Oracle9*i* Lite

Web-to-Go 開発者ガイド

リリース 5.0

2001 年 7 月
部品番号：J04358-01

ORACLE®

Oracle9*i* Lite Web-to-Go 開発者ガイド , リリース 5.0

部品番号：J04358-01

原本名：Oracle9*i* Lite Developer's Guide for Web-to-Go, Release 5.0

原本部品番号：A90110-01

目次

3 Web-to-Go アプリケーションの開発

4 Web-to-Go のチュートリアル

5 Web-to-Go アプリケーションの定義

A Web-to-Go サンプル・アプリケーション

B Web-to-Go Java パッケージ

図

表

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 を紹介し、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。
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が含まれています。
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が含まれています。

1

概要

この章では、Oracle Web-to-Go を紹介します。内容は次のとおりです。

- Web-to-Go とは
- 概念と用語
- 実装のためのチュートリアル
- 詳しい実装方法

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、管理および操作を容易にする機能が含まれています。Web-to-Go を使用して、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。Web-to-Go により、ユーザーはアプリケーションの実行が必要になった場合に、選択したアプリケーションやデータを Oracle8*i* 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することが可能になります。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時点で、Web-to-Go を使用して Oracle Lite から Oracle8*i* にデータをレプリケートし、整理統合して、アプリケーションおよびデータを同期させます。

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。

実装のためのチュートリアル

Web-to-Go の概念と用語を理解すると、Web-to-Go の実装処理が理解できます。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プロセスを順に説明します。

詳しい実装方法

Web-to-Go アプリケーションの開発および実装の概要を理解すると、第 3 章「Web-to-Go アプリケーションの開発」の説明を読み進めることができます。この章では、Web-to-Go アプリケーションの開発およびパッケージに関する詳細を提供します。さらに、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されています。

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、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』に説明されています。

2

概念

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。

- Web-to-Go
- Web-to-Go 環境
- 同期の概念
- Web-to-Go での開発
- アクセス制御の管理

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あり、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 Oracle8*i* とデータ変更を同期するユーザーをサポートします。

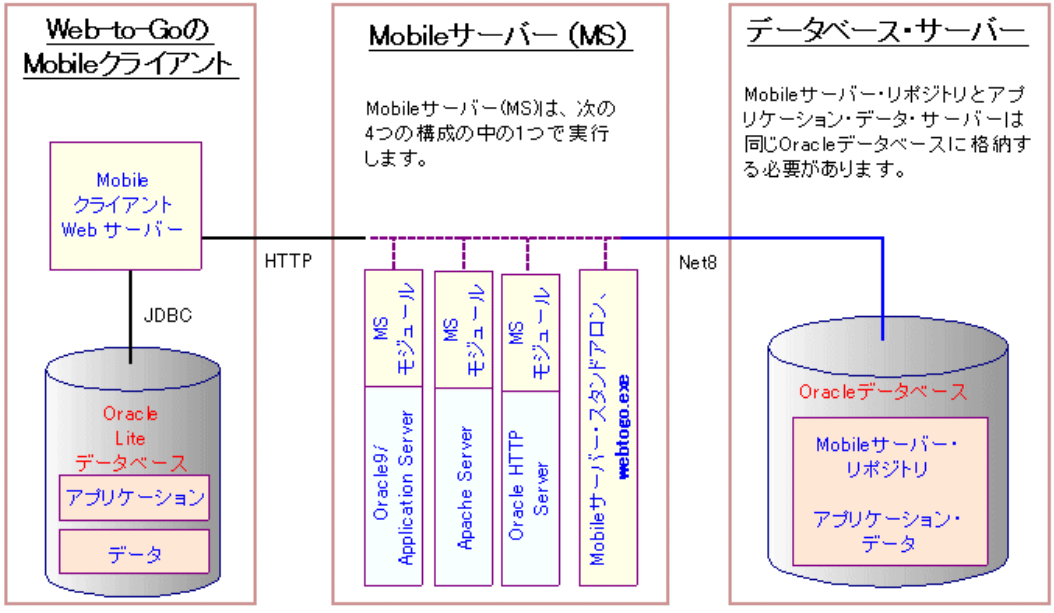
Web-to-Go は、Mobile サーバーから一元的に管理されます。Mobile サーバーのパッケージ・ウィザードおよび Mobile サーバー・コントロール・センターを使用して、管理者は Mobile アプリケーションをパブリッシュおよび管理できます。パッケージ・ウィザードを使用して、開発者は管理者がパブリッシュする予定のアプリケーションを定義し、管理者は定義されたアプリケーションを Mobile サーバーにパブリッシュします。アプリケーションの管理には Mobile サーバー・コントロール・センターを使用します。管理者はここからユーザーの作成と権限の割当てを実行できます。

ユーザーは、Web ブラウザを使用して Web-to-Go の 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 の 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。

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 の 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。

図 2-1 Web-to-Go 環境



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ので、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 Oracle8*i* 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、HTML ページやイメージ・ファイルなどの静的コンポーネント、Java サーブレットなどの動的コンポーネント、および表やシーケンスなどのデータベース・オブジェクトで構成できます。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と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変更をすべて自動的に 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 サーバーの機能を拡張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、表、スナップショットおよびシーケンスという 3 種類のデータベース・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。これにより、Web-to-Go はローカル・クライアント上のコンポーネント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・モードになったときにこのサポートを作成します。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。

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。

Web-to-Go の Mobile クライアント

クライアントの層は、Mobile クライアント Web サーバーおよび Oracle Lite データベースによって構成されます。Mobile クライアント Web サーバーは、ブラウザからアクセスできます。

サイト

Web-to-Go は、Web-to-Go の 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 と 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.odb
oldb40¥john¥entry.odb
```

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.odb
oldb40¥jane¥entry.odb
```

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 の 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 の 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、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

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 の 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、次に示す Web アプリケーション・サーバーのいずれか 1 つとともに稼働するハンドラです。

- Oracle9*i* Application Server（Oracle9*i*AS）
- Oracle HTTP Server
- Apache Server
- スタンドアロンの Mobile サーバーである **webtogo.exe**

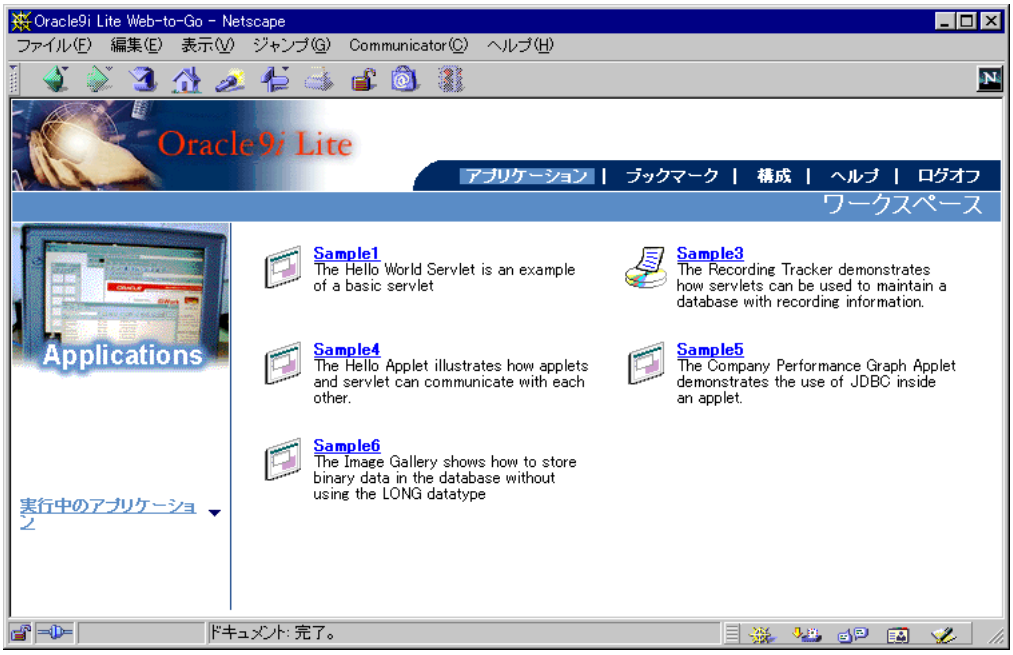
データベース・サーバー

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（Oracle8*i* 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持続リソース・リポジトリです。

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



同期の概念

Web-to-Go の 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 Oracle8*i* サーバーの間でデータ変更をレプリケートします。

データのレプリケーション

データ同期の際、Web-to-Go の Mobile クライアントはデータ変更を Oracle8*i* 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 Oracle8*i* 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 の 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 の Mobile クライアントにより取り出されます。Web-to-Go の 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、Oracle8*i* データベースの表には反映されません。同様に、MGP 実行後に発生

した Oracle8*i* 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 の 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、『Oracle9*i* Lite パブリッシュおよびディプロイ・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。

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。これは、デフォルトのモードです。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同時に、Oracle8*i* データ・サーバーとの同期を開始して、ユーザーのアプリケーションとデータを Oracle Lite データベースや Mobile サーバー・リポジトリと同期させます。

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。ユーザーがオンライン・モードで実行する場合、Web-to-Go の Mobile クライアントはユーザーのログイン要求を Oracle9*i* Application Server（Oracle9*i*AS）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、Oracle8*i* 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。

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

オンライン・モードに切り替えると、Web-to-Go はローカルに加えられたデータ変更を Oracle8*i* データ・サーバーにレプリケートします。Oracle8*i* のデータ・サーバーに対するデータ変更は、Web-to-Go の Mobile クライアントの Oracle Lite データベース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 の Mobile クライアントにダウンロードされます。

ユーザーがオンライン・モードからオフライン・モードに切り替えるときにも、データおよびアプリケーションが同期されます。

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ので、Mobile アプリケーションの開発が容易になります。詳細は、第 3 章「Web-to-Go アプリケーションの開発」、「Web-to-Go アプリケーションの作成」を参照してください。

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ので、Java サーブレットの開発が容易になります。詳細は、第 3 章「Web-to-Go アプリケーションの開発」、「Web-to-Go 用 Java サーブレットの開発」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ので、Mobile アプリケーションの開発を容易に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。

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ーブ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 Oracle8*i* に接続されます。接続プールにより、複雑な接続管理がアプリケーションから隠されます。

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。

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られます。詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

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。

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限をユーザーに割り当てられます。

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り、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
- Oracle8*i* ログイン・ユーザー名とパスワードの変更
- 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、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

3

Web-to-Go アプリケーションの開発

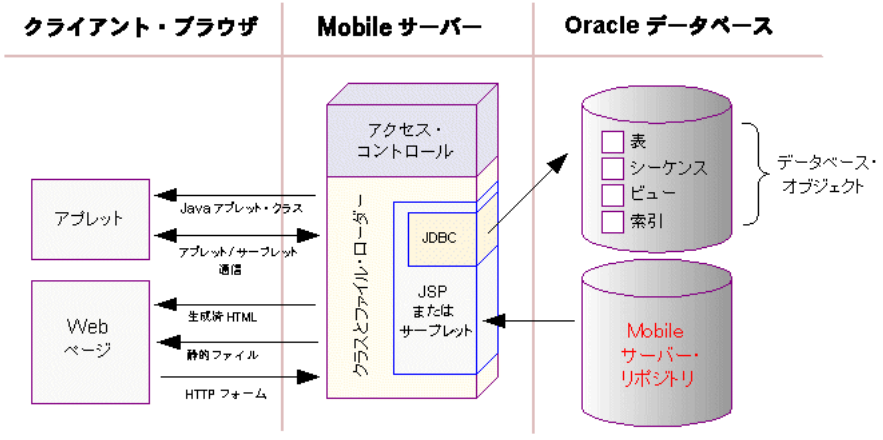
この章では、Web-to-Go アプリケーションの開発方法に関する情報を提供します。内容は次のとおりです。

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



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します。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

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。

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拡張します。

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 ページの機能を拡張します。オフライン・モードでは、Java アプレットは Web-to-Go の 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 Oracle8*i* データベースに直接接続できます。あるいは、Mobile サーバーを介して Oracle データベースに接続します。

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 サーバーの機能を拡張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。

JSP

Web-to-Go は、Java サーブレット・クラス API の拡張である JavaServer Pages（JSP）テクノロジを使用して動的 Web ページを生成します。開発者は、基になるコンテンツを変更せずに JSP を使用して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 と従来のエディタ）とビジネス・ロジックを融合させます。

データベース・コンポーネント

Web-to-Go は、表、スナップショットおよびシーケンスという 3 種類のデータベース・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。これにより、Web-to-Go はローカル・クライアント上のコンポーネント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・モードになったときにこのサポートを作成します。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。

表

Web-to-Go の Mobile クライアントは、表に対する切断状態でのサポートを作成します。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、クライアントがサーバーと同期をとるたびにデータをリフレッシュします。Web-to-Go はデータ変更を自動的に伝播します。

データベース表のスナップショットを作成する前に、データベースと表を Web-to-Go の Mobile クライアント用に構成する必要があります。詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

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スナップショット・テンプレートを含んだ WHERE 句を指定できます。スナップショット・テンプレートの登録方法の詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。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てアプリケーションを拡張する方法の詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

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）シーケンスにより、一意識別子を生成できます。シーケンスを使用すると、データベース表に新規レコードを挿入するときに、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ときに、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 ～ 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、シーケンス値は 100,000,000 個しか使用されません。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、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

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、ビューや索引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あり、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、各クライアントについて、そのクライアントが最初に同期するときに 1 回のみ DDL を実行します。

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、Java サーブレットから JDBC 接続を介してアクセスできます。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・コードの中で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 要求には同じ接続オブジェクトが割り当てられないことがあるため、サーブレットではメソッド `doPost()` または `doGet()` が完了したとき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。

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・ベースでありセッション・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 つしか作成されません。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。

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の種類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チェックして、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せると、ユーザーのロールのリストを取得できます。

JavaServer Pages の開発

Web-to-Go では、JavaServer Pages（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 の Mobile クライアント

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・ファイルとクラス・ファイルが両方とも存在するかどうかをチェックします。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。

Mobile サーバー または Web-to-Go の Mobile クライアント

Mobile サーバーまたは Web-to-Go の 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 の 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ときに、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 ファイルをリストします。「ファイル」パネルの「コンパイル」ボタンをクリックします。リストした 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・ウィザードによりアプリケーション・パッケージに追加されます。

Web-to-Go 用 Java サーブレットの開発

Web-to-Go Java サーブレットは、Mobile Development Kit（Web-to-Go 用）を使用して開発します。Mobile Development Kit（Web-to-Go 用）により、Web-to-Go サーブレットの開発プロセスが容易になります。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 サー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 サーブレットを実行およびデバッグできます。

重要： Mobile Development Kit（Web-to-Go 用）を使用するには、その前に JavaServer Web Development Kit（JSWDK）1.0.1 をインストールする必要があります。JSWDK は Sun 社の Web サイトからダウンロードできます。

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、すべてのファイルとクラスをファイル・システムから直接ロードします。
- セキュリティと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。
- 接続管理は提供しますが、Oracle Lite に対してのみです。ユーザーはスキーマ SYSTEM に接続されます。

Mobile Development Kit（Web-to-Go 用）でのアプリケーションのアクセス

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ために、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` 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。

サーブレットの作成

Web-to-Go では、サーブレットを使用して HTTP クライアント要求を処理します。サーブレットは、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。サーブ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();
    }
}
```

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** パッケージと **oracle.lite.html** パッケージは、どちらも付録 B「Web-to-Go Java パッケージ」で説明します。**oracle.lite.web.servlet** パッケージと **oracle.lite.web.applet** パッケージは、どちらも Mobile Server API 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容易になります。

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 コード内で容易に作成し操作できます。詳細は、付録 B「Web-to-Go Java パッケージ」、「oracle.html パッケージの使用方法」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 サーブレットを作成するために継承できるベース・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 フォームを容易に作成できます。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、『Oracle9*i* Lite Web-to-Go API リファレンス』の「oracle.lite.web.html パッケージの使用方法」を参照してください。付録 B「Web-to-Go Java パッケージ」、「oracle.lite.web.html パッケージの使用方法」も参照してください。

oracle.lite.web.servlet　サー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 サーブレット 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 サーバー上のサー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、『Oracle9*i* Lite Web-to-Go API リファレンス』の「oracle.lite.web.applet」を参照してください。

Web-to-Go のユーザー・コンテキスト

Web-to-Go は、Web-to-Go にログオ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。サーブレットは常にアプリケーションの一部であり、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 により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
- ユーザーがこのアプリケーションに対して持っているロール
- レジストリでこのユーザー用に格納されている名前と値のペア

サーブレットは、`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` からも取得できます。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・プロファイル・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 = ora_request.getUserProfile();
   .
   .
   .
}
```

注意： `OraUserProfile` は Web-to-Go の次のリリースでは使用されない予定なので、`java.security.principal` の使用をお薦めします。

Java コード内でのデータベース接続

サーブレットでは、通常の JDBC 接続をオープンする方法と同じように、次のメソッドをコールして Oracle8*i* に接続します。

```
DriverManager.getConnection ("jdbc:oracle:webtogo");
```

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 = ctx.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();
  .
  .
   .
}
```

サーブレットの実行

Web-to-Go サーブレットを作成した後は、そのサーブレットを実行します。

wtgpack.exe を使用したサーブレットの登録

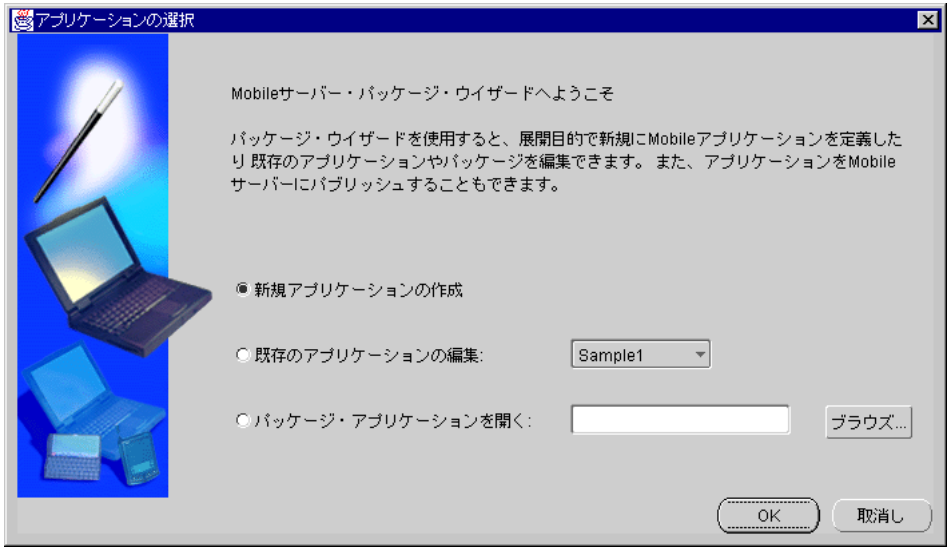
ブラウザからサーブレットにアクセスするには、事前にサーブレットを Mobile クライアント Web サーバーに登録する必要があります。サーブレットを登録するには、まずアプリケーションを登録し、次にこのアプリケーションにサーブレットを追加します。(Web-to-Go では複数のアプリケーションを登録できます。) ブラウザから Mobile クライアント Web サーバーに接続すると、登録済みの全アプリケーションのリストが表示されます。

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とサー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。

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¥wtgsdk¥root です。

パラメータ	定義
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。
XML_FILE	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、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』の初期化パラメータの説明を参照してください。

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： サー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。

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 デバッガの中で 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 のようなアプリケーション・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」も参照してください。

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
   ```

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」も参照。
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「Yes」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:¥oraHome1¥Mobile¥Server¥bin¥webtogo.ora");
WebToGoServer.setProperty ("debug", "true");
WebToGoServer.setProperty ("port", "80");
```

サーブレットの登録

サーブレットは、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。サーブレット・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」にこのサーブレットが追加され、次の URL を入力するとサーブレットにアクセスできます。

```
http://machine_name/servlets/Hello
```

このサーブレットは次の HTML を返します。

<HTML><HEAD><TITLE>

Hello World

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

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ような、独自のサー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 というサー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")
```

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できる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()` を使用してサー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。これにより、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");
```

サーブレットのデバッグ

ソフトウェア開発においては、コードを調べてバグを修正するためにデバッガが使用されることがよくあります。Web-to-Go では、Java サー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 サーブレットを実行することにより、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り、Web-to-Go アプリケーションをデバッグできます。詳細は、「Oracle JDeveloper の構成」を参照してください。

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。サーブレット・コードの中で、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、次のような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
```

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 サー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、サーバーとの接続や Oracle8*i* 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。

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とき、アプレット参照をパラメータとして渡すことにより、`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、サー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、サー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。

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 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、アプレットからサー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、サーブレットから生成することもできます。

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。このオプションの場合は、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・サーブレットは ORACLE_LITE_WEB_SESSION_ID パラメータに必要な <PARAM> タグを HTML に追加します。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・サーブレットは、VALUE 属性を Web-to-Go セッション ID に設定します。

サー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ときは、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、次のように 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 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();
   }
}
```

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。

---

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 Oracle8*i*、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;
}
```

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で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・サンプルで参照されている dBConnection.close() メソッドをコールすると接続をクローズできます。

アプレットーサーブレット通信の開発

Web-to-Go 環境では、Java サーブレットと通信する 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とき、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。この後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 サーブレットと通信するとき、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、アプレットとサーブ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 サーブ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。

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` クラスを拡張する 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();
   }
}
```

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 と 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;
}
```

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ときは、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.URLConnection();
   ..
   ..
   ..
   // 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();
}
```

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、ユーザーにドキュメントのアクセス権限が必要で、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);
}
```

Web-to-Go アプリケーションのデバッグ

Mobile Development Kit（Web-to-Go 用）と Java デバッガ（Oracle の 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。このセクションの例では、Oracle の JDeveloper Release 3.1 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。

Oracle JDeveloper の構成

次の項では、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 Oracle JDeveloper 3.1 を構成する方法を説明します。

デバッグ・プロジェクトの作成

デバッグ・プロジェクトを作成するには、次の手順に従います。

1. JDeveloper を起動します。
2. 「ファイル」→「新規プロジェクト」を選択します。「プロジェクト・タイプ（ステップ 1）」画面が表示されます。

図 3-4「プロジェクト・タイプ」画面

3. 「プロジェクト・ファイル名を指定」フィールドにプロジェクトのファイル名を入力します。
4. 「プロジェクト・タイプ」画面の 2 番目のセクション（「作成するプロジェクト・タイプ」）で、「空のプロジェクト」ラジオ・ボタンを選択します。

5. 「次へ」をクリックします。「プロジェクト・オプション（ステップ 2)」画面が表示されます。次の情報を入力します。

フィールド	説明
デフォルトのパッケージ名	このフィールドは空白のまま残します。
プロジェクト・ソースのパス	*Oracle_Home* ¥mobile¥sdk¥wtgsdk¥src¥sample1¥servlets
出力ディレクトリ	*Oracle_Home* ¥mobile¥sdk¥wtgsdk¥root¥sample1¥servlets

図 3-5「プロジェクト・オプション」画面

6. 「次へ」をクリックします。「プロジェクト情報（ステップ 3)」画面が表示されます。

図 3-6「プロジェクト情報」画面

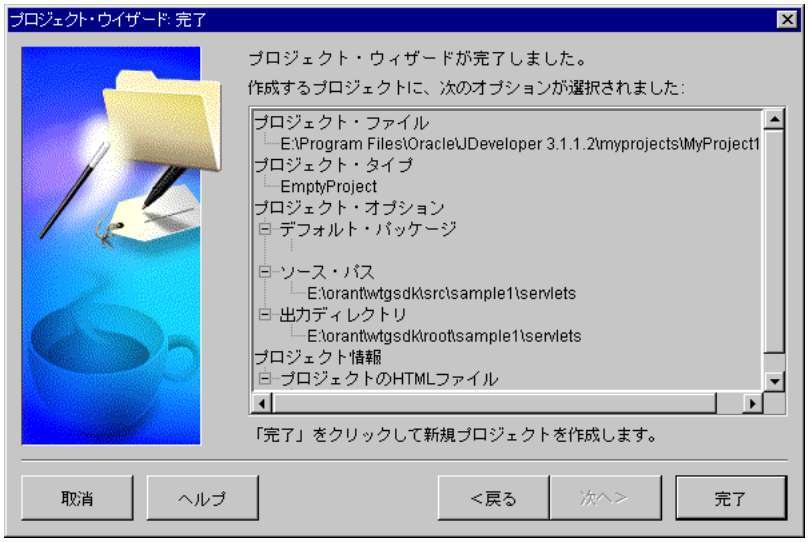
7. 次の各フィールドに情報値を入力します。

フィールド	説明
タイトル	プロジェクトの名前です。
作者	作成者の名前です。
著作権	著作権の日付です。
会社名	会社名です。
説明	プロジェクトの説明です。

8. 「次へ」をクリックします。「完了」画面が表示されます。

9. 「完了」画面にリストされているオプションを確認します。必要な場合は「戻る」をクリックします。「完了」をクリックします。

図 3-7「完了」画面



プロジェクト・プロパティの設定

プロジェクト・プロパティをデバッグ用に構成します。JDeveloper のナビゲータのメニュー・バーで、「プロジェクト」→「プロジェクト・プロパティ」を選択して「プロパティ」ダイアログ・ボックスを表示します。次のように構成します。

「パス」タブ

ターゲットの JDK バージョン:

ターゲット JDK のバージョンとして「Java version "JDK1.2.2_JDeveloper"」を選択します。

ソース・ルート・ディレクトリ:

このディレクトリがソースのルートを指していることを確認します。次のディレクトリに設定する必要があります。

Oracle_Home¥mobile¥sdk¥wtgsdk¥src¥sample1¥servlets

出力ルート・ディレクトリ:

出力ルート・ディレクトリを次のディレクトリに設定します。

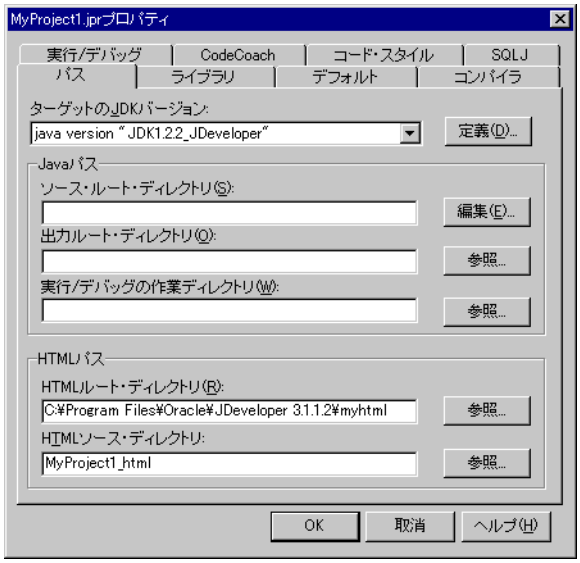
Oracle_Home¥mobile¥sdk¥wtgsdk¥root¥sample1¥servlets

Mobile サーバーはクラス・ファイルがこのディレクトリにあると想定します。

実行 / デバッグの作業ディレクトリ：

実行 / デバッグ・ディレクトリを Oracle_Home¥mobile¥sdk¥bin に設定します。

図 3-8「プロパティ」ダイアログ・ボックス



「ライブラリ」タブ

JDeveloper 3.1 では、CLASSPATH 設定のかわりにライブラリを使用します。これにより、一連の **.jar** ファイルの管理が容易になります。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¥consolidator.jar

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

WTGSDK ライブラリを作成するには、次の手順を実行します。

1. 「プロパティ」ダイアログ・ボックスから「ライブラリ」タブを選択します。
2. 「追加」をクリックします。

図 3-9「追加する Java ライブラリの選択」ダイアログ・ボックス

3. 「新規」をクリックします。
4. WTGSDK と入力します。
5. 「クラス・パス」フィールドの隣の「...」をクリックします。「ファイル」ダイアログ・ボックスが表示されます。
6. 前述の 6 つの **.jar** ファイルを選択し、「オープン」をクリックします。
7. 「OK」をクリックします。

注意： 標準の JDeveloper ライブラリの 1 つは WTGServer と呼ばれます。このライブラリを使用しないでください。

「コンパイラ」タブ

「デバッグ情報を含める」ボックスを選択します。「OK」をクリックして「プロパティ」ダイアログ・ボックスをクローズします。

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. JDeveloper 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。「ファイル」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¥wtgsdk¥scr¥sample1¥servlets にある Java ソース・ファイル **Helloworld.java** を選択し、「オープン」をクリックします。
3. ディレクトリ *Oracle_Home*¥mobile¥sdk¥wtgsdk¥src にあるファイル **RunWebServer.java** をプロジェクトに追加します。

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。「ブレークポイントの切替え」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. ナビゲータ・ウィンドウで、ファイル **RunWebServer.java** を選択します。
2. メニューから「実行 / デバッグ "RunWebServer"」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/
```

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

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。次のようにすると、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。

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・アプリケーションを作成するために使用する API の詳細は、次のディレクトリにある『Oracle9*i* Lite Web-to-Go API リファレンス』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 の 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 の 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、カスタマイズされた 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 Mobile サーバーに指示します。（**webtogo.ora** ファイルのパラメータの正しい設定値は、「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 の Mobile クライアントによって作成された **webtogo.odb** ファイル、Web-to-Go の Mobile クライアント用の **webtogo.ora** ファイル、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。）

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。

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際の出発点として、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、JavaServer Pages（JSP）と **.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 **.html** ファイルとイメージ・ファイルもすべて含まれています。

Mobile Development Kit（Web-to-Go 用）には、JSP ファイルのコンパイル、Web-to-Go の 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・リポジトリへの必要な全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）。

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、次のような機能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

コントロール・センターを作成するための API の使用の詳細は、次のディレクトリにある『Oracle9*i* Lite Web-to-Go API リファレンス』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。パッケージ・ウィザードを介してのみ可能です。詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

4

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。

- 概要
- アプリケーションの開発
- アプリケーションのファイナライズとパブリッシュ
- アプリケーションの管理
- Web-to-Go の 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には、概要のセクションと 4 つのセクションが含まれており、それぞれにトピックが含まれています。各セクションは、「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。各セクションを完了した時点で、その内容を復習する、関連ドキュメントを参照する、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

セクション	説明
概要	このチュートリアルの目的と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ファイナライズとパブリッシュ	「To Do List」アプリケーションのコンポーネントを Mobile サーバーに配置する方法を説明します。
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 の Mobile クライアントでのアプリケーションの実行	Web-to-Go の Mobile クライアントをインストールして使用する方法を説明します。

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り、Web-to-Go アプリケーションのライフ・サイクル全体をガイドします。「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状態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。

作業の準備

このチュートリアルでは、Mobile Development Kit（Web-to-Go 用）、Mobile サーバーおよび Web-to-Go のデモがすべて同一コンピュータ上にインストールされ、構成済み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とクライアント・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。

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 および Windows2000 でのユーザー・ログイン :	■ 開発用コンピュータ上の Windows NT および Windows2000 ログイン・ユーザーには、ADMINISTRATOR 権限が必要です。
インストール済みの Java コンポーネント :	■ Java Development Kit 1.2.2（以上）。 ■ Java Web Server Development Kit 1.0.1。
インストール済みの Oracle コンポーネント :	■ Mobile サーバーと Web-to-Go のデモ（Oracle9*i* Lite CD-ROM）。 ■ Mobile Development Kit（Web-to-Go 用）（Oracle9*i* Lite CD-ROM）。

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・モードおよびオフフライン・モードでテストします。

アプリケーションの開発

このセクションでは、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。
JavaServer Pages（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。

このセクションでは、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と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
- ステップ 6: アプリケーションの変更

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 文を使用して自分で作成するか、Mobile Development Kit（Web-to-Go 用）のデモをインストールする際に作成されたものを使用できます。

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。

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

Mobile Development Kit（Web-to-Go 用）デモのインストール中に、**webtogo.odb** という Oracle Lite データベースが自動的に作成されます。Mobile Development Kit（Web-to-Go 用）デモのインストールは DOS プロンプトで次のように入力します。

```
cd Oracle_Home¥mobile¥sdk¥wtgsdk¥src
sdkdemos.bat
```

CREATEDB 文を使用して自分で作成する場合は次のように入力します。

```
createdb webtogo webtogo
```

Mobile Development Kit（Web-to-Go 用）を使用する際に、この Oracle Lite データベースを使用して、データベースへのアクセスを必要とするアプリケーション・コンポーネントを開発できます。

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・オブジェクトをこの Oracle Lite データベースに作成します。

開発フェーズの最中に、「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 Oracle8*i* 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。

注意： データベース・オブジェクトは、この **webtogo.odb** データベースに作成します。

「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。

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 プロンプトで次のように入力します。

```
msql system/x@jdbc:polite:webtogo
@Oracle_Home¥mobile¥sdk¥wtgsdk¥src¥tutorial¥tutorial.sql
```

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 に似ています。詳細は、『Oracle9*i* Lite Palm 開発者ガイド』の付録 D、「Mobile SQL」を参照してください。

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、コードを見なおしてください。

JavaServer Pages

「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 ページの完了済み「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¥wtgsdk¥src¥tutorial¥ToDoList.jsp.

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¥wtgsdk¥src¥tutorial¥ToDoBean.java

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¥wtgsdk¥src¥tutorial¥InsertToDo.java

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¥wtgsdk¥src¥tutorial

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 サーブレットと JavaBean のコンパイル
3. JSP のインストール

必須アクション

1. 必要なライブラリ（Java Servlet Development Kit や Web-to-Go ライブラリなど）を含めるために CLASSPATH を設定します。CLASSPATH を設定するには、DOS プロンプトで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。

   ```
   cd 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 サー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、次のサーブレット・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
```

ステップ 4: アプリケーションの定義とサーブレットの登録

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、Mobile クライアント Web サーバーへのアプリケーションのサー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ーブレットをすべて 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 や JavaBean を登録する必要はありません。

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
- サーブレットの登録
- 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ので、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はありません。

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。

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、そのサーブレットを登録します。

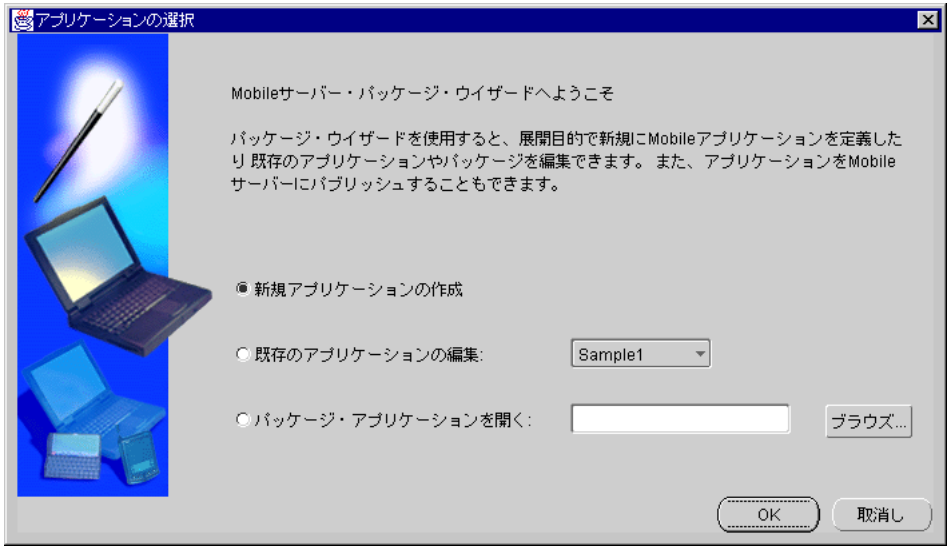
1. パッケージ・ウィザードをデバッグ・モードで起動します。DOS プロンプトで次を入力します。

   a. `cd Oracle_Home¥mobile¥sdk¥bin`

   b. `wtgpack -d`

   図 4-1 に示すように、パッケージ・ウィザードが表示され、新しいアプリケーションを作成するか、既存のアプリケーションを変更するかのオプションが表示されます。

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」を選択し、「OK」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。

図 4-2「アプリケーション」パネル

3. 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。

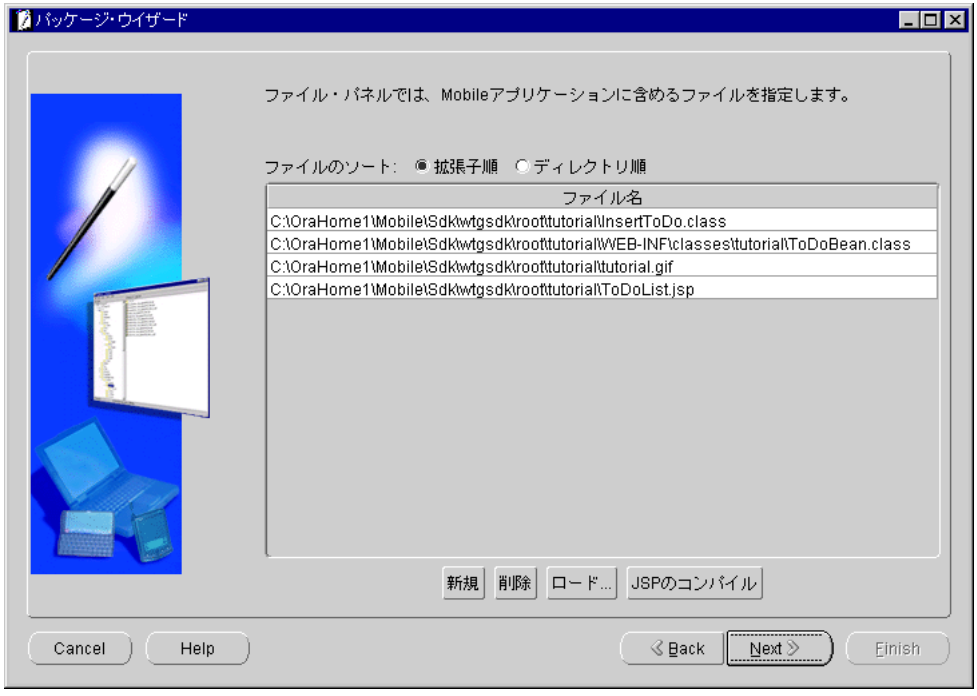
表 4-5「To Do List」アプリケーションの設定

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 new 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
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¥tutorial
アイコン	tutorial.gif

4. 「Next」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。

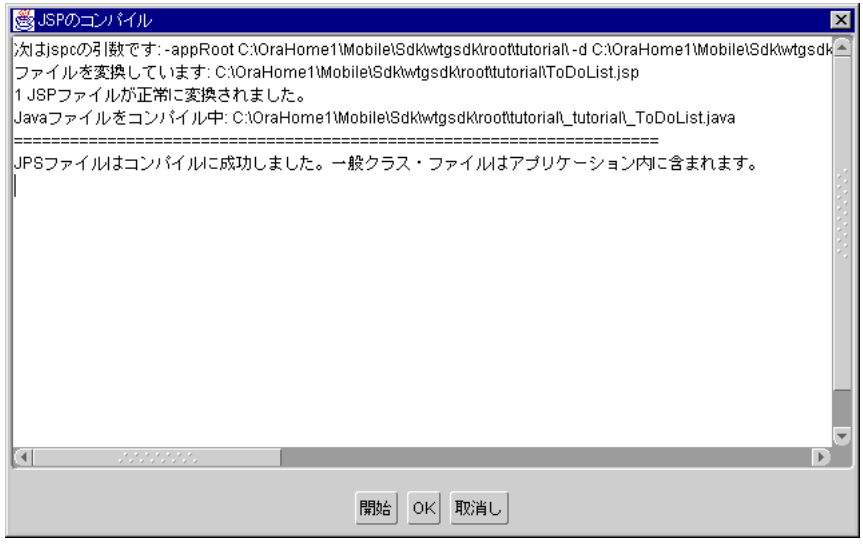
図 4-3 すべてのアプリケーション・ファイルのアップロード



5. 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 4-4 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。

6. 「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 4-5 新しく生成されたファイルは自動的に追加される

7. 「To Do List」アプリケーション・サーブレットを表示するには、「Next」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 つしか含まれていないため、「サーブレット」パネルに表示される行は 1 行のみです。

図 4-6「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。

8. サーブレット名を「insert」に変更し、「Next」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。「Finish」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。

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。

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 サー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、『Oracle9*i* Lite パブリッシュおよびディプロイ・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。

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、どのサーブレットがロードされたかが示されます。サーブレットに `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 4-7 アプリケーションのリスト

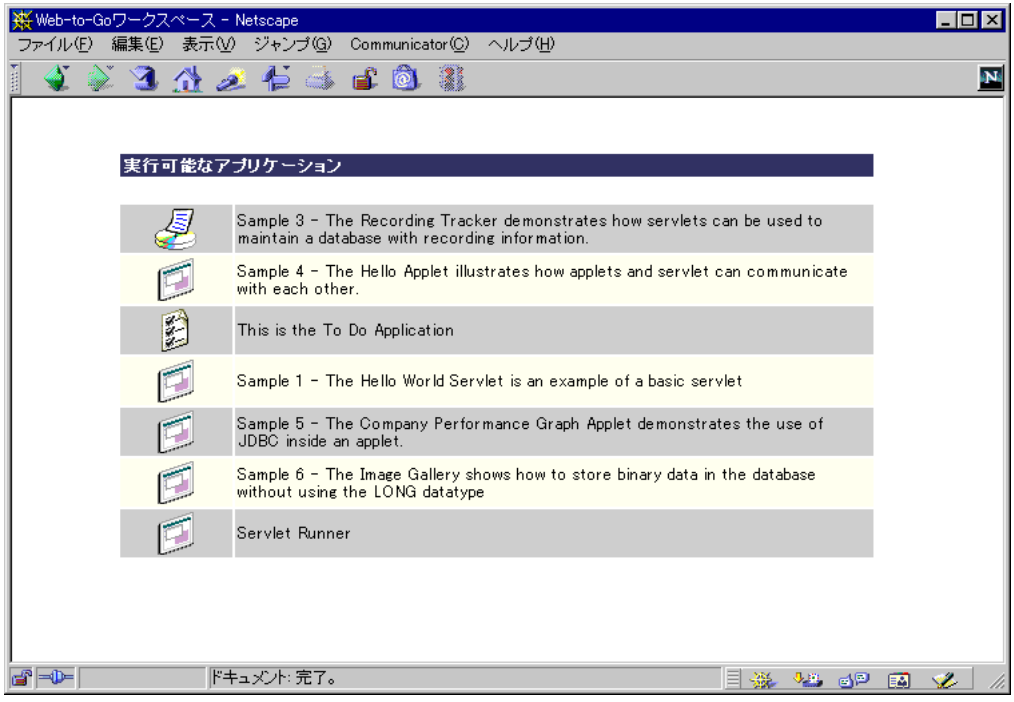


表 4-6 システムが現在認識しているアプリケーションのリスト

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み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。
Sample1	Hello World サーブレットは、基本的なサーブレットの例です。このアプリケーションは次の場所にあります。 *Oracle_Home*¥mobile¥sdk¥wtgsdk¥root¥sample1
Sample3	Recording Tracker は、サー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¥wtgsdk¥root¥sample3
Sample4	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
Sample5	Company Performance Graph（企業業績グラフ）アプレットは、アプレット内で JDBC を使用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¥sample5
Sample6	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。

ステップ 6: アプリケーションの変更

このステップでは、JSP と Java サーブレット・コードの両方を変更します。さらに、サーブレットをアプリケーションに追加し、Mobile Development Kit（Web-to-Go 用）Web サーバーに登録します。この手順はすべてオプションです。

JSP の変更

JSP に加えられる変更は、即時処理されます。Web ブラウザに「To Do List」アプリケーション・ページを再ロードすると、これらの変更を表示できます。

Java サーブレット・コードの変更

Java サーブレット・コードを変更する場合は、次の手順を実行して変更を完了する必要があります。

ステップ 7: サーバーの再起動

Mobile クライアント Web サーバーを再起動します。

必須アクション

1. ステップ 5 のアクション 2、b に説明されているように Java サーブレットを再コンパイルします。
2. Web サーバーのウィンドウで [Ctrl]+[C] を押して、Mobile クライアント Web サーバーを停止します。
3. 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`

   Web サーバーが起動され、どのサーブレットがロードされたかが示されます。サーブレットに `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。

アプリケーションへのサーブレットの追加

アプリケーションにサーブレットを追加して Mobile クライアント Web サーバーに登録するには、ステップ 4 に説明されているようにパッケージ・ウィザードを使用する必要があります。

これで「To Do List」アプリケーションの開発が正常に終了しました。

アプリケーションのファイナライズとパブリッシュ

このセクションでは、アプリケーション定義を完了してアプリケーションを Mobile サーバーにパブリッシュする方法を説明します。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

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。

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。

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モードで起動します。

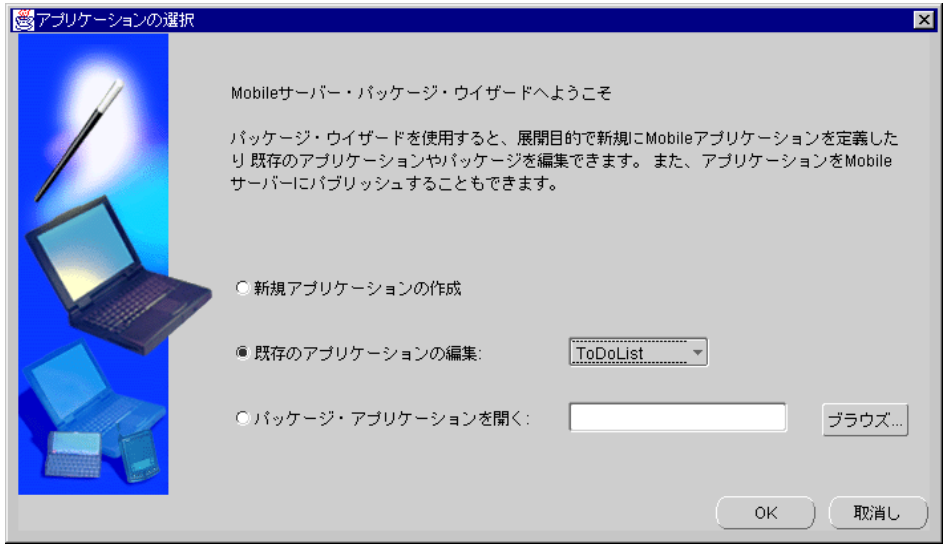
1. DOS プロンプトで次を入力します。

   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。

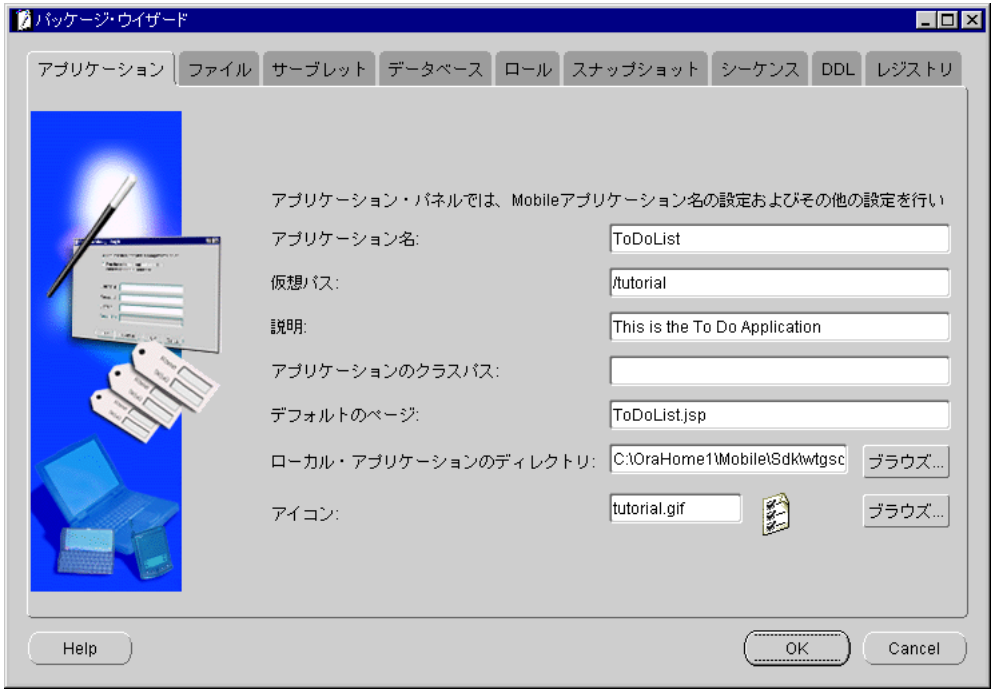
図 4-8「既存のアプリケーションの編集」の選択



c. 「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。

d. 「OK」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、「開発」セクションのステップ 4 で入力した情報と同じ情報が含まれています。

図 4-9「アプリケーション」パネル



詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

2. 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 new To Do Application
アプリケーションのクラスパス	

表 4-7 指定されている値（続き）

フィールド	値
デフォルトのページ	ToDoList.jsp
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¥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。

ステップ 2: アプリケーションのサーブレットの定義

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」サーブレットを配置します。

パッケージ・ウィザードを使用したサーブレット配置の詳細は、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。

「サーブレット」パネル

「サーブレット」パネルは、アプリケーションのサーブレットを配置するために使用します。「サーブレット」パネルには、「開発」フェーズのステップ 4 で含まれていた情報と同じ情報が含まれています。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、これらを Mobile クライアント Web サーバーへの登録用に選択します。「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 つしか含まれていないため、「サーブレット」パネルに表示される行は 1 行のみです。

図 4-10「サーブレット」パネル

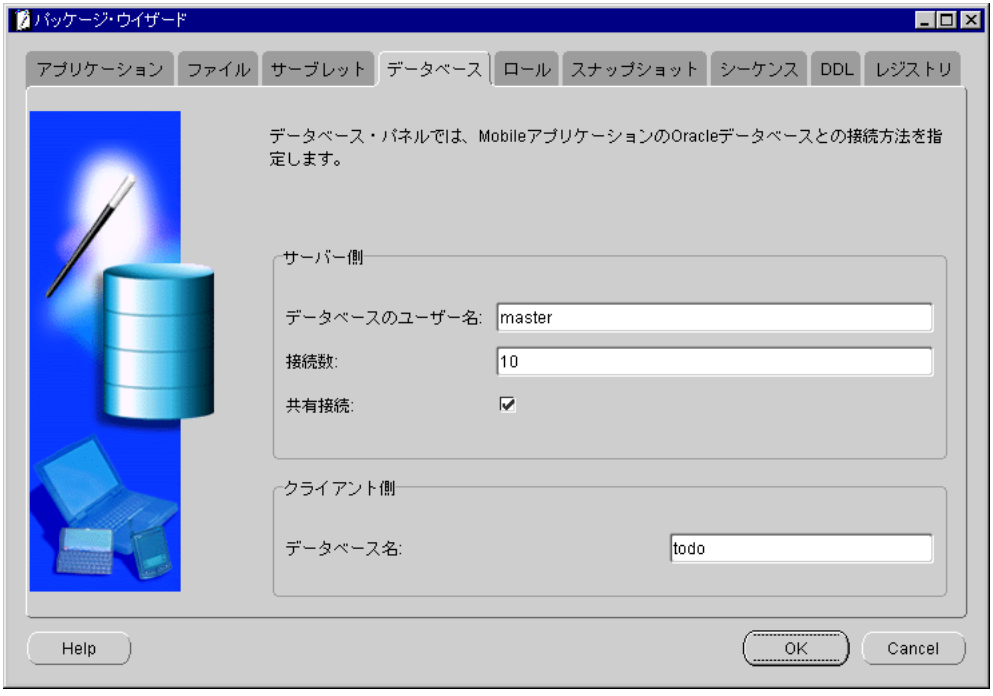
必須アクション

「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。

ステップ 3: Oracle8*i* 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 Oracle8*i* への接続を定義します。「データベース」パネルでは、Oracle8*i* 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、「データベース」パネルでは「To Do List」アプリケーションのデータベース・オブジェクトを含む Oracle8*i* 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



詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

必須アクション

次の各フィールドに指定された値を入力します。

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

この設定は、「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 の 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。

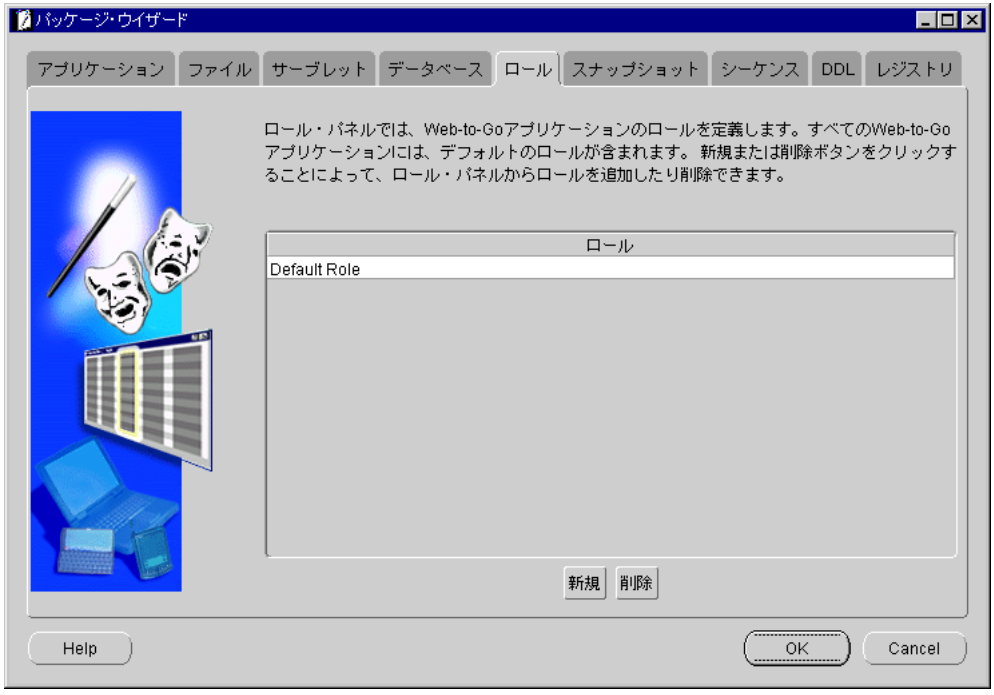
「ロール」タブをクリックします。「ロール」パネルが表示されます。

ステップ 4: アプリケーション・ロールの定義

「ロール」パネル（図 4-12 を参照）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。アプリケーションにはすべて、最低でもデフォルト・ロールと呼ばれるロールが 1 つあります。チュートリアル・アプリケーションには、特別なロールはありません。

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ることはありません。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「アプリケーション・ロール」を参照してください。

図 4-12「ロール」パネル



必須アクション

チュートリアル・アプリケーションではロールを使用しないため、このパネルはスキップできます。「スナップショット」タブをクリックします。図 4-13 に示すような「スナップショット」パネルが表示されます。

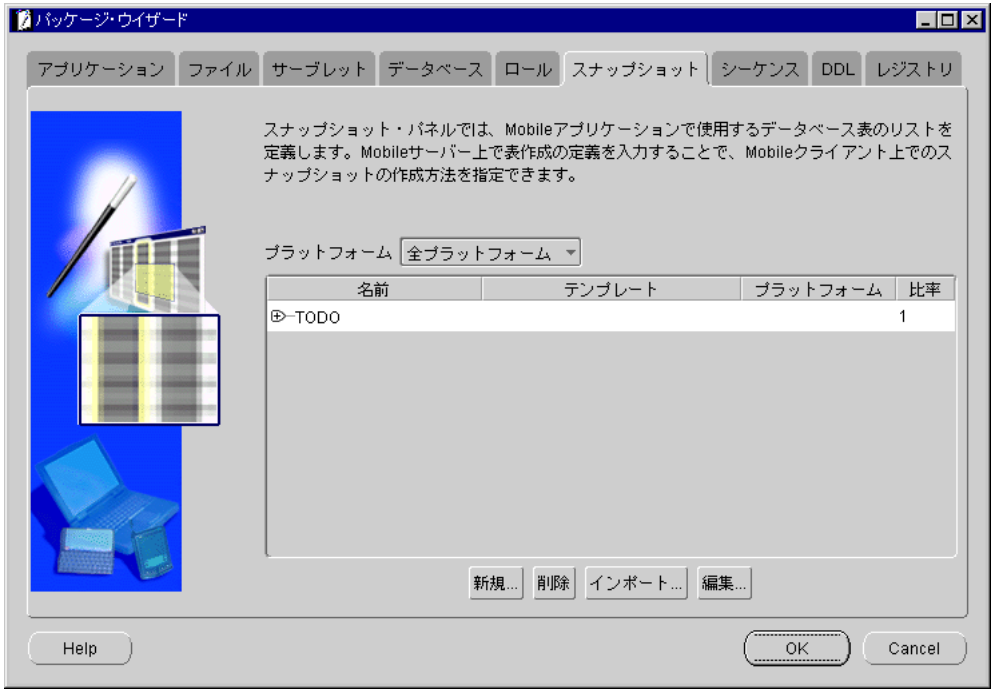
ステップ 5: アプリケーションのデータベース・オブジェクトの定義

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。

「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 4-13「スナップショット」パネル



必須アクション

アクションは不要です。「シーケンス」パネルを表示するには、「シーケンス」タブをクリックします。

「シーケンス」パネル

「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、Oracle8*i* に実際のシーケンスを作成します（「ステップ 8: Oracle8i へのデータベース・オブジェクトの作成」を参照）。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。

図 4-14「シーケンス」パネル

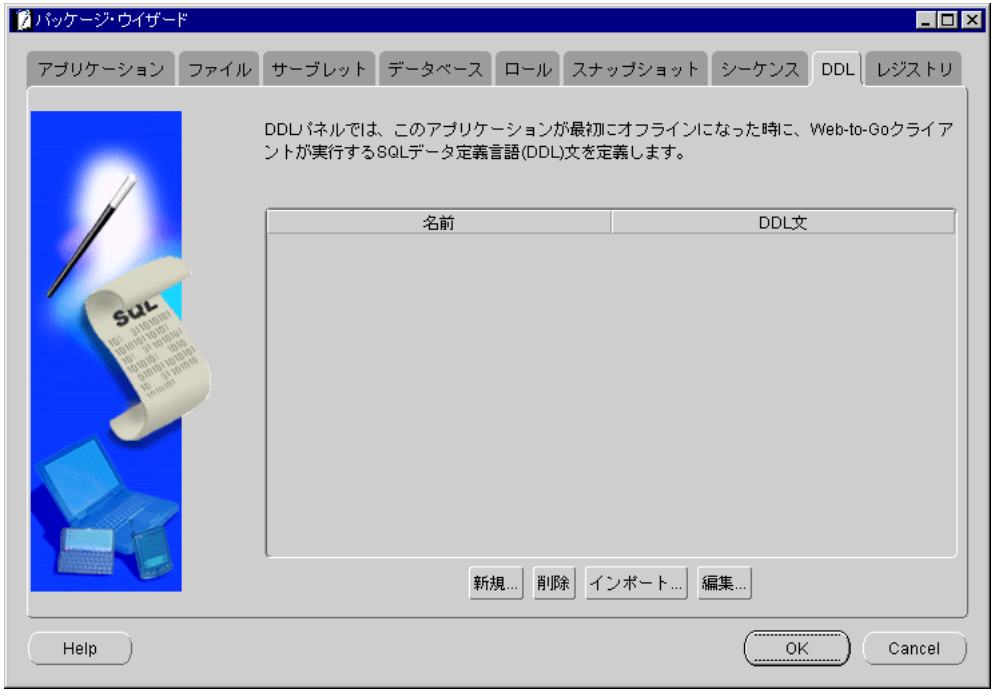
必須アクション

この時点で必要なアクションはありません。「DDL」パネルを表示するには、「DDL」タブをクリックします。

「DDL」パネル

「DDL」パネルは、「To Do List」アプリケーションの DDL（データ定義言語）文を定義するために使用します。「DDL」パネルでは、Web-to-Go が最初に同期するときにクライアント上で実行される DDL 文を定義します。DDL 文は、Web-to-Go では自動的に同期されないデータベース・オブジェクト（索引やビューなど）用に使用します。アプリケーション DDL はオプションです。「To Do List」アプリケーションにはアプリケーション DDL が含まれていません。

図 4-15「DDL」パネル



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。次の各フィールドに指定された値を入力します。

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

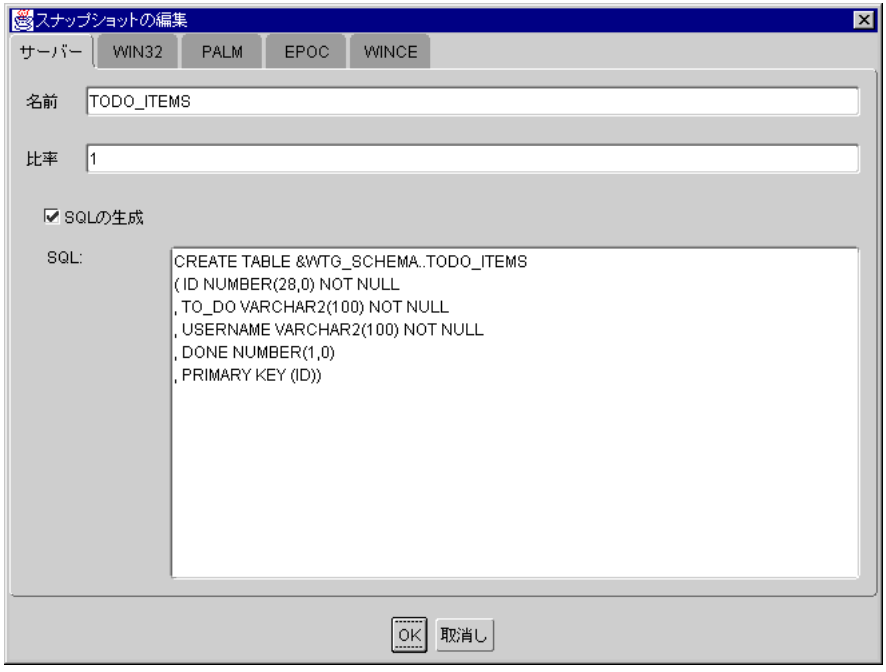
2. 「OK」をクリックします。「表」ウィンドウが表示され、使用可能な表のリストが表示されます。

図 4-16「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 4-17「スナップショットの編集」ダイアログ・ボックス



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。Win32 クライアント用のパネルが表示されます。「テンプレート」フィールドの SQL 文を変更して「OK」をクリックします。

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

図 4-18「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. 「シーケンス」タブをクリックします。「シーケンス」パネルが表示されます。

8. 「インポート」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。

   「シーケンス」ウィンドウには、使用可能なシーケンスのリストが表示されます。

図 4-19「シーケンス」ウィンドウ

9. TODO_SEQ シーケンスを選択し「追加」をクリックします。「閉じる」をクリックします。
10. 「DDL」タブをクリックします。「DDL」パネルが表示されます。
11. 「To Do List」アプリケーションには DDL が含まれていません。「レジストリ」タブをクリックします。「レジストリ」パネルが表示されます。

ステップ 6: アプリケーションのレジストリの定義

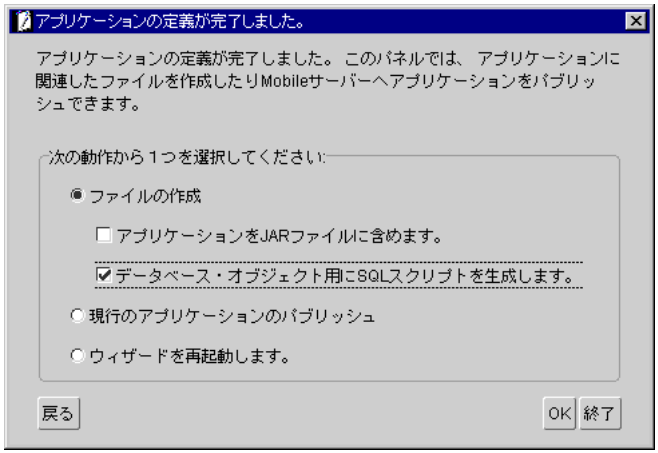
「レジストリ」パネルは、アプリケーションのカスタムの名前と値のペアを定義します。名前と値のペアには、アプリケーションの追加情報を指定します。「レジストリ」パネルには、名前と値のペアをすべて登録し、それぞれにデフォルト値を指定します。レジストリの名前と値のペアは配置フェーズで定義し、管理フェーズで変更します。

図 4-20「レジストリ」パネル

必須アクション

「OK」をクリックします。図 4-21 に示すように、「アプリケーションの定義が完了しました。」ダイアログ・ボックスが表示されます。

図 4-21「アプリケーションの定義が完了しました。」ダイアログ・ボックス



ステップ 7: 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が完了しました。」ダイアログ・ボックスを使用します。

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 は定義されていないので、このファイルは空です。

ステップ 8: Oracle8*i* 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 Oracle8*i* に作成します。

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¥wtgsdk¥root¥tutorial¥sql
sqlplus master/master@webtogo.world @ToDoList.sql
```

このスクリプトは、Oracle8*i* 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。

ステップ 9: アプリケーションのパブリッシュ

このステップでは、アプリケーションを Mobile サーバーにパブリッシュします。

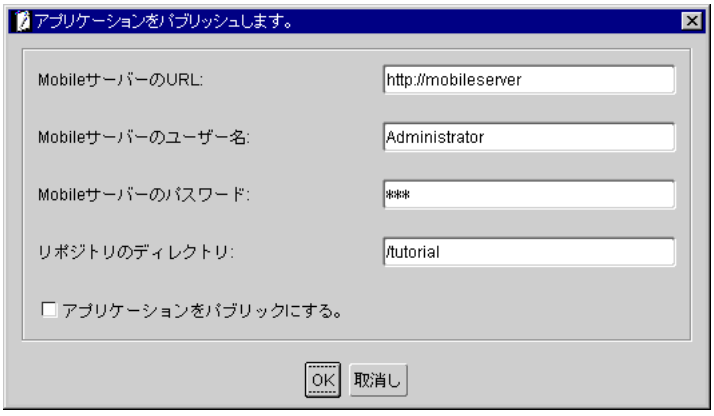
必須アクション

次の手順を実行して、配置フェーズを完了します。

1. 「アプリケーションの定義が完了しました。」ダイアログ・ボックスで「現行のアプリケーションのパブリッシュ」を選択して、「OK」をクリックします。

2. 図 4-22 に示すように、「アプリケーションをパブリッシュします。」ダイアログ・ボックスが表示されます。

図 4-22「アプリケーションをパブリッシュします。」ダイアログ・ボックス

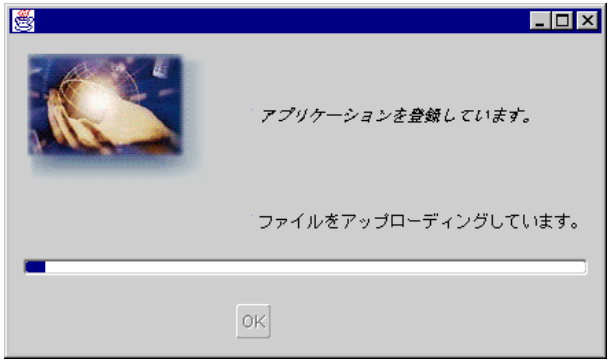


3. 次の各フィールドに指定された値を入力します。

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： *`server`* は、使用する Mobile サーバーの URL に置き換えてください。

4. 「To Do List」アプリケーションをパブリッシュするには、Mobile サーバーが実行中であることを確認してから「OK」をクリックします。配置の進捗状況を示すダイアログが表示されます。



5. 配置フェーズを終了すると、「メッセージ」ウィンドウが表示されます。「OK」をクリックしてから「終了」をクリックし、パッケージ・ウィザードを終了します。

これで「To Do List」アプリケーションの配置が正常に終了しました。

アプリケーションの管理

このセクションでは、「開発」セクションで作成およびテストし、「配置」セクションで配置したアプリケーションを管理する方法を説明します。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のセクション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、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

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。

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
   ```

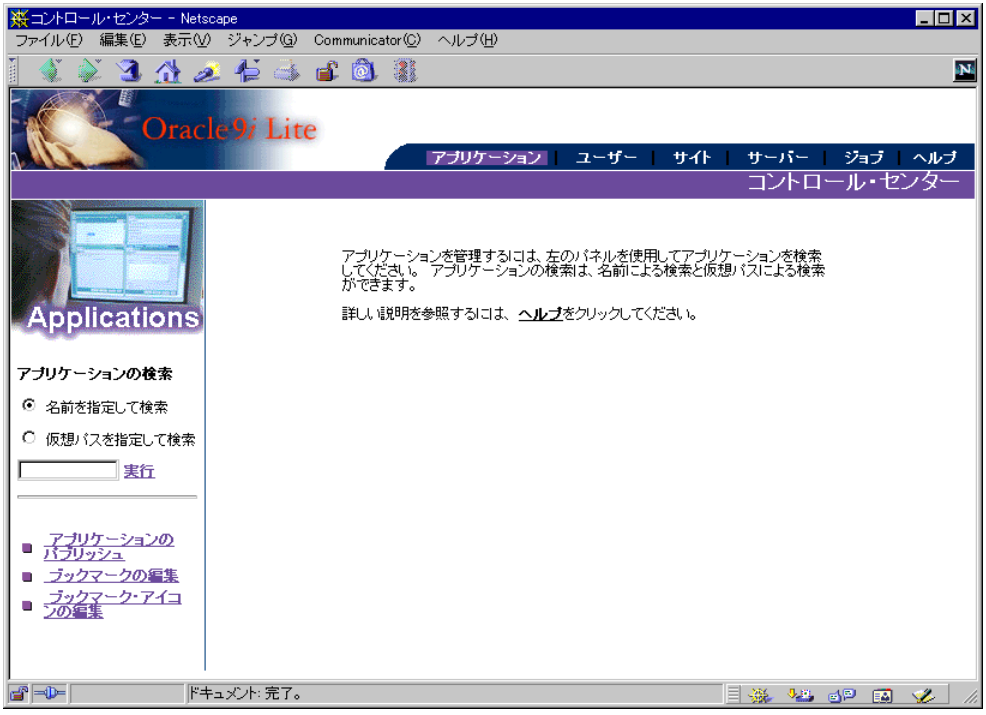
   注意： **`server`** 変数は、使用する Mobile サーバーのホスト名に置き換えてください。

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「Control Center」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 4-23 コントロール・センターの起動



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。

必須アクション

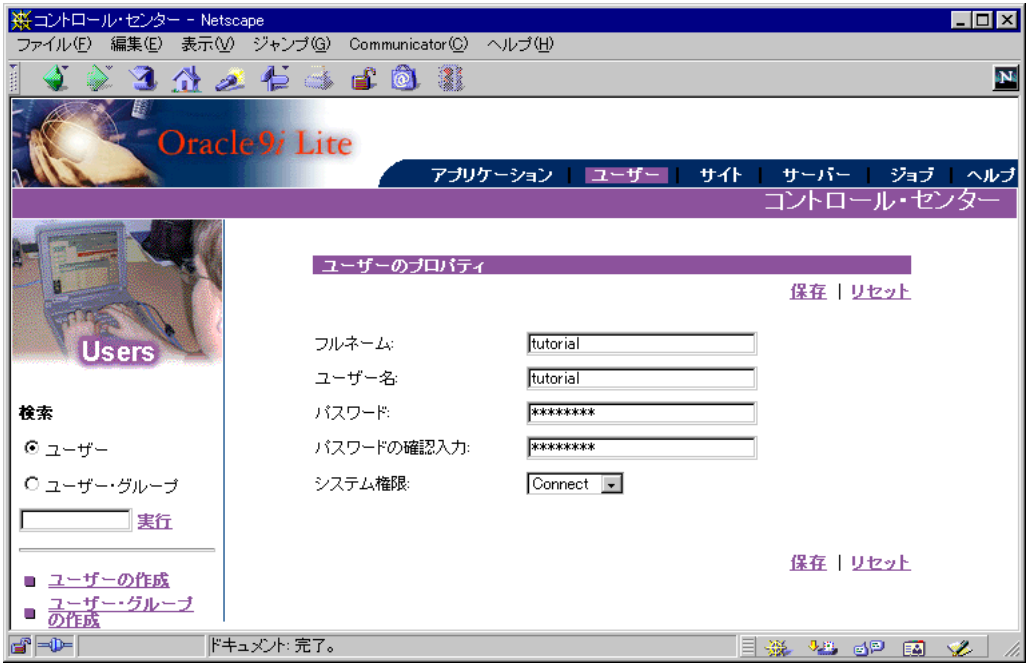
Mobile サーバーの新規ユーザーを作成するには、次の手順を実行します。

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
フルネーム :	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 :	Connect

図 4-24「ユーザーのプロパティ」パネル



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。

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。

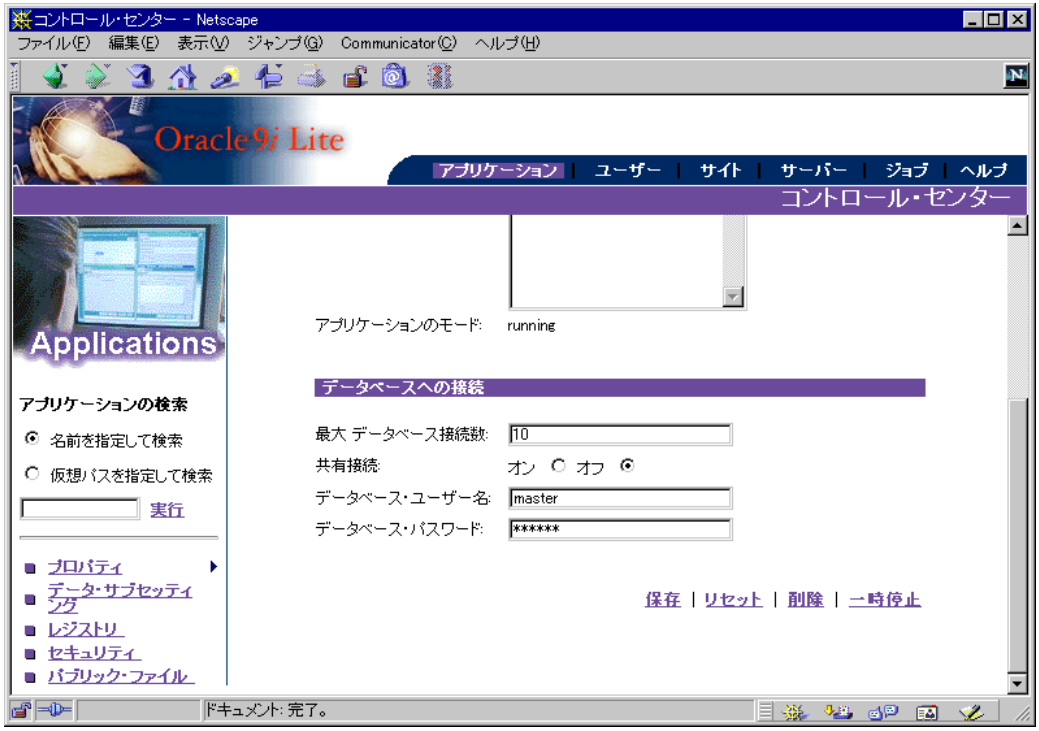
1. 右上にある「アプリケーション」タブをクリックします。「アプリケーション」メニューが左側のフレームに表示されます。
2. 「To Do List」アプリケーションを検索する条件を入力します。「名前を指定して検索」オプションを選択し、検索フィールドに「ToDo」と入力して「実行」をクリックします。ワークスペースに「To Do List」アプリケーションが表示されます。

> **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 4-25 アプリケーション・プロパティの設定



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。

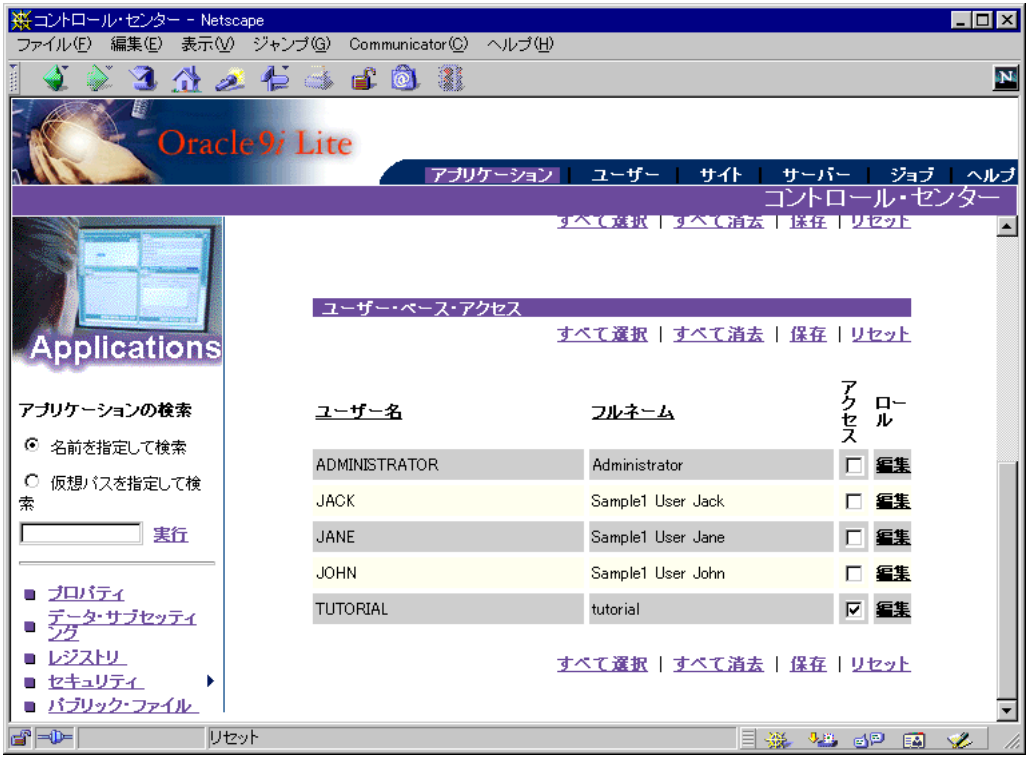
必須アクション

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「セキュリテイ」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 4-26 アプリケーションへのユーザー・アクセス権の付与



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 の 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。

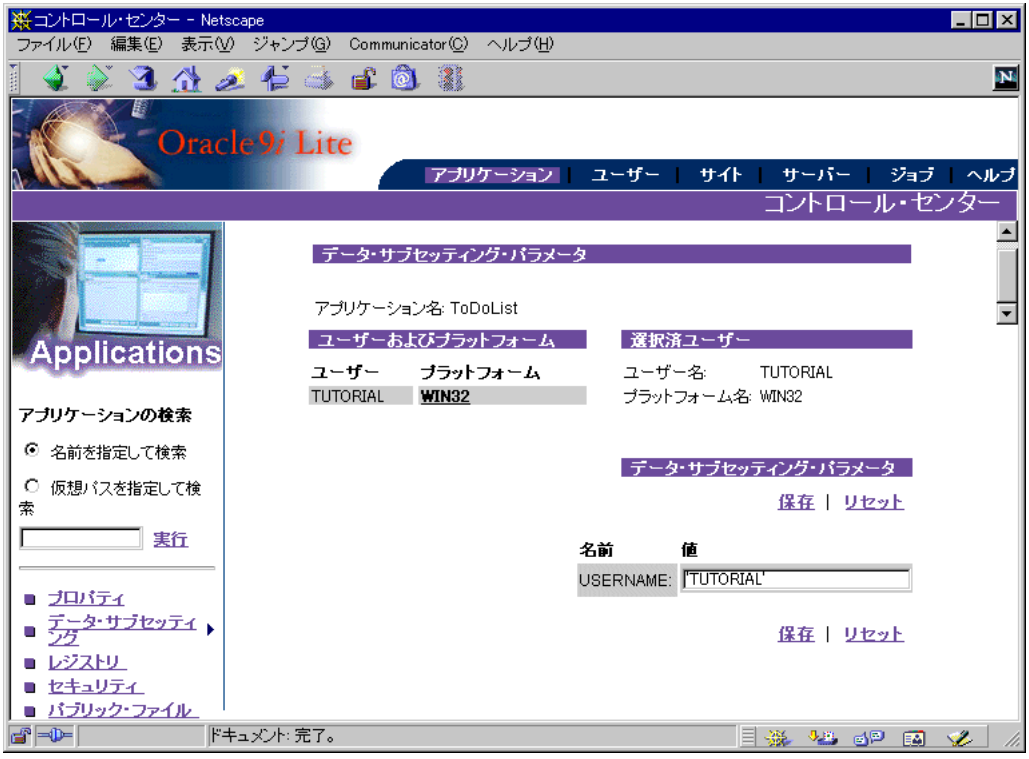
必須アクション

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・チュートリアルの場合は、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 4-27 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、『Oracle9*i* Lite パブリッシュおよびディプロイ・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。

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

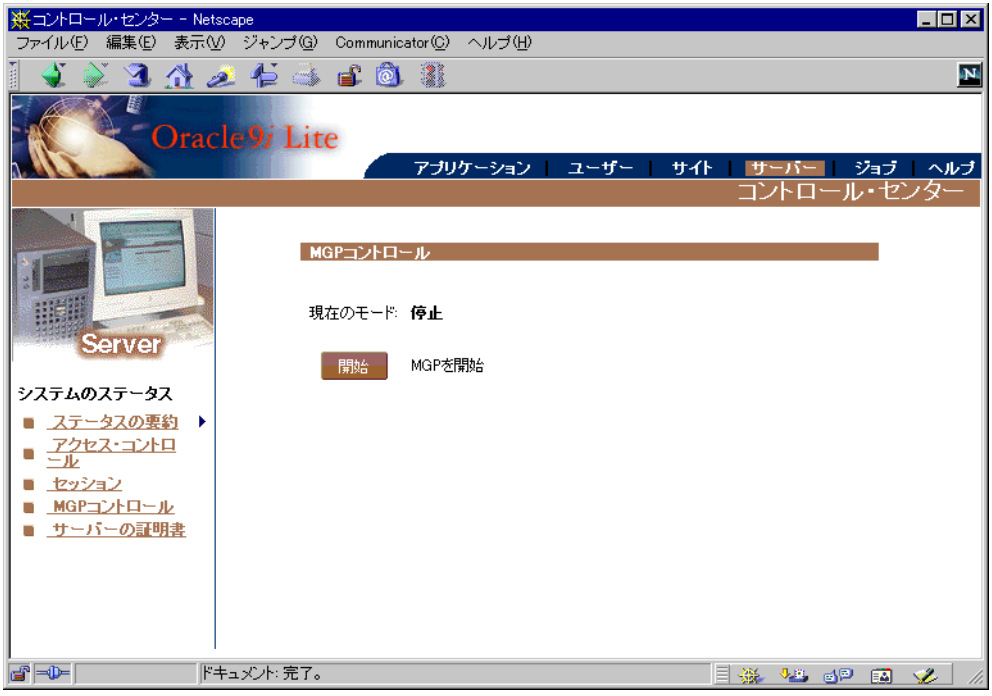
このステップでは、Web-to-Go コントロール・センターから Consolidator Message Generator and Processor（MGP）を起動します。Web-to-Go の Mobile クライアントは同期時に、データ変更を Mobile サーバーにアップロードします。MGP は、Oracle8*i* データベースにデータ変更を適用する別プロセスです。

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、「STATUS」ボタンが表示されます。

図 4-28 MGP の開始



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

このセクションでは、「開発」のセクションで作成し、「配置」のセクションで配置し、「管理」のセクションで管理したアプリケーションを使用する方法を説明します。ここでは次の作業を実行します。

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

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

このセクションでは、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 の 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。

必須アクション

Web-to-Go の 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

  Web-to-Go 用の Mobile クライアントを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

  Win32 用の Mobile クライアントを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

  ブランチ・オフィス・プログラムをダウンロードするには、ここをクリックしてください。

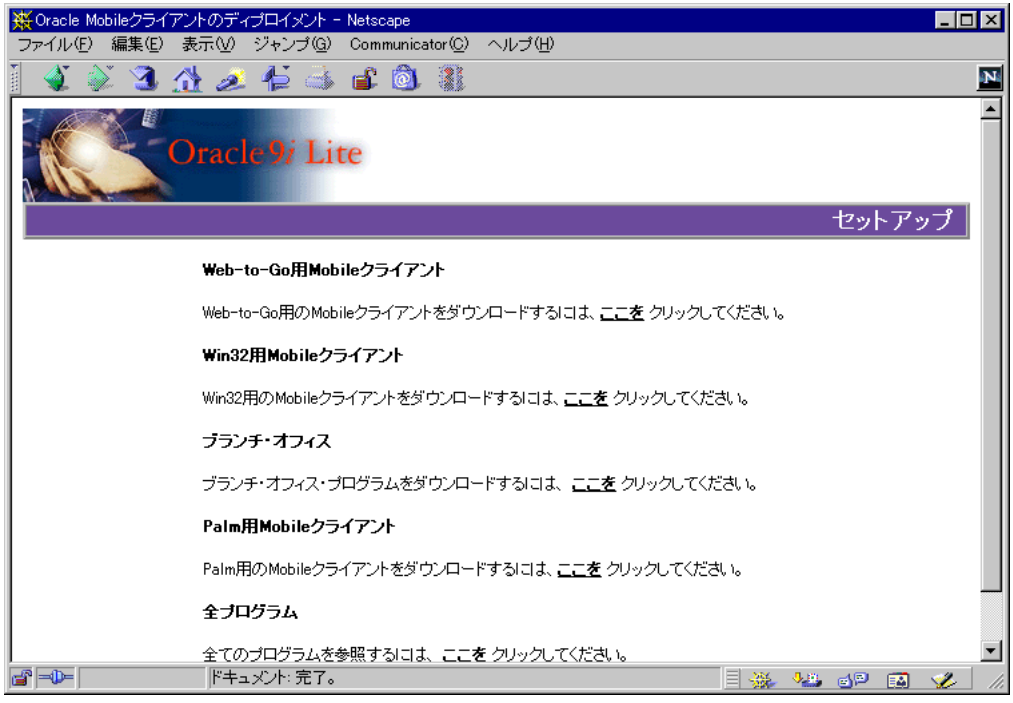
- 全プログラム

  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 の 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 4-29 Web-to-Go の Mobile クライアントのインストール



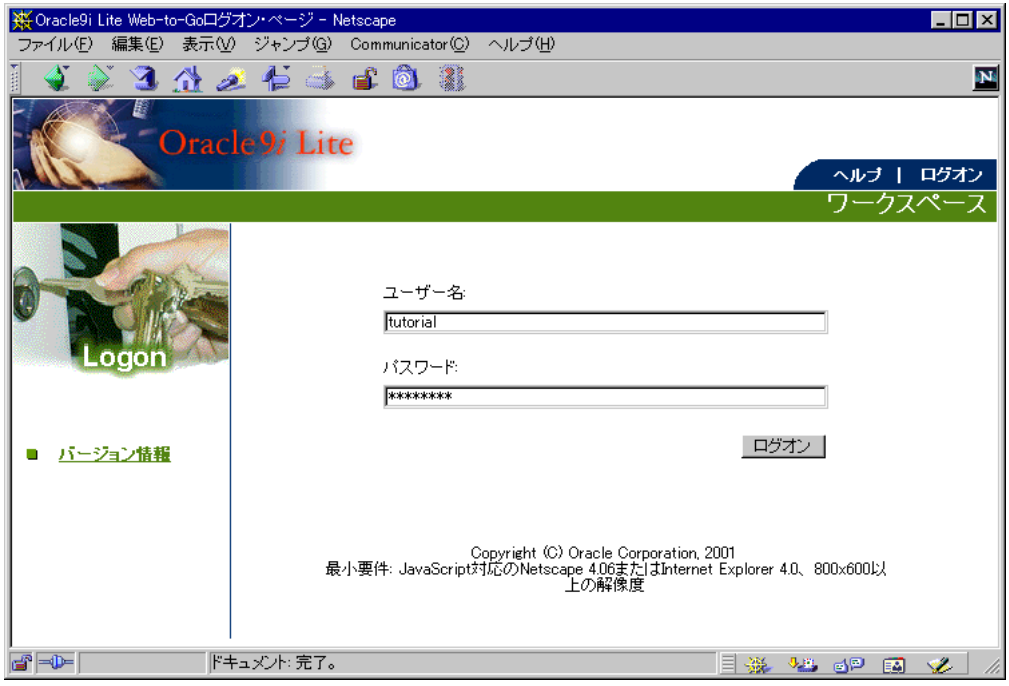
3. Netscape を使用している場合は、セットアップ・プログラムを保存する場所を選択して「OK」をクリックします。Windows エクスプローラで、**setup.exe** をダブルクリックしてセットアップ・プログラムを実行します。

   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 の 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 4-30 Web-to-Go ログオン・ページ



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

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

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 の Mobile クライアントのホスト名に置き換えてください。

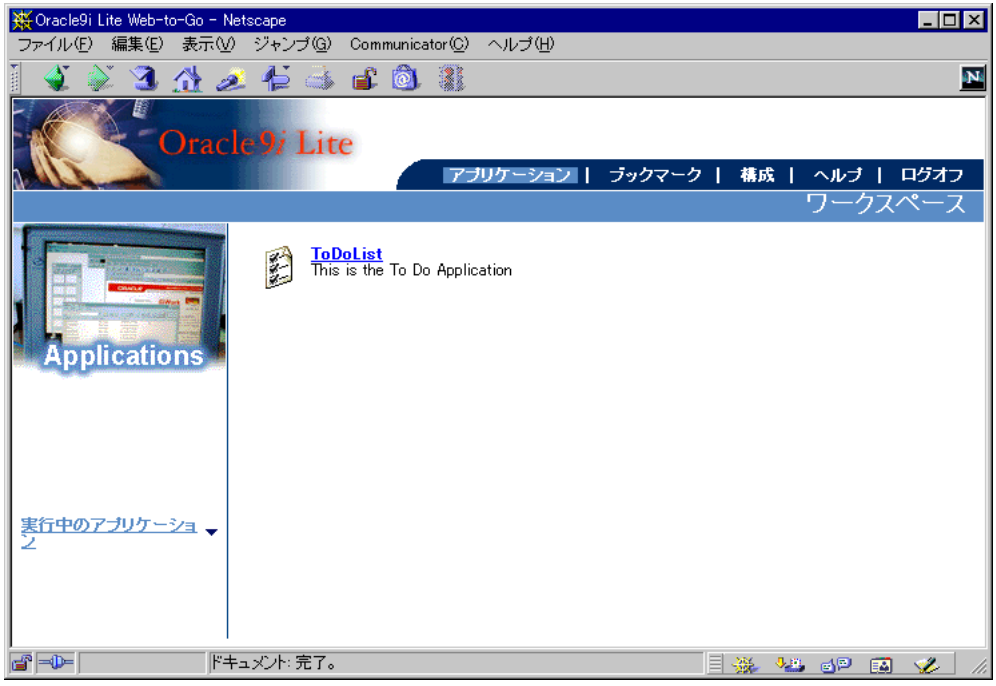
1. Web-to-Go にログオンします。これには、次の情報をログオン画面に入力し「ログオン」をクリックします。

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

2. これは Web-to-Go の 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。Web-to-Go の Mobile クライアントにより、アプリケーションとデータが自動的にダウンロードされます。

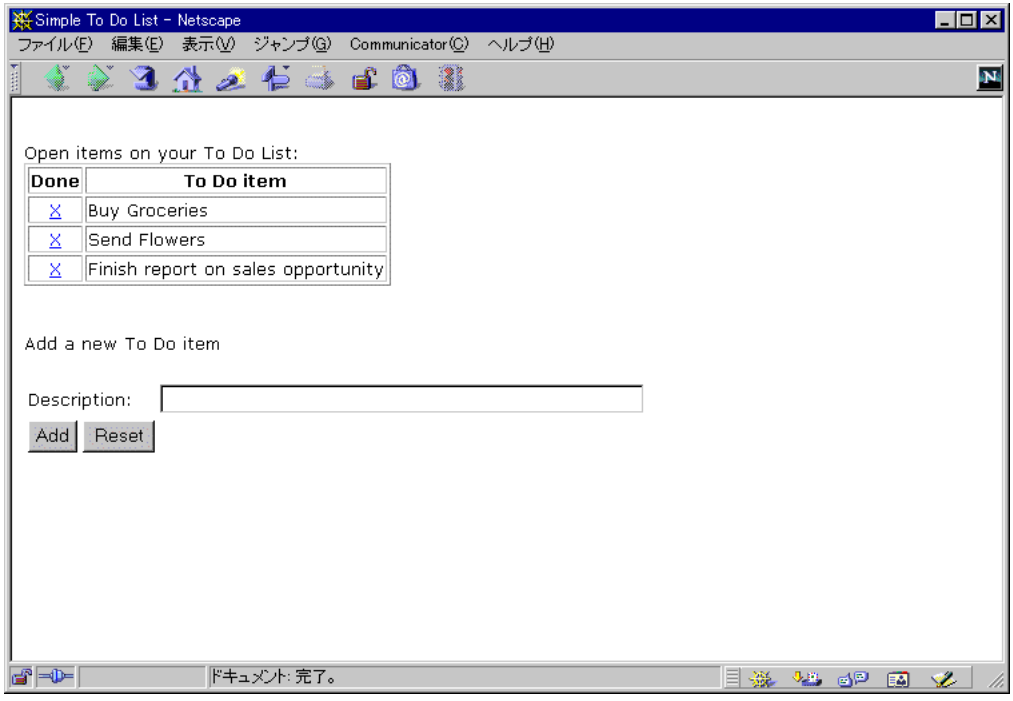
3. 「次へ」をクリックして、アプリケーションおよびデータのダウンロードを開始します。同期プロセスが完了すると、「To Do List」アプリケーションのアイコンのみがあるワークスペースが表示されます。

図 4-31 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. 「To Do List」アプリケーションのアイコンをクリックします。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 4-32「To Do List」アプリケーション



5. 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。
6. 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。

ステップ 3: Web-to-Go の Mobile クライアントの同期

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

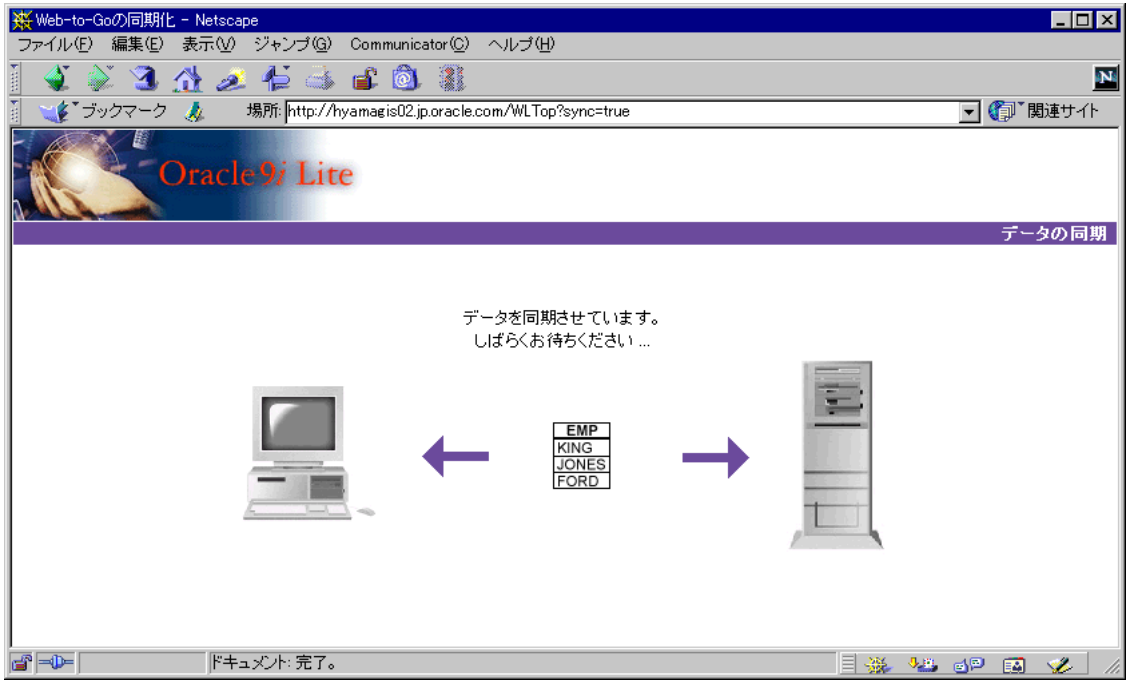
必須アクション

Web-to-Go の 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 の 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 4-33「Web-to-Go の同期化」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。

5

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。

- 概要
- 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
- パッケージ・ウィザードの使用

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。Oracle8*i* 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。

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。シーケンスのサポートにより、Web-to-Go の Mobile クライアントは、接続が切断されているときでも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。

Web-to-Go シーケンス

Web-to-Go では、クライアントがオフラインになったときにそのクライアント用のシーケンスを作成して、クライアントの一意の主キー値を決定します。Web-to-Go では、WINDOW シーケンスまたは LEAPFROG 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。

WINDOW シーケンス

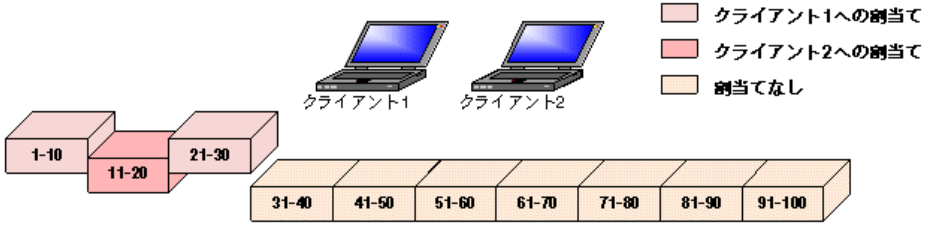
WINDOW シーケンスは、各クライアントに一意の値範囲を割り当てます。他のクライアントと値の範囲は重複しません。クライアントがシーケンスの範囲内の値をすべて使用すると、Web-to-Go は新しい一意の値範囲を持つシーケンスを再作成します。Web-to-Go がシーケンスを再作成するのは、クライアントが接続を切断しオフライン・モードになったときのみです。

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。クライアントがオフライン・モードに切り替わったと

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。

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つのクライアントへのWINDOWシーケンスの割当て

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。

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

シーケンス・パラメータ	定義
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 を超えると、使用可能な範囲が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接続を切断してオフライン・モードになったときに、シーケンスを再び作成します。Web-to-Go では、これを行うために次の SQL 文を実行します。

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

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。

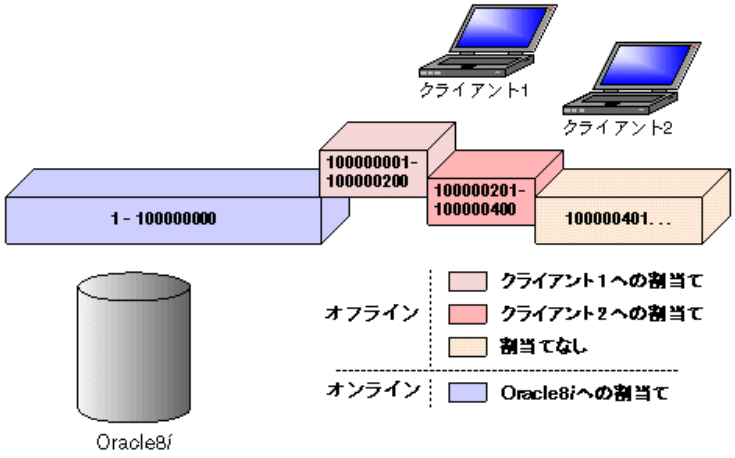
大きな WINDOW シーケンス

Oracle8*i* シーケンス用に 1 ～ 100000000 の大きな範囲を確保するには、Oracle8*i* サーバーで次の 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少なくとも 1 増分値のみ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、Oracle8*i* 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。



Oracle8*i*への大きなシーケンス番号を持つ
WINDOW シーケンスの割当て

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

偶数と奇数の WINDOW シーケンス

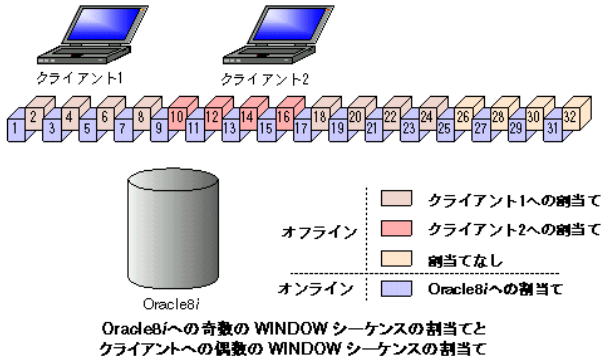
オンライン・モードとオフライン・モードのそれぞれに奇数または偶数のシーケンス値を割り当て、同じ増分値を指定すると、オンライン・モードとオフライン・モードに一意のシーケンス値を使用できます。次の手順により、オンライン・モード用に奇数のシーケンス値を確保し、オフライン・モードには偶数のシーケンス値の範囲を確保します。

Oracle8*i* 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



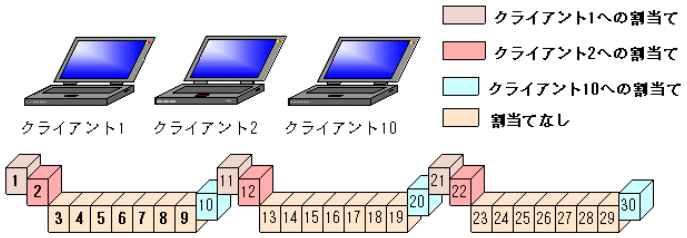
この方法を使用すると、オンライン用シーケンスの範囲は、Oracle シーケンス番号の上限（約 10^{26}）によってのみ制限されます。オフライン・クライアントの場合は、WINDOW_SIZE パラメータおよび THRESHOLD パラメータで増分ステップは考慮されていません。オフライン用の手順では、接続が切断されたクライアントに可能なシーケンス値は 100（200/2）個しかなく、これではすぐにシーケンスを再作成する必要が生じます。

LEAPFROG シーケンス

LEAPFROG シーケンスは、各クライアントに一意の一連の値範囲を割り当てます。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。シーケンス・スロットが不足しないように、LEAPFROG シーケンスの増分値はクライアント数の概算合計よりもかなり大きい値に設定するようにします。たとえば、クライアントの概算合計が 500 の場合、シーケンス増分値には 1000 を設定します。

例

次の例では、LEAPFROG シーケンスが 3 つのクライアントに割り当てられます。



3つのクライアントへのLEAPFROGシーケンスの割当て

LEAPFROG シーケンスの作成

Web-to-Go では、シーケンス定義を基に LEAPFROG シーケンスをローカルにメンテナンスします。たとえば、Web-to-Go に次のような LEAPFROG 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	LEAPFROG
START_VALUE	2
INCREMENT	1000

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 2 INCREMENT BY 1000;
```

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 3 INCREMENT BY 1000;
```

Web-to-Go ではシーケンスの作成回数を追跡し、各クライアントに一意の初期値を割り当てます。シーケンス・オブジェクトは似たようなプロパティを指定して Oracle データベース上に作成する必要があります。たとえば、次のとおりです。

```
CREATE SEQUENCE audiodb_seq START WITH 1 INCREMENT BY 1000;
```

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ー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ー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

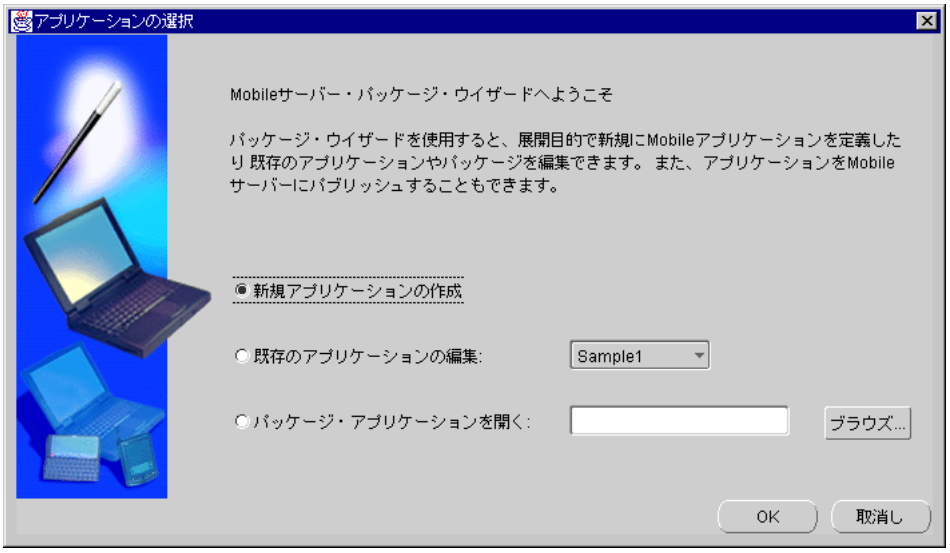
パッケージ・ウィザードを起動するには、DOS プロンプトで次のように入力します。

```
wtgpack
```

パッケージ・ウィザードが表示され、デフォルトで「ようこそ」パネルが表示されます。「ようこそ」パネルでは、次の機能を使用してパッケージ化されたアプリケーションの作成、編集またはオープンができます。

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。
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化されている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します。

図 5-1「ようこそ」ページ



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
```

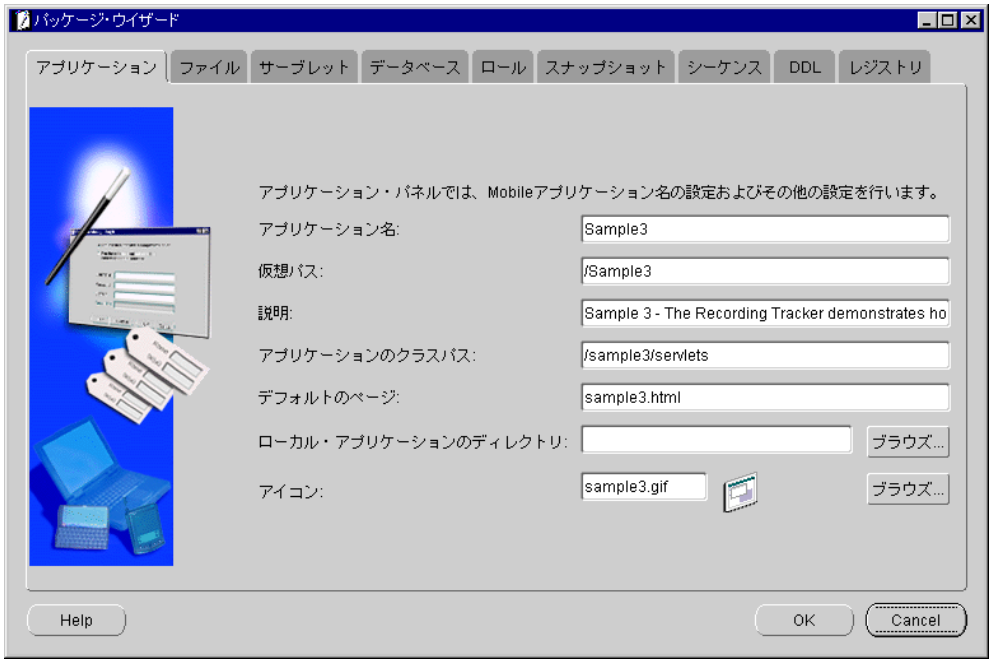
次の項では、パッケージ・ウィザードの各パネルの機能を説明します。

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、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。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
アプリケーション名	Web-to-Go アプリケーションの名前です。	○
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。また、これによりアプリケーションを一意に識別することもできます。	○
説明	Web-to-Go アプリケーションの簡単な説明です。	○
アプリケーションのクラスパス	セミコロン（;）で区切られたクラス・ディレクトリです。Web-to-Go は、ここに指定されている順番でアプリケーション・クラスを検索します。	
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.html の場合、「デフォルトのページ」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。	○
ローカル・アプリケーションのディレクトリ	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、入力するかまたは「ブラウズ」ボタンをクリックして選択します。	○
アイコン	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。	

図 5-2「アプリケーション」パネル

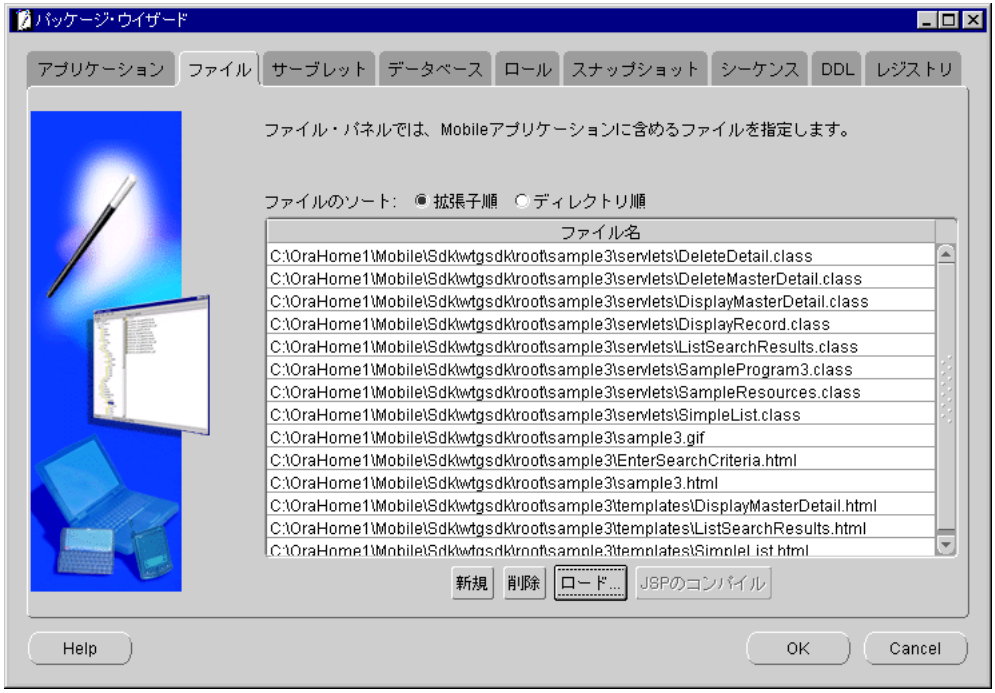


アプリケーション・ファイルのリスト表示

「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。	○

図 5-3「ファイル」パネル



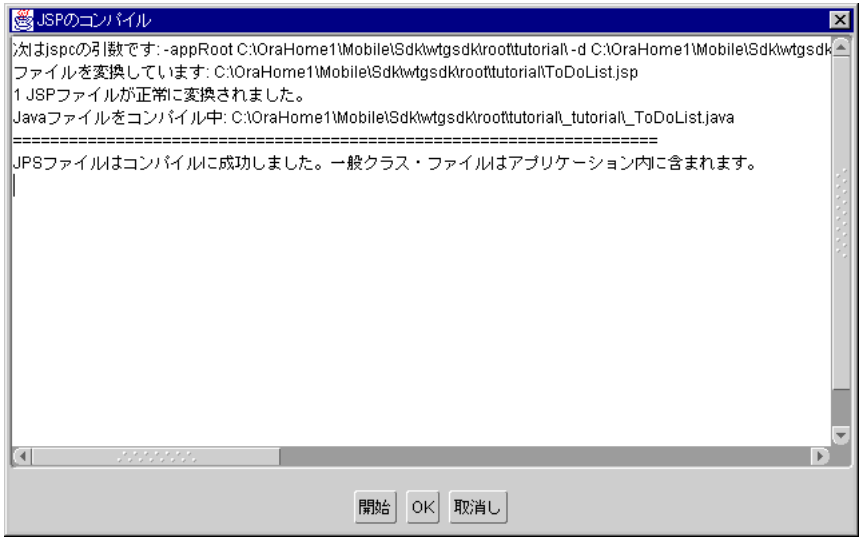
「ファイル」パネルにリストされているファイルはすべて、追加、削除、ロードまたはコンパイルできます。新規アプリケーションを作成する場合、「ファイル」パネルに進むと、ローカル・ディレクトリにリストされているすべてのファイルが、パッケージ・ウィザードにより自動的に分析およびロードされます。既存のアプリケーションを編集する場合は、「ロード」ボタンを使用して個々のアプリケーション・ファイルをロードできます。

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」パネルにリスト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 5-4「JSP のコンパイル」ダイアログ・ボックス



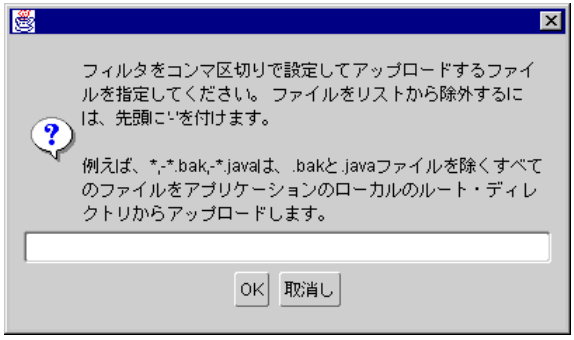
ソート

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。

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（カンマで区切られた）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
```

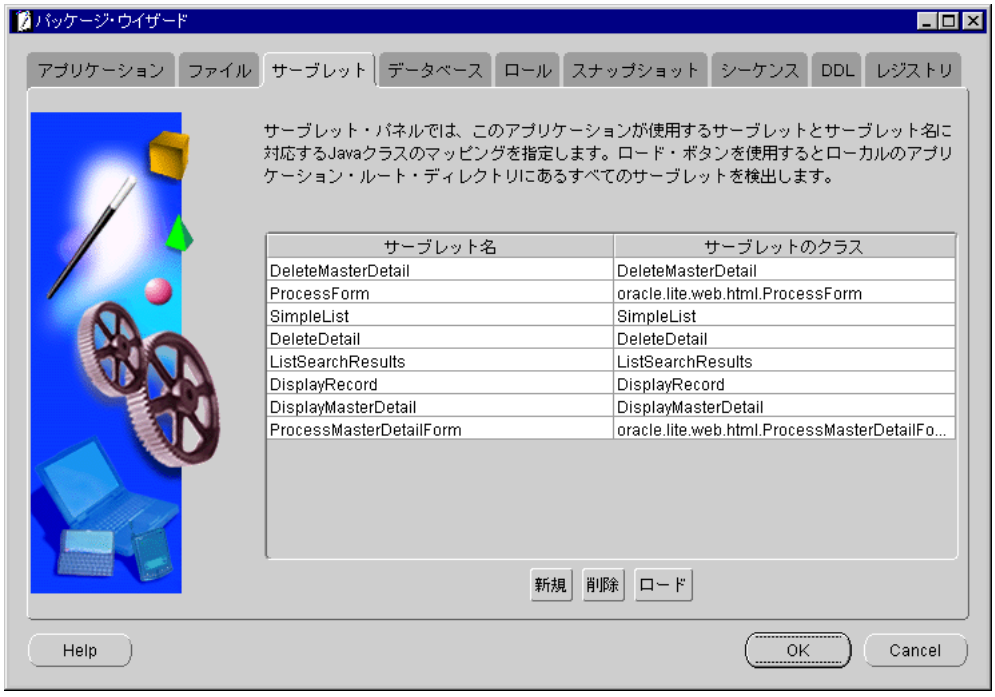


サーブレットの追加

パッケージ・ウィザードは、「ファイル」タブにあるサー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ーブレットは「サーブレット」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
サーブレット名	サーブレットの名前です。たとえば、`DeleteDetail` です。サーブレットは `application_virtualpath/` サーブレット名として参照します。	○
サーブレットのクラス	追加するサーブレットの完全修飾クラスです。	○

図 5-5「サーブレット」パネル



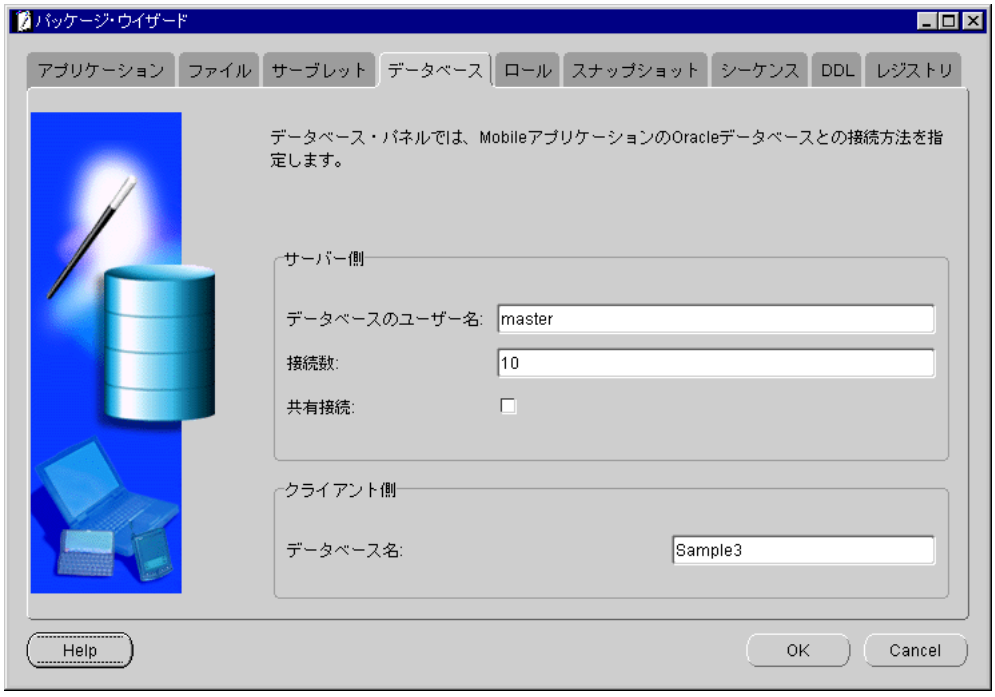
「サーブレット」パネルにリストされているサーブレットは、すべて追加、削除またはロードできます。新規アプリケーションを作成する場合、「ファイル」タブにリストされているファイルを基にすべてのサーブレットがパッケージ・ウィザードにより自動的にリストされます。既存アプリケーションを編集する場合は、「ロード」ボタンを使用して個々のサーブレットをロードできます。

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。	○
接続数	Web-to-Go アプリケーションから Oracle Server への同時接続数です。	○
共有接続	これを選択すると、複数のサーブレットで同一のデータベース接続を共有できます。複数のサー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ーブレットが接続されることを回避できます。ただし、あるサー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ー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 5-6「データベース」パネル

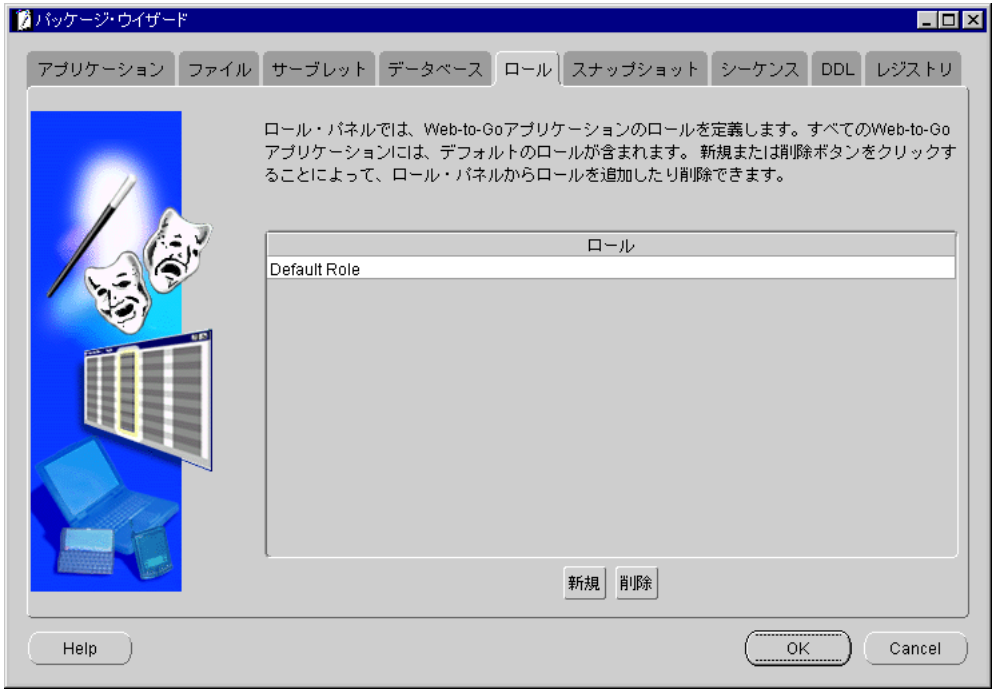


アプリケーション・ロールの定義

「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 Oracle8*i* 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 5-7「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。

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。

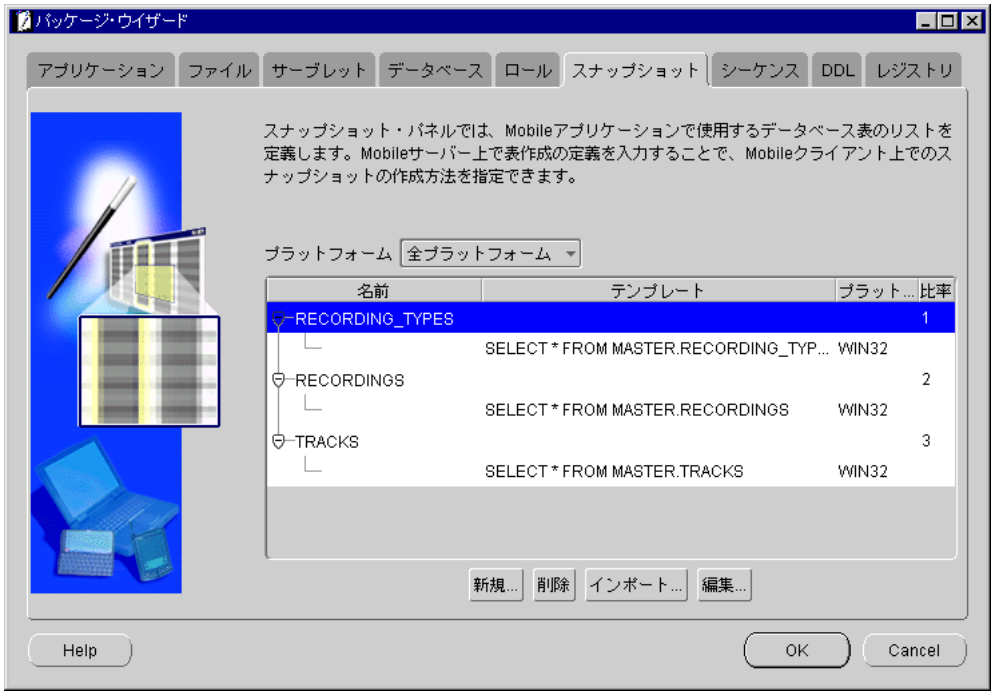
注意： 一度データベース接続を指定すると、パッケージ・ウィザードの残りのセッションで使用されます。Oracle8*i* と Oracle Lite データベースを切り替える必要があるが、すでに接続が確立されている場合は、パッケージ・ウィザード・アプリケーションを完全に終了して、再度 **wtgpack.exe** を実行します。

「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。**	○
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。	○
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。	

フィールド	説明	必須
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。 ■ Win32 ■ Palm ■ EPOC ■ 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表示されます。	

図 5-8「スナップショット」パネル



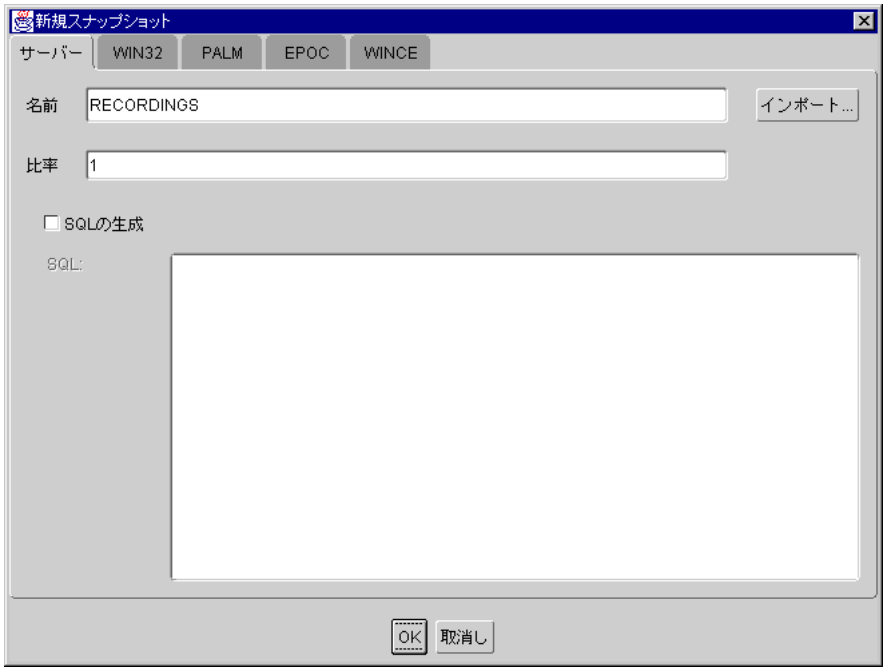
「スナップショット」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、「スナップショット」パネルにスナップショットを追加または削除できます。「インポート」または「編集」ボタンをクリックして、スナップショットをインポートまたは編集することもできます。

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。

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。「サーバー」タブをクリックすると、次の「サーバー」パネルが表示されます。

図 5-9「新規スナップショット」ー「サーバー」パネル



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。「名前」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。

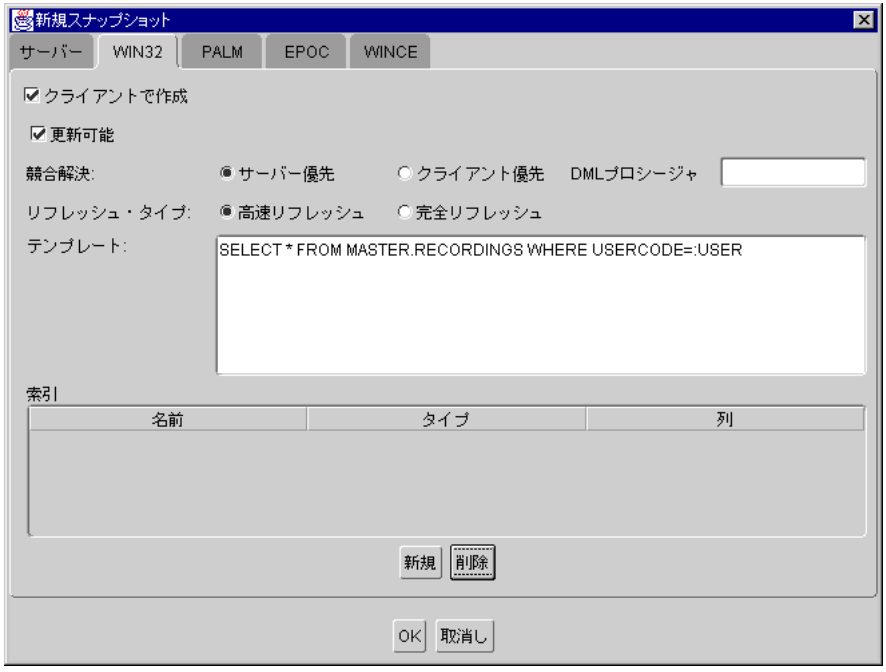
「比率」の説明は、「レプリケーション用スナップショットの定義」を参照してください。

「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 の 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 5-10「新規スナップショット」ー「クライアント」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 の 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。

スナップショットのインポート

Oracle8*i* データベース、あるいは 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 Oracle8*i* 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。

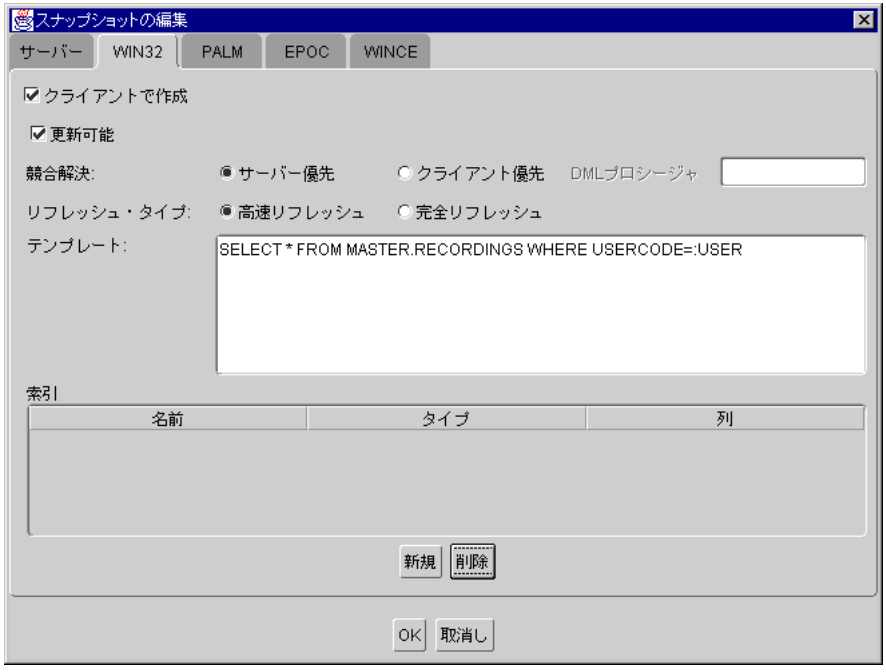
注意： Oracle8*i* データベースのデータベース URL を入力するときは、次の書式を使用します。`jdbc:oracle:oci8:@webtogo.world`。Oracle Lite の場合は、`jdbc:polite:webtogo` を使用します。

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。

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 5-11「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、「表の編集」ウィンドウ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 の 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。

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。	○
タイプ	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。 **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。 **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。	○

フィールド	説明	必須
初期値	Web-to-Go の 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 の 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	Oracle8*i* 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 の 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	Oracle8*i* 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	Oracle8*i* 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	Oracle8*i* 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 5-12「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。

シーケンスのインポート

Oracle8*i* 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。「シーケンスの編集」ウィンドウが表示されます。

シーケンスを編集するには、「シーケンスの編集」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、Oracle8*i* 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	Oracle8*i* データベース上のシーケンスの初期値です。
増分値	Oracle8*i* 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	Oracle8*i* 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	Oracle8*i* 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 の 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 の 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 の 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 の 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 の 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。

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 5-13「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。	○
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。	○

ビューや索引の定義のインポート

Oracle8*i* データベースからビューや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。

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 の名前と値の一意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。	○
値	名前に対応する値です。	○

図 5-14「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。

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「アプリケーション定義が完了しました」ウィンドウが表示されます。

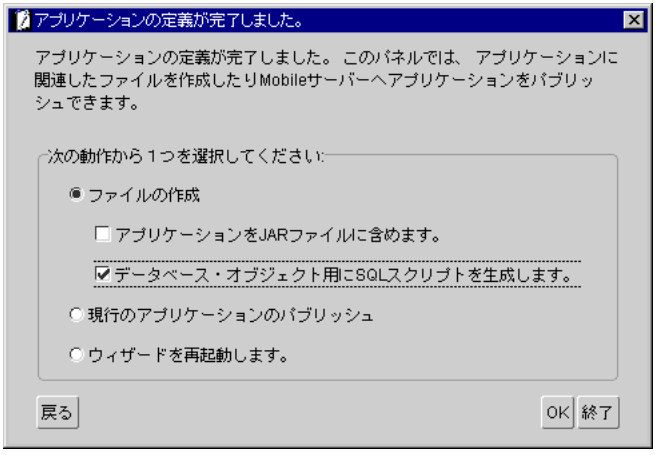
- ファイルの作成
- 現行のアプリケーションのパブリッシュ
- ウィザードの再起動

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 ファイルしかパブリッシュできません。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。

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・ディレクトリの下の 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。



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。「アプリケーションをパブリッシュします。」ウィンドウが表示されます。

図 5-15「アプリケーションをパブリッシュします。」ダイアログ・ボックス

「アプリケーションをパブリッシュします。」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	説明
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	`/todo`
アプリケーションをパブリックにする	

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。

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。

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。

A

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。

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。

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
```

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¥wtgsdk¥src¥sdkdemos.bat
```

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。

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。

Sample1 - Hello World

Sample1 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。

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。

Mobile サーバーの場合、次の場所になります。

```
Oracle_home¥Mobile¥Server¥samples¥sample1¥src
```

Mobile Development Kit（Web-to-Go 用）の場合、次の場所になります。

```
Oracle_home¥Mobile¥Sdk¥wtgsdk¥src¥sample1¥servlets
```

アプリケーション・ファイル

Sample1 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` サーブレットのソース・コードです。

Sample3 - Recording Tracker

Sample3 の Recording Tracker は、サー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。

Sample3 の使用方法

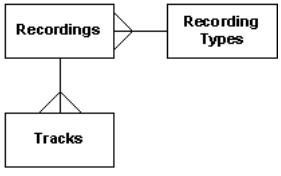
オフラインになると、Web-to-Go はデータのコピーを保持するスナップショットをローカル・クライアント上に自動的に作成します。スナップショット定義に副問合せを追加すると、スナップショットにレプリケートする行を制御できます。これにより、Web-to-Go は各ユーザー用の特定の行をローカル・クライアントに同期できます。Sample3 では、John と Jack は同じデータにアクセスできます。Jane は専用のデータを表示でき、他のユーザーはこのデータにはアクセスできません。スナップショットの副問合せの設定の詳細は、『Oracle9*i* Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

Sample3 のデータベース表

Sample3 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。



Sample3 のサーブレット

Sample3 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 を継承しています。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。

Sample3 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。

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）	*Oracle_Home*¥Mobile¥Sdk¥wtgsdk¥src¥sample3¥

アプリケーション・ファイル

Sample3 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 Sample3 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。
sample3.sql	sample3 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	sample3 のデータベース表を作成する SQL スクリプトです。
insert.sql	sample3 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。

ファイル	説明
drop.sql	sample3 のデータベース表を削除する SQL スクリプトです。
SampleProgram3.java	Sample3 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。

Sample4 - Hello Applet

Sample4 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。

Sample4 のサーブレット

Sample4 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。

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）	*Oracle_Home*¥Mobile¥Sdk¥wtgsdk¥src¥sample4¥ *Oracle_Home*¥Mobile¥Sdk¥wtgsdk¥root¥sample4¥

アプリケーション・ファイル

Sample4 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 Sample4 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。

Sample5 - Company Performance Graph

Sample5 の Company Performance Graph（会社業績グラフ）アプレットは、アプレット内で JDBC を使用する方法を示します。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、`oracle.lite.web.applet.AppletProxy` クラスは Oracle8*i* 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、このクラスは Oracle Lite に対する接続を返します。

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¥sample5¥src
Mobile Development Kit（Web-to-Go 用）	*Oracle_Home*¥Mobile¥Sdk¥wtgsdk¥src¥sample5¥src

アプリケーション・ファイル

Sample5 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
Sample5.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。
graph.ahtml	アプリケーションの開始ページです。
GraphApplet.java	アプレットの Java ソース・コードです。
LW_pfjbean.jar	グラフを生成するために使用する図形ライブラリです。
swingall.jar	JavaSoft の Swing ライブラリです。
sample5.sql	sample5 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	sample5 のデータベース表を作成する SQL スクリプトです。
insert.sql	sample5 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。

Sample6 - Image Gallery

Sample6 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。

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）	*Oracle_Home*¥Mobile¥Sdk¥wtgsdk¥src¥sample6¥

アプリケーション・ファイル

Sample6 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` サーブレットのソース・コードです。
GetImage.java	`GetImage` サーブレットのソース・コードです。
Upload.java	`Upload` サーブレットのソース・コードです。
ImageList.java	`ImageList` サーブレットのソース・コードです。
ViewImage.java	`ViewImage` サーブレットのソース・コードです。
RawImage.java	`RawImage` サーブレットのソース・コードです。
ImageList.html	`ImageList` サーブレットにより使用される HTML テンプレートです。

ファイル	説明
ViewImage.html	`ViewList` サーブレットにより使用される HTML テンプレートです。
sample6.sql	sample6 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	sample6 のデータベース表を作成する SQL スクリプトです。
drop.sql	sample6 のデータベース表を削除する SQL スクリプトです。

B

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。

- oracle.html パッケージの使用方法
- oracle.lite.web.html パッケージの使用方法

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。

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();
```

<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");
  ```

<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();
  ```

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、「HTML 要素の Java クラスへのマッピング」を参照してください。

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。

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。

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 <EM> 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>

</BODY>

</HTML>

フォーム

次のコード・フラグメントは、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>
```

リスト

次のコード・フラグメントは、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>
```

表

次のコード・フラグメントは、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>
```

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。

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

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
<P>	Paragraph または SimpleItem
<A href>	Link
<A name>	Anchor
<IMG>	Image
<IMG>（サーバー側のイメージ・マップ）	Image

HTML	Java クラス
<IMG>（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)

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
<UL>	UnOrderedList
<OL>	OrderedList
<DL>	DefinitionList
<DIR>	DirectoryList
<MENU>	MenuList
<LI>	ListItem
<DT>	DefinitionList.addDef
<DD>	DefinitionList.addDef

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

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>	Font
<EM>	setAttr(ATTR_PHRASE_EMPHASIS) または setEmphasis()
<STRONG>	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()
<B>	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()

HTML	Java クラス
<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()

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	File
<INPUT type=image	Image

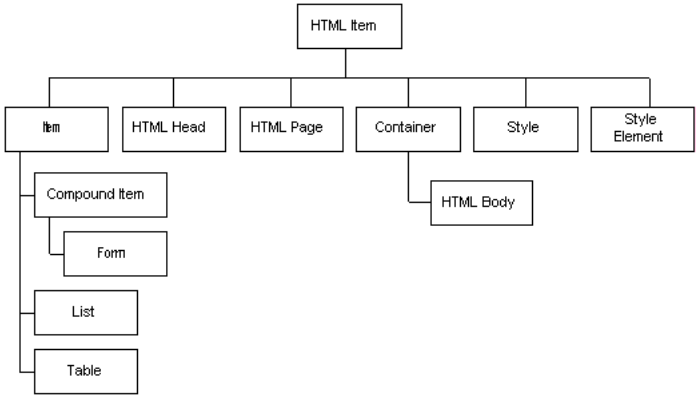
その他の要素

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

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。



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。

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。

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継承します。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継承します。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。

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）を容易に実装

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薦めします。

- 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容易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、IVAlign と ITableRules があります。

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", IVAlign.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。

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リストする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 サーブレットにより読み取られ、<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 サーブレットそのもの。

```
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);
            }
        }
```

oracle.lite.web.html パッケージの使用方法

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 クラスを拡張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。

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、『Oracle9*i* Lite Web-to-Go API リファレンス』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。

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。データを読取り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);
  }
}
```

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

ProcessForm を使用したデータの処理

ProcessForm サーブレットは、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。このサーブレットは、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 サーブレットは、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 サー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。

DeleteRecords サー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ー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。

ユーザー定義のサーブレット 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
getCommands()	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。
getCommands()	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、『Oracle9*i* Lite Web-to-Go API リファレンス』を参照してください。

例

EMP 表の行を削除するサー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" );
  }
}
```

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 サー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 サー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。このサーブレットは正しい 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 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。

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 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です。

Java Web Server Development Kit

Java Web Server Development Kit 1.0.1 は、JavaServer ページ（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。

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。

LEAPFROG シーケンス（Leapfrog Sequence）

Web-to-Go がサポートする 2 つのシーケンスのうちの 1 つで、オフライン・モードの Web-to-Go の 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。LEAPFROG 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。

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 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 サーブレット、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データ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、Oracle8*i* 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持続リソース・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でのデータベース・アクセスを可能にします。Web-to-Go の 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、ローカルな 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 の 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。

Oracle8*i*

Oracle8*i* 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 の 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 Oracle8*i* 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 の Mobile クライアント、Mobile サーバーおよび Oracle8*i* 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 の Mobile クライアント（Mobile Client for Web-to-Go）

Web-to-Go の 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。この層には、Mobile サーバーと Oracle Lite データベースが含まれます。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・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。オンラインに戻ると、Web-to-Go はデータの変更を Oracle8*i* 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 つのシーケンスのうちの 1 つで、オフライン・モードの Web-to-Go の 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。UNIQUE KEY 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、PRIMARY KEY 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。

位置づけ DELETE（Positioned DELETE）

位置づけ 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
```

位置づけ UPDATE（Positioned UPDATE）

位置づけ 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 の 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 の 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・ディテール関連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容易になります。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 の 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、UNIQUE KEY 制約や PRIMARY KEY 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・ディテール関連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。

シーケンス（Sequence）

順次数を生成するスキーマ・オブジェクト。シーケンスを作成した後は、これを使用してトランザクション処理用の一意の順次数を生成できます。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シーケンスが即時増分されます。
「WINDOW シーケンス」および「LEAPFROG シーケンス」も参照。

シノニム（Synonym）

表、ビュー、シーケンス、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。

PRIMARY KEY 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。

スキーマ（Schema）

表、ビュー、索引、シーケンスなどを含む、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 の 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 の 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、シーケンス、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 が Web-to-Go の Mobile クライアントと Oracle8*i* の間でデータをレプリケートするために使用するプロセス。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。オンラインに戻ると、Web-to-Go はデータの変更を Oracle8*i* 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、ADD、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・ウィザードでは、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とパッケージ化ができます。

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言います。

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 の 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 Oracle8*i* 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。レプリケーションの競合は、データを正しくサブセット化することで回避できます。パッケージ・ウィザードにより、開発者は競合の処理方法に関するルールを指定できます。

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

O

S

W

X

あ

お

こ

さ

し

す

て

と

な

は

ひ

ふ

ゆ

れ

ろ

わ