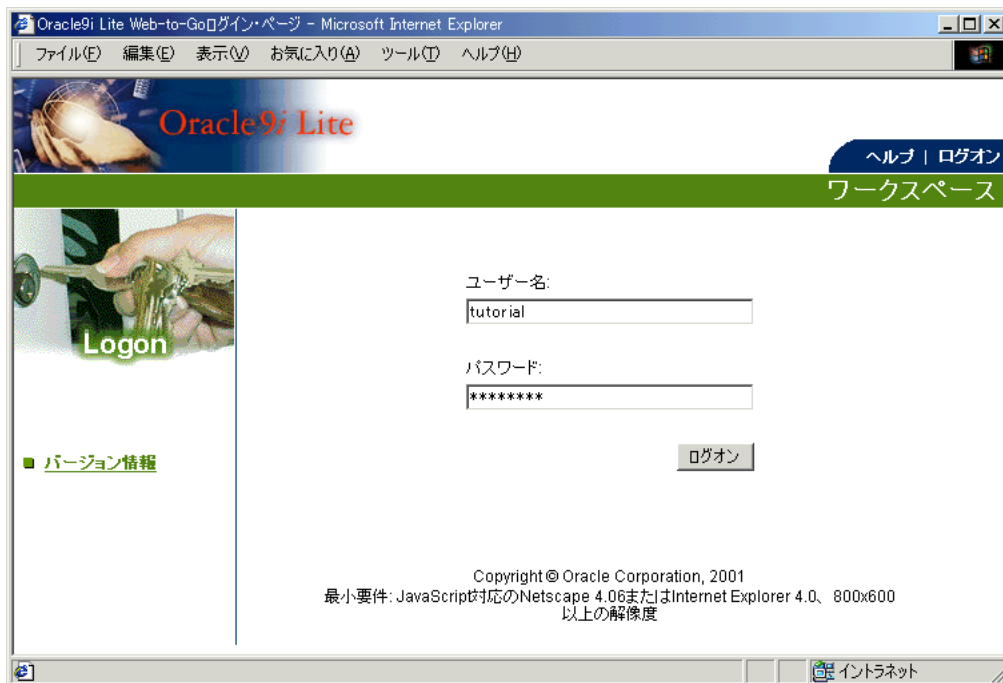
3. Netscape を使用している場合は、セットアップ・プログラムを保存する場所を選択して「OK」をクリックします。Windows エクスプローラで、**setup.exe** をダブルクリックしてセットアップ・プログラムを実行します。

Internet Explorer を使用している場合は、ブラウザのウィンドウからセットアップ・プログラムを実行します。

セットアップ・プログラムが起動されると、インストール・ディレクトリを指定するように求められます。

4. ディレクトリ（たとえば C:\orant）を選択し「OK」をクリックします。セットアップ・プログラムにより必要なコンポーネントがすべてダウンロードされ、使用するマシン上で Web-to-Go 用 Mobile クライアントが起動されます。インストールが完了すると、Web-to-Go のログオン・ページが表示されます。

図 4-29 Web-to-Go ログオン・ページ



4.6.2 ステップ 2: Web-to-Go 用 Mobile クライアントへのログイン

このステップでは、Web-to-Go 用 Mobile クライアントのセットアップを完了します。

4.6.2.1 必須アクション

ブラウザに Web-to-Go ログオン・ページが表示されます。ブラウザに Web-to-Go ログオン・ページが表示されていない場合は、次の URL を入力します。

`http://client`

注意： *client* 変数は、使用する Web-to-Go 用 Mobile クライアントのホスト名に置き換えてください。

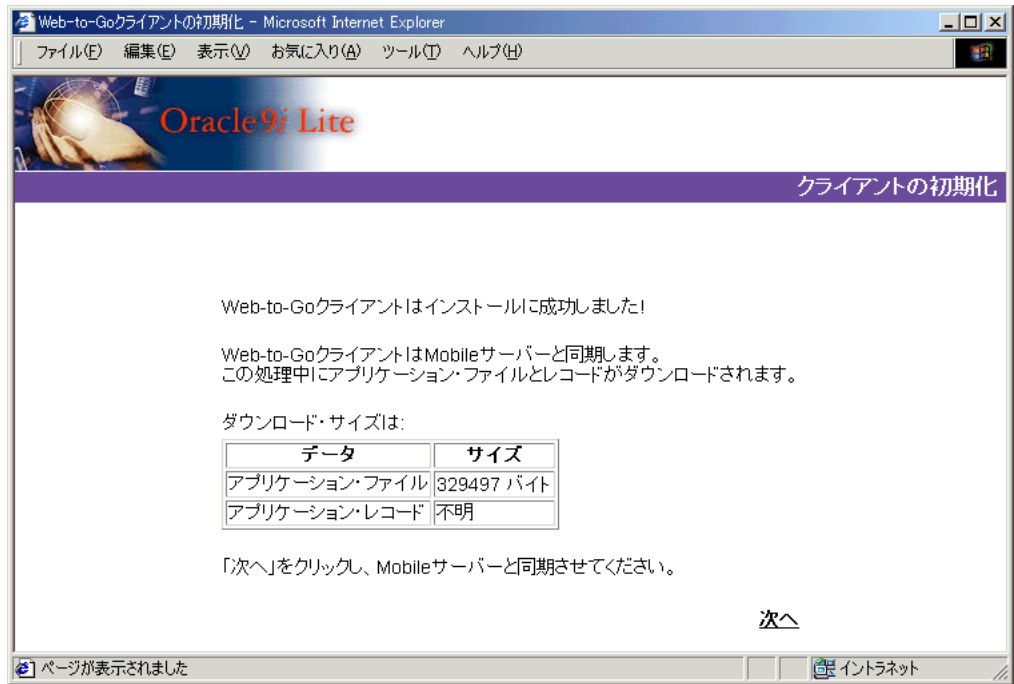
1. Web-to-Go にログオンします。これには、次の情報をログオン画面に入力し「ログオン」をクリックします。

フィールド	値
ユーザー名：	Tutorial
パスワード：	Tutorial

2. これが Web-to-Go 用 Mobile クライアントへの最初のログインであるため、セットアップを完了する必要があります。[図 4-30](#) に示すように、クライアントの初期化画面に次のメッセージが表示されます。

「Web-to-Go クライアントはインストールに成功しました！ Web-to-Go クライアントは Mobile サーバーと同期します。」

図 4-30 Web-to-Go 用 Mobile クライアントの初期化

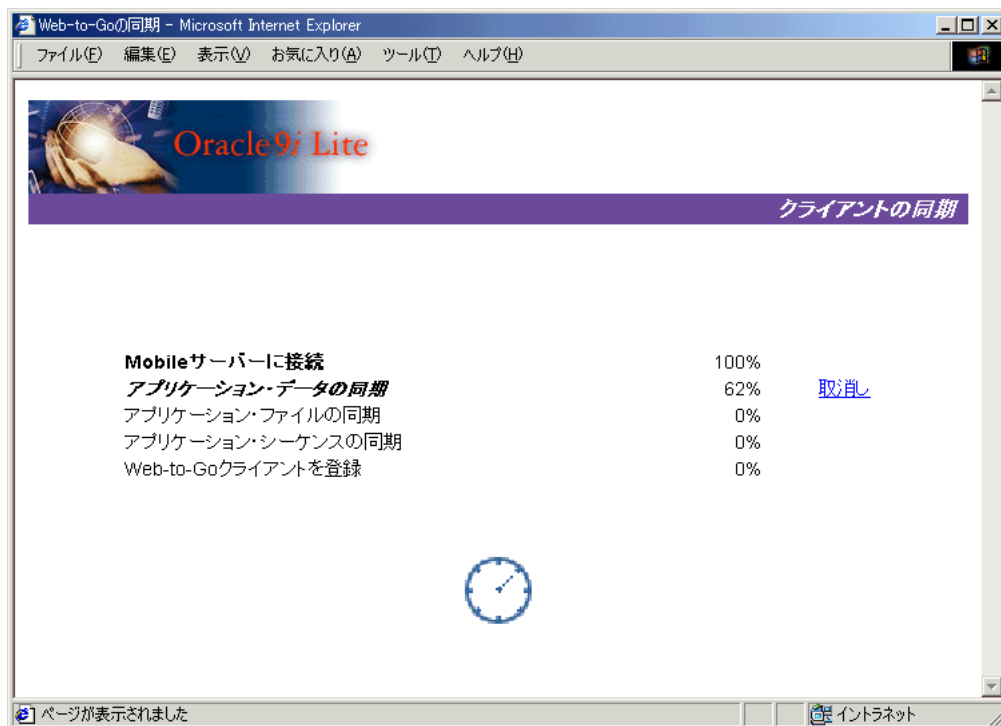


3. 「次へ」をクリックして、アプリケーションおよびデータのダウンロードを開始します。
図 4-31 に示すように、データの同期化の画面が表示されます。この図は、データの同期化の進行を表す画面に、次のメッセージが表示されているところを示しています。

「データの同期中です。お待ちください...」

画面の中央のグラフィックは、同期化中のデータを表しています。1 つの矢印が画面右側の Mobile サーバーを指し、もう 1 つの矢印が画面左側の Web-to-Go 用 Mobile クライアントを指しています。

図 4-31 「Web-to-Go の同期」画面



4. 同期処理が完了すると、Web-to-Go 用 Mobile クライアントは再起動する必要があります。これは自動的に行われます。図 4-32 に示すように、Web-to-Go 用 Mobile クライアントの再起動画面に次のメッセージが表示されます。

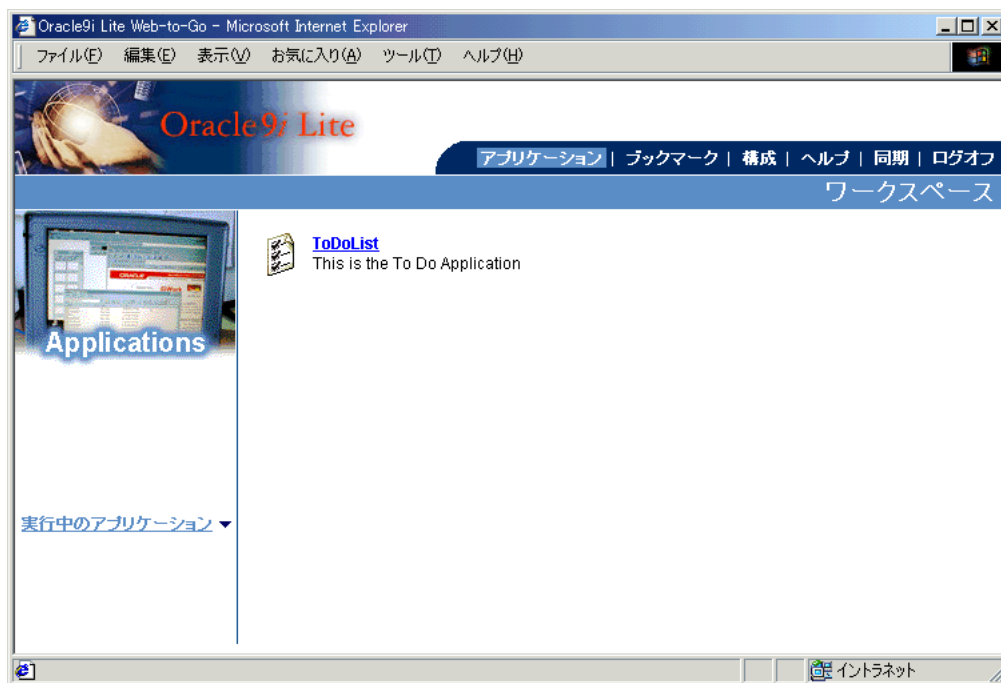
「新規または更新されたアプリケーション・ファイルがダウンロードされました。
Web-to-Go 用 Mobile クライアントが再起動されるまでお待ちください。」

図 4-32 「Web-to-Go 用 Mobile クライアントを再起動」画面



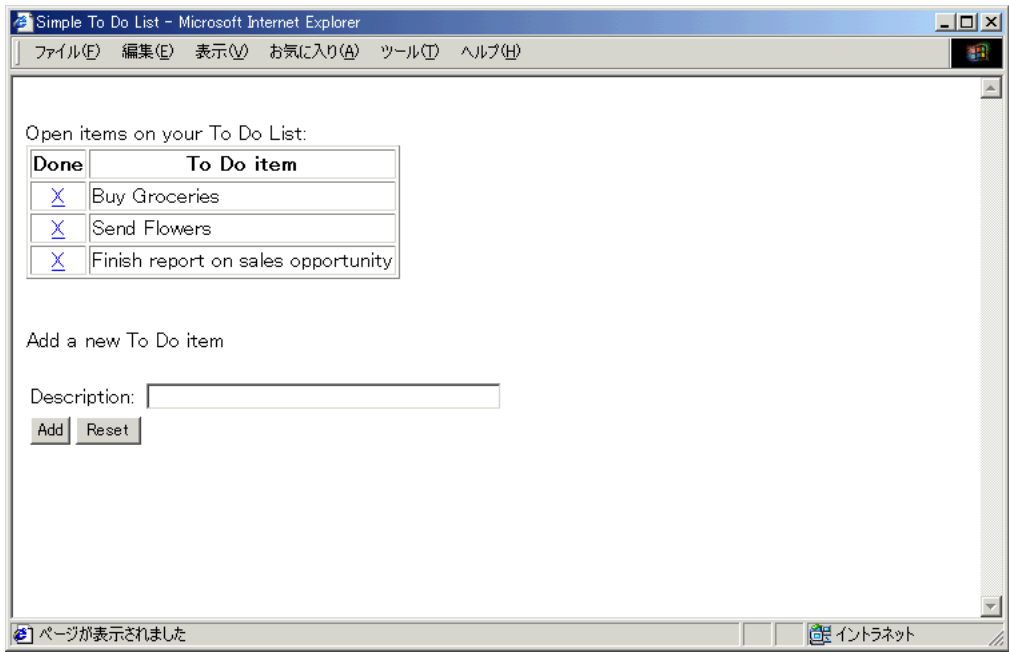
5. Web-to-Go 用 Mobile クライアントを再起動すると、図 4-33 に示すように、ワークスペースには、「To Do List」アプリケーション用のアイコンが 1 つと「ToDoList」というラベルの付いたリンクが表示されます。

図 4-33 同期プロセスの完了：アプリケーションのダウンロード終了



6. 「To Do List」アプリケーションのアイコンをクリックします。図 4-34 に示すように、Web-to-Go が「To Do List」アプリケーションをブラウザ内に起動します。

図 4-34 「To Do List」 アプリケーション



- 7. 何かテキストを入力して新しい「To Do」項目を作成し、「Add」をクリックしてこの項目をデータベースに保存します。
- 8. ブラウザのウィンドウをクローズしてアプリケーションを終了します。このアクションでワークスペースに戻ります。

4.6.3 ステップ 3: Web-to-Go 用 Mobile クライアントの同期

このステップでは、Web-to-Go 用 Mobile クライアントの同期を実行します。

4.6.3.1 必須アクション

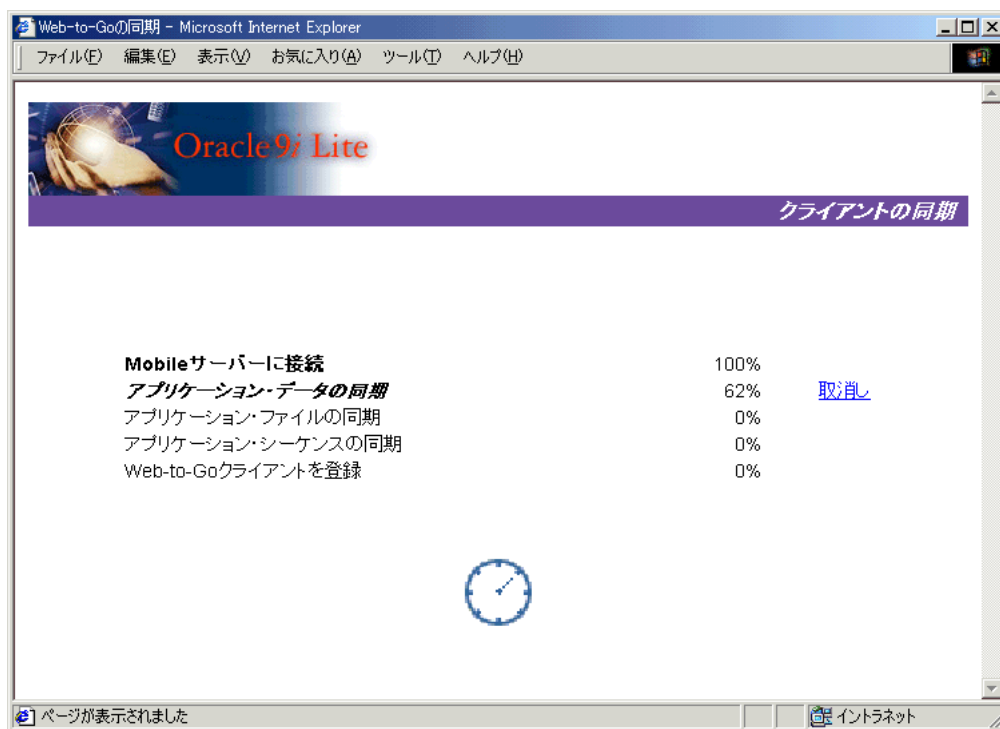
Web-to-Go 用 Mobile クライアントを Mobile サーバーと同期させるには、次の手順を実行します。

- 1. ワークスペースの右上にある「同期」タブを選択します。



Web-to-Go 用 Mobile クライアントは、アプリケーションとすべてのデータをローカル・マシンに同期します。

図 4-35 「Web-to-Go の同期」 ページ



同期プロセスが完了すると、ワークスペースが表示されます。

Web-to-Go アプリケーションの定義

この章では、Web-to-Go アプリケーションを定義およびパッケージ化し、管理者によってパブリッシュされる準備をする方法を説明します。内容は次のとおりです。

- 5.1 項「概要」
- 5.2 項「接続が切断されているクライアントのためのシーケンス・サポート」
- 5.3 項「パッケージ・ウィザードの使用」
- 5.4 項「スキーマ展開」
- 5.5 項「web.xml 形式のサポート」

5.1 概要

Web-to-Go アプリケーションを開発した後、パッケージ・ウィザードを使用してアプリケーションを定義します。パッケージ・ウィザードは、アプリケーション・ファイルとアプリケーションのメタ・データをパッケージ化して、Mobile サーバー・リポジトリにアップロードします。パッケージ・ウィザードを使用する前に、接続が切断されているアプリケーションに対して割り当てられるシーケンスのタイプを理解しておく必要があります。Oracle データベースの表をアドバンスト・レプリケーション用に構成する必要もあります。

5.2 接続が切断されているクライアントのためのシーケンス・サポート

多くのデータベース・アプリケーションが、シーケンス（順序）を使用して一意の識別子を生成しています。Web-to-Go アプリケーションでは、データベース表に新規レコードを挿入するときに、シーケンスを使用して一意の主キー値を生成します。オフライン・モードでは、シーケンスのサポートにより、Web-to-Go 用 Mobile クライアントはすべてのクライアントにわたって一意の主キー値を作成できます。

5.2.1 Web-to-Go シーケンス

Web-to-Go では、クライアントが Mobile サーバーと同期するときに、そのクライアントに対するシーケンスを作成して、Web-to-Go 用 Mobile クライアントに対する一意の主キー値を決定します。Web-to-Go では、WINDOW シーケンスを使用して、各シーケンスに一連の一意の数値が含まれるようにします。

Web-to-Go は、シーケンス定義を基にシーケンスをメンテナンスします。開発者は、パッケージ・ウィザードを使用してシーケンス定義を作成し、Mobile サーバーにパブリッシュします。次に、Web-to-Go がシーケンス定義を基に SQL 文を実行し、ローカルにシーケンスを作成します。ただし、シーケンス・オブジェクトは開発者自身が Oracle データベースに作成する必要があります。Oracle データベースにシーケンス・オブジェクトを作成するには、SQL の CREATE SEQUENCE 文を実行します。さらに、パッケージ・ウィザードは、オフラインの Web-to-Go アプリケーションに対してシーケンス・サポートを定義できます。

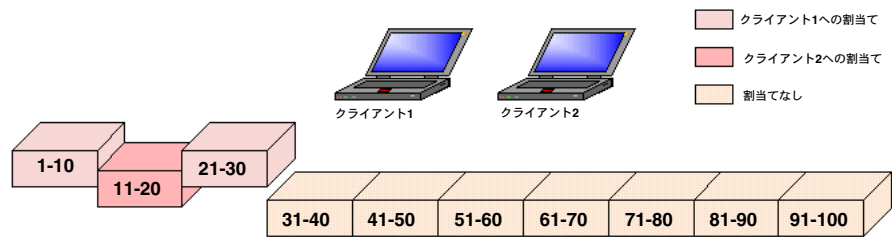
5.2.2 WINDOW シーケンス

WINDOW シーケンスは、各クライアントに一意の値範囲を割り当てます。他のクライアントと値の範囲は重複しません。クライアントがシーケンスの範囲内の値をすべて使用すると、Web-to-Go は新しい一意の値範囲を持つシーケンスを再作成します。Web-to-Go がシーケンスを再作成するのは、クライアントが Mobile サーバーと同期するときのみです。

Web-to-Go では、各 WINDOW シーケンスに対して、必要とする使用可能シーケンス数の最小値を含むしきい値を定義します。クライアントが Mobile サーバーと同期するとき、Web-to-Go は、シーケンス範囲がしきい値より大きいかどうかを検査します。大きくない場合、Web-to-Go は新しい一意の値範囲を持つシーケンスを自動的に再作成します。

5.2.2.1 WINDOW シーケンスの例

次の例では、WINDOW シーケンスが 2 つのクライアントに割り当てられます。



2つのクライアントに割り当てたWINDOWシーケンス

クライアント	シーケンスの範囲
クライアント 1	1-10
クライアント 2	11-20

クライアント 1 のシーケンス値が不足した場合、Web-to-Go は、21、22、...、30 のような一意の値を含む新しいシーケンスを作成します。

5.2.2.2 WINDOW シーケンスの作成

Web-to-Go では、Mobile サーバー・リポジトリ内のシーケンス定義を基に、クライアント上に WINDOW シーケンスを作成しメンテナンスします。たとえば、Mobile サーバー・リポジトリに次のような WINDOW シーケンス定義があるとします。

シーケンス・パラメータ	定義
NAME	AUDIODB_SEQ
TYPE	WINDOW
START_VALUE	1
INCREMENT	1
WINDOW_SIZE	200
THRESHOLD	25

最初のクライアントが接続を切断してオフライン・モードになったとき、Web-to-Go は次の SQL 文を実行してローカルにシーケンスを作成します。

```
CREATE SEQUENCE audiodb_seq START WITH 1 MAXVALUE 200 INCREMENT BY 1;
```

2 番目のクライアントが接続を切断してオフライン・モードになったとき、Web-to-Go は次の SQL 文を実行してローカルにシーケンスを作成します。

```
CREATE SEQUENCE audiodb_seq START WITH 201 MAXVALUE 400 INCREMENT BY 1;
```

Web-to-Go では割り当てられた範囲値を追跡し、各クライアントに必ず一意の範囲が割り当てられるようにします。最初のクライアントのシーケンスが 175 を超えた場合、使用可能な範囲は、指定されたしきい値の 25 より小さくなります。Web-to-Go は、この状況を検出し、クライアントが次に Mobile サーバーと同期するときにシーケンスを再作成します。Web-to-Go は、次の SQL 文を実行することでこれを行います。

```
DROP SEQUENCE audiodb_seq;
```

```
CREATE SEQUENCE audiodb_seq START WITH 401 MAXVALUE 600 INCREMENT BY 1;
```

5.2.2.3 オンライン・モード用の WINDOW シーケンスの定義

オンライン・モードでは、全ユーザーが 1 つのシーケンスを共有します。このシーケンス値の範囲は多数のユーザーをサポートする必要があり、さらにオフライン・モードでユーザーに割り当てられる値の範囲と重複しません。次の方法のいずれかを使用すると、この要件を満たすことができます。

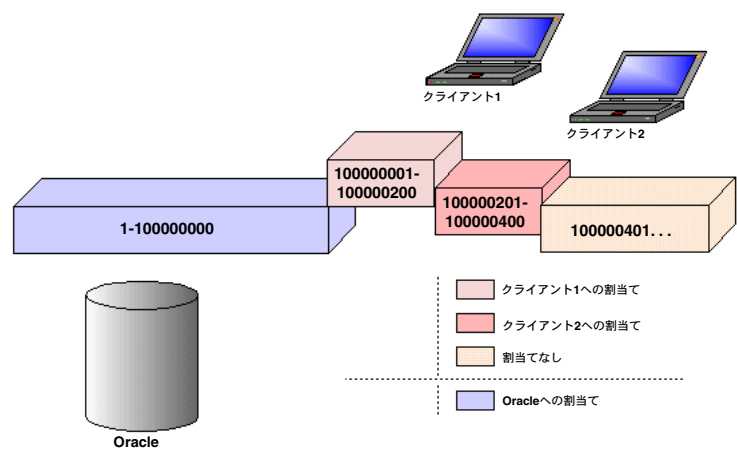
- Oracle シーケンス用として大きな値範囲を確保し、接続が切断されたクライアントにはこの範囲より上の値でシーケンスを作成できるようにします。
- オンライン・モードではシーケンスの数値に奇数を使用し、オフライン・モードでは偶数を使用します。この方法では、シーケンスの数値が必ず一意になるように、シーケンスを 2 ずつ増分することが必要になります。

大きな WINDOW シーケンス

Oracle データベースのシーケンス用に 1 ～ 100000000 の大きな範囲を確保するには、次の SQL 文を実行します。

```
CREATE SEQUENCE audiodb_seq START WITH 1 MAXVALUE 100000000 INCREMENT BY 1;
```

Web-to-Go は、接続が切断されたクライアントに対して、オンライン・ユーザー用に確保されている範囲よりも 1 増分値以上大きい数値から始まるシーケンス値を割り当てる必要があります。オンライン・ユーザーに確保されている範囲が 1 ～ 100000000 の場合は、接続を切断されたユーザーに割り当てられる最初の範囲の初期値は、100000001 以上にする必要があります。次の例では、Oracle データベース用にすでに確保されている大きな WINDOW シーケンスを考慮に入れた初期値を持つ WINDOW シーケンスを 2 人のクライアントに対して定義します。



Oracleに上位シーケンス番号を割り当てたWINDOWシーケンス

シーケンス・パラメータ	定義
NAME	AUDIODB_SEQ
TYPE	WINDOW
START_VALUE	100000001
INCREMENT	1
WINDOW_SIZE	200
THRESHOLD	25

クライアント	シーケンスの範囲
クライアント 1	100000001 - 100000200
クライアント 2	100000201 - 100000400

推奨：偶数と奇数の WINDOW シーケンス

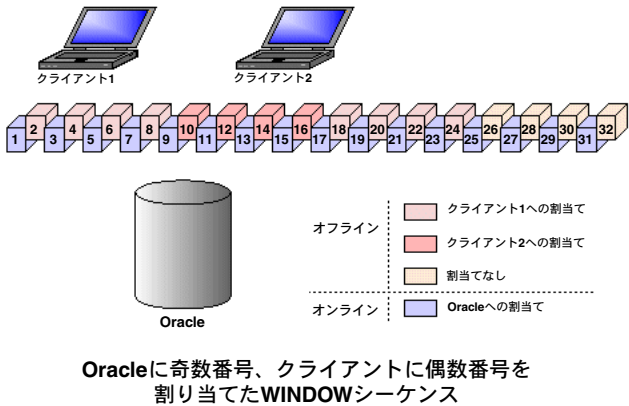
オンライン・モードとオフライン・モードのそれぞれに奇数または偶数のシーケンス値を割り当て、同じ増分値を指定すると、オンライン・モードとオフライン・モードに一意のシーケンス値を使用できます。次の手順により、オンライン・モード用に奇数のシーケンス値を確保し、オフライン・モードには偶数のシーケンス値の範囲を確保します。

Oracle データベース・データ・サーバー上で次の SQL 文を実行します。

```
CREATE SEQUENCE audiodb_seq START WITH 1 INCREMENT BY 2;
```

次のシーケンス定義をパッケージ・ウィザードに入力します。

シーケンス・パラメータ	定義
NAME	AUDIODB_SEQ
TYPE	WINDOW
START_VALUE	2
INCREMENT	2
WINDOW_SIZE	200
THRESHOLD	25



この方法を使用すると、オンライン用シーケンスの範囲は、Oracle シーケンス番号の上限 (約 10^{26}) によってのみ制限されます。オフライン・クライアントの場合は、WINDOW_SIZE パラメータおよび THRESHOLD パラメータで増分ステップは考慮されていません。オフライン用の手順では、接続が切断されたクライアントに存在する可能なシーケンス値は 100 (200/2) 個のみのため、すぐにシーケンスを再作成する必要があります。

5.3 パッケージ・ウィザードの使用

パッケージ・ウィザードは、次の目的に使用できるグラフィカル・ツールです。

- 新しい Web-to-Go アプリケーションの作成とパブリッシュ
- 既存の Web-to-Go アプリケーションの編集
- 簡単な配置のための Web-to-Go アプリケーションのパッケージ化
- Mobile サーバー・リポジトリへの Web-to-Go アプリケーションのパブリッシュ

新規 Web-to-Go アプリケーションを作成するときは、コンポーネントを定義して Mobile サーバー・リポジトリにパブリッシュします。場合によっては、既存の Web-to-Go アプリケーションのコンポーネントの定義を編集することがあります。たとえば、アプリケーションに新規サブレットを開発する場合は、パッケージ・ウィザードを使用してサブレットをアプリケーション定義に追加してから、変更したアプリケーションを Mobile サーバー・リポジトリにパブリッシュします。パッケージ・ウィザードを使用すると、簡単に配置できるようにアプリケーション・コンポーネントを **.jar** ファイルにパッケージ化することもできます。

パッケージ・ウィザードは、記述子ファイル **web.xml** などの WAR (Web Application Archive) 形式のファイルもサポートします。記述子ファイル **web.xml** の詳細は、[5.5 項「web.xml 形式のサポート」](#)を参照してください。WAR ファイルは、Java Servlet 2.3 仕様に従って JDeveloper などのツールを使用して作成します。

5.3.1 パッケージ・ウィザードの起動

パッケージ・ウィザードを起動するには、DOS プロンプトで次のように入力します。

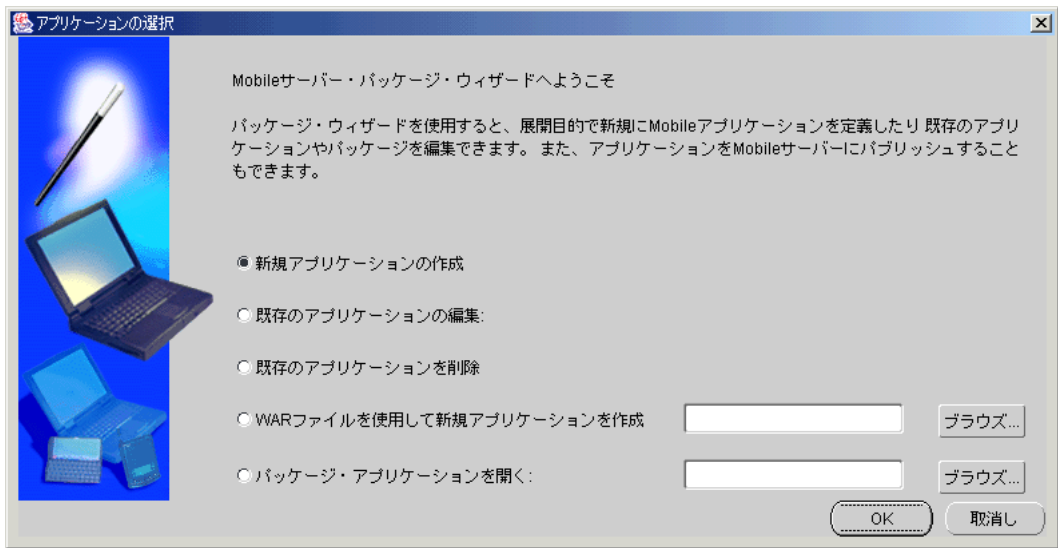
wtgpack

パッケージ・ウィザードが表示され、デフォルトで図 5-1 に示すような「アプリケーションの選択」ダイアログ・ボックスが表示されます。このダイアログ・ボックスでは、次の機能を使用して、パッケージ・アプリケーションを作成、編集、開くまたは削除できます。

機能	説明
新規アプリケーションの作成	このオプションを選択すると、新規アプリケーションを定義できます。
既存のアプリケーションの編集	このオプションを選択すると、既存のアプリケーションを編集できます。隣にあるドロップダウン・リストから既存のアプリケーションを選択できます。
既存のアプリケーションを削除	このオプションを選択すると、既存のアプリケーションを Mobile サーバー・リポジトリからでなく、XML ファイルから削除できます。

機能	説明
WAR ファイルを使用して新規アプリケーションを作成	このオプションを選択すると、WAR ファイルを使用してアプリケーションを作成できます。隣にあるフィールドに WAR ファイルの名前を入力するか、「ブラウズ」ボタンを使用してファイルを選択できます。
パッケージ・アプリケーションを開く	このオプションを選択すると、JAR ファイルとしてパッケージされたアプリケーションを選択できます。隣にあるフィールドにパッケージ・アプリケーションの名前を入力するか、「ブラウズ」ボタンを使用してアプリケーションを選択できます。

図 5-1 「アプリケーションの選択」ダイアログ・ボックス



5.3.2 プラットフォームの選択

プラットフォーム選択用のダイアログ・ボックスを使用して、アプリケーションをパッケージ化する対象プラットフォームを選択します。図 5-2 に示すように、このダイアログ・ボックスで、Web-to-Go、Palm、Win32 ネイティブまたは Handheld PC（ARM）などのプラットフォームを指定できます。WAR ファイルをパッケージ化する場合、Web-to-Go がプライマリ・アプリケーション・プラットフォームとして自動的に選択されます。

図 5-2 プラットフォーム選択用のダイアログ・ボックス



5.3.3 開発モードでのパッケージ・ウィザードの起動

パッケージ・ウィザードは、開発モードもサポートしています。このモードでパッケージ・ウィザードを使用して実行できるのは、Web-to-Go アプリケーション情報の定義、アプリケーション・ファイルのリスト、JSP のコンパイル、サーブレットの追加、およびレジストリの変更のみです。アプリケーションはローカル・マシンに対してパッケージ化されるため、接続情報やデータベース情報は必要ありません。

パッケージ・ウィザードを開発モードで起動するには、DOS プロンプトで次のように入力します。

```
wtgpack -d
```

次の項では、パッケージ・ウィザードの各パネルの機能を説明します。

5.3.4 新規アプリケーションの命名

「アプリケーション」パネルは、新しい Web-to-Go アプリケーションに名前を付けて、このアプリケーションを Mobile サーバー上のどこに格納するかを指定するために使用します。「アプリケーション」パネルには、[図 5-3](#) と次の表に示されているフィールドがあります。

フィールド	説明	必須
アプリケーション名	新しい Web-to-Go アプリケーションの名前です。 WAR ファイルをパッケージ化する場合、アプリケーション名に web.xml ファイル内の主要要素 <code><web-app></code> の下にある要素 <code><display name></code> の値を設定する必要があります。詳細は、 5.5 項「web.xml 形式のサポート」 を参照してください。	<input type="radio"/>
仮想パス	サーバー・リポジトリのルート・ディレクトリから Web-to-Go アプリケーション自体にマップされたパスです。仮想パスにより、アプリケーションのディレクトリ構造全体を参照する必要がなくなります。仮想ディレクトリ・パス内のすべてのサブディレクトリとすべてのファイルが、アプリケーションのパブリッシュ時に、まったく同じディレクトリ構造で Mobile サーバー・リポジトリにアップロードされることを示します。また、これによりアプリケーションが一意に識別されます。 アプリケーションのルート・ディレクトリ： 図 5-3 に示すように、名前 /Sample3 はアプリケーションの仮想パスを示します。アプリケーションの仮想パス名として入力した名前は、アプリケーションがパブリッシュされるときに、Mobile サーバー・リポジトリ内の アプリケーションのルート・ディレクトリ になります。したがって、仮想パス・フィールドに入力する名前がアプリケーションのルート・ディレクトリを指定できます。この名前はアプリケーション名とは別の名前でもかまいませんが、スペースは含めないでください。たとえば、アプリケーション名が Sales Office で、仮想パスを /Admin とすることも可能です。この場合、/Admin が Mobile サーバー・リポジトリ内のアプリケーションのルート・ディレクトリの名前になります。アプリケーションのルート・ディレクトリは、Mobile サーバー・リポジトリ内で実際にアプリケーション・ファイルが格納される場所です。 管理者がアプリケーションをパブリッシュするとき、パッケージ・ウィザードは、仮想パス名に入力された名前を自動的に Mobile サーバー・リポジトリ内のアプリケーションのルート・ディレクトリ名として使用します。ただし、アプリケーションをパブリッシュするときに管理者が別の名前を入力することで、Mobile サーバー・リポジトリ内のアプリケーションのルート・ディレクトリ名を変更できます。	<input type="radio"/>

フィールド	説明	必須
説明	Web-to-Go アプリケーションの簡単な説明です。 WAR ファイルをパッケージ化する場合、説明に web.xml ファイル内の主要要素 <web-app> の下にある要素 <description> の値を設定する必要があります。詳細は、 5.5 項「web.xml 形式のサポート」 を参照してください。	○
アプリケーションのクラスパス	アプリケーションのクラスパスは、アプリケーションのクラス（サーブレット、Bean）の場所を指定します。アプリケーションのデフォルトのクラスパスは、常にアプリケーションのルート・ディレクトリです。Web-to-Go がアプリケーションのクラスを検索する場所を追加指定するには、アプリケーションのクラスパスに他のディレクトリや、jar ファイルおよび zip ファイルを追加します。 エントリはセミコロン (;) で区切る必要があります。 さらに、アプリケーションのクラスパスには、Web-to-Go により次のクラスパスが自動的に付加されます。 1) アプリケーションのルート・ディレクトリ 2) パッケージ・ウィザードの「アプリケーション」パネルで指定されたクラスパス 3) WEB-INF/classes の下に置かれたクラス 4) ディレクトリ WEB-INF/lib 内に置かれたすべての jar ファイルと zip ファイル 5) /shared/WEB-INF/classes の下に置かれたクラス 6) ディレクトリ /shared/WEB-INF/lib 内に置かれたすべての jar ファイルと zip ファイル 7) SYSTEM クラスパス	×
デフォルトのページ	Web-to-Go アプリケーションのエントリ・ポイントとして機能する Web ページのサーバーでの場所です。これは、リポジトリのディレクトリを基にした相対パスです。たとえば、サーバー・ディレクトリが /apps で、デフォルトのページが index.htm の場合、「デフォルトのページ」への指定は /apps/index.htm になります。デフォルトのページはサーブレットでもかまいません。デフォルトのページを指定しない場合は、汎用ページが発行されます。 WAR ファイルをパッケージ化する場合、デフォルトのページに web.xml ファイル内の要素 <welcome-file-list> の値を設定する必要があります。詳細は、 5.5 項「web.xml 形式のサポート」 を参照してください。	○

フィールド	説明	必須
ローカル・アプリケーションのディレクトリ	<p>ローカル・マシン上でこのアプリケーションの全コンポーネントが含まれているディレクトリです。この場所は、入力するか「ブラウズ」ボタンをクリックして選択します。</p> <p>開発中は、アプリケーションのルート・ディレクトリはローカル・アプリケーションのディレクトリに設定します。</p> <p>注意：同一アプリケーションに異なるプラットフォーム用の複数のバージョンがある場合のプラットフォーム・アプリケーション・ファイルの配置の詳細は、この表の後の 5.3.4.1 項「ローカル・アプリケーション・ディレクトリでのプラットフォーム・ファイルの配置」を参照してください。</p>	○
アイコン	<p>Web-to-Go ワークスペースで Web-to-Go アプリケーションのアイコンとして使用される GIF イメージです。隣にあるフィールドにアイコンの名前を入力するか、「ブラウズ」ボタンを使用してアイコン・ファイルを選択します。</p> <p>WAR ファイルをパッケージ化する場合、説明フィールドに web.xml ファイル内の主要要素 <code><web-app></code> の下にある要素 <code><large-icon></code> の値をプライマリ選択肢として設定するか、または要素 <code><small-icon></code> の値をセカンダリ選択肢として設定する必要があります。詳細は、5.5 項「web.xml 形式のサポート」を参照してください。</p>	×

5.3.4.1 ローカル・アプリケーション・ディレクトリでのプラットフォーム・ファイルの配置

Mobile サーバー・リポジトリでは、同一アプリケーションの複数のバージョンをプラットフォームごとにパブリッシュして管理できます。同一アプリケーション（たとえば、Sales Force Automation）にプラットフォームごと（たとえば、Web-to-Go、Palm、WinCE および Windows 32）に複数の実装が存在し、それぞれが Oracle データベース・サーバーの同じアプリケーション表にアクセスできます。

たとえば、Compaq iPAQ 用に開発された C++ アプリケーションを、アプリケーション・コードを変更せずに Palm デバイス上で実行できるように再コンパイルできます。この場合、同一アプリケーションに 2 つのアプリケーション・バージョンが存在し、それぞれが同じデータベース表を使用します。

ローカル・アプリケーションのディレクトリは、異なるアプリケーション・バージョンが格納される、Windows 開発システム上のディレクトリです。各プラットフォーム用のアプリケーション・ファイルは、ローカル・アプリケーションのディレクトリの下、そのプラットフォーム専用のサブディレクトリに格納されます。このサブディレクトリは、プラットフォームに応じた識別名を持つ必要があります。パッケージ・ウィザードは、このアプリケーションの（ルート）ディレクトリの下、アプリケーション・ファイルを再帰的に読み込みます。

Web-to-Go 以外のプラットフォームで、複数のアプリケーションや同一アプリケーションの複数のアプリケーション・バージョンがある場合、各プラットフォームのサブディレクトリには、すべて表 5-1 に示された名前を正確に付ける必要があります。この表には、必須のサブディレクトリ名および対応するプラットフォームが示されています。

表 5-1 プラットフォームに応じた必須サブディレクトリ名のリスト

プラットフォーム	アプリケーション・ファイルの必須サブディレクトリ名
Windows 32	WIN32
WinCE SH3 WCE_SH3	wce¥<form_factor>¥<lang>¥SH3
WinCE SH4 WCE_SH4	wce¥<form_factor>¥<lang>¥SH4
WinCE MIPS WCE_MIPS	wce¥<form_factor>¥<lang>¥MIPS
WinCE StrongArm WCE_ARM	wce¥<form_factor>¥<lang>¥ARM
Palm	PALM

図 5-3 「アプリケーション」 パネル



例：

ローカル・マシン上のローカル・アプリケーションのディレクトリには、任意の名前（「Applications」など）を選択できます。このディレクトリには、次に示すように、異なるアプリケーション・バージョンがプラットフォームごとに格納されます。または Web-to-Go アプリケーション用として選択した独自のサブディレクトリ名で格納されます。

C:\Applications

Web-to-Go プラットフォーム用の Java アプリケーション・ファイルは、次に示すように Applications ディレクトリの直下に格納する必要があります。

C:\Applications\MyAPP

PALM などのプラットフォーム・タイプのディレクトリを指定しない場合、アプリケーションは Web-to-Go アプリケーションとして扱われます。Web-to-Go アプリケーションは、次に示すように、いずれのディレクトリにもパッケージ化して格納できます。

C:\

または

C:\Applications

または

C:\Applications\MyAPP

Palm プラットフォーム用の C++ アプリケーション・ファイルは、次に示すように Applications ディレクトリの下の \PALM サブディレクトリに格納する必要があります。

C:\Applications\PALM

iPAQ 用の C++ アプリケーション・ファイルは、\wce\<form factor>\<lang>\ARM サブディレクトリに格納する必要があります。<form factor> には、オペレーティング・システムを指定し、<lang> には使用する言語セットを指定します。詳細は、『Oracle9i Lite for Windows CE/Pocket PC 開発者ガイド』を参照してください。これらのサブディレクトリは、次に示すように Applications ディレクトリの下に格納されます。

C:\Applications\wce\pocket_pc\<lang>\SH4

ローカル・アプリケーションのディレクトリは C:\Applications ですが、3 つのサブディレクトリがあり、それぞれに特定のプラットフォーム用のアプリケーション・バージョンのファイルが格納されます。

C:\Applications // Local Application Directory

C:\Applications\MyAPP // Application subdirectory for the Web-to-Go version

C:\Applications\PALM // Application subdirectory for the Palm version

C:\Applications\WCE\Pocket_PC\us\SH4 // Application subdirectory for the Pocket
PC, US English, StrongArm version

5.3.5 アプリケーション・ファイルの表示

「ファイル」パネルは、アプリケーション・ファイルを表示し、ファイルがローカル・マシン上のどこにあるかを示すために使用します。パッケージ・ウィザードはローカル・アプリケーションのディレクトリの内容を分析し、各ファイルのローカル・パスを表示します。このパネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
ローカルのパス	各 Web-to-Go アプリケーション・ファイルの絶対パスです。リスト内の各エントリには、各ファイルまたはディレクトリの完全なパスが含まれています。	○

Web-to-Go 以外のプラットフォーム用のアプリケーションを選択する場合、アプリケーション・ファイルの配置方法については、そのプラットフォームの開発者ガイドを参照してください。

図 5-4 「ファイル」 パネル



「ファイル」パネルに表示されているファイルはすべて、追加、削除、ロードまたはコンパイルできます。新規アプリケーションを作成する場合、「ファイル」パネルに進むと、ローカル・ディレクトリに表示されているすべてのファイルが、パッケージ・ウィザードにより自動的に分析およびロードされます。既存のアプリケーションを編集する場合は、「ロード」ボタンを使用して個々のアプリケーション・ファイルをロードできます。

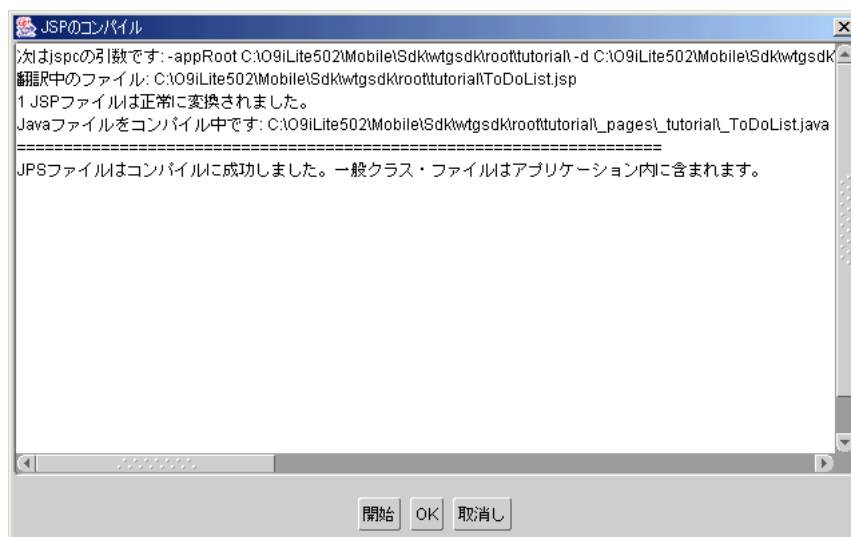
WAR ファイルを既存のアプリケーションにインポートする場合は、「ファイル」パネル上の「WAR ファイルのインポート」ボタンをクリックします。インポートする WAR ファイルの場所を入力するように要求するダイアログが表示されます。適切なファイルを見つけだし、「OK」ボタンをクリックすると、「ファイル」パネルに WAR ファイルの内容が表示されます。

5.3.5.1 JSP のコンパイル

「JSP のコンパイル」ボタンをクリックすると、「ファイル」パネルに表示されているすべての JSP ファイルがパッケージ・ウィザードによりコンパイルされます。

「JSP のコンパイル」ボタンを使用すると、JSP ファイルを配置用にコンパイルできます。「JSP のコンパイル」ボタンをクリックすると、次のような「JSP のコンパイル」ダイアログ・ボックスに詳細なコンパイル情報が表示されます。エラーがある場合は、該当する JSP ファイルを修正してから先へ進んでください。

図 5-5 「JSP のコンパイル」 ダイアログ・ボックス



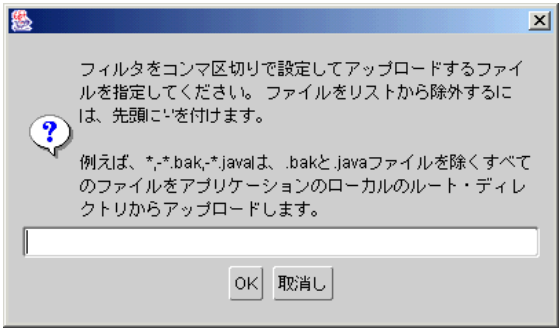
5.3.5.2 ソート

ファイルは、拡張子または含まれているディレクトリごとにソートできます。ファイルをソートするには、「拡張子順」または「ディレクトリ順」オプション・ボタンをクリックします。

5.3.5.3 フィルタ

「ロード」ボタンをクリックすると「入力」ダイアログ・ボックスが表示されます。「入力」ダイアログ・ボックスは、アップロード・プロセスからのアプリケーション・ファイルを含めるか除外するかを指定する（コンマで区切られた）フィルタのリストを作成するために使用します。ファイルを除外するには、ファイル名の前に負符号（-）を付けます。たとえば、**.bak** および **.java** 拡張子の付いたファイルを除くすべてのファイルをロードするには、次のように入力します。

***,-*.bak,-*.java**



5.3.6 サブレットの追加

パッケージ・ウィザードでは、「ファイル」タブにあるサブレットを分析して、Mobile サーバー上に定義できます。アプリケーションのサブレットは「サブレット」パネルに表示できます。このパネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
サブレット名	サブレットの名前です。たとえば、DeleteDetail です。サブレットは application_virtualpath/ サブレット名として参照します。	<input type="radio"/>
サブレットのクラス	追加するサブレットの完全修飾クラスです。	<input type="radio"/>

図 5-6 「サーブレット」 パネル



「サーブレット」 パネルに表示されているサーブレットは、すべて追加、削除またはロードできます。新規アプリケーションを作成する場合、「ファイル」タブに表示されているファイルを基にすべてのサーブレットがパッケージ・ウィザードにより自動的に表示されます。既存アプリケーションを編集する場合は、「ロード」ボタンを使用して個々のサーブレットをロードできます。

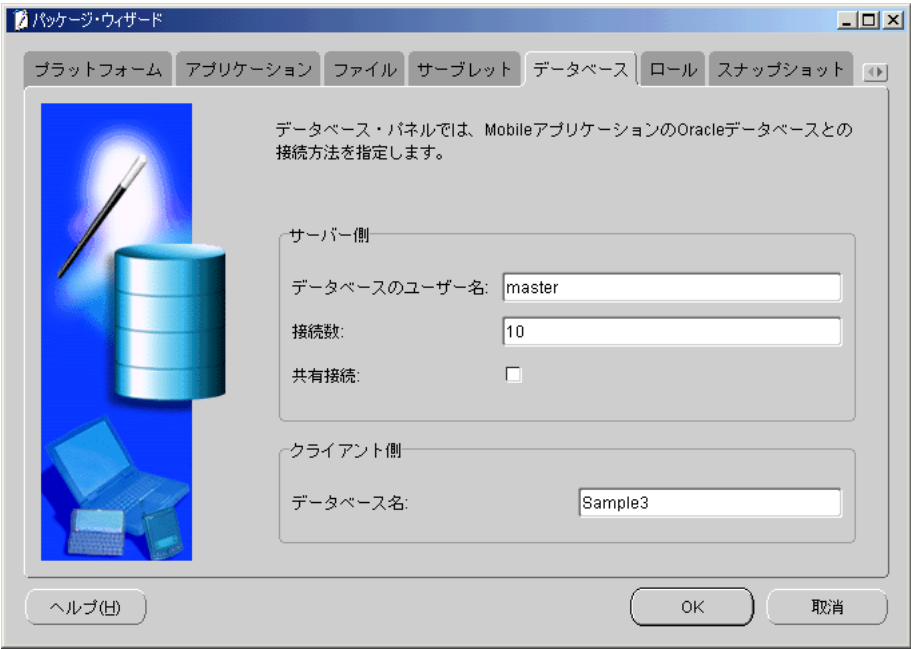
5.3.7 データベース情報の入力

「データベース」パネルは、Web-to-Go アプリケーション・ユーザーが Oracle サーバー上のレプリケーション・マスター・グループに接続する方法を指定するために使用します。このパネルに含まれるフィールドとチェックボックスは次のとおりです。

フィールド	説明	必須
データベースのユーザー名	Web-to-Go アプリケーションの Oracle データベース・アカウントのログイン名です。Web-to-Go アプリケーション・ユーザーはすべて同一のアカウントにアクセスします。アプリケーションには CONNECT.RESOURCE 権限が必要です。デフォルトのデータベースのユーザー名は「master」です。	<input type="radio"/>

フィールド	説明	必須
接続数	Web-to-Go アプリケーションから Oracle サーバーへの同時接続数です。	○
共有接続	これを選択すると、複数のサブレットで同一のデータベース接続を共有できます。複数のサブレットで接続を共有できると、Oracle の使用可能同時接続の最大値を超えるサブレットが接続されることを回避できます。ただし、あるサブレットがコミットを実行すると、接続を共有している他のサブレットにも影響します。	×
クライアント側データベース名	クライアント側に作成するデータベースの名前です。たとえば、Windows 32 ネイティブ・アプリケーションは、この名前を使用してクライアント・データベースにアクセスします。これは、Web-to-Go アプリケーションには必要ありません。	×

図 5-7 「データベース」 パネル



5.3.8 アプリケーション・ロールの定義

「ロール」パネルは、Web-to-Go アプリケーションのロールを定義するために使用します。開発者がアプリケーションのコード内にロールを作成し、パッケージ・ウィザードがこれを Oracle データベース用に宣言しなおします。アプリケーションを Mobile サーバーにパブリッシュした後は、Mobile サーバー・コントロール・センターを使用して、ユーザーとグループにロールを割り当てることができます。このパネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
ロール	Web-to-Go アプリケーションにロールを割り当てます。	

図 5-8 「ロール」パネル



Web-to-Go アプリケーションにはすべて、デフォルトのロールが含まれています。「ロール」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、ロールをパネルに追加または削除できます。

5.3.9 レプリケーション用スナップショットの定義

「スナップショット」パネルは、アプリケーションのレプリケーション・スナップショットを作成するために使用します。スナップショットはデータベース・オブジェクト（表またはビュー）と同じ名前を持ち、全アプリケーションを通して一意である必要があります。データベース・オブジェクトの作成時には、必ず一意の名前を使用してください。パッケージ・ウィザードでは、多数のプラットフォームに対してスナップショットを作成できます。Web-to-Go には、Windows 32 プラットフォームを使用します。

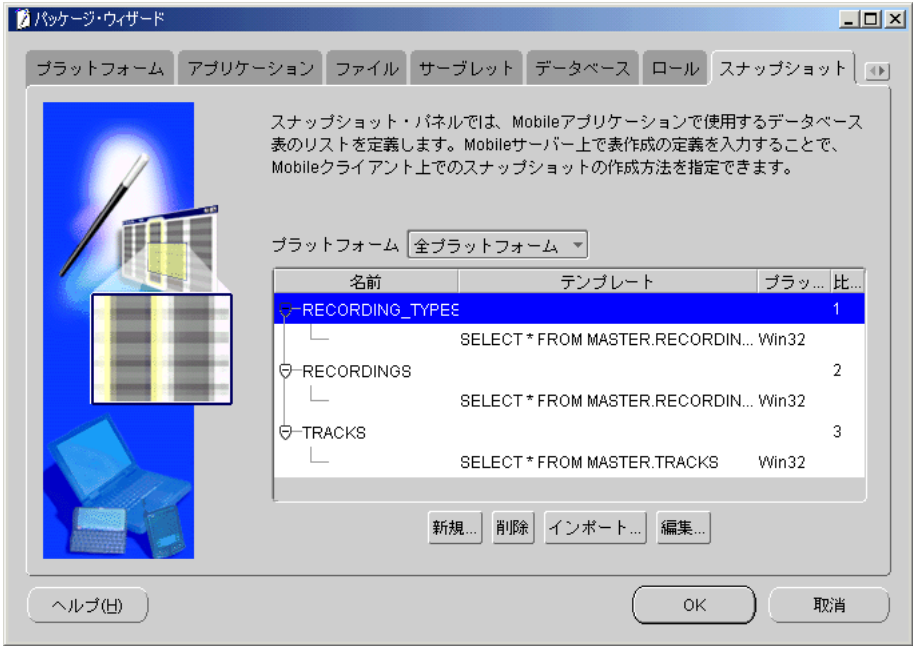
注意： データベース接続は、一度指定するとパッケージ・ウィザードの残りのセッションでも使用されます。Oracle データベースと Oracle Lite データベースを切り替える必要がありますが、すでに接続が確立されている場合は、パッケージ・ウィザード・アプリケーションを完全に終了して、再度 **wtgpack.exe** を実行します。

「スナップショット」パネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
名前	Web-to-Go アプリケーションに関連付けられているスナップショット（複数も可）の名前（複数も可）です。 基盤となるデータベース・オブジェクトと同じ名前である必要があります。	<input type="radio"/>
テンプレート	利用可能なスナップショット・テンプレートのリストです。テンプレートとは、スナップショットの作成に使用される SQL 文です。テンプレートには変数を含められます。テンプレートを Mobile サーバーにパブリッシュした後は、Mobile サーバー・コントロール・センターを使用して、ユーザー固有のテンプレート変数を指定できます。ただし、Mobile サーバー・コントロール・センターでスナップショットは変更できません。	<input type="radio"/>

フィールド	説明	必須
プラットフォーム	スナップショットのプラットフォームです。 Web-to-Go には、Win32 を使用します。ユーザーは異なるプラットフォームに対してスナップショットを作成できます。クライアントのデータを同期した場合、クライアント・アプリケーションを実行中のプラットフォームに適したスナップショットのみが取得されます。	
比率	表のレプリケートの順序です。マスター / ディテール関係を持つ表の場合、マスター表は最初にレプリケートする必要があるため、低い比率を持たせます。	
プラットフォーム	現在のスナップショットのプラットフォームのドロップダウン・リストです。ドロップダウン・リストには、次のプラットフォームをすべて含めることができます。 <ul style="list-style-type: none">■ Win32■ Palm■ Windows CE■ 全プラットフォーム ドロップダウン・リストからプラットフォームを選択すると、そのプラットフォーム用のスナップショットのみが「スナップショット」パネルに表示されます。たとえば、「全プラットフォーム」ドロップダウン・リストから「Win32」を選択すると、Win32 ベースのスナップショットのみが表示されます。ドロップダウンから「全プラットフォーム」オプションを選択すると、現在使用中のプラットフォームごとにすべてのスナップショットが表示されます。ユーザーが新規のスナップショットを追加した場合、ドロップダウン・リストには追加のプラットフォームがリストされます。	

図 5-9 「スナップショット」 パネル



「スナップショット」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、「スナップショット」パネルにスナップショットを追加または削除できます。「インポート」または「編集」ボタンをクリックして、スナップショットをインポートまたは編集することもできます。

注意：「スナップショット」パネルから複数のスナップショットをインポートできますが、「新規表」ダイアログ・ボックスから新規表を作成するときにインポートできるスナップショットは1つのみです。

5.3.9.1 新規スナップショットの作成

新規スナップショットを作成するには、「新規」ボタンをクリックします。「新規スナップショット」ダイアログ・ボックスが表示されます。図 5-10 に示すように、「サーバー」タブをクリックすると「サーバー」パネルが表示されます。このパネルには、スナップショット名、比率、所有者およびSQLを入力するフィールドと、SQLを生成するためのチェックボックスが含まれています。

図 5-10 「新規スナップショット」－「サーバー」パネル

The screenshot shows a Windows-style dialog box titled "新規スナップショット" (New Snapshot). It has a tab labeled "Win32". Inside the dialog, there are several input fields and buttons:

- A tab labeled "サーバー" (Server) is selected.
- A text field labeled "スナップショット名" (Snapshot Name) contains the text "sample3". To its right is a button labeled "インポート..." (Import...).
- A text field labeled "比率" (Ratio) contains the text "1".
- A text field labeled "所有者" (Owner) contains the text "master". Below this field is a yellow button labeled "所有者" (Owner).
- A checkbox labeled "SQLの生成" (Generate SQL) is unchecked.
- A large text area labeled "SQL:" is empty.
- At the bottom of the dialog are two buttons: "OK" and "取消し" (Cancel).

適切な情報を入力して、新しいスナップショットを作成します。「スナップショット名」フィールドに入力する名前は、基盤となるデータベース表名を示します。

「比率」の説明は、[5.3.9 項「レプリケーション用スナップショットの定義」](#)を参照してください。

「SQL の生成」を有効にする場合、データベース表を構成する CreateTable SQL 文を指定する必要があります。「SQL」フィールドに SQL 文を入力します。定義作業の最後に、SQL ファイルを生成するオプションがあります。

Web-to-Go 用 Mobile クライアントの場合は、「Win32」タブを使用します。

「Win32」タブをクリックすると、次のパネルが表示されます。

図 5-11 「新規スナップショット」－「Win32」パネル



「新規スナップショット」ダイアログ・ボックスで次の機能を変更して、Web-to-Go 用 Mobile クライアントに新規スナップショットを作成します。

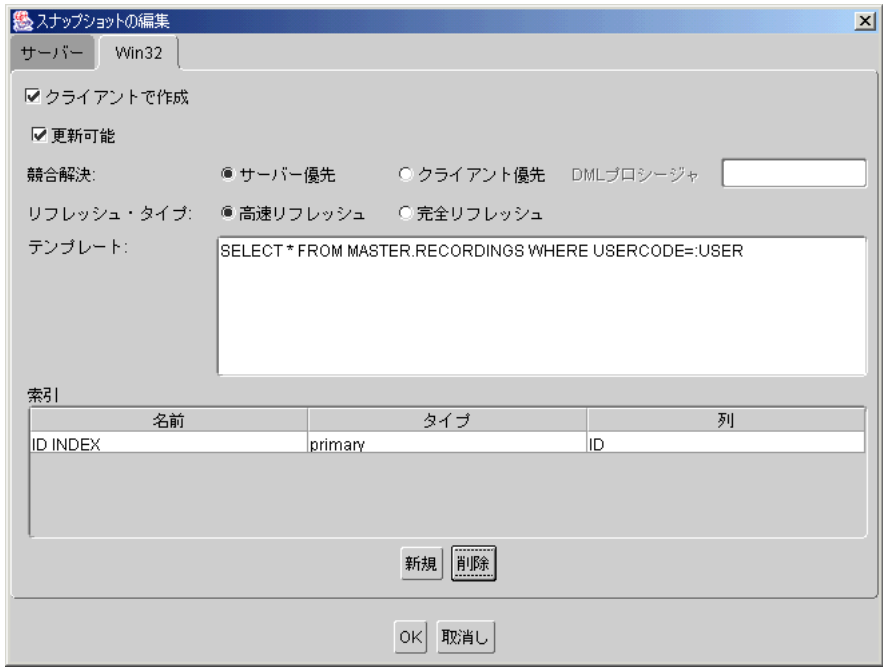
機能	説明
更新可能	このチェックボックスを選択すると、名前付きの表の更新可能スナップショットが作成されます。
テンプレート	名前付きの表のスナップショット・テンプレートを表示します。スナップショット・テンプレートは変更できます。管理者は、 Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数のインスタンスをこのテンプレートに生成できます。テンプレート変数の詳細は、「 レプリケーション用スナップショットの定義 」を参照してください。

5.3.9.2 スナップショットの索引の作成

パッケージ・ウィザードを使用してスナップショットの索引を作成するには、次の手順を実行します。

- 1. 「スナップショット」パネル（図 5-9 「スナップショット」パネル）から「編集」ボタンを選択して、既存のスナップショットから索引を作成します。スナップショットおよび索引を新規作成する場合は、「新規」ボタンを選択します。

図 5-12 索引の作成



- 2. 表示されるパネルでプラットフォーム・タブ（たとえば、Win32）を選択します。スナップショットを定義している SQL 文が「テンプレート」フィールドに表示されます。その下に「索引」表があります。新規索引を作成するには、この表の下にある「新規」ボタンを選択します。
- 3. 「索引」表には、次の 3 列があります。
 - 名前 — 索引の名前です。
 - タイプ — ドロップダウン・メニューから、「Regular」、「Primary」、「Unique」のいずれかのタイプを選択します。
 - 列 — 索引で使用する列名を入力します。

5.3.9.3 スナップショットのインポート

Oracle データベースまたは Oracle Lite データベースからスナップショットをインポートするには、「インポート」ボタンをクリックします。接続を指定していない場合は、データベース接続ウィンドウが表示されます。



スナップショットのインポート元の Oracle データベースまたは Oracle Lite データベースのユーザー名、パスワードおよびデータベース URL を入力します。「表」ウィンドウが表示されます。

注意： Oracle データベースのデータベース URL を入力する場合は、`jdbc:oracle:oci8:@webtogo.world` という書式を使用します。Oracle Lite の場合は、`jdbc:polite:webtogo` を使用します。



表のインポート元のスキーマを選択してから、表を選択します。「追加」をクリックしてから「閉じる」をクリックします。パッケージ・ウィザードの「スナップショット」パネルに表が表示されます。

5.3.9.4 スナップショットの編集

スナップショットを編集するには、「スナップショット」パネルからスナップショットを選択し、「編集」をクリックします。「スナップショットの編集」ダイアログ・ボックスが表示されます。

図 5-13 「スナップショットの編集」ダイアログ・ボックス（クライアント）



スナップショットを編集するには、「スナップショットの編集」ダイアログ・ボックスの次の機能を変更します。

機能	説明
クライアントで作成	このチェックボックスを選択すると、Web-to-Go 用 Mobile クライアント上のスナップショットを編集できます。
更新可能	このチェックボックスを選択すると、名前付きの表の更新可能スナップショットが作成されます。

機能	説明
テンプレート	名前付きの表のスナップショット・テンプレートを表示します。スナップショット・テンプレートは変更できます。管理者は、Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数のインスタンスをこのテンプレートに生成できます。

5.3.10 レプリケーション用のシーケンスの定義

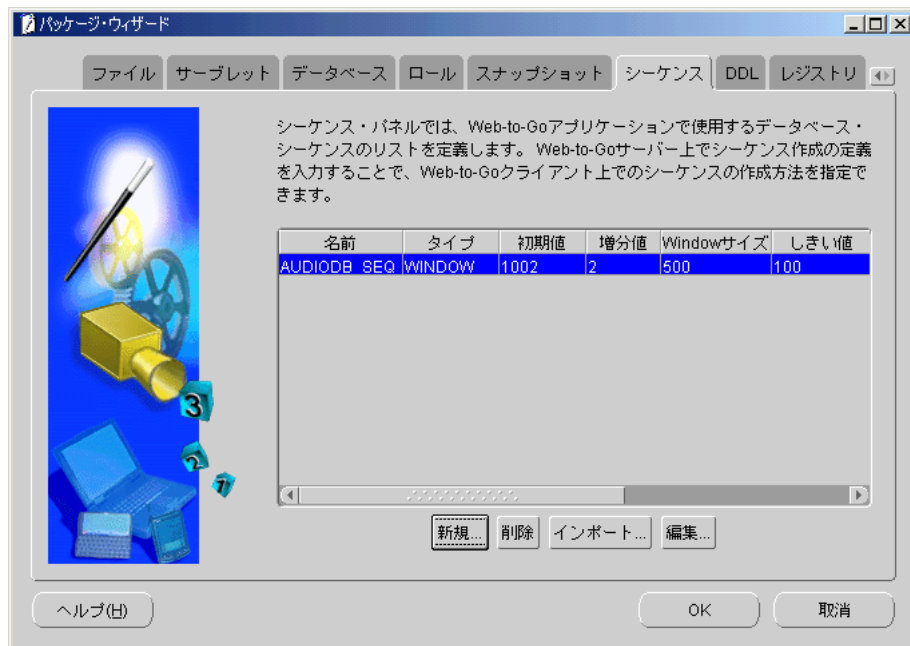
「シーケンス」パネルは、Web-to-Go アプリケーションのオフライン・シーケンスを定義するために使用します。Web-to-Go では、アプリケーションが接続を切断してオフライン・モードになる前に、シーケンスを使用してアプリケーションに一意の主キー値を割り当てます。これらの一意の主キー値は、クライアントがオンラインに戻ったときにレプリケーション用として使用されます。シーケンスという概念は重要ですが、これは、シーケンスを使用することにより、接続が切断されているアプリケーション間で主キー値の重複がなくなり、レプリケーションで競合が発生しなくなるためです。シーケンスにはすべて一意の名前が必要です。シーケンス名の前にアプリケーション名を付けてシーケンス名を変更すると、一意にできます。シーケンスの詳細は、この章の「[接続が切断されているクライアントのためのシーケンス・サポート](#)」を参照してください。

「シーケンス」パネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
名前	切断モードで Web-to-Go アプリケーションにより使用されるシーケンスの名前です。	<input type="radio"/>
タイプ	<p>切断モードで Web-to-Go アプリケーションにより使用されるシーケンスのタイプです。シーケンスには、WINDOW と LEAPFROG という 2 つのタイプがあります。</p> <p>WINDOW. WINDOW シーケンスは、各クライアントに一意の値範囲を割り当てます。WINDOW シーケンスは各クライアントごとに一意で、他のクライアントのシーケンスとは重なりません。クライアントがシーケンスの範囲内の値をすべて使用すると、Web-to-Go はクライアントが次にオフラインになったときに新しい一意の値範囲を持つシーケンスを再作成します。</p> <p>LEAPFROG. LEAPFROG シーケンスは、各クライアントに一意の一連の増分値を割り当てます。各シーケンスの初期値はクライアントごとに異なり、各シーケンスの増分は最大クライアント数より大きい値に設定されます。</p>	<input type="radio"/>

フィールド	説明	必須
初期値	Web-to-Go 用 Mobile クライアント上のシーケンスの初期値です。シーケンスはこの数値から開始し、定義された増分値に従って増分されます。	○
増分値	Web-to-Go 用 Mobile クライアント上でシーケンスが初期値から始まって増分される数値です。	○
Window サイズ	WINDOW シーケンスの場合に、数値の範囲を指定します。この情報は LEAPFROG シーケンスでは使用されません。	WINDOW のみ
しきい値	WINDOW シーケンスの場合に、必要な数の最小範囲を定義します。既存のシーケンスがこの範囲に達したときおよびクライアントがオフラインになったときに、Web-to-Go は新しいシーケンスを作成します。この情報は LEAPFROG シーケンスでは使用されません。	WINDOW のみ
サーバー側シーケンスの初期値	Oracle データベース上のシーケンスの初期値です。シーケンスはこの数値から開始し、定義された増分値に従って増分されます。この数値は、Web-to-Go 用 Mobile クライアント上のシーケンスの初期値とは異なる値にする必要があります。	
サーバー側シーケンスの増分値	Oracle データベース上でシーケンスが初期値から始まって増分される数値です。	
サーバー側シーケンスの最小値	Oracle データベース上の昇順シーケンスの最小の初期値です。たとえば、昇順シーケンスは 1 で始まり、昇順に増分します。	
サーバー側シーケンスの最大値	Oracle データベース上の降順シーケンスの最大の初期値です。たとえば、降順シーケンスは -1 で始まり、降順に減分します。	

図 5-14 「シーケンス」 パネル



「シーケンス」パネルで「新規」ボタンまたは「削除」ボタンをクリックすれば、シーケンスをパネルに追加または削除できます。

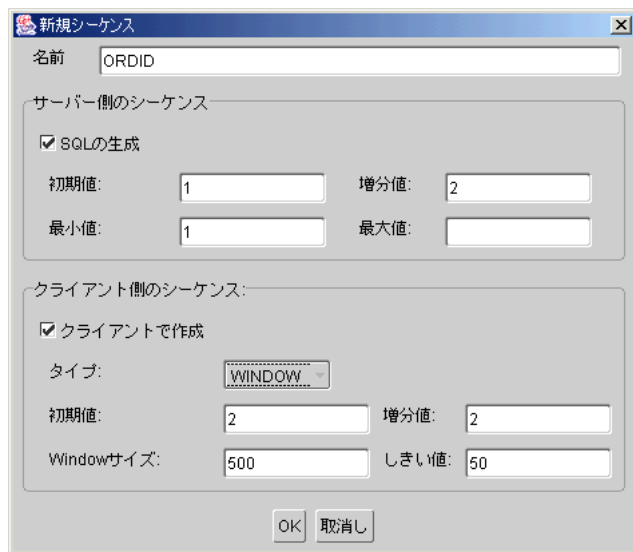
5.3.10.1 シーケンスのインポート

Oracle データベースからシーケンスをインポートするには、「インポート」ボタンをクリックします。「シーケンス」ウィンドウが表示されます。



インポートするシーケンスを選択し、「追加」をクリックしてから「閉じる」をクリックします。

シーケンスを編集するには、「シーケンス」パネルからシーケンスを選択し、「編集」をクリックします。「新規シーケンス」ウィンドウが表示されます。



シーケンスを編集するには、「新規シーケンス」ウィンドウの次の機能を変更します。

機能	説明
名前	シーケンスの名前です。
SQL の生成	このチェックボックスを選択すると、Oracle データベース上にシーケンスを作成するオプションが使用可能になります。ユーザーが入力した情報を使用して SQL スクリプトが作成され、Oracle Server 上にシーケンスが作成されます。
初期値	Oracle データベース上のシーケンスの初期値です。
増分値	Oracle データベース上でシーケンスが初期値から始まって増分される数値です。
最小値	Oracle データベース上の昇順シーケンスの最小の初期値です。たとえば、昇順シーケンスは 1 で始まり、昇順に増分できます。
最大値	Oracle データベース上の降順シーケンスの最大の初期値です。たとえば、降順シーケンスは -1 で始まり、降順に減分します。
クライアントで作成	このチェックボックスを選択すると、Web-to-Go 用 Mobile クライアント上にシーケンスを作成するオプションが使用可能になります。
タイプ	Web-to-Go 用 Mobile クライアント上のシーケンスのタイプを定義します。オプションには WINDOW シーケンスと LEAPFROG シーケンスがあります。
初期値	Web-to-Go 用 Mobile クライアント上のシーケンスの初期値です。
増分値	Web-to-Go 用 Mobile クライアント上でシーケンスが初期値から始まって増分される数値です。
Window サイズ	Web-to-Go 用 Mobile クライアント上の WINDOW シーケンスを構成する数値の範囲です。この情報は LEAPFROG シーケンスでは使用されません。
しきい値	WINDOW シーケンスにおける、必須数値の最小範囲です。既存のシーケンスがこの範囲に達したときとクライアントがオフラインになったときに、Web-to-Go は新しいシーケンスを作成します。この情報は LEAPFROG シーケンスでは使用されません。

5.3.11 アプリケーションの DDL の定義

「DDL」パネルは、Web-to-Go アプリケーションが初めてオフラインに切り替えられたときに実行できる DDL（データ定義言語）文を定義するために使用します。DDL 文にはすべて一意の名前が必要です。これを行う 1 つの方法は、DDL 名の前にアプリケーション名を付けて DDL 名を変更することです。アプリケーションを Mobile サーバーにパブリッシュした後は、Mobile サーバー・コントロール・センターを使用して、追加の DDL 文を作成できます。このパネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
名前	DDL 名です。	
DDL 文	DDL 文を Web-to-Go アプリケーションに定義します。DDL 文は、Web-to-Go アプリケーションがクライアントで実行されたときに実行されます。	

図 5-15 「DDL」パネル



「DDL」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、DDL をパネルに追加または削除できます。「新規」ボタンをクリックすると「新規 DDL」ダイアログ・ボックスが表示されます。



フィールド	説明	必須
名前	DDL 名です。	<input type="radio"/>
SQL	DDL 文を定義します。Web-to-Go アプリケーションがクライアントで実行されたときに、これらの SQL 文が実行されます。	<input type="radio"/>

5.3.11.1 ビューや索引の定義のインポート

Oracle データベースからビューおよび索引の定義をインポートするには、「インポート」ボタンをクリックします。「DDL のインポート」ウィンドウが表示されます。



索引定義をインポートするには、「索引」タブをクリックし、索引のインポート元のスキーマをクリックします。インポートする索引を選択し、「追加」をクリックしてから「閉じる」をクリックします。

ビュー定義をインポートするには、「ビュー」タブをクリックし、ビューのインポート元のスキーマをクリックします。インポートするビューを選択し、「追加」をクリックしてから「閉じる」をクリックします。

5.3.12 レジストリでの名前と値のペアの定義

「レジストリ」パネルは、Web-to-Go アプリケーションの名前と値のペアを定義するために使用します。レジストリには、Web-to-Go の一意の名前と値のペアが含まれます。名前はすべて一意にする必要があります。これを行う 1 つの方法は、名前の前にアプリケーション名を付けて名前を変更することです。このパネルに含まれるフィールドは次のとおりです。

フィールド	説明	必須
名前	名前です。	<input type="radio"/>
値	名前に対応する値です。	<input type="radio"/>

図 5-16 「レジストリ」パネル



「レジストリ」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、名前と値のペアをパネルに追加または削除できます。

5.3.13 アプリケーションの完了

パッケージ・ウィザードの全パネルを完了すると、次のオプションの含まれた「アプリケーションの定義の完了」ウィンドウが表示されます。

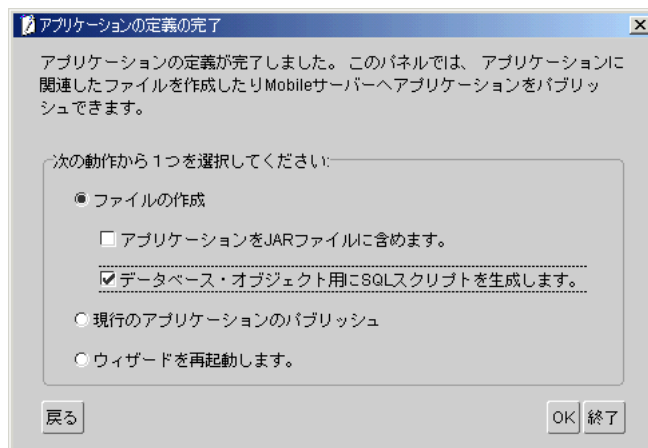
- ファイルの作成
- 現行のアプリケーションのパブリッシュ
- ウィザードを再起動します。

5.3.13.1 XML ファイル

パッケージ・ウィザードは、アプリケーション情報のすべてを XML ファイルに自動的に出力します。パッケージ・ウィザードはローカル・マシン上で XML ファイルを保持します。さらに、Mobile サーバーに XML ファイルをパブリッシュするオプションも提供しています。Mobile サーバーの実行中は、XML ファイルのみをパブリッシュできます。パッケージ・ウィザードは常に、Web-to-Go アプリケーションの XML ファイルをローカル・マシン上に保持します。

5.3.13.2 ファイルの作成

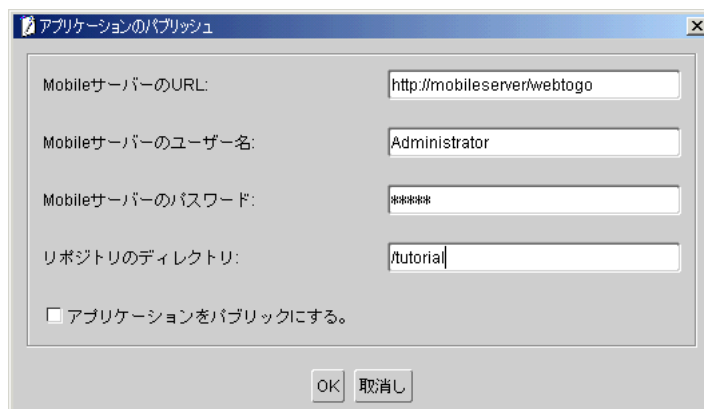
「ファイルの作成」オプションを使用すると、アプリケーション・コンポーネントを .jar ファイルにパッケージ化したり、データベース・オブジェクトを作成する SQL スクリプトを生成できます。アプリケーション・コンポーネントを .jar ファイルにパッケージ化するには、「ファイルの作成」をクリックしてから「アプリケーションを JAR ファイルに含めます。」をクリックします。 .jar ファイルの位置を指定できます。SQL スクリプトを生成するには、「ファイルの作成」をクリックしてから「データベース・オブジェクト用に SQL スクリプトを生成します。」をクリックします。生成されたスクリプトは、アプリケーションのローカル・ルート・ディレクトリの下に SQL サブディレクトリ内に入れられます。SQL スクリプトは、サーバー側の表、シーケンスおよび DDL に関して指定した情報を使用します。この SQL スクリプトをデータベースに対して実行して、これらのデータベース・オブジェクトを作成できます。



5.3.13.3 アプリケーションのパブリッシュ

「現行のアプリケーションのパブリッシュ」オプションを使用すると、パッケージ・ウィザードで作成し定義したアプリケーションをパブリッシュできます。Web-to-Go アプリケーションをパブリッシュするには、「現行のアプリケーションのパブリッシュ」ボタンをクリックしてから「OK」をクリックします。「アプリケーションのパブリッシュ」ウィンドウが表示されます。

図 5-17 「アプリケーションのパブリッシュ」ダイアログ・ボックス



「アプリケーションのパブリッシュ」ダイアログ・ボックスの指定のフィールドに必要な情報を入力します。

フィールド	値
Mobile サーバーの URL	http://server
Mobile サーバーのユーザー名	Administrator
Mobile サーバーのパスワード	admin
リポジトリのディレクトリ	/tutorial
アプリケーションをパブリックにする。	「アプリケーションをパブリックにする。」チェックボックスは選択しないでください。

注意： アプリケーションを Mobile サーバーにパブリッシュするには、パブリッシュ権限が必要です。Mobile サーバー管理者は Mobile サーバー・コントロール・センターを使用して権限を割り当てます。

5.3.13.4 パッケージ・ウィザードの再起動

「ウィザードを再起動します。」オプションを使用すると、パッケージ・ウィザードを再起動できます。このオプションを使用すると、パッケージ・ウィザードの「ようこそ」パネルに戻ります。パッケージ・ウィザードを再起動するには、「ウィザードを再起動します。」をクリックしてから「OK」をクリックします。

5.3.14 アプリケーションの編集

パッケージ・ウィザードを起動して「既存のアプリケーションの編集」を選択すると、アプリケーションを編集できます。Web-to-Go の DTD ファイルに準拠する XML 文書を作成または変更すると、アプリケーションを手動で作成または編集できますが、アプリケーションの作成または変更にはパッケージ・ウィザードの使用が最適です。

5.4 スキーマ展開

既存のアプリケーションにスキーマ展開による変更を含めるには、アプリケーションを再パブリッシュする必要があります。1 つ以上のアプリケーション表を変更し、クライアントにこれらの変更を反映する場合、次の項の説明に従って、パッケージ・ウィザードを使用してアプリケーションを再パブリッシュする必要があります。

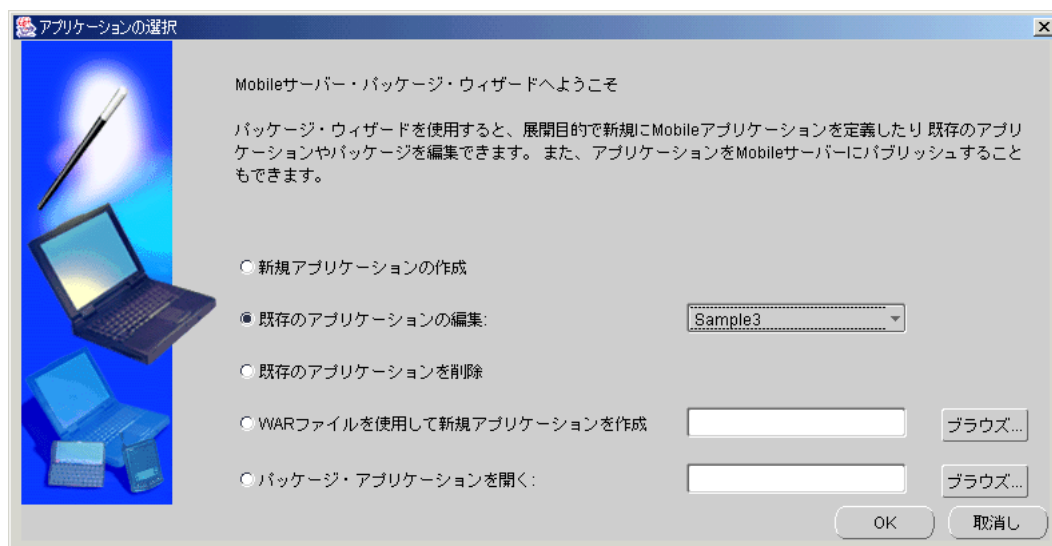
5.4.1 パッケージ・ウィザードの起動

パッケージ・ウィザードを起動するには、コマンド・プロンプトで次のように入力します。

```
wtgpack
```

パッケージ・ウィザードが表示され、デフォルトで図 5-18 に示すような「アプリケーションの選択」ダイアログ・ボックスが表示されます。「既存のアプリケーションの編集」オプションを選択します。

図 5-18 「アプリケーションの選択」ダイアログ・ボックス

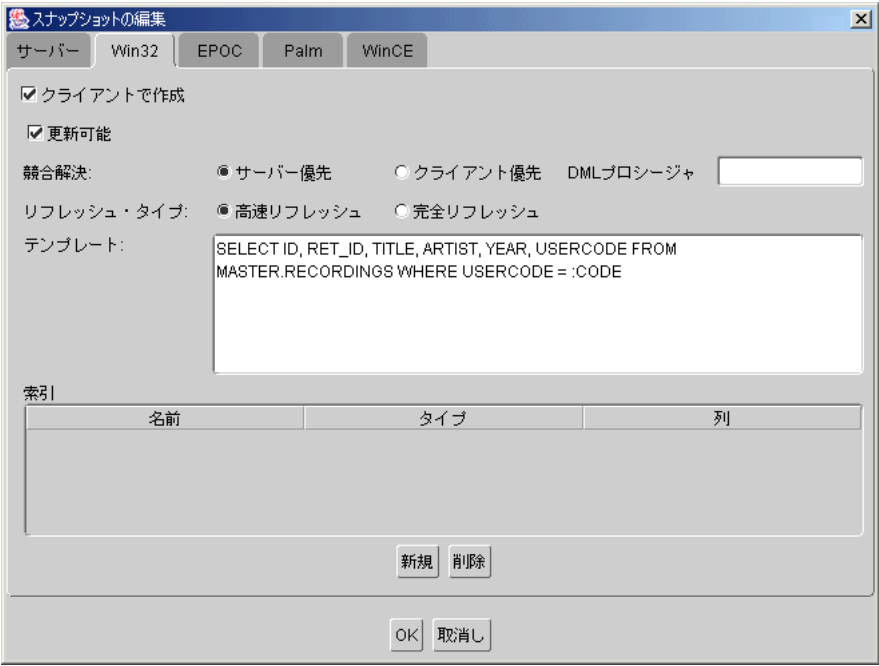


5.4.2 スナップショットの編集

ご使用のアプリケーションにアプリケーション・スキーマを含めるには、「スナップショット」タブでスナップショットの問合せを編集します。元のスナップショットの問合せに列名が含まれている場合、スナップショットの問合せを編集して、問合せに新しい列を追加する必要があります。スナップショットの問合せに列名が含まれていない場合（select * from emp など）は、問合せを編集する必要はありません。5.4.3 項「アプリケーションの完了」の手順に進んでください。

スナップショットを編集するには、「スナップショット」パネルからスナップショットを選択し、「編集」をクリックします。図 5-19 に示すように、「スナップショットの編集」ダイアログ・ボックスが表示されます。

図 5-19 「スナップショットの編集」 ダイアログ・ボックス (クライアント)



「スナップショットの編集」ダイアログ・ボックスの「テンプレート」フィールドで変更を入力して、スナップショットを編集します。「テンプレート」フィールドについては、次の表を参照してください。

フィールド	説明
テンプレート	名前付きの表のスナップショット・テンプレートを表示します。スナップショット・テンプレートは変更できます。管理者は、Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数のインスタンスをこのテンプレートに生成できます。

5.4.3 アプリケーションの完了

すべてのスナップショットの問合せを編集した後、パッケージ・ウィザードの終了画面に進みます。「アプリケーションの定義の完了」ダイアログ・ボックスが表示されます。[5.3.13 項「アプリケーションの完了」](#)の手順に従ってください。

アプリケーションのパブリッシュが終了すると、すべてのスキーマの変更はアプリケーションに取り込まれます。次に Mobile サーバーと同期する際、クライアントは、レプリケートされるスキーマの変更を取得します。

5.5 web.xml 形式のサポート

WAR ファイルのサポートで重要なのは、記述子ファイル web.xml の読み込み要件です。

5.5.1 web.xml ファイルの場所

アプリケーション実行中に読み込みを行うために、アプリケーション開発者は、記述子ファイル web.xml を特定のディレクトリに配置する必要があります。web.xml ファイルは、`<application_root_directory>/WEB-INF/web.xml` に配置する必要があります。たとえば、サンプル・ファイル sample3.war をパブリッシュすると、このサンプル・ファイルによって web.xml ファイルが sample3/WEB-INF/web.xml に配置されます。

5.5.2 web.xml でサポートされるタグ

パッケージ・ウィザードは、Java Servlet 2.3 仕様で定義されている web.xml タグのサブセットをサポートします。Java Servlet 2.3 仕様については、http://java.sun.com/dtd/web-app_2_3.dtd でサポートされるタグのほとんどは、サーブレット・ランタイム・エンジンで使用されます。一部のタグは、次の場合にのみサポートされます。

- パッケージ・ウィザードを使用して、WAR ファイルに基づく新規アプリケーションをパッケージする場合。
- コントロール・センターを使用して、WAR ファイルに基づくアプリケーションをパブリッシュする場合。

サポートされるタグについては、表 5-2 「サポートされる web.xml タグ」を参照してください。

表 5-2 サポートされる web.xml タグ

タグ	説明
<code><web-app></code>	サポート対象。開始タグ（必須）。
<code><icon></code>	サポート対象。新しいパッケージ・ウィザードのパッケージ・アプリケーションの場合、またはコントロール・センターを使用して WAR ファイルをパブリッシュする場合のみ。
<code><small-icon></code>	サポート対象。アイコンのセカンダリ選択肢。
<code></small-icon></code>	
<code><large-icon></code>	サポート対象。アイコンのプライマリ選択肢。
<code></large-icon></code>	
<code></icon></code>	サポート対象。

表 5-2 サポートされる web.xml タグ (続き)

タグ	説明
<display name> </display name>	サポート対象。新しいパッケージ・ウィザードのパッケージ・アプリケーションの場合、またはコントロール・センターを使用して WAR ファイルをパブリッシュする場合のみ。
<description> </description>	サポート対象。新しいパッケージ・ウィザードのパッケージ・アプリケーションの場合、またはコントロール・センターを使用して WAR ファイルをパブリッシュする場合のみ。
<distributable></distributable>	サポート対象外。タグは無視されますが、挿入しないでください。
<context-param> <param-name> <param-value> <description></description> </context-param>	サポート対象。ランタイム・エンジンで使用。 サポート対象。ランタイム・エンジンで使用。 サポート対象。ランタイム・エンジンで使用。 サポート対象。ランタイム・エンジンで使用。
<servlet> <icon> <small-icon></small-icon> <large-icon></large-icon> </icon>	サポート対象。ただし、一部のタグのみ。 <servlet> レベルではサポート対象外。 <servlet> レベルではサポート対象外。 <servlet> レベルではサポート対象外。 <servlet> レベルではサポート対象外。
<servlet-name></servlet-name> <display-name></display-name> <description></description>	サポート対象。ランタイム・エンジンで使用。 <servlet> レベルではサポート対象外。 <servlet> レベルではサポート対象外。
<servlet-class></servlet-class>	サポート対象。ランタイム・エンジンで使用。
<jsp-file></jsp-file>	<servlet> レベルではサポート対象外。
<init-param> <param-name> <param-value> <description></description> </init-param>	サポート対象。ランタイム・エンジンで使用。 サポート対象。ランタイム・エンジンで使用。 サポート対象。ランタイム・エンジンで使用。 サポート対象。ランタイム・エンジンで使用。

表 5-2 サポートされる web.xml タグ (続き)

タグ	説明
<load-on-startup>	サポート対象。ランタイム・エンジンで使用。
<security-role-ref>	サポート対象。ランタイム・エンジンで使用。
<description></description>	サポート対象。ランタイム・エンジンで使用。
<role-name></role-name>	サポート対象。ランタイム・エンジンで使用。
<role-link></role-link>	サポート対象。ランタイム・エンジンで使用。
</security-role-ref>	サポート対象。
</servlet>	サポート対象。
<servlet-mapping>	サポート対象。ランタイム・エンジンで使用。
<servlet-name></servlet-name>	サポート対象。ランタイム・エンジンで使用。
<url-pattern></url-pattern>	サポート対象。ランタイム・エンジンで使用。
</servlet-mapping>	サポート対象。ランタイム・エンジンで使用。
<mime-mapping>	サポート対象。ランタイム・エンジンで使用。
<extension></extension>	サポート対象。ランタイム・エンジンで使用。
<mime-type></mime-type>	サポート対象。ランタイム・エンジンで使用。
</mime-mapping>	サポート対象。
<welcome-file-list>	サポート対象。新しいパッケージ・ウィザードのパッケージ・アプリケーションの場合、またはコントロール・センターを使用して WAR ファイルをパブリッシュする場合のみ。
<welcome-file></welcome-file>	サポート対象。新しいパッケージ・ウィザードのパッケージ・アプリケーションの場合、またはコントロール・センターを使用して WAR ファイルをパブリッシュする場合のみ。
</welcome-file-list>	サポート対象。新しいパッケージ・ウィザードのパッケージ・アプリケーションの場合、またはコントロール・センターを使用して WAR ファイルをパブリッシュする場合のみ。
<error-page>	サポート対象。ランタイム・エンジンで使用。
<error-code></error-code>	サポート対象。ランタイム・エンジンで使用。
<exception-type></exception-type>	サポート対象。ランタイム・エンジンで使用。
<location></location>	サポート対象。ランタイム・エンジンで使用。

表 5-2 サポートされる web.xml タグ (続き)

タグ	説明
</error-page>	サポート対象。ランタイム・エンジンで使用。
</web-app>	サポート対象。終了タグ (必須)。

BC4J のチュートリアル

このチュートリアルでは、簡単なアプリケーションを使用して、Business Component for Java (BC4J) アプリケーションの作成、配布および使用方法を説明します。

このチュートリアルの構成は、次のとおりです。

- [6.1 項「概要」](#)
- [6.2 項「アプリケーションの開発」](#)
- [6.3 項「JSP アプリケーションのパッケージ化」](#)
- [6.4 項「コントロール・センターからの JSP アプリケーションのパブリッシュおよび構成」](#)
- [6.5 項「BC4J アプリケーションのテスト」](#)
- [6.6 項「Web-to-Go 用 Mobile クライアント上での BC4J アプリケーションの実行」](#)
- [6.7 項「サンプル・アプリケーションの配布」](#)

6.1 概要

Oracle BC4J (Business Components for Java) は、Oracle9i JDeveloper の統合開発環境 (IDE: Integrated Development Environment) の一部を構成し、再利用可能な Java コンポーネントを作成し管理するツールを Java 開発者に提供します。

BC4J は、高パフォーマンスのインターネット・アプリケーション (E-Commerce、B2B システムなど) に対応する、再利用可能なビジネス・コンポーネントを構築および配布する開発者を対象に、業界標準に準拠したサーバー・サイド Java および XML のフレームワークを提供します。BC4J を使用して作成するアプリケーションは、5 つの基本的なフレームワーク・コンポーネント (エンティティ・オブジェクト、アソシエーション、ビュー・オブジェクト、ビュー・リンクおよびアプリケーション・モジュール) で構成されます。これらのコンポーネントは、他のコンポーネントと相互に関連付けられるため、データベース表内にビューを作成できます。データは、必要に応じて組み合わせたり、フィルタ処理をしたり、ソートすることができます。

アプリケーション開発で BC4J を使用すると、データベース指向コンポーネントが自動的に生成されます。そのため、Web-to-Go 開発者は、ビジネス・アプリケーション開発時にデータベース関連コンポーネントの作成に時間をかけずに、ビジネス・ロジックの作成に集中することができます。

このチュートリアルで使用する BC4J のサンプル・アプリケーションの項目は、リレーショナル・データベースに格納されます。このサンプル・アプリケーションは、従業員の詳細データをメンテナンスします。

6.1.1 作業の準備

Java でビジネス・コンポーネントを作成する前に、開発用コンピュータが、次に指定されている要件を満たしていることを確認する必要があります。

6.1.1.1 開発用コンピュータの要件

表 6-1 に、開発用コンピュータの構成およびインストール要件を示します。

表 6-1 開発用コンピュータの要件

要件	説明
Windows NT、Windows 2000 および Windows XP でのユーザー・ログイン：	Windows NT、Windows 2000 および Windows XP ログイン・ユーザーには、開発用コンピュータの管理者権限が必要です。
インストール済の Java コンポーネント：	Java Development Kit 1.3.1 以上
インストール済の Oracle コンポーネント：	Mobile サーバーまたは Mobile Development Kit (Oracle9i Lite CD-ROM) Oracle8i リリース 8.1.7 以上 Oracle9i JDeveloper リリース 9.0.3

注意： BC4J チュートリアルは、Mobile Development Kit 内の **9iLite_BC4J_Tutorial.jar** という名前の JAR ファイルに収められています。このファイルは、<Oracle_Home>%mobile%sdk%wtgsrc%bc4jtutorial% ディレクトリにあります。この JAR ファイルを使用して、BC4J チュートリアルを Mobile サーバーにパブリッシュしてから、[6.7 項「サンプル・アプリケーションの配布」](#)に説明されている手順に従って、チュートリアルの残りの作業を進めてください。**9iLite_BC4J_Tutorial.jar** にパッケージされているものと同じアプリケーションを開発するには、[6.2 項「アプリケーションの開発」](#)～[6.6 項「Web-to-Go 用 Mobile クライアント上での BC4J アプリケーションの実行」](#)に説明されている手順に従い、[6.7 項「サンプル・アプリケーションの配布」](#)に進んでください。

6.2 アプリケーションの開発

この項では、Oracle9i Lite 用の BC4J アプリケーションの開発を、段階的に説明します。

BC4J アプリケーションを開発するには、次の作業を実行する必要があります。

1. データベース接続を作成します。
2. BC4J コンポーネントを作成します。
3. WTGJdbc 接続を使用できるように、BC4J コンポーネントを構成します。
4. BC4J コンポーネントをシンプル・アーカイブとして構築し配布します。
5. BC4J コンポーネントにアクセスするための、JSP アプリケーションを書き込みます。
6. JSP アプリケーションをシンプル・アーカイブとして配布します。
7. BC4J コンポーネントを Mobile サーバーに配布します。

6.2.1 データベース接続の作成

データベース接続を作成するには、次の手順を実行します。

1. Oracle9i JDeveloper のプロジェクト・ワークスペース内の「Connections」アイコンを右クリックし、「Connections」メニューをクリックします。Connection Manager ウィザードが表示されます。


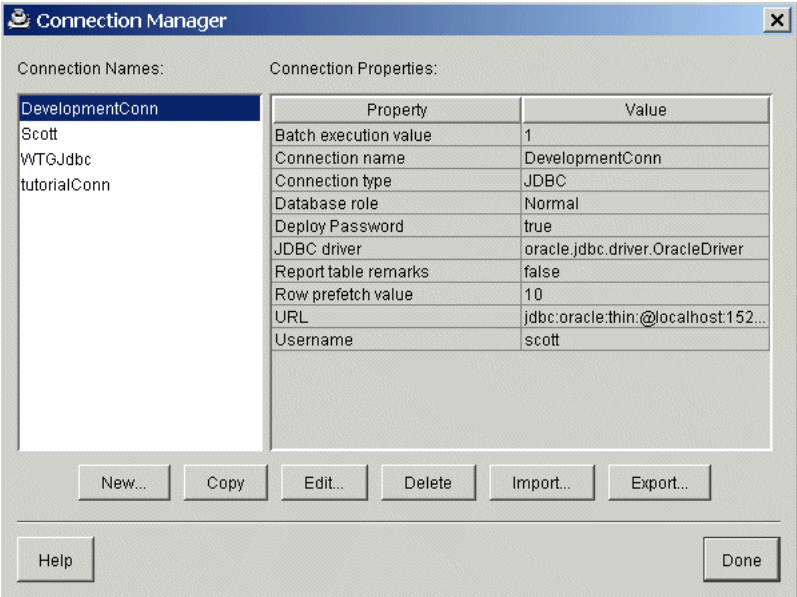
 **図 6-1** に Connection Manager ウィザードを示します。

図 6-1 Connection Manager ウィザード



2. 「New」をクリックします。「Connection」ダイアログ・ボックスが表示されます。
- 図 6-2 に「Connection」ダイアログ・ボックスを示します。このダイアログ・ボックスは、「tutorialConn」および「WTGJdbc」のデータベース接続を指定できます。

図 6-2 「Connection」 ダイアログ・ボックス

Connection

Connection Name:

Connection Type: ☒ JDBC ☐ IIOP

Please enter any applicable security information:

Username: Password:

Role: ☐ Include password in deployment archive

Select a JDBC Driver:

Select a connection method:

Please enter your database connection information:

TNS Service:

Host ID:

SID: Port:

Network Protocol: Other:

Row Prefetch: Batch Value: ☐ Report TABLE_REMARKS

The connection name is used to uniquely identify this connection.

- 「tutorialConn」と「WTGJdbc」という2つの接続を作成する必要があります。「tutorialConn」接続は、アプリケーションの開発およびテストを行うための `oracle.jdbc.driver.OracleDriver` を使用して、プライマリ Oracle データベースに接続します。「WTGJdbc」接続は、`oracle.lite.web.WTGJdbcDriver` を使用して、Oracle Lite データベースに接続します。シンプル・アーカイブとしてアプリケーションを配布する前に、「tutorialConn」から「WTGJdbc」への接続を切り替えてもかまいません。「WTGJdbc」接続は、アプリケーション配布時に使用され、独自のドライバを使用します。

「tutorialConn」のデータベース接続を作成するには、次の表に示されているとおり「Connection」ダイアログ・ボックスにデータを入力します。

「Connection」ダイアログ・ボックスで、「tutorialConn」に指定する必要がある値を、表 6-2 に示します。

表 6-2 「tutorialConn」 接続要件

フィールド	値
接続名 :	tutorialConn
ユーザー名 :	scott
パスワード :	tiger
SID :	データベース SID (Oracle システム識別子)

4. 「WTGJdbc」のデータベース接続を作成するには、次の表に示されているとおり「Connection」ダイアログ・ボックスにデータを入力します。
- 「Connection」ダイアログ・ボックスで、「WTGJdbc」に指定する必要がある値を、[表 6-3](#)に示します。

表 6-3 「WTGJdbc」 接続要件

フィールド	値
接続名 :	WTGJdbc
JDBC ドライバの選択 :	他の JDBC
クラス名 :	oracle.lite.web.WTGJdbcDriver
データソース URL:	jdbc:oracle:webtogo

注意：「Connection」ダイアログ・ボックスは、[表 6-2](#) および [表 6-3](#) に指定されている各フィールドに値を入力する必要があります。他の値は、すべてデフォルト値のまま残してください。

5. 「Test Connection」をクリックして、「tutorialConn」データベース接続をテストします。システムは、データベース接続が成功したことを知らせるメッセージを表示します。
6. 「OK」をクリックします。これで、データベース接続が作成されました。

6.2.2 BC4J コンポーネントの作成

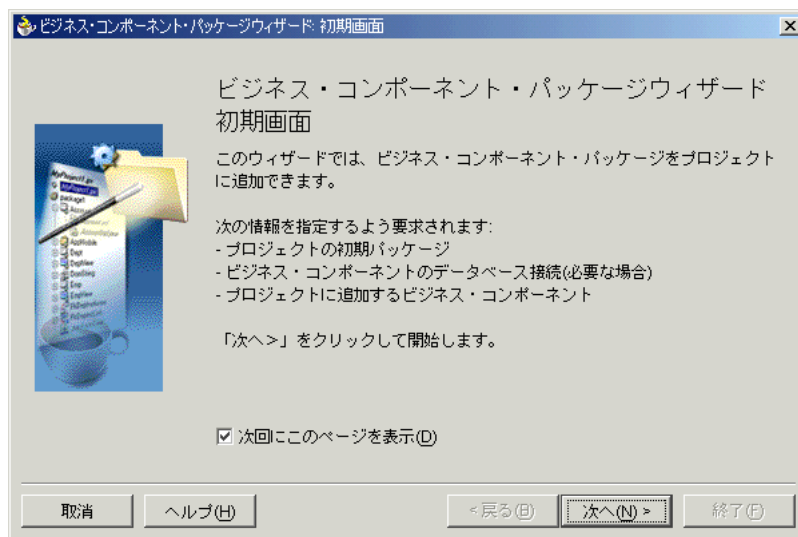
Oracle9i JDeveloper を使用すると、「tutorialapp」という名前の BC4J コンポーネントを作成できます。

「tutorialapp」という BC4J コンポーネントを作成するには、次の手順を実行します。

1. Oracle9i JDeveloper で、「File」メニューから「New」を選択します。表示される「新規」ダイアログ・ボックスでは、左側のパネルの「Projects」および右側のパネルの「空のオブジェクト」が、デフォルトとして事前に選択されています。「OK」をクリックすると、Oracle9i JDeveloper によって「Project.jpr」という空のプロジェクト名が新規作成されます。
2. 「Project.jpr」を「tutorialapp.jpr」という名前に変更します。この変更名の新しいプロジェクトが作成されます。
3. Oracle9i JDeveloper のワークスペースにある「tutorialapp.jpr」を右クリックします。「New Business Components Packages...」を選択します。「ビジネス・コンポーネント・パッケージウィザード：初期画面」ダイアログ・ボックスが表示されます。「次へ」をクリックします。

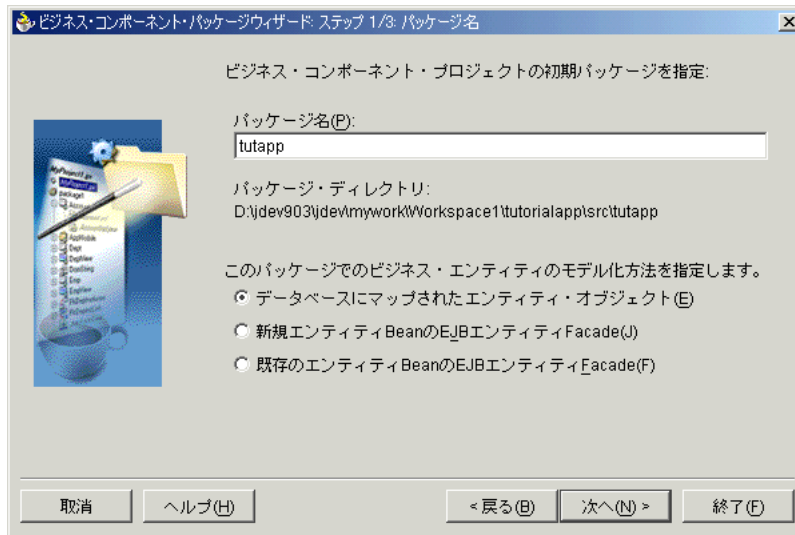
図 6-3 に「ビジネス・コンポーネント・パッケージウィザード：初期画面」ダイアログ・ボックスを示します。

図 6-3 「ビジネス・コンポーネント・パッケージウィザード：初期画面」ダイアログ・ボックス



4. 図 6-4 に示すとおり、「ビジネス・コンポーネント・パッケージウィザード: ステップ 1/3: パッケージ名」ダイアログ・ボックスが表示されます。「パッケージ名」フィールドに、`tutapp` と入力します。「次へ」をクリックします。

図 6-4 「ビジネス・コンポーネント・パッケージウィザード: ステップ 1/3: パッケージ名」



5. 図 6-5 に示すとおり、「ビジネス・コンポーネント・パッケージウィザード: ステップ 2/3: 接続」ダイアログ・ボックスが表示されます。表 6-4 に示されている値を選択し、「次へ」をクリックします。

図 6-5 「ビジネス・コンポーネント・パッケージウィザード: ステップ 2/3: 接続」

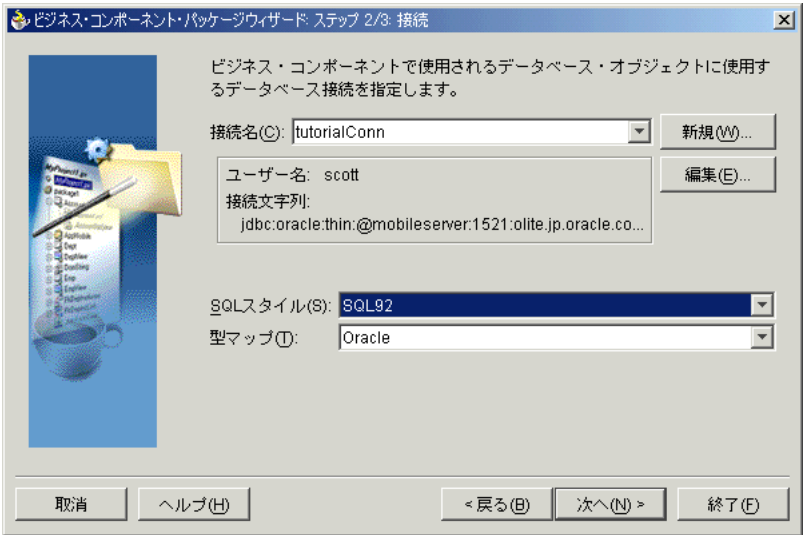


表 6-4 「ビジネス・コンポーネント・パッケージウィザード: ステップ 2/3: 接続」の値

フィールド	説明
接続名 :	tutorialConn
SQL スタイル :	SQL92
型マップ	Oracle

6. 「ビジネス・コンポーネント・パッケージウィザード: ステップ 3/3: ビジネス・コンポーネント」ダイアログ・ボックスで、図 6-6 に示されているように、左側のパネルに表示されているリストから「EMP」を選択し、「選択済」リストに移動します。「終了」をクリックします。

図 6-6 「ビジネス・コンポーネント・パッケージウィザード：ステップ 3/3: ビジネス・コンポーネント」 ダイアログ・ボックス



7. この段階で、Oracle9i JDeveloper によって「tutorialapp」という名前の BC4J コンポーネントが作成されます。

6.2.3 WTGJdbc 接続を使用するための BC4J コンポーネントの構成

WTGJdbc 接続が使用できるように BC4J コンポーネントを構成するには、次の手順を実行します。

1. 「TutorialappModule」を右クリックして、「Configurations...」オプションをダブルクリックします。「Configuration Manager」が表示されます。
2. 「Oracle Business Component Configuration」ダイアログ・ボックスで「Edit」をクリックします。「WTGJdbc」を JDBC 接続として選択します。
3. 「OK」をクリックします。これで、WTGJdbc 接続が使用できるように BC4J コンポーネントが構成されました。

6.2.4 シンプル・アーカイブとしての BC4J コンポーネントの構築および配布

シンプル・アーカイブとして BC4J コンポーネントを構築し配布するには、次の手順を実行します。

1. tutorialapp.jpr ファイルを右クリックし、「Create Business Components Deployment Profiles」オプションを選択します。配布ウィザードが表示されます。
2. 表示されているリストから「Simple Archive Files」オプションを選択し、「Selected」リストに移動します。
3. 「Selected」プラットフォームの「Simple Archive Files」セクションの下で、「Deploy」チェックボックスを選択します。
4. 「OK」をクリックします。システムによって、「tutorialapp.jar」と「tutorialappCommonCS.jar」という名前の2つの jar ファイルが作成されます。

6.2.5 BC4J コンポーネントにアクセスするための JSP アプリケーションの作成

BC4J コンポーネントにアクセスする JSP アプリケーションを作成するには、次の手順を実行します。

1. Oracle9i JDeveloper で、「File」メニューの下にある「空のプロジェクト」オプションを選択します。システムによって、「MyProject.jpr」という名前の空のプロジェクトが自動的に作成されます。
2. 「File」メニューから「Rename...」オプションを選択し、「MyProject.jpr」を「tutorialclientapp.jpr」に変更します。
3. Oracle9i JDeveloper のワークスペースにある tutorialclientapp.jpr ファイルをクリックします。「File」メニューをクリックし、「New」を選択します。「Web Tier」オプションをクリックし、「ビジネス・コンポーネント JSP アプリケーション」を選択します。「OK」をクリックすると、「ビジネス・コンポーネント JSP アプリケーション・ウィザード」が表示されます。
4. 「New」をクリックします。「ビジネス・コンポーネント JSP アプリケーション・ウィザード」が表示されます。「Next」をクリックします。ウィザードには、「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」ダイアログ・ボックスが表示されます。
5. 「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」ダイアログ・ボックスの「新規作成」ボタンをクリックします。「BC4J クライアント・データ・モデル定義ウィザード」が表示されます。
6. 「次へ」をクリックします。「BC4J クライアント・データ・モデル定義ウィザード - ステップ 1/2: 定義」ダイアログ・ボックス (図 6-7、図 6-8、図 6-9 および図 6-10 を参照) が表示されます。

図 6-7 「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ようこそ」

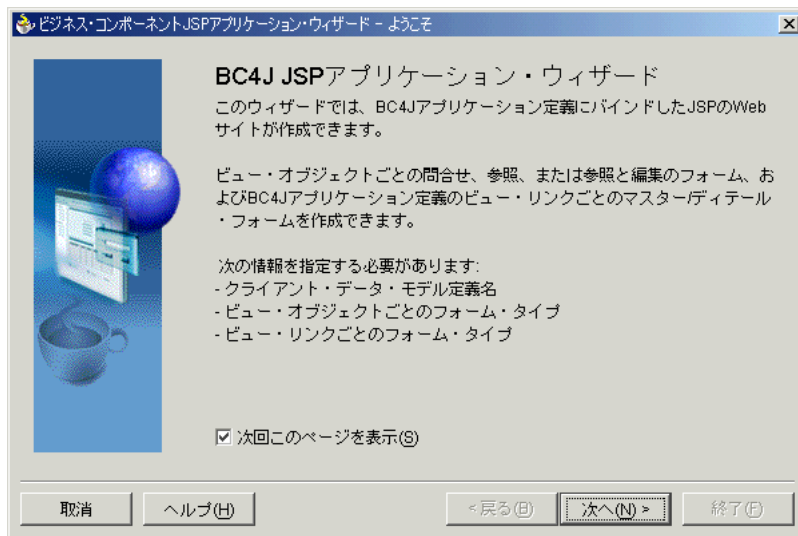


図 6-8 「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」

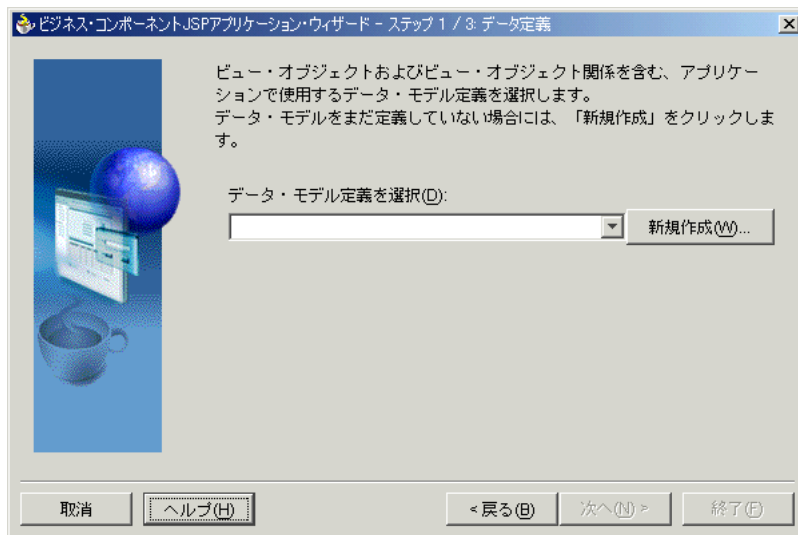


図 6-9 「BC4J クライアント・データ・モデル定義ウィザード - ようこそ」

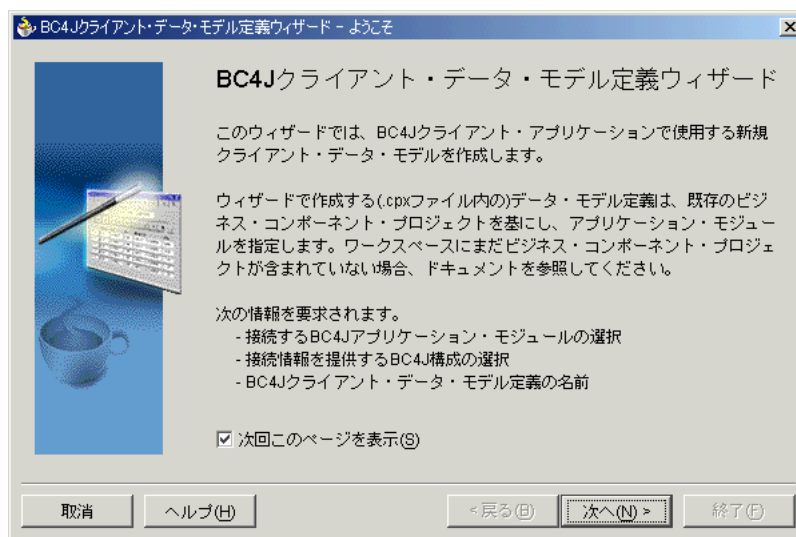
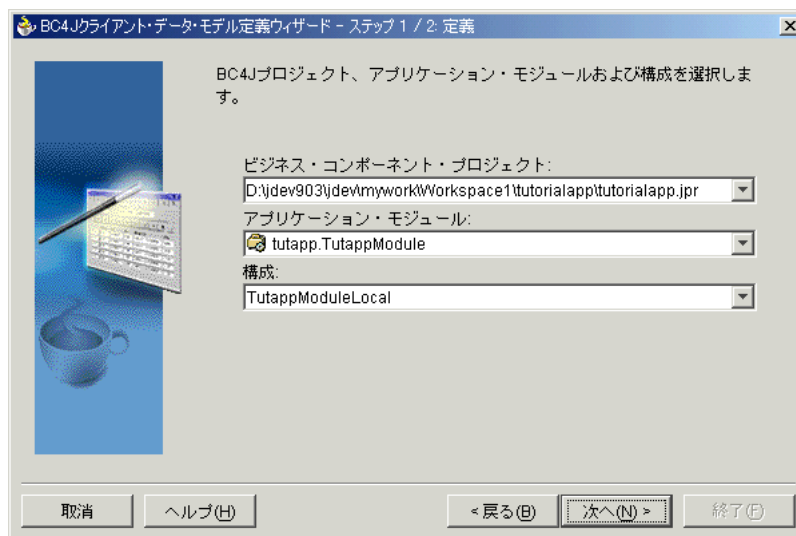
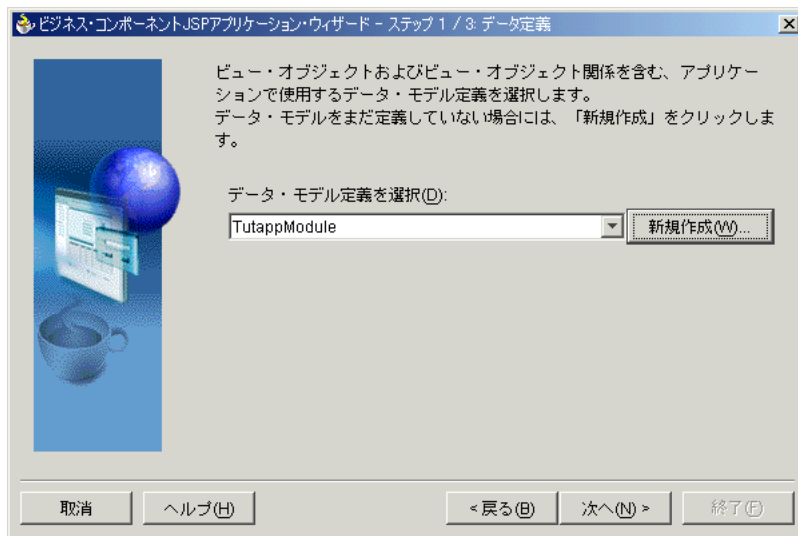


図 6-10 「BC4J クライアント・データ・モデル定義ウィザード - ステップ 1/2: 定義」



7. デフォルトとして選択されている値を確認し、「次へ」をクリックします。「BC4J Client Data Model Definition Wizard - Step 2 of 2: Definition Name」ダイアログ・ボックス内に「TutappModule」がデフォルトの定義名として表示されます。「次へ」をクリックします。
8. 「終了」をクリックします。「ビジネス・コンポーネント JSP アプリケーション・ウィザード」ダイアログ・ボックスが表示されます。「次へ」をクリックします。
9. 図 6-11 に示すとおり、「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」ダイアログ・ボックスで、「TutappModule」をデータ・モデル定義として選択します。「次へ」をクリックします。

図 6-11 「ビジネス・コンポーネント JSP アプリケーション・ウィザード - ステップ 1/3: データ定義」



10. 次に表示される「Business Components JSP Application Wizard - Step 2 of 3: View Object Forms」および「Business Components JSP Application Wizard - Step 3 of 3: View Link Form」の2つのダイアログ・ボックスで、デフォルトの選択を承認し、「次へ」をクリックします。表示される「Summary」ウィンドウで、「終了」をクリックします。

6.2.6 シンプル・アーカイブとしての JSP アプリケーションの配布

シンプル・アーカイブとして JSP アプリケーションを配布するには、次の手順を実行します。

1. Oracle9i JDeveloper で、**tutorialclientapp.jpr** ファイルをクリックしてから、**tutappclient_jpr_war.deploy** ファイルを選択します。
2. 「Deploy WAR file」を選択します。**tutappclient_jpr_war.war** ファイルが作成されます。配布場所を取得するには、Oracle9i JDeveloper の「Deployment Log」テキスト領域をチェックしてください。

6.3 JSP アプリケーションのパッケージ化

JSP アプリケーションをパッケージするには、次の手順を実行します。

1. 次の場所の下に、「bc4jtutapp」というサブディレクトリを作成します。

```
<Oracle_Home>\%Mobile%\Sdk\wtg\jdk\root
```

2. **tutappclient_jpr_war.war** ファイルを解凍して、bc4jtutapp ディレクトリに格納します。
3. すべての JSP ファイルを編集して、

```
;charset=windows-1252
```

という文字列を、次の文字列から削除します。

```
<%@page language="Java"errorpage="errorpage.jsp"
Content-Type="text/html; charset=windows-1252"%>
```

4. **Emp.xml** を編集して、

```
DBObjectName="SCOTT.EMP"
```

という文字列を、次のように変更します。

```
DBObjectName="EMP"
```

5. **EmpView.xml** を編集して、

```
LIST="SCOTT.EMP"
```

という文字列を、次のように変更します。

```
LIST="EMP Emp"
```

6. 「Command Prompt」ウィンドウを使用して、パッケージ・ウィザードを実行し、表 6-5 に示されている入力内容を指定します。

表 6-5 パッケージ・ウィザードの入力内容

画面	入力	詳細
プラットフォーム	Web-to-Go	なし
アプリケーション	アプリケーション名	BC4J 9iLite Tutorial Application
アプリケーション	仮想パス	/bc4jtutorial
アプリケーション	説明	BC4J 9iLite Tutorial Application
アプリケーション	アプリケーションのクラスパス	なし
アプリケーション	デフォルトのページ	main.html
アプリケーション	ローカル・アプリケーションのディレクトリ	<Oracle_Home>%Mobile%Sdk%wtgsdk%root%bc4jtutapp
ファイル	パッケージ・ウィザードは、すべてのファイルを、ローカル・アプリケーション・ディレクトリの下にロードします。	なし

表 6-6 に、パッケージ・ウィザードで、デフォルトで作成されるサーブレット名およびそのクラスが表示されます。

表 6-6 サーブレット名およびサーブレットのクラス

画面	サーブレット名	サーブレットのクラス
サーブレット	EMDServlet	oracle.jbo.server.emd.EMDServlet
サーブレット	ImageServlet	oracle.cabo.image.servlet.ImageServlet
サーブレット	TecateServlet	oracle.cabo.image.servlet.TecateServlet
サーブレット	BajaServlet	oracle.cabo.servlet.BajaServlet
サーブレット	OrdPlayMediaServlet	oracle.ord.html.OrdPlayMediaServlet

表 6-7 に、パッケージ・ウィザードで指定する必要がある、サーバー側およびクライアント側のデータベース値を示します。

表 6-7 データベース値

画面	入力	詳細
データベース	サーバー側データベースのユーザー名	scott
データベース	接続数	0
データベース	共有接続	このチェックボックスは選択しないでください。
データベース	クライアント側のデータベース名	クライアント・データベース

- 「スナップショット」セクションの下の「インポート」をクリックします。「データベースの接続」ダイアログ・ボックスで次の値を指定すると、Oracle データベースに接続できます。

表 6-8 に、「データベースの接続」ダイアログ・ボックスで指定する必要がある値を示します。

表 6-8 「データベースの接続」ウィンドウの説明

フィールド	説明
ユーザー名 :	scott
パスワード :	tiger
データベースの URL:	jdbc:oracle:oci8:@webtogo.world

- データベース接続値を指定した後、表のリストから「Emp」を選択します。
- 「編集」をクリックし、比率を「0」から「1」に変更します。
- 「ロール」、「シーケンス」、「DDL」および「レジストリ」フィールドは、デフォルト値のまま残します。
- アプリケーションをパッケージして、JAR ファイルに格納します。

6.4 コントロール・センターからの JSP アプリケーションのパブリッシュおよび構成

コントロール・センターから JSP アプリケーションを構成するには、次の手順を実行します。

1. 「Command Prompt」 ウィンドウを使用して、webtogo-d0 と入力し、Mobile サーバーを起動します。
2. 次の URL を使用して、ローカル・ホストを参照します。
`http://localhost`
3. 管理者のユーザー名およびパスワードを使用して、Mobile サーバーにログインします。
4. 「コントロール・センター」 をクリックします。
5. 「アプリケーション」 タブをクリックしてから、作成した JAR ファイルをパブリッシュします。
6. 「ユーザー」 タブをクリックし、「tutorial」という新規ユーザーを作成します。tutorial をパスワードとして入力します。新規ユーザー用の権限として「ユーザー」を割り当てます。
7. 「アプリケーション」 タブをクリックします。
8. 「BC4J 9iLite Tutorial Application」 リンクをクリックします。
9. 「プロパティ」 をクリックします。
10. 「Database Connectivity」 セクションの下に「データベースのパスワード」フィールドに、tiger と入力します。
11. 「アクセス」 をクリックして、BC4J 9iLite チュートリアル・アプリケーションへのアクセス権をユーザーに付与します。

6.5 BC4J アプリケーションのテスト

BC4J アプリケーションをテストするには、「tutorial」ユーザーとして Mobile サーバーにログインします。「tutorial」ワークスペースにある「BC4J 9iLite Tutorial Application」を、ダブルクリックします。「Business Components JSP Application」ウィンドウが表示されます。「EmpView」をクリックし、従業員表レコード全体を参照します。

6.6 Web-to-Go 用 Mobile クライアント上での BC4J アプリケーションの実行

Web-to-Go 用 Mobile クライアント上で BC4J アプリケーションを実行するには、次の手順を実行します。

1. 次の URL を使用して、データベース・サーバーの IP アドレス設定を確認します。
`http://server_ip_address/setup`
2. BC4J をサポートする Web-to-Go 用 Mobile クライアントをダウンロードし、インストールします。
3. 次の URL を使用して、クライアント・マシンのローカル・ホストを確認します。
`http://localhost` in client machine
4. 「tutorial」をユーザー名およびパスワードとして使用して、クライアント・マシンにログインします。
5. クライアント・マシンがサーバーのアプリケーションおよびデータを同期した後、「BC4J 9iLite Tutorial Application」リンクをクリックして、クライアント・マシン上のアプリケーションをテストします。

6.7 サンプル・アプリケーションの配布

サンプル・アプリケーションを配布するには、次の手順を実行します。

1. システム・ユーザーとして、データベースにログインします。SCOTT スキーマが存在しない場合は、bc4j.sql スクリプトを実行します。
2. <Oracle_Home>%Mobile%Sdk%wtgSDK%src%bc4jtutorial> にある **9iLite_BC4J_Tutorial.jar** ファイルを、Mobile サーバーにパブリッシュします。この場合、コントロール・センターを使用し、次の仮想パスを入力します。
`/bc4jtutorial`
3. 「コントロール・センター」をクリックし、「アプリケーション」タブを選択します。
4. 「9iLite BC4J アプリケーション」をクリックします。
5. 「プロパティ」リンクを選択し、データベース・パスワードとして「tiger」と入力します。「保存」をクリックします。
6. 「tutorial」ユーザーが存在しない場合は、「ユーザー」権限を持つ「tutorial」というユーザーを作成します。
7. 「tutorial」ユーザーに、9iLite BC4J アプリケーションへのアクセス権を付与します。

8. 次の URL を使用して、BC4J をサポートするクライアント・マシンを参照します。
`http://servername:port/webtogo/setup`
9. Web-to-Go 用 Mobile クライアントをダウンロードし、インストールします。
10. Web-to-Go クライアントが起動していない場合は、起動します。
11. 「tutorial」をユーザー名とパスワードとして使用して、Web-to-Go クライアントにログインします。
12. 同期プロセスが完了すると、「9iLite BC4J Application」リンクが表示されます。
13. 「9iLite BC4J Application」リンクをクリックして BC4J チュートリアル・アプリケーションにアクセスします。

Web-to-Go サンプル・アプリケーション

この付録には、Web-to-Go のサンプル・アプリケーションが含まれています。内容は次のとおりです。

- [A.1 項「概要」](#)
- [A.2 項「Sample 1 - Hello World」](#)
- [A.3 項「Sample 3 - Recording Tracker」](#)
- [A.4 項「Sample 4 - Hello Applet」](#)
- [A.5 項「Sample 6 - Image Gallery」](#)
- [A.6 項「Sample 7 - Employee Data Applet」](#)

A.1 概要

Web-to-Go には 5 種類のサンプル・プログラムが含まれており、Mobile Development Kit (Web-to-Go 用) または Mobile サーバーとともに自動的にインストールされます。

A.1.1 Mobile サーバー

デモは、バッチ・ファイル **instdemo.bat** を実行してインストールできます。バッチ・ファイルは、次の場所にあります。

```
<Oracle_Home>%Mobile%Server%samples
```

コマンド構文は、次のとおりです。

```
instdemo.bat [SYSTEM_password] [repository_owner] [repository_password]
```

たとえば、次のとおりです。

```
instdemo manager mobileadmin manager
```

A.1.2 Mobile Development Kit (Web-to-Go 用)

デモは、バッチ・ファイル **sdkdemos.bat** を実行してインストールできます。バッチ・ファイルは、次の場所にあります。

```
<Oracle_Home>%Mobile%sdk%wtg%sdk%src%sdkdemos.bat
```

A.1.3 Mobile Development Kit (Web-to-Go 用) からのサンプル・プログラムのアクセス

Mobile Development Kit (Web-to-Go 用) では、ブラウザから次の URL にアクセスして、サンプル・プログラムにアクセスできます。

```
http://server:7070/
```

ブラウザには、別個のサンプル・プログラムごとに複数のアイコンが表示されます。サンプル・プログラムを起動するには、該当するプログラムのアイコンをクリックします。

A.1.4 Mobile サーバーからのサンプル・プログラムのアクセス

Web-to-Go デモをインストールするとき、Mobile サーバーにより次のサンプル・ユーザーが自動的に作成されます。

ユーザー	パスワード
john	john
jack	jack
jane	jane

サンプル・ユーザーは、Mobile サーバーにログオンし、ワークスペース上のサンプル・アプリケーションのアイコンをどれかクリックしてサンプル・プログラムにアクセスできます。

A.2 Sample 1 - Hello World

Sample 1 の Hello World は、単純な HTML ページをブラウザに返すサーブレットです。HttpServlet の基本的なメソッドを示し、POST メソッドと GET メソッドの違いを示します。

A.2.1 ソース・コードの場所

ソース・コードの場所は、Mobile サーバーと Mobile Development Kit (Web-to-Go 用) のどちらをインストールしたかにより異なります。

インストール・タイプ	ソース・コードの場所
Mobile サーバー	<Oracle_Home>%Mobile%Server%samples%sample1%src
Mobile Development Kit (Web-to-Go 用)	Java ソースは、次の場所にあります。 <Oracle_Home>%Mobile%Sdk%wtg%src%sample1%servlets

A.2.2 アプリケーション・ファイル

Sample 1 には、次のアプリケーション・ファイルが含まれています。

ファイル	説明
HelloWorld.java	HelloWorld サブレットのソース・コードです。

A.3 Sample 3 - Recording Tracker

Sample 3 の Recording Tracker は、サブレットを使用してレコード情報でデータベースをメンテナンスする方法を示します。このプログラムでは、データベースの検索とレコードの入力および追跡を実行できます。レコードは RECORDINGS 表に格納されますが、ユーザーがこの表にアクセスするときに表示されるのはユーザー自身のデータのみです。

A.3.1 Sample 3 の使用方法

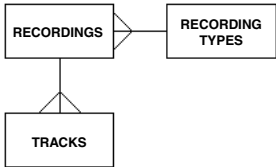
オフラインになると、Web-to-Go はデータのコピーを保持するスナップショットをローカル・クライアント上に自動的に作成します。スナップショット定義に副問合せを追加すると、スナップショットにレプリケートする行を制御できます。これにより、Web-to-Go は各ユーザー用の特定の行をローカル・クライアントに同期できます。Sample 3 では、John と Jack は同じデータにアクセスできます。Jane は専用のデータを表示でき、他のユーザーはこのデータにはアクセスできません。スナップショットの副問合せの設定の詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

A.3.2 Sample 3 のデータベース表

Sample 3 には、次のデータベース表が含まれています。

- RECORDINGS
- RECORDING TYPES
- TRACKS

これらのデータベース表は、次のエンティティ関連図に表示されます。



A.3.3 Sample 3 のサーブレット

Sample 3 には、6 種類の Java サーブレットが含まれています。これらのサーブレットは、HTML を生成する 2 つの方法を示しています。DisplayRecord サーブレットは、oracle.html パッケージを使用して HTML ファイル全体を生成します。DisplayMasterDetail、ListSearchResults および SimpleList のサーブレットは、ベース・クラス oracle.lite.web.html.TemplateParser と静的 HTML テンプレートを使用して HTML を生成します。いずれの場合も、データは DBTable クラスを使用して HTML 形式で表示されます。データ変更は、oracle.lite.web.html パッケージの一部である汎用サーブレット ProcessForm によって処理されます。DeleteDetail および DeleteMasterDetail は、クラス oracle.lite.web.html.DeleteRecords を継承します。これらのサーブレットは、要求を実行した後、ブラウザを別の URL にリダイレクトします。

A.3.4 Sample 3 のリソース・バンドル

Recording Tracker プログラムには、ListResourceBundle クラスを使用してロケール固有文字列のリソースを管理する方法も示されています。リソース・バンドル内のテキスト文字列をすべて切り離すことにより、別の言語に簡単に翻訳できるプログラムや、より多くの言語サポートを追加できるプログラムを作成できます。詳細は、SampleResources.java を参照してください。

A.3.5 ソース・コードの場所

ソース・コードの場所は、Mobile サーバーと Mobile Development Kit (Web-to-Go 用) のどちらをインストールしたかにより異なります。

インストール・タイプ	ソース・コードの場所
Mobile サーバー	<Oracle_Home>%Mobile%Server%samples%sample3%src
Mobile Development Kit (Web-to-Go 用)	Java ソースは、次の場所にあります。 <Oracle_Home>%Mobile%Sdk%wtgtsdk%src%sample3%

A.3.6 アプリケーション・ファイル

Sample 3 には、次のアプリケーション・ファイルが含まれています。

ファイル	説明
EnterSearchCriteria.html	静的 HTML ファイルです。
sample3.gif	Web-to-Go のワークスペースに表示される Sample 3 のアイコンです。
sample3.html	アプリケーションの開始ページです。

ファイル	説明
sample3.sql	Sample 3 のデータベース・オブジェクトをインストールする SQL スクリプトです。
table.sql	Sample 3 のデータベース表を作成する SQL スクリプトです。
insert.sql	Sample 3 のデータベース表にデータを移入する SQL スクリプトです。
drop.sql	Sample 3 のデータベース表を削除する SQL スクリプトです。
SampleProgram3.java	Sample 3 アプリケーションの静的定義を含むソース・コードです。
SampleResources.java	サーブレットにより使用される文字列リソースのソース・コードです。
DisplayMasterDetail.java	DisplayMasterDetail サーブレットのソース・コードです。
DisplayRecord.java	DisplayRecord サーブレットのソース・コードです。
SimpleList.java	SimpleList サーブレットのソース・コードです。
ListSearchResults.java	ListSearchResults サーブレットのソース・コードです。
DeleteDetail.java	DeleteDetail サーブレットのソース・コードです。
DeleteMasterDetail.java	DeleteMasterDetail サーブレットのソース・コードです。
DisplayMasterDetail.html	DisplayMasterDetail サーブレットにより使用される HTML テンプレートです。
ListSearchResults.html	ListSearchResults サーブレットにより使用される HTML テンプレートです。
SimpleList.html	SimpleList サーブレットにより使用される HTML テンプレートです。

A.4 Sample 4 - Hello Applet

Sample 4 の Hello Applet は、アプレットとサーブレットが相互に通信する方法を示します。Java アプレットが、Mobile サーバー上で実行されているサーブレットをコールします。このサーブレットは、文字列をアプレットに送信して応答し、この文字列をアプレットが表示します。

A.4.1 Sample 4 のサーブレット

Sample 4 には、サーブレットが 2 つ含まれています。AppServlet サーブレットは、ブラウザに対してアプレットを起動するように指示する HTML を生成します。この HTML には、Mobile サーバーのセッション情報を含むアプレット・パラメータが含まれています。HelloServlet は、アプレットとサーブレットの通信の一環としてアプレットによりコールされます。

A.4.2 ソース・コードの場所

ソース・コードの場所は、Mobile サーバーと Mobile Development Kit (Web-to-Go 用) のどちらをインストールしたかにより異なります。

インストール・タイプ	ソース・コードの場所
Mobile サーバー	<Oracle_Home>%Mobile%Server%samples%sample4%src
Mobile Development Kit (Web-to-Go 用)	Java ソースは、次の場所にあります。 <Oracle_Home>%Mobile%Sdk%wtgSDK%src%sample4%servlets

A.4.3 アプリケーション・ファイル

Sample 4 には、次のアプリケーション・ファイルが含まれています。

ファイル	説明
Sample4.gif	ワークスペースに表示される Sample 4 のアイコンです。
Sample4.html	アプリケーションの開始ページです。
HelloApplet.java	アプレットの Java ソース・コードです。
AppServlet.java	AppServlet サーブレットの Java ソース・コードです。
HelloServlet.java	HelloServlet サーブレットの Java ソース・コードです。

A.5 Sample 6 - Image Gallery

Sample 6 の Image Gallery は、LONG データ型を使用しないでバイナリ・データをデータベースに格納する方法を示します。サンプル・プログラムが Mobile サーバーにイメージをアップロードするとき、イメージは 255 バイトのチャンクに分割されます。この結果、イメージは RAW データ型の列に格納できます。

A.5.1 ソース・コードの場所

ソース・コードの場所は、Mobile サーバーと Mobile Development Kit（Web-to-Go 用）のどちらをインストールしたかにより異なります。

インストール・タイプ	ソース・コードの場所
Mobile サーバー	<Oracle_Home>%Mobile%Server%samples%sample6%src
Mobile Development Kit（Web-to-Go 用）	Java ソースは、次の場所にあります。 <Oracle_Home>%Mobile%Sdk%wtgSDK%src%sample6%

A.5.2 アプリケーション・ファイル

Sample 6 には、次のアプリケーション・ファイルが含まれています。

ファイル	説明
sample6.gif	Web-to-Go のワークスペースで使用されるアプリケーション用アイコンです。
loadImage.html	イメージをアップロードする HTML フォームです。
DeleteImage.java	DeleteImage サブプレットのソース・コードです。
GetImage.java	GetImage サブプレットのソース・コードです。
Upload.java	Upload サブプレットのソース・コードです。
ImageList.java	ImageList サブプレットのソース・コードです。
ViewImage.java	ViewImage サブプレットのソース・コードです。
RawImage.java	RawImage サブプレットのソース・コードです。
ImageList.html	ImageList サブプレットにより使用される HTML テンプレートです。
ViewImage.html	ViewList サブプレットにより使用される HTML テンプレートです。
sample6.sql	Sample 6 のデータベース・オブジェクトをインストールする SQL スクリプトです。
table.sql	Sample 6 のデータベース表を作成する SQL スクリプトです。
drop.sql	Sample 6 のデータベース表を削除する SQL スクリプトです。

A.6 Sample 7 - Employee Data Applet

Sample 7 の Employee Data Applet は、アプレット内で JDBC を使用方法を示します。アプレットは、`oracle.lite.web.applet.AppletProxy` クラスを使用して、データベースに接続します。このクラスは、ユーザーの接続モードに応じて、適切なデータベースに対するデータベース接続を返します。オンライン・モードの場合は、Oracle データベースに対する接続を返します。オフライン・モードの場合または Mobile Development Kit (Web-to-Go 用) を使用している場合は、Oracle Lite に対する接続を返します。

A.6.1 ソース・コードの場所

ソース・コードの場所は、Mobile サーバーと Mobile Development Kit (Web-to-Go 用) のどちらをインストールしたかにより異なります。

インストール・タイプ	ソース・コードの場所
Mobile サーバー	<Oracle_Home>%Mobile%Server%samples%sample7%src
Mobile Development Kit (Web-to-Go 用)	<Oracle_Home>%Mobile%Sdk%wtgSDK%src%sample7%

A.6.2 アプリケーション・ファイル

Sample 7 には、次のアプリケーション・ファイルが含まれています。

ファイル	説明
sample7.gif	ワークスペースに表示されるアプリケーション・アイコンです。
sample7.ahtml	アプリケーションの開始ページです。
AppApplet.java	アプレットの Java ソース・コードです。
ErrorDialog.java	エラー・ダイアログ用の Java ソース・コードです。
sample7.sql	Sample 7 のデータベース・オブジェクトをインストールする SQL スクリプトです。
table.sql	Sample 7 のデータベース表を作成する SQL スクリプトです。
insert.sql	Sample 7 のデータベース表にデータを移入する SQL スクリプトです。
drop.sql	古い表を削除するための SQL スクリプトです。

Web-to-Go Java パッケージ

この付録には、Web-to-Go の Java パッケージのサンプルが含まれています。内容は次のとおりです。

- B.1 項「[oracle.html](#) パッケージの使用方法」
- B.2 項「[oracle.lite.web.html](#) パッケージの使用方法」

B.1 oracle.html パッケージの使用方法

Java コード内で HTML を生成するには、oracle.html パッケージにあるクラスを使用します。これらのクラスを使用すると、HTML 文字列を生成し、文字列を出力ストリームに書き込むことができます。

B.1.1 HtmlPage オブジェクトの作成

Oracle.html パッケージにあるクラスを使用して HTML を生成するには、HTML ページの HEAD セクションと BODY セクションを含む HtmlPage オブジェクトを作成します。ページの <TITLE> は、HtmlPage コンストラクタ内に指定するか、後で HtmlHead オブジェクトを作成するときに指定します。

```
HtmlPage htmlpage = new HtmlPage();
```

B.1.2 <HEAD> 領域へのタグの追加

<HEAD> 領域にタグを追加するには、HtmlHead オブジェクトが必要です。HtmlHead オブジェクトを取得するには、次のいずれかを実行します。

- HtmlHead オブジェクトを作成します。

```
HtmlHead htmlhead = new HtmlHead();  
htmlhead.setBase("http://www.newbase.com");
```
- HtmlPage オブジェクトからオブジェクトを取得します。

```
htmlpage.getHead().setBase("http://www.newbase.com");
```

B.1.3 <BODY> タグへのコンテンツの追加

<BODY> タグにコンテンツを追加するには、ページに HtmlBody オブジェクトが必要です。HtmlBody オブジェクトを取得するには、次のいずれかを実行します。

- HtmlBody オブジェクトを作成します。

```
HtmlBody htmlbody = new HtmlBody();
```
- HtmlPage オブジェクトからオブジェクトを取得します。

```
htmlbody = htmlpage.getBody();
```

B.1.4 HTML ページの <BODY> セクションへのタグの追加

oracle.html パッケージ内のクラスを使用して、HTML ページの <BODY> セクションにタグを追加します。たとえば、次の行はページにヘッダーを追加します。

```
htmlbody.Heading(1, "The first heading");
```

HTML 項目と、対応する Java クラスのリストは、B.1.6 項「[HTML 要素の Java クラスへのマッピング](#)」を参照してください。

B.1.5 クラスを使用した HTML 生成の例

この項には、クラスを使用して HTML を生成する例が含まれています。

B.1.5.1 単純な書式化されていないテキスト

```
import oracle.html.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        // Set the content type for the response
        res.setContentType("text/html");
        // get the output stream
        ServletOutputStream out = res.getOutputStream();

        // Create an HtmlPage object
        HtmlPage hp = new HtmlPage("Hello World");

        // Add a string object ("Hello World!") to the page
        hp.getBody().addItem("Hello World!");

        // Print header info to output stream
        hp.print(out);
    }
}
```

このプログラムは次の HTML を生成します。

```
Content-type: text/html
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
</HEAD>
<BODY>
Hello World!</BODY>
</HTML>
```

最初の行は、ユーザー・エージェント（通常はブラウザ）に返される文書の HTTP 応答ヘッダーです。

この HTML 文書の内容は、「Hello World!」という簡単な文字列です。

B.1.5.2 ヘッダーとテキスト属性

Item 抽象クラスには、ほとんどのマークアップ・サポートが定義されています。Item 抽象クラスは、ターゲット・オブジェクトの <I> 属性を設定する `setItal()` メソッドや、ターゲット・オブジェクトの 属性を設定する `setEmphasis()` メソッドなどのメソッドを定義します。次の例では、Item クラスの様々なメソッドを使用しています。

```
import oracle.html.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        // Set the content type for the response
        res.setContentType("text/html");

        // get the output stream
        ServletOutputStream out = res.getOutputStream();

        // Create an HtmlPage object
        HtmlPage hp = new HtmlPage();

        // Get default HtmlBody object from HtmlPage object
        HtmlBody bd = hp.getBody();

        // Add a <H1> to the page
        bd.addItem(new SimpleItem("Welcome! Java programmers!").setHeading(1));
```



```

    // Print the text string "Hello World!" in different Heading formats
    for (int i=2; i<5; i++) {
        bd.addItem(new SimpleItem("Hello World!").setHeading(i));
    }

    // Print the string "Java is cool!" in bold
    bd.addItem(new SimpleItem("Java is cool!").setBold());
    bd.addItem(SimpleItem.LineBreak);

    // Print out the contents of this page
    hp.print(out);

}
}

```

次の出力が生成されます。

Content-type: text/html

```

<HTML>
<HEAD>
</HEAD>
<BODY>
<H1>Welcome!Java programmers!</H1>
<H2>Hello World!</H2>
<H3>Hello World!</H3>
<H4>Hello World!</H4>
<B>Java is cool!</B><BR>
</BODY>
</HTML>

```

B.1.5.3 フォーム

次のコードは、HTML フォーム・コードを生成します。

```

// Create a Form object
Form form = new Form("GET", "http://www.myhome.com/wrb/doit");

// Create a TextField object and add it to the form
form.addItem(new TextField("Name"));

// Create a Submit object and adds to form
form.addItem(new Submit("foo", "Submit!"));

// Add the form object to a HtmlBody object
body.addItem(form);

```

次の出力が生成されます。

```
<FORM METHOD="GET" ACTION="http://www.myhome.com/wrb/doit">
<INPUT TYPE=TEXT NAME="Name">
<INPUT TYPE=SUBMIT NAME="foo" VALUE="Submit!">
</FORM>
```

B.1.5.4 リスト

次のコードは、順序リストを生成します。

```
OrderedList orderedlist = new OrderedList(); // Create an OrderedList Object

// Add new items to the list
orderedlist.addItem(new SimpleItem("Item 1"));
orderedlist.addItem(new SimpleItem("Item 2"));

// Add the list object to the body
body.addItem(orderedlist);
```

次の出力が生成されます。

```
<OL>
<LI>Item 1
<LI>Item 2
</OL>
```

B.1.5.5 表

次のコードは、表を生成します。

```
DynamicTable tab = new DynamicTable(2); // create a two-column table

// create the rows and add them to the table; assume NUM_ROWS is an integer
TableRow rows[] = new TableRow[NUM_ROWS];
int cellcount=0;
for (int i=0; i< NUM_ROWS; i++) {
    // allocate TableRow
    rows[i] = new TableRow();

    // populate cells with data
    rows[i].
        addCell(new TableDataCell(Integer.toString(++cellcount)));
        addCell(new TableDataCell("foo"));

    // add the rows to the table
    tab.addRow(rows[i]);
}
```

```
// add the table to the body  
body.addItem(tab);
```

次の出力が生成されます。

```
<TABLE BORDER=1 FRAME=LHS RULES=ALL>  
<TR><TD>1</TD><TD>fooga</TD>  
<TR><TD>2</TD><TD>fooga</TD>  
<TR><TD>3</TD><TD>fooga</TD>  
</TABLE>
```

B.1.6 HTML 要素の Java クラスへのマッピング

次の表は、oracle.html パッケージ内の Java クラスに HTML 要素がどのようにマッピングされるかを示したものです。

B.1.6.1 HEAD 要素

HEAD 要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
<HEAD>	HtmlHead
<TITLE>	HtmlHead または HtmlPage
<STYLE>	Style
<SCRIPT>	Script
<ISINDEX>	
<BASE>	
<META>	MetaInfo
<LINK>	HeadLink

B.1.6.2 BODY 要素

BODY 要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
<BODY>	HtmlBody
<H1> から <H6>	Heading
<ADDRESS>	Address

HTML	Java クラス
<P>	Paragraph または SimpleItem
<A href>	Link
<A name>	Anchor
	Image
 (サーバー側のイメージ・マップ)	Image
 (クライアント側のイメージ・マップ)	ImageMap
<MAP>	ImageMapArea
<APPLET>	Applet
<PARAM>	Applet.addParam()
 	LineBreak
<PRE>	Preformat
<BLOCKQUOTE>	BlockQuote
<HR>	HorizontalRule
<DIV>	
<CENTER>	setAttr(ATTR_ALGN_CENTER)

B.1.6.3 リスト要素

リスト要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
	UnOrderedList
	OrderedList
<DL>	DefinitionList
<DIR>	DirectoryList
<MENU>	MenuList
	ListItem

HTML	Java クラス
<DT>	DefinitionList.addDef
<DD>	DefinitionList.addDef

B.1.6.4 表要素

表要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
<TABLE>	DynamicTable
<CAPTION>	DynamicTable.setCaption()
<TR>	TableRow
<TH>	TableHeaderCell
<TD>	TableDataCell

B.1.6.5 テキスト・レベル要素

テキスト・レベル要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
<BASEFONT>	BaseFont
	Font
	setAttr(ATTR_PHRASE_EMPHASIS) または setEmphasis()
	setAttr(ATTR_PHRASE_STRONG) または setStrongEmphasis()
<DFN>	setAttr(ATTR_PHRASE_DEFINITION) または setDefinition()
<CODE>	setAttr(ATTR_PHRASE_CODE) または setCode()
<SAMP>	setAttr(ATTR_PHRASE_SAMPLE) または setSample()
<KBD>	setAttr(ATTR_PHRASE_KEYBOARD) または setKeyboard()
<VAR>	setAttr(ATTR_PHRASE_VARIABLE) または setVariable()
<CITE>	setAttr(ATTR_PHRASE_CITATION) または setCite()
<TT>	setAttr(ATTR_FONT_TELETYPE) または setTeletype()
<I>	setAttr(ATTR_FONT_ITALIC) または setItal()

HTML	Java クラス
	setAttr(ATTR_FONT_BOLD) または setBold()
<U>	setAttr(ATTR_FONT_UNDERLINE) または setUnderline()
<STRIKE>	setAttr(ATTR_FONT_STRIKE) または setStrike()
<BIG>	setAttr(ATTR_FONT_BIG) または setFontBig()
<SMALL>	setAttr(ATTR_FONT_SMALL) または setFontSmall()
<SUB>	setAttr(ATTR_FONT_SUB) または setFontSubscript()
<SUP>	setAttr(ATTR_FONT_SUPER) または setFontSuperscript()

B.1.6.6 フォーム要素

フォーム要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
<FORM>	Form
<SELECT>	Select
<OPTION>	Option
<TEXTAREA>	TextArea
<INPUT type=checkbox	CheckBox
<INPUT type=password	PasswordField
<INPUT type=text	TextField
<INPUT type=radio	Radio
<INPUT type=submit	Submit
<INPUT type=reset	Reset
<INPUT type=hidden	Hidden
<INPUT type=file	ファイル
<INPUT type=image	Image

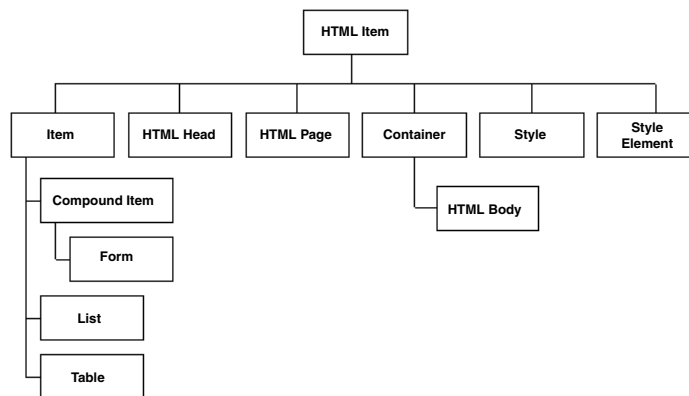
B.1.6.7 その他の要素

その他の要素と Java クラスのマッピングを次の表に示します。

HTML	Java クラス
<!-- ... -->	Comment
<FRAME>	Frame
<FRAMESET>	Frameset
<EMBED>	Embed
<OBJECT>	XObject
<LINK>	HeadLink

B.1.7 oracle.html パッケージのクラス階層

oracle.html パッケージ内の主なクラスを次に示します。



B.1.8 IHtmlItem インタフェース

IHtmlItem インタフェースは、HTML コンポーネントの基本的な要件と動作を定義します。このインタフェースのデフォルト実装は IHtmlItemImpl クラスです。このクラスは、実質的にこのパッケージ内にある他の全クラスのベース・クラスとして機能します。また、このパッケージ内の HTML コンポーネントを様々な形で処理できます。このインタフェースは、コンポーネントの内容を文字列オブジェクトに変換するメソッド (toString) と、オブジェクトの内容を HTML として任意の出力ストリーム・オブジェクトに書き込むメソッド (print()) も定義します。カスタム・オブジェクトには、このパッケージを利用できるように、すべてこのインタフェースを実装する必要があります。

B.1.9 Item 抽象クラス

oracle.html パッケージのもう 1 つの基本的なクラスは `Item` 抽象クラスです。このクラスは、マークアップ・コンポーネントとみなすことができるすべてのコンポーネントに、マークアップ機能を提供します。たとえば、`SimpleItem` オブジェクト（文字列を処理するオブジェクト）は `Item` のサブクラスで、色やフォント・サイズなどのマークアップ・コンポーネントに関連する機能を持っています。

B.1.10 CompoundItem クラスと Container クラス

2 つの主なコンテナ・クラス `CompoundItem` と `Container` は、包含関係により他の HTML コンポーネントと関係のある複雑な HTML コンポーネントに対して、基本的なインフラを提供します。クラス `CompoundItem` はクラス `Item` から導出されます。つまり、この型のオブジェクトは、`Item` クラスにより提供されている多数のマークアップ機能も拡張します。このパッケージにある他のクラスの多くは、このクラスからの機能を使用または拡張します。これに比べて、`Container` クラスは `CompoundItem` をコンパクトにしたもので、`Item` クラスから導出されたものではありません。このため、このタイプのオブジェクトにはオーバーヘッドが少ないという利点がありますが、マークアップ機能には欠けます。`Container` クラスから導出されるクラスの主なものとしては、`HtmlBody` や `ImageMap` があります。

B.1.11 HTML ページとオブジェクトの作成

HTML ページを動的に作成するには、（ほとんどの場合）次のクラスからオブジェクトをインスタンス化する必要があります。

- `HtmlHead`
- `HtmlBody`
- `HtmlPage`

スタイル・シート・サポートを追加するために、`Style` クラスと `StyleElement` クラスを使用することがあります。Oracle では、CSS1（カスケード・スタイル・シート 1）を簡単に実装できるように、様々なコンポーネント（クラス `Item` や `HtmlHead` など）へのフックを提供しています。

このパッケージ内のクラスの多くでは、HTML オブジェクトを簡単に作成できるようになっています。HTML を使い慣れていないユーザーは、次のクラスの使用をお勧めします。

- `List`
- `Form`
- `TABLE`（または `DynamicTable`）

場合によって、既存の HTML タグの属性をグループ化する手段としてインタフェースを使用することがあります。このようなインタフェースの目的は、Java プログラマが各種属性の値を簡単に指定できるようにすることです。このようなインタフェースの例としては、IAlign と ITableRules があります。

B.1.12 oracle.html パッケージの継承

CompoundItem クラスまたは Container クラスからコンポーネントを導出して、独自の HTML コンポーネントを作成できます。特定のレイアウト・スタイルを定義し、それをテンプレートして使用するようなハイレベルな HTML クラスを作成できます。たとえば、会社のロゴ、ホーム・ページへのハイパーリンク、さらに著作権表示へのハイパーリンクを持つ CompanyBanner クラスを作成できます。会社のバナー広告を含めたいときは、CompanyBanner オブジェクトを作成し、会社ロゴの GIF ファイルとハイパーリンク用の 2 つの URL を指定します。CompanyBanner クラスのサンプルを次に示します。

```
class CompanyBanner extends CompoundItem {
    // Constructor takes an image and 2 links as arguments
    public CompanyBanner (String logoGIF, String homepageLink,
                          String copyrightLink) {
        addItem(new Link(homepageLink,
                          new Image(logoGIF, "Company Logo", IAlign.TOP, true)));
        addItem(new Link(copyrightLink, "Copyright Notice"));
    }
}
```

これで、CompanyBanner をソース・コードに追加するのみで HTML が生成されます。

```
// Add a company banner
bd.addItem(new CompanyBanner("img/oracle.gif", "http://www.oracle.com",
                              "http://www.oracle.com/copyright.html");
```

演算ロジックを含む HTML クラスを作成することもできます。たとえば、顧客の購入情報をデータベースから問い合わせた結果を HTML 形式で出力する BalanceSheet クラスを作成できます。顧客の貸借対照表の作成は、BalanceSheet オブジェクトをインスタンス化して顧客の ID を指定するのみで済みます。次に BalanceSheet 項目を HtmlPage に追加します。

B.1.12.1 動的コンテンツの使用方法

oracle.html パッケージを使用すると、静的 HTML ページに動的コンテンツを追加できます。これを行うには、標準の HTML ツールまたはテキスト・エディタを使用して HTML ページを作成し、この HTML ページ上の動的コンテンツを挿入する箇所に次のタグを挿入します。

```
<WRB_INC NAME="dynItem1" VALUE="defaultValue">
```

この種の HTML ページはテンプレートと呼ばれます。テンプレートは Web-to-Go アプリケーション・リポジトリに格納できます。Java サーブレットで、パッケージ `oracle.lite.web.html` の `WebToGoHtmlFile` クラスを使用して HTML テンプレートをインポートし、`setItemAt()` メソッドを使用して `<WRB_INC>` タグを動的データに置き換えます。`setItemAt()` メソッドは 2 つの引数をとります。`<WRB_INC>` タグの識別子と、その場所に挿入する動的データです。

これには利点が 2 つあります。

- 好みの HTML エディタを使用して HTML ページの外観を作成できます。
- Java サーブレット・コードを変更しなくても、HTML ページ上の静的データを変更できます。

次の例は、2 つの `<WRB_INC>` タグを持つ静的 HTML ページを示します。この 2 つのタグは、ユーザーの名前とそのユーザーがショッピング・カートに入れている品目を示す表に置き換えられます。この例は次の 3 つのファイルで構成されています。

1. Java サーブレットを起動する HTML ファイル。このファイルには、ユーザーの名前を入力するフォームが含まれています。このフォームは POST メソッドを使用してサブミットされます。

```
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <H1>Login page</H1>
    <form action="/Servlets/shop" method="post">
      <p>Username
      <input type="text" name="userName">
      <p>
      <input type="submit" name="submit" value="Log in">
    </form>
  </body>
</html>
```

2. Java サーブレットによって読み込まれ、`<WRB_INC>` タグを動的データに置き換える HTML テンプレート・ファイル。

```
<HTML>
<HEAD>
<TITLE>Items in Shopping Cart</TITLE>
</HEAD>
<BODY>
<P>Hello
<WRB_INC NAME="user_name" VALUE="Unable to determine your name">
<P>You have the following items in your shopping cart:
<P>
```

```

<WRB_INC name="cart_contents" VALUE="Unable to get shopping cart contents">
<p>
</BODY>
</HTML>

```

3. Java サブレットそのもの。

```

import oracle.html.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.lite.web.servlet.*;
import oracle.lite.web.html.*;

public class shop extends HttpServlet
{

    // Implement doPost() method to handle HTTP POST requests
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {

        // Retrieve the User Profile from the HttpRequest
        OraHttpServletRequest ora_request = (OraHttpServletRequest) req;
        OraUserProfile      oraUserProfile = ora_
request.getUserProfile();

        // Set the content type for the response
        res.setContentType("text/html");

        // get the output stream
        ServletOutputStream out = res.getOutputStream();

        // Get the servlet context
        ServletContext servletContext = getServletContext();

        // Create a new HTML Page based upon the template file in the
repository
        HtmlPage hp = new WebToGoHtmlPage(
            new WebToGoFile(servletContext, "/templates/cart.html"));

        // get user name from query string
        String user = req.getParameterValues("userName")[0];

        // Substitute the WRB_TAB with the username
        hp.setItemAt("user_name", new SimpleItem(user));

```

```
        // add OrderedList
        Orderedlist ol = new Orderedlist();
        ol.addItem(new SimpleItem("hats"));
        ol.addItem(new SimpleItem("gloves"));
        ol.addItem(new SimpleItem("glasses"));
        ol.addItem(new SimpleItem("jackets"));

        // Substitute the WRB_TAB with the list
        hp.setItemAt("cart_contents", ol);

        // Print the HTML page to the output stream
        hp.print(out);
    }
}
```

B.2 oracle.lite.web.html パッケージの使用法

TemplateParser クラスを使用すると、静的 HTML テンプレートに基づく HTML ページの作成プロセスが簡単になります。TemplateParser クラスを継承するサーブレットを作成する場合に必要なことは、動的 HTML を作成する Java コードの作成のみです。HTML テンプレートのタグを動的 HTML に置き換える Java コードを作成する必要はありません。

B.2.1 TemplateParser クラスを使用した HTML テンプレートの処理

次のコードでは、oracle.html パッケージの「動的コンテンツの使用法」の項で説明したショッピング・カートの例を使用し、TemplateParser クラスを使用してこの例を変更しています。

```
import oracle.html.*;
import oracle.lite.web.html.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class shop extends TemplateParser {

    // Return the full path to the template file
    public String getFileName(HttpServletRequest req)
    {
        return "/templates/cart.html";
    }

    // return an array with tag names that need to be replaced

    public String[] [] getWRB_TAGS(HttpServletRequest req)
    {

```

```

        return new String[] {
            {
                // tag name      error message      method
                { "user_name", "Error obtaining UserName", "getUserName" },
                { "cart_contents", "Error obtaining Order List", "getList" }
            };
        }

// get the user name from the HTTP request
public String getUserName(HttpServletRequest req)
{
    return getSingleParameterValue(req, "userName");
}

// Create a list with ordered items
public String getList(HttpServletRequest req)
{
    OrderedList ol = new OrderedList();
    ol.addItem(new SimpleItem("hats"));
    ol.addItem(new SimpleItem("gloves"));
    ol.addItem(new SimpleItem("glasses"));
    ol.addItem(new SimpleItem("jackets"));
    return ol.toHTML();
}
}

```

TemplateParser クラスは抽象クラスであるため、サブクラスでは
 getFileName(HttpServletRequest req) および
 getWRB_TAGS(HttpServletRequest req) という 2 つのメソッドを実装する必要があります。

メソッド	機能
getFileName()	HTML テンプレート・ファイルへのフル・パスを返します。
getWRB_TAGS()	WRB_INC タグ名からメソッド名へのマッピングを含む 2 次元の文字列配列を返します。これらのメソッドでは、WRB_TAG タグを置き換える動的コンテンツを生成する必要があります。

これらのメソッドの詳細は、「Web-to-Go API Specification」を参照してください。

実際の動的 HTML を生成し HTML を文字列として返すメソッドは、すべてクラスで実装する必要があります。このようなメソッドは、サーブレットが POST 要求または GET HTTP 要求を処理するときに自動的にコールされます。各メソッドには次のシグネチャが必要です。

```
public String methodName(HttpServletRequest req)
```

前述の例では、`getUserName()` および `getList()` という 2 つのメソッドが、動的 HTML コンテンツを生成するように実装されています。

B.2.1.1 DBTable を使用したデータベース・データの表示

DBTable クラスを使用すると、データベース・データの取出しと表示が簡単に行えます。データを読み込み専用またはテキスト・フィールドとして表示すると、ユーザーがデータを変更できます。DBTable オブジェクトがリンクされるデータベース表は、常に 1 つのみです。DBTable オブジェクトが `toHTML()` メソッドを使用して出力されると、データ・フィールドの説明を含む非表示 HTML が自動的に生成されます。DBTable オブジェクトが HTML フォームに含まれている場合、このフォームはサーブレット `ProcessForm` により自動的に処理されます。このサーブレットは、データ内の変更をすべて対応するデータベース表に適用します。たとえば、次のとおりです。

```
import oracle.html.*;
import oracle.lite.web.html.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class DisplayEmp extends TemplateParser
{

    // Return the full path to the template file in
    // the Web-to-go repository
    public String getFileName(HttpServletRequest req)
    {
        return "/templates/emp.html";
    }

    // return an array with tag names that need to be replaced

    public String[] [] getWRB_TAGS(HttpServletRequest req)
    {
        return new String[] []
        {
            {
                // tag name      error message      method
                { "emp_table", "Error obtaining EMP table", "showEmp" }
            };
        }
    }
}
```

```
// Create a list with ordered items
public String showEmp(HttpServletRequest req)
{
    DBTable dbTable = new DBTable("EMP");

    // Add a primary key column.
    dbTable.addColumn(new DBPrimaryKey("empno", java.sql.Types.NUMERIC, "", ""))
);

    // Add a regular column that is visible and updateable
    dbTable.addColumn(new DBColumn("ename", java.sql.Types.VARCHAR, "Employee Name"

)

                                .setUpdate(true).setVisible(true));

    // Set the orientation of each record
    dbTable.setOrientation(DBTable.HORIZONTAL);

    // Draw a border around the data
    dbTable.setBorder(1);

    // Obtain a database connection using the class method retrieveConnection()
    Connection conn;
    try
    {
        conn = retrieveConnection(req);
    }
    catch (SQLException e)
    {
        // Return an HTML text containing the error message
        return "<br>"+ "Error: "+e.getMessage();
    }

    // Fetch data and return it as a string
    return FetchAndFormatData(conn, dbTable);
}
}
```

B.2.2 emp.html

テンプレート・ファイル EMP.HTML を次に示します。

```
<HTML>
  <HEAD>
    <TITLE>Display EMP</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION="/Servlets/ProcessForm" METHOD="POST" >
      <br>
      <P>Emp table:</p>
      <P>
        <WRB_INC NAME="emp_table" VALUE="Error:No method found to create emp table">
      </P>
      <input type=submit>
      <input type=hidden name=on_success value="/success.html">
    </FORM>
  </BODY>
</HTML>
```

このサーブレットは、EMP 表から次の出力を生成します。

Employee Name
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER

B.2.2.1 ProcessForm を使用したデータの処理

ProcessForm サブレットは、DBTable クラスにより生成されたデータを含む HTML フォームを自動的に処理します。更新と挿入は自動的に検出され、基になっているデータベース表に適用されます。このサブレットは、フォームを処理した後、フォーム変数 `on_success` の値を取得します。この値は URL とみなされ、ProcessForm サブレットは、その URL からリフレッシュするように指示した HTML をブラウザに返します。開発者はこのメカニズムを使用して、HTML フォームの解析後に実行するアクションを指定できます。

前の項で、DBTable クラスを使用して生成された HTML フォームの例を説明しました。このフォームは、サブミット時に ProcessForm サブレットを使用して自動的に処理されます（HTML テンプレートの FORM ACTION に注意）。処理後、ブラウザは URL 「/success.html」をロードします。

B.2.3 DeleteRecords サブレットを使用したレコードの削除

DeleteRecords クラスにより、1 つ以上の表からレコードを削除するプロセスが簡単になります。DeleteRecords クラスを継承する独自のサブレットを作成すると、文やエラー処理に時間を費やす必要がなく、SQL 構文にのみ集中できます。

ユーザー定義のサブレット DeleteEMP はクラス DeleteRecords の継承ですが、これは次の URL から起動できます。

`http://server_name/DeleteEMP?id=102`

DeleteRecords クラスは抽象クラスであるため、サブクラスでは `getCommands()` メソッドと `onSuccess()` メソッドを実装する必要があります。

メソッド	機能
<code>getCommands()</code>	スーパークラスにより実行される SQL の DELETE コマンドの配列を返します。
<code>onSuccess()</code>	SQL 文が正常に実行された後にコールされます。

これらのメソッドの詳細は、「Web-to-Go API Specification」を参照してください。

例

EMP 表の行を削除するサブレットの例を次に示します。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.lite.web.html.*;

public class DeleteEMP extends DeleteRecords
{
```

```
// Build the SQL delete commands

public String[] getCommands(String id)
{
    String[] commands = new String[1];
    commands[0] = "delete from emp where empno="+id;
    return commands;
}

// This method is called automatically by the super class after executing
// the SQL commands.
public void onSuccess(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{
    // Set the output stream type
    res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
    // Return the HTML to the browser
    out.print("Records deleted" );
}
}
```

B.2.4 マスター・ディテール・フォームの作成と処理

マスター・ディテール表は、DBTable クラスと DBDetailTable クラスを使用して Java サブレットからアクセスできます。これらのクラスは基になるデータベース表からデータを取り出し、このデータを HTML 形式で表示します。これらのクラスを使用すると、基になるデータベース表に対してデータの表示、更新、挿入を行えるマスター・ディテール HTML フォームを簡単に作成できます。HTML フォームの処理に ProcessMasterDetailForm サブレットが使用されると、表のメタ情報を含む非表示フィールドが自動的に生成され使用されます。

ProcessMasterDetailForm はデータの中または新規データ行の中の変更を検出します。このサブレットは正しい SQL 文を作成し、この文を実行して基になるデータベース表にデータの変更を適用します。新規マスター・レコードと新規ディテール・レコードに対して同時に挿入が実行される場合でも、マスターとディテールとの間の外部キー・リレーションシップは自動的にメンテナンスされます。

マスター・ディテール・フォームを生成するコードの例は、「Web-to-Go API Specification」の「DisplayMasterDetail.java.html」を参照してください。

用語集

3 層 Web モデル (Three-Tier Web Model)

クライアント、中間層およびサーバーを含むインターネット・データベース構成。
Web-to-Go アーキテクチャは 3 層 Web モデルに準拠しています。

Apache Server

National Center for Supercomputing Applications (NCSA) から発表されたパブリック・ドメインの HTTP サーバー。

Java Servlet Development Kit

Java サーブレットの開発のために JavaSoft 社が提供しているツール。

Java Web Server Development Kit

Java Web Server Development Kit 1.0.1 は、JavaServer Pages (JSP) と Java サーブレットの開発のために JavaSoft 社が提供しているツールです。

JavaServer Pages

JavaServer Pages (JSP) とは、開発者がページの基になるコンテンツを変更せずにページのレイアウトを変更できるようにするテクノロジーです。JSP は HTML と Java コードを使用し、動的コンテンツとビジネス・ロジックを結び付けたプレゼンテーションを可能にします。

Java アプレット (Java Applets)

ブラウザで実行される小規模なアプリケーションで、動的コンテンツを追加することにより HTML ページの機能を拡張します。

Java サーブレット (Java Servlets)

Java で作成されているプロトコルで、プラットフォームに依存しないサーバー側コンポーネント。Java サーブレットは Java 対応のサーバーを動的に拡張し、要求 - 応答方式を使用して作成されたサービスのための汎用フレームワークを提供します。

JDBC

Java Database Connectivity (JDBC) は Java クラスの標準セットで、リレーショナル・データに対してベンダーに依存しないアクセスを提供します。JDBC クラスは ODBC をモデルにしたもので、複数データベースへの同時接続、トランザクション管理、単純問合せ、バインド変数によるコンパイル済文の操作、ストアド・プロシージャへのコールなどの標準機能を提供します。JDBC では、静的 SQL と動的 SQL の両方がサポートされます。

MIME

Multipurpose Internet Mail Extensions (MIME) とは、メッセージの内容を記述するためにインターネット上で使用されるメッセージ形式です。MIME は、HTTP サーバーが配布対象ファイルのタイプを記述するために使用します。

MIME タイプ (MIME Type)

Multipurpose Internet Mail Extensions (MIME) により定義されているファイル形式。

Mobile Development Kit (Web-to-Go 用) (Mobile Development Kit for Web-to-Go)

Mobile Development Kit (Web-to-Go 用) を使用すると、アプリケーション開発者は、Java サブレット、JavaServer Pages (JSP) または Java アプレットで構成される Web-to-Go アプリケーションの開発とデバッグを行えます。

Mobile サーバー (Mobile Server)

Mobile サーバーは、Mobile サーバー 3 層モデルのアプリケーション・サーバー層に常駐し、Mobile クライアントからの変更要求を処理してデータベース・サーバー内のデータを変更します。

Mobile サーバー・リポジトリ (Mobile Server Repository)

Mobile サーバー・リポジトリとは、Oracle データベースに常駐する仮想ファイル・システムです。このリポジトリは、すべてのアプリケーション・ファイルとアプリケーション定義を含む永続リソース・リポジトリです。

ODBC

Open Database Connectivity (ODBC) は Microsoft 社の標準で、様々なプラットフォーム上のデータベース・アクセスを可能にします。Web-to-Go 用 Mobile クライアント上では、トラブルシューティング用に ODBC サポートを使用可能にします。ODBC サポートを使用すると、ローカルの Oracle Lite データベースに格納されているクライアントのデータを表示できます。

Oracle Lite

Oracle Lite は、Web-to-Go 用 Mobile クライアントのデータベース・コンポーネントです。クライアントがオフライン・モードのときは、アプリケーションとデータは Oracle Lite に格納されます。

Oracle データベース (Oracle database)

Oracle データベースは、Mobile サーバーのデータベース・コンポーネントです。Web-to-Go 用 Mobile クライアントがオンライン・モードのときは、アプリケーションとデータは Oracle データベースに格納されます。

SQL

Structured Query Language (SQL) は、リレーショナル・データベース・エンジンのほとんどで使用される非手続き型データベース・アクセス言語です。SQL 文はデータ・セットに対して実行される操作を記述します。SQL 文がデータベースに送られると、データベース・エンジンは指定されたタスクを実行するプロシージャを自動的に生成します。

Web-to-Go

Oracle Web-to-Go は、Web ベースのモバイル・データベース・アプリケーションを作成および配置するためのフレームワークです。Web-to-Go には、Web-to-Go 用 Mobile クライアント、Mobile サーバーおよび Oracle データベースで構成される 3 層データベース・アーキテクチャが含まれます。サーバーから一元管理され、Web-to-Go アプリケーションは Web-to-Go がサーバーに接続されたとき（オンライン）またはサーバーから切断されたとき（オフライン）に実行できます。オフラインのときは Web-to-Go はデータをローカルにキャッシュし、オンラインに戻ったときにそのデータをサーバーと同期します。

Web-to-Go 用 Mobile クライアント (Mobile Client for Web-to-Go)

Web-to-Go 用 Mobile クライアントは、Web-to-Go の 3 層 Web モデルのクライアント層です。Mobile サーバーと Oracle Lite データベースが含まれます。オフライン・モードに切り替わると、Web-to-Go はユーザーのアプリケーションとデータを Oracle Lite にレプリケートします。オンライン・モードに戻ると、Web-to-Go はデータの変更を Oracle データベースにレプリケートします。

WINDOW シーケンス (Window Sequence)

Web-to-Go がサポートする 2 つの順序のうちの 1 つで、オフライン・モードの Web-to-Go 用 Mobile クライアントに対して一意の主キー値を提供するために使用されるもの。WINDOW シーケンスには、一意の値範囲が含まれます。他のクライアントと値の範囲は重複しません。クライアントが順序の範囲内の値をすべて使用すると、Web-to-Go は新しい一意の値範囲を持つ順序を再び作成します。

一意キー (Unique key)

表の一意キーは、表の各列での一意の列または列グループです。一意キー制約を満たすには、一意キーの値が表の複数の行に出現することはできません。ただし、主キー制約とは異なり、単一列からなる一意キーは NULL 値を含むことができます。

位置付け DELETE (Positioned DELETE)

位置付け DELETE 文により、カーソルの現在行が削除されます。書式は次のとおりです。

```
DELETE FROM table
WHERE CURRENT OF cursor_name
```

位置付け UPDATE (Positioned UPDATE)

位置付け UPDATE 文により、カーソルの現在行が更新されます。書式は次のとおりです。

```
UPDATE table SET set_list
      WHERE CURRENT OF cursor_name
```

オフライン・モード (Offline Mode)

Web-to-Go 用 Mobile クライアントが Mobile サーバーから切断されている状態。オフライン・モードでは、クライアント・アプリケーションはローカルに実行され、データは Oracle Lite でアクセスおよび格納されます。「[オンライン・モード](#)」も参照。

オンライン・モード (Online Mode)

Web-to-Go 用 Mobile クライアントが Mobile サーバーに接続されている状態。「[オフライン・モード](#)」も参照。

外部キー (Foreign Key)

外部キーとは表またはビューに存在する列または列グループのことで、その値は別の表またはビューに存在する行を参照します。外部キーには、一般に、別の表の主キー値と一致する値が含まれます。「[主キー](#)」も参照。

結合 (Join)

2 つの異なる表またはビューに存在するキー（主キーと外部キーの両方）の間に確立された関係。結合は、リレーショナル・データベース内の重複したデータを排除するために正規化された表のリンクに使用します。結合リンクの一般的なものとしては、1 つの表の主キーを別の表の外部キーにリンクして、マスター・ディテール・リレーションを確立するものがあります。結合は SQL 文の WHERE 句条件に対応します。

コントロール・センター (Control Center)

Mobile サーバー・コントロール・センターはブラウザ内で実行される Web ベースのアプリケーションで、これを使用すると Web-to-Go アプリケーションとそのユーザーの管理が簡単になります。管理者はコントロール・センターを使用して、ユーザーまたはグループに対するアクセス権の付与と取消し、スナップショット・テンプレート変数の変更、Web-to-Go からのアプリケーションの削除などの機能を実行します。

サイト (Site)

Web-to-Go は、Web-to-Go 用 Mobile クライアント上の各ユーザーに対してデータベースを作成します。このデータベースはサイトと呼ばれます。1 つのクライアントに複数のサイトを含められますが、サイトは 1 人のユーザーに 1 つのみ可能です。ユーザーは、異なるクライアント上に複数のサイトを所有できます。

索引 (Index)

表内のそれぞれの行に対する高速アクセスを提供するデータベース・オブジェクト。索引を作成すると、表のデータに対して実行される問合せおよびソート操作を高速化できます。また、索引を使用して、一意キー制約や主キー制約などの制約を表に対して規定することもできます。

索引はいったん作成されると自動的にメンテナンスされ、データベース・エンジンにより可能なかぎりデータ・アクセスのために使用されます。

参照整合性 (Referential Integrity)

参照整合性は、レコードが追加、修正または削除されたときにメンテナンスされるマスター・ディテール・リレーション内の表間のリンクの精度として定義されます。

マスター・ディテール・リレーションを注意深く定義しておくことにより、参照整合性が高まります。データベース内の制約によって、データベース（クライアント / サーバー環境でのサーバー）レベルの参照整合性が規定されます。

参照整合性の目的は、孤立したレコード（マスター・レコードとの有効なリンクを持たないディテール・レコード）が作成されないようにすることです。参照整合性を規定する規則により、結果として孤立したレコードを作成するような、マスター・レコードの削除や更新、またはディテール・レコードの挿入や更新を予防できます。

シノニム (Synonym)

表、ビュー、順序、スナップショットまたは別のシノニムに対する代替名（エイリアス）。

主キー (Primary Key)

表の主キーは、表内の各行を一意に識別するのに使用される 1 つの列または列グループです。主キーを使用すると表のレコードにすばやくアクセスできます。また主キーは、2 つの表またはビューの間の結合の基礎として頻繁に使用されます。それぞれの表に対して、主キーは 1 つしか定義できません。

主キー制約を満たすには、主キー値が表の 2 つ以上の列で使用されたり、主キーの一部の列に NULL 値が含まれないようにします。

順序 (Sequence)

連続した番号を生成するスキーマ・オブジェクト。順序を作成した後は、これを使用してトランザクション処理用の一意の順序番号を生成できます。これらの一意の整数には、主キー値を含むことができます。トランザクションで順序番号が生成される場合、トランザクションをコミットしたかロールバックしたかにかかわらず順序が即時増分されます。

[「Web-to-Go」](#) も参照。

スキーマ (Schema)

表、ビュー、索引、順序などを含む、名前の付いたデータベース・オブジェクトの集まり。

スナップショット (Snapshot)

スナップショットとは、Web-to-Go が Oracle データベースからリアルタイムで取得するアプリケーション・データのコピーで、オフラインになる前にクライアントにダウンロードされます。スナップショットは、データベース表全体のコピー、または表の行のサブセットのコピーです。ユーザーが初めてオンラインからオフラインに切り替えるとき、Web-to-Go はクライアント・マシン上にスナップショットを自動的に作成します。その後、オンラインまたはオフラインに切り替えるたびに、Web-to-Go はスナップショットの複雑さに応じて、スナップショットを最新のデータでリフレッシュするか、全体を再作成します。

整合性制約 (Integrity Constraint)

表の 1 つ以上の列に入力できる値を制限する規則。

接続 (Connected)

サーバーに接続されているユーザー、アプリケーションまたはデバイスを指す一般的な用語。Web-to-Go 用 Mobile クライアントは、オンライン・モードのときに接続されています。

切断 (Disconnected)

サーバーに接続されていないユーザー、アプリケーションまたはデバイスを指す一般的な用語。Web-to-Go 用 Mobile クライアントは、オフライン・モードのときに切断されています。

データベース・オブジェクト (Database Object)

データベース・オブジェクトとは、表、ビュー、順序、索引、スナップショットまたはシノニムなどの名前の付けられたデータベース構造体です。

データベース・サーバー (Database Server)

Web-to-Go の 3 層 Web モデルの 3 番目の層。アプリケーション・データを格納します。

同期 (Synchronization)

Web-to-Go 用 Mobile クライアントと Oracle データベースの間でデータをレプリケートするために Web-to-Go が使用するプロセス。Web-to-Go は、ユーザーがオフライン・モードに切り替えたときに、ユーザーのアプリケーションおよびデータを Oracle Lite にレプリケートします。オンライン・モードに戻ると、Web-to-Go はデータの変更を Oracle データベースにレプリケートします。

トランザクション (Transaction)

リレーショナル・データベース内の選択されたデータに対して加えられる一連の変更。トランザクションは通常、INSERT、UPDATE、DELETE などの SQL 文を使用して実行します。トランザクションは、コミットされた（変更が永続的になる）とき、またはロールバックされた（変更が破棄された）ときに完了します。

トランザクションの前に問合せが実行されることがよくあります。問合せを使用して、変更対象の特定のレコードをデータベースから選択しておきます。[「SQL」](#) も参照。

パッケージ・ウィザード (Packaging Wizard)

パッケージ・ウィザードを使用すると、開発者は新規または既存の Mobile サーバー・アプリケーションの定義およびパッケージできます。

ビュー (View)

1 つ以上の表（または他のビュー）から選択されたデータをカスタマイズして表したもの。ビューは「仮想的な表」のようなもので、複数の表（実表と呼ばれます）およびビューからのデータを関連させ、組み合わせることができます。ビューは表示されるデータの選択条件を指定できるため、一種の「格納された問合せ」といえます。

ビューは、表のように、行と列に編成されます。ただし、ビューには、データそのものは含まれません。ビューを使用すると、複数の表またはビューを 1 つのデータベース・オブジェクトとして扱うことができます。

表 (Table)

行と列に編成されたデータを格納するデータベース・オブジェクト。上手に設計されたデータベースでは、各表に単一のトピックに関する情報（たとえば、従業員や顧客の住所など）が格納されます。

マスター・ディテール・リレーション (Master-Detail Relationship)

1 つの表またはビュー（ディテール表またはビュー）の複数行が、別の表またはビュー（マスター表またはビュー）の単一のマスター行に関連付けられている場合に、マスター・ディテール・リレーションがデータベース内の表またはビューの間に存在するといえます。

マスター行およびディテール行は通常、ディテール表またはビュー内の外部キー列と一致するマスター表またはビュー内の主キー列により結合されます。

主キーの値を変更した場合、アプリケーションでは、外部キーの値が主キーの値と一致するように一連の新しいディテール・レコードを問い合わせる必要があります。たとえば、EMP 表内のディテール・レコードが、DEPT 表内のマスター・レコードと同期される場合、DEPT 内の主キーは DEPTNO で、EMP 内の外部キーは DEPTNO にします。「[主キー](#)」および「[外部キー](#)」も参照。

モードの切替え (Switching Modes)

Web-to-Go 用 Mobile クライアントがオフラインに切り替えたりオンラインに戻るために使用するプロセス。クライアントがオフライン・モードに切り替わると、オフラインで作業するために必要なすべてのアプリケーションとデータが Oracle Lite にダウンロードされます。クライアントがオンラインに戻ったときに、Oracle Lite に対するデータ変更を Oracle データベースと同期します。

レジストリ (Registry)

レジストリには、Web-to-Go の一意の名前と値のペアが含まれます。レジストリの名前はすべて一意である必要があります。

レプリケーション (Replication)

分散データベース・システムを構成する複数のデータベース内で、データベース・オブジェクトをコピーしメンテナンスするプロセス。1つのサイトに適用された変更が取得されローカルに格納されてから、各リモート・サイトに転送され適用されます。レプリケーションは、共有データに対する高速のローカル・アクセスをユーザーに提供し、データ・アクセスの代替オプションを提供してアプリケーションの使用を保護します。1つのサイトが使用不可になっても、残りのサイトに対して問い合わせたり更新できます。

レプリケーションの競合 (Replication Conflict)

レプリケーションの競合は、同一のデータに対して矛盾する変更が加えられたときに発生します。データの正しいサブセッティングによって、レプリケーションの競合を回避できます。パッケージ・ウィザードを使用すると、開発者は競合の対処方法についての規則を指定できます。

ワークスペース (Workspace)

Mobile サーバーのワークスペースとは、Web-to-Go アプリケーションに対するアクセスをユーザーに提供する Web ページです。Web-to-Go は、ユーザーが Web-to-Go にログインした後に、ユーザーのブラウザ内にワークスペースを生成します。ワークスペースは、ユーザーが使用できるすべてのアプリケーションのアイコン、リンクおよび説明を表示します。アプリケーションを使用できるようになるのは、管理者がアプリケーションを Web-to-Go システムにパブリッシュし、ユーザーに対してアクセス権を付与した後です。

索引

A

addMIMEHandler() メソッド, 3-18
addRegistryEntry() メソッド, 3-18
AppletProxy クラス, 3-19
ASPHandler, 3-18

D

「DDL」パネル
 パッケージ・ウィザード, 5-35
doGet() メソッド, 3-5
doPost() メソッド, 3-5

G

getConnection() メソッド, 3-11, 3-21
getResultObject() メソッド, 3-23
getServletContext() メソッド, 3-11
getUserPrincipal() メソッド, 3-10
getUserProfile() メソッド, 3-11
getValue() メソッド, 3-18

H

HTML ファイル
 アプリケーション内, 3-3
HTML ページ
 静的, 3-20
 タグ, 3-20
 動的, 3-20
HttpServlet クラス, 3-23

J

jar ファイル
 パッケージ・ウィザードによるパッケージ化, 5-38
java.security.Principal, 3-10
java.servlet.http.HttpServletRequest クラス, 3-10
Java アプレット, 3-3
Java サーブレット, 3-3
 Web-to-Go SDK を使用した開発, 3-7
Java パッケージ, 3-9
 oracle.lite.web.applet, 3-10
 oracle.lite.web.html, 3-10
 oracle.lite.web.servlet, 3-10
JSP, 3-3
 Web-to-Go SDK を使用した開発, 3-6
 Web-to-Go クライアントまたは Web-to-Go サー
 バーでの開発, 3-7

M

MIME タイプ
 登録, 3-18
Mobile サーバー
 基本機能, 2-4

O

oracle.html, 3-10
Oracle JDeveloper
 構成, 3-25
Oracle Lite データベース
 ODBC を介する接続, 3-18
 Web-to-Go SDK を介する接続, 3-18

S

setProperty() メソッド, 3-16
setSessionId() メソッド, 3-23
showDocument() メソッド, 3-24

W

Web-to-Go
 概要, 1-2
 基本構造, 2-2
webtogo.ora ファイル, 3-14
Web-to-Go SDK
 アプリケーションへのアクセス, 3-8
Web-to-Go SDK Web サーバー
 制限事項, 3-7
WebToGoServer.class, 3-15
Web-to-Go Web サーバー
 setProperty() メソッドを使用したプロパティの設定, 3-16
Web-to-Go クライアント
 Web-to-Go セットアップ・プログラムでのインストール, 4-44
 コンポーネントと基本機能, 2-3
Web-to-Go リポジトリ
 アクセス, 3-11
WINDOW シーケンス, 5-2
 大きな値範囲の確保, 5-4
 オンライン・モード用の定義, 5-4
 偶数と奇数のシーケンス番号, 5-5
 作成, 5-3
wtgdebug.exe, 3-15

X

XML ファイル, 5-38

あ

アプリケーション
 Java アプレット, 3-3
 Java サubreット, 3-3
 JSP, 3-3
 SQL ファイルの作成, 4-29
 Web-to-Go SDK を使用した開発, 4-4
 WebToGoServer.class を使用したデバッグ, 3-15
 Web-to-Go クライアントでの実行, 4-44

コンパイル, 4-6
コンポーネント, 3-2
静的コンポーネント, 3-3
定義, 4-7
デバッグ, 3-15
動的コンポーネント, 3-3
配置手順, 4-18
 パッケージ・ウィザードによる定義, 4-19
 パッケージ・ウィザードによる編集, 5-40
 パッケージ・ウィザードによる命名, 5-10
 パブリッシュ, 5-39
 ファイルの表示, 5-15
アプリケーションの作成, 3-2
アプリケーションのルート・ディレクトリ, 5-10
「アプリケーション」パネル
 パッケージ・ウィザード, 5-10
アプリケーション・ロール
 パッケージ・ウィザードによる定義, 5-21
アプレット
 サubreットとの通信, 3-22
 作成, 3-19
 データベース・アクセス, 3-21

お

オープン・リソース API, 3-32
オフライン・モード, 2-7
オンライン / オフライン・モード, 2-7

か

仮想バス, 5-10

く

クライアント同期モード
 オンライン / オフライン・モード, 2-7

こ

コントロール・センター
 アプリケーション・プロパティの設定, 4-38
 オープン・リソース API, 3-32
 起動, 4-35
 新規ユーザーの作成, 4-37
 ユーザー・アクセス権の付与, 4-40

さ

サードレット
 アプレットとの通信, 3-22
 作成, 3-8, 3-23
 デバッグ, 3-18
 動的な追加, 3-17
 登録, 4-7
 パッケージ・ウィザードによる追加, 5-18
 パッケージ・ウィザードによる登録, 3-12
「サードレット」パネル
 パッケージ・ウィザード, 5-18
サイト, 2-4

し

シーケンス, 3-4, 5-2
 WINDOW シーケンス, 5-2
 レプリケーション, 5-30
シーケンス値, 3-5
「シーケンス」パネル
 パッケージ・ウィザード, 5-30

す

スナップショット, 3-4
 インポート, 5-28
 パッケージ・ウィザードによる作成, 5-24
 編集, 5-29
「スナップショット」パネル
 パッケージ・ウィザード, 5-22

て

データベース・オブジェクト
 作成, 4-4
データベース・コンポーネント, 3-4
 シーケンス, 3-4
 スナップショット, 3-4
データベース・コンポーネントのアクセス, 3-5
データベース接続, 3-5
 Java コード内, 3-11
 アプリケーション・ベース, 3-5
 セッション・ベース, 3-5
「データベース」パネル
 パッケージ・ウィザード, 5-19

と

同期, 2-6, 2-7
同期プロセス, 2-7

な

名前と値のペア
 登録, 3-18

は

パッケージ・ウィザード, 4-19
 「DDL」パネル, 5-35
 jar ファイルのパッケージ化, 5-38
 「アプリケーション」パネル, 4-20, 5-10
 開発モードでの使用, 5-9
 概要, 5-7
 起動, 5-7
 「サードレット」パネル, 5-18
 「シーケンス」パネル, 5-30
 スナップショットのインポート, 5-28
 スナップショットの編集, 5-29
 「スナップショット」パネル, 4-24, 5-22, 5-24
 「データベース」パネル, 4-22, 5-19
 ビューと索引のインポート, 5-36
 ファイルのソート, 5-17
 「ファイル」パネル, 5-15
 「レジストリ」パネル, 5-37
 「ロール」パネル, 5-21

ふ

「ファイル」パネル
 パッケージ・ウィザード, 5-15

ゆ

ユーザー
 スナップショット・テンプレートの値の定義, 4-41
ユーザー・コンテキスト, 3-10
ユーザー・サイト, 2-4
ユーザー・プロファイル, 3-10

る

ルート・ディレクトリ, 5-10

れ

「レジストリ」 パネル
パッケージ・ウィザード, 5-37

ろ

「ロール」 パネル
パッケージ・ウィザード, 5-21

わ

ワークスペース
一般的な機能, 2-5
「オフライン」 タブ, 2-7
「オンライン」 タブ, 2-7
カスタマイズ, 3-29
「構成」 タブ, 2-7
「同期」 タブ, 2-7