Web サービスのプログラミング

Forte™ for Java™ プログラミングシリーズ

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 816-7449-10
2002 年 6 月 , Revision A

原典： *Building Web Services*
Part No: 816-4059-10
Revision A

目次

図目次

はじめに

このマニュアルでは、Forte™ for Java™ 統合開発環境 (IDE) を使用して Web サービスおよび Web サービスクライアントを構築し配備する方法を説明します。

このマニュアルは、Web サービス開発者を対象に書かれていますが、Web サービスの一般的な知識を必要とするユーザーに役立つ概念も記載しています。

このマニュアルで説明しているプログラム例は、実際に作成することができます。作業環境については、以下の Web サイトにあるリリースノートを参照してください。

`http://sun.co.jp/forte/ffj/documentation/index.html`

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

```
c:\>cd MyWorkSpace\MyPackage
```

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

```
% cd MyWorkSpace/MyPackage
```

お読みになる前に

このマニュアルを理解するには、次のような知識が必要です。

- Java™ プログラミング言語
- Enterprise JavaBeans™ (EJB™) コンポーネントモデル
- JavaServer Pages™ (JSP™) 構文
- HTML 構文
- J2EE アプリケーションのアセンブルと配備に関する概念

J2EE の基本的な概念を理解するために役立つ資料を、次に示します。

- Java 2 Platform, Enterprise Edition Blueprints
  `http://java.sun.com/j2ee/blueprints`
- *Java 2 Platform, Enterprise Edition Specification*
  `http://java.sun.com/j2ee/download.html#platformspec`
- The J2EE Tutorial (for J2EE SDK version 1.3)
  `http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html`
- Java Servlet Specification Version 2.3
  `http://java.sun.com/products/servlet/download.html#specs`
- JavaServer Pages Specification Version 1.2
  `http://java.sun.com/products/jsp/download.html#specs`

Web サービスの標準についての基本的な概念を理解するために役立つ資料を、次に示します。

- SOAP 1.1 Specification—
  `http://www.w3.org/TR/2000/NOTE-SOAP-20000508`
- Apache SOAP 2.2 Implementation of the SOAP 1.1 Specification—
  `http://xml.apache.org/soap/`
- kSOAP 1.0 (a SOAP API suitable for the Java 2 Microedition)
  `http://www.ksoap.org/`
- WSDL 1.1 Specification—
  `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`
- UDDI 2.0 Specification—
  `http://www.uddi.org/specification.html`

- JAXR 1.0. API Specification—
  `http://jcp.org/jsr/detail/93.jsp`
- Sun ONE Architecture Guide—
  `http://wwws.sun.com/software/sunone/docs/arch/index.html`

注 - Sun では、本マニュアルに掲載した第三者の Web サイトのご利用に関しましては責任はなく、保証するものでもありません。また、これらのサイトあるいはリソースに関する、あるいはこれらのサイト、リソースから利用可能であるコンテンツ、広告、製品、あるいは資料に関して一切の責任を負いません。Sun は、これらのサイトあるいはリソースに関する、あるいはこれらのサイトから利用可能であるコンテンツ、製品、サービスのご利用あるいは信頼によって、あるいはそれに関連して発生するいかなる損害、損失、申し立てに対する一切の責任を負いません。

内容の紹介

第 1 章では、Forte for Java 4 統合開発環境 (IDE) の Web サービスの標準規格および Web サービスの機能に関する概要を説明します。

第 2 章では、Web サービスを開発しテストするためのワークフローの概略を示し、Web サービス開発ツールの使用方法を説明します。また、UDDI レジストリを使用して Web サービスをほかの開発者が利用できるようにする方法についても説明します。

第 3 章では、Web サービスを使用できるクライアントの作成方法を説明します。また、Web サービスの UDDI レジストリの検索方法および検索した Web サービスを使用できるクライアントの作成方法についても説明します。

第 4 章では、XML オペレーションの作成と編集、Web サービスのブロックの構築方法を説明します。また、この作業に使用するツールについても説明します。

付録 A では、Forte for Java 4 ネイティブコネクタツールを使用して、サービス内の C++ ライブラリメソッドを利用する方法を説明しています。

付録 B では、Web サービス内のオブジェクト参照のターゲットを手動で指定する方法を説明します。

付録 C では、配備記述子の表示と編集方法について説明します。また、EJB モジュールあるいは Web サービスモジュールの配備記述子に伝達される IDE フィールドも示します。

書体と記号について

次の表と記述は、このマニュアルで使用している書体と記号について説明しています。

書体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コーディング例。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `machine_name% You have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表わします。	`machine_name% su` `Password:`
AaBbCc123 または ゴシック	コマンド行の可変部分。実際の名前または実際の値と置き換えてください。	`rm` *filename* と入力します。 `rm` ファイル名 と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』

書体または記号	意味	例
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合、バックスラッシュは、継続を示します。	`machinename% grep 'ˆ#define \` `  XV_VERSION_STRING'`
➤	階層メニューのサブメニューを選択することを示します。	作成:「返信」➤「送信者へ」

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	`machine_name%`
UNIX の Bourne シェルと Korn シェル	`machine_name$`
スーパーユーザー (シェルの種類を問わない)	#

関連マニュアル

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

オンラインで入手可能なマニュアル

次のマニュアルは、Forte for Java のポータルサイトおよび `docs.sun.com` の Web サイトから入手できます。

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

- リリースノート (HTML 形式)

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

- インストールガイド (PDF 形式)

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

- Forte for Java プログラミングシリーズ (PDF 形式)

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

  - 『Web コンポーネントのプログラミング』

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

  - 『J2EE アプリケーションのプログラミング』

    EJB モジュールや Web モジュールを J2EE にアセンブルする方法を説明しています。また、J2EE アプリケーションの配備や実行についても説明しています。

  - 『Enterprise JavaBeans コンポーネントのプログラミング』

    Forte for Java の EJB ビルダーウィザードや、他の IDE コンポーネントを使用し、EJB コンポーネント (コンテナ管理や Bean 管理の持続性の機能を持つセッション Bean やエンティティ Bean) を作成する方法を説明しています。

  - 『Web サービスのプログラミング』

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

- 『Java DataBase Connectivity の使用』

  Forte for Java の JDBC 生産性向上ツールを使用し、JDBC アプリケーションを作成する方法について説明しています。

- Forte for Java チュートリアル (PDF 形式)

  Forte for Java の Edition ごとに用意されています。Forte for Javaのツールを使用してアプリケーションを作成する方法を、順を追って説明しています。

  チュートリアルアプリケーションは、以下のサイトからもアクセスできます。

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

`docs.sun.com` (`http://docs.sun.com`) の Web サイトでは、他のサンのマニュアルの参照、印刷、購入をすることもできます。

オンラインヘルプ

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

プログラム例

Forte for Java の機能を紹介したプログラム例とチュートリアルアプリケーション (各 Edition のチュートリアルで説明されているアプリケーションを含む) を、以下の Forte for Java のポータルサイトからダウンロードすることができます。

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

Javadoc

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

ご意見の送付先

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

`docfeedback@sun.com`

電子メールのタイトルに、対象マニュアルの Part No. を明記してください。

第1章

Web サービス： 概要

このマニュアルでは、Forte™ for Java™ 4 統合開発環境 (IDE) を使用して簡単なあるいは複雑な Web サービスを構築する方法を説明しています。ここでは、IDE の使用方法について一般的な知識があることを前提としています。

Web サービスの業界標準は、発展途上です。標準を実装するソフトウェアも進化しています。Web サービスの意味は新しい技術を使い始めようとする開発者にとって混乱しやすいものになっています。この章では、Web サービスの概要について説明し、IDE の Web サービス機能を使用する方法を説明する後続の章の基礎を示します。

Web サービスとは

一般に、「Web」サービスは、標準規格に準拠し、外部からの利用が可能な広範にわたる業界の問題を解決する分散アプリケーションコンポーネントです。この項では、Web サービスに関連する問題について説明し、Web サービスの標準に関する概要を示します。

業界の問題

現代の民間企業または公的機関の基本的な問題の 1 つは、個別に開発されさまざまなソフトウェアおよびハードウェアプラットフォーム上で稼動する多様なコンピュータアプリケーションを統合する方法です。分散環境の増加により企業内のアプリケーションの機能の一部をネットワーク上で別のアプリケーションから利用できることが必要とされるようになりました。これらの機能は、企業内だけでなくビジネスパートナーなどの関連企業やサービスを提供している企業とは組織的関係または契約関係を持たない外部企業によって使用されます。

クライアントとサービス開発を分離することにより、標準の Web サービス技術に準拠するさまざまなクライアントタイプ (ブラウザベースのクライアントおよびワイヤレスクライアント) から、目的とする Web サービスへのアクセスが可能になります。ここでは、「クライアント」とは、他のサービスを含むあらゆるサービスのユーザーを意味します。

企業内のアプリケーション

従来の企業システムでは、関連する一連のアプリケーション間には、しばしばインタフェースが必要でした。例として、支払い口座、受取り口座、総勘定元帳、および部門予算を管理するソフトウェアがあります。別の例としては、人事データ、賃金台帳、利益用のソフトウェアがあります。賃金台帳システムは、一般的に源泉課税、銀行への直接預金、およびその他の企業外のシステムが提供する情報に関わることが必要です。

プログラムされたインタフェースおよび統合用の共有データの従来の使用には、関連するアプリケーションの論理およびデータ構造の調整と、ハードウェアおよびソフトウェアプラットフォームの違いを処理することが必要とされます。この手法では、企業内であってもメンテナンスの問題が発生します。1 つのアプリケーション内で変更を行うと、別のアプリケーションでいくつかの変更を行う必要が出てくる場合があります。異種コンピュータシステムに依存する企業や業界では、税法および法令報告要件の変更は、困難でコストがかかります。

企業間のアプリケーション共用

近年、インターネットコマース、資金の電子決済、経済および社会制度に必要なその他のネットワークアクティビティが急激に成長しています。生産、供給、取引は広範囲に分散し、複雑な流通経路とその追跡が必要になっています。企業は、プロセス、規則と外部企業および政府機関との関係に適合する必要があります。これらの開発により、異なるプラットフォーム間で相互運用可能であり、ネットワーク間で共用できる、変更可能なコンピュータアプリケーションの必要性が高まりました。

幅広く共用されているインターネットアプリケーションの例として、オンラインカタログがあります。多くの Web サイトではブラウザベースのインタフェースを顧客に提供しています。より複雑な例として、多くの異なるタイプの部品を多様なベンダーに注文するビジネスアプリケーションがあります。ここでの取引は、複数のサプライヤが関わる可能性があり、決定は動的に変化する価格および納期に依存し、取引に関わ

るアイテムは技術仕様に互換性がなければなりません。この方法でビジネスを行うことを可能にする企業の目的は、コンピュータベンダーおよび規格グループが Web サービスインフラストラクチャを開発する努力の背後にある主要な動機になります。

ハイパフォーマンスユーティリティプログラム

もう一つの業界要件は、アプリケーションが下位レベルプログラミング言語で記述された標準コンピュータを使用したり、集約アルゴリズムにアクセスしたりするためのものです。例には、保険計算、株式分析式、天気予報、最適ルート選択、および一般目的の統計ルーチンが含まれます。これらのプログラムの中には、非常に複雑で高度に最適化され、広範囲に確立されていて、記述し直す必要のないものがあります。Web サービスでは、これらのプログラムの機能を新しいアプリケーションに提供することができます (付録 A では、Web サービス内の C および C++ 共用ライブラリにアクセスする方法を説明しています)。

標準ソリューション

Web サービスは、分散再利用可能アプリケーションコンポーネントで選択して EJB コンポーネントなどのビジネスサービスの機能を提供し、その機能を標準インターネットプロトコルを通じてアプリケーションで利用可能にすることができます。

Web サービスアーキテクチャは、これらの要件を満たしています。

- **相互運用性**
  Web サービスコンポーネントを使用するアプリケーションは、サービスを実行するハードウェアおよびソフトウェアプラットフォームを認識する必要がありません。Web サービスは、標準の Web サービス技術を使用しているかぎり、さまざまな種類のクライアント (Web アプリケーション、ワイヤレスアプリケーション、およびその他のサービス) でアクセス可能です。
- **カプセル化**
  Web サービスコンポーネントを使用するアプリケーションは、コンポーネントの内部プログラミングの詳細に関わる必要がありません。
- **可用性**
  Web サービスの開発者は、十分な情報を含めて公開し、他の開発者が Web サービスを見つけてそれを使用できるクライアントアプリケーションコンポーネントを作成できるようにします。

Forte for Java 4 IDE は、これらの要件に合致し、1 つまたは複数のビジネスコンポーネントの下位レベルメソッドを Web サービスに適した上位レベル機能に組み込むことができます。

次の項では、Web サービスを定義する規格、技術、およびアーキテクチャについて説明します。

Web サービスの規格

Web サービスアーキテクチャは、SOAP、WSDL、UDDI の 3 つの関連規格に基づいています。規格ドキュメントは長文ですが、この 3 つのアーキテクチャは一般に次のように説明されます。

- SOAP (Simple Object Access Protocol)
  Web サービスが呼び出されるメカニズムとデータが返される方法を定義します。SOAP クライアントは SOAP サービス上で、メソッドの呼び出し、XML 形式でのオブジェクトのやりとりを行うことができます。
- WSDL (Web Service Definition Language)
  Web サービスの外部インタフェースを記述して開発者がサービスを使用できるクライアントを作成できるようにします。
- UDDI (Universal Discovery, Description, and Integration) レジストリ
  Web サービスに関する情報を持っています。これには WSDL ファイルの場所および実行サービスの場所が含まれるので、この情報を使用して、開発者がサービスにアクセスするクライアントを作成して Web サービスの機能をアプリケーションに組み込むことができるようになります。

注 - Web サービスの実装とは、一般に XML ドキュメントを HTTP 上で渡すことです。このマニュアルを理解するには、XML および HTTP の知識が必要になります。

注 - Web サービス用語は、技術とともに変化します。たとえば、WSDL という用語は頭字語ですが、名詞としても使われ、Web サービスの記述を意味します。SOAP の頭字語は使用されていますが、SOAP プロトコルは「略語」でなくなり、最近の規格草案では、頭字語として使用しないように書かれています。このマニュアルでは一般的な業界での使用に従っています。また、用語の意味があいまいな場合には意味を説明しています。

SOAP

SOAP (Simple Object Access Protocol) は、W3C (World Wide Web Consortium) 規格であり、XML を使用したオブジェクト受け渡し用のプロトコルを定義します。SOAP 実行システム (SOAP 規格の実装) を使用すると、クライアントが SOAP 利用可能サービス上でメソッドを呼び出して XML 形式でオブジェクトを渡すことができます。

次の概要は W3C SOAP 1.1 仕様からの抜粋です。詳細については、`http://www.w3.org/TR/2000/NOTE-SOAP-20000508`を参照してください。

「SOAP は、情報を分散環境で交換するための軽量プロトコルです。XML ベースのプロトコルで、次の 3 つで構成されます。メッセージの内容と処理方法を記述するフレームワークを定義するエンベロープ、アプリケーション定義データ型のインスタンスを表すエンコーディング規則セット、およびリモートプロシージャの呼び出しと応答を表す規則。SOAP は、その他の多様なプロトコルと組み合わせて使用できますが、このマニュアルに定義されている唯一のバインディングでは、SOAP を HTTP および HTTP Extension Framework と組み合わせて使用する方法を説明しています。」

SOAP RPC メッセージのペイロードには、アプリケーションデータ (オブジェクトおよびプリミティブ型) がシリアル化され XML として含まれています。

WSDL

WSDL (Web Service Description Language) は、W3C 規格の Web サービスの外部インタフェースを記述するために使用される XML ベース言語です。

WSDL ファイルへのリンクを提供することにより、ほかのユーザーが Web サービスを利用できるようにします。この章で後述する UDDI レジストリは、Web サービスに関する情報を公開する場所を提供します。UDDI レジストリを検索してサービスを使用することを決めた開発者は、サービスおよび WSDL ファイルの場所を検出でき、この情報を使用してリモートリクエストをサービスに発行することができる SOAP クライアントを作成できます。

次の概要は W3C WSDL 1.1 仕様からの抜粋です。詳細については、`http://www.w3.org/TR/2001/NOTE-wsdl-20010315`を参照してください。

「WSDL は、ネットワークサービスを、ドキュメント指向またはプロシージャ指向情報のいずれかを含むメッセージ上で動作するエンドポイントセットとして記述する XML 形式です。操作とメッセージは抽象的に記述され、エンドポイントを定義するために具象ネットワークプロトコルおよびメッセージ形式にバインドされます。関連す

る具象エンドポイントは抽象エンドポイント (サービス) に組み込まれます。WSDL は、エンドポイントおよびそのメッセージを、通信に使用されているメッセージ形式またはネットワークプロトコルに関わらず、記述できるように拡張できます。」

UDDI

UDDI (Universal Description, Discovery, and Integration) は、Web サービスに関する情報を含む Web ベースのレジストリ用のプロトコルです。

UDDI レジストリは「public」、「private」、または「hybrid」を使用できます。これらの記述用語は、技術的な区別ではありませんが、レジストリの範囲および使用方法が異なります。

- public レジストリは、インターネット上で幅広く利用できる
  レジストリの所有者はほかの開発者 (競争相手を含む) がサービスをレジストリ内で公開することにより、一般ユーザーがレジストリを検索してレジストリエントリをダウンロードできるようにすることができます。
- private レジストリは、1 つの企業に制限される
  この場合、レジストリによって企業内のビジネスコンポーネントの共用が容易になります。
- hybrid レジストリは、1 つの企業の範囲を超えて利用できるが、制限がある
  たとえば、レジストリはビジネスパートナーまたは業界内の指定開発グループが利用できるようにすることができます。レジストリホストは、レジストリ内でサービスを公開できるユーザーとレジストリを検索できるユーザーを判別します。

Web サービスのレジストリエントリには十分な情報を含めて、開発者がサービスにバインドしてメソッドを呼び出すことができるクライアントアプリケーションを作成できるようにします。「クライアント」は相対的な用語で、Web サービスへの呼び出しを発行するコンポーネントを指します。レジストリエントリはサービスを記述する WSDL ファイルの場所 (クライアントの作成に必要) および実行サービスの場所 (クライアントの実行に必要) を含んでいる必要があります。

レジストリエントリには、サービスおよびプロバイダに関する追加情報を含めることができます。たとえば、潜在ユーザーは、サービスをサポートするためにプロバイダがどのようにコミットしているか、プロバイダが「証明書」を持っているか、またサービスのソースとして信頼できるかどうかを知りたい場合があります。この問題は Web サービスの問題ではなく、ソースの信頼性は SSL ベースの証明によるもので、特

定の数社が証明書認可機関として広く認識されています。同じことが従来の技術にもあてはまり、米国のクレジットカード会社または米国政府を例にとると、National Bureau of Standards がその例です。

public レジストリは、数千または数百万のサービスに成長する可能性があります。そのため、UDDI 規格はサービスの分類をサポートしています。業界、地域、製品、サービスなどのカテゴリに基づいて、さまざまな分類がすでに利用可能です。UDDI レジストリ実装により、サービスの公開およびレジストリ検索用の機能が提供されます。

UDDI 仕様ドキュメント、論文、記事、および参加企業のリストについては、`http://www.uddi.org`を参照してください。Sun Microsystems, Inc. も UDDI プロジェクトに参加している企業の 1 つです。

Web サービスの公開と使用

図 1-1 は、UDDI レジストリを通じて、Web サービスを利用可能にする方法を示しています。この例では、P 社が Web サービス提供側で、R 社が Web サービス要求側です。また private レジストリを使用して P 社および R 社を 1 つの会社内の部門として考慮することができます。

1. P 社は Web サービスおよび外部インタフェースを記述する WSDL ファイルを作成します。P 社は Web サービスを UDDI レジストリに公開して、WSDL ファイルの場所や実行 Web サービスの場所を含めた Web サービスに関する情報を提供します。

   P 社は、Web サービスの実行とネットワーク上で利用可能にすることに責任を持ちます。

2. R 社は、UDDI レジストリ内で特定の機能を提供する Web サービスを検索して、P 社が作成した Web サービスを選択します。

   多くの検索条件が考えられます。たとえば、R 社は特定のバージョンの Web サービスを検索する場合があります。2 社がすでに一緒に機能していて、P 社が R 社に特定の Web サービス用の UDDI レジストリ識別子を提供している場合などには、検索が詳細になります。

3. R 社は、WSDL 記述および Web サービスに関するその他の情報を使用して、Web サービスへの要求を行うクライアントコンポーネントを作成します。R 社はクライアントをアプリケーションに組み込みます。

4. R 社は、アプリケーションを実行します。実行時に、クライアントコンポーネントは Web サービスにバインドして SOAP 要求を実行し、データを XML として HTTP 上で渡します。

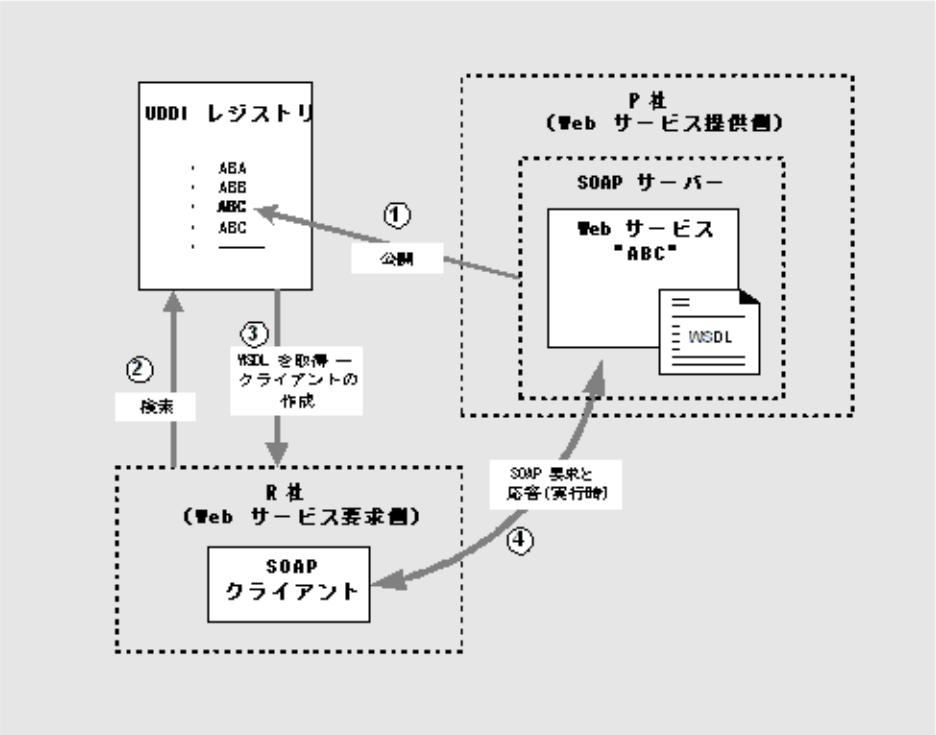

図 1-1　Web サービスの公開と使用

このプロセスには、次のタスクが含まれます。

1. P 社は、UDDI レジストリホストからレジストリを記述する許可を取得します。P 社は名前、場所情報、およびビジネス分類情報を含むビジネスエントリを通じてレジストリ内に自身を確立します。

2. P 社は、レジストリ内に Web サービス用の WSDL をポイントする URL を持つエントリを作成します。このタイプのレジストリエントリは、技術モデル (*tModel*) と呼ばれます。

3. P 社は、レジストリ内に tModel と関連付けられた 1 つまたは複数のサービスエントリを作成します。各サービスエントリは、Web サービスの実行インスタンスをポイントする URL を持ちます。たとえば、P 社、ビジネスパートナー、一般ユー

ザー、テストで使用する 4 つの異なる実行インスタンスがあります。すべてのインスタンスは同じ機能、同じ入出力を持ちますが、パフォーマンスおよび可用特性は異なります。

4. R 社は、レジストリ内の tModel およびサービスエントリからの情報を使用してクライアントを作成します。tModel は WSDL へのポインタを備え、Web サービスの外部インタフェースに関する構造的な情報を持ちます。サービスエントリは Web サービスの実行インスタンスをポイントし、サポート、パフォーマンス、および特定の実行インスタンスに対して P 社によって提供される可用性に関する追加情報を持ちます。
5. R 社は、この情報を使用して指定 Web サービス用のさまざまなクライアントを作成して、カスタマイズします。

Forte for Java Web サービス

Forte for Java 4 Web サービスは J2EE コンポーネントとして実装されます。Web サービスはビジネスコンポーネントとともに単一のアプリケーションにパッケージ化します。

Forte for Java 4 IDE は、次の主要な機能をコーディングなしでサポートします (ビジネスコンポーネントが利用可能な場合)。次に示す機能およびその他の機能については、このマニュアルの後続の章で説明しています。

- SOAP RPC Web サービスの作成
- サービスを記述する WSDL の生成
- サービス用の SOAP クライアントの生成
- 生成されたクライアントおよび JSP コンポーネントを使用したサービスの IDE 内でのテスト
- サポートされているアプリケーションサーバーへのサービスのアセンブルと配備
- UDDI レジストリへのサービスの公開
- Web サービス用の UDDI レジストリの検索
- SOAP クライアントの外部 Web サービス記述 (WSDL ファイルまたは UDDI レジストリエントリ) からの生成

Forte for Java 4 IDE 内で自動的に生成されたエントリをカスタマイズできます。

Forte for Java 4 Web サービスは、ビジネスコンポーネントのメソッドへの直接呼出しを行います。「XML オペレーション」への呼び出しを行うこともできます。XML オペレーションは、複数のビジネスメソッド呼び出しを単一の上位レベルメソッドにカプセル化します。この機能は、この章でアーキテクチャとして説明しています。使用方法は第 4 章で説明しています。

多層アーキテクチャ

図 1-2 は、Forte for Java 4 IDE を使用して開発した Web サービスアプリケーションの主要コンポーネントを示しています。Web サービスは、Web と J2EE アプリケーションサーバーの EJB コンテナを実行時につなぐ論理エントリであり、多層アーキテクチャを備えています。

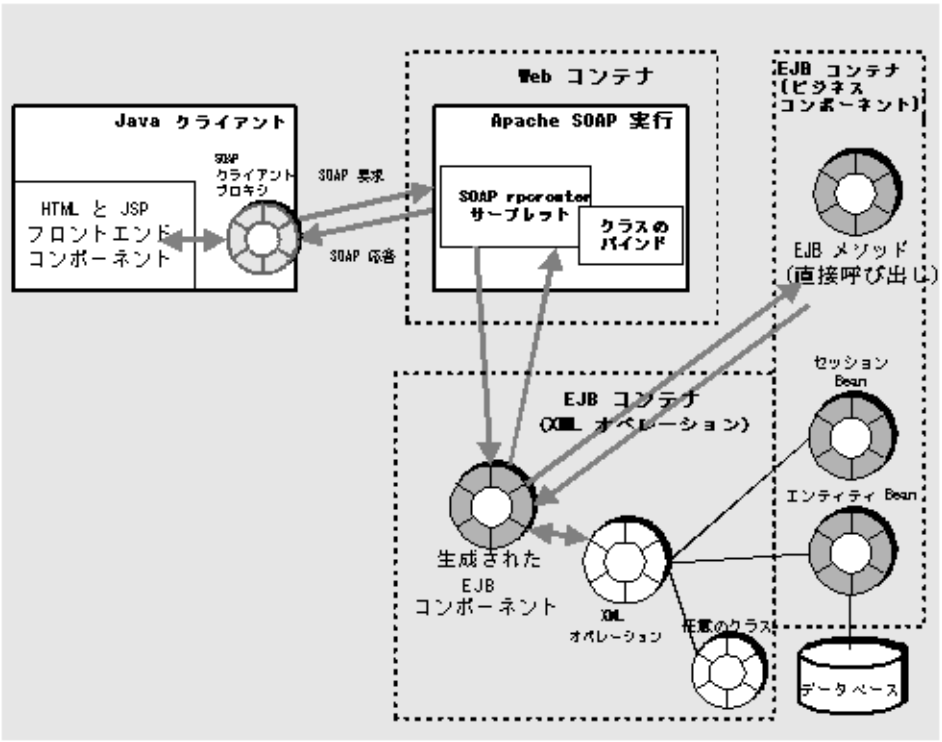

図 1-2　Forte for Java 4 Web サービスおよびクライアント (多層モデル)

クライアント側に、「SOAP クライアントプロキシ」があります。これは、サーバー上の SOAP 実行システムと通信するためにクライアント上の SOAP 実行システムを使用して SOAP 要求を送信し、SOAP 応答を受信する Java クラスです。SOAP 要求は

XML ラッパーで、Web サービス上のメソッド呼び出しおよび入力データをシリアル化形式で含みます。SOAP 応答は XML ラッパーで、メソッドの戻り値をシリアライズ形式で含みます。クライアント側には、SOAP クライアントプロキシへの呼び出しを行い、戻り値を処理するフロントエンドクライアントコンポーネントがあります。このコンポーネントは、HTML および JSP ページで構成されるか、Java Swing コンポーネント、別の種類のクライアント、または別のサービスを使用できます。

サーバー側では、Apache SOAP 実行 (SOAP 1.1 標準の実装) は、Web コンテナ内にあり、クライアントと Web サービス間でやりとりする SOAP メッセージを処理します。

SOAP 要求は、生成された EJB コンポーネント (IDE によって Web サービス用のインフラストラクチャとして自動作成されたセッション Bean) 上でメソッド呼び出しに変換されます。生成されたセッション Bean は、EJB コンテナ内のビジネスサービス上でメソッド呼び出しを行うか、XML オペレーションを呼び出します。

Web 主体アーキテクチャ

IDE で新しい Web サービスを作成するときに、アーキテクチャを選択できます。多層アーキテクチャ (デフォルト。図 1-2 に図示) または「Web 主体アーキテクチャ」を選択できます。手順については、20 ページの「SOAP RPC Web サービスの作成」を参照してください。

多層アーキテクチャで作成した Web サービスは、アプリケーションサーバー内のコンポーネント上でだけビジネスメソッドを呼び出すことができます。Web 主体アーキテクチャで作成した Web サービスは、Web 層ビジネスコンポーネントおよび EJB ビジネスコンポーネント上でメソッドを呼び出すことができます。Web 主体 Web サービスが Web 層ビジネスコンポーネントだけを使用している場合は、サービスは Web サーバーに配備できるので、アプリケーションサーバーは必要ありません。

Web 主体アーキテクチャを選択すると、EJB セッション Bean ではなく Java クラスに基づき、IDE によってサポート構造が生成されます。このため、サポート構造が Web コンテナ内で実行できるようになります。

注 - Web 主体アーキテクチャはより柔軟性があります。これは EJB ビジネスコンポーネントと Web 層ビジネスコンポーネントの両方をサポートできるためです。アプリケーションが EJB ビジネスコンポーネントだけを使用している場合は、多層アーキテクチャを選択します。ビジネス EJB コンポーネントと同じコンテナにローカルな生成されたサポート構造を持つと、より効率的です。

アプリケーションアセンブリおよび配備には、3 つの代替方法があります。

- 多層アーキテクチャでは、IDE はアプリケーションサーバーへの配備用の EAR ファイルを作成します。
- Web ビジネスコンポーネントだけを持つ Web 主体アーキテクチャでは、IDE は Web サーバーへの配備用の WAR ファイルを作成します。
- EJB コンポーネントを持つ Web 主体アーキテクチャでは、IDE はアプリケーションサーバーへの配備用の EAR ファイルを作成します。

詳細については、28 ページの「Web サービスのアセンブルと配備」を参照してください。

ソフトウェアバージョン

Web 規格および技術は、発展途上です。Forte for Java 4 IDE は次のバージョンの各規格およびソフトウェア実装を使用します。

- SOAP 1.1 specification
  `http://www.w3.org/TR/2000/NOTE-SOAP-20000508`
- Apache SOAP 2.2 implementation of the SOAP 1.1 specification
  `http://xml.apache.org/soap/`
- kSOAP 1.0 (a SOAP API suitable for the Java 2 Microedition
)`http://www.ksoap.org/`
- WSDL 1.1 specification
  `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`
- UDDI 2.0 specification
  `http://www.uddi.org/specification.html`
- JAXR 1.0. API specification
  `http://jcp.org/jsr/detail/93.jsp`
- J2EE 1.3 コンポーネント
- J2EE 1.3 Reference Implementation

XML オペレーション

IDE は、既存のアプリケーションコンポーネントの Java メソッドを展開する Web サービスを簡単に作成する手助けとなります。ただし、EJB コンポーネントの中には Web サービスを通じて外部から利用するように設計されていないものがあります。追加 Java コンポーネントをプログラミングするか、XML オペレーションを作成して、Web サービスインタフェースレイヤーを作成する必要があります。

注 - 単にビジネスコンポーネントの個々のメソッドが Web サービスとして利用される場合は、XML オペレーションをバイパスできます。この手順は、第 2 章で説明します。

Forte for Java 4 IDE ユーザーインタフェースを使用すると、コーディングを追加することなく既存の J2EE コンポーネントに基づいて複雑なアプリケーションを作成できます。次のようなことができます。

- 異なるタイプの複数ビジネスコンポーネントの機能を組み合わせるWeb サービスを作成します。
- ビジネスコンポーネント上で複数メソッドを希望の順序で選択して呼び出し、戻り値を 1 つのメソッドから別のメソッドに渡す Web サービスを作成します。

EJB コンポーネントは、コンポーネントの内部論理を理解して管理できるクライアントによって使用されることを想定して設計されます。このメソッドは、Web サービスとして直接利用されるには論理設計上のレベルが下位であるか、または目的のオブジェクトを目的の形式で返すには、異なるコンポーネント上のいくつかのメソッド呼び出しを必要とする場合があります。

Web サービスモデルで設計された新しいビジネスコンポーネントは、適切な上位レベル機能のみを提供するメソッドを持っている場合があります。ただし、既存のビジネスコンポーネントまたは Web サービスの基本として設計されていないコンポーネントは、必要な上位レベルメソッドを持っていない場合があります。

Forte for Java 4 IDE は、この問題を仲介の役割を果たす「XML オペレーション」を提供することによって解決します。XML オペレーションは、複数の EJB メソッドを Web サービスに適した単一の上位レベルビジネス機能に結合することができます。第 4 章 で XML オペレーションを作成する方法を説明します。

XML オペレーションとは ?

XML オペレーションは、いくつかのビジネスメソッドをカプセル化することができます。外部クライアントに対しては、オペレーションは Web サービス内への単一の RPC 呼び出しのように見えます。

XML オペレーションをコードレスエディタを使用して定義します。第 4 章 では、XML オペレーションの機能とエディタの使用方法を説明します。

XML オペレーションは、特定の Web サービス要求が処理される方法を指定する論理エンティティです。すべての XML オペレーションを作成して、ほかのコンポーネントのビジネスメソッドとともに Web サービスに追加します。次に Web サービスの実行クラスを IDE 内に生成します。これにより、EJB セッション Bean および各 XML オペレーションに対して 1 つのクラスが作成されます。クライアントが Web サービスに要求を送信すると、要求はセッション Bean 上でメソッド呼び出しに変換されます。要求はビジネスメソッドへの単一の直接呼び出しとして、または XML オペレーションによって定義されるメソッド呼び出しの複雑なセットとして処理されます。

注 - 単一の XML オペレーション内のすべてのメソッドは同じ状態にあり、オペレーションが継続している間、データを共有できます。

要約：

- Forte for Java 4 Web サービス内の XML オペレーションは、上位レベルビジネスメソッドの役割を果たす
- RPC 呼び出しは、Web サービスの一部となる生成された EJB コンポーネント内のメソッドとして実装される
- 生成された EJB コンポーネントは、ビジネスコンポーネントのさらに下位レベルコンポーネントメソッドに対するクライアントとして機能する
- 各 XML オペレーション要求は単一のトランザクション内で実行され、単一のユニット内で複数 EJB メソッド呼び出しのグループ化を可能にする

Forte for Java 4 XML オペレーション機能は、手動コーディングと比較して次の主な利点があります。

- コードレスエディタを使用するので、Java 言語のより詳細な知識がなくてもアプリケーションコンポーネントと Java メソッドの動作に関する知識があれば十分に XML オペレーションを構築できる

- 手動でステートレスセッション Bean を設計、コーディング、テストする必要がなく、既存のビジネスコンポーネントに対するサービスインタフェースを簡単に変更できる
- コードレスエディタで XML オペレーションを編集し、サービスコンポーネント用の実行クラスを再生成することによってサービスインタフェースを変更できる

要求応答メカニズム

Forte for Java 4 Web サービスは、簡易なものと、複雑なものの 2 つの機能を提供します。ビジネスコンポーネントへの直接メソッド呼び出しと XML オペレーションの 2 つのが、それに相当します。Web サービスを使用しているクライアント側から見ると、XML オペレーションがいくつかのメソッド呼び出しをカプセル化しているため、この 2 つの機能は類似しています。各 XML オペレーションは、特定のクライアント要求メッセージに対する応答を定義します。Web サービス開発者は、XML オペレーションを定義して既存のコンポーネントから生成します。

Web サービスがクライアント要求を受信すると、要求を XML ドキュメント (XML 入力ドキュメント) の形式で適切な XML オペレーションに転送します。XML オペレーションは、ビジネスコンポーネント上で 1 つまたは複数のメソッドを呼び出し、これらのメソッド呼び出しの戻り値を XML ドキュメント (XML 出力ドキュメント) に変換してドキュメントをクライアントに返します。

XML オペレーションが実行されると、Web サービスは次を実行します。

1. XML 入力ドキュメントを解析し、XML オペレーションが呼び出すように定義されているメソッドのパラメータにドキュメントの要素をマッピングする
2. XML オペレーションに定義されているメソッドを指定された順序で呼び出す

メソッドからの戻り値は、別のメソッドへの入力として渡すことができ、高度なオペレーションを構築できます。

3. メソッドの戻り値を XML オペレーションの定義に従って XML 出力ドキュメントにフォーマットする
4. XML 出力ドキュメントを返す

たとえば、図 1-3 は、3 つの異なるオブジェクト上でメソッドを呼び出す `ProductName` という XML オペレーションを示しています。

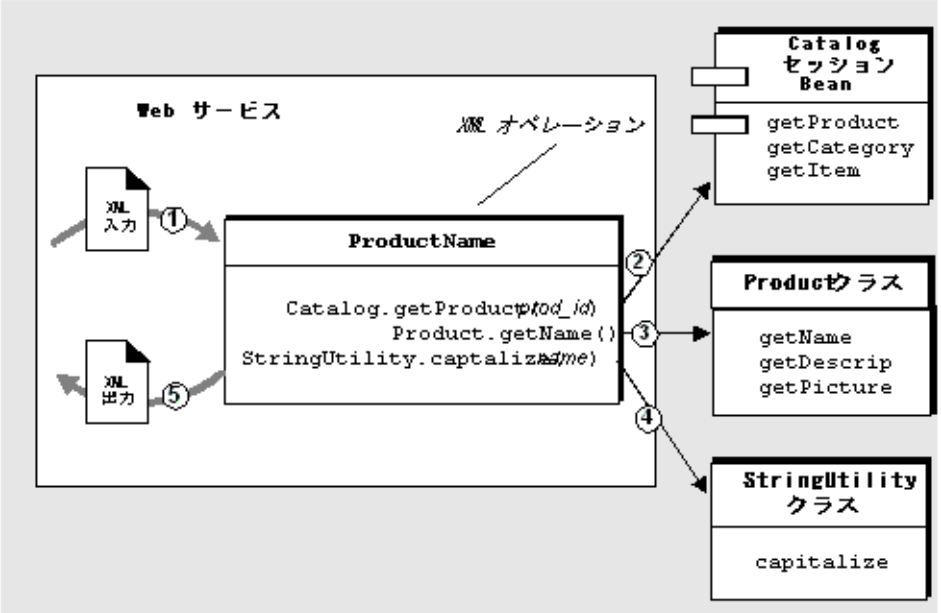

図 1-3　クライアント要求を満たすために複数のメソッドを呼び出す XML オペレーション

`ProductName` XML オペレーションは、製品 ID を要求パラメータとして受け取り、対応する製品名を大文字で返します。実行時には次の処理が行われます。

1. ドキュメントの「prod_id 要素」の値を `Catalog.getProduct` メソッドへの入力パラメータとして使用し、XML 入力ドキュメントを解析する
2. `Catalog.getProduct` メソッドを呼び出し、メソッドは `Product` クラスのインスタンスを返す
3. `getName` メソッドを `Product` オブジェクト上で呼び出し、メソッドは製品名を含む `String` オブジェクトを返す
4. Static (静的) メソッド `StringUtility.capitalize` を呼び出し、製品名をパラメータとして含む `String` オブジェクトを渡す。このメソッドは、最初に大文字でフォーマットされた製品名を含む `String` オブジェクトを返す。
5. 大文字で示された製品名を XML ドキュメントとして含む `String` オブジェクトを、フォーマットして返す

第2章

Web サービスの構築

この章では、Web サービスの開発に使用できるツールおよび手順について説明します。この手順では、エンタープライズ Bean など、メソッドを Web サービスとして利用するビジネスコンポーネントが用意されていることを前提としています。

Web サービス開発タスク

次のタスクは相互依存していますが、全体が順序だてられているわけではありません。たとえば、Web サービスを作成しながらメソッド参照を追加することもできますし、後から追加することもできます。Web サービス開発は反復プロセスです。

タスクの一般的な開発ワークフロー順序は、次のとおりです。

- SOAP RPC Web サービスの作成
- XML オペレーションの開発 (必要に応じて)
- Web サービスへのメソッド参照の追加
- 実行時オブジェクトへの参照の解決
- 環境エントリの追加
- 実行時クラスの生成
- J2EE アプリケーションとしての Web サービスのアセンブルと配備
- テストクライアントの作成
- Web サービスのテスト
- UDDI レジストリの使用

SOAP RPC Web サービスの作成

新しい Web サービスを作成する手順は、次のとおりです。

1. 「新規 Web サービス」ウィザードを開きます。

   エクスプローラで、Web サービスを作成する Java パッケージをマウスの右ボタンでクリックし、「新規」>「Web サービス」>「Web サービス」を選択します。

   または、IDE のメインウィンドウから「ファイル」 >「新規」>「Web サービス」>「Web サービス」を選択して同じウィザードを開くことができます。

   図 2-1 は、「新規 Web サービス」ウィザードを示しています。

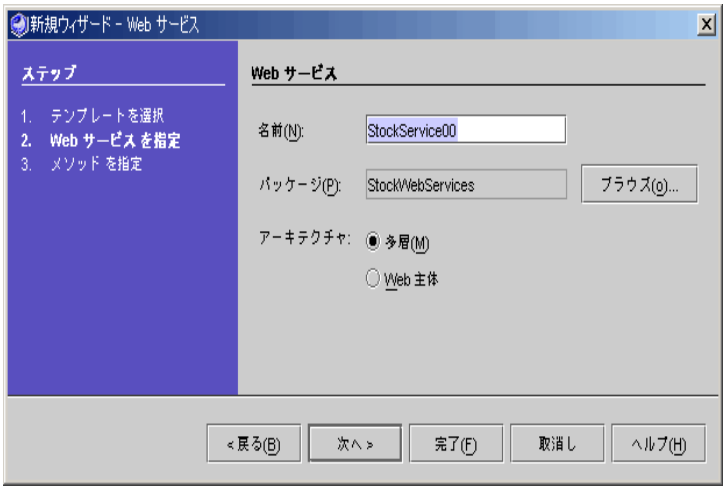

図 2-1　「新規 Web サービス」ウィザード

注 - このウィザードには、Web サービスアーキテクチャを選択できるラジオボタン「多層 (デフォルト)」と「Web 主体」があります。2 つのアーキテクチャの違いは、10 ページの「Forte for Java Web サービス」 で説明します。アプリケーションが EJB ビジネスコンポーネントだけを使用している場合は、両方のアーキテクチャを使用できます。ビジネスコンポーネントと同じ EJB コンテナに生成されたサポート構造が配置されるため、「多層」のほうが適しています。

---

2. 「パッケージ」フィールドに Web サービスを作成する場所が正しく指定されていることを確認します。

   「新規 Web サービス」ウィザードを開いたときに、エクスプローラで Java パッケージが選択されている場合は、パッケージの名前がウィザードの「パッケージ」フィールドに表示されます。目的のパッケージを見つけて選択するには、ウィザードの「ブラウズ」ボタンをクリックします。

3. 「名前」フィールドで、Web サービスの名前を入力します。

   ウィザードで指定されているパッケージ内の既存の Web サービスの名前を入力すると、強調表示された次のようなメッセージが表示されます。「この名前の付いた Web サービスがパッケージ内に既に存在します。」

4. 「完了」をクリックして、Web サービスを作成するか、「次」をクリックしてメソッド参照を追加します。

   「完了」をクリックすると、新しい Web サービスがエクスプローラに表示されます。サービスは、指定パッケージにノードとして青の球形アイコン () 付きで表示されます。

   「次へ」をクリックすると、「メソッド」ブラウザが表示されます。ブラウザ内で 1 つまたは複数のメソッドを選択して「完了」をクリックし、Web サービスにメソッド参照を追加します。次に「完了」をクリックして Web サービスを作成します。

   Web サービスを作成した後からメソッド参照を追加することができます (22 ページの「メソッド、XML オペレーション、JAR ファイルへの参照の追加」を参照してください)。

5. エクスプローラで、Web サービスをマウスの右ボタンでクリックし、「プロパティ」を選択します。

   「SOAP RPC URL」プロパティは、次の形式になっています。

   `http://`*`hostname`*`:`*`portnum`*`/`*`webserviceName`*`/servlet/rpcrouter`

   `hostname` および `portnum` の値をサーバーインストールと一致するように設定します。*`webserviceName`* の値を変更します。この値はコンテキストルートまたは「Web コンテキスト」と呼ばれます。この値は、後の手順で作成する J2EE アプリケーション WAR ノードの Web コンテキストプロパティと一致している必要があります (30 ページの「J2EE アプリケーションのアセンブル」を参照してください)。

注 - Web サービスが複数ある場合は、各 SOAP RPC URL が固有のものであることを確認してください。

図 2-2 は、SOAP RPC URL のコンテキストルートがデフォルト値から `StockApp` に変更されている Web サービスのプロパティを示しています。

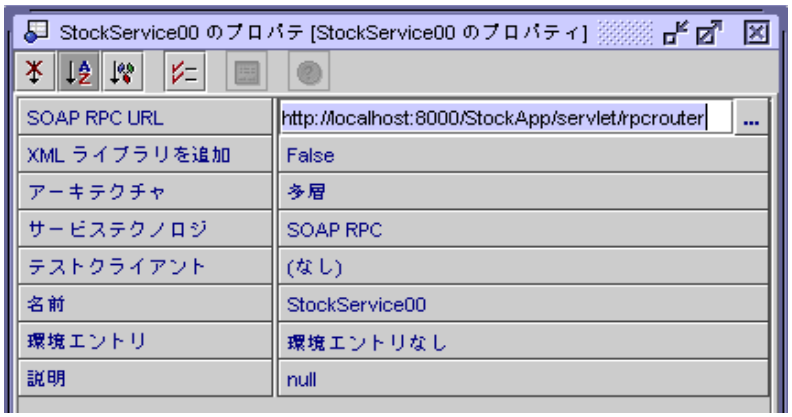

図 2-2　コンテキストルートが `StockApp` に設定されている Web サービスプロパティ

XML オペレーションの開発

XML オペレーションは、複数のビジネスメソッド呼び出しを Web サービス用の単一の上位レベルメソッドに統合する手段です。単にビジネスコンポーネントの個々のメソッドを Web サービスとして利用する場合は、XML オペレーションは必要ありません。

XML オペレーションの役割についての詳細は、14 ページの「XML オペレーション」を参照してください。XML オペレーションを開発する方法についての詳細は、第 4 章を参照してください。

メソッド、XML オペレーション、JAR ファイルへの参照の追加

注 - エンタープライズ Bean や、メソッドが、Web サービスから利用されるビジネスコンポーネントとして用意されている必要があります。

Web サービスに参照を追加する手順は、次のとおりです。

1. エクスプローラで、Web サービスをマウスの右ボタンでクリックし、「参照を追加」を選択します。

   ファイルブラウザが表示されます。

2. 目的のビジネスメソッド、XML オペレーション、および JAR ファイルを選択します。

   複数の項目を選択するには、Ctrl キーを押しながら項目をクリックします。

   EJB Bean クラス、ホームインタフェース、またはリモートインタフェースを表すノードでなく、論理 EJB ノード (Bean アイコン付きのノード) の EJB メソッドを選択してください。論理 EJB ノードからメソッドを追加することによって、メソッドを呼び出すために必要な実行時情報を提供できます。図 2-3 は、論理 EJB ノードの例を示しています。

図 2-3　エクスプローラの論理 EJB ノード

3. 「了解」をクリックします。

   Web サービスに追加された参照は 図 2-4 に示されているように Web サービスのサブノードに表示されます。

   参照の構成は次のとおりです。

   - メソッドは、**メソッド**ノードの下
   - XML オペレーションは、XML **オペレーション**ノードの下
   - JAR アーカイブファイルは、**ライブラリ**ノードの下

図 2-4　Web サービス参照

IDE は、参照オブジェクトを含むパッケージの名前を持つ参照に接頭辞を付けます。XML オペレーション参照および JAR 参照の前に矢印記号 (->) が付加されます。

注 - 「コピー」および「ペースト」コマンドを使用して Web サービスに参照を追加することもできます。たとえば、EJB メソッドを選択して「編集」>「コピー」を選択し、Web サービスノードを選択して「編集」>「ペースト」を選択します。

Web サービスからの参照の削除

参照を削除する手順は、次のとおりです。

1. エクスプローラで、削除する参照を選択します。
2. 参照をマウスの右ボタンでクリックし、「削除」を選択します。

   参照が削除されます。

実行時オブジェクトへの参照の解決

このタスクについての詳細は、58 ページの「オブジェクトのインスタンス化と参照の解決」を参照してください。この手順は、XML オペレーションを使用している場合や、デフォルト参照で正常に動作しない場合にだけ必要になります。

環境エントリの追加

環境エントリは、Web サービスの EJB モジュール配備記述子に格納するデータです。環境エントリは、Web サービスでターゲットオブジェクトを作成または検出するメソッドに対するパラメータとして使用するために提供されています。

「オブジェクト参照を解決」ダイアログの「メソッド」、「コンストラクタ」、「Static メソッド」フィールドに指定されているメソッドでパラメータを指定できる場合は、パラメータの値を環境エントリにマップできます。環境エントリは配備記述子内に格納されているため、配備時に実行時環境に適した値に設定できます。

環境エントリを追加する手順は、次のとおりです。

1. Web サービスノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。

図 2-5　Web サービスのプロパティ

2. 「環境エントリ」プロパティをクリックします。

   省略記号 (...) ボタンが表示されます。図 2-5 は、諸略記号 (...) ボタン付きの「エントリ」プロパティを示しています。

3. 省略記号 (...) ボタンをクリックします。

   図 2-6 に示されている「環境エントリ」ダイアログが表示されます。

図 2-6　「環境エントリ」ダイアログ

4. 「追加」をクリックします。

   図 2-7 に示されている「環境エントリの追加」ダイアログが表示されます。

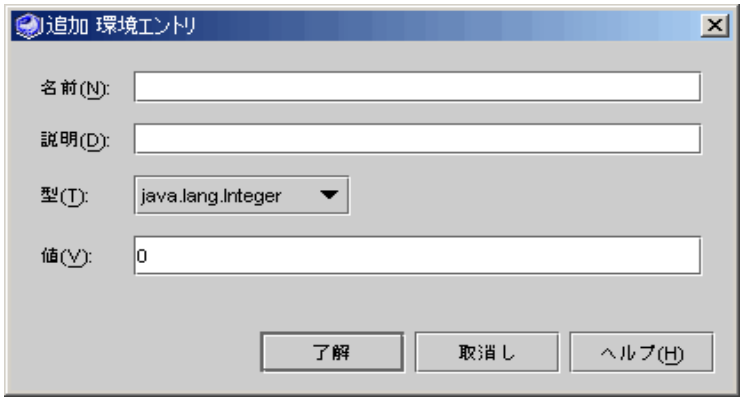

図 2-7　「環境エントリの追加」ダイアログ

5. 「環境エントリの追加」ダイアログ内の各フィールドに次の項目を入力します。

   - **名前**
     環境エントリに名前を割り当てます。
   - **説明**
     環境エントリの説明を入力します。
   - **型**
     環境エントリをマップするメソッドパラメータの型を入力します。
   - **値**
     環境エントリの値を割り当てます。

6. 「了解」をクリックします。

   環境エントリが作成されます。環境エントリは次に Web サービスの実行時クラスを生成するときに、Web サービス EJB モジュール配備記述子に伝達されます。

実行時クラスの生成

Web サービスを J2EE アプリケーションとして組み立て、テスト用に配備するには、Web サービスの実行時クラスを生成する必要があります。

アーキテクチャが多層の場合は、図 2-8 に示されているように Web サービスを実装する 4 つのクラスが生成されます。そのうち 3 つは生成された EJB コンポーネント用のクラスです。

アーキテクチャが Web 主体の場合は、Web サービスを実装する 2 つのクラスが生成されます。

XML オペレーションがある場合は、それぞれに対して 1 つの追加クラスが作成されます。

Web サービスの実行クラスを生成する手順は、次のとおりです。

1. エクスプローラで Web サービスを選択します。
2. マウスの右ボタンでクリックして、「Java ファイルの生成/コンパイル」を選択します。

   Web サービスの実行クラスが生成され、コンパイルされます。クラスファイルは、このディレクトリ構造内に置かれます。

   - 各 XML オペレーションクラスがある場合は、対応する XML オペレーションと同じパッケージに置かれます。
   - Web サービスクラスは、Web サービスと同じパッケージに置かれます。

図 2-8　ノードおよび参照の Web サービス階層

図 2-8 は、Web サービス用の実行クラスを生成した後のノードおよび参照の階層を示しています。この例の Web サービスは多層アーキテクチャであるため、4 つのクラスが生成されます。

XML オペレーションの詳細については、第 4 章を参照してください。

Web サービスのアセンブルと配備

この節の説明は、Web サービスで必要な EJB モジュールが Forte for Java 4 IDE で利用可能であり、Web サービスの実行クラスを生成していることを前提としています。

配備手順は、J2EE リファレンス実装用です。配備は他のアプリケーションサーバーと類似していますが、ホスト名、ポート番号、およびその他の特殊要件についてはアプリケーションサーバーの担当者に問い合わせる必要があります。

Web サービスが Web 主体アーキテクチャで EJB ビジネスメソッドを参照しない場合、アセンブルおよび配備手順は、簡略化されます。次の説明は、簡単な Web 主体ケースから始まり、J2EE アプリケーションとしてアセンブルおよび配備される Web サービス用の手順へと続きます。

Web 主体アプリケーション

Web サービスを Web 主体アーキテクチャで作成する場合や、Web サービスが EJB ビジネスコンポーネントを持たない場合は、WAR ファイルだけが作成されます。この場合、J2EE アプリケーションを作成する代わりに、次のような簡略化手順に従ってください。

- Web サービスノードをマウスの右ボタンでクリックし、「実行」を選択します。

  Web サービスが Web 主体アーキテクチャで EJB メソッドを参照しない場合は、「実行」メニュー項目のみがアクティブになります。

  または、次の手順で行うこともできます。

1. Web サービスノードをマウスの右ボタンでクリックし、「WAR ファイルをアセンブル」を選択します。

   IDE によって、Web サービスと同じパッケージ内に WAR ファイルが作成され、エクスプローラに WAR ノードが表示されます。

2. WAR ファイルをマウスの右ボタンでクリックし、「 Web モジュールとしてアンパック」を選択します。

   IDE によって、WAR ファイルが選択したディレクトリに配置され、そのディレクトリが IDE ファイルシステムにマウントされます。

3. エクスプローラで、パッケージ解除した Web モジュールに移動し、`WEB-INF` ノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。

4. 「コンテキストルート」プロパティを、Web サービスの「SOAP RPC URL」プロパティと一致するように変更します (図 2-2を参照してください)。

   「SOAP RPC URL」にコンテキストルート値が次の形式で表示されます。
   `http://`*hostname*`:`*portnum*`/`*contextRoot*`/servlet/rpcrouter`.

注 - 「コンテキストルート」プロパティ値 `WEB-INF` に、次のようにスラッシュ (/) を付ける必要があります。*/contextRoot*。

---

5. Web モジュールを Web サーバーに配備します。

   この方法についての詳細は、Forte for Java 4 プログラミングシリーズ の『Web コンポーネントのプログラミング』を参照してください。

   次の手順では、Web サービスが EJB ビジネスコンポーネントを使用していることを前提としています。

J2EE アプリケーションのアセンブル

Web サービスおよび参照されるコンポーネントを含む J2EE アプリケーションを作成する手順は、次のとおりです。

1. J2EE アプリケーションを作成するパッケージをマウスの右ボタンでクリックし、「新規」>「J2EE」>「アプリケーション」を選択します。

   新規アプリケーションウィザードが表示されます。

2. 「名前」フィールドにアプリケーションの名前を入力し、「完了」をクリックします。

   J2EE アプリケーションノードがパッケージに追加されます。

3. アプリケーションノードをマウスの右ボタンでクリックし、「モジュールの追加」を選択します。

   「アプリケーションにモジュールを追加」ダイアログが表示されます。

4. Web サービスおよび Web サービスによって参照されるすべての EJB モジュールを選択して、「了解」をクリックします。

   図 2-9 は、次の下位ノードを持つ J2EE アプリケーションノードを示しています。

   - EJB JAR ノード *webserviceName*`_EjbJar`
     webserviceName は、Web サービスの名前。このノードは、生成された Web サービスの EJB モジュールに対応する
   - WAR ノード *webserviceName*`_War`
     webserviceName は、Web サービスの名前。このノードは、生成された Web サービスのWeb モジュールに対応する
   - アプリケーションに追加した各 EJB モジュール用の EJB JAR ノード

図 2-9　Web サービス用 J2EE アプリケーション

5. 必要に応じて Web サービスの WAR ノードの Web コンテキストプロパティを編集します。

   デフォルトの Web コンテキストプロパティは、Web サービス WAR ノードに割り当てられます。このプロパティの値は、Web サービスへのアクセスに使用される URL の一部で、Web サービスの「SOAP RPC URL」プロパティ内のコンテキストルート値に一致していることが必要です (図 2-2を参照してください)。

   デフォルト設定を編集する手順は、次のとおりです。

   a. Web サービスの WAR ノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。

   b. Web コンテキストプロパティをクリックし、新しい値を入力して Enter キーを押します。

   図 2-10 は、「Web コンテキスト」値が Web サービスの「SOAP RPC URL」プロパティのコンテキストルート値に一致するように、 `StockApp` に変更されている Web サービス WAR ファイルのプロパティを示しています。

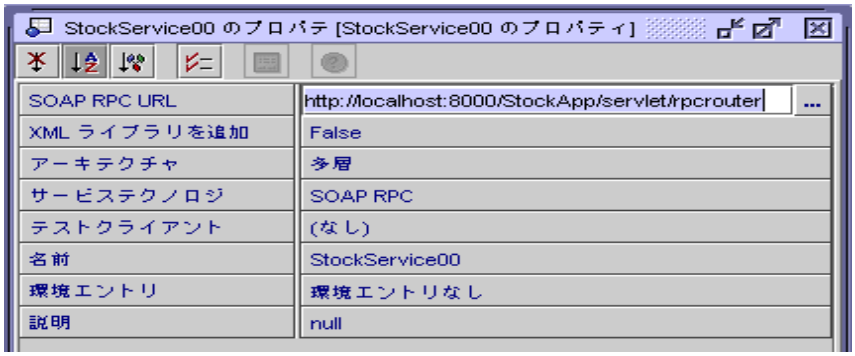

図 2-10 「Web コンテキスト」が `StockApp` に設定されている Web サービス WAR ファイルプロパティ

6. 必要に応じて J2EE アプリケーションの「アプリケーションサーバー」プロパティを編集します。

   このプロパティでは、配備用のアプリケーションサーバーインスタンスを指定する必要があります。プロパティは、IDE のデフォルトアプリケーションサーバーにデフォルト設定されています。この値は「エクスプローラ」ウィンドウの「実行」タブで設定できます。

   プロパティを編集するには、J2EE アプリケーションノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。「アプリケーションサーバー」プロパティ値をクリックして、省略記号 (...) をクリックします。図 2-11 に示されているように利用可能なアプリケーションサーバーインスタンスを選択するダイアログが表示されます。

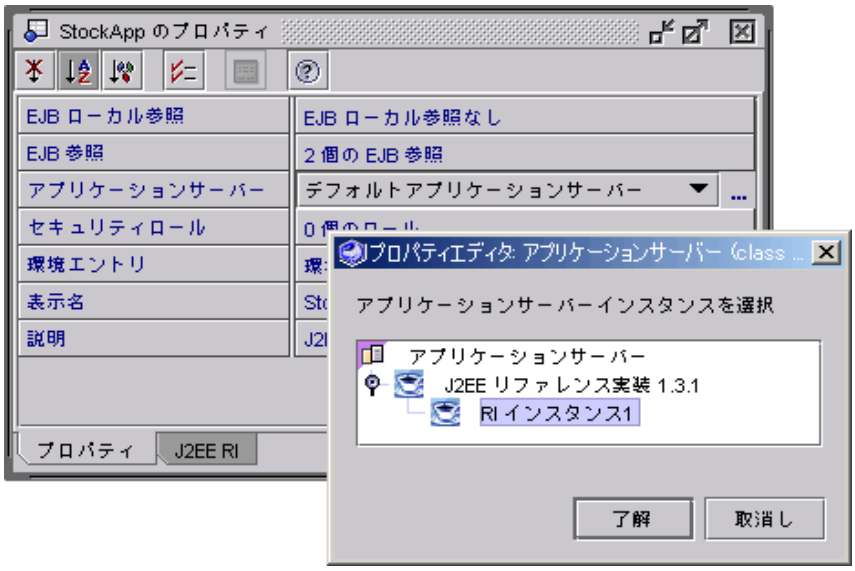

図 2-11 配備用のアプリケーションサーバーインスタンス

RI サーバーの構成

注 - Forte for Java 4 IDE とともにインストールされた RI サーバーを使用している場合は、以下の手順の 1 から 5 までをスキップしてください。

外部 RI アプリケーションサーバーを構成する手順は、次のとおりです。

1. 「エクスプローラ」ウィンドウの「実行時」タブをクリックします。

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

3. J2EE リファレンス実装ノードをマウスの右ボタンでクリックして「サーバーインスタンスの追加」を選択し、J2EE リファレンス実装サーバーインスタンスを追加します。

   図 2-12 は、「エクスプローラ」ウィンドウの「実行時」タブを示しています。

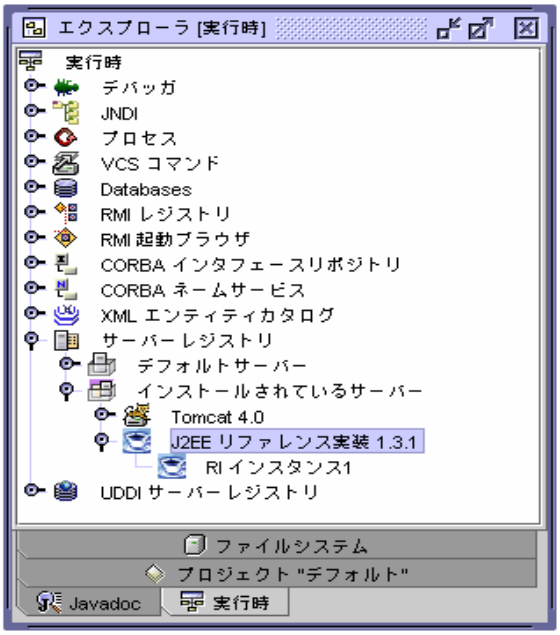

図 2-12 アプリケーションサーバーインスタンスの作成

4. J2EE リファレンス実装のプロパティシートを開きます。

   ふたたび、J2EE リファレンス実装ノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。

5. RIHome プロパティの値を、J2EE SDK がインストールされているディレクトリに設定します。

6. (オプション) データベースアクセスを構成します。

   EJB コンポーネントの多くは、データベースアクセス用に RI サーバーを構成する必要があります。`J2EE_HOME/config/resource.properties` ファイルを編集して必要なドライバおよびデータソースを追加します (`J2EE_HOME` は環境変数で、J2EE SDK がインストールされているディレクトリを参照します)。

   RI サーバー管理コマンドを使用して `resource.properties` ファイルを編集できます。たとえば、コマンドウィンドウを開いて次のコマンドを入力します。

   `j2eeadmin -addJdbcDriver` *yourDriver*

   `j2eeadmin -addJdbcDatasource` *yourDatasource*

   この項目の詳細については、『Forte for Java, Enterprise Edition インストールガイド』を参照してください。

J2EE アプリケーションの配備

Web サービスアプリケーションを J2EE RI サーバーに配備する手順は、次のとおりです。

1. J2EE アプリケーションノードをマウスの右ボタンでクリックし、「配備」を選択します。

   この操作により、アプリケーションのパッケージ処理が開始されます。処理が完了すると、アプリケーションは RI サーバーに配備されます。

2. パッケージ処理および配備のステータスメッセージを確認します。

   この処理には数分間かかります。IDE ステータス行には、パッケージの進捗状況を示すメッセージが表示されます。RI サーバーのコンソールには、配備の進捗状況が示されます。

   配備が完了すると、Web サービスがテストできる状態になります。

注 - RI サーバーを開始する必要はありません。アプリケーションを配備すると、IDE によってサーバーが自動的に開始されます。外部 RI サーバーを使用している場合は、配備を開始する直前に実行されていないことを確認してください。

テストクライアントの作成

この節では、Forte for Java 4 IDE のクライアント配備機能を一覧表示し、テストクライアントを作成する方法を説明します。詳細および Web サービスクライアントの作成手順については、第 3 章を参照してください。

Web サービスクライアントの機能

Forte for Java 4 IDE は、対話形式のダイアログおよびメニューコマンドを使用した次のタスクをサポートします。

- Forte for Java 4 Web サービスからのクライアントの生成
- WSDL からのクライアントの生成
- UDDI レジストリに公開されたサービスへアクセスするクライアントの生成

Web サービス用のデフォルトテストクライアントの設定

IDE を使用して Web サービスからクライアントを生成する場合、「新規クライアント」ダイアログにクライアントをサービス用のデフォルトのテストクライアントに設定できるチェックボックスが表示されます。

デフォルトのテストクライアントを変更するには、エクスプローラで Web サービスノードをマウスの右ボタンでクリックし、「プロパティ」を選択して「テストクライアント」プロパティを編集します。

注 - クライアントを Web サービス用のデフォルトのテストクライアントに設定すると、Web サービスが J2EE アプリケーションにアセンブルされるときにクライアントが自動的に Web サービス WAR ファイルにパッケージ化されます。

Web サービスのテスト

デフォルトのテストクライアントを使用して Web サービスをテストできます。Web サービスのアーキテクチャによって、2 つの手順があります。

J2EE アプリケーションでのテスト

Web サービスが多層アーキテクチャであるか、または Web 主体アーキテクチャで EJB ビジネスメソッドを使用している場合の手順は、次のとおりです。

1. クライアントを生成して、Web サービスのデフォルトのテストクライアントに設定します。

2. Web サービスを J2EE アプリケーションにアセンブルします。

3. アプリケーションノードをマウスの右ボタンでクリックして「実行」を選択します。または、テストクライアントノードをマウスの右ボタンでクリックし、「実行」を選択します。

   IDE は Web サービスアプリケーションを配備して、 J2EE アプリケーション内で指定されている J2EE アプリケーションサーバー内で実行し、デフォルトのテストクライアントを実行します。Web サービスがデフォルトのテストクライアントなしでアセンブルされている場合は、入力ページがないことを示すページが表示されます。

   または、Web サービスアプリケーションを配備してからテストクライアントノードをマウスの右ボタンでクリックし、「実行」を選択することもできます。IDE はテストクライアントをアセンブルして WAR ファイルを IDE のデフォルト Tomcat Web コンテナに配備し、クライアントを実行します。

J2EE アプリケーションなしでのテスト

Web サービスが Web 主体アーキテクチャで EJB ビジネスメソッドを使用していない場合は、J2EE アプリケーションはありません。この場合は、次の手順でテストを実行します。

1. クライアントを生成して、Web サービスのデフォルトのテストクライアントに設定します。

2. Web サービスの「SOAP RPC URL」プロパティが内部 Tomcat Web サーバーインストールのポート (8081) を参照していることを確認します。

   「SOAP RPC URL」プロパティを表示または変更するには、エクスプローラで Web サービスノードをマウスの右ボタンでクリックして「プロパティ」を選択します。

3. Web サーバーノードをマウスの右ボタンでクリックして、「実行」を選択します。

   IDE は WAR ファイルをアセンブルしてデフォルト Tomcat Web コンテナに配備し、Web サービスを実行します。WAR プロパティを表示するには、エクスプローラで WAR ノードをマウスの右ボタンでクリックして、「プロパティ」を選択します。

   テストは反復プロセスです。Web サービスを編集して実行時クラスおよびクライアントを再生成し、条件を満たすまでテストします。

UDDI レジストリの使用

この節では、IDE を使用して UDDI レジストリを介して Web サービスをほかの開発者が利用できるようにする方法を説明します。

次の基本タスクを実行する必要があります。

1. ほかの開発者およびエンドユーザーがネットワーク上でアクセス可能なアプリケーションサーバーまたは Web サーバーに実行 Web サービスを配備します。
2. Web サービスから WSDL を生成して、ほかの開発者がネットワーク上でアクセス可能な Web サーバーに公開します。
3. ほかの開発者がネットワーク上でアクセス可能な UDDI レジストリに Web サービスを公開します。

IDE では、Web サービスを配備したときにタスク 1 および 2 を実行します。この節では、主にタスク 3 用に提供されているウィザードについて説明します。

注 - クライアント開発は 第 3 章で説明します。説明には、UDDI レジストリ内で Web サービスを見つける方法および Web サービスを使用できるクライアントを生成する方法が含まれます。

WSDL の生成

UDDI レジストリを使用することなく、WSDL ファイルをほかの開発者と共有することができます。WSDL ファイルは共有ドライブに配置するか、または電子メールの添付ファイルとして送信できます。UDDI レジストリを使用することなく WSDL からクライアントを生成する方法についての詳細は、74 ページの「WSDL からのクライアントの作成」を参照してください。

Web サービス用の実行クラスを生成した後で、次のように WSDL ファイルを生成します。

1. エクスプローラで、Web サービスノードをマウスの右ボタンでクリックし、「WSDL を生成」を選択します。

   エクスプローラ階層の Web サービスノードと同じレベルの同じパッケージ内に WSDL ノードが表示されます。WSDL ノードは Web サービスと同じ名前をもっていますが、緑の球形 () の別のアイコンで区別されます。

2. エクスプローラで、WSDL ノードをマウスの右ボタンでクリックし、「開く」を選択します。

   「ソースエディタ」ウィンドウは読み取り専用モードで表示されます。WSDL は XML ドキュメントであり、読み取ったり、コピーすることができます。WSDL ファイルはディレクトリ階層でも見つけることができます。Web サービスパッケージのディレクトリの下にあり、ファイル名は *webserviceName*`.wsdl` になります。

注 - この手順は、Web アプリケーションを配備し、IDE を使用して Web サービスを UDDI レジストリに公開している場合には必要ありません。WSDL は、IDE の配備および公開プロセスで管理されます。

UDDI レジストリオプションの管理

IDE は既知の UDDI レジストリのリストおよび各レジストリにアクセスするために必要な情報 (URL、ユーザー ID、およびパスワードなど) を保持します。また、Web サービスクライアントを作成するとき、または Web サービスをレジストリに公開するときに使用されるデフォルト情報 (ビジネス、カテゴリ、および識別子など) も保持します。

注 - IDE ウィザードを使用して UDDI レジストリにアクセスする前に、レジストリ情報にデフォルト値を設定することをお勧めします。デフォルト値を設定することによって、ウィザードで値を繰り返し入力する手間を省くことができます。

UDDI レジストリオプションを管理するには、「ツール」 →「オプション」 →「分散アプリケーションサポート」 →「UDDI レジストリ」を IDE のメインウィンドウから選択します。図 2-13 に示されている「オプション」ダイアログが表示されます。

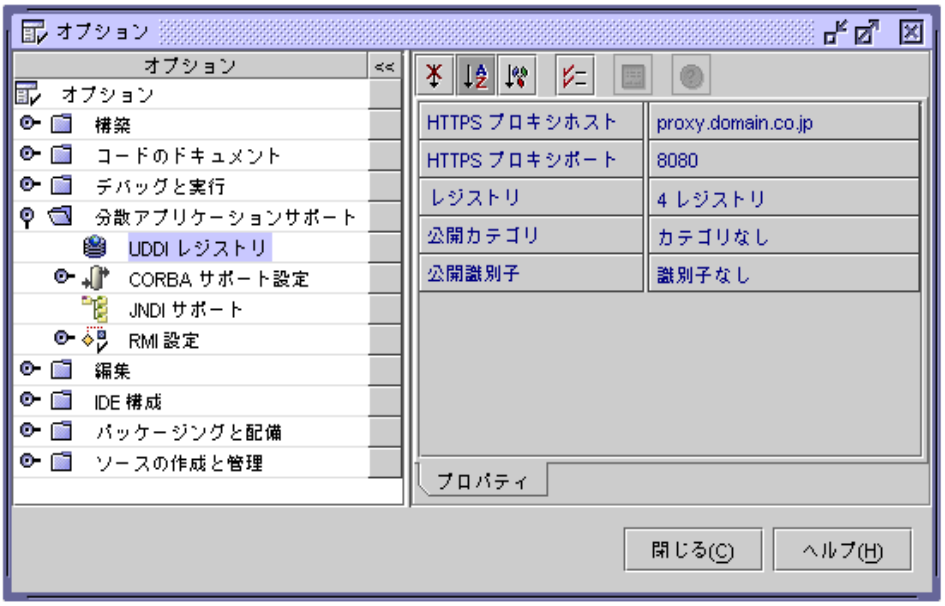

図 2-13 「UDDI レジストリオプション」ダイアログ

デフォルト公開カテゴリおよび識別子の設定

Web サービスに関連付けられている公開カテゴリおよび公開識別子は、Web サービスを公開するときに UDDI レジストリに保存されます。

識別子はビジネスまたはテクニカルモデル (tModel) に固有のデータ要素です。カテゴリはビジネス、サービスまたは tModel を分類するデータ要素です。カテゴリまたは識別子をユーザークエリに指定して、ビジネス、サービスまたはレジストリ内の tModel を見つけることができます。

デフォルトの公開カテゴリを設定するには、UDDI レジストリの「オプション」ダイアログで「公開カテゴリ」をクリックして(図 2-13を参照してください) 、表示される省略記号 (...) ボタンをクリックします。図 2-14 に示されている「公開カテゴリ」ダイアログが表示されます。

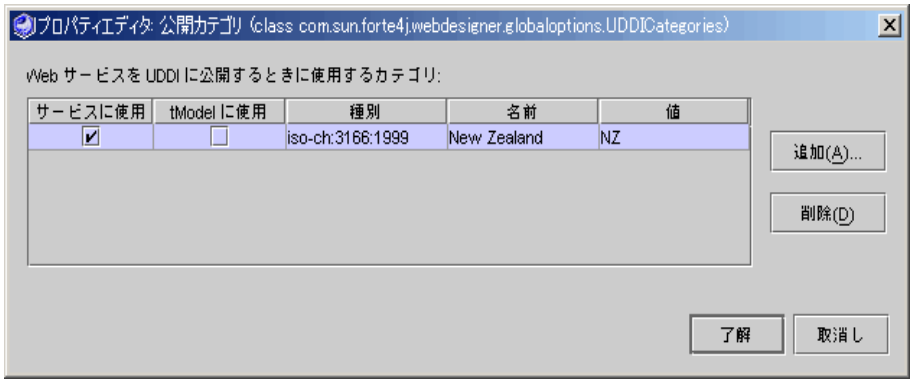

図 2-14 「UDDI 公開カテゴリ」プロパティエディタ

「サービスに使用」チェックボックスまたは「tModel に使用」チェックボックスをクリックして、レジストリに公開するときのカテゴリをサービスまたは tModels のデフォルトとして設定します。

カテゴリを削除するには「削除」をクリックします。

カテゴリを追加するには「追加」をクリックします。図 2-15 に示されているように「UDDI カテゴリ (分類)」ダイアログが表示されます。カテゴリは業界で定められた標準です。

注 - Web サービスを UDDI レジストリに公開するとき、IDE では tModel に関連付けられる隠しカテゴリを自動的に追加します。カテゴリの名前は `uddi-org:types`、値は `wsdlSpec` です。このカテゴリは tModel が Web サービスの WSDL 記述を持っていることを示します。これは tModels にだけ適用され、Web サービスの分類には使用されません。このカテゴリは IDE では表示されませんが、レジストリツールの検索で検出することができます。

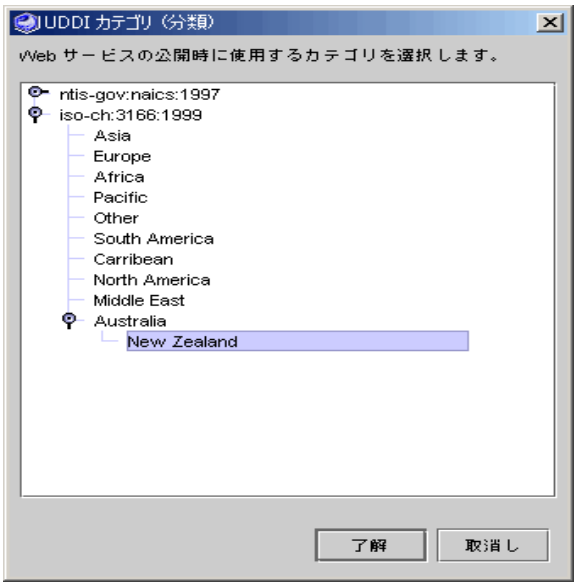

図 2-15 UDDI カテゴリ (分類)

ノードを展開して、目的のカテゴリを選択し、「了解」をクリックします。カテゴリは「公開カテゴリ」ダイアログに表示されます。複数の項目を選択するには、Ctrl キーを押しながら項目をクリックします。連続する複数の項目を選択するには、Shift キーを押しながら項目をクリックします。

デフォルトの公開識別子を設定するには、UDDI レジストリの「オプション」ダイアログで「公開識別子」をクリックして (図 2-13を参照してください)、 表示される省略記号 (...) ボタンをクリックします。図 2-16 に示されている「公開識別子」ダイアログが表示されます。

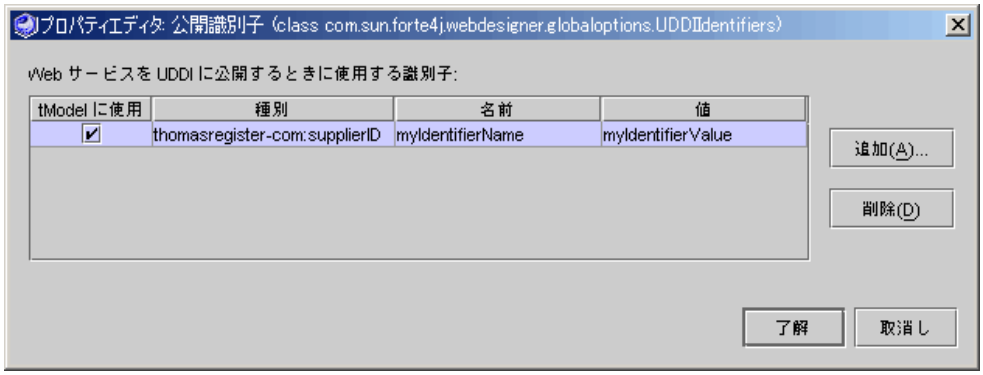

図 2-16 「UDDI 公開識別子」プロパティエディタ

識別子は削除または追加でき、tModel のデフォルトとして設定することができます。

識別子を削除するには、識別子を選択して「削除」をクリックします。

識別子を追加するには「追加」をクリックします。図 2-17 に示されているように「UDDI 識別子を追加」ダイアログが表示されます。識別子タイプと tModel UUID は業界で定められた標準です。

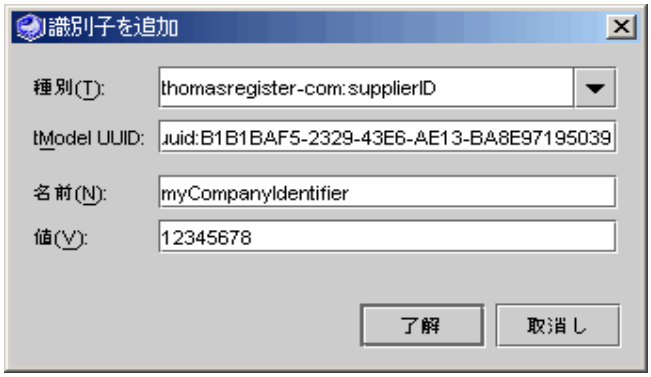

図 2-17 「UDDI 識別子を追加」ダイアログ

型を選択すると、選択した型に対する tModel UUID が表示されます。ユーザー定義済みの型の tModel UUID (レジストリキー) がわかっている場合は、独自のユーザー定義型を指定することができます。

名前と値を指定して「了解」をクリックすると、「公開識別子」ダイアログに戻ります。

「了解」をクリックします。

IDE 内でのレジストリ情報の編集

IDE で既知のレジストリに関する情報を編集するには、「オプション」ダイアログの「UDDI レジストリ」の「レジストリ」プロパティの省略記号 (...) ボタンをクリックします (図 2-13を参照してください)。図 2-18 に示されている「プロパティエディタ」ダイアログが表示されます。

プロパティエディタには、選択したレジストリに関する既知のレジストリおよび詳細情報が表示されます。画面の最上位部分には既知のレジストリの表が表示されます。レジストリ名をクリックすると、画面の下部に詳細情報が表示されます。

レジストリ情報を編集する方法は、次のとおりです。

- レジストリをリストに追加するには、「追加」をクリックします。「照会 URL」、「公開 URL」、および「ブラウザツール URL」に内容を説明するレジストリ名および値を指定します。
- レジストリを削除するには、選択してから「削除」をクリックします。
- デフォルトレジストリを変更するには、希望するレジストリを選択して「デフォルト」列のチェックボックスをクリックします。
- レジストリ名を編集するには、現在の名前をダブルクリックして新しい名前を入力します。
- デフォルトのログインを設定するには、「名前」および「パスワード」に値を指定します。

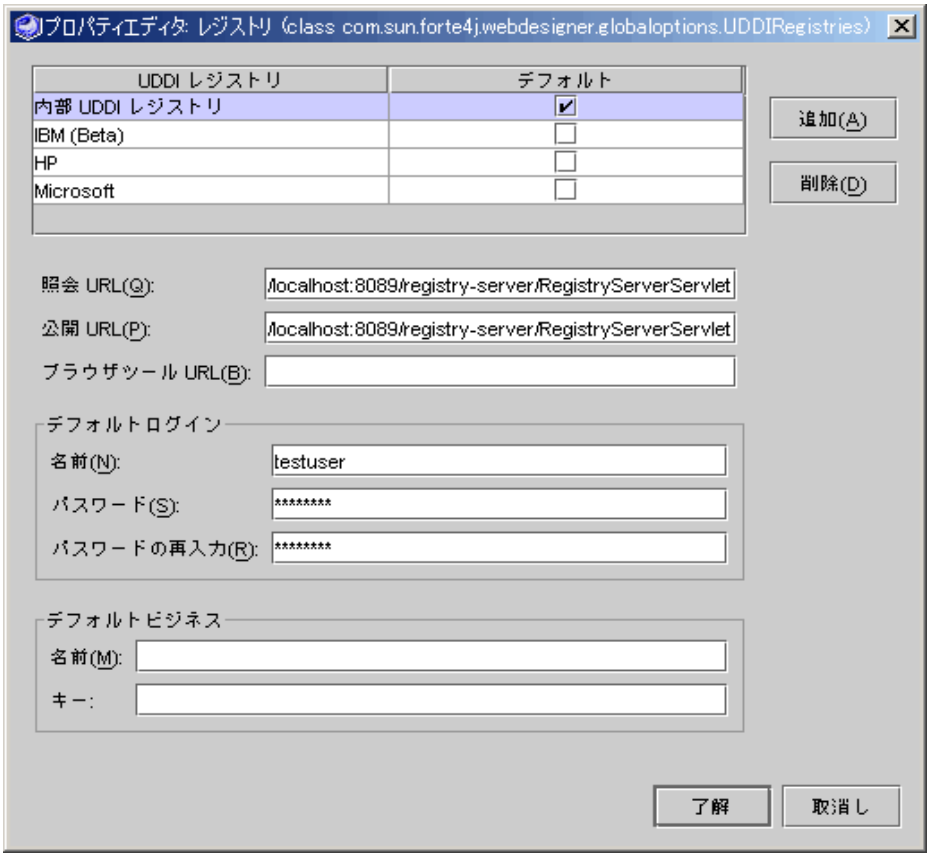

図 2-18 UDDI レジストリプロパティエディタ

IDE が UDDI ウィザードでデフォルトとして使用するビジネスを設定することもできます。デフォルトのビジネスを設定するには、「名前」および「キー」の値を指定します。

注 - デフォルトのログイン名およびパスワードを設定できます。これは便利ですが、パスワードが暗号化されずに IDE ファイルシステムに保存さるため、環境によってはセキュリティ上のリスクがあります。パスワードを毎回入力するほうが安全です。

注 - クライアントを生成するために Web サービスを公開しているときや、レジストリを検索しているときに UDDI レジストリプロパティを編集することができます。76 ページの「クライアントの作成： 手順」を参照してください。

外部 UDDI レジストリへのアクセス

UDDI レジストリにアクセスするには、IDE に特定の情報を提供する必要があります (43 ページの「IDE 内でのレジストリ情報の編集」を参照してください)。

レジストリを検索して Web サービスクライアントを生成するには、照会 URL が必要です。Web サービスをレジストリに公開するには、公開 URLが必要です。この情報を、レジストリを管理するレジストリオペレータから取得します。

セキュリティはレジストリによって異なります。レジストリに公開するときにログイン名とパスワードが必要な場合は、レジストリオペレータに確保してください。パブリックレジストリでは、Web ブラウザから独自のアクティビティを設定できるツールが提供されている場合があります。

Web サービスの UDDI レジストリへの公開

この節では、公開タスクと用語、および公開手順について説明します。

公開タスクと用語

UDDI レジストリ内の Web サービスエントリは、「ビジネス」エントリ、「テクニカルモデル (*tModel*)」エントリ、オプションのカテゴリ、および識別子と関連付けられています。Web サービスを UDDI レジストリに公開するとき、IDE ウィザードで次のタスクを実行できます。

- 新しいビジネスエントリを追加するか、または既存のビジネスエントリを検出してビジネスエントリを Web サービスエントリに関連付けます。(このウィザードで追加したビジネスには、カテゴリおよび識別子は関連付けられません。)
- 新しい tModel エントリを追加するか、または既存の tModel エントリを検出して tModel エントリを Web サービスエントリに関連付けます。
- 1 つまたは複数の標準業界カテゴリおよび識別子を、オプションで Web サービスエントリまたは tModel エントリに関連付けます。

ビジネスは Web サービスを含む組織エンティティです。レジストリ内の各ビジネスエントリは固有のキーを持っています。

tModel は、WSDL をポイントする URL (オーバービュー URL) を含むレジストリエントリで、Web サービスの外部インタフェースを説明します。この情報は IDE またはほかの開発者ソフトウェアで実行時に、Web サービス上でメソッドを呼び出すことができるクライアントを作成するために使用されます。

UDDI レジストリ内の Web サービスエントリは、エントリポイント (エンドポイント URL または「サービス URL」) を指定します。この URL は、クライアントが Web サービスの実行インスタンスを検出するために使用します。公開中に、IDE ではデフォルトでこの URL を Web サービスの「SOAP RPC URL」プロパティの値に設定します。この値は編集できます。

標準業界カテゴリおよび識別子は、UDDI レジストリ内の Web サービスエントリおよび tModel エントリと関連付けて後に続く検索を簡略化することができます。UDDI レジストリ検索の詳細については、75 ページの「UDDI レジストリからのクライアントの作成」を参照してください。

ビジネスは、複数の異なるサービスを UDDI レジストリに公開できます。各サービスは、tModel エントリに関連付けられている WSDL によって構造的に記述されます。複数のサービスエントリで同じ tModelを参照できます。たとえば、サービスプロバイダとして企業内で使用するサービスの 1 つのインスタンス、ビジネスパートナー用の別のインスタンス、および一般ユーザー用の別のインスタンスをサポートする場合があります。多様なインスタンスは tModel によって参照される WSDL 内の記述に従って同じ外部インタフェースを持ちますが、パフォーマンスおよび可用特性は異なります。

Web サービスを公開すると、tModel への参照が自動的に含まれます。

公開手順

この手順を実行する前に、Web サービスアプリケーションを配備する必要があります。IDE は、サービスの WSDL ファイルのコピーを配備したサービス内に配置します。サービスを公開すると、tModel がこの WSDL ファイルを参照します。

Web サービスを UDDI レジストリに公開する手順は、次のとおりです。

1. エクスプローラで、Web サービスノードをマウスの右ボタンでクリックし、「UDDI に公開」を選択します。

   図 2-19 に示されている「新規 Web サービスを UDDI に公開」ダイアログが表示されます。

   このダイアログでは、次のことができます。

   - 「サービス名」を UDDI レジストリに設定する
   - 「ネットワークアクセスポイント」の種類を http (デフォルト) または https に設定する
   - デフォルトで Web サービスの「SOAP RPC URL」プロパティの値に設定されている「ネットワークアクセスポイント」のアドレス URL を編集する
   - 「サービスの説明」にテキストを指定する

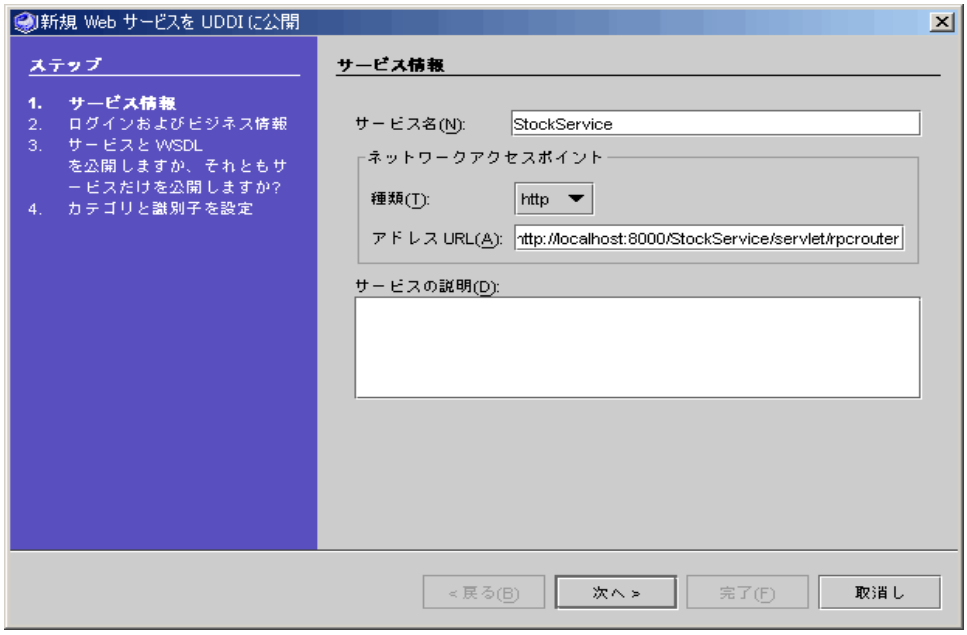

図 2-19 「新規 Web サービスを UDDI に公開」ダイアログ

注 - 「アドレス URL」のホスト名は `localhost` にデフォルト設定されます。この値をホストのネットワークマシン名に変更します。

---

2. 「次へ」をクリックして、図 2-20 に示されているように「ログインおよびビジネス情報」ダイアログを表示します。

   このダイアログでは、次のことができます。

   - 公開用のターゲット UDDI レジストリを選択する
   - 選択したレジストリに公開できるように、登録用のユーザー ID およびパスワードを入力する
   - レジストリ内の既存のビジネスを検出する。または新しいビジネスを追加する

   選択した UDDI レジストリ用のユーザー ID、パスワード、およびビジネスのデフォルト値をすでに設定している場合は、デフォルト値が表示されます。デフォルト値は、各フィールドに別の値を入力するか、ペーストすることによって変更できます。

ビジネスを指定するには、「ビジネス名」に値を入力します。「検索」、「追加」ボタンがアクティブになります。

- **既存のビジネスの場合は「検索」をクリックします。**「ビジネスキー」の既存の値が表示されます。レジストリ内の複数のビジネスが「ビジネスキー」の値に一致する場合は、レジストリサーバーによって最初に返された「ビジネスキー」値が表示されます。
- **新しいビジネスの場合は「追加」をクリックします。**IDE はビジネスエントリを UDDI レジストリに追加して、「ビジネスキー」の生成値を表示します。この方法で追加したビジネスには、カテゴリおよび識別子は関連付けられていません。

「編集」をクリックして、選択した UDDI レジストリの情報を編集します。UDDI レジストリプロパティエディタが表示され、公開プロセス中にレジストリデフォルト値を変更できます。43 ページの「IDE 内でのレジストリ情報の編集」を参照してください。

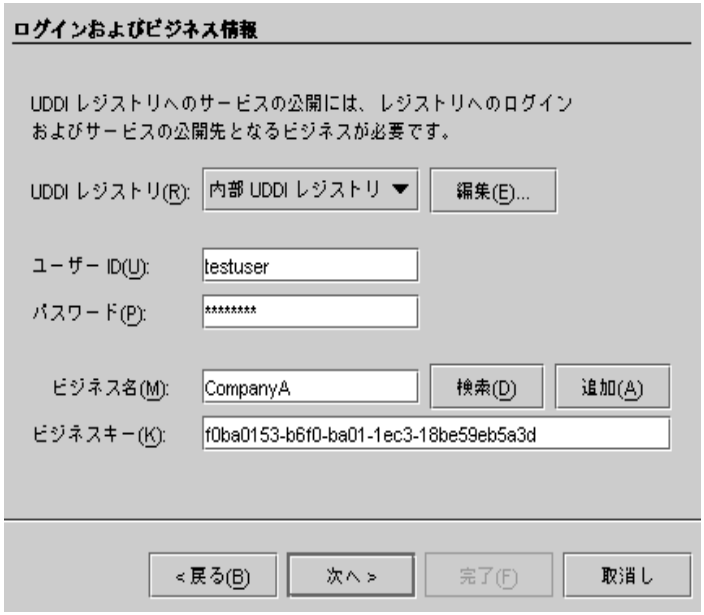

図 2-20 「UDDI ログインおよびビジネス情報」ダイアログ

3. 「次へ」をクリックして、図 2-21 に示されているように UDDI tModel を選択するためのダイアログを表示します。

   この手順では、公開するビジネスを指定して詳細を規定します。新しい tModel をレジストリ内に作成するか、または既存の tModel を検出して、tModel を Web サービスに関連付けます。

新しい tModel を作成すると、IDE 命名規則に基づいて、配備した Web サービスの Address URL から WSDL File URL が派生されます (図 2-19を参照してください)。Address URL を変更した場合、または Web サービスを IDE の外部に配備した場合は、WSDL File URL の値は空白のままになります。

URL は、別の値を入力することによって変更できます。

「tModel の説明」にテキストを指定できます。

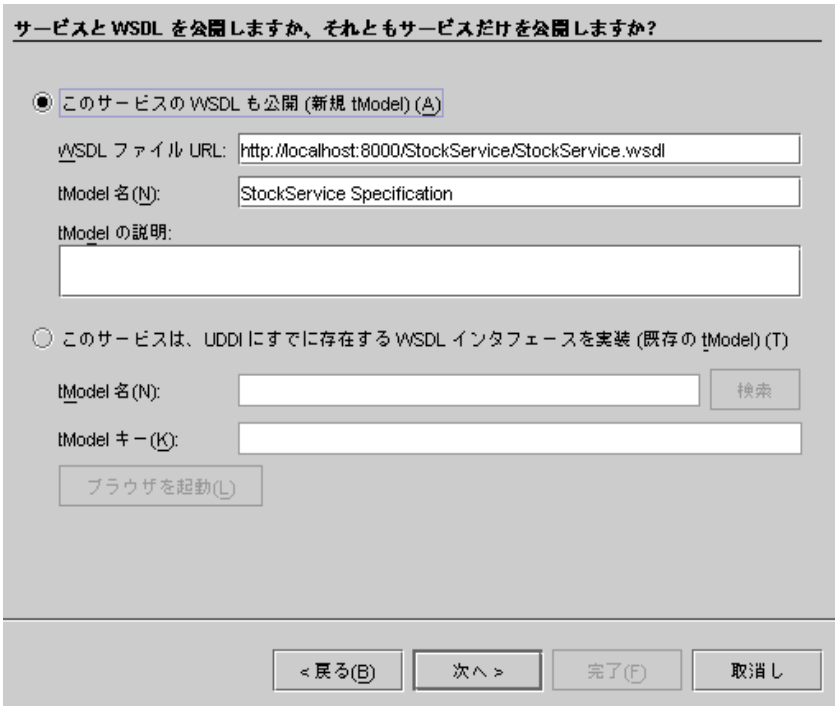

図 2-21 UDDI tModel を選択するダイアログ

UDDI 内の既存の tModel を使用している場合は、tModel 名を入力して「検索」をクリックします。tModel キー が表示されます。レジストリ内の複数の tModel が「tModel 名」の値に一致する場合は、レジストリサーバーによって最初に返された「tModel キー」値が表示されます。キー値がすでにある場合は、「tModel キー」フィールドに入力するか、またはペーストします。

注 - ブラウザツールを提供している UDDI レジストリに Web サービスを公開している場合は、それらのツールを使用して tModel キーを検索できます。「ブラウザを起動」をクリックすると、UDDI レジストリのブラウザツール URL によって指定された Web ページにデフォルトの Web ブラウザが表示されます (図 2-18を参照してください)。

4. 「次へ」をクリックします。

   図 2-22 に示されているように「UDDI カテゴリと識別子を設定」ダイアログが表示されます。

   ダイアログには、デフォルトのカテゴリおよび識別子が表示されます。40 ページの「デフォルト公開カテゴリおよび識別子の設定」に示されているように、「編集」をクリックしてカテゴリや識別子を追加または削除します。

図 2-22 UDDI カテゴリおよび識別子を設定

5. 「完了」をクリックします。

   UDDI レジストリに Web サービスが公開されます。これには数分間かかります。IDE の最上位ウィンドウステータス行に正常に完了したことを示すメッセージが表示されます。

内部 UDDI レジストリ

内部 UDDI レジストリは開発プロセスのエンドツーエンドテストの簡便性のために Sun の Java™ Wb Services Developer Pack (Java WSDP) の UDDI レジストリサーバーが、単一のユーザー用に IDE に統合され、バンドルされています。このレジストリは専用 Tomcat Web サーバー (アプリケーション開発に使用される内部 Tomcat サーバーとは異なる) 内で実行するサーブレットで構成され、Apache Xindice XML データベースを格納場所として使用します。IDE は、レジストリサーバーを開始または停止したときに、Tomcat サーバーおよび Xindice データベースサーバーを自動的に開始または停止します。

注 - 内部 UDDI レジストリは単一のユーザー用に設定されています。名前は `testuser` で、パスワードは `testuser` です。この名前およびパスワードを内部レジストリのデフォルトとして設定します。

内部 UDDI レジストリサーバーの開始と停止

内部 UDDI レジストリサーバーを開始する手順は、次のとおりです。

1. 「エクスプローラ実行時」タブ付き区画で UDDI サーバーレジストリノードを展開します。

   「内部 UDDI レジストリ」ノードが表示されます。

2. 「UDDI レジストリ」ノードをマウスの右ボタンでクリックし、図 2-23 に示されているように「サーバーを起動」を選択します。

   IDE 出力ウィンドウにサーバー開始メッセージが表示されます。IDE が前の Tomcat サーバープロセスを停止したことを示すメッセージも表示されます。

注 - 内部 UDDI レジストリサーバーがすでに実行されている場合は、「サーバーを起動」メニュー項目はアクティブになりません。

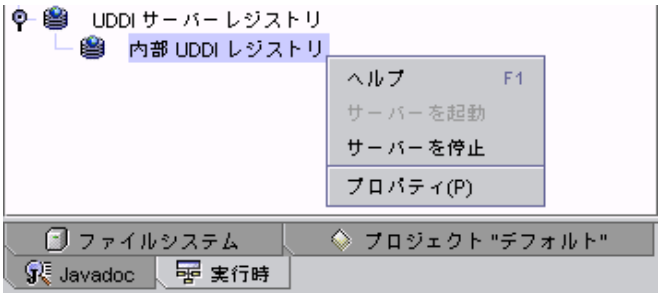

図 2-23 内部 UDDI レジストリサーバーの開始

内部 UDDI レジストリサーバーを停止する手順は、次のとおりです。

1. 「エクスプローラ実行時」タブ付き区画で UDDI サーバーレジストリノードを展開します。

   内部 UDDI レジストリノードが表示されます。

2. 「UDDI レジストリ」ノードをマウスの右ボタンでクリックして、「サーバーを停止」を選択します。

   IDE 出力ウィンドウにサーバー開始メッセージが表示されます。IDE が前の Tomcat サーバープロセスを停止したことを示すメッセージも表示されます。

注 - 内部 UDDI レジストリサーバーが実行されていない場合は、「サーバーを停止」メニュー項目はアクティブになりません。

サンプルレジストリブラウザの使用

Java WSDP は、サンプルレジストリブラウザを提供します。これには Forte for Java 4 IDE にはない追加機能が含まれます。

サンプルレジストリブラウザを使用すると、内部レジストリからサービスを削除したり、名前およびキー以外のビジネス情報を追加したりなど、IDE ウィザードで提供されていないその他のアクションを実行できます。ただし、その名前が示すとおり、サンプルレジストリブラウザは完全な機能をすべて備えたツールではありません。たとえば、tModel を表示または削除することはできません。

Java WSDP ソフトウェアおよびサンプルプログラムは Forte for Java 4 ホームディレクトリの下の `jwsdp` ディレクトリにあります。サンプルプログラムソースは `wsdp/samples` ディレクトリにあります。

詳細およびダウンロード可能なチュートリアルについては、Java WSDP Web サイト `http://java.sun.com/webservices/webservicespack.html`を参照してください。

この節の目的は、サンプルレジストリブラウザののための参考となる一部の情報を説明するもので、その機能のすべてを説明することではありません。

注 - サンプルレジストリブラウザは、Forte for Java 4 IDE 外の独自のウィンドウで実行します。ただし、主な用途は内部レジストリサーバーにアクセスすることです。外部レジストリは一般に検索および公開用の独自のツールセットを備えています。

サンプルレジストリブラウザを開始する前に、52 ページの「内部 UDDI レジストリサーバーの開始と停止」 に説明されているように IDE を使用して内部 UDDI レジストリを開始します。

サンプルレジストリブラウザを開始する手順は、次のとおりです。

1. コマンドウィンドウを開いて現在のディレクトリを *FFJ_HOME*/jwsdp/bin に変更します。
2. 次のコマンドを入力します。
   - Microsoft Windows システムの場合 **jaxr-browser**
   - Solaris 環境の場合 **./jaxr-browser.sh**

   図 2-24　に示されているようにサンプルレジストリブラウザが表示されます。

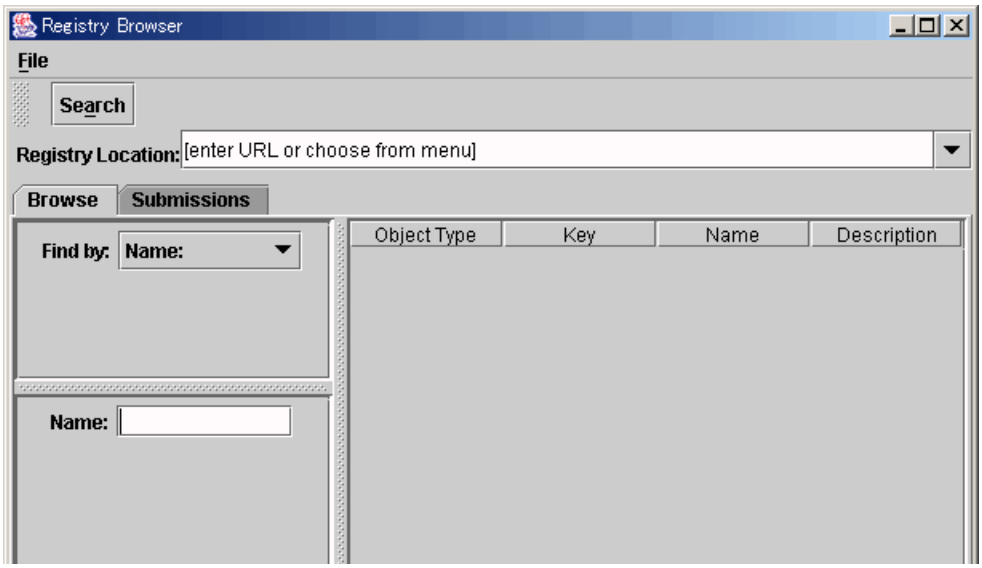

図 2-24　JWSDP サンプルレジストリブラウザ

内部レジストリ用にサンプルレジストリブラウザを構成する手順は、次のとおりです。

1. 図 2-25 に示されているように、「Registry Location」リストで最後の URL を選択します。

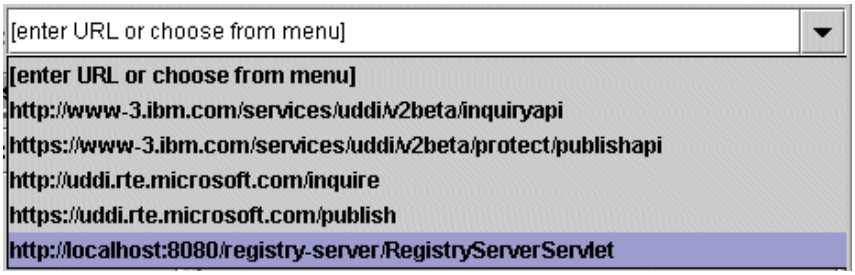

図 2-25 JWSDP サンプルレジストリブラウザ URL 選択

2. 図 2-26 に示されているように URL のポート番号を 8080 から内部レジストリ Tomcat Web サーバー用の正しいポート番号に変更します。

   IDE では、このサーバー用にデフォルトのポート番号として 8089 が設定されています。

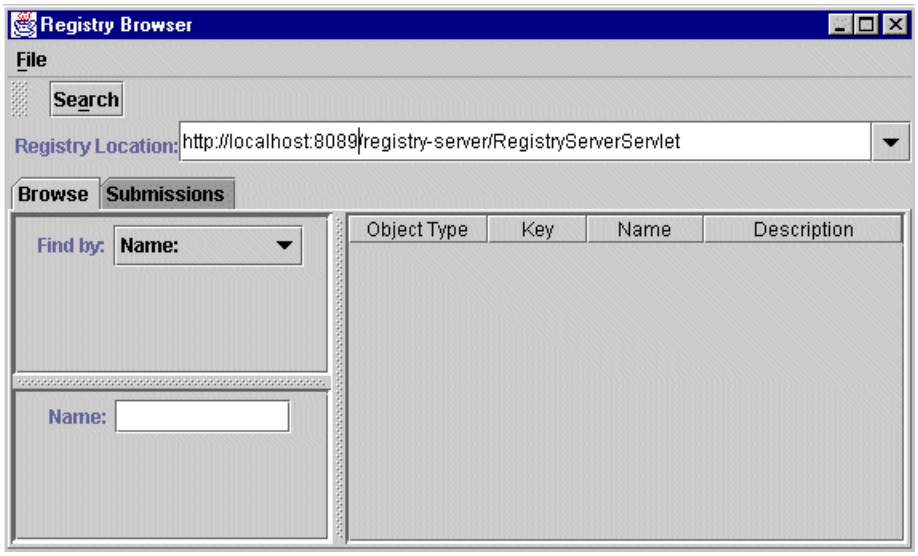

図 2-26 内部レジストリ URL を備えた JWSDP サンプルレジストリブラウザ URL

サンプルレジストリブラウザのメインウィンドウには、左側に「Browse」と「Submission」タブが付いた区画があります。メインウィンドウには、右側にタブ付き区画で選択したオブジェクトに関する詳細情報を表示する区画があります。メインウィンドウの左上には、「Search」、「Submit」、およびその他のコンテキスト依存アクション用のボタンがあります。

図 2-27 は、会社名による検索の結果を示した例を示します。

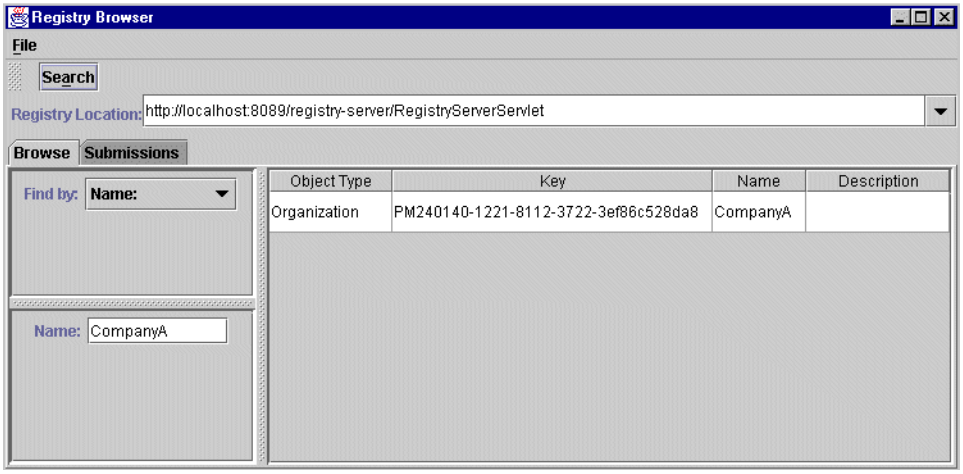

図 2-27　選択したビジネスを表示した JWSDP サンプルレジストリブラウザ

図 2-28 は、「Submission」タブ付き区画を使用した例を示しています。

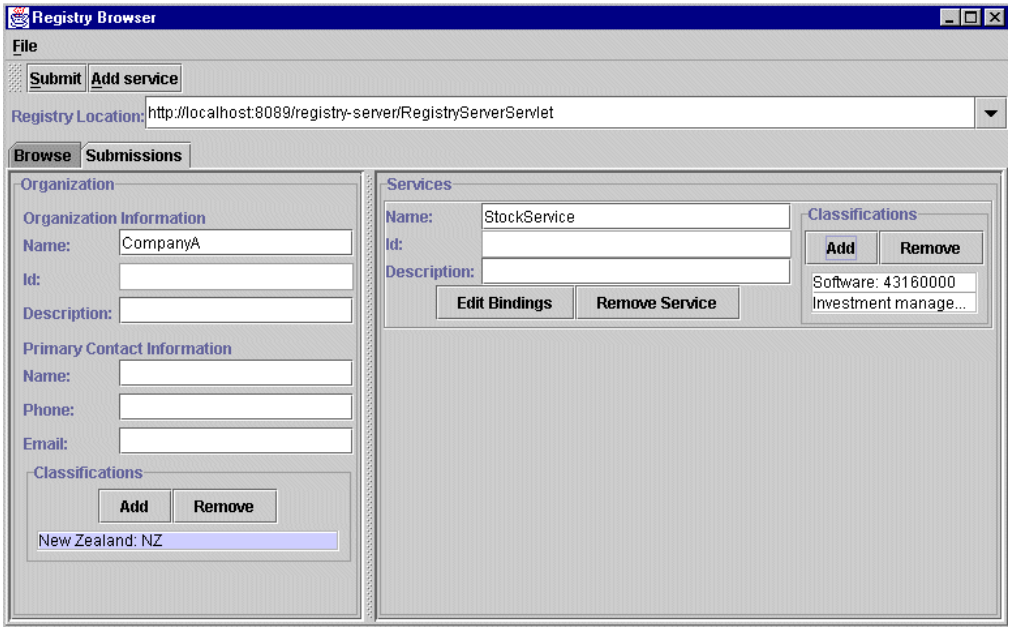

図 2-28　「Submission」タブ付き区画を表示した JWSDP サンプルレジストリブラウザ

サンプルレジストリブラウザの追加機能を確認するために、次に示す Java WSDP Web サイトにあるチュートリアルを使用してダウンロードおよび作業を行うこともできます。`http://java.sun.com/webservices/webservicespack.html`

オブジェクトのインスタンス化と参照の解決

XML オペレーションを開発するときに、XML オペレーションが呼び出すメソッドを指定します。これらのメソッドを実行時に呼び出すために、Web サービスは特定のオブジェクトを必要とします。各メソッド呼び出しに対して、次のものを見つけるか、またはインスタンス化します。

- メソッドが定義されているクラスのインスタンス
- メソッドによってパラメータとして要求される各クラスのインスタンス

このタスクを実行するために、Web サービスでは、これらの各ターゲットオブジェクトへの参照とし、ターゲットオブジェクトが存在しない適切なクラスのオブジェクトをインスタンス化する方法の定義を維持します。XML オペレーションにメソッド呼び出しを追加すると、デフォルトオブジェクト参照とターゲットオブジェクト定義が自動的に Web サービスに追加されます。これらのデフォルトは一般的に適切な設定であり、編集する必要はありません。

ただし、オブジェクト参照のターゲットを手動で指定して、要件に合うように新しいターゲットオブジェクト定義を編集して作成できます。IDE 外で作成されたエンタープライズ Bean へのオブジェクト参照を手動で解決することが必要な場合があります。

この項目の詳細については、付録 Bを参照してください。

配備記述子

Web サービス用の実行時クラスを生成すると、Web モジュールおよび EJB モジュール配備記述子も生成されます。Web サービス J2EE アプリケーションを組み立てると、これらの配備記述子がアプリケーションに含まれます。配備記述子は、アプリケーションの実行時プロパティを設定するために使用される XML ファイルです。J2EE アプリケーションはこれらの記述子のフォーマットを定義します。

配備記述子は、開発中はいつでもソースエディタ内で参照できます。配備記述子を編集することもできます。ただし、配備記述子を編集すると、実行時クラスを生成し直した場合に配備記述子は再生成されず、編集内容が保持されます。配備記述子の編集後に Web サービスに対して行なった変更は、配備記述子には伝達されません。したがって、配備記述子の編集は、開発サイクルの最後にだけ行なってください。

この項目の詳細については、付録 C を参照してください。

配列およびコレクションのサポート

この節では、`Array` および `Collection` 型の入力および出力データのサポートについて概説します。この項目の詳細については、第 4 章を参照してください。

配列

IDE の配列のサポートの概要は、次のとおりです。

- 配列出力は直接メソッド呼び出しおよび XML オペレーションでサポートされています。
- 配列入力は直接メソッド呼び出しでサポートされています。
- 配列入力は XML オペレーションでサポートされていません。

コレクション

コレクションは、`java.util.Collection` を実装する Java クラスです。IDE のコレクションのサポートの概要は、次のとおりです。

- コレクション出力は直接メソッド呼び出しおよび XML オペレーションでサポートされています。
- コレクション入力は、サーバー上およびクライアントプロキシ内での直接メソッド呼び出しでサポートされていますが、生成された JSP ページ内での呼び出しはサポートされていません。
- コレクション入力は XML オペレーションではサポートされていません。

第3章

Web サービスクライアントの作成

Forte for Java 4 IDE では、クライアントコードを記述することなく Web サービスクライアントを作成する方法を提供します。この章では、次の各タスクを実行するために使用する IDE ツールおよび手順について説明します。

- Forte for Java 4 Web サービスからのクライアントの生成
- WSDL からのクライアントの生成
- UDDI レジストリエントリからのクライアントの生成

IDE で独自の Web サービスおよびクライアントを作成している場合は、サービス用にデフォルトのテストクライアントを設定することができます。詳細については、35 ページの「Web サービス用のデフォルトテストクライアントの設定」を参照してください。

Forte for Java Web サービスからのクライアントの作成

Web サービスをテストするのに適した簡易クライアントおよびクライアント SOAP プロキシを開発中に、自動的に生成することができます。クライアントは編集したり、ビジネス用のより洗練されたクライアントに置換したりすることができます。

クライアントの作成

パッケージノード、Web サービスノード、または IDE のメインウィンドウメニューからクライアントを作成できます。

Web サービスノードから開始する場合

1. エクスプローラで、「新規 Web サービスクライアント」ウィザードを開いて Web サービスノード (青い球のアイコン付き) をマウスの右ボタンでクリックし、「新規クライアント」を選択します。

   図 3-1 に示されている「新規クライアント」ダイアログが表示されます。

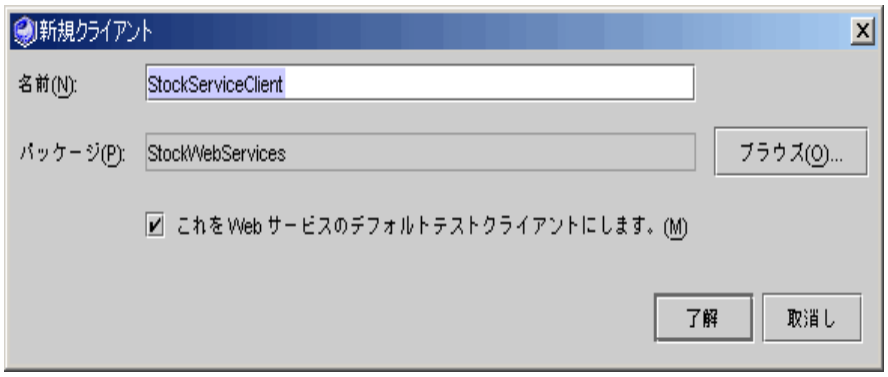

図 3-1　「新規クライアント」ダイアログ

2. 「パッケージ」フィールドに Web サービスクライアントを作成する目的の場所が指定されていることを確認します。

   目的のパッケージを見つけて選択するには、「ブラウズ」ボタンをクリックします。

3. 「名前」フィールドで、新しい Web サービスクライアントの名前を入力します。

   ウィザードで指定されているすでにパッケージ内に存在する Web サービスクライアントの名前を入力すると、強調表示された次のようなメッセージが表示されます。この名前をもつ Web サービスクライアントはすでにパッケージに存在しています。

4. 「了解」をクリックします。

   新しい Web サービスクライアントがエクスプローラに表示されます。クライアントノード ( ) が指定したパッケージの下に表示されます。

Java パッケージノードから開始する場合

1. Java パッケージノードをマウスの右ボタンでクリックし、「新規」>「Web サービス」>「Web サービスクライアント」を選択します。

   または、「ファイル」>「新規テンプレート」>「Web サービス」>「Web サービス」>「Web サービスクライアント」を IDE のメインウィンドウから選択します。

   この場合、図 3-2 に示されているように「Web サービスクライアント」ダイアログが表示されます。

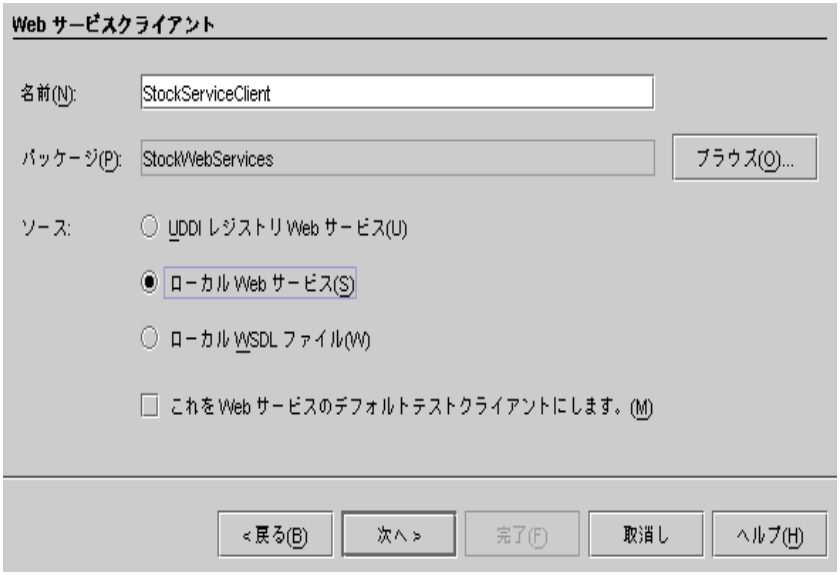

図 3-2　「新規 Web サービスクライアント」ダイアログ

2. 「名前」および「パッケージ」を設定します。
3. 「ソース」で「ローカル Web サービス」を選択します。
4. 「次へ」をクリックしてブラウズウィンドウを開き、ブラウズウィンドウで目的の Web サービスを選択して「完了」を選択します。

   新しい Web サービスクライアントがエクスプローラに表示されます。クライアントノード ( ) が指定したパッケージの下に表示されます。

クライアントの SOAP 実行プロパティの設定

IDE は Apache SOAP 実行環境 (デフォルト) を使用するクライアントまたは kSOAP 実行環境を使用するクライアントを生成できます (kSOAP は、Java 2 Microedition に適した SOAP API です)。

クライアント型を変更する手順は次のとおりです。

1. クライアントを作成します。
2. クライアントノードをマウスの右ボタンでクリックして、「プロパティ」を選択します。「Soap 実行」プロパティのデフォルト値は `Apache SOAP` です。
3. プロパティ値をクリックします。選択値 `Apache SOAP` および `kSoap` が 図 3-3 に示されているように表示されます。`kSoap` を選択します。

   クライアントを生成すると、kSOAP プロキシが作成されます。

図 3-3　クライアント SOAP 実行プロパティ

注 - 以降、特に指定がない限り Apache SOAP クライアントを想定して説明しています。

クライアントの生成

クライアントの SOAP プロキシを生成してユーザーインタフェース (HTML および JSP ページ) をテストするには、エクスプローラで Web サービスクライアントノードをマウスの右ボタンでクリックして、「クライアントプロキシを生成」を選択します。

エクスプローラ階層に生成された SOAP プロキシノードが表示されます。また、クライアントノードの下に下位生成されたドキュメントノードが、SOAP プロキシを呼び出す HTML および JSP ページとともに表示されます。図 3-4 は、この手順を示しています。

図 3-4　エクスプローラ階層表示内のクライアントドキュメントおよび SOAP プロキシ

クライアント HTML ページおよび JSP ページ

次のクライアントドキュメントも生成されます。

- 各メソッドまたは XML オペレーション用に 1 つの JSP ページ。これらの各 JSP ページは、対応するビジネスメソッドまたは XML オペレーションを実行し、結果を HTML にフォーマットする機能を提供します。開発者は、JSP ページをクライアントとして使用し、ビジネスメソッドおよび XML オペレーションをテストできます。Web ページデザイナは、生成された JSP ページは、より洗練されたユーザーフレンドリーなクライアントの基本部分としてカスタマイズして使用することができます。
- 1つの HTML 開始ページ。開始ページは、生成されたすべての JSP ページに対応する 1 つの HTML フォームを表示します。この HTML フォームは JSP ページを起動するために必要な URL を提供します。入力パラメータが必要な場合、このフォームにはユーザー入力用のフィールドが含まれます。
- 1つの HTML エラーページ。

開始ページおよびエラーページの名前は、IDE 内でクライアントノードのプロパティとして表示されます。

HTML または JSP コードを表示するには、目的のドキュメントのノードをマウスの右ボタンでクリックして、「開く」を選択します。開始ページと JSP ページのコードサンプルは、図 3-5 および 図 3-6 に示されています。

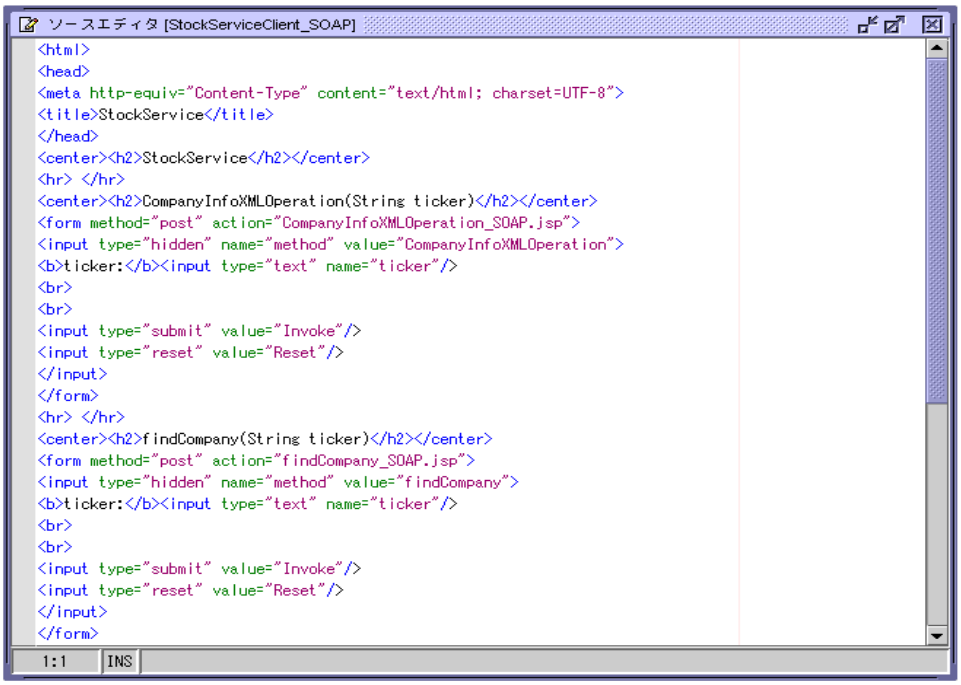

図 3-5　クライアントサンプル HTML 開始ページ

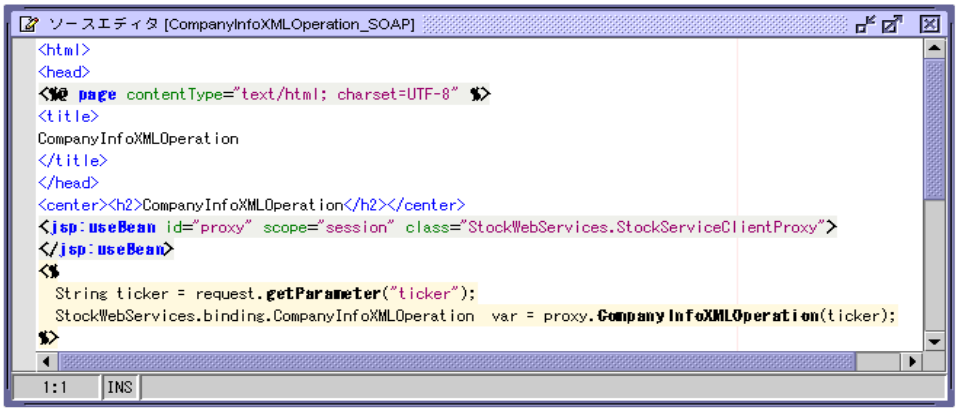

図 3-6　クライアントサンプル JSP ページ

クライアント SOAP プロキシ

生成されたプロキシクラスの Java ソースコードを表示するには、プロキシノードをマウスの右ボタンでクリックして「開く」を選択します。プロキシソースコードのサンプルフラグメントは、図 3-7 に示されています。

IDE を使用すると、プロキシソースを編集できますが、IDE 内でクライアントを生成し直した場合は、プロキシソースの変更は保持されません。

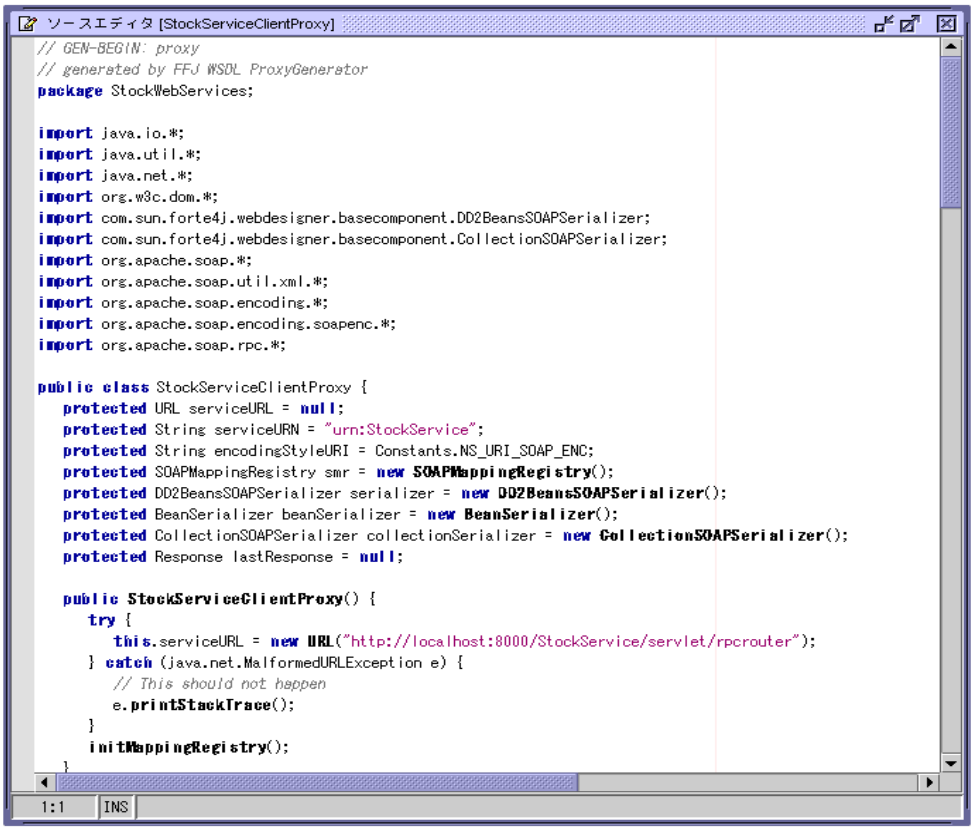

図 3-7　クライアント SOAP プロキシサンプル Java コード

生成されたプロキシクラスは、 Java Swing クライアントやより洗練された HTML および JSP ページとともに使用できます。

独自のフロントエンドクライアントコンポーネントの使用

kSOAP 実行を使用するクライアントは、携帯電話や PDA など、一般的にユーザーインタフェースが表示領域の制約を受ける小さい J2ME デバイスで実行します。開発者はしばしば Java AWT、WML、およびその他のプレゼンテーション技術を使用してユーザーインタフェースを作成します。

クライアントの「SOAP 実行」プロパティを kSoap に設定した場合、IDE は「生成されたドキュメント」ノードを作成せず、HTML または JSP ページを生成しません。

クライアントの「SOAP 実行」プロパティが Apache SOAP の場合、特定の状況においては生成されたドキュメントを抑制したい場合があります。たとえば、すでにフロントエンドビジネスクライアントを持っていて、IDE でクライアント SOAP プロキシのみを生成して、独自のクライアントコンポーネントを表示する場合などです。これを行うには次の手順に従ってください。

1. クライアントを作成します。
2. クライアントノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。「プレゼンテーションレイヤを生成」プロパティのデフォルト値は True です。
3. 「プレゼンテーションレイヤを生成」プロパティの値を False に変更します。

   クライアントを生成すると、クライアント SOAP プロキシのみが作成されます。
4. (オプション) クライアントノードをマウスの右ボタンでクリックし、「参照を追加」>「ドキュメント」を選択します。

   (ドキュメントを含むディレクトリをマウントして IDE エクスプローラで表示できるようにした場合は、Web デザインツールで作成された HTML や JSP ページなど IDE 以外で作成されたドキュメントへの参照を追加することができます。)

   「ドキュメントへの参照を追加」ブラウザが表示されます。独自の HTML および JSP ページを含むフォルダへ移動して、ブラウザ内で「了解」をクリックし、クライアントに参照を追加します。参照を含むドキュメントフォルダが作成されます。ドキュメントフォルダはエクスプローラのクライアントノードの下に、クライアントの 生成されたドキュメントフォルダと同じ階層レベルに表示されます。

   クライアントノードをマウスの右ボタンでクリックして「ライブラリおよびクラス」に参照を追加することもできます。参照はテストアプリケーションの WAR ファイルに含められます。ライブラリおよびクラスは、Swing テストクライアントの作成に使用できます。

注意 - 参照されるフォルダ内のノードは、実際のファイルやフォルダを表すリンクです。ファイルまたはフォルダへの参照を削除すると、参照のみが削除され、ファイルまたはフォルダはそのまま残ります。ただし、参照されるフォルダ内のノードを削除すると、ノードがリンクされているファイルまたはフォルダが削除されます。実際のファイルまたはフォルダを削除しない場合は、参照されるフォルダ内のノードを削除しないでください。

クライアントアプリケーションのアセンブル

クライアントアプリケーションをアセンブルするには、エクスプローラでクライアントノードをマウスの右ボタンでクリックし、「アセンブル」を選択します。WAR ファイルが作成され、ノードが WAR アイコン ( ) およびフォーム名 *clientName*_Client とともにエクスプローラに表示されます。

WAR ファイルの内容を表示するには、エクスプローラでノードをマウスの右ボタンでクリックして「WAR の内容を表示」を選択します。図 3-8 は、この表示を示しています。

Web サービスを変更して実行クラスを生成し直す場合は、IDE でクライアントを自動的に生成し直すことができます。これを行うには、エクスプローラでクライアントをマウスの右ボタンでクリックし、「WSDL を再フェッチ」を選択します。これによりクライアントプロキシおよびドキュメントが生成し直されます。(「WSDL を再フェッチ」コマンド名は、サービスに基づいて、クライアントに関連付けられている内部 WSDL を表します。この WSDL は、エクスプローラには表示されません。)

その後、クライアントをアセンブルし直すことができます。または、次にクライアントを実行したときにクライアントが自動的にアセンブルし直されます。

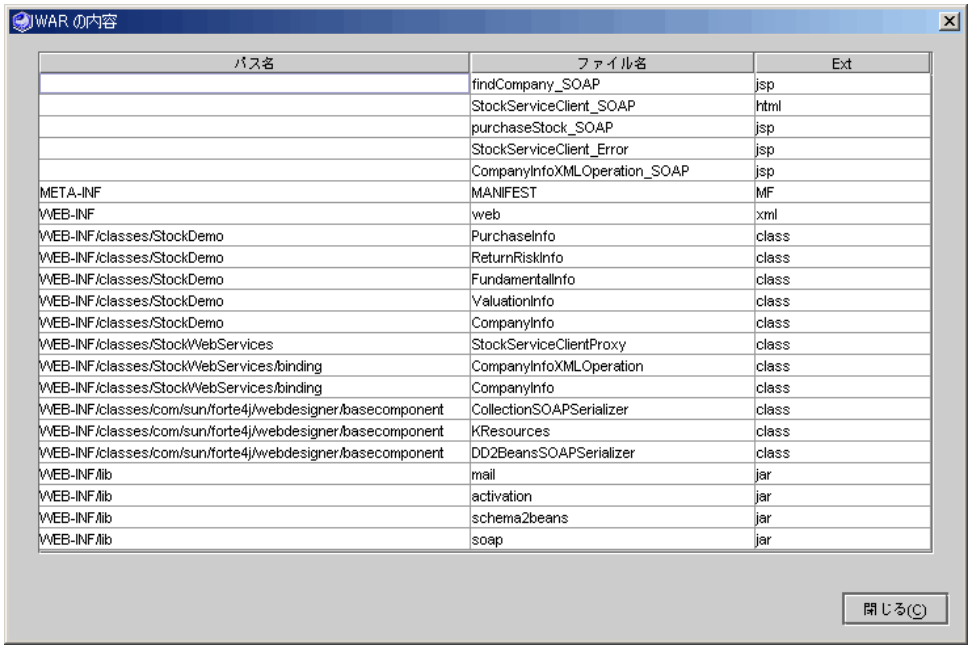

図 3-8　クライアントサンプル WAR ファイル表示

クライアントの実行

クライアントによって参照される Web サービスは、すでに配備されている必要があります。

クライアントを実行するには、エクスプローラでノードをマウスの右ボタンでクリックして、「実行」を選択します。

組み込み Apache Tomcat Web サーバーが自動的に起動してクライアントアプリケーションが配備され、クライアントの開始ページがデフォルトの Web ブラウザに表示されます。図 3-9は、この結果を示しています。

または、35 ページの「Web サービス用のデフォルトテストクライアントの設定」で説明しているようにクライアントをデフォルトのテストクライアントにすることもできます。クライアントを開始するには、Web サービスの J2EE アプリケーションをマウスの右ボタンでクリックし、「実行」を選択します。

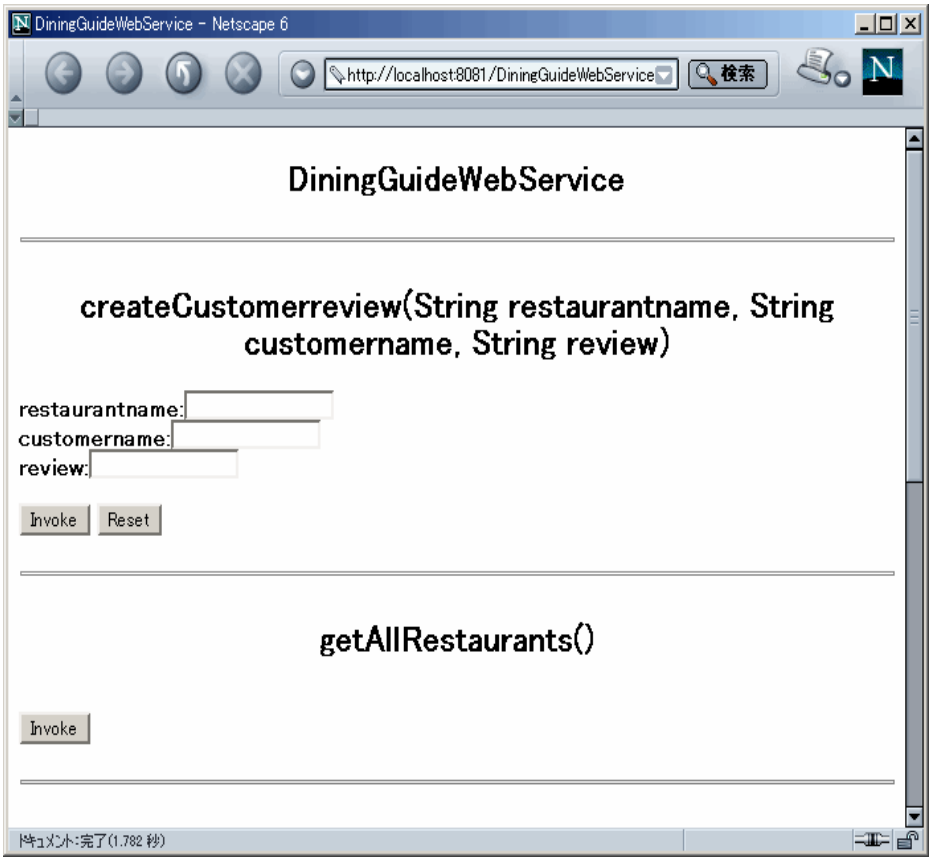

図 3-9　クライアント開始ページ

図 3-10 は、レストランのレビューアプリケーションのために生成されたテストクライアントの表示です。

図 3-10 顧客レビューコレクションのクライアント出力表示

注 - 生成されたテストクライアントはレストランレビューのコレクションを SOAP エンベロープを含めて XML ドキュメントとして表示します。エンドユーザー向けの生産クライアントでは、技術的な情報の表示を省いて、よりユーザーフレンドリーに出力をフォーマットしたい場合があります。

WSDL からのクライアントの作成

WSDL は、Web サービスの外部インタフェースの簡易表示を提供します。Web サービスの WSDL 記述を作成する手順については、37 ページの「WSDL の生成」を参照してください。WSDL および UDDI レジストリに関するその他の詳細については、6 ページの「WSDL」 および 37 ページの「UDDI レジストリの使用」を参照してください。

WSDL は、UDDI レジストリを使用していない場合でも役立ちます。プロジェクト内の何人かが別の場所または別のスケジュールで異なるいくつかのクライアントを開発している場合があります。彼らに、クライアントを生成するための WSDL のコピーを渡すことができます。WSDL ファイルは XML ドキュメントであり、共有ネットワークドライブに配置したり、電子メール添付ファイルとして送信したり、ディスクで配布したりすることができます。開発者が、Web クライアントをテストするためには、サービスの実行インスタンスにネットワークでアクセスできる必要があります。

次の手順は、WSDL ファイルがあること、ファイル名拡張子が `.wsdl` であること、ファイルを含んでいるディレクトリが Forte for Java 4 IDE にマウントされていることを前提としています。`.wsdl` ファイルはエクスプローラにノードとして緑の球形アイコン () 付きで表示されます。この命名規則に従わない WSDL ファイルを受け取った場合は、名前を変更するか、またはファイル名拡張子 `.wsdl` を付けてコピーを作成できます。

WSDL からクライアントを生成する手順は、次のとおりです。

1. 「ファイル」>「新規テンプレート」>「Web サービス」>「Web サービス」>「Web サービスクライアント」を IDE のメインウィンドウから選択します。

   図 3-2 に示されているように「Web サービスクライアント」ダイアログが表示されます。

2. 「名前」および「パッケージ」を設定し、「ソース」で「ローカル WSDL ファイル」を選択します。

3. 「次へ」をクリックしてブラウズウィンドウを開きます。

4. `WSDL` ノードをブラウズウィンドウで選択します。

5. 「完了」をクリックします。

   クライアントノードが作成されます。

6. エクスプローラで、クライアントノードをマウスの右ボタンでクリックし、「クライアントプロキシを生成」を選択します。

   クライアント SOAP プロキシとノードが作成されます。また、HTML および JSP ページのセットが生成されたドキュメントノード内のクライアントノードの下に作成されます。

7. その後の手順は、61 ページの「Forte for Java Web サービスからのクライアントの作成」と同じです。

   生成されたドキュメントノードなしでクライアント SOAP プロキシのみを生成する手順は、次のとおりです。

1. エクスプローラで、`WSDL` ノードをマウスの右ボタンでクリックし、「クライアントプロキシを生成」を選択します。

   クライアント SOAP プロキシとノードが作成されます。

2. クライアント SOAP プロキシへの呼び出しを行う独自のクライアントコンポーネントを作成します。

   これは、クライアント SOAP プロキシを使用するために簡単にカスタマイズできる開発済みのクライアントがすでにある場合、または Java Swing クライアントなど、IDE によって生成された HTML および JSP ページとは異なるクライアントを使用する場合に適しています。

UDDI レジストリからのクライアントの作成

UDDI レジストリからクライアントを作成する必要があるビジネスニーズおよびシナリオについては、第 1 章を参照してください。

クライアントの作成： 計画と実装

ワークフローは、次のタスクで構成されます。

- Web サービスを選択するための基準の決定
- 検索する UDDI レジストリの選択
- Web サービス用の UDDI レジストリの検索
- Web サービスに関する情報のダウンロード
- クライアントの生成

- クライアントのカスタマイズ

タスク 1 は、詳細な計画が必要であり、アプリケーションの設計プロセスの一部となります。Web サービスの特性を考えるだけでは不十分です。サービスを提供するビジネスについて考え、そのビジネスがどのようなサポートを提供するか、サービスの使用コストはどのくらいなのかについても考える必要があります。プライベートレジストリを使用していて、所属するプロジェクト企業、またはプロジェクトを共同計画しているビジネスパートナーがサービスプロバイダとなる場合には、分析は簡単になります。

クライアントの作成： 手順

この項では、Forte for Java 4 IDE を使用してレジストリを選択する方法、サービス用のレジストリを検索する方法、およびサービスの実行インスタンスにアクセス可能なクライアントの生成方法について説明します。

1. Java パッケージノードをマウスの右ボタンでクリックし、「新規」>「Web サービス」>「Web サービスクライアント」を選択します。

   または、「ファイル」>「新規」>「Web サービス」>「Web サービス」>「Web サービスクライアント」を IDE のメインウィンドウから選択します。

   図 3-11 に示されているように「Web サービスクライアント」ダイアログが表示されます。

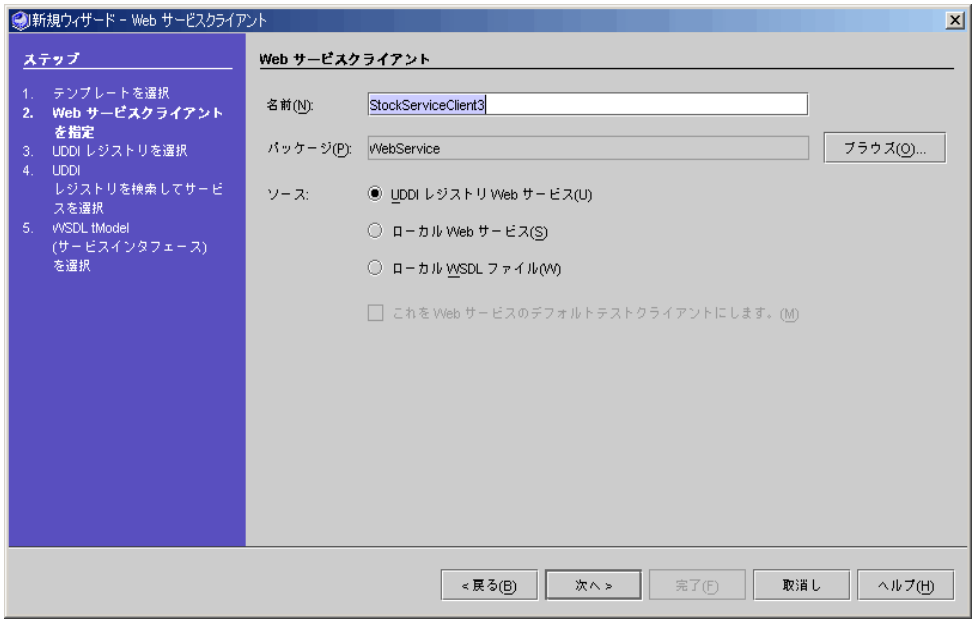

図 3-11 「新規 Web サービスクライアント」ウィザード

2. 「名前」および「パッケージ」を設定し、「ソース」の「UDDI レジストリ Web サービス」ラジオボタンを選択します。

3. 「次へ」をクリックして、図 3-12 に示されているように「UDDI レジストリを選択」ダイアログを表示します。

   ダイアログに UDDI レジストリのリストが表示されます。目的のレジストリを選択するか、または「編集」をクリックしてレジストリ情報を編集するか、デフォルトのレジストリを変更します (43 ページの「IDE 内でのレジストリ情報の編集」を参照してください)。

注 - IDE 内の最初のデフォルトレジストリは、IDE にバンドルされている内部 UDDI レジストリです。このレジストリの詳細については、52 ページの「内部 UDDI レジストリ」を参照してください。

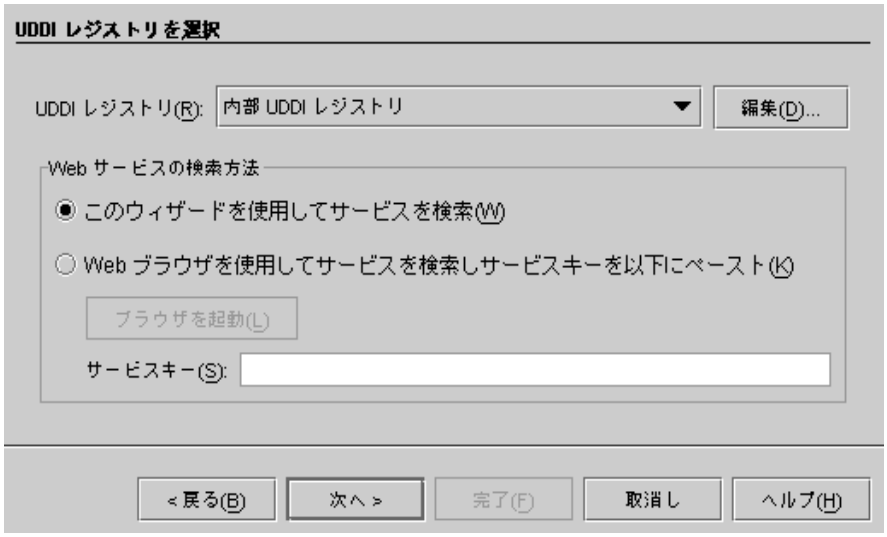

図 3-12 「UDDI レジストリ選択」ダイアログ

ほとんどの UDDI レジストリは、レジストリ検索およびレジストリエントリの追加、編集、削除用のブラウザベースのツールを提供します。レジストリのブラウザツール URL は、IDE 内で編集できるプロパティの 1 つです (43 ページの「IDE 内でのレジストリ情報の編集」を参照してください)。

選択したレジストリを独自のブラウザツールで検索する手順は、次のとおりです。

a. 「Web ブラウザを使用してサービスを検索」ラジオボタンを選択します。

「ブラウザを起動」ボタンがアクティブになります。

Web サービスのレジストリキーを前の検索で取得している場合は、その値を「サービスキー」フィールドにペーストして「次へ」をクリックします。「完了」をクリックすると、中間 UDDI レジストリ検索手順をバイパスしてクライアントを作成できる最後のウィザード手順が表示されます。

b. 「ブラウザを起動」をクリックします。

デフォルトの Web ブラウザがブラウザツール URL Web ページに表示されます。

c. ブラウザツールを使用して、クライアントでアクセスする Web サービスの場所を見つけます。

d. Web サービスレジストリキーをコピーして「UDDI レジストリを選択」ダイアログの「サービスキー」フィールドにペーストします。

e. 「次へ」をクリックします。

「完了」をクリックして、中間 UDDI レジストリ検索手順をバイパスしてクライアントを作成できる最後のウィザード手順が表示されます。

この後の手順では、Forte for Java 4 IDE を通じて検索を行うことを前提としています。これは「Web サービスの検索方法」の下のデフォルトのラジオボタンです。

4. 「UDDI レジストリ」リストからレジストリを選択し、「次へ」をクリックして、図 3-13 に示されているように UDDI レジストリを検索するダイアログを表示します。

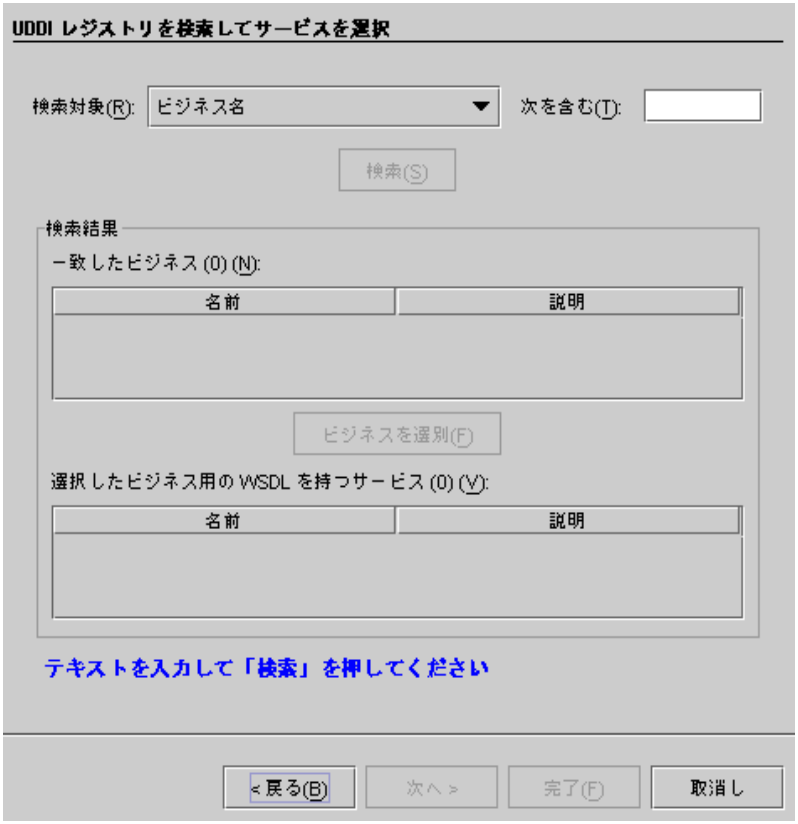

図 3-13 UDDI レジストリを検索するダイアログ

名前に指定した文字列を含むビジネスを検索できます。ビジネス名ではなく、型を基準にして検索することもできます。検索の型のリストは 図 3-14 に示されています。

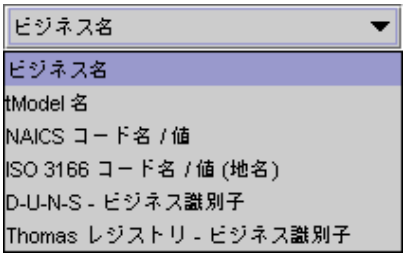

図 3-14 UDDI レジストリ検索タイプ

tModel 名、NAICS コード名、または ISO 3166 コード名を使用して検索できます。

D-U-N-S ビジネス識別子または Thomas レジストリビジネス識別子を検索することもできます。この場合、値「次を含む」ではなく「次と等しい」を検索します。前の検索による識別子がわかっている場合があります。プライベート UDDI レジストリを使用している場合は、プロジェクトまたは企業内の誰かに識別子を聞いてください。

注 - 検索タイプに正確な値が必要な場合は、「UDDI レジストリ検索」ダイアログ内のラベル (図 3-13) が「次を含む」から「次と等しい」に変わります。

NAICS または ISO 3166 コード名検索の場合は、「次を含む」フィールドに名前またはコードを入力できます。たとえば次のような検索ができます。

- ISO 3166 を選択して `United Kingdom` または `GB` の検索文字列を指定します。
- NAICS を選択して `Other Financial Vehicles` または `52599` の検索文字列を指定します。

tModel 名に対する検索が、検索文字列に合致する tModels に基づくサービスを含むすべてのビジネスを返します。

NAICS Code Name または ISO 3166 Code Name に対する検索が、検索文字列に合致するカテゴリを持つすべてのビジネスを返します。

注 - 「次を含む」フィールドに文字列を指定すると、IDE によって次のように文字列がワイルドカードにラップされます。`%`*yourstring*`%`。オプションでさらにワイルドカードを入力することができます。たとえば、名前に文字列 *bio*`%`*tech* を含むビジネスを検索できます。`%` は任意の文字列に一致します。文字列検索は大文字と小文字を区別します。

次の手順では、ビジネス名で検索を行うことを前提としています。

5. 「次を含む」フィールドに文字列を指定して、「検索」をクリックします。

   Matching Businesses 表には、図 3-15 に示されているようにレジストリ名に検索文字列を含むすべてのビジネスが表示されます。各ビジネスは名前とオプションの説明とともに一覧表示されます。一致するビジネスの数も表示されます。

   表からビジネスを選択して、この手順の手順 7 に進みます。

   検索文字列およびレジストリのサイズによっては、検索で多数のビジネスが返される場合があります。これは、次第に数十万から数百万のエントリに増加することが予測される公開レジストリで発生する可能性があります。検索によって返されたリストが大きすぎる場合は、詳細な検索文字列を入力するか、またはレジストリブラウザツールを使用して、より詳細な検索を実行します。

注 - 実際のシナリオでは、開発チームまたはプランナーはどのようなビジネスを対象とするかについて詳しいアイデアを持っていると考えられます。レジストリ内のビジネスを検出するときに大切なことは、正しいスペルを取得すること、または (多くの部門、部署およびプロジェクトを持つ巨大企業の場合は) 名前のどの部分がレジストリで使用されているかを判別することです。

図 3-15 一致するビジネスが表示された「UDDI レジストリ検索」ダイアログ

6. 「ビジネスを選別」をクリックして検索を絞ります。

   レジストリには WSDL tModel エントリを持たないサービスがある場合があります。これらのサービス用のクライアントを作成できないため、IDE では「一致したビジネス」表からこれらを除外することができます。「ビジネスを選別」をクリックすると、IDE は結果セット内のビジネスに関連付けられているサービス用の UDDI レジストリを検索します。IDE は、「`http://`」で始まり、「`wsdl`」または「`asmx`」で終

わる「オーバービュー URL」フィールドを備えた OverviewDoc を持つ tModels 用のサービスをチェックします。このテストをパスしないビジネスは「一致したビジネス」表から除外されます。

IDE は、図 3-16 に示されているように「取消し」ボタンを備えた進捗モニターウィンドウが表示されます。フィルタ処理には多少時間がかかります。時間がかかりすぎる場合は、「取消し」をクリックして、より詳細な検索文字列を使用して検索をやり直すことができます。

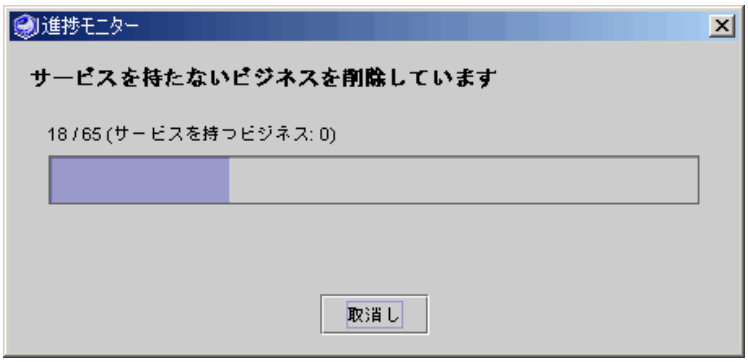

図 3-16 UDDI レジストリフィルタビジネス進捗モニター

7. 「検索結果」リストからビジネスを選択します。

図 3-17 に示されているように、選択したビジネスに関連付けられているサービスが表示され、「次へ」ボタンがアクティブになります。WSDL tModel エントリがあるサービスのみが表示されます。

注 - レジストリは、決定的な助けとなるビジネスまたはサービスに関する十分な情報を持っていない場合があります。公的レジストリは、まだ初期段階で、使用パターンおよび実践が時間経過につれて明らかになる広範囲な Web サービスインフラストラクチャの要素として表示できます。非公開レジストリでは、レジストリの所有者が、サービスを公開するための基準をより詳細に管理することができます。たとえば、非公開レジストリでは、エントリをドキュメント化する基準およびレジストリに公開されるサービスの実行可用性に関する要件がある場合があります。開発およびテスト用のサービスと生産アプリケーションで使用するサービスには区別がある場合があります。

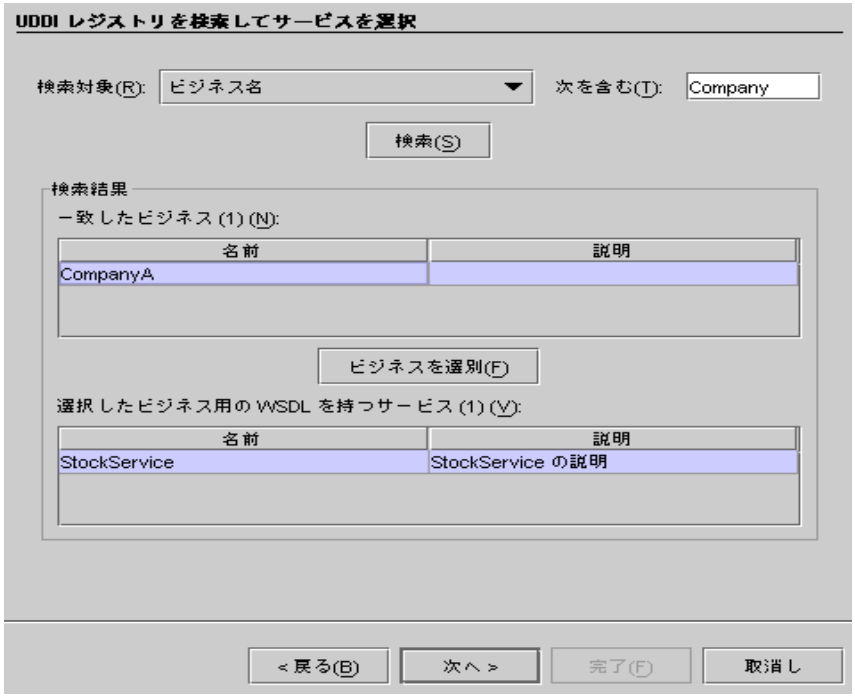

図 3-17 UDDI レジストリ選択サービス

8. サービスを選択して「次へ」をクリックし、図 3-18 に示されているように「サービスインタフェース」ダイアログを表示します。

   ダイアログに、サービスとそれが参照する tModels および WSDL に関する詳細情報が表示されます。

   サービスに対して、名前、ネットワークエンドポイント、およびキーが表示されます。ネットワークエンドポイントは、クライアントが実行サービスインスタンスにアクセスするために使用できる URL です。

   各 WSDL tModel に対して、名前およびオーバービュー URL が表示されます。オーバービュー URL は、Forte for Java 4 IDE がクライアントの作成に使用する WSDL 用のロケータです。

注 - Web サービスプロバイダは、指定サービスインスタンス用の tModels を公開する場合があります。たとえば、1 つの tModel はサービスメソッドを完全に使用できるようにしたり、別の tModel はサービスメソッドのサブセットへのアクセスを提供する場合があります。これは、2 つの異なる tModels から生成されたクライアントに反映されます。

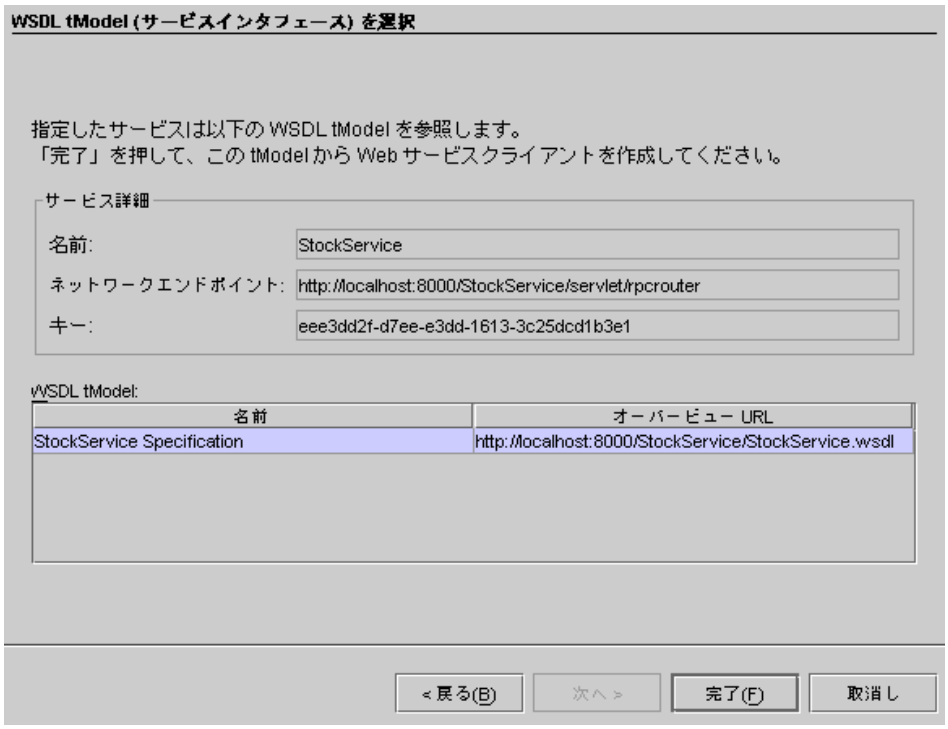

図 3-18 UDDI レジストリ表示サービス詳細と tModel

9. WSDL tModel を選択して「完了」をクリックし、クライアント作成手順を完了します。

   Forte for Java 4 IDE によって Web サービスクライアントが作成されます。エクスプローラに、指定したパッケージの下に新しいクライアントノードとして表示されます。

10. その後の手順は、61 ページの「Forte for Java Web サービスからのクライアントの作成」と同じです。

サービスエンドポイント URL

クライアントの実行時にサービスインスタンスにアクセスするには、クライアントプロキシがサービスのエンドポイント URL を持っている必要があります。IDE はプロキシ内にデフォルト URL を設定できます。または URL は実行時に `serviceURL` パラメータ内のプロキシに渡すことができます。これはデフォルトをオーバーライドします。

Forte for Java 4 IDE によって生成されたクライアント JSP ページは、実行時に URL を渡しませんが、クライアントプロキシが `serviceURL` 用のデフォルト値を持っているとみなします。デフォルト URL は、クライアントプロキシを生成するために使用する WSDL から取得されます。

この章の最初で説明したように、クライアントは、独自の Web サービス、IDE に表示される WSDL ファイル、または UDDI レジストリ内で見つけたサービスから生成することができます。これらすべてのケースで、IDE は Web サービスを説明する WSDL を含む隠しファイルを作成し、WSDL に基づいてクライアントを作成します。ファイル名拡張子 `.wsdx` を持つ隠しファイルは、IDE 内で表示または編集することはできません。クライアントと同じディレクトリ内で隠しファイルを検出して、ファイル名拡張子 `.wsdl` を付けてファイルのコピーを作成して渡し、ほかの開発者が各自の IDE のインスタンス内でクライアントを生成できるようにします。

注 - IDE 外の隠しファイル内の WSDL を編集する場合は、変更内容は次にクライアントを生成したときに失われます。

隠し WSDL ファイル内のデフォルト URL は、次のように派生されます。

- 独自の Web サービスからクライアントを生成する場合、IDE ではサービス URL をサービスの SOAP RPC URL プロパティと等しく設定します。
- 独自の Web サービスから明示的に WSDL を生成する場合、WSDL はエクスプローラに表示され、IDE はサービス URL をサービスの SOAP RPC URL プロパティと等しく設定します。WSDL を直接 XML ドキュメントとして付与された場合は、サービス URL を含んでいる場合と含んでいない場合があります。サービス URL を含むファイルの名前には `.wsdl` 拡張子が付いていて、それを含むディレクトリが IDE にマウントされている場合に、WSDL が、エクスプローラに表示されます。

- エクスプローラ内に表示されている WSDL ファイルを選択してクライアントを生成した場合、IDE は WSDL を隠しファイルにとして `.wsdx` 拡張子を付けてコピーし、後者を使用してクライアントを生成します。WSDL がサービス URL の値を含んでいない場合は、サービス URL 用のデフォルト値を持たないプロキシが生成されます。
- UDDI レジストリからクライアントを生成する場合は、WSDL が IDE にダウンロードされ、`.wsdx` ファイル拡張子が付けられた隠しファイルとして格納されます。UDDI サービスエントリが URL を提供する場合、URL はオリジナルの WSDL にコード化されているかどうかにかかわらず、ダウンロードされた WSDL に配置されます。
- 生成されたプロキシが前述したシナリオに値が指定されていないためにデフォルト値を持たない場合は、実行時にサービス URL をプロキシに渡す必要があります。

第4章

XML オペレーションの開発

XML オペレーションは、Web サービスアクセス用に設計されていない既存のビジネスコンポーネント用の Web サービスインタフェースを作成する効率的な方法を提供します。

たとえば、顧客からの注文を受け付けるために、ビジネスコンポーネントとして在庫のチェック、顧客信用チェック、オーダーの出荷、および請求書の処理を行うメソッドを持っている Web サービスを作成しているとします。これらのメソッドを単一の XML オペレーションとして組み合わせて注文を受け付けるようにすることができます。

XML オペレーションは、直接メソッド呼び出しと並ぶ Web サービスの構築ブロックです。各 XML オペレーションでは、特定のクライアント要求に対する応答を定義します。

この章では、XML オペレーションを作成および編集する方法を説明します。また、この作業に使用するツールについても説明します。XML オペレーションの概要については、14 ページの「XML オペレーション」を参照してください。

ツールの概要

XML オペレーションの開発に使用するツールは、エクスプローラおよびソースエディタから利用できます。

エクスプローラでノードを選択してオペレーションの作成場所を指定し、「新規ウィザード」ウィザードを開いて、XML オペレーション定義を作成します。この手順は、エクスプローラでのクラスまたはその他のオブジェクトの作成手順と同様です。(実行する手順の詳細は、97 ページの「XML オペレーションの作成」を参照してください。

XML オペレーションをさらに開発するには、エクスプローラまたはソースエディタからツールを利用します。どちらのウィンドウからも同じコマンドが利用できますが、ソースエディタはコマンドを発行してその結果を同じウィンドウで参照できるため、より便利です。図 4-2 および図 4-1 は、ソースエディタで表示された XML オペレーションを示しています。

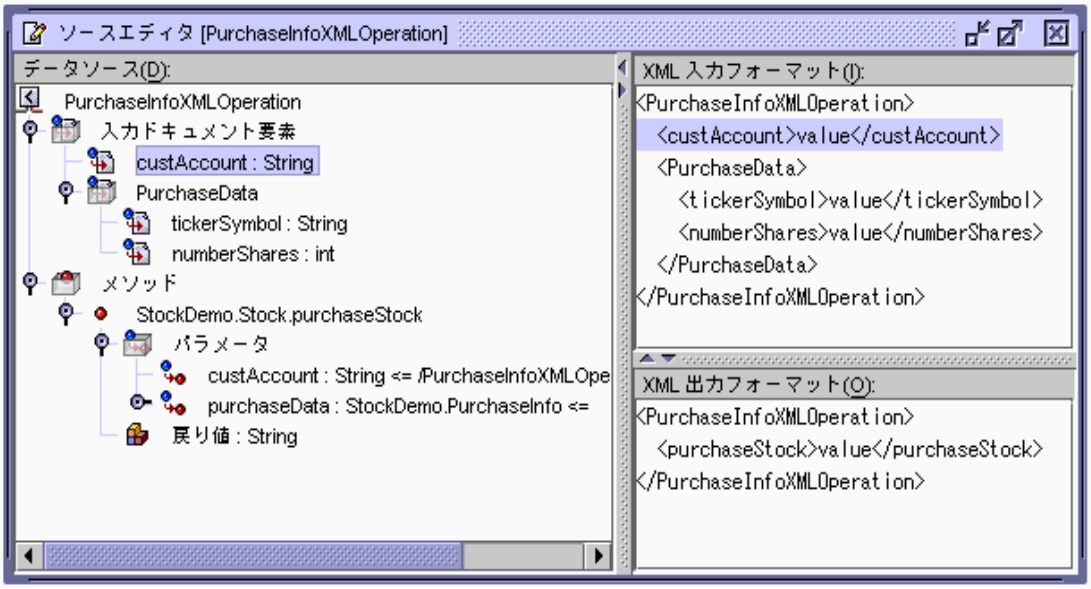

図 4-1　XML オペレーションソースエディタ。複雑な入力を表示

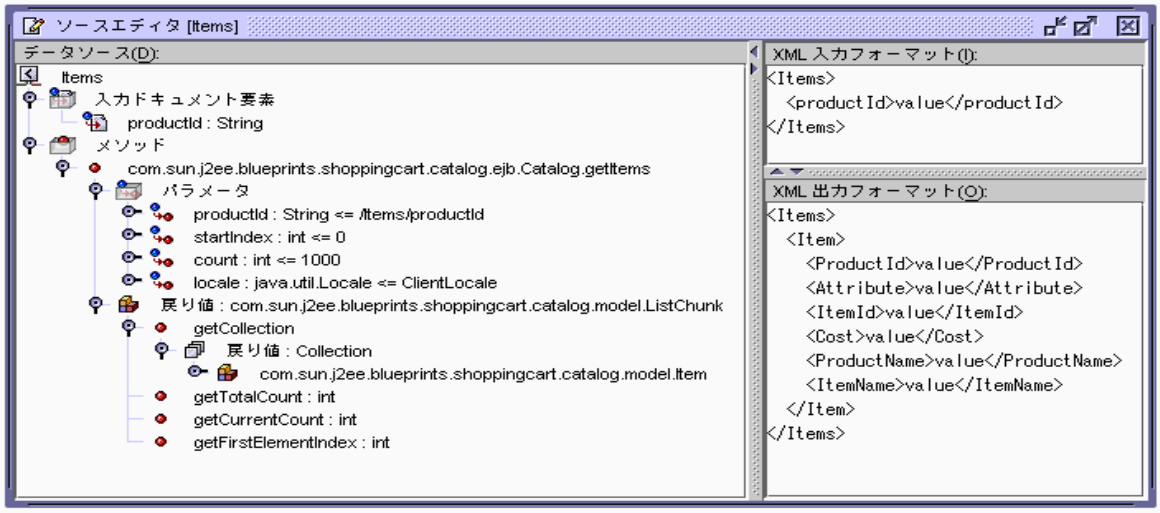

図 4-2　XML オペレーションソースエディタ。複雑な出力を表示

ソースエディタは、XML オペレーションを次の 3 つの区画に表示します。

- 「データソース」区画
  XML オペレーションで編集コマンドを表示して発行する場合にこの区画を使用します。
- 「XML 入力フォーマット」区画
  この表示専用区画には、XML 入力ドキュメントのフォーマットが表示されます。
- 「XML 出力フォーマット」区画
  この表示専用区画には、XML 出力ドキュメントのフォーマットが表示されます。

「データソース」区画

「データソース」区画には、XML オペレーションがツリー形式で表示されます。構造上の各ノードは、XML 入力ドキュメント要素、XML オペレーションによって呼び出されるメソッド、これらのメソッドに対するパラメータ、メソッドの戻り値、および体系化されたノードを表します。各ノードの型は、独自のコマンドとプロパティを持ちます。

次の方法で XML オペレーションを編集できます。

- ノードを選択してからメニューコマンドを選択します。
- ノードをダブルクリックしてプロパティシートを表示して、そのプロパティを編集します。

最上位レベルにおいて、「データソース」区画には「入力ドキュメント要素」 ノードと「メソッド」ノードの 2 つの組織ノードが含まれます。

入力ドキュメント要素ノード

XML 入力ドキュメントには、クライアント要求を定義するデータが含まれます。これは、**入力ドキュメント要素**ノードとして表されます。このノードを展開すると、XML 入力ドキュメント要素を参照して編集できます。これらの要素はサブノードとして表されます。図 4-3 は、ソースエディタで入力ドキュメント要素ノードを展開したときの表示を抜粋したものです。

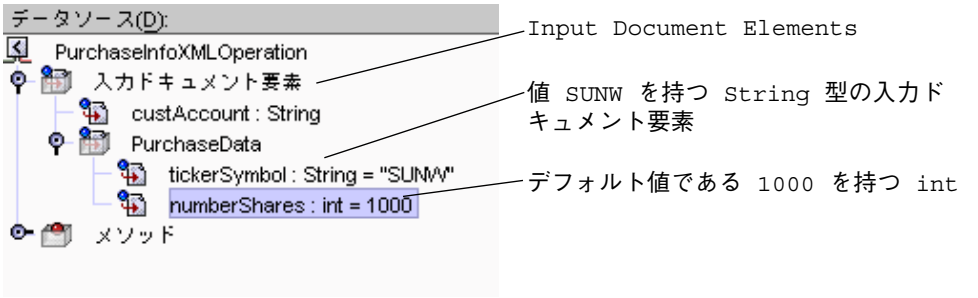

図 4-3　Input Document Elements ノード

これらのノードから、XML 入力ドキュメントに対して次のタイプの編集オペレーションを実行できます。

- 要素の追加 (103 ページの「入力ドキュメント要素の追加」を参照)
- 要素の削除 (108 ページの「メソッドまたは入力ドキュメント要素の削除」を参照)
- 要素の名前変更 (105 ページの「入力ドキュメント要素の名前変更」を参照)
- 要素の並べ替え (108 ページの「メソッドまたは入力ドキュメント要素の並び替え」を参照)
- 要素のデフォルトの値の指定 (106 ページの「入力ドキュメント要素にデフォルト値を指定する」を参照)
- 要素の固定 (107 ページの「入力ドキュメント要素を常時に設定する」を参照)

メソッドノード

Methods ノードからは、次のことが実行できます。

- XML オペレーションによって呼び出されるメソッドを指定する。オーバーロードメソッドを含む
- データソース上で呼び出しをメソッドに追加することによって、XML オペレーションでどのデータを取得するかを指定する
- メソッドパラメータをソースにマッピングすることによって、メソッドパラメータ用の値を指定する
- XML オペレーションのメソッド呼び出しによって返されるクラスを展開または縮小して、詳細データを取得する

- XML 出力ドキュメントから返されたクラスのフィールドを選択して除外し、クライアントに送信するデータの量を減らす
- メソッドによって返されたオブジェクトを Web サービス内で他の XML オペレーションで利用できるようにする
- メソッド戻り値を配置する
- 継承メソッドを表示して選択する

メソッドの実行と返されるータ

XML オペレーションは、他の実行時オブジェクト上でメソッドを実行します。**メソッドノード**にメソッド呼び出しを追加することによって XML オペレーションが実行するメソッドを指定します。メソッド呼び出しを追加することにより、XML オペレーションをプログラムしてデータを返したり、その他のタイプの処理を実行することができます。

メソッドが呼び出される順序を一旦削除して、配列し直すこともできます。XML オペレーションは、上から下に、順番にメソッドを実行します。

メソッドを追加、削除、または並び替えると、戻り値に対応する要素が追加、順序変更、および削除され、XML 出力ドキュメントに影響します。そのような変更は、XML 出力区画に表示されます。これらのトピックについての詳細は、101 ページの「XML オペレーションへのメソッドの追加」、108 ページの「メソッドまたは入力ドキュメント要素の並び替え」、108 ページの「メソッドまたは入力ドキュメント要素の削除」を参照してください。

パラメータの指定

メソッドがパラメータをとる場合は、パラメータはそのメソッドの **パラメータ**ノードの下に一覧表示されます。デフォルトでは、XML オペレーションは XML 入力ドキュメントの要素を類似した名前のパラメータにマッピングすることで、各パラメータの値を取得します。

ただし、XML 入力要素は、任意の方法でメソッドパラメータにマップし直すことができます。XML オペレーション内の 2 つのメソッドが、類似した名前のパラメータを取る場合にマップのし直しが必要になることがあります。その場合には、XML オペレー

ションはデフォルトで両方のパラメータを同じ XML 入力要素にマップします。これが適切でない場合は、新しい入力要素を作成してパラメータの 1 つをそれにマップし直します。

また、入力要素以外のソースのタイプにもパラメータをマップできます。たとえば、次のものに対してパラメータをマップできます。

- 固定値
- XML オペレーション内の別のメソッド呼び出しの戻り値
- Web サービスに定義されていて要求に応じてインスタンス化されるターゲットオブジェクト
- 別の XML オペレーションでメソッド呼び出しによって返され、明示的に共有されているオブジェクト
- システム共有オブジェクト

メソッドパラメータをソースにマップする方法についての詳細は、109 ページの「メソッドパラメータのソースへのマッピング」を参照してください。ターゲットオブジェクトについての詳細は、58 ページの「オブジェクトのインスタンス化と参照の解決」を参照してください。システム共有オブジェクトについての詳細は、114 ページの「システム共有オブジェクト」を参照してください。

展開または縮小したデータの取得

オブジェクト (またはオブジェクトの配列またはコレクション) を返すメソッド呼び出しを追加する場合、オブジェクトのクラスタイプがメソッドのサブノードとして表示されます。このクラスが文字列 `get` で始まるメソッドを含んでいる場合は、メソッドはこのクラスのサブノードとして表示されます。これらの「取得」メソッドによって返されたオブジェクトも、デフォルトでクラスノードとして表示されますが、そのメソッドを表すサブノードは表示されません。

そのようなクラスノードを選択して展開することができます。クラスノードを展開すると、クラス内の取得メソッドのすべてのノードが追加され、同様にこれらのメソッドの戻り値に対応する要素が XML 出力に追加されます。

反対に、クラスを縮小することで、取得メソッドが「データソース」区画に表示されないようにすることもできます。また、メソッドを個別に削除することもできます。どちらの場合も、XML オペレーションが実行されたときにはメソッドは呼び出されません。これらのメソッドに対応する XML 出力の要素は、自動的に削除されます。

クラスを展開または縮小するたびに、XML 出力に対応する変更は「XML 出力フォーマット」区画に表示されます。このトピックについての詳細は、114 ページの「クラスを展開する」を参照してください。

クライアントに返されるデータの調整

返されたクラスが XML 出力に取り込みたくないデータを保持している場合があります。たとえば、顧客アカウントクラスが「アカウント ID」フィールドと、内部目的にのみ使用される対応する取得メソッドを持っている場合があります。そのような場合は、XML 出力から、このメソッドに対応する要素を選択して除外することができます。

XML 出力から要素を除外しても、XML オペレーションによって呼び出されるメソッドまたはこれらのメソッドにより、XML オペレーションに返されるデータセットに影響はありません。実行により影響を受けるのは、XML 出力ドキュメントからクライアントに返されるデータセットのみです。不必要な要素を除外することによって、アプリケーションコンテナ間で渡されるデータを最小限におさえることができ、パフォーマンスが最適化されます。このトピックについての詳細は、112 ページの「XML 出力ドキュメントから要素を除外する」を参照してください。

開発ワークフロー

XML オペレーションを開発する手順の概要は、次のとおりです。

1. XML オペレーションを作成します。

   この手順では、次を持つ XML オペレーションが作成されます。

   - 1 つのメソッド呼び出し
   - メソッドパラメータに基づいたデフォルト XML 入力ドキュメント
   - メソッドの戻り値に基づいたデフォルト XML 出力ドキュメント
   - XML 入力要素のメソッドパラメータに対するデフォルトマッピング

2. (オプション) 次の手順のいくつかまたはすべてを実行して XML オペレーションを編集します。

   - メソッド呼び出しを追加または削除します。
   - 入力ドキュメント要素を追加、削除、または名前変更します。
   - メソッドパラメータをソースにマップします。
   - 返されたオブジェクトを共有します。
   - 返されたクラスを展開または縮小します。
   - 要素の名前を変更するか、XML 出力ドキュメントから削除します。

3. XML オペレーションを Web サービスに追加します。

   Web サービスがない場合は、作成する必要があります。詳細は、20 ページの「SOAP RPC Web サービスの作成」 および 22 ページの「メソッド、XML オペレーション、JAR ファイルへの参照の追加」を参照してください。

4. XML オペレーションを Web サービスでテストします。

   このトピックについての詳細は、35 ページの「テストクライアントの作成」を参照してください。

5. XML オペレーションを編集して Web サービスの実行時クラスを生成し直して、条件を満たすまでテストします。

XML オペレーションの作成

XML オペレーションは単一で作成するか、またはエンタープライズ Bean に基づいて XML オペレーションのグループを生成することができます。

XML オペレーションの作成

XML オペレーションを作成する手順は、次のとおりです。

1. エクスプローラで、XML オペレーションを作成するフォルダをマウスの右ボタンでクリックし、「新規」>「Web サービス」>「XML オペレーション」を選択します。

   図 4-4に示されている「 新規ウィザード - XML オペレーション」ダイアログが表示されます。

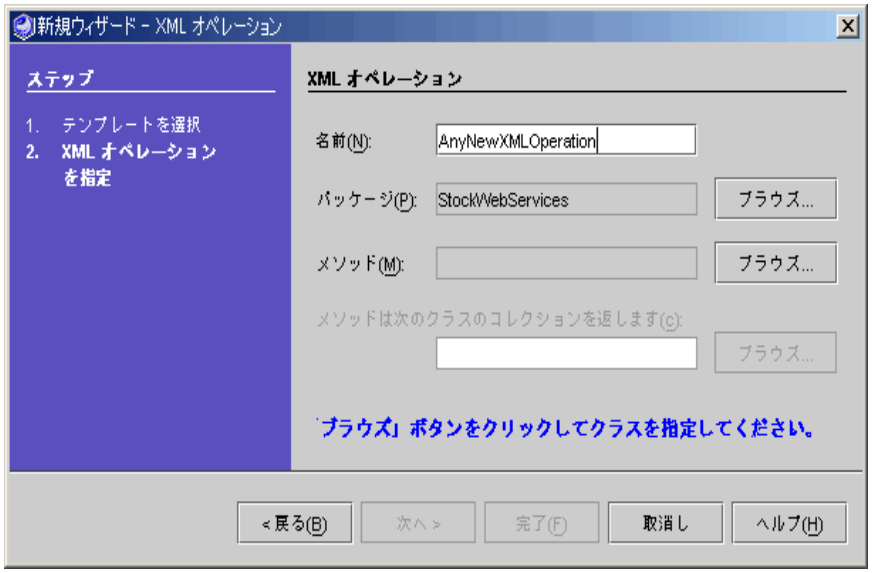

図 4-4　「新規ウィザード - XML オペレーション」ダイアログ

2. 「名前」フィールドで、XML オペレーションの名前を入力します。
3. 「パッケージ」フィールドで XML オペレーションを作成する場所が正しく指定していることを確認します。
4. 「メソッド」フィールドの横にある「ブラウズ」ボタンをクリックします。

   図 4-5に示されている「メソッドの選択」ダイアログが開きます。

図 4-5　「メソッドの選択」ダイアログ

5. XML オペレーションに取り込むメソッドを指定して、「了解」ボタンをクリックします。

   メソッドは、クラス、Bean、インタフェース、EJB コンポーネントから追加することができます。

   EJB のホームおよびリモートインタフェース内に定義されているメソッドを含めることができますが、EJB ローカルインタフェース内にのみ定義されているメソッドを含めることはできません。

   EJB メソッドを追加している場合は、EJB Bean クラスまたはホームインタフェースやリモートインタフェースを表すノードでなく、論理 EJB ノード (Bean アイコン付きのノード ) にブラウズしてください。論理 EJB ノードからメソッドを追加することによって、メソッドを呼び出すために必要な実行時情報を提供できます。XML オペレーションを作成するときに、取り込めるメソッドは 1 つです。XML オペレーションでさらにメソッドが必要な場合は、後から追加することができます。XML オペレーションへのメソッドの追加については、101 ページの「XML オペレーションへのメソッドの追加」を参照してください。

6. 選択したメソッドが配列またはコレクションを返す場合は、その中に含まれるオブジェクトのクラス、親クラス、またはインタフェースを選択します。

   「XML オペレーション」ダイアログの「メソッドはこのクラスのコレクションを戻します」フィールドの横にある「ブラウズ」ボタンをクリックします。ファイルブラウザが開きます。ファイルブラウザを使用して、クラスまたはインタフェースを選択します。

7. 「完了」をクリックします。

   XML オペレーションが作成され、ソースエディタ内で表示されて、編集することができます (図 4-2 参照)。XML オペレーションの編集についての詳細は、101 ページの「XML オペレーションの編集」を参照してください。

エンタープライズ Bean からの XML オペレーションの生成

XML オペレーションを単一で作成する他に、エンタープライズ Bean、EJB モジュール、あるいは 1 つまたは複数のエンタープライズ Bean を含むパッケージに基づいて XML オペレーションのグループを生成することができます。これにより、各 エンタープライズ Bean のホームインタフェースおよびリモートインタフェース上の各メソッドに対して 1 つの XML オペレーションが生成されます。生成された XML オペレーションへの参照は、自動的に Web サービスに追加されます。

エンタープライズ Bean から XML オペレーションを生成するには、先に Web サービスを作成している必要があります。新規 Web サービスの作成手順についての詳細は、20 ページの「SOAP RPC Web サービスの作成」を参照してください。

エンタープライズ Bean から XML オペレーションを作成する手順は、次のとおりです。

1. Web サービスをマウスの右ボタンでクリックし、「EJB からオペレーションを作成」を選択します。

   ファイルブラウザが表示されます。

2. エンタープライズ Bean、EJB モジュール、またはエンタープライズ Bean を含むパッケージにブラウズして「了解」をクリックします。

3. 「完了」をクリックします。

4. クラス、親クラス、またはエンタープライズ Bean 内のメソッドによって返された配列またはコレクションに含まれているオブジェクトのインタフェースを指定します。

   メソッドが配列またはコレクションを返した場合は、図 4-6 に示されているように「何の Collection ですか ?」ダイアログが表示されます。

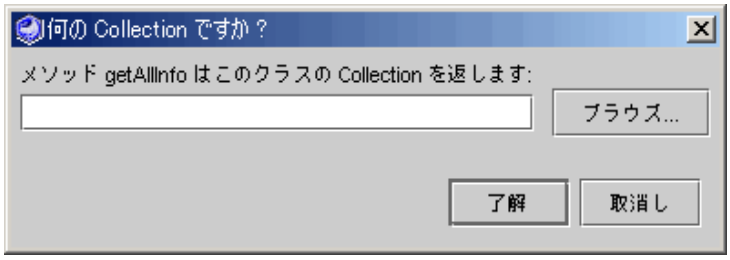

図 4-6　「何の Collection ですか ?」ダイアログ

コレクションを返すメソッドの名前がダイアログに示されます。

オブジェクト型を指定する手順は次のとおりです。

a. 「ブラウズ」ボタンをクリックします。

   ファイルブラウザが表示されます。

b. クラス、親クラス、あるいは、配列またはコレクションに含まれているオブジェクトのインタフェースに移動して「了解」をクリックします。

注 - このダイアログは、配列またはコレクションを返す各メソッドに対して 1 回ずつ、自動的に表示されます。

XML オペレーションが生成され、それらへの参照が Web サービスに追加されます。必要のない XML オペレーションを削除して、その他のオペレーションを要件に応じて編集することができます。

XML オペレーションの編集

ここでは、XML オペレーションを編集する方法を説明します。

XML オペレーションへのメソッドの追加

XML オペレーションにメソッドを追加する手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開きます。
2. 「データソース」区画で、メソッドノードをマウスの右ボタンでクリックし、「メソッドを追加」を選択します。

   図 4-7に示されている「メソッドを追加」ダイアログが表示されます。

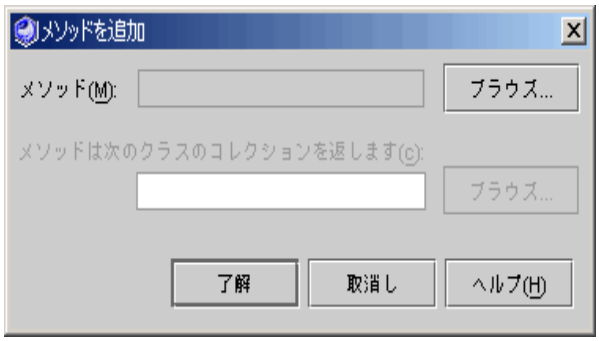

図 4-7 「メソッドを追加」ダイアログ

3. 「ブラウズ」をクリックしてメソッドを選択します。

   図 4-8 に示されている「メソッドの選択」ダイアログが表示されます。

図 4-8　「メソッドの選択」ダイアログ

4. XML オペレーションに取り込むメソッドを指定して、「了解」ボタンをクリックします。

   メソッドは、クラス、Bean、インタフェース、EJB コンポーネントから追加することができます。

   EJB のホームおよびリモートインタフェース内に定義されているメソッドを含めることができますが、EJB ローカルインタフェース内にのみ定義されているメソッドを含めることはできません。

   EJB メソッドを追加している場合は、EJB Bean クラスまたはホームインタフェースやリモートインタフェースを表すノードでなく、論理 EJB ノード (Bean アイコン付きのノード) にブラウズしてください。論理 EJB ノードからメソッドを追加することによって、メソッドを呼び出すために必要な実行時情報を提供できます。

5. 選択したメソッドが配列またはコレクションを返す場合は、その中に含まれるオブジェクトのクラス、親クラス、またはインタフェースを選択します。

   「メソッドを追加」ダイアログの「メソッドは次のクラスのコレクションを返します」フィールドの横にある「ブラウズ」ボタンをクリックします。ファイルブラウザが開きます。ファイルブラウザを使用して、クラスまたはインタフェースを選択します。

6. 「了解」をクリックします。

   このアクションにより、次の結果が得られます。

   - メソッドがメソッドノードに追加される
   - メソッドに対するパラメータがパラメータノードに追加される

- パラメータに対応する要素が入力ドキュメント要素ノードと「XML 入力フォーマット」区画に追加されまる。これらの要素は、パラメータに値を指定するためにマップされる。
- メソッドの戻り値に対応する要素は、「XML 出力フォーマット」区画に追加されまる。

注 - 後からメソッドを別のパッケージに移動するか、またはメソッドのクラスをメソッドのシグニチャーが変更されるような方法で編集する場合は、XML オペレーションからメソッド呼び出しを削除して追加し直す必要があります。メソッドシグニチャーを変更すると、名前、パラメータ、戻り値のクラス、および例外のリストへの変更が含まれます。

入力ドキュメント要素の追加

入力ドキュメント要素を追加する手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開きます。
2. 「データソース」区画で、**入力ドキュメント要素**ノードをマウスの右ボタンでクリックし、「入力ドキュメント要素を追加」を選択します。

   図 4-9 に示されている「入力ドキュメント要素の追加」ダイアログが表示されます。

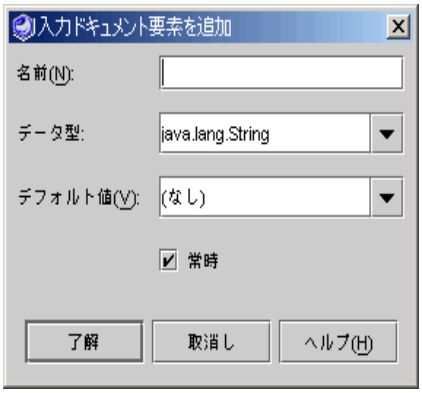

図 4-9　「入力ドキュメント要素の追加」ダイアログ

3. 「名前」フィールドに要素の名前を入力します。

4. 「データ型」コンボボックスをクリックして、要素のデータ型を選択します。

   この入力ドキュメント要素をターゲットオブジェクトをインスタンス化するためのパラメータとして使用する場合は、(`String` のように) クラスをデータ型として選択する必要があります。プリミティブ (`int` または `double` のように) は動作しません。

5. (オプション) 要素のデフォルトの値を指定します。

   要素のデフォルトの値を指定する場合は、「デフォルト値」フィールドに入力します。

   入力した値は、クライアント要求がこの要素を提供しない場合に使用されます。

6. この要素がメソッドパラメータにそれ以上マップされなくなったときに自動的に削除されるようにする場合は、「常時」チェックボックスの選択を解除します。

   メソッドパラメータのソースをマップするたびに、メソッドパラメータに現在マップされていない入力要素が削除されます。

7. 「了解」をクリックします。

   新規入力要素が「データソース」区画の入力ドキュメント要素ノードに追加されます。「XML 入力フォーマット」区画には、更新された XML 入力ドキュメントが表示されます。

入力ドキュメント要素の名前変更

入力ドキュメント要素の名前を変更する手順は、次のとおりです。

1. 図 4-10 に示されているように、入力ドキュメント要素を選択して「プロパティ」ウィンドウを開きます。
2. 「名前」プロパティをクリックします。

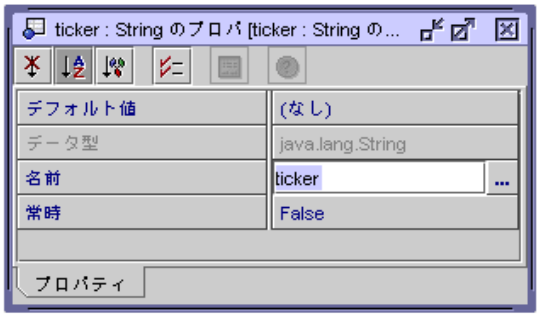

図 4-10 入力ドキュメント要素のプロパティダイアログ

3. 入力ドキュメント要素の名前を入力して Enter キーを押します。

   入力ドキュメント要素の名前が変更されます。新しい名前は入力ドキュメント要素内のノードと「XML 入力フォーマット」区画の両方に追加されます。

出力ドキュメント要素の名前変更

XML 出力ドキュメントの各要素は、デフォルトで要素の値を返すメソッドにちなんだ名前が付けられます。たとえば、append という名前のメソッドに呼び出しを追加すると、append という名前の XML 出力ドキュメントに要素が追加されます。getName という名前のメソッドに呼び出しを追加すると、Name という名前の要素が追加されます。これらの要素は、メソッド呼び出しの「タグ名」プロパティの値を変更することによって名前を変更することができます。

出力ドキュメント要素の名前を変更する手順は、次のとおりです。

1. 図 4-11 に示されているようにメソッドフォルダ内で、要素の値を返すメソッドを選択して、メソッドの「戻り値」ノードを選択し、「プロパティ」ウィンドウを表示します。
2. 「タグ名」プロパティをクリックします。

図 4-11 出力ドキュメント要素のプロパティダイアログ

3. 要素の名前を入力して Enter キーを押します。

   出力ドキュメント要素の名前が変更されます。新しい名前が「XML 出力フォーマット」区画に表示されます。

入力ドキュメント要素にデフォルト値を指定する

入力ドキュメント要素にデフォルト値を指定する手順は、次のとおりです。

1. 図 4-12 に示されているように入力ドキュメント要素のプロパティシートを開きます。

2. 「デフォルト値」プロパティをクリックします。

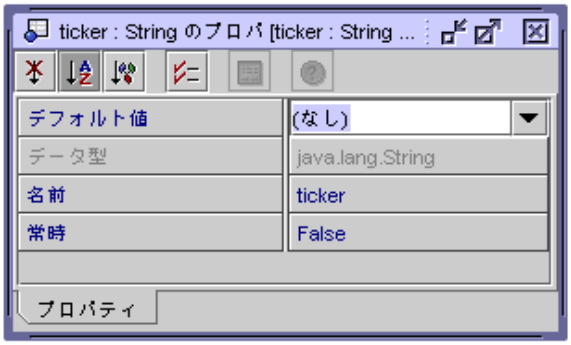

図 4-12 入力ドキュメント要素プロパティ (デフォルト値)

3. 値を入力するか、null 値を指定する場合は「(なし)」を選択します。

   この値は、クライアント要求がこの要素を提供しない場合に使用されます。

入力ドキュメント要素を常時に設定する

IDE では、入力ドキュメント要素の「常時」プロパティが有効になっていない場合は、メソッドパラメータにマップされない入力ドキュメント要素を自動的に削除します。

たとえば、XML オペレーションに対するパラメータをとるメソッドを追加すると、IDE では入力ドキュメント要素を自動的に追加してメソッドパラメータにマップします。次にそのメソッドパラメータを別のソース (たとえば、別のメソッドの戻り値)にマップし直すと、その入力ドキュメント要素はそれ以上メソッドパラメータにマップされないため、IDE によって削除されます。

入力ドキュメント要素がマップされない場合に自動的に削除されないようにするには、「常時」プロパティを有効にします。

入力ドキュメントの「常時」プロパティを有効にする手順は、次のとおりです。

1. 図 4-12 に示されているように入力ドキュメント要素のプロパティシートを開きます。
2. 「常時」プロパティをクリックします。
3. コンボボックスで「True」を選択して Enter キーを押します。

   入力ドキュメント要素が常時に設定されます。

メソッドまたは入力ドキュメント要素の並び替え

XML オペレーション内の入力ドキュメント要素とメソッドを並び替えることができます。

メソッドを並び替えると、メソッドが呼び出される順序が変更され、XML 出力ドキュメント内の要素の順序が変更されます。メソッドはソースエディタ内に一覧されている順序で上から下へ呼び出されます。

XML オペレーション内のメソッドの戻り値は、XML オペレーション内の別のメソッドへのパラメータとして使用できます (このトピックについての詳細は、109 ページの「メソッドパラメータのソースへのマッピング」を参照してください)。 XML オペレーションにそのような依存性がある場合は、パラメータを指定しているメソッドがパラメータを要求するメソッドの前に呼び出されていることを確認する必要があります。

入力ドキュメント要素を並び替える機能は、開発の簡便性のためのものです。実行時には、入力ドキュメント要素の順序は Web サービスに大きな影響を与えません。

メソッドまたは入力ドキュメント要素を並び替える手順は、次のとおりです。

1. ソースエディタ内で XML オペレーションを開いて、並び替えるメソッドまたは入力ドキュメント要素を選択します。
2. メソッドまたは入力ドキュメント要素をマウスの右ボタンでクリックし、「上へ移動」または「下へ移動」を選択します。

メソッドまたは入力ドキュメント要素の削除

XML オペレーションから入力ドキュメント要素とメソッドを削除することができます。メソッドを削除すると、XML オペレーションが実行されたときにメソッドは呼び出されません。削除されたメソッドに対応する XML 出力の要素も、削除されます。

メソッドまたは入力ドキュメント要素を削除する手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開いて、削除するメソッドまたは入力ドキュメント要素を選択します。
2. メソッドまたは入力ドキュメント要素をマウスの右ボタンでクリックし、「削除」を選択します。

メソッドパラメータのソースへのマッピング

メソッドパラメータをソースにマップする手順は、次のとおりです。

1. ソースエディタ内で XML オペレーションを開いて、マッピングするメソッドパラメータを選択します。
2. パラメータをマウスの右ボタンでクリックし、「ソースの変更」を選択します。

   図 4-13　に示されている「メソッドパラメータソース」ダイアログが表示されます。

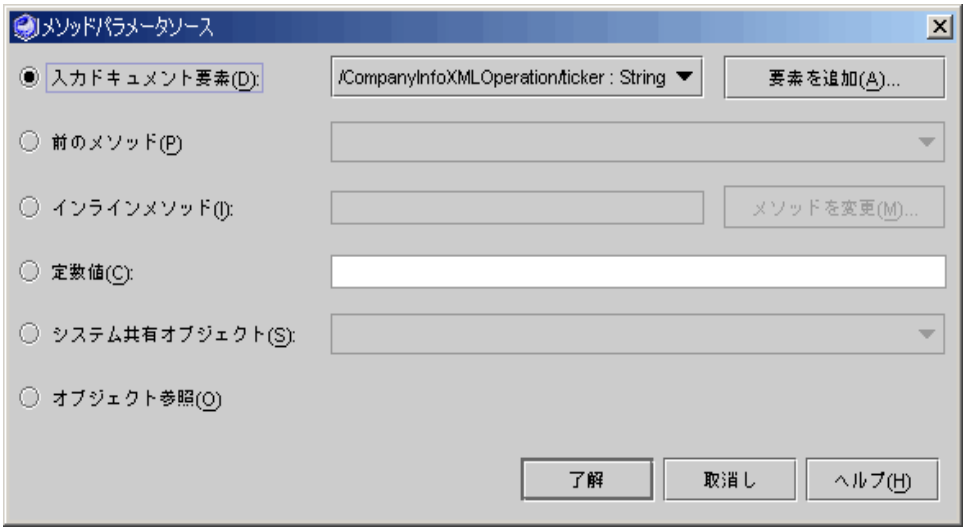

図 4-13　「メソッドパラメータソース」ダイアログ

3. ソースのタイプを選択します。

   次の表は、利用可能なソースタイプの説明です。

ソースタイプ	説明
入力ドキュメント要素	パラメータを XML 入力ドキュメントの要素にマップする場合に、このラジオボタンを選択します
前の メソッド	パラメータをこの XML オペレーションの前のメソッド呼び出しの戻り値にマップする場合に、このラジオボタンを選択します
インライン メソッド	パラメータをインラインメソッド呼び出しの戻り値にマップする場合に、このラジオボタンを選択します。「メソッドを変更」をクリックして目的のメソッドを選択します

ソースタイプ	説明
定数値	パラメータを定数値にマップする場合に、このラジオボタンを選択します
システム共有オブジェクト	Web サービスは、Web サービスにアクセスする Web ブラウザクライアントに関するデータを含むシステム共有オブジェクトを実行時に保持します。システム共有オブジェクトは、Web サービスによってインスタンス化され、HTTP 要求および J2EE セキュリティメカニズムから取得されたデータが入力されます システム共有オブジェクトは、ユーザー名に関するデータを保持します。データは `String` オブジェクトおよび `java.security.Principal` オブジェクトとして提供されます パラメータをこれらのタイプのデータの 1 つにマップする場合に、このラジオボタンを選択します。
オブジェクト参照	パラメータを Web サービスに定義されているターゲットオブジェクトにマップする場合に、このラジオボタンを選択します。このオプションは、パラメータのタイプがクラス (`String` を含む) の場合にのみ利用可能です このラジオボタンを選択すると、要求されたクラスの新しいオブジェクトにパラメータをマップする Web サービス内に、デフォルトオブジェクト参照が追加されます。この参照を構成し直すことて、選択したオブジェクトを解決できます。このトピックについての詳細は、58 ページの「オブジェクトのインスタンス化と参照の解決」を参照してください

4. パラメータの値のソースを指定します。

   ソースのタイプが「入力ドキュメント要素」、「メソッドの戻り値」、「定数値」、または「システム共有オブジェクト」に設定されている場合は、有効になっているフィールドを使用してソースを指定します。次の表は、指定されたソースのタイプに依存して実行される動作について説明しています。

フィールド名	動作
入力ドキュメント要素	有効になっているコンボボックスから入力ドキュメント要素を選択します。コンボボックスには、パラメータで必要な型の XML オペレーションに含まれるすべての入力ドキュメント要素が一覧表示されます。入力ドキュメント要素を選択すると、その値がパラメータにマップされます
前のメソッドまたはインラインメソッド	有効になっているコンボボックスからメソッドを選択します。コンボボックスには、パラメータに適切した型を返す現在のメソッドの前に呼び出されるメソッドが一覧表示されます。メソッドを選択すると、その戻り値がパラメータにマップされます
定数値	有効になっているフィールドで、値を示す文字列を入力します。`String` または `char` 型の値の場合は、値の一部である場合以外は引用符を入力しないでください
システム共有オブジェクト	リストからオブジェクトを選択します。利用できるオブジェクトのリストは、パラメータの型によって異なります。 `java.security.Principal` 型のパラメータの場合は、コンボボックスに 1 つのオブジェクト `UserPrincipal` が表示されます `java.lang.String` 型のパラメータの場合は、コンボボックスに 1 つのオブジェクト `UserName` が表示されます

5. 「了解」をクリックします。

メソッド戻り値を配置する

メソッド戻り値を配置する手順は、次のとおりです。

1. ソースエディタ内で XML オペレーションを開いて、メソッドノードの下の戻り値 `Returns` ノードを見つけます。
2. ノードをマウスの右ボタンでクリックし、「プロパティ」を選択します。

   プロパティの 1 つは「キャスト」で、値はデフォルトの (なし) です。もう 1 つのプロパティは「データ型」です。「キャスト」の値は、Java 規則に準拠している任意の型に変更できます。
3. 「了解」をクリックします。

継承メソッドを表示および選択する

継承メソッドを表示する手順は、次のとおりです。

1. ソースエディタ内で XML オペレーションを開いて、ノードの下の戻り値ノードを見つけます。

   または、エクスプローラで、戻り値ノードを探すこともできます。
2. ノードをマウスの右ボタンでクリックし、「展開」を選択します。

   「展開」ウィンドウが表示され、取得メソッドが自動的に選択されます。
3. 「継承されたメソッドを表示」チェックボックスを選択します。

   継承されたメソッドが表示されます。
4. 目的のメソッドを選択して「了解」をクリックします。

   選択したメソッドがエクスプローラおよび「データソース」区画に表示されます。

XML 出力ドキュメントから要素を除外する

XML 出力ドキュメントから要素を除外する手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開きます。
2. 「データソース」区画で、メソッドノードを開きます。

3. 除外する要素に対応する「データソース」区画のノードを識別します。

   「データソース」区画でノードを選択すると、対応する要素が「XML 出力フォーマット」区画で強調表示されます。たとえば、図 4-14 は、「データソース」区画内で選択されたクラスノードおよび「XML 出力フォーマット」区画でそれに対応する強調表示されている要素を示しています。

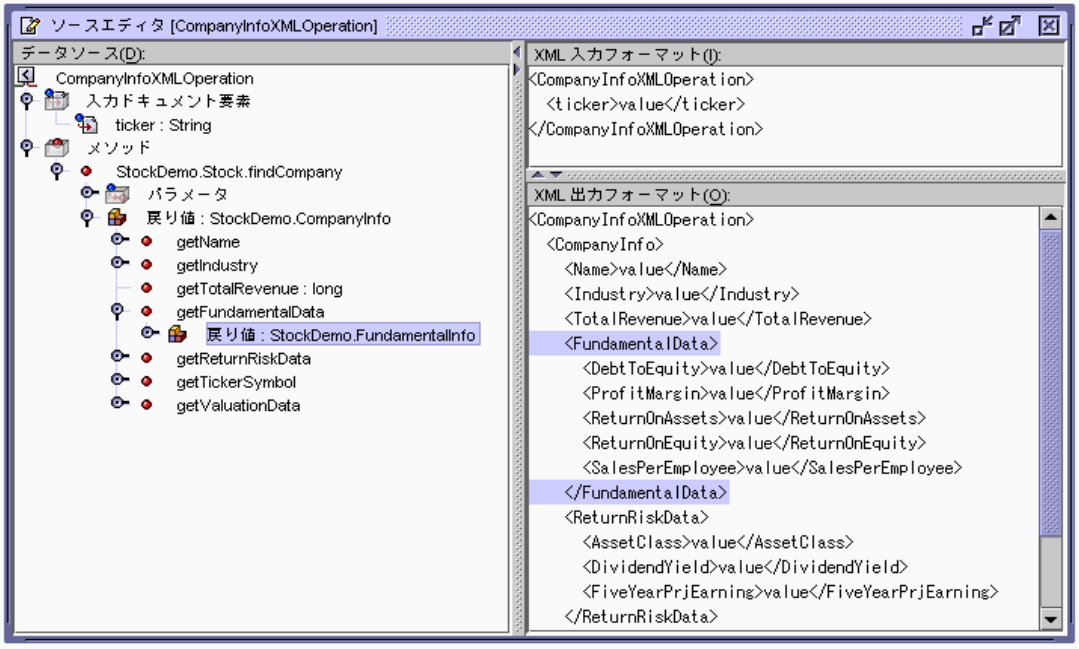

図 4-14　ソースエディタ: (出力要素を除外する)

4. ノードをマウスの右ボタンでクリックし、「出力からタグを省略」を選択します。

   対応する要素が「XML 出力フォーマット」区画から削除されます。

XML 出力ドキュメントに要素を含める

デフォルトで、XML オペレーションのメソッド呼び出しの戻り値はすべて、XML 出力ドキュメントに含まれます。これらの戻り値を表す要素のいずれかを除外した場合は、XML 出力ドキュメントに追加し直すことができます。

XML 出力ドキュメントに要素を含める手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開きます。

2. 「データソース」区画で、メソッドノードを開きます。
3. 含める要素に対応するノードを識別します。

   含めることができるのは、配列またはコレクション、クラス、またはプリミティブを返すメソッドを表す型のノードです。
4. ノードをマウスの右ボタンでクリックし、「出力にタグを取り込み」を選択します。

   選択したノードに対応する要素が「XML 出力フォーマット」区画に追加されます。

クラスを展開する

クラスを展開する手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開きます。
2. 「データソース」区画で、メソッドノードを開きます。
3. クラスをマウスの右ボタンでクリックして「展開」を選択します。

   クラス上のすべての取得メソッドはサブノードとし、クラスノードに追加されます。クラスおよび取得メソッドの戻り値に対応する要素は、「XML 出力フォーマット」区画に追加されます。

クラスを縮小する

クラスを縮小する手順は、次のとおりです。

1. ソースエディタで XML オペレーションを開きます。
2. 「データソース」区画で、メソッドノードを開きます。
3. クラスをマウスの右ボタンでクリックし、「縮小」を選択します。

   クラスの取得メソッドを表すノードが「データソース」区画から削除されます。対応する要素およびクラスに対応する要素は、「XML 出力フォーマット」区画から削除されます。

システム共有オブジェクト

Web サービスは、Web サービスにアクセスする Web ブラウザクライアントに関するデータを含むシステム共有オブジェクトを実行時に保持します。システム共有オブジェクトは、Web サービスによってインスタンス化され、HTTP 要求および J2EE セ

キュリティメカニズムから取得されたデータが入力されます。システム共有オブジェクトの使用方法についての詳細は、109 ページの「メソッドパラメータのソースへのマッピング」 および 図 4-13を参照してください。

SOAP 1.1 の場合は、クライアントと Web サービス間で保持されるセッションはありません。したがって、システム共有オブジェクトはユーザー名に制限され、J2EE セキュリティメカニズムから取得され、`String` オブジェクトおよび `java.security.Principal` オブジェクトとして提供されます。

`java.security.Principal` 型のパラメータの場合は、「メソッドパラメータソース」ダイアログ内の「システム共有オブジェクト」選択肢に 1 つのオブジェクト `UserPrincipal` が表示されます。

`java.lang.String` 型のパラメータの場合は、「システム共有オブジェクト」選択肢に 1 つのオブジェクト `UserName` が表示されます。

Static ユーティリティメソッド

ビジネスメソッドによって返されたデータを、Web ページに表示するために、何らかの処理が必要になる場合があります。たとえば、EJB メソッドが `double` 型の値を返すときに、その値を通貨形式で表示したいことがあります。最適な方法は、ビジネスコンポーネントに新しいメソッドを追加するのではなく、Static ユーティリティメソッドを記述するか、再利用することです。

Static ユーティリティメソッドの構成

使いやすくするために、作成する Static ユーティリティメソッドを IDE でいくつかのクラスに分類します。

サービス固有の場所に配備される Web サービスに固有のユーティリティクラスは、Web サービスを含むパッケージなどに配置します。汎用ユーティリティクラスは、他のサービスや Web サービス開発者が簡単に再利用できるパッケージに配置します。

Static ユーティリティメソッドの使用

XML オペレーションで Static ユーティリティメソッドを使用する手順は、次のとおりです

1. エクスプローラで Static ユーティリティクラスをマウントします。

2. ソースエディタで XML オペレーションを開きます。

3. XML オペレーションにユーティリティメソッドへの呼び出しを追加します。

   この方法についての詳細は、101 ページの「XML オペレーションへのメソッドの追加」を参照してください。

   ユーティリティメソッド呼び出しは、処理するデータを返すメソッド呼び出しの後に配置する必要があります。メソッドを配置し直すには、マウスの右ボタンでクリックして「上へ移動」または「下へ移動」を選択します。

   ユーティリティメソッドを追加すると、メソッドの戻り値に対応する要素が XML 出力ドキュメントに追加されます。この要素は「XML 出力フォーマット」区画に表示されます。

   メソッドの入力パラメータに対応する新しい入力ドキュメント要素も追加されます。この入力ドキュメント要素は無視することができます。次の手順でメソッドパラメータソースをマップし直すと削除されます。

4. ユーティリティメソッドの入力パラメータを、データコンポーネントに対するメソッド呼び出しによって返された値にマップします。

   データコンポーネントに対するメソッド呼び出しでは、ユーティリティメソッドで処理する値が返されます。そのため、データコンポーネントからの出力を、ユーティリティメソッドの入力にマップする必要があります。

   a. ユーティリティメソッドノードを展開して、パラメータノードを展開します。

   b. パラメータをマウスの右ボタンでクリックし、「ソースの変更」を選択します。

      図 4-15 に示されている「メソッドパラメータソース」ダイアログが表示されます。

   c. 「前のメソッド」を選択します。

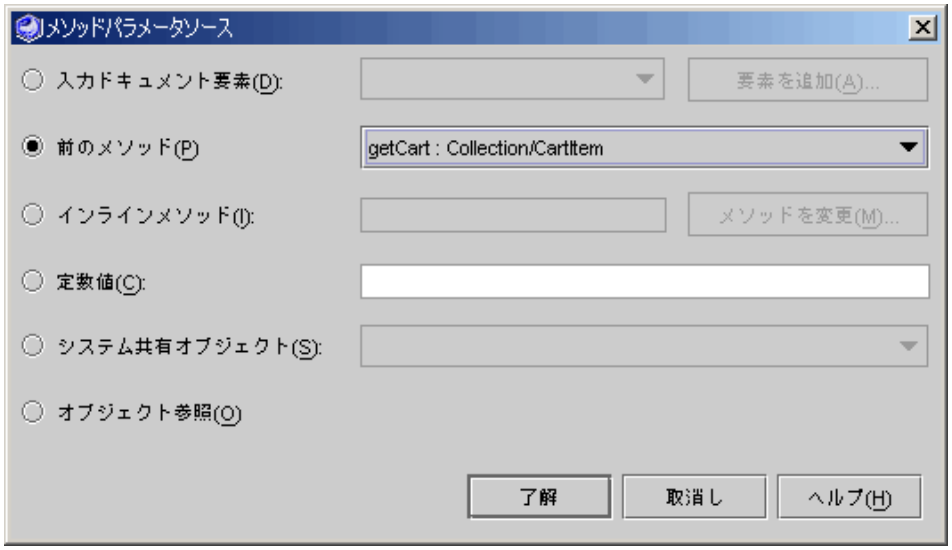

図 4-15 「メソッドパラメータソース」ダイアログ

d. 有効になっているコンボボックスで、データコンポーネントからデータを返すメソッドを選択して「了解」をクリックします。

パラメータのマッピングについての詳細は、図 4-13 および 109 ページの「メソッドパラメータのソースへのマッピング」を参照してください。

5. データコンポーネントのメソッド呼び出しの戻り値を XML 出力ドキュメントから除外するには、データコンポーネントを呼び出すメソッドをマウスの右ボタンでクリックし、「出力からタグを省略」を選択します。

   要素が「XML 出力フォーマット」区画から削除されます。クライアントでデータを処理する必要がありますが、多くの場合、データコンポーネントによって返される raw データではありません。

6. (オプション) ユーティリティメソッドの「タグ名」プロパティを適切な名前に設定します。

7. (オプション) 出力ドキュメント要素の名前を変更します。

   ユーティリティメソッドを XML オペレーションに追加すると、対応する要素が XML 出力ドキュメントに追加されます。デフォルトで、この要素にはメソッドにちなんだ名前 (`formatAsDollars` など) が付けられます。多くの場合、別の名前 (`Price` など) のほうが適しています。要素の名前を変更するには、ユーティリティメソッドの「タグ名」プロパティの値を変更します。この方法についての詳細は、105 ページの「出力ドキュメント要素の名前変更」を参照してください。

付録 A

C++ 共用ライブラリの統合

この付録では、Web サービス内の C および C++ ネイティブライブラリの機能を使用する方法を説明しています。ここでは、IDE の一般的な知識があること、このマニュアルのここまでのページを読んでいること、特に Web サービスおよび Web サービスクライアントの作成方法を説明している第 2 章および第 3 章を読んでいることを前提としています。

統合プロセスには、おおまかに C++ 開発者と Java Web サービス開発者に対応する 2 つの側面があります。このマニュアルは、Java Web サービス開発者を対象に書かれています。この場合、C++ 開発者が C++ コンポーネントを準備するときにどのようなことを行うのか、一般的な知識があると Forte for Java の Web サービスからメソッドを利用するのに役立ちます。以下の項ではプロセスの C++ 側のサマリーを示します。詳細については、Forte for Java 4 IDE 内のネイティブコネクタツールのオンラインヘルプを参照してください。

C++ 開発者のタスク

C++ 開発者の役割は、ネイティブコネクタファイルおよびネイティブコネクタコンポーネントを提供することです。

ネイティブコネクタファイルは、IDE 内で Java 開発者によってメソッドを選択して直接メソッド呼出しとして、またはより複雑な XML オペレーションの一部として Web サービスに追加するために使用されます。ネイティブコネクタファイル名には、最後に `.cdf` 拡張子が付きます。このファイルはエクスプローラにオレンジの球を含んだアイコン付きで表示されます ()。ネイティブコネクタファイルは XML ドキュメントです。

ネイティブコネクタコンポーネントは、実行時に Web サービスによって使用されます。Java ラッパークラスは必須記述子およびインタフェースとともに生成されます。ネイティブ共用ライブラリ内の C++ メソッドは、生成されたラッパー内で Java メソッドとしてアクセスされます。

ネイティブコネクツール (Native Connector Tool: NCT) は、IDE 内で C++ 開発者がネイティブコネクタファイルを作成および編集し、ネイティブコネクタコンポーネントを生成するために使用します。

NCT は、ネイティブコネクタを作成するためのウィザードを提供します。ウィザードは、Java コンポーネントに対して可視にするネイティブ関数およびデータオブジェクトを選択するエディタで、Java Native Interface (JNI) を使用してネイティブコードと Java コンポーネント間のインタフェースを作成するコードジェネレータです。作成される Java ラッパークラスは、Java 言語を使用できる場所ではどこでも使用できます。

プロセスの主要ステップは、次のとおりです。

1. `-xnativeconnect` コンパイラオプションを使用してライブラリソースコードをコンパイルし、拡張ネイティブライブラリを作成します。

   この手順では、Forte Compiler Collection (FCC) 内のコンパイラを使用する必要があります。

2. 拡張ネイティブライブラリ用のネイティブコネクタファイルを作成します。

3. ネイティブコネクタファイル (NCT エディタ使用) を編集して、Java API 内で展開するオブジェクト、メソッドおよびプロパティを選択します。

   注 - メソッドを個別に追加する場合は、クラスの引数なしのコンストラクタを追加する必要があります。これは、Forte for Java 4 Web サービス要件です。「すべてを追加」をクリックしてメソッドを追加する場合は、コンストラクタは自動的に追加されます。

   ---

4. ネイティブコネクタファイルを生成して、ネイティブコネクタ (ラッパー) コンポーネントを作成します。

   この手順は、図 A-1 に説明されています。

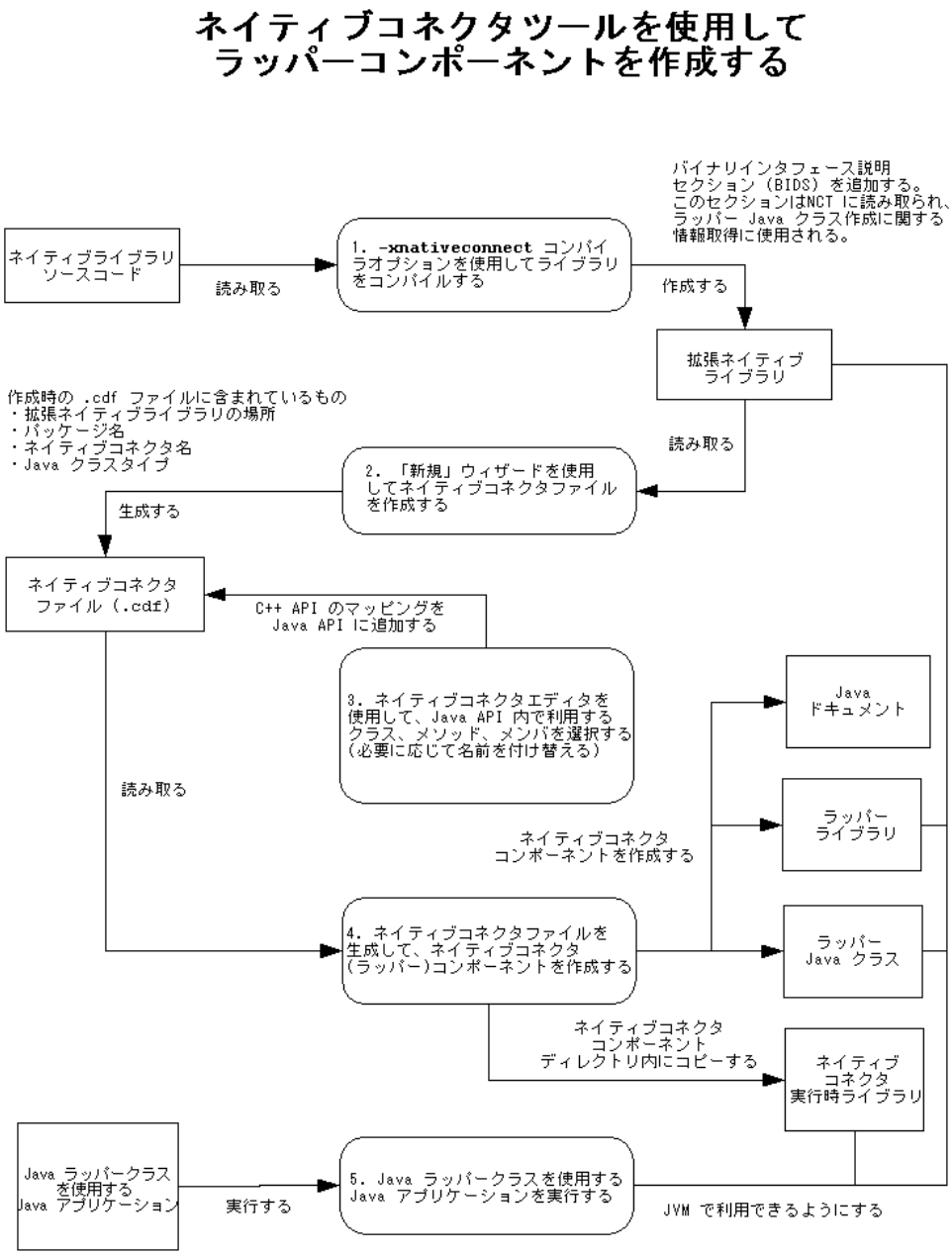

図 A-1　ネイティブコネクタツールのワークフロー

Java Web サービス開発者のタスク

Java Web サービス開発者は、C および C++ 共用ライブラリのメソッドを直接メソッド呼び出しまたは XML オペレーションの一部として展開できます。同じ Web サービスも EJB コンポーネントおよび (Web 主体アーキテクチャが使用されている場合は) Web 層コンポーネントのメソッドを展開できます。Web サービスの構築、テスト、アセンブル、および配備の方法については、第 2 章を参照してください。

次の手順では、C++ 開発者が必須 C++ 共用ライブラリおよびネイティブコネクタファイルを提供していることを前提としています。Web サービスに C または C++ メソッドを追加する手順は、次のとおりです。

1. ネイティブコネクタファイルを含むディレクトリをマウントします。

   ファイル名には、`.cdf` 拡張子が付いてる必要があります。図 A-2 は、`myConnector` という名前のネイティブコネクタノードがエクスプローラで展開されているところを示しています。利用可能メソッドは、メソッドとフィールド補助ノードに表示されます。

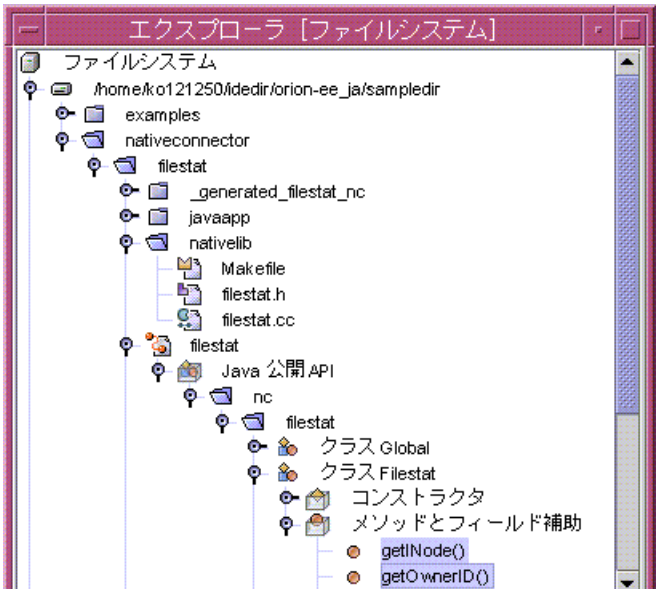

図 A-2　ネイティブコネクタノード

2. エクスプローラで、Web サービスをマウスの右ボタンでクリックし、「参照の追加」を選択します。

   ファイルブラウザが表示されます。

3. ネイティブコネクタノードを選択してファイルブラウザツリービューで展開して目的のメソッドを選択します。

   複数のメソッドを選択するには、Ctrl キーを押しながらメソッドをクリックします。

4. ファイルブラウザで「了解」をクリックします。

   選択したメソッドが Web サービスに追加されます。

   メソッドは「新規 Web サービス」ウィザードからも追加できます。ここではファイルブラウザを新しい Web サービスを作成するプロセスの一部として表示します。

注 - 手順 1 および手順 3 とは別に、これは第 2 章で説明されている、Java コンポーネントメソッドを Web サービスに展開するために使用するのと同じ手順です。

ソフトウェア要件

必須プラットフォームおよびオペレーティングシステムの詳細は、以下の Web サイトのリリースノートを参照してください。

`http://sun.co.jp/forte/ffj/documentation/index.html`

Forte for Java 4 IDE をインストールするときには、インストールウィザードを使用してコアプラットフォームおよびモジュールを含めたすべてと Solaris Developer をインストールしてください。

J2EE RI インストールのカスタマイズ

Forte for Java 4 インストーラでは、J2EE Reference Implementation を最上位レベル Forte for Java ディレクトリ `$FFJ_HOME/j2sdkee1.3.1` のすぐ下のディレクトリにインストールします。外部 RI を使用することもできます。外部 RI を使用している場合は、次の手順を変更して RI がある場所に対応するファイルを参照してください。

1. ファイル $J2EE_HOME/lib/security/server.policy を編集します。
   このファイルの下のほうに "default permissions granted to all domains" というラベルが付いたセクションがあります。最初の権限は、queuePrintJob 実行権をすべてのドメインに付与します。次の行を最初の権限の後に追加します。

   ```
   Permission java.lang.RuntimePermission "loadLibrary.*"
   ```

2. ファイル $J2EE_HOME/bin/setenv.sh を編集します。
   SYSTEM_LIB_DIR 変数を定義する行を探します。次の 2 行をその行の後に挿入します。

   ```
   LD_LIBRARY_PATH=$J2EE_HOME/nativelib
   export LD_LIBRARY_PATH
   ```

アプリケーションサーバーへの共用ライブラリのインストール

図 A-1 に示されているように実行時には 3 つの共用ライブラリが使用されます。ライブラリは、アプリケーションサーバーをホストするシステム上にインストールし、アプリケーションサーバーが認識できるようにします。

J2EE RI は、ネイティブライブラリ用の nativelib という名前のディレクトリを持ちます。3 つの共用ライブラリをこのディレクトリにコピーします。J2EE RI では LD_LIBRARY_PATH の値を使用してライブラリを検出します。

ターミナルウィンドウから次のようなコマンドを実行して共用ライブラリをインストールします。

```
cd /mydisk1/NativeConnectorDemo/_generated_NCDemo_nc/sparc-SunOS
cp *.so $FJEE_HOME/j2sdkee1.3.1/nativelib
```

付録B

オブジェクトのインスタンス化と参照の解決

XML オペレーションを開発するときに、XML オペレーションが呼び出すメソッドを指定します。これらのメソッドを実行時に呼び出すために、Web サービスは特定のオブジェクトを必要とします。各メソッド呼び出しに対して、次のものを見つけるか、またはインスタンス化します。

- メソッドが定義されているクラスのインスタンス
- メソッドによってパラメータとして要求される各クラスのインスタンス

このタスクを実行するために、Web サービスでは、これらの各ターゲットオブジェクトへの参照とターゲットオブジェクトが存在しない適切なクラスのオブジェクトをインスタンス化する方法の定義を維持します。XML オペレーションにメソッド呼び出しを追加すると、デフォルトオブジェクト参照とターゲットオブジェクト定義が自動的に Web サービスに追加されます。これらのデフォルトは一般的に適切な設定であり、編集する必要はありません。

ただし、オブジェクト参照のターゲットを手動で指定して、要件に合うように新しいターゲットオブジェクト定義を編集して作成できます。IDE 外で作成されたエンタープライズ Bean へのオブジェクト参照を手動で解決することが必要な場合があります。

この節では、次の作業を行う手順を示します。

- オブジェクト参照のターゲットを指定する
- 新規ターゲットオブジェクトを定義する
- ターゲットオブジェクト定義を編集する

オブジェクト参照のターゲットを指定する

オブジェクト参照のターゲットを指定する手順は、次のとおりです。

1. 「オブジェクト参照を解決」ダイアログを開きます。

   エクスプローラで、Web サービスをマウスの右ボタンでクリックし、「オブジェクト参照を解決」を選択します。「オブジェクト参照を解決」ダイアログ (図 B-1を参照) が表示されます。

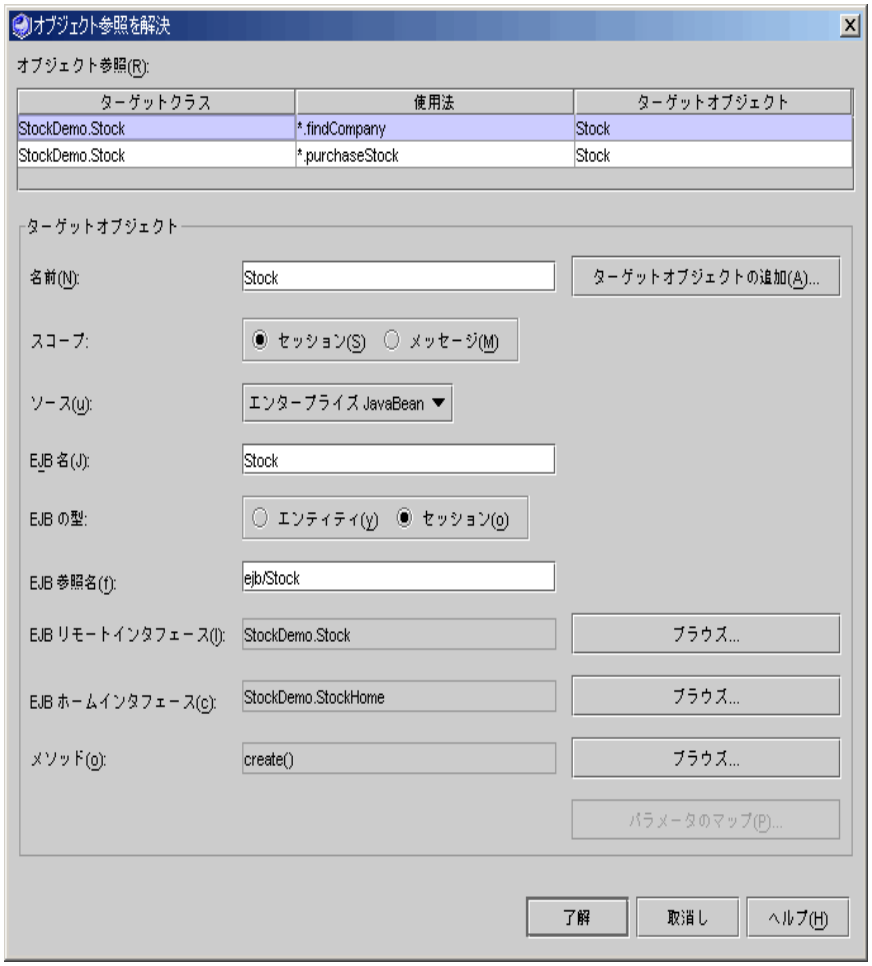

図 B-1　「オブジェクト参照を解決」ダイアログ

Web サービスが XML オペレーションを参照している場合は、図 B-2 のようにダイアログ内の表に XML オペレーション名が記載された追加の列が表示されます。

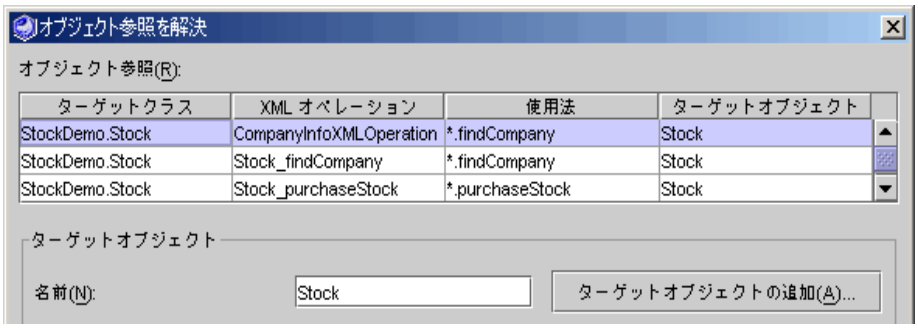

図 B-2　XML オペレーションがある「オブジェクト参照を解決」ダイアログ

2. 編集するオブジェクト参照を選択します。

   オブジェクト参照は表形式でダイアログの上部に一覧表示されます。各行は、1 つの参照を表しています。次の表は、列の説明です。

列名	説明
ターゲットクラス	参照で要求されるオブジェクトのクラス
XML オペレーション	ターゲットオブジェクトへの参照を保持する XML オペレーションの名前
使用法	XML オペレーションがターゲットオブジェクトを使用する方法を示す。使用方法は 2 つあり、XML オペレーションは次の処理を行う。 • ターゲットオブジェクト上でメソッドを呼び出す • ターゲットオブジェクトをメソッド呼び出しへのパラメータとして渡す この列では、呼び出されたメソッドの名前を提供し、メソッド呼び出しを簡単にするためのターゲットオブジェクトの使用方法を示す (アスタリスクを付ける)。 たとえば、`*.getCustomer` の値は `getCustomer` というメソッドがターゲットオブジェクト上で呼び出されることを示す。 `updateCustomer(customerInfo:*)` の値は、ターゲットオブジェクトが `customerInfo` パラメータとして `updateCustomer` メソッドに渡されることを示す。
ターゲットオブジェクト	参照を解決するオブジェクトの名前

3. 参照を解決するオブジェクトを選択します。

   編集しているオブジェクト参照を表している行で、「ターゲットオブジェクト」列をクリックしてドロップダウンリストからオブジェクトを選択します。このリストは、必要なクラスの利用可能なオブジェクトをすべて表示し、このダイアログ内に定義済みのターゲットオブジェクトを含んでいます。

新規ターゲットオブジェクトを定義する

すでに定義されているターゲットオブジェクトがオブジェクト参照に適していない場合は、新しいターゲットオブジェクトを定義できます。

新規ターゲットオブジェクトを定義する手順は、次のとおりです。

1. 「オブジェクト参照を解決」ダイアログを開きます。

   エクスプローラで、Web サービスをマウスの右ボタンでクリックし、「オブジェクト参照を解決」を選択します。「オブジェクト参照を解決」ダイアログが表示されます(図 B-1を参照)。

2. 新規ターゲットオブジェクトを定義する参照を選択します。

   ダイアログに現在のターゲットオブジェクトの定義が表示されます。

3. 「ターゲットオブジェクトの追加...」をクリックします。

   「ターゲットオブジェクトの追加」ダイアログが表示されます。

4. 「ターゲットオブジェクト名」フィールドにターゲットオブジェクトの名前を入力します。

5. 「了解」をクリックします。

   ダイアログで新しく定義されたターゲットオブジェクトへの参照が自動的に解決されます。新しいターゲットオブジェクトは既存のものと同じ定義を使用します。参照される名前のみが変更されます。定義の編集手順については、130 ページの「ターゲットオブジェクト定義を編集する」を参照してください。

ターゲットオブジェクト定義を編集する

ターゲットオブジェクト定義は、Web サービスが次の処理を行う方法を指定します。

- すでにインスタンス化されているターゲットオブジェクトを指定する
- 新規ターゲットオブジェクトをインスタンス化する

Web サービスは、同じターゲットオブジェクトに対して解決する複数の参照を持つことができます。たとえば、同じセッション Bean が Web サービス内の多くの XML オペレーションで使用されます。そのようなシナリオでは、ターゲットオブジェクト定義を編集すると、これらの参照すべてに影響します。これは、同じオブジェクトに対して解決するためです。この動作がアプリケーションに適していない場合は、1 つまたは複数の参照に対して新しいターゲットオブジェクト定義を作成する必要があります。新規定義の作成手順については、129 ページの「新規ターゲットオブジェクトを定義する」を参照してください。

ターゲットオブジェクト定義を編集する手順は、次のとおりです。

1. 「オブジェクト参照を解決」ダイアログを開きます。

   エクスプローラで、Web サービスをマウスの右ボタンでクリックし、「オブジェクト参照を解決」を選択します。「オブジェクト参照を解決」ダイアログが表示されます(図 B-1を参照)。

2. 必要に応じて「名前」フィールドを編集します。

   このフィールドでは、ターゲットオブジェクトが参照される名前を指定します。ターゲットオブジェクトの値もダイアログの上部にあるオブジェクト参照リストの「ターゲットオブジェクト」列に表示されます。このフィールドは必須です。

3. 必要に応じて「スコープ」フィールドを編集します。

   この必須フィールドでは、ターゲットオブジェクトが参照される範囲を定義します。オプションは次のとおりです。

   - セッション
     ターゲットオブジェクトはセッションが継続している間、セッション内のすべての XML オペレーションで参照できます。

   - メッセージ
     ターゲットオブジェクトはそれをインスタンス化したクライアント要求の実行中にかぎり、参照できます。1 つの要求に対して 1 つの XML オペレーションだけが実

行されるため、ターゲットオブジェクトへのアクセスはその 1 つの XML オペレーションに制限されます。XML オペレーションが実行されるたびに、新しいターゲットオブジェクトが作成されます。

注 - SOAP 1.1 仕様はセッションをサポートしていません。したがって、範囲を選択しても現在のリリースの Forte for Java 4 IDE で作成した Web サービスの実行特性には影響ありません。

---

4. 必要に応じて「ソース」フィールドを編集します。

   このフィールドでは、ターゲットオブジェクトが取得またはインスタンス化されるメカニズムの種類を指定します。オプションは次のとおりです。

   - **エンタープライズ JavaBean**
     ターゲットオブジェクトは、エンタープライズ Bean のリモートまたはホストインタフェースです。ホームインタフェースは JNDI ルックアップから取得されます。リモートインタフェースは、対応するホームインタフェースへのメソッド呼び出しによって取得されます。

   - **コンストラクタ**
     ターゲットオブジェクトは、ターゲットクラス上のコンストラクタメソッドへの呼び出しによって返されます。

   - **`static` メソッド**
     ターゲットオブジェクトは、`static` メソッドへの呼び出しによって返されます。

5. 必要に応じて残りのフィールドを編集します。

「ソース」フィールドは、ダイアログ内の他のどのフィールドが表示されるか、および入力用に有効になるかに影響します。ターゲットオブジェクトのソースに適している次の表のいずれかの説明を参照してダイアログ内の残りのフィールドを編集します。

ソースがエンタープライズ Bean の場合は、表 B-1 を参照してください。

表 B-1 「ソース」フィールドに「エンタープライズ JavaBean」が設定されている場合に有効になるフィールド

フィールド	説明
EJB 名	EJB モジュール配備記述子内に定義されているエンタープライズ Bean の名前
EJB の型	「エンティティ」または「セッション」のいずれかのエンタープライズ Bean の型。必須。
EJB 参照名	ターゲットオブジェクトを指定する JNDI ルックアップ内の文字列。デフォルト値は、「名前」フィールドの値に、接頭辞として文字列 `ejb/` を付けたものです。必須。
EJB リモートインタフェース	Enterprise Bean のリモートインタフェース。ターゲットオブジェクトがホームインタフェースの場合でも必須。
EJB ホームインタフェース	エンタープライズ Bean のホームインタフェース。必須
メソッド	リモートインタフェースを返すためにホームインタフェースで呼び出される検索または生成メソッド。検索メソッドは、エンティティ Bean で使用され、生成メソッドはセッション Bean で使用される。ターゲットオブジェクトがリモートインタフェースの場合のみ必須

ソースがコンストラクタの場合は、表 B-2 を参照してください。

表 B-2　「ソース」フィールドが「コンストラクタ」に設定されている場合に有効になるフィールド

フィールド	説明
クラス	ターゲットオブジェクトのコンストラクタが定義されているクラス。このフィールドは、「コンストラクタ」フィールド用のコンストラクタを選択したときに自動的に記入される。読み取り専用。
コンストラクタ	ターゲットオブジェクトをインスタンス化するために使用されるコンストラクタメソッド。コンストラクタのクラスは、「クラス」フィールドに指定される。必須

ソースが `static` メソッドの場合は、表 B-3 を参照してください。

表 B-3　「ソース」フィールドが「`static` メソッド」に設定されている場合に有効になるフィールド

フィールド	説明
クラス	ターゲットオブジェクトを返す静的 (Static) メソッドが定義されているクラス。このフィールドは、「静的 (Static) メソッド」フィールドで静的 (Static) メソッドを選択すると自動的に記入されます。読み取り専用。
`static` メソッド	ターゲットオブジェクトをインスタンス化するために使用される静的 (Static) メソッド。必須

6. 必要に応じてメソッドパラメータをソースにマップします。

   「パラメータのマップ」ボタンが有効になっている場合は、手順に従って「メソッド」、「コンストラクタ」、または「静的 (Static) メソッド」フィールドに指定されているメソッドのパラメータに値を指定します。

   a. 「パラメータのマップ」をクリックします。

      「パラメータのマップ」ダイアログが表示されます。

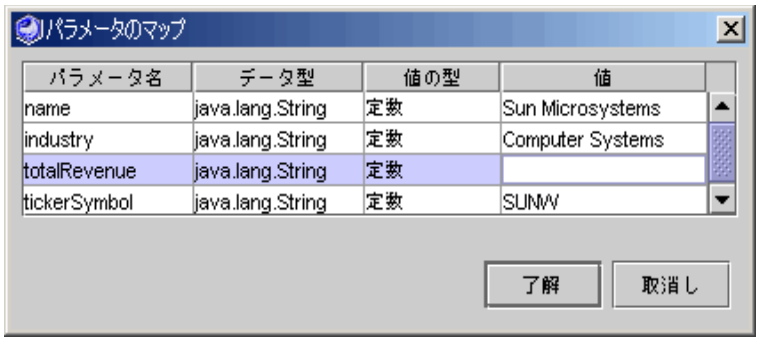

図 B-3　「パラメータのマップ」ダイアログ

b. マッピングするパラメータを表す行を指定します。

各行は、1 つのメソッドパラメータを表します。メソッドで複数のパラメータを指定できる場合は、「パラメータ名」列を使用してパラメータを識別します。メソッドがソースコードとして提供されている場合、この列はパラメータの名前を示します。メソッドが JAR ファイルで提供されている場合、「パラメータ名」列には、メソッドが必要とするパラメータの順序を param1、param2、などの名前を表示することで示します。

c. パラメータの値の型を指定します。

「値の型」列をクリックし、リストからオプションを選択します。次の表は、オプションの説明です。

オプション	説明
定数	パラメータを「値」フィールドに指定されている値にマップする

オプション	説明
環境エントリ	パラメータを「値」フィールドに指定されている環境エントリの値にマップする。環境エントリの設定に関して詳細は、25 ページの「環境エントリの追加」を参照してください。
ターゲットオブジェクト	パラメータを「値」フィールドに指定されているオブジェクトにマップします。オブジェクトは、Web サービスの「オブジェクト参照を解決」ダイアログ内で定義されているターゲットオブジェクト
入力ドキュメント要素	パラメータを「値」フィールドに指定されている入力ドキュメント要素にマップする。入力ドキュメント要素のデータ型はオブジェクト (たとえば、`String`) でなければならない。基本型 (たとえば、`int`) は使用できない。入力ドキュメント要素のデータ型の宣言については、103 ページの「入力ドキュメント要素の追加」を参照してください。

d. 「値」フィールドにパラメータの値を指定します。

次の表は、パラメータの値の型に依存して値を指定する方法を説明しています。

値の型	ユーザー操作
定数	「値」フィールドにリテラル値を入力します
環境エントリ	「値」フィールドに環境エントリの名前を入力します。 環境エントリの設定に関して詳細は、25 ページの「環境エントリの追加」を参照してください
ターゲットオブジェクト	「値」フィールドをクリックし、リストからオブジェクトを選択します。 このリストには、Web サービスに定義されている「データ型」フィールド内に指定されているクラスのすべてのターゲットオブジェクトが含まれます。
入力ドキュメント要素	「値」フィールドをクリックし、リストから入力ドキュメント要素を選択します。 このリストには、次の両方の入力ドキュメント要素が含まれます。 • 指定したデータ型 • ターゲットオブジェクトを参照する各 XML オペレーションに定義済み 2 番目の要件により、マッピングしているパラメータがすべての状況において値に対するソースを持っていることが保証されます。たとえば、2 つの XML オペレーションは同じターゲットオブジェクトを参照できますが、どちらの XML オペレーションが先に実行されるかはわかりません。したがって、それぞれの XML オペレーションでパラメータの値を提供して、どちらが先に実行されてもパラメータ値が利用できるようにする必要があります。入力ドキュメント要素についての詳細は、92 ページの「入力ドキュメント要素ノード」を参照してください

付録C

配備記述子

Web サービス用の実行時クラスを生成すると、多層または Web 主体アーキテクチャのどちらを選択したかによって Web モジュールと EJB モジュール配備記述子も生成されます。Web サービス J2EE アプリケーションを組み立てると、これらの配備記述子がアプリケーションに含まれます。配備記述子は、アプリケーションの実行時プロパティを設定するために使用される XML ファイルです。J2EE アプリケーションはこれらの記述子のフォーマットを定義します。

配備記述子は、開発中はいつでもソースエディタ内で参照できます。配備記述子を編集することもできます。ただし、配備記述子を編集すると、実行時クラスを生成し直した場合には再生成されません。配備記述子の編集内容は保持されます。配備記述子の編集後に Web サービスに対して行なった変更は、配備記述子には伝達されません。したがって、配備記述子の編集は、開発サイクルの最後にだけ行なってください。

配備記述子に伝達されるフィールド

ここでは、標準で Web サービスの配備記述子に伝達される IDE 内のフィールドを示します。これらの配備記述子の 1 つを編集した後に、標準で配備記述子に伝達されるフィールドの値を変更した場合は、配備記述子を手動で編集して新しい値を入力する必要があります。

EJB モジュール配備記述子に伝達されるフィールド

「オブジェクトの解決」ダイアログには、EJB モジュール配備記述子に伝達されるいくつかのフィールドが含まれます。配備記述子内の対応する要素は、すべて `ejb-ref` 要素のサブ要素です。次の表は、フィールドの一覧です。

「オブジェクトの解決」ダイアログ内のフィールド	EJB モジュール配備記述子内の要素
EJB 参照名	`description`
EJB 参照名	`ejb-ref-name`
EJB の型	`ejb-ref-type`
EJB ホームインタフェース	`home`
EJB リモートインタフェース	`remote`
EJB 名	`ejb-link`

注 - EJB メソッド呼び出しを XML コンポーネントの 1 つに追加すると、新しいターゲットオブジェクト定義が作成されます。メソッド呼び出しを特定の EJB コンポーネントに初めて追加するときに、「オブジェクトの解決」ダイアログに新しいターゲットオブジェクト定義が作成されます。一般的に、この定義は EJB モジュール配備記述子に新しい `ejb-ref` 要素として伝達されます。

Web サービスのプロパティシートには、「環境エントリ」という名前のプロパティがあります。このプロパティ内のフィールドは、EJB モジュール配備記述子に伝達されます。配備記述子内の対応する要素は、すべて `env-entry` 要素のサブ要素です。次の表は、フィールドの一覧です。

「環境エントリ」プロパティ内のフィールド	EJB モジュール配備記述子内の要素
名前	`env-entry-name`
説明	`description`
型	`env-entry-type`
値	`env-entry-value`

Web モジュール配備記述子に伝達されるフィールド

Web サービスにデフォルトのテストクライアントを設定する場合、クライアントの開始ページおよびエラーページは Web モジュールの配備記述子に伝達されます。次の表は、フィールドの一覧です。

プロパティ	Web モジュール配備記述子内の要素
エラーページ	`location` (`error-page` のサブ要素)
開始ページ	`welcome-file` (`welcome-file-list` のサブ要素)

デフォルトのテストクライアントの詳細については、35 ページの「Web サービス用のデフォルトテストクライアントの設定」および 65 ページの「クライアント HTML ページおよび JSP ページ」を参照してください。

配備記述子を参照する

Web モジュールまたは EJB モジュール配備記述子を表示する手順は、次のとおりです。

- **Web サービスノードをマウスの右ボタンでクリックし、これらのメニュー項目の 1 つを選択します。**
  - 「配備記述子」>「Web モジュール」>「表示」
  - 「配備記述子」>「EJB モジュール」>「表示」

配備記述子は、ソースエディタ内で読み取り専用モードで表示されます。

配備記述子を編集する

Web モジュールまたは EJB モジュール配備記述子を編集する手順は、次のとおりです。

1. Web サービスノードをマウスの右ボタンでクリックし、これらのメニュー項目の 1 つを選択します。

- 「配備記述子」>「Web モジュール」>「最終的な編集」
- 「配備記述子」>「EJB モジュール」>「最終的な編集」

図 C-1 に示されている「最終的な編集」ダイアログが開きます。このダイアログは配備記述子を編集した後に、Web サービスの実行クラスを生成し直したときに IDE によって記述子が再生成されないことを示します。

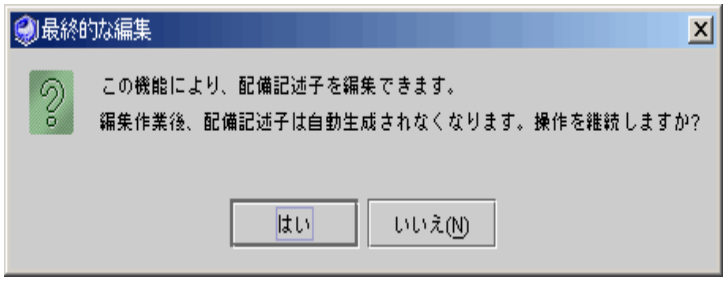

図 C-1　「配備記述子の最終的な編集」ダイアログが開きます。

2. 配備記述子を生成し直す必要がない場合は、「はい」をクリックします。それ以外の場合は、「いいえ」をクリックします。

   配備記述子は、ソースエディタ内で編集モードで表示されます。Web サービス用の実行時クラスを生成し直す場合は、配備記述子は再生成されません。

索引

W

X

え

お

か

く

け

さ

し

た

て

に