# BEA WebLogic Server™

## BEA WebLogic Express™

# Wireless Application 開発 プログラマーズ ガイド

BEA WebLogic Server バージョン 6.1
マニュアルの日付：2003 年 4 月 24 日

## 著作権



## 限定的権利条項



## 商標または登録商標



**WebLogic Server Wireless Application 開発プログラマーズ ガイド**

| パート番号 | マニュアルの日付 | ソフトウェアのバージョン |
|---|---|---|
| なし | 2001 年 9 月 19 日 | WebLogic Server バージョン 6.1 |

# 目次

## このマニュアルの内容



## 1. はじめに



## 2. WebLogic Server での WAP の使い方



## 3. WebLogic Server での i-Mode の使い方

## 4. 無線加入者に対応した Web アプリケーションの記述

# このマニュアルの内容

このマニュアルでは、BEA WebLogic Server™ プラットフォームを使用して、従来のデスクトップ ブラウザだけでなく、さまざまなタイプの無線デバイスと接続できる Web アプリケーションを設計および記述する方法について説明します。

このマニュアルの内容は以下のとおりです。

- 第 1 章「はじめに」では、WebLogic Server を使用して Web アプリケーションの対象範囲を無線加入者に拡張する前に、知っておく必要のある基本的な情報について説明します。
- 第 2 章「WebLogic Server での WAP の使い方」では、Wireless Application Protocol（WAP）アプリケーション環境に適したコンテンツを提供する方法、および WAP ゲートウェイと WebLogic Server をコンフィグレーションして使用する方法について説明します。
- 第 3 章「WebLogic Server での i-Mode の使い方」では、i-Mode アプリケーション環境に適したコンテンツを提供する方法、および i-Mode ゲートウェイと WebLogic Server をコンフィグレーションして使用する方法について説明します。
- 第 4 章「無線加入者に対応した Web アプリケーションの記述」では、WebLogic Server を使用して Web アプリケーションの対象範囲を無線加入者に拡張する方法について、例とともに示します。

# 対象読者

このマニュアルは、WebLogic Server プラットフォームで動作するトランザクション対応 Java アプリケーションの構築に関心があるアプリケーション開発者を対象としています。WebLogic Server、Java™ 2、Enterprise Edition（J2EE）プログラミング、および無線 Web 技術に読者が精通していることを前提として書かれています。

# e-docs Web サイト

BEA 製品のドキュメントは、BEA の Web サイトで入手できます。BEA のホーム ページで [ 製品のドキュメント ] をクリックします。

# このマニュアルの印刷方法

Web ブラウザの [ ファイル | 印刷 ] オプションを使用すると、Web ブラウザからこのマニュアルを一度に 1 章ずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。PDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Server ドキュメントのホーム ページを開き、[ ドキュメントのダウンロード ] をクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader は Adobe の Web サイト（`http://www.adobe.co.jp`）で無料で入手できます。

# 関連情報

BEA の Web サイトでは、WebLogic Server の全マニュアルを提供しています。WebLogic Server を使用してアプリケーション サービスを記述するときに参考となる WebLogic Server の他のマニュアルは、次のとおりです。

- 『BEA WebLogic Server の紹介』
- 『管理者ガイド』
- 『WebLogic XML プログラミング ガイド』

# サポート情報

BEA のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、docsupport-jp@bea.com 宛に電子メールでお寄せください。寄せられた意見については、ドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、ご使用のソフトウェア名とバージョン名、およびマニュアルのタイトルと作成日付をお書き添えください。本バージョンの BEA WebLogic Server について不明な点がある場合、または BEA WebLogic Server のインストールおよび動作に問題がある場合は、BEA WebSUPPORT（http://www.bea.com）を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

# 表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

| 表記法 | 適用 |
|---|---|
| 〔Ctrl〕+〔Tab〕 | 同時に押すキーを示す。 |
| 斜体 | 強調または本のタイトルを示す。 |
| 等幅テキスト | コード サンプル、コマンドとそのオプション、Java クラス、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。<br>例 :<br>`import java.util.Enumeration;`<br>`chmod u+w *`<br>`config/examples/applications`<br>`.java`<br>`config.xml`<br>`float` |
| 斜体の等幅テキスト | コード内の変数を示す。<br>例 :<br>`String *CustomerName*;` |
| すべて大文字のテキスト | デバイス名、環境変数、および論理演算子を示す。<br>例 :<br>LPT1<br>BEA_HOME<br>OR |
| { } | 構文内の複数の選択肢を示す。 |

| 表記法 | 適用 |
| --- | --- |
| [ ] | 構文内の任意指定の項目を示す。<br>例 :<br>`java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]` |
| \| | 構文の中で相互に排他的な選択肢を区切る。<br>例 :<br>`java weblogic.deploy [list\|deploy\|undeploy\|update] password {application} {source}` |
| ... | コマンドラインで以下のいずれかを示す。<br>■ 引数を複数回繰り返すことができる。<br>■ 任意指定の引数が省略されている。<br>■ パラメータや値などの情報を追加入力できる。 |
| .<br>.<br>. | コード サンプルまたは構文で項目が省略されていることを示す。 |

# 1 はじめに

BEA WebLogic Server™ プラットフォームを使用して Web アプリケーションの対象範囲を無線加入者に拡張する前に、以下の内容に目を通してください。

- 概要
- 無線データ プロトコル
- WAP プロトコルとは
- i-Mode プロトコルとは
- その他の無線データ プロトコル
- その他の無線マークアップ言語
- 無線データ プロトコルと無線マークアップ言語の進歩
- 補足情報

## 概要

無線インターネット対応の加入者基盤は拡大を続けています。電気通信企業および無線サービス プロバイダは、無線加入者に対してよりパーソナライズされた体験を提供しながらも、利益を生み出す新しいインターネット サービスを迅速に開発およびデプロイする方法を探しています。

無線加入者は、デスクトップ インターネット ユーザやラップトップ インターネット ユーザとは異なる欲求を持っています。このため、インターネット サービスは、さまざまな種類の無線デバイスを使用する加入者に合わせて最適な体験を提供するよう設計しなければなりません。このような方法でインターネット サービスを開発するには、無線環境に固有の技術的な問題についての深い理解が必要です。

# 無線データ プロトコル

Hypertext Transfer Protocol（HTTP）、Transport-Layer Security（TLS）、Transmission Control Protocol（TCP）、Hypertext Markup Language（HTML）などのインターネット技術を利用して接続を設定および解除し、無線ネットワークで Web コンテンツを転送するのは、いくつかの理由により効率的ではありません。たとえば、次のような理由が挙げられます。

- HTTP と TCP は、無線ネットワークに関連する断続的なカバレッジ、長いレイテンシ、および制限された共有バンド幅に対応するよう最適化されていません。
- HTTP は、そのヘッダとコマンドを、より効率的な圧縮バイナリ形式ではなくテキスト形式で送信しますが、これは無線ネットワークの低い帯域幅には適していません。
- HTTP セッションを設定および破棄する場合、無線デバイス（クライアント）とサーバ間で多くのメッセージのやりとりが必要になります。
- デスクトップ ブラウザ向けの HTML Web コンテンツは、携帯電話、個人用携帯型情報端末（PDA）、ページャ、双方向ラジオ、およびスマートフォンの小型画面には有効に表示できません。
- キーボードとマウスがないので、画面内および画面間の操作が面倒です。操作は、数値キーバッドを押すか、またはペンで入力することによって行います。

これらの製品を使用する無線サービスは、多くの場合、低速、高コストで、難しい操作を強いられます。このため、無線データ プロトコルという、Web ベースのデータを無線ネットワークで転送するための特別なプロトコルが作成されました。Wireless Application Protocol（WAP）や i-Mode などの無線データ プロトコルは、無線環境に固有の制約を満たすよう設計されています。これらのプロトコルは、データの圧縮率の高いバイナリ転送を使用し、長いレイテンシと低～中程度の帯域幅に合わせて最適化されています。これらのプロトコルは、プロトコル データ ユニット（PDU）連結と遅延確認応答をサポートして、送信メッセージの数を減らします。また、これらのプロトコルに関連付けられているマークアップ言語により、小さい画面を有効に使用し、画面内および画面間を簡単に移動できるようになります。

無線データ プロトコルに関連付けられているマークアップ言語を読み取るには、無線デバイスにマイクロブラウザが必要です。マイクロブラウザは、このマークアップ言語を解釈するための特別なクライアント ソフトウェアです。たとえば、WAP 対応デバイスには Wireless Markup Language（WML）マイクロブラウザが、i-Mode 対応デバイスには compact HTML（cHTML）マイクロブラウザがそれぞれ搭載されています。

# WAP プロトコルとは

Wireless Application Protocol（WAP）仕様は、無線データ プロトコル（無線デバイス上のマイクロブラウザが無線ネットワークにインストールされている WAP ゲートウェイと通信するための標準化された手段）とマークアップ言語の Wireless Markup Language（WML）から構成されています。WML は、WAP 対応デバイス用のコンテンツとユーザ インタフェースを指定するための Extensible Markup Language（XML）です。

WAP セッションは、断続的なカバレッジに対応し、Cellular Digital Packet Data（CDPD）、Code Division Multiple Access（CDMA）、Global System for Mobiles（GSM）、Time Division Multiple Access（TDMA）、General Packet Radio Service（GPRS）などのさまざまな無線ベアラ ネットワーク上で動作します。ヨーロッパおよび米国のほとんどの WAP サービスは回路切り替え（ダイアルアップ）ですが、WAP はパケット切り替えネットワーク上でも動作します。

# i-Mode プロトコルとは

i-Mode 仕様は、無線データ プロトコル（無線デバイス上のマイクロブラウザが無線ネットワークにインストールされている i-Mode ゲートウェイと通信するための標準化された手段）とマークアップ言語の compact HTML（cHTML）から構成されています。cHTML は HTML のサブセットで、i-Mode 対応デバイス用のコンテンツとユーザ インタフェースを指定するために使用されます。

**注意：** 出版物によって、i-Mode は「i-Mode」、「I-Mode」、「I-mode」、「i-mode」、「imode」など、さまざまに表記されています。このマニュアルでは、「i-Mode」という表記を採用しています。

WAP はオープンでグローバルな仕様ですが、i-Mode は現在のところ日本の NTT DoCoMo によって開発およびデプロイされている独自の閉じられた仕様です。i-Mode の唯一の実装は NTT DoCoMo のモバイル インターネット アクセス システムですが、ヨーロッパと米国の複数の通信企業が i-Mode への関心を表明しています。

現時点では、i-Mode は NTT DoCoMo の PDC-P モバイル音声システムのみで動作しますが、i-Mode はどのような基盤無線ベアラ ネットワーク上でも同様に動作します。

# その他の無線データ プロトコル

その他の無線データ プロトコルには、以下のものがあります。

- Phone.com（前 Unwired Planet、現 Openwave Systems Inc.）によって開発された Handheld Device Transport Protocol（HDTP）
- OmniSky Corporation によって開発された OmniSky
- Mobitex Operators Association によって開発された Mobitex

また、無線業界は、Voice over IP（VoIP）技術にも関心を寄せています。この技術は、共通の IP インフラストラクチャ上で音声およびデータ トラフィックを移動するためのものです。無線業界は、VoIP および無線データ プロトコル技術を統合して、音声および Web データを同じ無線チャネルで伝送できるようにする方法を探しています。

# その他の無線マークアップ言語

その他の無線マークアップ言語には、以下のものがあります。

- Phone.com（前 Unwired Planet、現 Openwave Systems Inc.）によって開発された Handheld Device Markup Language（HDML）
- Palm, Inc. によって開発された Web Clipping
- World Wide Web Consortium（W3C）によって開発された Extensible HTML（XHTML）（Basic）
- VoiceXML Forum（AT&T、IBM、Lucent、および Motorola によって設立された業界組織）によって開発された VoiceXML

**注意：** HDTP は、HDML タグを含んだ Web コンテンツを伝送します。HDTP と HDML は、WAP と WML の開発に強い影響を与えました。

# 無線データ プロトコルと無線マークアップ言語の進歩

現在存在するさまざまな無線データ プロトコルと無線マークアップ言語は、将来単一の無線データ プロトコルと無線マークアップ言語へと進化していくものと予想されています。その根拠として、WAP プロトコルの採用企業は WAP-NG という新世代のプロトコルに移行しようとしており、AT&T と NTT DoCoMo はどちらも WAP-NG の採用を決定しています。さらに、WAP と i-Mode は、共通のマークアップ言語である XHTML（Basic）に移行しようとしています。

# 補足情報

以下に、無線データ プロトコル、無線マークアップ言語、無線デバイスの製造業者、および無線標準と無線仕様に関する補足情報を参照できる Web サイトを示します（カテゴリ順に列挙）。

## 関連する WebLogic 無線情報

BEA によるホワイト ペーパー「Beyond the Wire — Developing Software for Many Devices」

BEA 無線ニュースグループ

BEA 無線 FAQ

## 一般的な無線情報

The Wireless FAQ

FierceWireless

MBizCentral

Unstrung

WirelessDevNet

## 無線デバイス製造業者

OmniSky

Palm

PocketPC

RIM BlackBerry

Symbian Limited

## 無線標準および仕様

XHTML Basic W3C 勧告

VoiceXML Forum の仕様

The Short Message Peer to Peer（SMPP）Forum

SMS プロトコル ― SMPP 仕様

SMS プロトコル ― UCP/EMI 仕様

SMS プロトコル ― CIMD2 仕様

SNPP のページ

# 2 WebLogic Server での WAP の使い方

以下の節では、Wireless Application Protocol（WAP）に対応したコンテンツを提供する方法、および WebLogic Server をコンフィグレーションして WAP ゲートウェイと一緒に使用する方法について説明します。

- 概要
- WAP アプリケーション環境
- WAP ゲートウェイ
- その他の情報源

## 概要

Wireless Application Protocol（WAP）は、携帯電話などの無線デジタル デバイス向けのインターネットおよび Web ベース サービスを開発するために WAP Forum によって開発およびデプロイされた、オープンでグローバルな仕様です。この設立メンバーには、Ericsson、Motorola、Nokia、Phone.com（前 Unwired Planet、現 Openwave Systems Inc）という主要な無線ベンダが含まれています。

WAP 仕様は、無線ネットワークの限界（狭いバンド幅、長いレイテンシ、そして予測不能な可用性と安定性）と無線デバイスの限界（CPU、メモリ、バッテリ寿命が限られていること、ユーザ インタフェースがシンプルであること）を解決しようとするものです。i-Mode 仕様は、無線通信の本質的な要素を 2 つ定義しています。これらは、無線プロトコルとアプリケーション環境です。

WAP ゲートウェイは、無線ネットワーク上のクライアントとインターネット上のアプリケーション サーバにホストされているアプリケーションを接続するものです。WAP ゲートウェイは、無線クライアントからアプリケーション サーバ

へ要求をルーティングすることにより、電気通信とコンピュータ ネットワークの間にブリッジを構築します。これは、いずれか一方のネットワークにだけ必要ですが、物理的にはどちらのネットワークにも置くことができます。

# WAP アプリケーション環境

WAP アプリケーション環境は、ナローバンド デバイス用の無線アプリケーションのための、ネットワーク中立なフレームワークを定義します。WAP アプリケーション環境の主要な構成要素は、Wireless Markup Language（WML）と WMLScript（WMLS）の 2 つです。

## WML

WAP アプリケーション用の WML は、TCP/IP アプリケーション用の HTML に似ています。WML は、WAP 対応デバイスのマイクロブラウザとのインタフェース用として特別に設計された、XML ベースの言語です。Wireless Markup Language 仕様では、WML ドキュメントのタグと構造が定義されています。

1 つの WML ドキュメントは、1 枚または複数枚のカードの集まりです。集まりの中の各カードは、明確に定義された 1 つのインタラクション単位と見なされます。一般に、1 枚のカードは無線デバイスの 1 つの画面に収まる十分な情報を保持します。無線クライアントに WML ドキュメントを提供する方法については、2-3 ページの「WAP ゲートウェイ」を参照してください。WML の一般情報については、2-6 ページの「その他の情報源」を参照してください。

## WMLScript

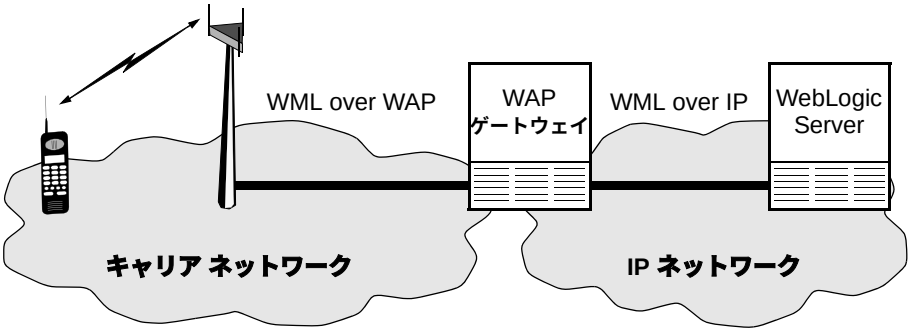
WMLScript は、WAP アーキテクチャに対して一般的なスクリプティング能力を提供します。WMLScript は、ナローバンド通信および無線クライアントの限界を克服するように設計されています。たとえば、WMLScript はサーバにアクセスすることなくフォーム入力を検証できます。

WMLScript は .wmls ファイルに記述されており、それらのファイルをドキュメント ルートに置くことによって無線クライアントに提供できます。ドキュメント ルートとは、WebLogic Server 上で公開されるファイルを格納するルート ディレクトリです。詳細については、「Web アプリケーションの基本事項」のディレクトリ構造に関する情報を参照してください。WMLScript の一般情報については、2-6 ページの「その他の情報源」を参照してください。

# WAP ゲートウェイ

次の図に示すとおり、WAP ゲートウェイは、無線クライアントを含んだ無線ネットワークとアプリケーション サーバを含んだコンピュータ ネットワークのブリッジとして機能します。

**図 2-1 WAP アプリケーションのアーキテクチャ**



# WAP ゲートウェイの機能

通常、WAP ゲートウェイには以下の機能が組み込まれています。

- プロトコル ゲートウェイ ― プロトコル ゲートウェイは、WAP プロトコル スタックから WWW プロトコル スタック（HTTP および TCP/IP）にリクエストを変換します。
- コンテンツ エンコーダおよびデコーダ ― コンテンツ エンコーダは、無線データ ネットワーク上を飛ぶパケットの数を削減するために、Web コンテンツを圧縮エンコードされた形式に変換します。

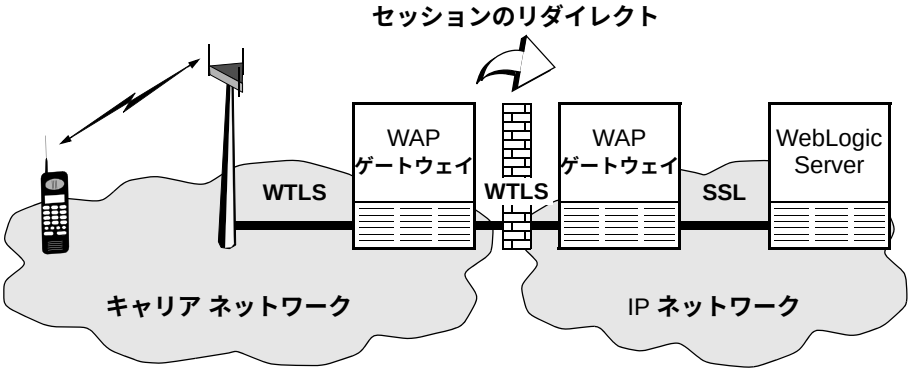
ある無線クライアントが WebLogic Server 上で実行中の WAP アプリケーションにリクエストを送ると、そのリクエストは、まずそれをデコードする WAP ゲートウェイを通って HTTP に変換され、それから適切な URL に転送されます。次に、その応答がゲートウェイを通ってルーティングされ、WAP に変換され、エンコードされて、無線クライアントに転送されます。このプロキシ アーキテクチャによって、アプリケーション開発者はネットワークや端末に依存しないサービスを構築できます。

# WAP ゲートウェイのセキュリティとセキュリティの考慮事項

WAP プロトコル スタックのセキュリティ レイヤは、Wireless Transport Layer Security（WTLS）と呼ばれています。WTLS は、確立されている Transport-Layer Security（TLS）プロトコル仕様をベースとしています。

WAP プロトコルを採用しているセキュアな接続では、WTLS（WAP サイド）から SSL（IP サイド）、および SSL から WTLS への切り替え中に WAP ゲートウェイにごくわずかなセキュリティ リスクが存在します。WAP プロトコルではセッションをキャリアのゲートウェイから企業のゲートウェイにリダイレクトできるため、企業側は、WAP ゲートウェイを自社のファイヤウォールの内側に置くことによってこの最小限のリスクを制御できます。次の図に示すように、企業は WAP ゲートウェイが動作するサーバを制御された環境下に置いて、あらゆるセキュリティ リスクを排除できます。

**図 2-2 WAP セッションのリダイレクト**




キャリアは信頼されている組織であり、音声、ファックス、コンピュータなどのデータの保護に絶えず責任を持っているので、企業が独自の WAP ゲートウェイをホストする必要がない場合もあります。

# WAP ゲートウェイ ベンダ

WAP ゲートウェイを提供するベンダの数は増加し続けています。WebLogic Server は、あらゆる WAP 対応ゲートウェイで正常に機能します。WAP 準拠のゲートウェイおよび他の WAP 製品の現行リストについては、WAP Forum がまとめた「WAP Deployment Fact Sheet」を参照してください。

# その他の情報源

以下に、WebLogic Server プログラミング、WAP、および WML に関する補足情報を参照できる Web サイトを（カテゴリ別に）示します。

## 関連する WebLogic テクノロジ

『WebLogic JSP プログラマーズ ガイド』

『WebLogic HTTP サーブレット プログラマーズ ガイド』

『WebLogic XML プログラミング ガイド』

「Web アプリケーションの基本事項」

## 一般的な WAP 情報

WAP Forum

Ericsson: Developers' Zone

Motorola

Nokia: WAP Solutions for Mobile Business

Openwave Developer Resources

## WAP 仕様およびホワイトペーパー

WAP 仕様

WAP ホワイト ペーパー

## WAP ツールキット

Nokia WAP Toolkit

Motorola Tools and Downloads

Openwave Software Development Kit

Ericsson Developers' Zone

# 3 WebLogic Server での i-Mode の使い方

以下の節では、i-Mode アプリケーション環境に対応したコンテンツを提供する方法、および WebLogic Server をコンフィグレーションして i-Mode ゲートウェイと一緒に使用する方法について説明します。

- 概要
- i-Mode マークアップ言語
- i-Mode ゲートウェイ
- その他の情報源

## 概要

i-Mode は、携帯電話などの無線デジタル デバイス向けのインターネットおよび Web ベース サービスを開発するために NTT DoCoMo によって開発およびデプロイされた閉じられた仕様です。i-Mode の唯一の実装は DoCoMo のモバイル インターネット アクセス システムですが、ヨーロッパと米国の複数の通信企業が i-Mode への関心を表明しています。

**注意：** i-Mode は、NTT DoCoMo が保有する商標およびサービス マークです。

i-Mode 仕様は、無線ネットワークの限界（狭いバンド幅、長いレイテンシ、そして予測不能な可用性と安定性）と無線デバイスの限界（CPU、メモリ、バッテリ寿命が限られていること、ユーザ インタフェースがシンプルであること）を解決しようとするものです。i-Mode 仕様は、無線通信の本質的な要素を 2 つ定義しています。これらは、無線プロトコルとマークアップ言語です。

i-Mode ゲートウェイは、無線ネットワーク上のクライアントとインターネット上のアプリケーション サーバにホストされているアプリケーションを接続するものです。i-Mode ゲートウェイは、無線クライアントからアプリケーション サーバへ要求をルーティングすることにより、電気通信とコンピュータ ネットワークの間にブリッジを構築します。

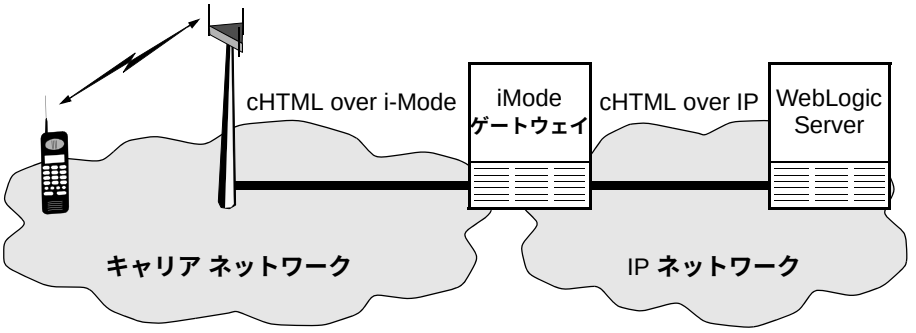
# i-Mode マークアップ言語

i-Mode アプリケーションは、compact Hypertext Markup Language（cHTML）を採用しています。これは、i-Mode 対応デバイス上のマイクロブラウザとのインタフェースとして設計されています。cHTML は HTML のサブセットで、携帯電話環境向けのいくつかの拡張機能を備えています。NTT DoCoMo Supported Tags and Specs（英語版）では、cHTML ドキュメントのタグと構造が定義されています。

# i-Mode ゲートウェイ

次の図に示すとおり、i-Mode ゲートウェイは、無線クライアントを含んだ無線ネットワークとアプリケーション サーバを含んだコンピュータ ネットワークのブリッジとして機能します。

**図 3-1 i-Mode アプリケーションのアーキテクチャ**




i-Mode ゲートウェイには、一般にプロトコル ゲートウェイが含まれます。プロトコル ゲートウェイは、i-Mode プロトコル スタックから WWW プロトコル スタック（HTTP および TCP/IP）にリクエストを変換します。

ある無線クライアントが WebLogic Server 上で実行中の i-Mode アプリケーションにリクエストを送ると、そのリクエストは、まず i-Mode ゲートウェイを通過し、それから適切な URL に転送されます。次に、その応答がゲートウェイを通過し、i-Mode に変換されて、無線クライアントに転送されます。このプロキシ アーキテクチャによって、アプリケーション開発者はネットワークや端末に依存しないサービスを構築できます。

# その他の情報源

以下に、WebLogic Server プログラミング、i-Mode 、および cHTML に関する補足情報を参照できる Web サイトを（カテゴリ別に）示します。

## 関連する WebLogic テクノロジ

『WebLogic JSP プログラマーズ ガイド』

『WebLogic HTTP サーブレット プログラマーズ ガイド』

『WebLogic XML プログラミング ガイド』

「Web アプリケーションの基本事項」

## 一般的な i-Mode 情報

All about i-mode

Q&A list for i-mode

List of i-mode compatible HTML 2.0 tags

NTT DoCoMo（英語版）Supported Tags and Specs

# 4 無線加入者に対応した Web アプリケーションの記述

以下の節では、WebLogic Server を使用して Web アプリケーションの対象範囲を無線加入者に拡大する方法を、サンプルを通して説明します。

- 概要
- 無線サンプル アプリケーションの保存場所
- 基本的な Web アプリケーション設計コンセプトについて
- WAP 専用 Web アプリケーションの記述
- 複数ターゲット Web アプリケーションの記述

## 概要

インターネット対応無線デバイスの普及率の上昇により、アプリケーション開発者には、既存のデスクトップ ブラウザだけではなく、それ以外のブラウザにも接続できる Web アプリケーションを設計することが求められています。また、無線デバイスのフォーム ファクタ、ユーザ入力インタフェース、および無線（OTA）プロトコルとマークアップ言語の使い方はデバイスごとに大幅に異なっているので、開発者は Web アプリケーションの設計時にさまざまな種類の無線デバイスを考慮に入れる必要があります。

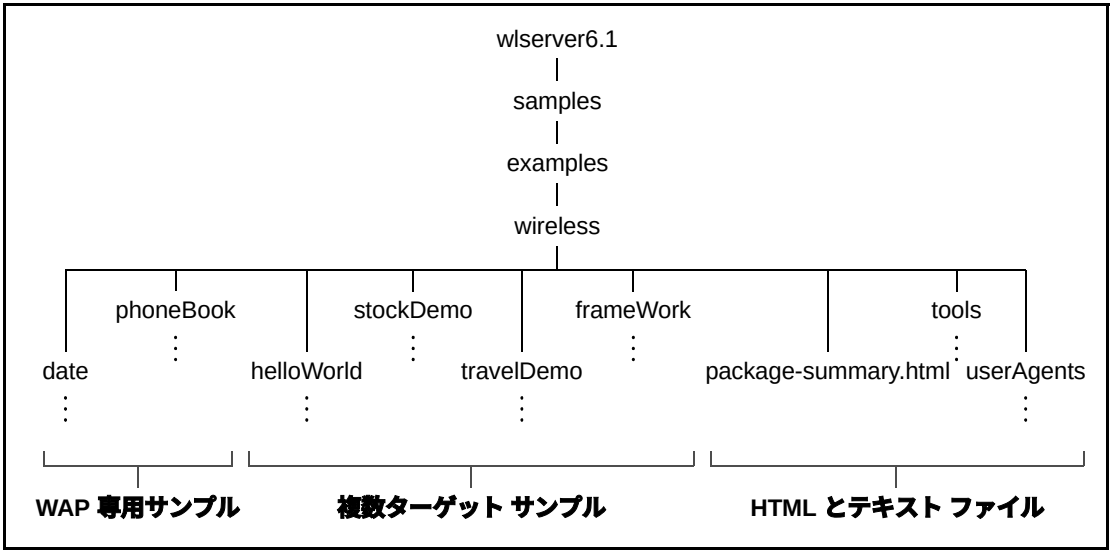
WebLogic Server に組み込まれている既存のアプリケーションは、無線 Web に対応しています。実際のところ、優れたアプリケーションの場合、大幅なコードの修正をしなくても、アプリケーションのバックエンド全体がインターネット対応無線デバイスに対応しています。最も単純な例では、アプリケーション開発者はフロントエンド ソフトウェアを記述するだけで、アプリケーションとインターネット対応無線デバイスを接続できます。

以下の節では、Web アプリケーション サービスを無線デバイスに適用するためのさまざまな方法について説明します。これらの方法は、サービスを無線デバイスに適用するための方法にしか過ぎませんが、WebLogic Server J2EE 機能を使用して独自の方法を開発するのに役立ちます。また、これらの方法を明確に理解できるよう、5 つの単純なサンプル アプリケーションも示します。

# 無線サンプル アプリケーションの保存場所

WebLogic Server ソフトウェアのインストール中、次の図に示すとおり、インストール プログラムによって 5 つの無線サンプル アプリケーションのディレクトリとファイルが `wireless` ディレクトリに置かれます。

**図 4-1 WebLogic Server 無線ファイル ツリー**




上に示した製品ディレクトリ、`wlserver6.1` は WebLogic Server 6.1 のデフォルト ディレクトリです。このデフォルト名は、インストール時に変更される場合があります。

次の表では、WebLogic Server 無線ディレクトリ構造の最上位のディレクトリとファイルについて簡単に説明します。5 つのサンプルには、無線デバイスまたはデスクトップ ブラウザから JavaServer Pages（JSP）と Java サーブレットにアクセスする方法が示されています。

| ディレクトリ名 | 説明 |
|---|---|
| `date` | `date` サンプル用のサンプル コードとリソース（サンプルの構築、コンフィグレーション、および実行の詳細を説明した `package-summary.html` ファイルなど）が格納されている。<br>`date` サンプルには、(1) WAP ゲートウェイを介した WML クライアント デバイスと WebLogic Server 間の接続性、および (2) 動的 WML ドキュメント（WML タグ付きのコンテンツ）の生成が示されている。 |
| `phoneBook` | `phoneBook` サンプル用のサンプル コードとリソース（サンプルの構築、コンフィグレーション、および実行の詳細を説明した `package-summary.html` ファイルなど）が格納されている。<br>`phoneBook` サンプルには、(1) WAP ゲートウェイを介した WML クライアント デバイスと WebLogic Server 間の接続性、および (2) 静的 WML ドキュメントの提供、および (3) WML クライアントから受信したサーブレットへのインタラクティブなリクエストが示されている。 |

| ディレクトリ名 | 説明 |
|---|---|
| helloWorld | helloWorld サンプル用のサンプル コードとリソース（サンプルの構築、コンフィグレーション、および実行の詳細を説明した package-summary.html ファイルなど）が格納されている。<br>helloWorld サンプルには、リクエスト側の無線またはデスクトップ デバイスに固有の静的マークアップ言語ドキュメントを提供するための基礎となるサンプル フレームワークが示されている。ドキュメントは、WML、HDML、cHTML、または HTML。 |
| stockDemo | stockDemo サンプル用のサンプル コードとリソース（サンプルの構築、コンフィグレーション、および実行の詳細を説明した package-summary.html ファイルなど）が格納されている。<br>stockDemo サンプルには、リクエスト側の無線またはデスクトップ デバイスに固有の単純な動的マークアップ言語ドキュメントを提供するための基礎となるサンプル フレームワークが示されている。ドキュメントは、WML、HDML、cHTML、または HTML。 |
| travelDemo | travelDemo サンプル用のサンプル コードとリソース（サンプルの構築、コンフィグレーション、および実行の詳細を説明した package-summary.html ファイルなど）が格納されている。<br>travelDemo サンプルには、複数のレベルのユーザ対話が含まれており、リクエスト側の無線またはデスクトップ デバイスに固有のより複雑なマークアップ言語ドキュメントを提供するための基礎となるサンプル フレームワークが示されている。ドキュメントは、WML、HDML、cHTML、または HTML。 |

| ディレクトリ名 | 説明 |
|---|---|
| frameWork | 単純なプロタイプ フレームワーク用のサンプル コードとリソース（フレームワークの詳細を説明した package-summary.html ファイルなど）が格納されている。<br>frameWork ディレクトリ（helloWorld、stockDemo、および travelDemo サンプルで使用されるサンプル フレームワークが格納されている）は、リクエスト側の無線またはデスクトップ デバイスに固有の JSP 内からマークアップ言語ドキュメントを生成する。ドキュメントは、WML、HDML、cHTML、または HTML。 |
| package-summary.html（ファイル） | WebLogic Server で提供される 5 つの無線サンプル アプリケーションの概要と、関連情報の参照先リストが格納されている。 |
| tools | WAP デバイス用に画像を WML ビットマップ（WBMP）フォーマットに変換するツール、および無線クライアント デバイスをエミュレートするツールへの参照先を列挙した 2 つのファイルが格納されている。 |
| userAgents | 実際のデバイスの userAgent サンプルを列挙したファイルが格納されている。userAgent 情報は、HTTP User-Agent ヘッダに表示される。 |

# 基本的な Web アプリケーション設計コンセプトについて

アプリケーション開発者は、Web アプリケーションのデータとビジネス ロジックを有線デスクトップ デバイスと無線デバイスの両方で使用できます。しかし、画面のレイアウトと移動方法は、デバイスの種類に応じて調整する必要があります。多くの場合、デスクトップ ブラウザ用のアプリケーション フローは、無線デバイス マイクロブラウザ用のそれとは大幅に異なります。

例として、デスクトップ ブラウザに e- コマース サイトを提供するための次のアプリケーション フローを考えてみましょう。

e- コマース サイトのホーム ページには、電子店舗、検索フォーム、および現在の特別商品の広告という、異なるセクションへのリンクが存在します。店舗の一部、商品カテゴリ、および特定の商品を選択すると、その商品の詳しい説明と、その商品をショッピング カートに追加するためのオプションが表示されます。その追加オプションを選択すると、ショッピングを続けるか、または会計に進むためのオプションが表示されます。

このアプリケーション フローは、消費者が電子店舗内をさらに見て回り、当初の目的よりさらに多くの商品を購入するように設計されています。デスクトップ ブラウザとマウスを使用する場合、このフローは非常によく機能します。しかし、このフローを使用して e- コマース サイトをデータ対応無線デバイスに提供した場合、消費者はフラストレーションのために電子店舗から立ち去ってしまう可能性があります。無線デバイスは、閲覧にはまったく適していないからです。代わりの方法として、これと同じ e- コマース サイトを携帯電話に提供するための次のアプリケーション フローを考えてみましょう。

このサイトのホーム画面では、消費者は商品の UPC バーコードまたは何らかのユニークな ID を入力できます。次の画面には、商品をカートに追加する、会計に進む、または商品の詳細を表示するためのリンクが表示されます。各ページには「総合電子店舗」へのリンクが表示され、顧客がいつでも閲覧できるようになっています。

このアプリケーション フローは、デスクトップ ブラウザ用のアプリケーション フローとは反対になっていることに注意してください。無線デバイス用のアプリケーション フローを作成するための鍵は、無線デバイスの入力および選択メカニズムを使用したユーザ対話を減らすことにあります。

したがって、Web アプリケーションを開発するときには、無線デバイスの限界を考慮し、適切なコンテンツを提供するための最も効率的で柔軟な方法を見つけ出さなければなりません。以下の節では、こうした考慮事項についていくつか説明します。

■ 4-10 ページの「セッション トラッキング」

# フォーム ファクタ

ほとんどの無線デバイスは、その画面のサイズと形状の違いから、異なるフォーム ファクタを持っています。無線マークアップ言語（WML、HDML、cHTML、Web Clipping など）は、これらのフォーム ファクタに対応するよう特別に設計されています。一部の WAP ゲートウェイ（WAP や i-Mode など）は HTML を異なるマークアップ言語に自動的に変換できますが、実際上、最も優れたアプリケーションは、WML を直接使用して無線ユーザ固有のニーズに合わせてインタフェースを調整します。このようにすることで、無線デバイスのフォーム ファクタを最大限に活用して、ユーザに最良の体験を提供できます。

多くの無線デバイスは、10 ～ 20 文字の行を 4 つまたは 5 つしか表示できません。このため、ユーザが満足する情報をいかにして小さい画面に提供するかが課題となっています。

# 物理的なユーザ インタフェース

ほとんどの無線デバイスは、キーボードとマウスが存在しないことから、非常に単純なインタフェースを搭載しています。データの入力や画面間の移動は、数字キーまたは数字ボタンを押すか、ペンで入力することによって行います。

アプリケーションを設計および記述するときには、入力するデータの量に加え、データ入力のメカニズムも考慮する必要があります。可能であれば、アプリケーションの必須データ入力ポイントをオプション データ入力ポイントから分離して、必須データ フィールドだけが無線デバイスに表示されるようにします。

マークアップ言語（WML、HDML、cHTML、Web Clipping など）を使用すると、無線デバイスのキーとボタンに処理を割り当てるアプリケーション フロントエンドを記述できます。携帯電話の場合、一例として、特定の処理（[ 前へ ]、[ 戻る ]、[OK]、[ 送信 ]、[ 選択 ]、[ 実行 ]、[ 次へ ]、[ 終了 ]、[ ホーム ] など）をラベルのない電源キーに割り当て、新しい処理をラベル付きのソフト キーに再び割り当てるためのアプリケーション フロントエンドをプログラミングできます。PDA デバイスの場合、新しい処理をソフト ボタン（シルクに印刷されたボタン）とハード ボタンに再び割り当てるためのアプリケーション フロントエンドをプログラミングできます。アプリケーションのデスクトップブラウザ

バージョンのボタンとリンクに代わり、アプリケーションの無線デバイス バージョンにキーまたはボタンを割り当てることができるのは、非常に便利なデータ入力および画面移動機能です。

# メモリの制限

ほとんどの無線デバイスは、メモリをほとんど搭載していません。WML カードをデックにグループ化する場合、1 つのデックが最小のダウンロード単位になるようにする必要があります。つまり、情報はカード単位ではなくデック単位で無線クライアントにダウンロードされます。こうしたメモリ制限のため、大量のカードで構成されるデックや、単一の大きいカードの使用を回避することをお勧めします。

**注意：** デック内のカードの数またはサイズを制限するのは、メモリだけではありません。一部の無線ゲートウェイでは、無線デバイスに提供される単一のペイロードのサイズに制限があります。

# 複数のクライアント タイプのサポート

Web アプリケーションの最も重要な部分は、ユーザと対話するために記述された特定の種類のマークアップ言語（従来の HTML）ではなく、そのユニークなコンテンツとバックエンドのデータベース対話です。優れたバックエンド機能であれば、修正を行わなくても、無線クライアントに同じサービスを提供できます。アプリケーション開発者は、マークアップ言語固有のフロントエンド（WML、HDML、cHTML、...）を開発しさえすれば、同じバックエンド機能を無線デバイスに提供できます。また、異なる種類のデバイス（携帯電話、PDA、従来のデスクトップ ブラウザなど）に応じてカスタマイズされたプレゼンテーションを定義し、特定のデバイスに固有の機能（リクエスト側デバイスの地理的位置に基づく携帯電話のパーソナライゼーションなど）を追加する必要があります。

HTML ベースのブラウザと無線クライアント タイプへのユーザ アクセスを処理するための基本的な方法は 2 つあります。ブラウザベースの HTML およびマイクロブラウザベースの WML/HDML/cHTML/... 入力ポイント用に別個の URL（`www.foo.com`、`www.wap.foo.com`、`www.imode.foo.com` など）を使用するよう Web アプリケーションをプログラミングする方法か、または、単一の URL（`www.foo.com` など）を使用してリクエスト側のブラウザ タイプに応じてコンテ

ンツを生成するよう Web アプリケーションをプログラミングする方法です。後者の方法では、その URL に存在する Web アプリケーションは、リクエストの HTTP `User-Agent` ヘッダを調べるか、または無線ゲートウェイからリクエスト側に関する情報を取得することによってブラウザの種類を特定します。HTTP `User-Agent` ヘッダ情報へのアクセス例については、WebLogic Server の `samples\examples\servlets` ディレクトリに格納されている `SnoopServlet` サンプルを参照してください。

同じ 2 つの方法は、開発者が市販の異なる無線デバイスのさまざまな機能とディスプレイ サイズを利用しようとする場合にも使用できます。無線デバイスのディスプレイ サイズは、現時点ではテキスト 4 行ないし 8 行分です（これについては近い将来大幅に改善される見通しです）。クライアントのタイプを調べることによって、アプリケーションは使用可能な場合にだけ追加のグラフィカル領域を使用できます。最も単純な手段は、最小公分母（4 行）に合わせたコンテンツを作成することです。ただし、この方法ではほとんどのデバイス上で最良のユーザ体験を提供できません。

一般に、すべてのデバイス（既存のデスクトップ ブラウザを含む）で同じ URL を使用して、アプリケーションへのアクセスをできるだけ簡単にすることをお勧めします。

# パーソナライゼーション

デバイスの種類に基づいてアプリケーションが提供するコンテンツをカスタマイズできるよう、パーソナライゼーション メカニズムを使用することを検討する必要があります。たとえば、あるデバイス（デスクトップ ブラウザなど）でユーザが見たいサービスは、別のデバイス（携帯電話）でそのユーザが見たいサービスとは大幅に異なる場合があります。BEA WebLogic Personalization Server のようなツールは、この種の柔軟性をアプリケーションに提供するのに非常に役立ちます。

また、パーソナライゼーション メカニズムを使用すると、デバイスの種類と場所に基づいてポータルを作成することができます。たとえば、米国の東海岸を訪れたときにはデバイス（携帯電話など）で特定のポータルを表示し、西海岸を訪れたときには同じデバイスで別のポータルを表示できます。さらに、デバイスの場所に基づいてポータルをパーソナライズすることもできます。

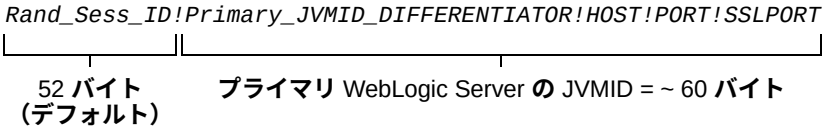
たとえば、デバイスが自宅から 20 マイルを超える場所にある場合はホーム ページにその土地のレストランの名前が表示され、自宅から 20 マイル以内にある場合はレストラン名が表示されないようにできます。

また、パーソナライゼーション メカニズムを使用すると、時刻または曜日に基づいて異なるデバイスにアラームを送信することができます。たとえば、ユーザが週末にページャを携帯していない場合、音声アラームを生成して留守番電話に記録できます。

# セッション トラッキング

セッション トラッキングは、複数の Web ページにわたってユーザの動きを追跡するのに役立ちます。『WebLogic HTTP サーブレット プログラマーズ ガイド』で説明されているように、サーバは各クライアント セッションに対し、セッション ID および関連付けられた `javax.servlet.http.HttpSession` オブジェクト（セッションの有効期間だけ WebLogic Server 上に存在する）を割り当てます。クライアントは、セッション ID を各リクエストに付与し、サーバが `HttpSession` オブジェクト内のセッション データを見つけられるようにします。

WebLogic Server のクラスタにデプロイされていないアプリケーションの場合、セッション ID は次の形式となります。




WebLogic Server のクラスタにデプロイされたアプリケーションの場合、セッション ID はさらに 60 バイトが追加された次の形式となります。

```
Rand_Sess_ID!Primary_JVMID_DIFFERENTIATOR!HOST!PORT!SSLPORT!
      SECONDARY_JVMID_DIFFERENTIATOR!HOST!PORT!SSLPORT
```

サーバは、クライアントにクッキーを設定することによって、セッション ID を保存しようとします（クッキーは、Web ブラウザを識別し、それによってカスタマイズおよび円滑化されたサービスを提供するためにインターネットで使用されます）。クッキーが設定されたクライアントは、サーバに送信される各ユーザ リクエストにそのクッキーを含めるようになります。サーバは、クッキーのセッション ID を自動的に解析し、適切な `HttpSession` オブジェクトからセッション データを取り出します。

ただし、ほとんどの無線デバイスはクッキーをサポートしておらず、また、一部の無線ゲートウェイはクライアントのクッキーを処理しないため、別のセッション トラッキング手法を使用しなければならない場合もあります。

# セッション トラッキングの代替手法 ー URL 書き換え

URL 書き換えでは、サーブレットがクライアントに送り返すページ上のハイパーリンクにセッション ID がエンコードされます。ユーザが以後これらのリンクをクリックすると、WebLogic Server は URL からそのセッション ID を抽出し、適切な `HttpSession` オブジェクトを見つけ出します。

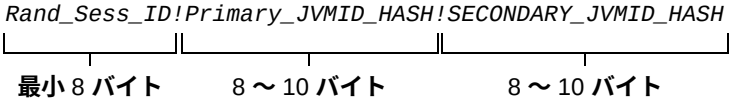
均等なランダム分散を使用してセッションのセキュリティを保証するために、セッション ID には特定数の文字が含まれなければなりません。しかしながら、多くのデバイスは URL を最大 128 文字（バイト）に制限しているので、セッション ID の長さが問題となる可能性があります。

セッション ID の長さを制限する方法は以下の 2 通りです。

- Web アプリケーションのデプロイメント記述子、`weblogic.xml` の `<session-descriptor>` 要素で、`IDLength` 属性を設定することにより、セッション ID のうちランダムに生成される部分の長さを制限できます。最小値は 8 バイトです。
- ドメインの `config.xml` ファイルの `<WebServer>` 要素で、`WAPEnabled` 属性を `true` に設定することにより、セッション ID のその他の部分（JVMID）の長さを制限できます。

  サービス パックが適用されていない WebLogic Server 6.1 では、`WAPEnabled` を `true` に設定すると、セッション ID のうちのランダムに生成される部分にまでセッション ID が制限されます。つまり、セッション ID には、プライマリ サーバおよびセカンダリ サーバの情報が含まれなくなります。

  サービス パック 1 以降を適用した WebLogic Server 6.1 では、`WAPEnabled` を `true` に設定すると、WebLogic Server のクラスタにデプロイされたアプリケーションのセッション ID は次の形式に変更されます。




  サーバは、*`Primary_JVMID_HASH`* と *`SECONDARY_JVMID_HASH`* の各値を、サーバに格納されているプライマリ サーバおよびセカンダリ サーバ情報にマッピングし、適切な `HttpSession` オブジェクトからセッション データを取り出します。

長いセッション ID の形式（WAPEnabled を false に設定）と、制限されたセッション ID の形式（WAPEnabled を true に設定）のどちらの場合でも、ステートのインメモリ レプリケーションが使用されます。つまり、セッション永続性は、WebLogic Cluster の要件ではないということです。WebLogic Server でインメモリ レプリケーションを使用していない場合、JVMID は URL にエンコードされません。WAPEnabled 属性の詳細については、「config.xml 要素と属性」を参照してください。

## セッション トラッキングのより優れた代替手法

さらに優れた方法は、セッション ID が指定される WML postfield 要素を、WML go 要素と一緒に組み込むことです。WML では、go 要素は URL への移動を示します。go 要素には、1 つまたは複数の postfield 要素を含めることができます。これらの要素は、要求の間に元のサーバに送られる情報を指定します。以下の WML コード（スクリプトレット）では、セッション ID はセッションから取得され、JSESSIONID postfield の値を設定するために使用されます。

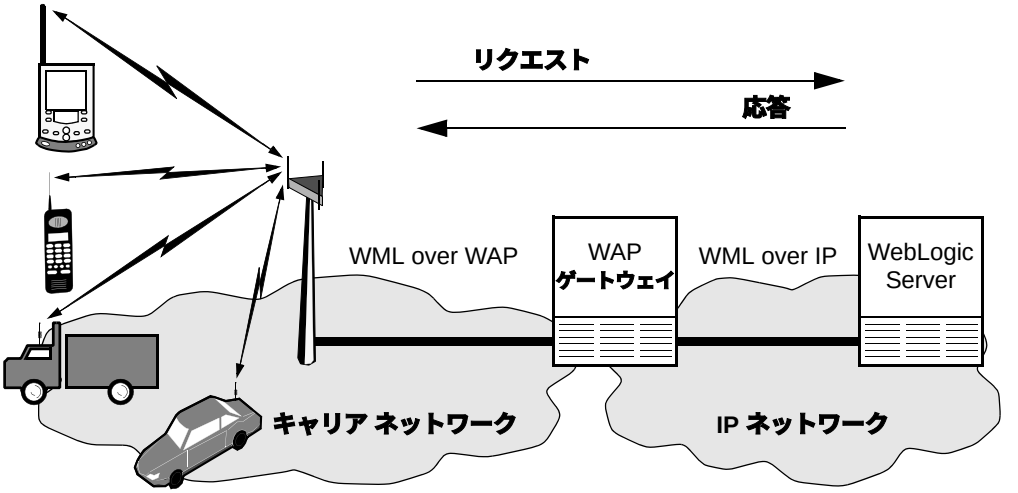
```
<go href="index.jsp" method="post"> <postfield name="JSESSIONID"
     value="<%= session.getId() %>"/></go>
```

上記のコードを使用すると、JSESSIONID=sessionID というメッセージ エンティティを持つ URL index.jsp への HTTP POST が実行されます。ここで sessionID とは、session.getId() 呼び出しから取得される ID です。index.jsp 内部からは、request.getParameter("JSESSIONID") 呼び出しで HTTP リクエストからパラメータを取得することによってセッション ID を取得できます。

# WAP 専用 Web アプリケーションの記述

date および phoneBook アプリケーションは、WAP 専用 Web アプリケーションのサンプルです。これらのサンプル アプリケーションは、次の図に示すとおり、WML クライアントから WebLogic Server にアクセスする方法を表しています。

**図 4-2 WAP 専用サンプル アプリケーションのデータ フロー**




WML クライアントからのリクエストは、WAP ゲートウェイを介して HTTP リクエストの形式で WebLogic Server に転送されます。WebLogic Server は、静的ファイルか、JSP または Java サーブレットとして記述された HTTP サーブレットを提供することによって HTTP リクエストに応答できます。WAP アプリケーションでは、静的ファイルは通常 WML ファイルになり、JSP とサーブレットは WML を動的に生成するために使用されます。

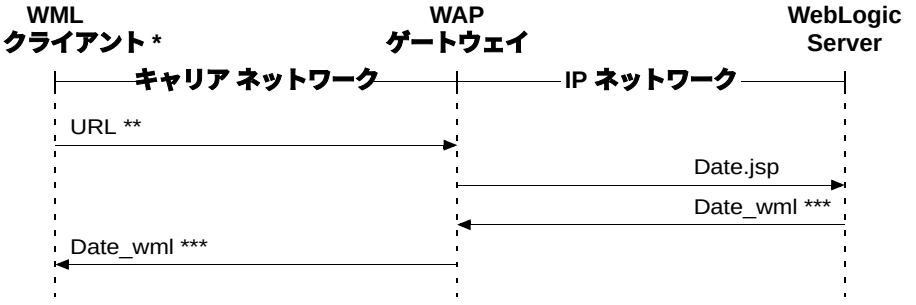
WAP アプリケーションを作成するには、単一の種類のアプリケーション クライアント、つまり WML 対応マイクロブラウザを搭載した無線デバイスをサポートする WML フロントエンドを記述します。WML は XML をベースとしているため、『WebLogic XML プログラミング ガイド』で、WebLogic Server から XML を生成する例を参照してください。

`date` および `phoneBook` サンプル アプリケーションは比較的シンプルなので、J2EE と WebLogic Server API を使用して WAP アプリケーションを開発する方法を最初に学ぶのに最適です。ただし、これらのサンプルは、WAP アプリケーション モデルを実施に移すための複数の方法の 1 つに過ぎず、あくまで実例としてとらえる必要があることに注意してください。

# date サンプルを使った作業

date サンプルには、JSP から WML ドキュメントを生成して WML クライアントに提供する仕組みが示されています。次の図に、そのデータ フローの概要を示します。

**図 4-3 date アプリケーションへのアクセス**




*** マイクロブラウザ**

** URL = http://wls_machine:listen_port/examplesWebApp/Date.jsp

***** Date.jsp によって生成されて返される WML ドキュメント**

WAP ゲートウェイは、クライアントから受信した WML リクエストを HTTP サーブレット リクエストに変換し、変換したリクエストを WebLogic Server 上で動作する Date.jsp に転送します。Date.jsp は、現在の日付と時刻を調べ、その結果を次の WML ドキュメントとして返すことによってこのリクエストに応答します。

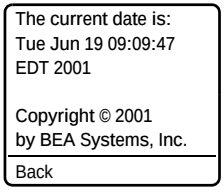
**コード リスト 4-1 date ディレクトリの Date.jsp**

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Copyright (c) 2001 by BEA Systems, Inc.
     All Rights Reserved. -->

<!-- set document type to WML -->
<%@ page contentType="text/vnd.wap.wml" %>

<wml>
  <template>
```

```
    <do type="prev" label="back">
      <prev/>
    </do>
  </template>

  <card title="WML DATE EXAMPLE" id="frstcard">
    <p>
      <small>The current date is:
      <br/>
      <%= new Date() %>
      <br/>
      Copyright &#xA9; 2001 by BEA Systems, Inc.
        All Rights Reserved.</small>
    </p>
  </card>
</wml>
```

リクエスト側の WML クライアントはマークアップ言語ドキュメントにアクセスし、次のサンプル ディスプレイに示すとおり、現在の日付と時刻を出力します。

**図 4-4 date のサンプル ディスプレイ**




`Date.jsp` の `<%@ page contentType="text/vnd.wap.wml" %>` 行は、生成されるドキュメントの MIME タイプを WML MIME タイプに設定します。これは、WML を直接 JSP またはサーブレットから生成するときには常に必要です。この行が存在しない場合、MIME タイプはデフォルトによって HTML MIME タイプとなり、WAP ゲートウェイはそのドキュメントを WML に変換しようとして不適切な結果が生じる場合があります。

**注意：** MIME（Multipurpose Internet Mail Extensions）は、標準のインターネット電子メールの機能を拡張するための仕様です。MIME は、インターネット メールを介した伝送のためにさまざまなメディア タイプを表現およびエンコードするための標準化された方法を提供します。

`date` サンプルに必要な MIME タイプ（次の表を参照）は、WebLogic Server に登録されています。

**表 4-1 WAP 専用サンプル アプリケーション用の MIME タイプの定義**

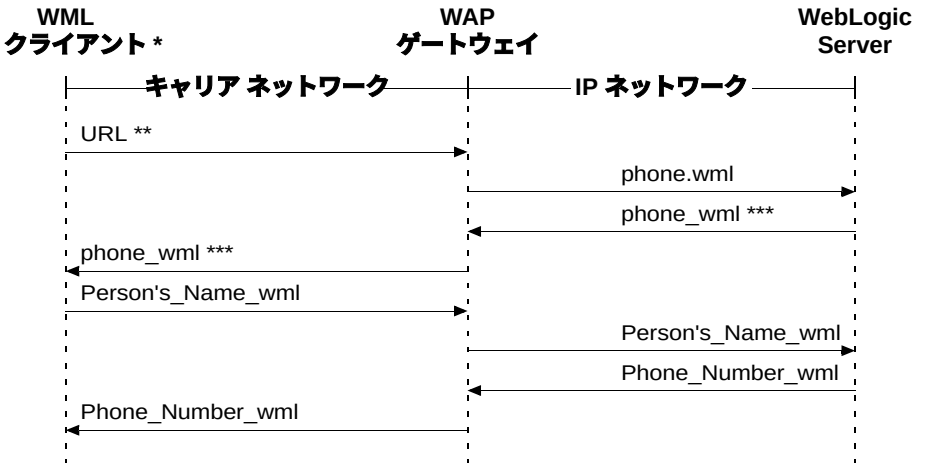
| 拡張子 | MIME タイプ | 説明 |
|---|---|---|
| .wml | text/vnd.wap.wml | WML ソース ファイル |
| .wmlc | application/vnd.wap.wmlc | WML コンパイル済みファイル |
| .wmls | text/vnd.wap.wmlscript | WMLScript ソース ファイル |
| .wmlsc | application/vnd.wap.wmlscriptc | WMLScript コンパイル済みファイル |
| .wbmp | image/vnd.wap.wbmp | WML ビットマップ |

これらの MIME タイプは、インストールされている WebLogic Server の config\examples\applications\examplesWebApp\WEB-INF\web.xml ファイルで参照できます。このファイルに記述されている MIME タイプは参照専用です。したがって、このファイル内の MIME タイプを追加または削除しても、WebLogic Server に登録されている MIME タイプは変更されません。

# phoneBook サンプルを使った作業

phoneBook サンプルは、WML クライアントから受信したフォームのデータを処理する方法を示しています。次の図に、そのデータ フローの概要を示します。

**図 4-5 phoneBook アプリケーションへのアクセス**




*** マイクロブラウザ**

** URL = http://wls_machine:listen_port/examplesWebApp/Date.jsp

***** phone.wml ファイルから返される WML ドキュメント**

次のリストに示す phone.wml ファイルは、select 要素を使用して、電話番号を調べるためのオプションをユーザに提供します。

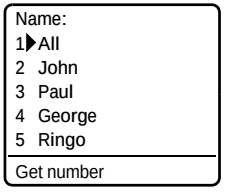
**コード リスト 4-2 phoneBook ディレクトリの phone.wml**

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Copyright (c) 2000-2001 by BEA Systems, Inc.
     All Rights Reserved. -->

<wml>
  <card id="card1" title="Phone Book" newcontext="true">
    <p>
```

```
      Name:
      <select name="name" value="" title="Name">
        <option value="">All</option>
        <option value="John">John</option>
        <option value="Paul">Paul</option>
        <option value="George">George</option>
        <option value="Ringo">Ringo</option>
      </select>
    </p>
    <do type="accept" label="Get number">

      <!-- Edit the URL below to point to the appropriate
           hostname and listenport of your WebLogic Server -->
      <go href="http://localhost:7001/examplesWebApp/
           PhoneServlet?name=$(name:escape"/>
    </do>
  </card>
</wml>
```

リクエスト側の WML クライアントはマークアップ言語ドキュメントにアクセスし、次のサンプル ディスプレイに示すとおり、電話番号を検索するためのオプションを出力します。

**図 4-6 phoneBook のサンプル ディスプレイ**



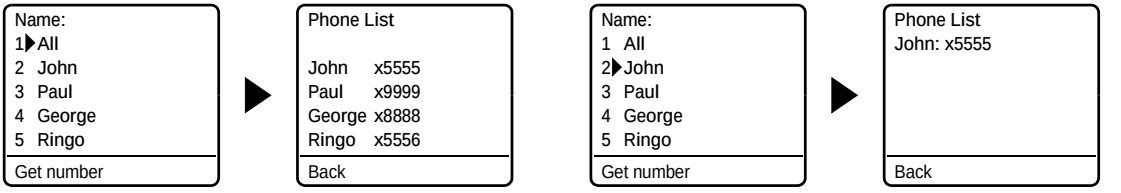

ユーザの入力（All または名前）に基づいて、クライアントは WAP ゲートウェイを介して次の HTTP リクエストを PhoneServlet（WebLogic Server の samples\examples\servlets ディレクトリに格納されている既存のサーブレット）に送信します。

```
http://wls_machine:listen_port/examplesWebApp/phoneServlet?name=
    $(name:escape)
```

クエリ パラメータの name は、サーブレットの URL に追加されます。このサンプルでは、PhoneServlet は phonelist というテキスト ファイル（WebLogic Server の samples\examples\servlets ディレクトリに格納されている）から

WML 応答を生成し、WAP ゲートウェイはその応答を WML クライアントに転送します。次の図に、phoneBook アプリケーション フローとユーザ入力を示します。

**図 4-7 phoneBook アプリケーションの使用例**




**注意：** WAP ゲートウェイをコンフィグレーションして HTML から WML に自動的に変換することも可能ですが、WML はほとんどの WAP クライアント デバイスの表示上の限界に対処できるように設計されているので、WML を直接生成することをお勧めします。

# WAP 専用サンプル アプリケーション用の追加ソフトウェアのインストール

date および phoneBook サンプル アプリケーションを実行するには、以下のソフトウェアをインストールする必要があります。

- WML クライアント デバイスまたは適切なエミュレーション ソフトウェア
- WML ゲートウェイまたは適切なエミュレーション ソフトウェア

WML クライアント エミュレーション ソフトウェアを販売しているベンダのリストについては、WebLogic Server の samples\examples\wireless\tools\emulators.txt ファイルを参照してください。

# WAP 専用サンプル アプリケーションの実行

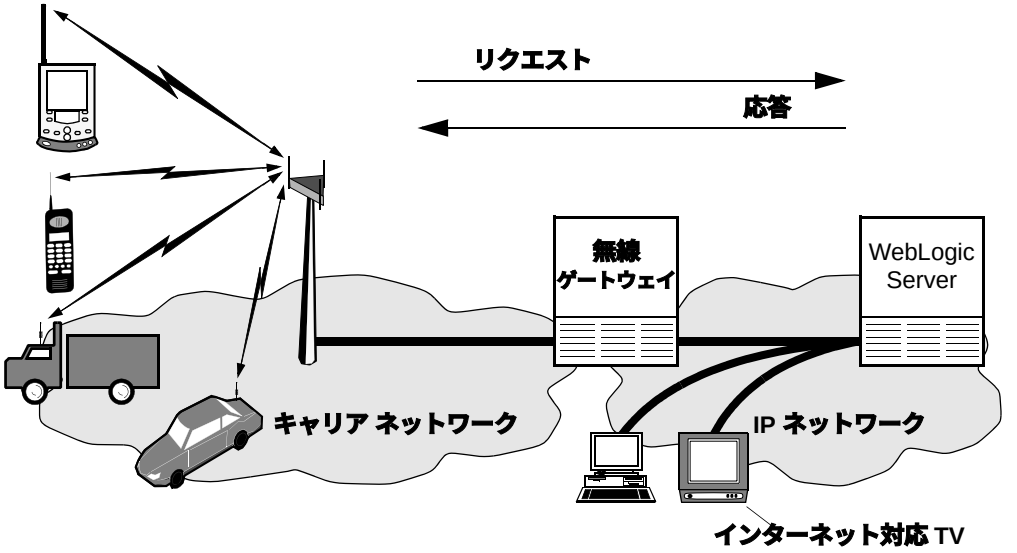
サンプル Web アプリケーションのディレクトリ階層は、WebLogic Server の config\examples\applications ディレクトリに格納されています。date および phoneBook サンプルを構築すると、config\examples\applications\examplesWebApp ディレクトリにはサンプル アプリケーション用に作成されたアプリケーション アーカイブが格納されます。

date または phoneBook サンプル アプリケーションの構築、コンフィグレーション、および実行については、対象サンプルのソース ファイルが格納されているサンプル ディレクトリ（date、phoneBook）内の package-summary.html ファイルを参照してください。

# 複数ターゲット Web アプリケーションの記述

helloWorld、stockDemo、および travelDemo アプリケーションは、複数ターゲット アプリケーションの記述例です。これらのサンプル アプリケーションは、次の図に示すとおり、WML、HDML、cHTML、および HTML クライアントから WebLogic Server にアクセスする方法を表しています。

**図 4-8 複数ターゲット サンプル アプリケーションのデータ フロー**




各サンプル アプリケーション（helloWorld、stockDemo、および travelDemo）は、以下の種類のアプリケーション クライアントをサポートする 4 つのフロント エンドを備えています。それらは、WML 対応マイクロブラウザ搭載デバイス、HDML 対応マイクロブラウザ搭載デバイス、cHTML 対応マイクロブラウザ搭載デバイス（注意を参照）、および HTML 対応マイクロブラウザまたはブラウザ搭載のデバイスです。複数ターゲット アプリケーションを記述すると、アプリケーション開発者は標準の Web 技術による既存の開発努力を保持しつつ、無線加入者へのアクセスを取得できます。

**注意：** これらのサンプルは非常にシンプルなので、サンプル用の HTML JSP ページは cHTML JSP ページとしても機能します。

helloWorld、stockDemo、および travelDemo アプリケーションは比較的シンプルなので、J2EE と WebLogic Server API を使用して複数ターゲット Web アプリケーションを開発する方法を最初に学ぶのに最適です。ただし、これらのサンプルは、複数ターゲット アプリケーション モデルを実施に移すための複数の方法の 1 つに過ぎず、あくまで実例としてとらえる必要があることに注意してください。

# サンプル フレームワークについて

helloWorld、stockDemo、および travelDemo サンプルは、単純なプロトタイプ フレームワークを使用してデバイスに依存しないページを選択します。デバイス リクエストを処理するためにサンプル フレームワークが呼び出されると、このフレームワークは、リクエストの HTTP User-Agent ヘッダを調べることによってリクエスト側デバイスの種類を特定して、使用する JSP ページとそのページに渡す値を指定したオブジェクトを返します。サンプル フレームワークは、1 つの URL を使用して、リクエスト側デバイスの種類に応じてコンテンツ（WML、HDML、cHTML、HTML）を生成するよう設計されています。

# MIME タイプの登録

helloWorld、stockDemo、および travelDemo サンプルに必要な MIME タイプ（次の表を参照）は、WebLogic Server に登録されています。

**表 4-2 複数ターゲット サンプル アプリケーション用の MIME タイプの定義**

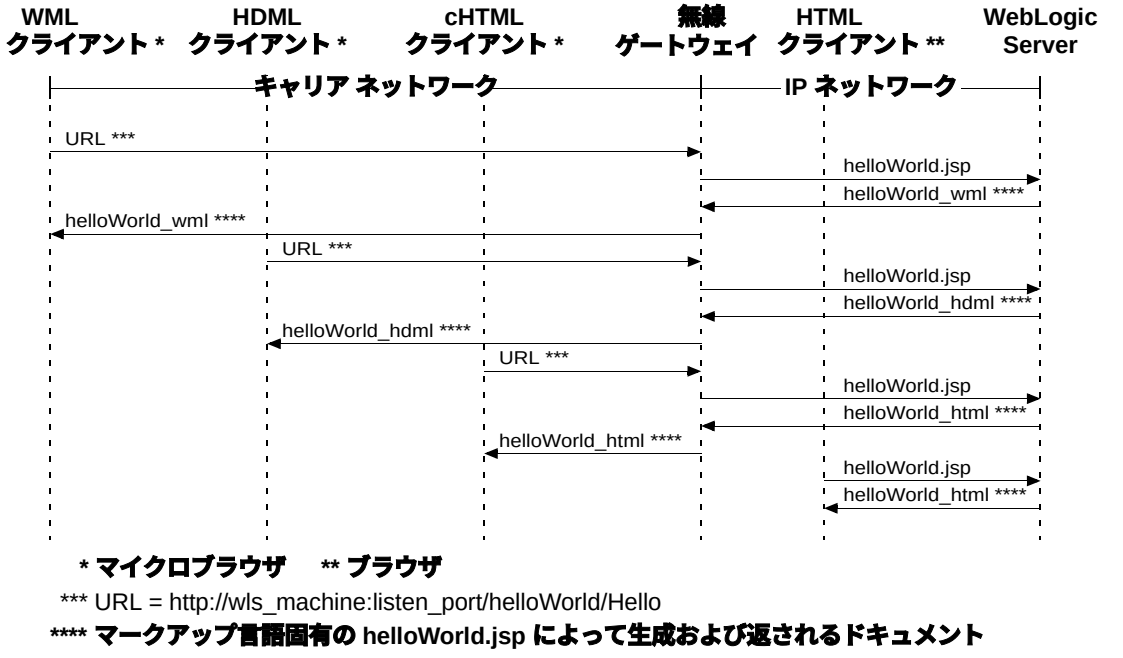
| 拡張子 | MIME タイプ | 説明 |
|---|---|---|
| .wml | text/vnd.wap.wml | WML ソース ファイル |
| .wmlc | application/vnd.wap.wmlc | WML コンパイル済みファイル |
| .wmls | text/vnd.wap.wmlscript | WMLScript ソース ファイル |
| .wmlsc | application/vnd.wap.wmlscriptc | WMLScript コンパイル済みファイル |
| .wbmp | image/vnd.wap.wbmp | WML ビットマップ |
| .hdml | text/x-hdml | HDML ソース ファイル |
| .bmp | image/bmp | HDML ビットマップ |

これらの MIME タイプは、インストールされている WebLogic Server の `config\examples\applications\examplesWebApp\WEB-INF\web.xml` ファイルで参照できます。このファイルに記述されている MIME タイプは参照専用です。したがって、このファイル内の MIME タイプを追加または削除しても、WebLogic Server に登録されている MIME タイプは変更されません。

# helloWorld サンプルを使った作業

`helloWorld` サンプルは、(1) リクエスト側デバイスに固有の単純なマークアップ言語の生成、(2) 生成されたドキュメントのリクエスト側デバイスへの提供、を示したものです。次の図に、そのデータ フローの概要を示します。

**図 4-9 helloWorld アプリケーションへのアクセス**




*** マイクロブラウザ　** ブラウザ**

*** URL = http://wls_machine:listen_port/helloWorld/Hello

****** マークアップ言語固有の helloWorld.jsp によって生成および返されるドキュメント**

無線クライアントからのリクエストの場合、クライアントの無線ゲートウェイはクライアントから受信したマークアップ言語リクエストを HTTP サーブレット リクエストに変換し、そのリクエストを WebLogic Server 上の `helloWorld` の HTTP サーブレットに転送します。デスクトップからのリクエストの場合、デスクトップ クライアントは WebLogic Server 上の `helloWorld` の HTTP サーブレットに直接 HTTP サーブレット リクエストを送信します。

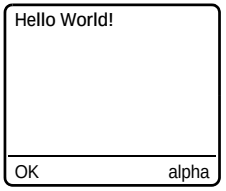
HTTP サーブレットは、そのリクエストのデバイス タイプと JSP ページを解決するためにそのリクエストをサンプル フレームワークに渡します。サンプル フレームワークは、`helloWorld` サンプル ディレクトリの `wml`、`hdml`、または `html` サブディレクトリ内の適切な `helloWorld.jsp` ページの場所を返します。選択された `helloWorld.jsp` は、マークアップ言語固有のドキュメントを生成してリクエスト側デバイスに返すことによって応答を行います。このドキュメントは、ユーザに `Hello World!` という文字列を表示します。次に、このドキュメントの WML バージョンを示します。

**コード リスト 4-3 helloWorld ディレクトリの wml サブディレクトリ内の helloWorld.jsp**

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Copyright (c) 2001 by BEA Systems, Inc.
     All Rights Reserved. -->

<%@ page contentType="text/vnd.wap.wml"%>

<wml>
  <template> <do type="prev" label="back"> <prev/></do> </template>
  <card id="firstcard">
    <p>Hello World!
    <br/> <a title="next" href="#secondcard">Next</a>
    </p>
  </card>
  <card id="secondcard">
    <p> Goodbye from WML.</p>
  </card>
</wml>
```

リクエスト側のデバイスはマークアップ言語ドキュメントにアクセスし、次のサンプル ディスプレイに示すとおり、`Hello World!` を出力します。

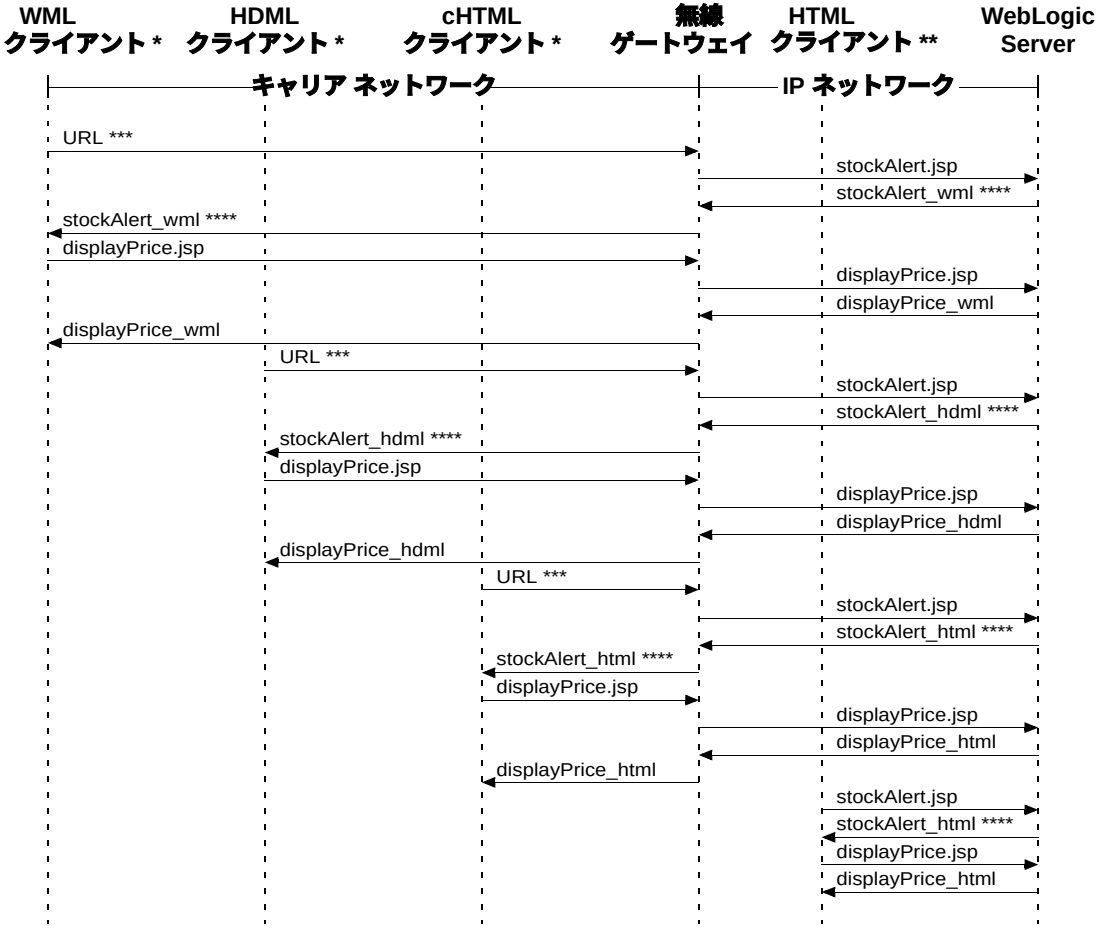
**図 4-10　helloWorld サンプル ディスプレイ**




# stockDemo サンプルを使った作業

stockDemo サンプルには、クライアントから受信した単純なフォームのデータを処理する方法が示されています。次の図に、そのデータ フローの概要を示します。

**図 4-11 stockDemo アプリケーションへのアクセス**




*** マイクロブラウザ ** ブラウザ**

*** URL = http://wls_machine:listen_port/stockDemo/Stock

****** マークアップ言語固有の stockAlert.jsp によって生成および返されるドキュメント**

stockDemo の HTTP サーブレットは、HTTP サーブレット リクエストを受信すると、そのリクエストのデバイス タイプと JSP ページを解決するためにそのリクエストをサンプル フレームワークに渡します。サンプル フレームワークは、stockDemo サンプル ディレクトリの wml、hdml、または html サブディレクトリ

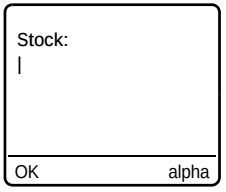
内の適切な stockAlert.jsp ページの場所を返します。選択された stockAlert.jsp は、マークアップ言語固有のドキュメントを生成してリクエスト側デバイスに返すことによって応答を行います。このドキュメントは、株式クエリを入力するための Stock: プロンプトを含んだフォームをユーザに表示します。次に、このドキュメントの HDML バージョンを示します。

**コード リスト 4-4 stockDemo ディレクトリの hdml サブディレクトリ内の stockAlert.jsp**

```
<%@ page contentType="text/x-hdml"%>

<!-- Copyright (c) 2001 by BEA Systems, Inc.
     All Rights Reserved. -->

<HDML VERSION="3.0" TTL="0" PUBLIC="TRUE">

<ENTRY name="first" KEY="stock">
<ACTION TYPE="ACCEPT" TASK="GO"
     DEST="/stockDemo/StockAlert?stock=$(stock)">
Stock:
</ENTRY>

</HDML>
```

リクエスト側のデバイスはマークアップ言語ドキュメントにアクセスし、次のサンプル ディスプレイに示すとおり、Stock: プロンプトを出力します。

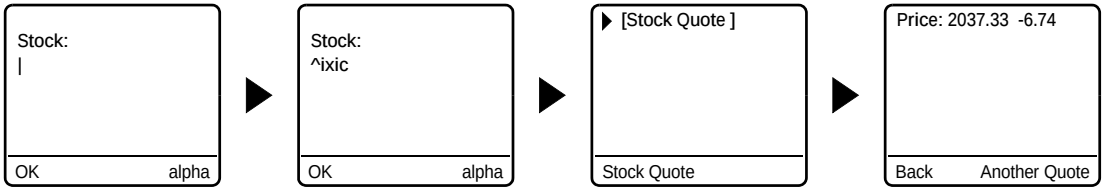
**図 4-12 stockDemo のサンプル ディスプレイ**




ユーザの入力（株式名）に基づいて、HTTP リクエストが stockDemo の HTTP サーブレットに対して行われ、クエリ パラメータ（stock）がサーブレットの URL に追加されます。サンプル フレームワークは、適切な displayPrice.jsp ページ（stockDemo サンプル ディレクトリの wml、hdml、または html サブディレクトリに格納されている）の場所と、リクエストされた株式の値（YaHoo! から取得）を返します。選択された stockAlert.jsp は、マークアップ言語固有

のドキュメントを生成してリクエスト側デバイスに返し、リクエスト側デバイスはそのマークアップ言語ドキュメントにアクセスしてそのドキュメントを画面に出力します。次の図に、stockDemo アプリケーション フローとユーザ入力を示します。

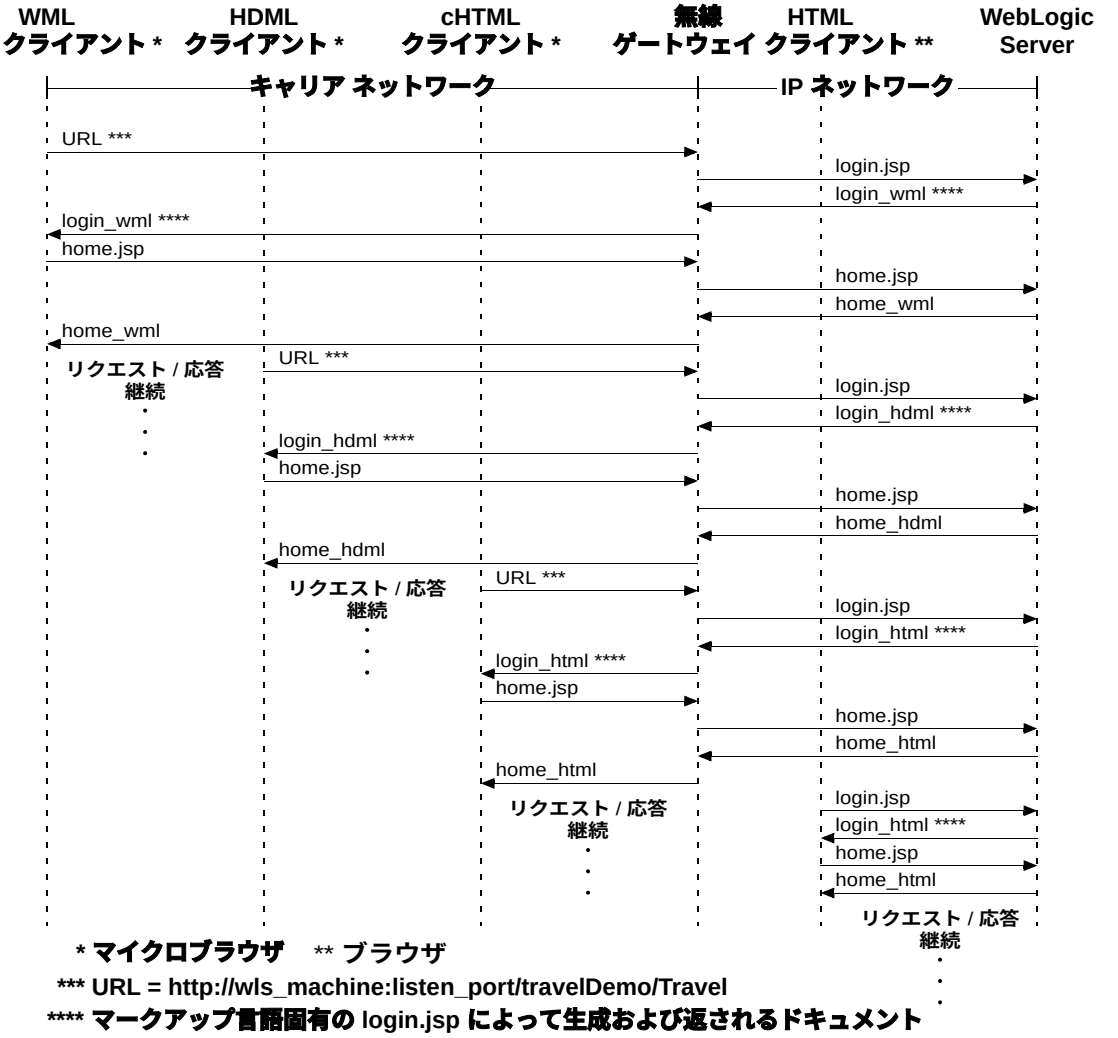
**図 4-13 携帯電話からの株式相場のチェック — サンプル**




# travelDemo サンプルを使った作業

travelDemo サンプルには、クライアントから受信したより複雑なフォームのデータを処理する方法が示されています。次の図に、そのデータ フローの概要を示します。

**図 4-14 travelDemo アプリケーションへのアクセス**



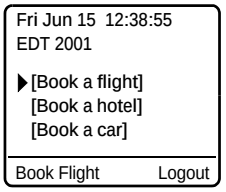
travelDemo サンプル アプリケーションは、ログイン フォームから始まる一連のフォーム ドキュメントをリクエスト側デバイスに送信します。ユーザ名 x とパスワード y を使用してログインに成功すると、travelDemo の home.jsp はデバイス固有の旅行フォーム ドキュメントを生成してリクエスト側デバイスに返します。次に、そのドキュメントの HTML バージョンを示します。

**コード リスト 4-5 travelDemo ディレクトリの html サブディレクトリ内の home.jsp**

```
<html>

<!-- Copyright (c) 2001 by BEA Systems, Inc.
     All Rights Reserved. -->

<HEAD>
<TITLE>Home</TITLE>
<meta name="palmcomputingplatform" content="true">
</HEAD

<body>
<h1>TravelDemo Homepage</h1>
<ul>
<li><a href="<%=
     response.encodeURL("/travelDemo/FindAFlight")%>">
          Book a flight</a>
<li><a href="<%=
     response.encodeURL("/travelDemo/NotImplemented")%>">
          Book a hotel</a>
<li><a href="<%=
     response.encodeURL("/travelDemo/NotImplemented")%>">
          Book a car</a>
<li><a href="<%=
     response.encodeURL("/travelDemo/NotImplemented")%>">
          View reservations</a>
<li><a href="<%=
     response.encodeURL("/travelDemo/NotImplemented")%>">
          View account</a>
<li><a href="<%=
     response.encodeURL("/travelDemo/Logout")%>">Logout</a>
</ul>
</body>
</html>
```

リクエスト側のデバイスは旅行フォーム ドキュメントにアクセスして、次のサンプル ディスプレイに示すとおり、そのドキュメントを出力します。

**図 4-15 travelDemo のサンプル ディスプレイ**

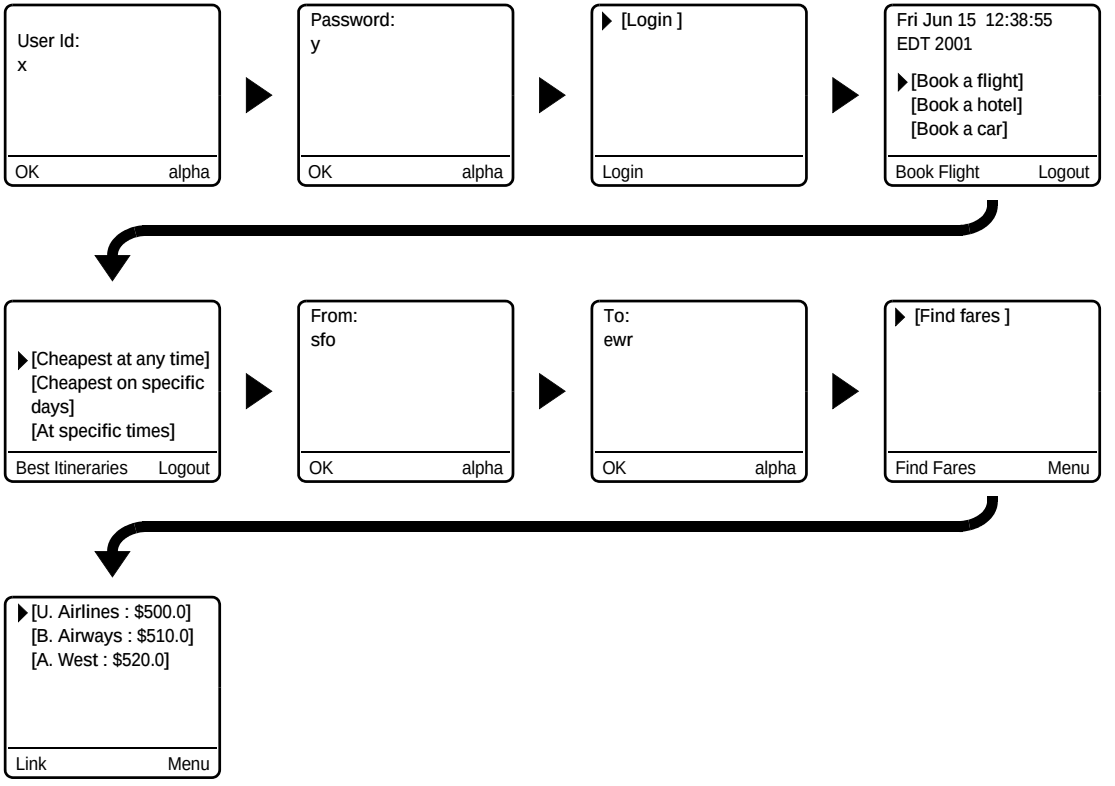



このドキュメントには、フライト、ホテル、車の予約、予約またはアカウントの確認、またはログ アウトのオプションが表示されます。実際に表示されるオプションは、リクエスト側デバイスの種類によって異なります。

**注意：** `Book a hotel`、`Book a car`、`View reservations`、および `View account` は、`travelDemo` サンプル アプリケーションでは実装されていません。

ユーザが `Book a flight` を選択すると、新しい画面が表示され、`Cheapest at any time` および `Cheapest on specific days` というオプションが示されます。これらのオプションの 1 つを選択した場合、次の `From:` プロンプトでは `sfo`（サンフランシスコ国際空港）が、`To:` プロンプトでは `ewr`（ニューアーク国際空港）が唯一の有効なユーザ入力となります。次の図に、`travelDemo` アプリケーション フローとユーザ入力を示します。

**図 4-16 携帯電話からのフライトの予約 — サンプル**




ユーザの入力に基づいて、`travelDemo` は応答を生成してリクエスト側デバイスに転送します。この応答は「あらかじめ決められた」応答です。このため、ユーザーは `travelDemo` サンプル アプリケーションにアクセスしても、実際にフライト、ホテル、車の予約や、予約またはアカウントの確認を行うことはできません。

# 複数ターゲット サンプル アプリケーション用の追加ソフトウェアのインストール

helloWorld、stockDemo、および travelDemo サンプル アプリケーションを実行するには、以下のソフトウェアをインストールする必要があります。

- 無線クライアント デバイス（WML、HDML、cHTML、HTML）または適切なエミュレーション ソフトウェア
- 無線ゲートウェイまたは適切なエミュレーション ソフトウェア

無線クライアント エミュレーション ソフトウェアを販売しているベンダのリストについては、WebLogic Server の samples\examples\wireless\tools\emulators.txt ファイルを参照してください。

# 複数ターゲット サンプル アプリケーションの実行

サンプル Web アプリケーションのディレクトリ階層は、WebLogic Server の config\examples\applications ディレクトリに格納されています。helloWorld、stockDemo、および travelDemo サンプルを構築する場合、config\examples ディレクトリにはサンプル アプリケーション用に作成される展開形式の war ファイルが格納され、config\examples\applications ディレクトリにはサンプル フレームワーク用に作成される jar ファイルが格納されます。

helloWorld、stockDemo、または travelDemo サンプル アプリケーションの構築、コンフィグレーション、および実行手順については、対象サンプルのソース ファイルが格納されているサンプル ディレクトリ（helloWorld、stockDemo、travelDemo）内の package-summary.html ファイルを参照してください。これらのサンプルを構築すると、サンプル フレームワークも構築されます。