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

Part No. 816-2844-01
2001 年 10 月 , Revision A

原典： *Building Web Services*
Part No: 816-1400-10
Revision A

目次

はじめに　ix

1. Web サービス - 概要　1

Web サービスの利点　1

Web サービスインタフェース　2

Web サービスのアーキテクチャ　4

要求応答メカニズム　4

HTTP クライアントのアクセス　7

統合クライアントのアクセス　8

2. Web サービスの構築　9

Web サービス開発ワークフロー　9

Web サービスの作成　10

XML オペレーション、ファイル、およびライブラリの追加　11

Web サービスへの参照の追加　14

Web サービスからの参照の削除　16

オブジェクトのインスタンス化と参照の解決　17

オブジェクト参照のターゲットを指定する　18

新規ターゲットオブジェクトを定義する　20

ターゲットオブジェクト定義を編集する　20
環境エントリの追加　27
Web サービスのセキュリティ保護　29
認証の設定　30
ドキュメントへのアクセスの制限　32
XML オペレーションへのアクセスの制限　34
実行時クラスとクライアントページ　35
実行時クラスの生成　37
クライアントページの生成の制御　37
開始ページとエラーページの設定　39
ESP レジストリ内での公開　40
配備記述子　43
配備記述子に伝達されるフィールド　43
配備記述子を参照する　46
配備記述子を編集する　46
3. XML オペレーションの開発　47
ツールの概要　47
「データソース」区画　48
入力ドキュメント要素ノード　49
メソッドノード　49
開発ワークフロー　53
XML オペレーションの作成　54
XML オペレーションの作成　54
EJB からの XML オペレーションの生成　57
XML オペレーションの編集　58
XML オペレーションへのメソッドの追加　58
入力ドキュメント要素の追加　60

入力ドキュメント要素の名前変更　61
出力ドキュメント要素の名前変更　62
入力ドキュメント要素にデフォルト値を指定する　62
入力ドキュメント要素を常時に設定する　63
メソッドまたは入力ドキュメント要素の並び替え　65
メソッドまたは入力ドキュメント要素の削除　65
メソッドパラメータのソースへのマッピング　66
XML 出力ドキュメントから要素を除外する　69
XML 出力ドキュメントに要素を含める　69
クラスを展開する　70
クラスを縮小する　70
キャッシングおよびナビゲーションのサポート　71
オブジェクトの共有　81
Static ユーティリティメソッド　83

4. Web サービス開発シナリオ　87

単一開発者シナリオ　87
背景　87
開発フロー　88
開発チームシナリオ　91
背景　92
開発フロー　92

A. Web サービスの組み立て、配備およびテスト　97

Web サービスの準備ができていることを確認する　98
RI サーバーの構成と起動　98
J2EE アプリケーションの組み立て　100
Web サービスの配備　103

Web サービスのテスト　103

統合プレゼンテーションサーバーを使用したテスト　105

HTTP クライアントを使用したテスト　106

B. Web サービスアプリケーションのアーキテクチャ　107

B2B クライアントによってアクセスされる Web サービス　107

プレゼンテーションサーバーを使用した Web サービス　110

索引　113

図目次

図 1-1　バックエンドサービスへの XML ベースのインタフェース　3

図 1-2　Web サービスの J2EE 実行時環境　4

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　6

図 1-4　要求データを渡す 2 つの方法　7

図 1-5　Web サービスと同じ Web コンテナ内で実行する統合プレゼンテーションサーバー　8

図 2-1　Web サービス内のファイル、XML オペレーション、およびフォルダへのリファレンスの階層配列　13

図 2-2　「オブジェクト参照を解決」ダイアログ　18

図 3-1　ソースエディタで開かれた XML オペレーション　48

図 3-2　入力ドキュメント要素ノード　49

図 3-3　論理 EJB ノード　56

図 B-1　B2B クライアント要求に応答する Web サービス　108

図 B-2　統合 ESP プレゼンテーションサーバーを使用した Web サービス　110

はじめに

Forte for Java プログラミングシリーズの『Web サービスのプログラミング』へようこそ。このマニュアルでは、Forte™ for Java™ Web Services IDE モジュールを使用して Web サービスを構築し、実装する方法を説明しています。

対象読者

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。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。

また、JavaServer Pages™ (JSP) テクノロジを使用して Web サービス用のプレゼンテーションを構築する開発者に役立つ情報も記載しています。Web Services モジュールを使用して構築された Web サービスによって提供されるキャッシュナとナビゲーションサポートを JSP 開発者が使用する方法について説明しています。

このマニュアルを理解するには、以下の知識が必要になります。

- Java プログラミング言語
- Java 2 Platform、Enterprise Edition の概念
- XML および HTML 構文
- JavaServer ページ構文

お読みになる前に

J2EE の基本的な概念を理解するために役立つ資料を、以下に示します。

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

マルチプラットフォーム リリース

このマニュアルで説明されるタスクは、次のプラットフォームおよびオペレーティングシステムで実行できます。

- Solaris™ 8 SPARC™ プラットホーム版
- Microsoft Windows 2000 SP2
- Microsoft Windows NT 4.0, SP6
- Red Hat Linux 6.2

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

```
C:\>cd 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
```

表記上の規則

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

表 P-1　表記上の規則

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

コード例は次のように表示されます。

- C シェルプロンプト

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

■ Bourne シェルおよび Korn シェルのプロンプト

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

■ スーパーユーザーのプロンプト

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

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

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

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

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

内容の紹介

第 1 章では、Forte for Java 統合開発環境 (IDE) で Web Services モジュールを使用して構築した Web サービスの概要を説明します。

第 2 章では、Web サービスを開発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、Web サービスのテストに使用できるクライアントページを生成する方法も説明します。

第 3 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。

第 4 章では、Web サービス開発シナリオのいくつかを示します。

付録 A では、Web サービスを J2EE アプリケーションとして組み立て、アプリケーションを J2EE リファレンス実装 (RI) サーバーに配備し、クライアントを使用してテストする方法を説明します。

付録 B では、Web Services IDE モジュールを使用した Web サービスの構築の実行時アーキテクチャの詳細を説明します。

関連マニュアル

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

オンラインで入手可能なマニュアル

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

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

- リリースノート (HTML 形式)

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

- インストールガイド (PDF 形式)

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

- Forte for Java プログラミングシリーズ (PDF 形式)

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

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

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

- 『持続プログラミング』 - Part No. 816-2850-01

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

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

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

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

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

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

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

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

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

- Forte for Java チュートリアル (PDF 形式)

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

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

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

- 『Forte for Java, Enterprise Edition チュートリアル』 - Part No. 816-2848-01

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

オンラインヘルプ

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

プログラム例

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

Javadoc

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

ご意見の送付先

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

docfeedback@sun.com

電子メールのタイトルに、対象マニュアルの Part No.（このマニュアルの場合は 816-2844-01）を明記してください。

第1章

Web サービス - 概要

この章では、Forte™ for Java™ 統合開発環境 (IDE) で Web Services モジュールを使用して構築した Web サービスの概要を説明します。

Forte for Java Web Services モジュールは、World Wide Web で、バックエンドビジネスサービスを Web サービスとして公開することができる強力なテクノロジを提供します。Web サービスは、アプリケーションのビジネスコンポーネントを標準のインターネットプロトコルやメッセージ形式で提供するプログラム方式です。HTTP を介して XML ドキュメントを渡すことで通信することのできるクライアントは、基盤となるビジネスサービスの実装方法に関係なく、Web サービスにアクセスできます。

Web サービスの利点

ビジネスコンポーネントを Forte for Java Web サービスとして公開すると、次のような利点があります。

- **役割に基づいた開発が行えます。**Web サービスは、プレゼンテーションから切り離されたビジネスロジックです。Java 開発者および Web デザイナは、独立して作業することができます。Web デザイナは、プログラミングツールを使用する必要はなく、自分でツールを選択できます。
- **B2B アプリケーションとのインタフェースが提供されます。**Web サービスは、標準のプロトコルおよびメッセージ形式に基づいてインタフェースを提供します。
- **アプリケーションを統合できます。**多種多様なビジネスコンポーネントを Web サービスに組合わせることで、複雑なアプリケーションを作成できます。

- **コンポーネントが再利用できます。**ビジネスコンポーネントは、複数の Web サービスで再利用することができ、多くのクライアントからアクセスできます。
- **柔軟性が提供されます。**ビジネスコンポーネントおよびクライアントの実装は、それぞれの XML インタフェースが維持される限り独立して変更できます。XML インタフェース自体は柔軟性があります。クライアントを変更しなくても、XML 要素を追加または削除することができます。
- **プレゼンテーションコンポーネントを生成できます。**Web サービスから JavaServer Pages™ ファイルを自動的に生成できます。Web デザイナは、これらのページをカスタマイズして HTML クライアント用のプレゼンテーションを簡単に作成できます。
- **スケーラブルで拡張可能な、セキュリティ保護された OS 非依存プラットフォームが提供されます。**Web サービスは Java 2 Platform、Enterprise Edition (J2EE™) アプリケーションとして実装されます。J2EE プラットフォームの利点 - スケーラビリティ、拡張性、セキュリティ、信頼性、およびオペレーティングシステムへの非依存性が得られます。Web サービスは、任意の J2EE アプリケーションサーバーに配備できます。

Web サービスインタフェース

Web サービスは、XML ドキュメントを交換することでクライアントと通信します。XML 標準が使用されることは、データソースの基礎となるテクノロジの知識を持たなくても、クライアントがデータを Web サービスから入手できることを意味しています。Web サービス公開データソースに接続しているクライアントは、要求を実行し、XML ドキュメントとしてフォーマットされた応答データを受け取りますが、その基礎となるプロトコル、コンポーネントモデル、API、およびオペレーティングシステムは認識しません (図 1-1 を参照)。

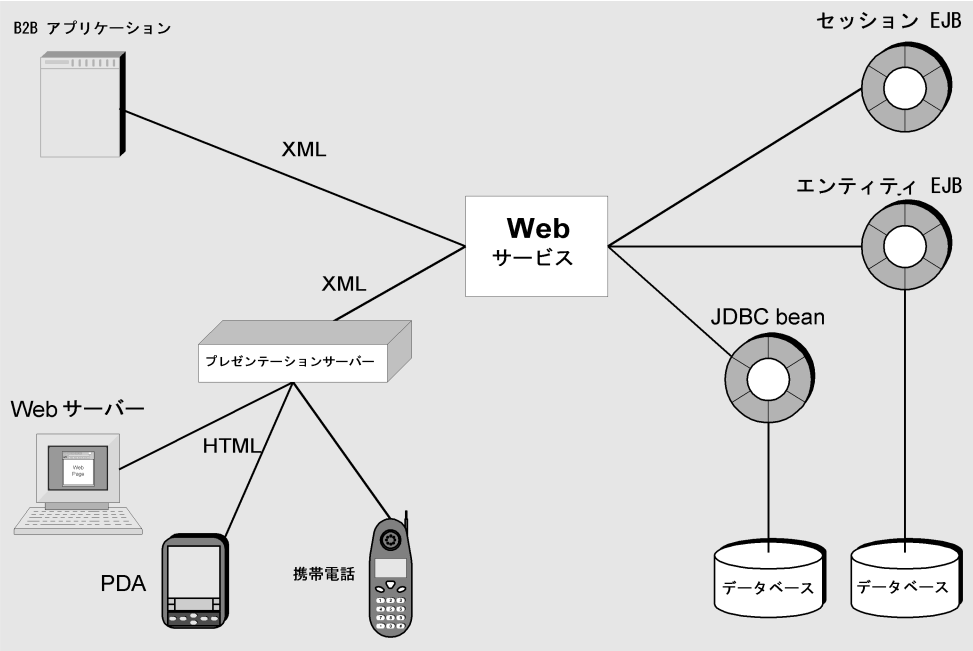


図 1-1　バックエンドサービスへの XML ベースのインタフェース

Web サービスの一般的なクライアントは、次の通りです。

- 企業間 (B2B) クライアント
- その他の Web サービス
- プレゼンテーションサーバー

Web ブラウザ、携帯電話、PDA などのその他のクライアントは、プレゼンテーションサーバーを経由して Web サーバーにアクセスします (図 1-1 を参照)。プレゼンテーションサーバーは、XML ドキュメントをクライアントのタイプに適した HTML や WML ドキュメントに変換する Web アプリケーションです。

Web サービスクライアントは、XML ドキュメントを介してサービスを要求し、応答として XML ドキュメントを受け取ります。内部的には、Web サービスはこれらの要求 XML ドキュメントをアプリケーションビジネスのコンポーネント上でメソッド呼び出しに変換します。ビジネスコンポーネントは、要求に応じて機能を実行し、Web サービスはメソッド戻り値を応答 XML ドキュメントに変換します。

この XML 変換メカニズムを実装するために、手動でコーディングする必要はありません。Web Services モジュールは、ビジネスコンポーネントからこれらの変換を実行する実行時クラスを生成します。既存のビジネスコンポーネントから XML や Java のコードを手動でコーディングしなくても、Web サービス全体を作成できます。

Web サービスのアーキテクチャ

Forte for Java Web Services モジュールを使用して開発した Web サービスは、J2EE アプリケーションとして実装されます。Web サービスを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。Web サービスは実行時に Web と J2EE アプリケーションサーバーの EJB コンテナをスパン化する論理エントリです (図 1-2)。

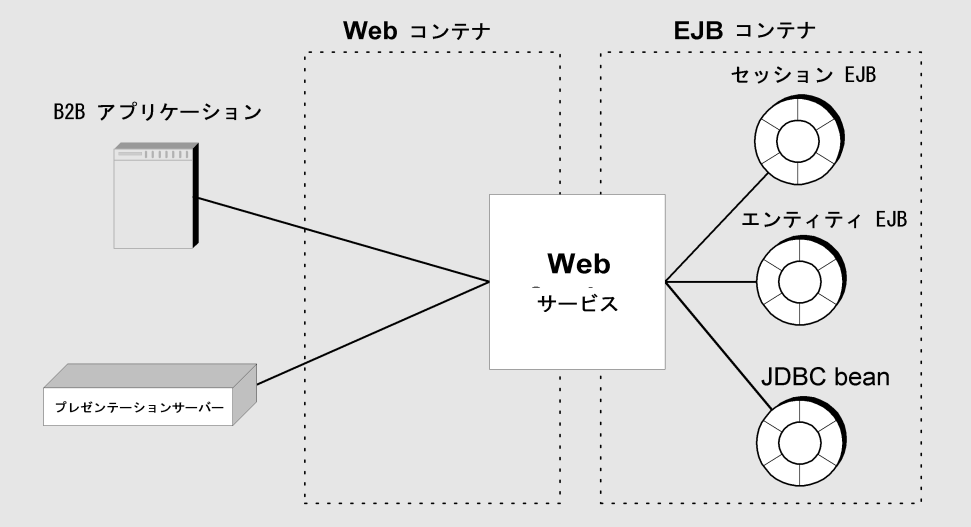


図 1-2　Web サービスの J2EE 実行時環境

要求応答メカニズム

Web サービスは、XML オペレーションと呼ばれるいくつかの論理エントリで構成されています。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ます。XML オペレーションは、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します。
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します。
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します。
4. XML 出力ドキュメントを返します。

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。

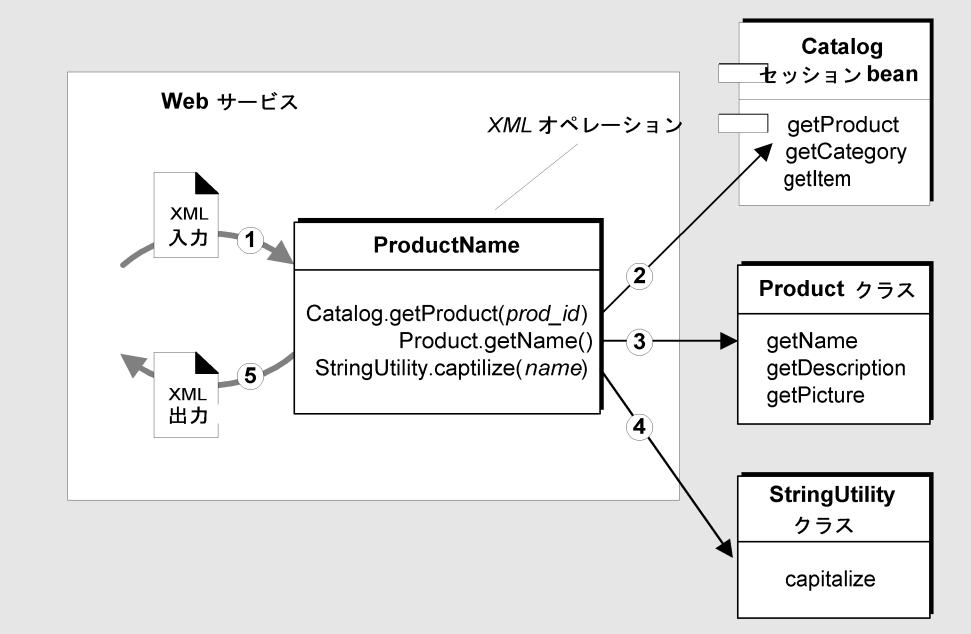


図 1-3　クライアント要求を満たすために複数のメソッドを呼び出す XML オペレーション

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。実行時には次のことが行われます。

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します。
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します。
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 オブジェクトを返します。
4. Static (静的) メソッド StringUtility.capitalize を呼び出す、製品名をパラメータとして含む String オブジェクトを渡します。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 オブジェクトを返します。
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、返します。

HTTP クライアントのアクセス

HTTP アクセスは、Java サーブレットによって Web サービスに提供されます。クライアントは、Web サービスサーブレットに次の 2 つのどちらかの方法で要求データを渡します (図 1-4 を参照)。

- HTTP ポストメソッドを使用して要求データを指定する XML データを渡す
- Web サービスへのアクセスに使用される URL 内に照会パラメータとして要求データを入れる

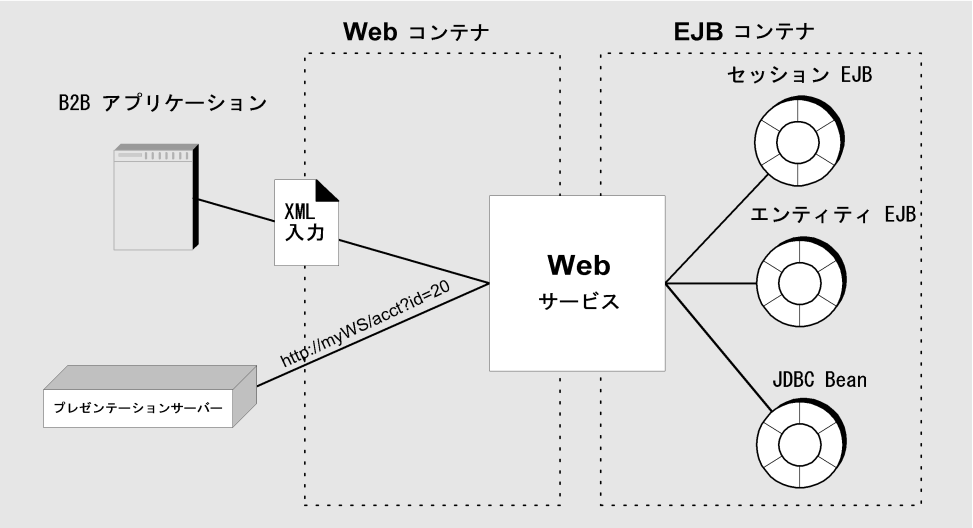


図 1-4　要求データを渡す 2 つの方法

Web サービスが XML ドキュメントを受け取ると、適切な XML オペレーションを決定するために、解析して要求を処理してから転送します。URL 内に埋め込まれている Web サービスが要求データを受け取ると、URL 照会パラメータを XML ドキュメントに変換する、追加された最初の手順を実行します。

Web サービスサーブレットを介した HTTP アクセスは、Web サービスに対してリモートアクセスを必要とする次のクライアントに適しています。

- B2B クライアント
- その他の Web サービス
- プレゼンテーションサーバー

HTTP を Web サービスとプレゼンテーションサーバー間の通信プロトコルとして使用することは可能ですが、最適な方法ではありません。Web サービスは、複数の HTTP 接続間でプレゼンテーションサーバーのクライアントのセッション状態を維持できません。

統合クライアントのアクセス

Web Services モジュールは、プレゼンテーションサーバーが Web サービスの統合クライアントであるモデルをサポートします。このモデルでは、プレゼンテーションサーバーを Web サービスおよびビジネスコンポーネントを含んでいる同じ J2EE アプリケーションにパッケージ化します。プレゼンテーションサーバーは Web サービスと同じ Web コンテナ内で実行し、Web Services モジュールとともに提供される JSP タグライブラリによって Web サービスと通信します (図 1-5)。

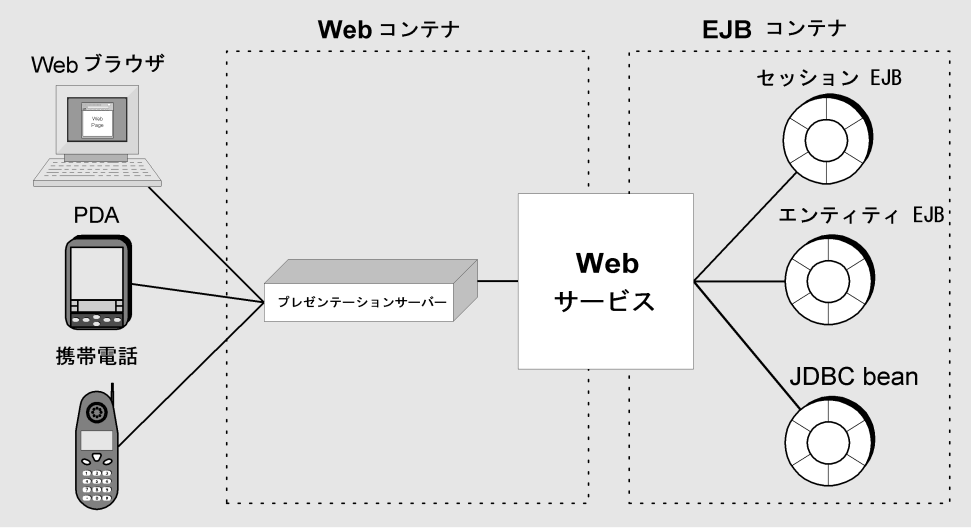


図 1-5　Web サービスと同じ Web コンテナ内で実行する統合プレゼンテーションサーバー

このモデルには、主に 3 つの利点があります。

- Web サービスは、プレゼンテーションサーバーのクライアントのセッション状態を維持できます。
- Web サービスは、JSP セキュリティメカニズムを使用してセキュリティを実装できます。
- プレゼンテーションサーバーのクライアントの応答時間を最適化できます。

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するツールについて説明します。また、Web サービスにアクセスするために使用する JSP ページの生成方法についても説明します。

Web サービスを J2EE アプリケーションとしてアセンブルし、アプリケーションを J2EE リファレンス実装サーバーに配備し、クライアントでテストする方法については、付録 A を参照してください。Web サービスを iPlanet™ アプリケーションサーバー (iAS) に配備する方法についての詳細は、リリースノートを参照してください。

Web サービス開発ワークフロー

次の手順は、統合プレゼンテーションサーバーを使用して Web サービスを開発するワークフローの概要を示しています。

1. Web サービスを作成します。

   このタスクについての詳細は、10 ページの「Web サービスの作成」を参照してください。

2. Web サービスで必要な XML オペレーションを開発します。

   XML オペレーションを開発する方法についての詳細は、第 3 章を参照してください。

3. XML オペレーションを Web サービスに追加します。

   このタスクについての詳細は、11 ページの「XML オペレーション、ファイル、およびライブラリの追加」を参照してください。

4. プレゼンテーションサーバーによって使用されるファイル、たとえば、JSP、HTML、GIF、JAR ファイルなどを追加します。

   このタスクについての詳細は、11 ページの「XML オペレーション、ファイル、およびライブラリの追加」を参照してください。

5. 必要に応じて、実行時オブジェクトに参照を解決します。

   このタスクについての詳細は、17 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。

6. (オプション) セキュリティパラメータを定義します。

   このタスクについての詳細は、29 ページの「Web サービスのセキュリティ保護」を参照してください。

7. Web サービスの実行時クラスを生成します。

   このトピックについての詳細は、35 ページの「実行時クラスとクライアントページ」を参照してください。

8. (オプション) Web サービスを ESP レジストリに公開します。

   このトピックについての詳細は、40 ページの「ESP レジストリ内での公開」を参照してください。

9. Web サービスを J2EE アプリケーションとしてアセンブルして配備します。

   このトピックについての詳細は、付録 A を参照してください。

10. Web サービスをテストします。

    このトピックについての詳細は、付録 A を参照してください。

11. Web サービスを編集して、実行時クラスを再生成し、条件を満たすまでテストします。

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. 「テンプレートウィザードから新規作成」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フォルダをマウスの右ボタンでクリックし、「新規」>「Web Services」>「Web Service」を選択します。

2. 「名前」フィールドで、Web サービスの名前を入力します。

3. 「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。

   パッケージが指定されていない場合は、ブラウズボタンをクリックしてパッケージを選択します。

4. 「完了」をクリックします。

   新しい Web サービスがエクスプローラに表示されます。

   これで、XML オペレーション、ドキュメント、またはライブラリを Web サービスに追加できるようになります。詳細は、11 ページの「XML オペレーション、ファイル、およびライブラリの追加」を参照してください。XML オペレーションの作成方法については、第 3 章を参照してください。

XML オペレーション、ファイル、およびライブラリの追加

次に対する参照を追加して、Web サービスを構築します。

- XML オペレーション
- Web コンテナで要求される JAR アーカイブファイル
- JSP ページ、HTML ファイル、およびイメージファイルなどのドキュメントファイル
- ドキュメントファイルを含むフォルダ

このような参照を Web サービスに追加すると、種類別にソートされて、Web サービスノード内のフォルダに配置されます。

- XML オペレーションフォルダ内の XML オペレーション
- ドキュメントファイルフォルダ内にドキュメントファイルを含むドキュメントファイルおよびフォルダ
- ライブラリフォルダ内の JAR アーカイブファイル

参照される XML オペレーション、ファイル、またはフォルダに従い、各リファレンスに名前をが付けられます。これらの名前の前には、参照される XML オペレーション、ファイル、またはフォルダのパッケージが付きます。参照の前には -> シンボルが付加され、簡単に識別できるようになっています。

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。XML オペレーション、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ます。参照された XML オペレーション、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。参照されるフォルダ内のノードは、実際のファイルまたはフォルダを削除する場合以外は削除しないでください。

図 2-1 は、次の対象への参照を含むサンプル Web サービスのエクスプローラでの表示を示しています。

- XML オペレーション
- JAR ファイル
- HTML ファイル
- JSP ページおよび GIF イメージファイルのサブフォルダを含むフォルダ

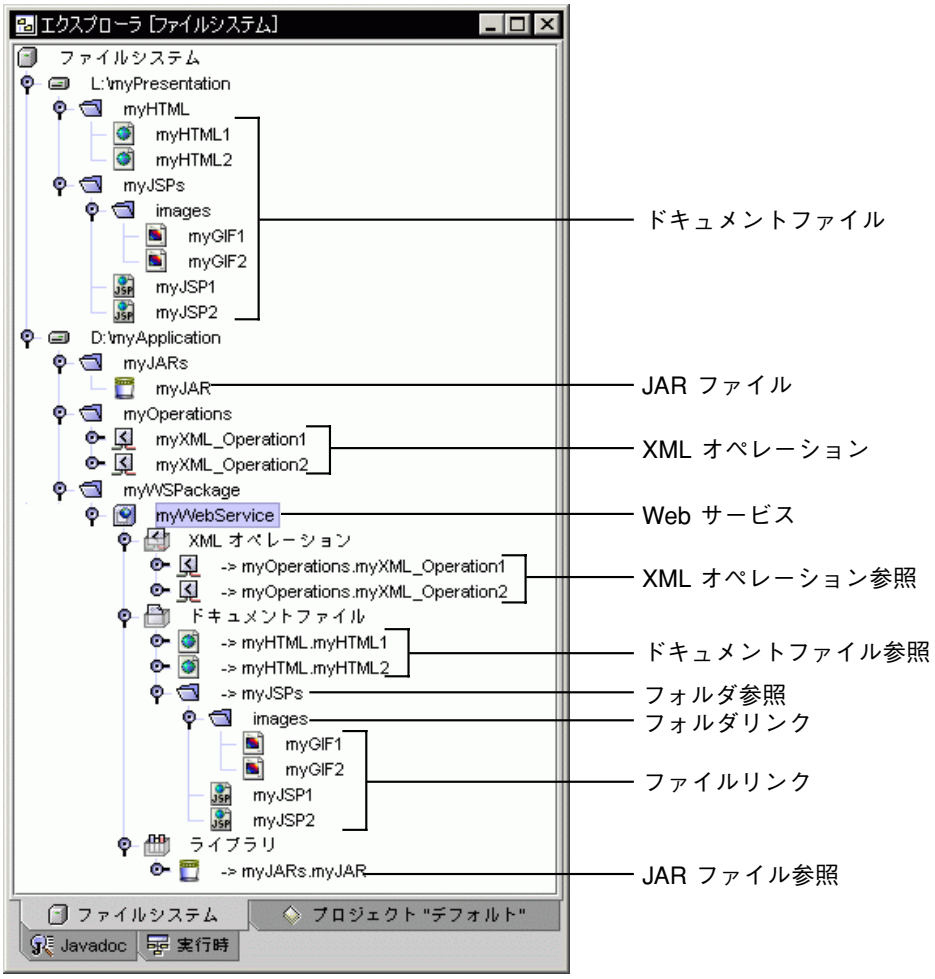


図 2-1　Web サービス内のファイル、XML オペレーション、およびフォルダへのリファレンスの階層配列

図 2-1 では、マウントされたファイルシステム `L:\myPresentation` は、Web サービスによって参照されるドキュメントファイルを含んでいます。このファイルシステムは、Web ページデザイナが HTML ファイルおよび JSP ページを開発しているリモートマシン上に配置することができます。

その他のマウントされたファイルシステム (`D:\myApplication`) は、Web サービス開発者の作業領域です。ここには XML オペレーション、JAR ファイル、および Web サービスが含まれています。

Web サービスを組み立てるときに、参照されるすべての XML オペレーション、JAR、およびドキュメントファイルは自動的に Web サービスアプリケーションに含まれます。参照されるフォルダに含まれるドキュメントファイルは、Web サービスアプリケーションにも含まれますが、参照されるフォルダ自体は含まれません。

直接参照されるドキュメントファイルと参照されるフォルダに含まれるドキュメントファイルは、組み立てられたアプリケーション内の同じ論理場所に配置されます。たとえば、図 2-1 では、myHTML.myHTML1 (参照) と myJSP1 (参照されるフォルダ内にある) は、両方ともアセンブルされた Web アプリケーションのドキュメントルートに配置されます。次の例は、myJSP1 をポイントする myHTML1 内の HTML ハイパーリンクコードを示しています。

```
<a href="/myJSP1.jsp">some text</a>
```

次の例は、myHTML1 をポイントする myJSP1 内のハイパーリンクコードを示しています。

```
<a href="/myHTML1.html">some text</a>
```

次の例は、myGIF1 をポイントするハイパーリンクを示しています。

```
<a href="/images/myGIF1.gif">some text</a>
```

前の例では、イメージフォルダは参照されるフォルダ (myJSPs) 内にあるため、URL 内に指定されます。直接は参照されません。

Web サービスへの参照の追加

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、「参照の追加」を選択します。

   ファイルブラウザが開きます。

2. 参照を追加する XML オペレーション、ファイル、およびフォルダを選択します。

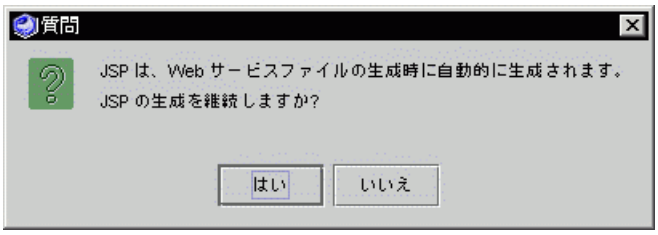
   複数の項目を選択するには、Ctrl キーを押しながら項目をクリックします。

3. 「了解」をクリックします。

   参照が作成され、Web サービスのサブノードであるフォルダ内に配置されます。

   - XML オペレーションは、XML オペレーションフォルダに配置されます。
   - ファイル (JSP、HTML、GIF など) は、ドキュメントファイルフォルダに配置されます。
   - JAR アーカイブファイルは、ライブラリフォルダ内に配置されます。

4. 追加した参照が JSP ページまたは HTML ファイルなどのドキュメントへの参照の場合、次の「質問」ダイアログが開きます。



このダイアログが開いた場合は、Web サービス用の実行時クラスを生成すると、クライアントページが生成されるように Web サービスが構成されていることを意味しています。これはデフォルトの構成です。実行時クラスを生成したときにクライアントページの作成を続行する場合は、「はい」をクリックします。生成されたページは、同じ名前を持つ既存のページを上書きします。

実行時クラスを生成したときにクライアントページを生成しない場合は、「いいえ」をクリックします。Web サービスでカスタムクライアントページを使用している場合には、この選択が適しています。たとえば、以前に生成されたクライアントページがあり、それをカスタマイズする場合や、最初から作成したクライアントページがある場合などです。

このダイアログでは、Web サービスの「クライアントの生成」プロパティを設定します。「はい」をクリックすると、プロパティが「Yes」に設定され、「いいえ」をクリックするとプロパティが「No」に設定されます。このプロパティは、Web サービスのプロパティシートに手動で設定できます。

Web サービス用の実行時クラスの生成についての詳細は、35 ページの「実行時クラスとクライアントページ」を参照してください。

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、XML オペレーションを選択して「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。

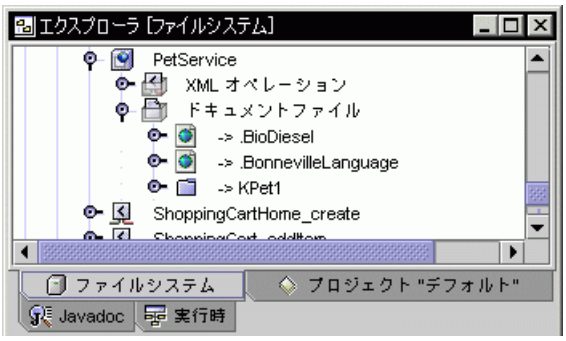
Web サービスからの参照の削除

Web サービスから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. 選択した項目が実際に参照であり、実際の XML オペレーション、JAR ファイルまたはフォルダでないことを確認します。

   次の図で示されているように、参照の名前の前に -> 記号が付きます。



3. 参照をマウスの右ボタンでクリックして「削除」を選択します。

   参照が削除されます。

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、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 EJB 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、次を行う方法を説明する手順を示します。

- 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」ダイアログが開きます。

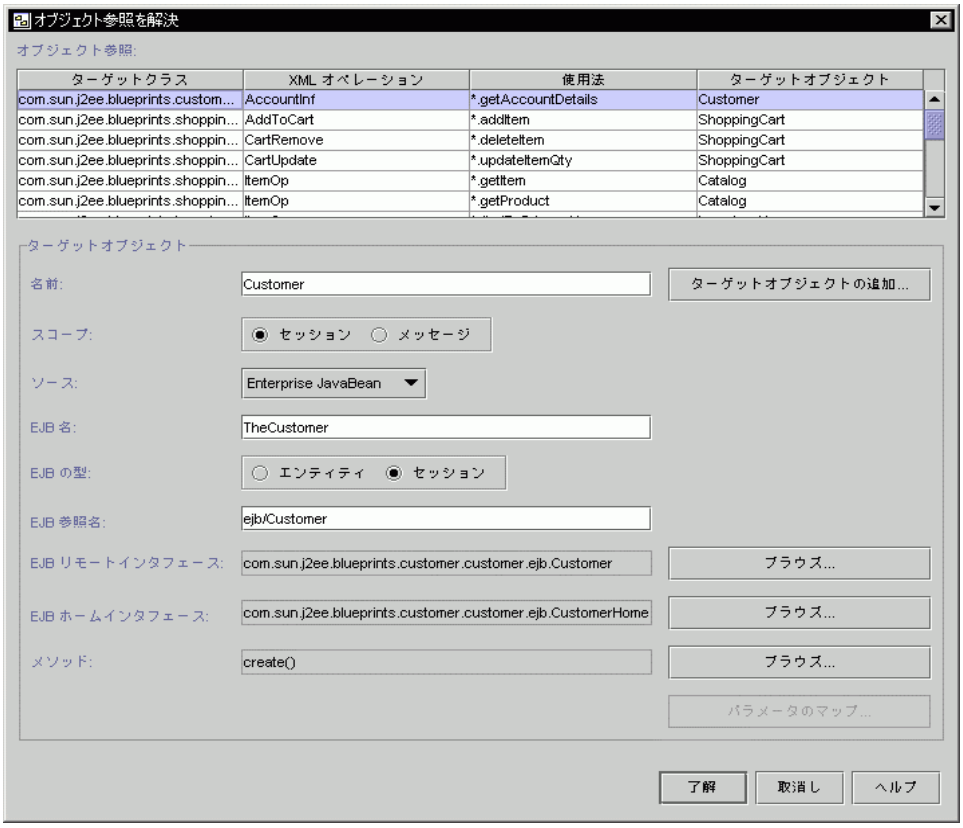


図 2-2　「オブジェクト参照を解決」ダイアログ

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ダイアログの上部に表形式で一覧表示されています。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します。2 つの使用法があり、 XML オペレーションは、次を実行します。 • 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します (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します。 `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します。
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、次のものを含んでいます。

- このダイアログ内ですでに定義されているターゲットオブジェクト
- 共有オブジェクト

リスト内の共有オブジェクトは、XML オプションの名前が接頭辞として付いています。共有オブジェクトについての詳細は、81 ページの「オブジェクトの共有」を参照してください。

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」ダイアログが開きます (図 2-2 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」ダイアログが開きます。

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。新規ターゲットオブジェクトでは、以前と同じ定義が使用されており、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、20 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 サービスが次のこと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ことが必要です。新規定義の作成手順については、20 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」ダイアログが開きます (図 2-2 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。こ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 オペレーションのみが実行されるため、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。

   このフィールドは必須です。

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。

   - Enterprise JavaBean。ターゲットオブジェクトは、EJB 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 ルックアップを通して取得されます。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を通して取得されます。

- **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。
- **静的 (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することによってダイアログ内の残りのフィールドを編集します。ソースによって、次の表を参照してください。

- Enterprise JavaBean の場合は、表 2-1 を参照。
- コンストラクタの場合は、表 2-2 を参照。
- 静的 (Static) メソッドの場合は、表 2-3 を参照。

表 2-1　「ソース」フィールドが「Enterprise JavaBean」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
EJB 名	EJB モジュール配備記述子内に定義されている EJB 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 EJB 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	EJB 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	EJB 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 で使用されます。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。

表 2-2　「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ます。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ます。必須。

表 2-3　「ソース」フィールドが「静的 (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

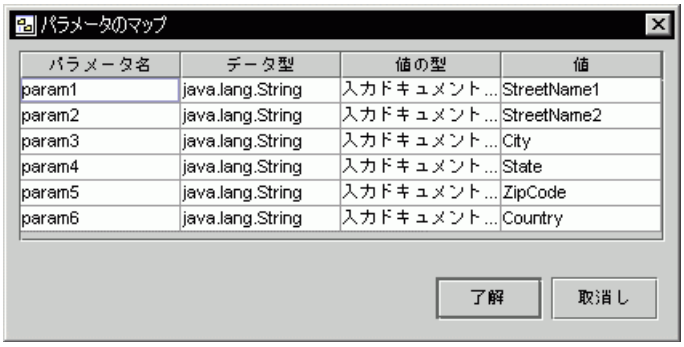
フィールド	説明
クラス	ターゲットオブジェクトを返す静的 (Static) メソッドが定義されているクラス。このフィールドは、「静的 (Static) メソッド」フィールドで静的 (Static) メソッドを選択すると自動的に記入されます。読み取り専用。
静的 (Static) メソッド	ターゲットオブジェクトをインスタンス化するために使用される静的 (Static) メソッド。必須。

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」ダイアログが開きます。



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します。

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します。環境エントリの設定に関して詳細は、27 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か共有オブジェクトです。共有オブジェクトについての詳細は、81 ページの「オブジェクトの共有」を参照してください。
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します。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`) でなければなりません。基本型 (たとえば、`int`) は使用できません。入力ドキュメント要素のデータ型の宣言については、60 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。

値の型	ユーザー操作
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、27 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と共有オブジェクトが含まれます。 共有オブジェクトを選択すると、アプリケーションロジックで、パラメータメソッドが実行される前にオブジェクトがインスタンス化されていることを確認する必要があります。 共有オブジェクトについての詳細は、81 ページの「オブジェクトの共有」を参照してください。
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、49 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。

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。つまり、「オブジェクトの解決」ダイアログの「メソッド」、「コンストラクタ」、「静的 (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。
2. 「環境エントリ」プロパティをクリックします。

3. 「...」ボタンをクリックします。

   「環境エントリ」ダイアログが開きます。

4. 「追加」をクリックします。

   「追加 環境エントリ」ダイアログが開きます。

5. ダイアログ内のフィールドに入力します。

フィールド	説明
名前	環境エントリの名前。
説明	環境エントリの説明。
型	環境エントリをマップするメソッドパラメータの型。
値	環境エントリの値。

6. 「了解」をクリックします。

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

Web サービスのセキュリティ保護

ここでは、統合プレゼンテーションサーバーを持つ Web サービスのセキュリティについて説明します。プレゼンテーションサーバーについての詳細は、2 ページの「Web サービスインタフェース」および 8 ページの「統合クライアントのアクセス」を参照してください。

Web Services モジュールでは、Web サービスをセキュリティ保護するためのメカニズムを提供しています。これらのメカニズムを使用すると、ユーザーにロールを割り当て、特定のロールにリソースへのアクセス権を与えることによって、リソースへのアクセスを制限できます。セキュリティメカニズムのそれぞれのタイプは、特定のタイプのリソースを保護します。利用可能なセキュリティメカニズムは次のとおりです。

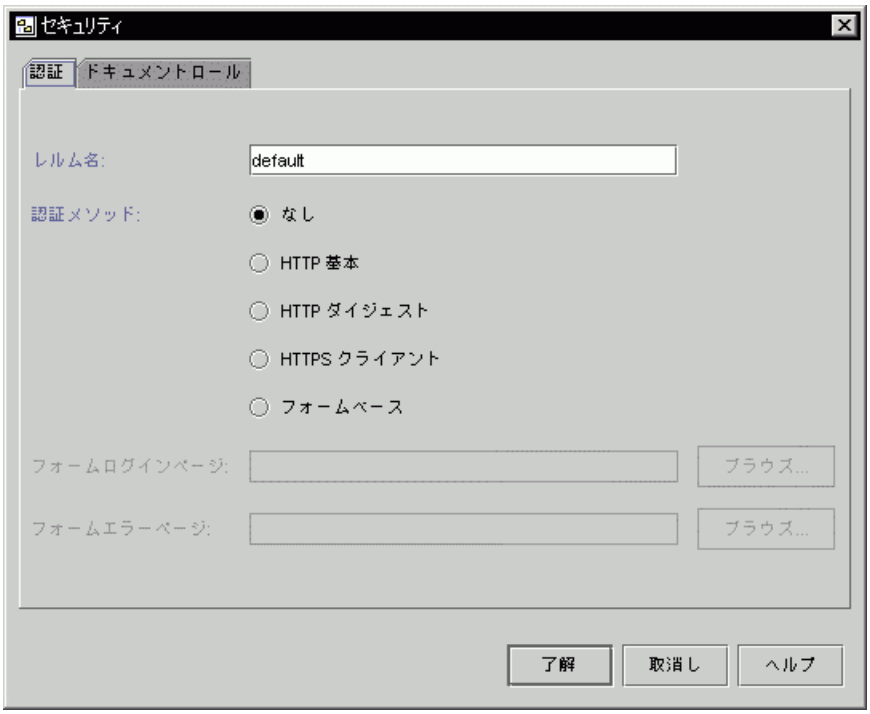
- JSP セキュリティ。このセキュリティメカニズムは、プレゼンテーションサーバーが実行する J2EE Web コンテナによって実装されます。JSP セキュリティを使用すると、次のことが実行できます。
  - ユーザーの認証。認証メカニズムにより、プレゼンテーションサーバーのクライアントに ID およびパスワードの入力を要求することによって Web サービスの正当なユーザーであることが確認できます。J2EE 認証サービスによってユーザーが検証され、確認されると認証されたロールがユーザーのセッションに割り当てられます。
  - ドキュメントへの制限アクセス (HTML ファイルおよび JSP ページなど)。ユーザーは認証後に、ロールがドキュメントへのアクセスを許可されている場合のみ、特定のドキュメントにアクセスできます。
- XML オペレーションセキュリティ。各 XML オペレーションにはロールプロパティがあります。このプロパティが設定されている場合は、そのプロパティ内にロールが指定されているクライアントのみが XML オペレーションにアクセスできます。XML オペレーションでは複数のオブジェクト上でメソッドを呼び出すことができるため、このメカニズムによりユーザーのビジネスコンポーネントとクラスに対して一定レベルのセキュリティが提供されます。
- EJB セキュリティ。EJB コンポーネント開発者は、EJB コンポーネントモデルによって提供される宣言またはプログラムセキュリティメカニズムを使用してコンポーネントにセキュリティ制約を実装できます。これらのコンポーネントが Web サービスによってアクセスされると、セキュリティ制約が強制され、例外が発生します。このメカニズムは EJB コンポーネントの詳細セキュリティを提供します。

認証の設定

認証を設定する手順は、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、「セキュリティ」を選択します。

   「セキュリティ」ダイアログが開き、「認証」タブが表示されます。



2. セキュリティ設計で特定の領域名が必要な場合は、「レルム名」フィールドに入力します。

   ほとんどの場合、デフォルト値を使用します。

3. 認証メソッドを選択します。

次の表は、オプションの説明です。J2EE 認証メソッドの追加情報については、『*Java2 Platform, Enterprise Edition Specification*』を参照してください。

オプション	説明
HTTP の基本	このメカニズムは、Web サーバーと Web ブラウザによって実装されます。これにより、 Web ブラウザに標準ログインウィンドウが表示されます。ユーザーは、ユーザー名とパスワードを入力します。入力されたデータは HTML 形式で送信され、Web サーバーで認証が行われます。
HTTP ダイジェスト	HTTP の基本に似ています。現在、Web サーバーではあまりサポートされていません。
HTTPS クライアント	HTTP を SSL で使用してセキュリティ保護された暗号化接続を提供します。クライアントからの公開鍵証明書が必要です。一般的に B2B アプリケーションおよびシングルサインオンアプリケーションで使用されます。
フォームベース	HTML ログインフォームはプレゼンテーションサーバー内に定義され、ユーザーに Web ページとして表示されます。フォームの構造は詳細に定義されています。名前とパスワードは `j_username` および `j_password` パラメータとして渡されます。フォームの `method` 属性は `post` で、`action` 属性は `j_security_check` です。

4. 認証メソッドがフォームベースの場合は、ログインページの URL を「フォームログインページ」フィールドに入力するか、「ブラウズ」をクリックしてページに移動します。

ログインページは JSP ページまたは HTML ファイルを使用できます。たとえば、`/login.jsp` と指定します。

Web サービスでクライアントページの生成を選択した場合は、このフィールドに指定できるログイン JSP ページも生成されます。生成されたクライアントページについての詳細は、35 ページの「実行時クラスとクライアントページ」を参照してください。

5. 認証メソッドがフォームベースの場合は、エラーページの URL を「フォームエラーページ」フィールドに入力するか、「ブラウズ」をクリックしてページに移動します。

   エラーページは JSP ページまたは HTML ファイルを使用できます。たとえば、`/error.jsp` と指定します。エラーページは、認証が何らかの理由により失敗した場合に表示されます。

   Web サービスでクライアントページの生成を選択した場合は、このフィールドに指定できるログインエラー JSP ページも生成されます。生成されたクライアントページについての詳細は、35 ページの「実行時クラスとクライアントページ」を参照してください。

6. 「了解」をクリックして変更内容を保存し、ダイアログを閉じます。

ドキュメントへのアクセスの制限

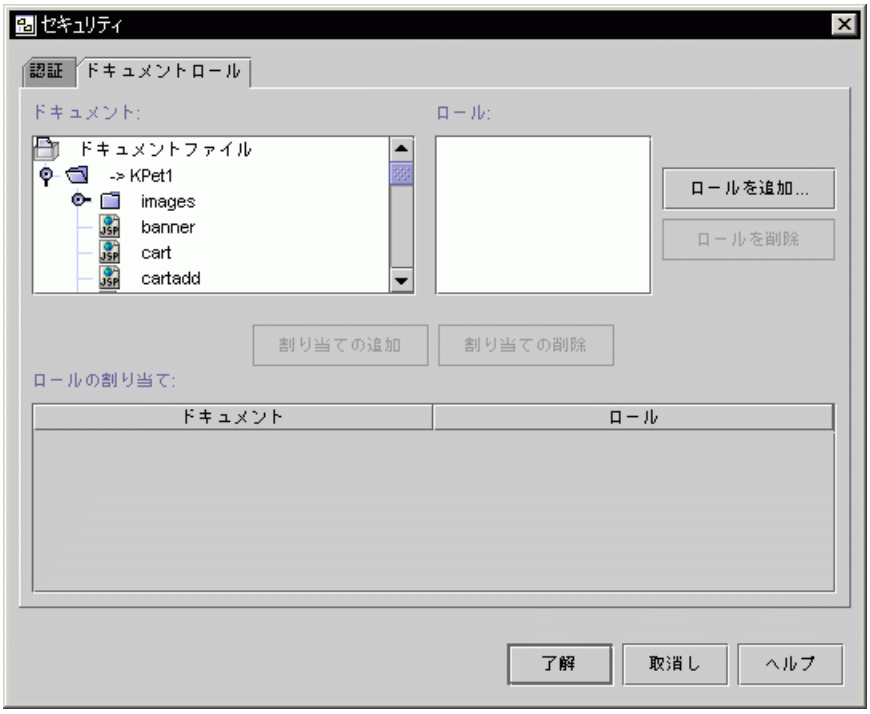
ロールをドキュメントファイルに割り当てることによってドキュメントセキュリティを実装します。認証済みで必要なロールを所有しているユーザーのみがセキュリティ保護されたドキュメントにアクセスできます。ユーザーがログインする前にセキュリティ保護されたドキュメントにアクセスしようとすると、認証プロセスが自動的に起動してログインページが表示されます。認証済みユーザーがセキュリティ保護されたドキュメントにアクセスしようとしているが、必要なロールを所有していない場合は、アクセスが拒否されます。

ドキュメントへのアクセスを制限する手順は、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、「セキュリティ」を選択します。

   「セキュリティ」ダイアログが開きます。

2. 「ドキュメントロール」タブをクリックします。



「ドキュメント」区画には、Web サービスに含まれているすべてのドキュメントが一覧表示されます。Web サービスにドキュメント参照を追加する方法についての詳細は、11 ページの「XML オペレーション、ファイル、およびライブラリの追加」を参照してください。

3. セキュリティ設計で要求されるロールを追加します。

各ロールに対して、次のことを実行します。

a. 「ロールを追加...」をクリックします。

「ロールを追加」ダイアログが開きます。

b. ロールの名前を入力し、「了解」をクリックします。

入力した名前が「ロール」区画に追加されます。

4. セキュリティ設計で要求される役割をドキュメントに割り当てます。

各ロール割り当てに対して、次のことを行います。

a. 「ドキュメント」区画でドキュメントを選択し、「ロール」区画でロールを選択します。

b. 「割り当ての追加」をクリックします。

ドキュメントはロールと組合わせられて「ロールの割り当て」区画に追加されます。

この手順を速やかに実行するには、「ドキュメント」区画で 1 つまたは複数のドキュメントを選択して、Ctrl キーを押しながら「ロール」区画で 1 つまたは複数のロールをクリックします。「割り当ての追加」をクリックすると、選択された各ドキュメントは選択した各ロールと組合わせられて「ロールの割り当て」区画に追加されます。

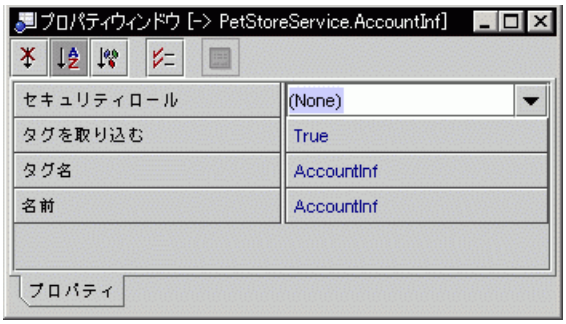
5. 「了解」をクリックして変更内容を保存し、ダイアログを閉じます。

XML オペレーションへのアクセスの制限

ロールを XML オペレーションに割り当てることによって XML オペレーションセキュリティを実装します。認証済みで必要な役割を所有しているユーザーのみがセキュリティ保護された XML オペレーションにアクセスできます。ユーザーがログインする前にセキュリティ保護された XML オペレーションにアクセスしようとすると、例外が発生します。

XML オペレーションへのアクセスを制限する手順は、次のとおりです。

1. エクスプローラで、XML オペレーションを選択します。
2. XML オペレーションのプロパティシートで「セキュリティロール」プロパティをクリックします。



3. XML オペレーションへのアクセスを付与するロールの名前を入力して Return キーを押します。

   これで XML オペレーションへのアクセスが指定されたロールに制限されます。

注 - XML オペレーションからアクセス制限を削除するには、「セキュリティロール」プロパティをクリックして、リストから（None）を選択し、Enter キーを押します。

実行時クラスとクライアントページ

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。生成するクラスは次のとおりです。

- **各 XML オペレーションに対して 1 つのクラス。**このクラスは、XML オペレーションの機能を実装します。
- **1 つの Web サービスクラス。**Web サービスを J2EE アプリケーションとして組み立てると、このクラスはステートフルセッション EJB コンポーネントとしてパッケージ化されます。実行時に、このセッション Bean はクライアント要求を特定の XML オペレーションにディスパッチします。また、返されたデータセットのキャッシュおよび管理を含む Web サービスのセッションベースの状態情報も管理します。

デフォルトで、実行時クラスを生成すると次のクライアントページも生成されます。

- **各 XML オペレーションに対して 1 つの JSP ページ。**これらの各 JSP ページは、対応する 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。開発は、JSP ページを クライアントとして使用して 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 ページをカスタマイズして、HTML クライアントが Web サービスにアクセスするために使用できるプレゼンテーションサーバーを構築するために使用できます。

- **1つの HTML 開始ページ。**開始ページは、生成された各 JSP ページに対応する HTML フォームを表示します。これらの HTML ファイルは JSP ページを起動するために必要な URL を提供します。これらのフォームには、入力パラメータが必要な場合のユーザー入力用のフィールドが含まれます。デフォルトで、このページは Web サービスの Web モジュール配備記述子に開始ページとして宣言されます。

- **1 つの HTML Web サービスエラーページ。**デフォルトで、このページは Web サービスの Web モジュール配備記述子にエラーページとして宣言されます。

- **1 つのログイン JSP ページと 1 つのログインエラー JSP ページ。**セキュリティを実装する場合は、これらのページを Web サービスアプリケーション用のログインおよびログインエラーページとして使用できます。ログインおよびログインエラーページを指定する方法については、29 ページの「Web サービスのセキュリティ保護」を参照してください。

クライアントページの生成は、Web サービスの「クライアントの生成」プロパティで制御されます。「クライアントの生成」プロパティについての詳細は、37 ページの「クライアントページの生成の制御」を参照してください。

実行時クラスの生成

Web サービスの実行時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 サービスの実行時クラスが生成され、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。

   Web サービスの「クライアントの生成」プロパティが有効になっている場合は、クライアントページが生成され、Documents という名前のフォルダに置かれます。このフォルダは Web サービスと同じレベルで作成されます。

クライアントページの生成の制御

デフォルトで、Web サービスの実行時クラスの生成時に、クライアントページも生成します。クライアントページの生成を制御するには、Web サービスの「クライアントの生成」プロパティを使用します。このプロパティが「True」に設定されている場合は、Web サービス用の実行時クラスを生成するとクライアントページが生成されます。

生成されたページは Web サービスと同じレベルで作成される Documents という名前のフォルダに置かれ、ページは Web サービスによって明示的に参照されます。たとえば、エクスプローラで Web サービスノードを展開した場合は、Documents フォルダを指す明示的な参照を見つけることができません。ただし、Web サービス J2EE アプリケーションを組み立てると、Web サービスの「クライアントの生成」プロパティが「True」に設定されている場合は、生成されたクライアントページがアプリケーションの Web モジュールに含まれます。このページは、アプリケーションを配備した後に Web サービスのテスト用に利用できます。生成されたクライアントページを使用して Web サービスをテストする方法についての詳細は、付録 A を参照してください。

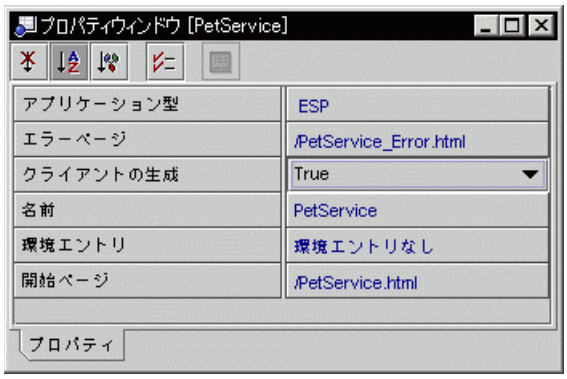
クライアントページを生成してから「クライアントの生成」プロパティを「False」に設定した場合は、生成された Documents ディレクトリは削除されませんが、Web サービスで参照されなくなり、次に配備したときにアプリケーションに自動的に含ま

れなくなります。「クライアントの生成」プロパティを「False」に設定して Documents ディレクトリを配備されたアプリケーションに含める場合は、Documents ディレクトリを Web サービスに追加する必要があります。このタスクを実行する手順についての詳細は、11 ページの「XML オペレーション、ファイル、およびライブラリの追加」を参照してください。

Web サービスを J2EE アプリケーションとして Web サービスにアセンブルして配備する方法についての詳細は、付録 A を参照してください。

Web サービスの「クライアントの生成」プロパティを設定する手順は、次のとおりです。

1. Web サービスのプロパティシートを開きます。
2. 「クライアントの生成」プロパティをクリックします。



3. コンボボックスから値を選択して Enter キーを押します。

Web サービスの実行時クラスを生成するときに、クライアントページを生成する場合は、「True」を選択します。生成されたページは、同じ名前を持つ既存のページを上書きします。

クライアントページを生成しない場合は、「False」を選択します。カスタマイズクライアントページがすでに存在する場合には、この選択が適しています。

開始ページとエラーページの設定

Web サービスの統合プレゼンテーションサーバー用の開始ページとエラーページを宣言できます。開始ページは Web サービスのデフォルトページです。プレゼンテーションサーバーは、Web サービスを表す Web コンテキストに移動するためのページを指定する URL を要求する開始ページを Web ブラウザに返します。エラーページは JSP ページの処理中にエラーが発生した場合に返されます (JSP ページが自身のエラーページを指定しないかぎり)。

デフォルトで、開始ページとエラーページは実行時クラスを生成するときに Web サービスに対して生成されます。実行時クラスの生成についての詳細は、35 ページの「実行時クラスとクライアントページ」を参照してください。

開始ページまたはエラーページを設定する手順は、次のとおりです。

1. Web サービスのプロパティシートを開きます。
2. 「開始ページ」または「エラーページ」プロパティをクリックします。

3. 「...」ボタンをクリックします。
4. 開いたファイルブラウザを使用して開始ページまたはエラーページを選択します。

次に Web サービスの実行時クラスを生成するときに、情報は Web サービスの Web モジュール配備記述子に伝達されます。

ESP レジストリ内での公開

Web Services モジュールは、Macromedia DreamWeaver などのサードパーティの設計ツールを使用してプレゼンテーションサーバー Web ページの配備をサポートします。このサポートは、Forte for Java Enterprise Service Presentation Toolkit (Forte ESP Toolkit) とともに提供されます。Forte ESP Toolkit では、Web ページデザイナが Forte ESP レジストリ内で公開される XML データソースへのリンクを作成できるようにするサードパーティツールにプラグイン機能を提供します。Web Services モジュールを使用すると、Web サービスを Forte ESP レジストリに公開できます。

Web サービスを ESP レジストリに公開する手順は、次のとおりです。

1. ESP レジストリサーブレットが配備され、実行されていることを確認します。

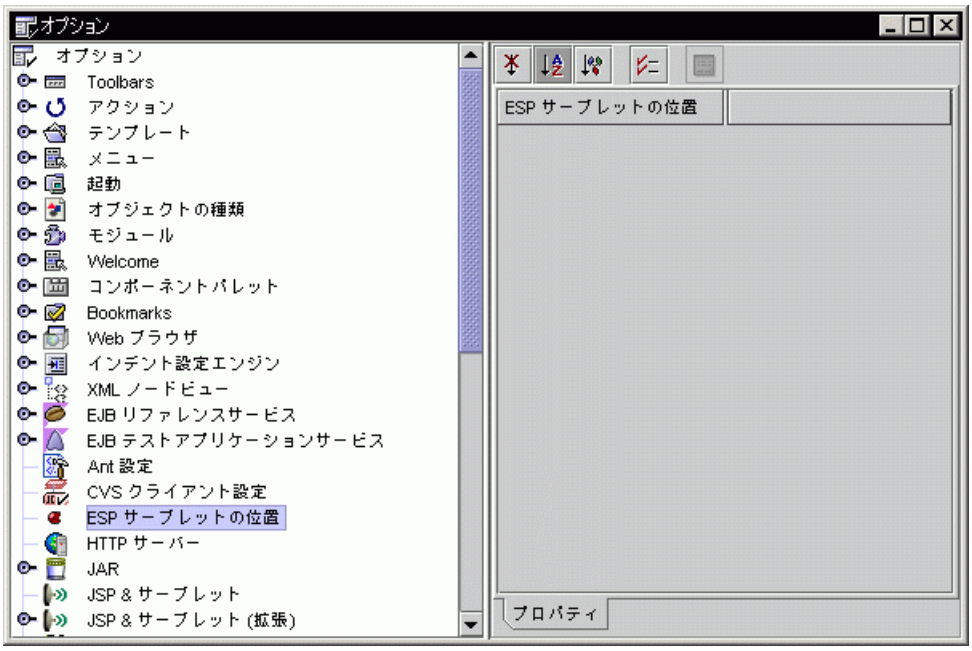
   このトピックについての詳細は、Forte for Java プログラミングシリーズの『XML データサービス用 JSP のプログラミング』 を参照してください。

2. Forte ESP レジストリサーブレットの場所を指定します。

   a. 「ツール」>「オプション」を選択します。

      「オプション」ダイアログが開きます。

   b. 左の区画の「ESP サーブレットの位置」プロパティを選択します。



c. 右の区画の「ESP サーブレットの位置」プロパティをクリックします。

d. サーブレットの URL を入力して Enter キーを押します。

次の構文を使用してください。

`http://`*hostName*`:`*portNumber*`/`*servletContextRoot*

hostName は、サーブレットが配備されているマシンの名前に置き換えます。*portNumber* は、Web サービスが待機しているポートの番号に置き換えます。*servletContextRoot* は、Web サーバー内のサーブレットのコンテキストルートに置き換えます。(コンテキストルートはサーブレットの場所を Web サーバーと関連して指定する WAR ファイル配備記述子内の XML 要素です。) サーブレットがデフォルトのコンテキストルートを使用して配備された場合は、コンテキストルートの値は `ForteESPServlet` になります。

たとえば、デフォルトのコンテキストルートおよび Web サービスを公開するために使用しているマシンと同じポート番号を使用してサーブレットを Tomcat Web サーバーに配備した場合は、次の値を使用します。

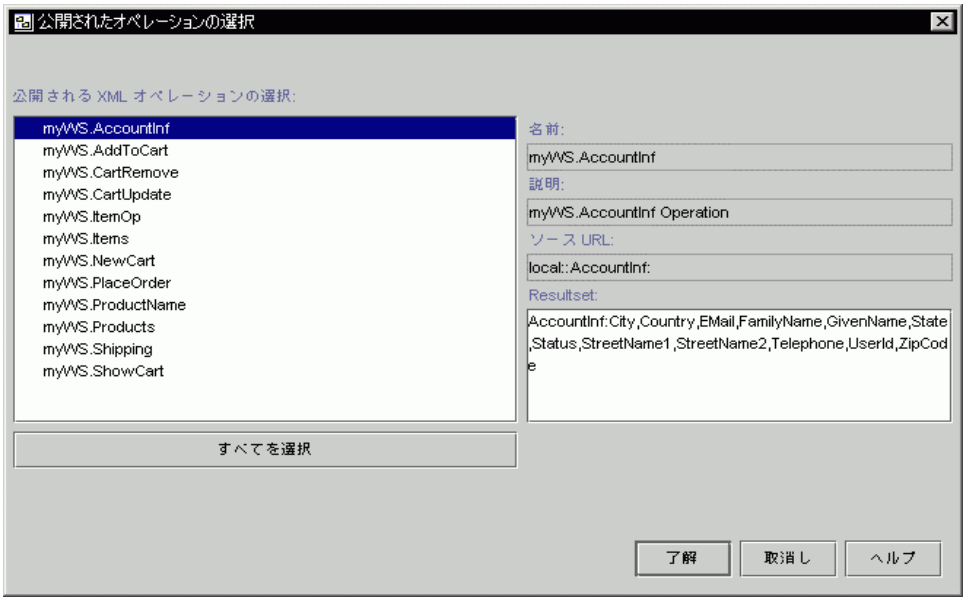
```
http://localhost:8080/ForteESPServlet
```

J2EE リファレンス実装と同様に配備した場合は、次の値を使用します。

```
http://localhost:8000/ForteESPServlet
```

3. 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、「ESP レジストリ」>「Web サービスの公開」を選択します。

   「公開されたオペレーションの選択」ダイアログが開きます。



   Web サービスに含まれている XML オペレーションが左側にリストされています。ESP レジストリにすでに公開されている XML オペレーションは、警告アイコンが付いています。ESP レジストリで必要な読み取り専用の情報は、各 XML オペレーションに対して右側に表示されます。

4. 公開する XML オペレーションを選択して、「了解」をクリックします。

   選択した XML オペレーションが ESP レジストリに追加されます。

注 - 選択したオペレーションがすでに公開されている場合は、警告ダイアログが開き、現在の登録を新しいものと置き換えるかどうかが尋ねられます。

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、実行時クラスを再生成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、フィールドの説明です。

「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`

「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 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」と言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、フィールドの説明です。

「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 サービスのプロパティシートには、Web モジュール配備記述子に伝達される 2 つのプロパティがあります。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)

「セキュリティ」ダイアログ内のすべてのフィールドは、Web モジュール配備記述子に伝達されます。「認証」タブのフィールドに対応する配備記述子内の要素は、すべて login-config 要素のサブ要素です。次の表は、フィールドの一覧です。

「認証」タブのフィールド	Web モジュール配備記述子内の要素
認証メソッド	auth-method
レルム名	realm-name
フォームログインページ	form-login-page (form-login-config のサブ要素)
フォームエラーページ	form-error-page (form-login-config のサブ要素)

「セキュリティ」ダイアログの「ドキュメントロール」タブで、「ロール」フィールド内の各エントリは Web モジュール配備記述子に別々の security-role 要素として伝達されます。データは security-role 要素のサブ要素に書き込まれます。次の表は、サブ要素の説明です。

「ドキュメントロール」タブのフィールド	Web モジュール配備記述子内の要素
ロール (派生)	description (security-role のサブ要素)
ロール	role-name (security-role のサブ要素)

「セキュリティ」ダイアログの「ドキュメントロール」タブで、「割り当て」フィールド内の各エントリは Web モジュール配備記述子に別々の security-constraint 要素として伝達されます。データは security-constraint 要素のサブ要素に書き込まれます。次の表は、サブ要素の一覧です。

「割り当て」フィールドから派生した Web モジュール配備記述子内の要素
web-resource-name (web-resource-collection のサブ要素)
url-pattern (web-resource-collection のサブ要素)
http-method (web-resource-collection のサブ要素)
role-name (auth-constraint のサブ要素)
transport-guarantee (user-data-constraint のサブ要素)

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、これらのコマンドの 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」

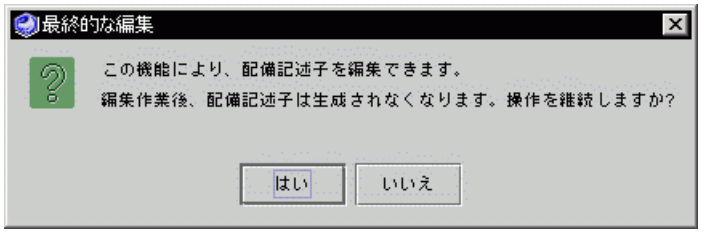
  配備記述子は、ソースエディタ内で読み取り専用モードで開きます。

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、これらのコマンドの 1 つを選択します。
   - 「配備記述子」>「Web モジュール」>「編集」
   - 「配備記述子」>「EJB モジュール」>「編集」

   「最終的な編集」ダイアログが開きます。



2. 配備記述子を再生成する必要がない場合は、「はい」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開きます。Web サービス用の実行時クラスを再生成する場合は、配備記述子は再生成されません。

第3章

XML オペレーションの開発

XML オペレーションは、Web サービスの構築ブロックです。各 XML オペレーションは、特定のクライアント要求に対する応答を定義します。

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、4 ページの「要求応答メカニズム」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作成場所を示し、「テンプレートウィザードから新規作成」ウィザードを開いて、XML オペレーション定義を作成する最初の手順を実行します。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と同様です。(実行する手順の詳細は、54 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を利用できますが、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、さらに便利です。図 3-1 は、ソースエディタで開かれた XML オペレーションを示しています。

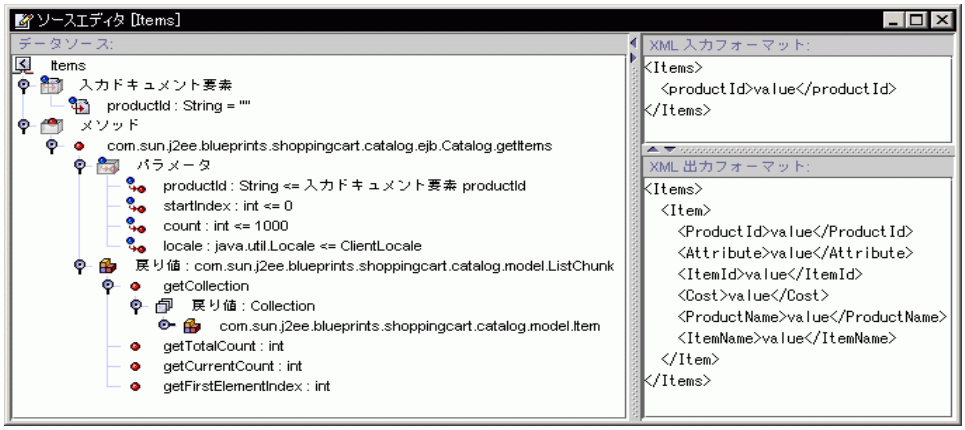


図 3-1　ソースエディタで開かれた XML オペレーション

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。各ノードのタイプは、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する
- ノードをダブルクリックしてプロパティシートを開き、そのプロパティを編集する

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指定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 3-2 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。

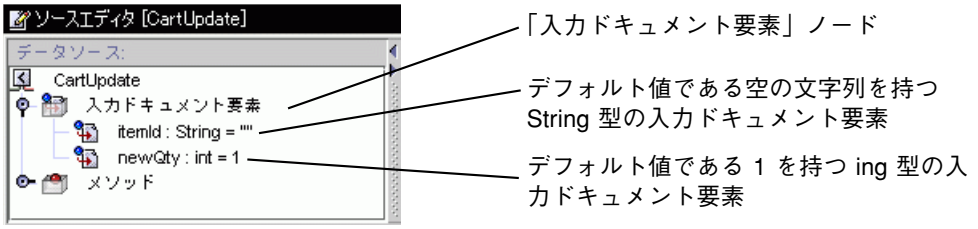


図 3-2　入力ドキュメント要素ノード

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操作を実行できます。

- 要素の追加 (60 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 (65 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 (61 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 (65 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 (62 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 (63 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

メソッドノードからは、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

- 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
- クライアントに対して、大きな結果セット内を移動するサポートを提供する
- 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

メソッドの実行とデータの返し

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削除して、再配列することもできます。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を追加、順序変更、および削除することによって 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、58 ページの「XML オペレーションへのメソッドの追加」、65 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」、65 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。

パラメータ値の指定

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、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再マップすることができます。これは、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 つをそれに再マップします。

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 サービスに定義し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、明示的に共有するオブジェクト
- 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、66 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、17 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。XML オペレーションによって返される共有オブジェクトについての詳細は、81 ページの「オブジェクトの共有」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、82 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 `Collection`) 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、70 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取得メソッドを保持している場合は、 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、69 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。

キャッシングとナビゲーションの管理

XML オペレーションが大きなデータセットを返す場合、ユーザーがサーバーを酷使したり、処理が困難なほど、大きい Web ページを受け取ることがないようにデータセットを管理したい場合があります。

メソッド呼び出しから返されるデータセットをキャッシュして、1 要求辺り 1 部分を基本としてクライアントに返されるデータを複数の部分に分することができます。また、返された Web ページ上で「次へ」、「戻る」、「最初」、「最後」ボタンのサポートを提供することで、クライアントがキャッシュされたデータセット内を簡単に

移動できるようになります。キャッシングおよびナビゲーションを有効にする方法についての詳細は、71 ページの「キャッシングおよびナビゲーションのサポート」を参照してください。

別の XML オペレーションとのオブジェクトの共有

XML オペレーションメソッド呼び出しによって返されるオブジェクトは通常、呼び出しを行なった XML オペレーション内で、要求の持続時間のみアクセス可能です。ただし、オブジェクトを特定の名前の下に明示的に共有して、セッションが持続している間、Web サービス内のすべての XML オペレーションで利用できるようにすることができます。このトピックについての詳細は、81 ページの「オブジェクトの共有」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、次の通りです。

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。

   a. メソッド呼び出しを追加または削除します。

   b. 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。

   c. メソッドパラメータをソースにマップします。

   d. 返されたオブジェクトを共有します。

e. 返されたクラスを展開または縮小します。

f. 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。

g. キャッシングとページナビゲーションを有効にします。

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、10 ページの「Web サービスの作成」および 11 ページの「XML オペレーション、ファイル、およびライブラリの追加」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、付録 A 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再生成し、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 EJB 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 Services」>「XML Operation」を選択します。

「テンプレートウィザードから新規作成」ウィザードが開きます。

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。

「メソッドの選択」ダイアログが開きます。

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。

注 - 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。図 3-3 は、論理 EJB ノードの例を示しています。

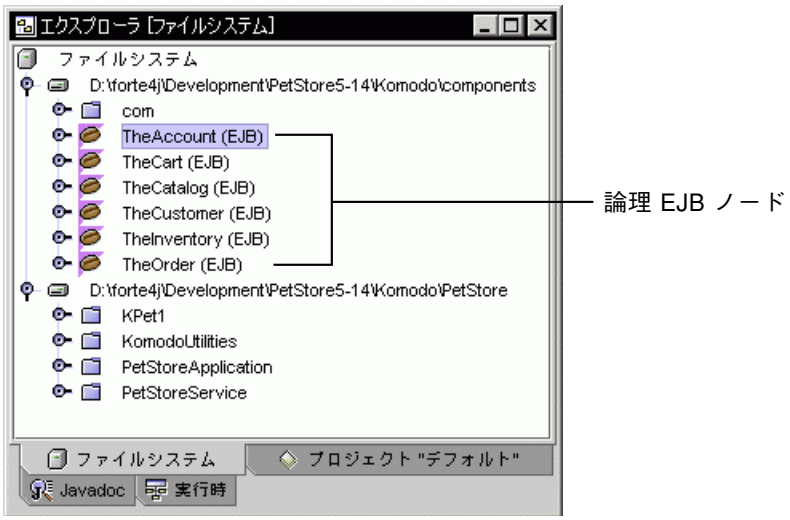


図 3-3　論理 EJB ノード

XML オペレーションを作成するときに、1 つのメソッドを取り込むことができます。XML オペレーションで追加の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、58 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 `Collection` を返した場合は、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。開いた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開いて、編集することができます (図 3-1 参照)。XML オペレーションの編集についての詳細は、58 ページの「XML オペレーションの編集」を参照してください。

EJB 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、Enterprise Bean、EJB モジュール、あるいは 1 つまたは複数の Enterprise 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 Enterprise 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。

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、10 ページの「Web サービスの作成」を参照してください。

EJB 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。

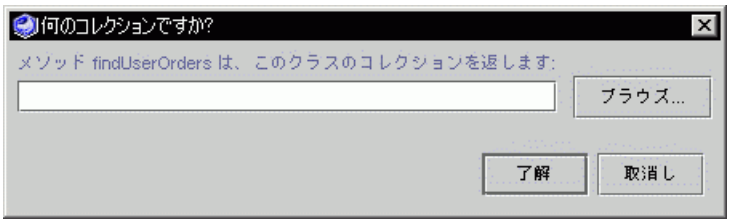
   ファイルブラウザが開きます。

2. Enterprise Bean、EJB モジュール、または Enterprise 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 Enterprise Bean 内のメソッドによって返された Java 配列または `Collection` 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。

   配列または `Collection` が返された場合は、「何のコレクションですか?」ダイアログが開きます。配列または `Collection` を返すメソッドの名前がダイアログの上部に示されます。



   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開きます。

   b. クラス、親クラス、あるいは、配列または `Collection` 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。

      このダイアログは、配列または `Collection` を返す各メソッドに対して 1 度ずつ開きます。

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。

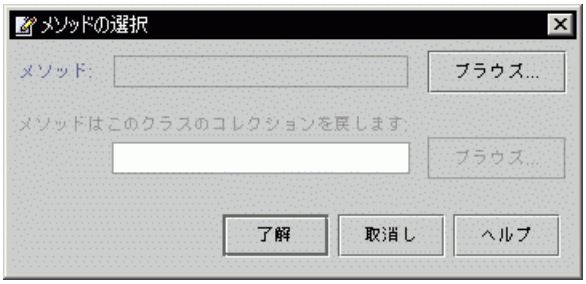
XML オペレーションへのメソッドの追加

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。

   「メソッドの選択」ダイアログが開きます。



3. 追加するメソッドを選択します。

   「メソッド」フィールドの「ブラウズ...」ボタンをクリックします。開いたファイルブラウザを使用して、メソッドを選択します。

注 - 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。図 3-3 は、論理 EJB ノードの例を示しています。

---

4. 選択したメソッドが配列または `Collection` 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。開いた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。

5. 「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ます。
   - 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ます。
   - 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」区画に追加されます。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ます。

- 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。

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なおす必要があります。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て「入力ドキュメント要素を追加」を選択します。

   「入力ドキュメント要素を追加」ダイアログが開きます。

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、たとえば、`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。この場合、`int` または `double` などの基本型は使用できません。

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。

   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。

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. 入力ドキュメント要素を選択して「プロパティ」ウィンドウを表示します。
2. 「名前」プロパティをクリックします。

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。

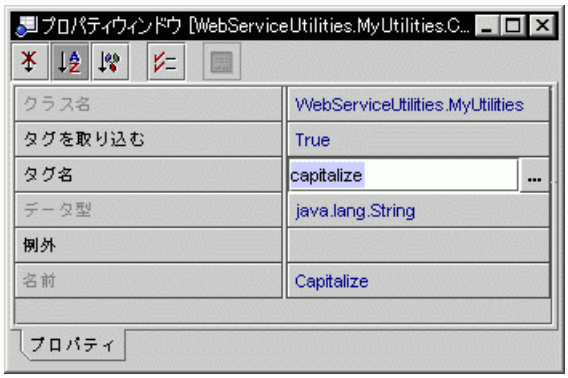
   入力ドキュメント要素の名前が変更されます。新しい名前は入力ドキュメント要素フォルダ内のノードと「XML 入力フォーマット」区画の両方に追加されます。

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. 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、「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。



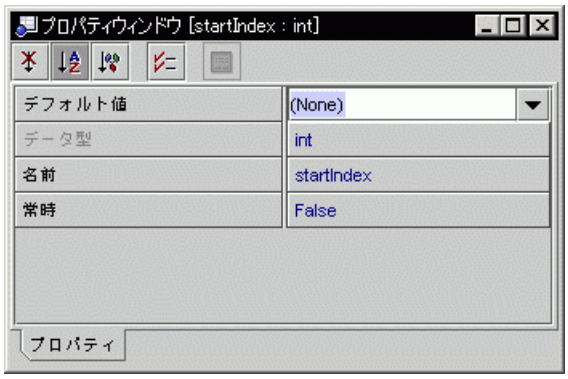
3. 要素の名前を入力して Enter キーを押します。

   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. 入力ドキュメント要素のプロパティシートを表示します。
2. 「デフォルト値」プロパティをクリックします。



3. 値を入力するか、null 値を指定する場合は「(None)」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

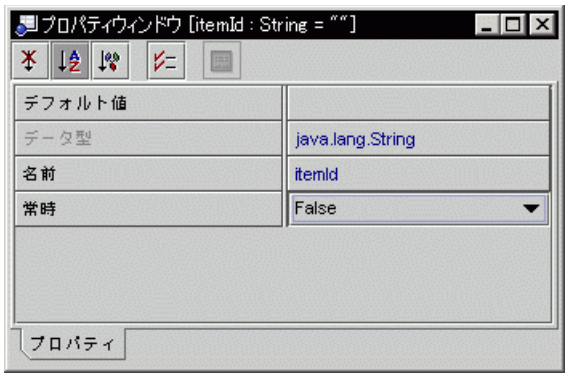
IDE では、入力ドキュメント要素の「常時」プロパティが有効になっていない場合は、メソッドパラメータにマップされない入力ドキュメント要素を自動的に削除します。

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にマップし直すと (たとえば、別のメソッドの戻り値に)、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. 入力ドキュメント要素のプロパティシートを表示します。
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、66 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て「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て「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。

   「メソッドパラメータソース」ダイアログが開きます。

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、このオプションを選択します。
メソッドの戻り値	パラメータをこの XML オペレーションの別のメソッド呼び出しの戻り値にマップする場合に、このオプションを選択します。
定数値	パラメータを定数値にマップする場合に、このオプションを選択します。

ソースタイプ	説明
システム共有オブジェクト	Web サービスは、統合プレゼンテーションサーバーを介して 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保持します。データはシステム共有オブジェクトに格納されます。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てクライアントセッションに関連付けられます。システム共有オブジェクトは、Web ブラウザクライアントに関する次のタイプのデータを保持します。 • **ユーザー名**。ユーザー名は、J2EE セキュリティメカニズムから取得されます。ユーザー名は `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。 • **ロケール**。クライアントロケールは、HTTP 要求から取得されます。ロケールは `java.util.Locale` オブジェクトとして提供されます。 • **セッション ID**。セッション ID 番号は、HTTP 要求から取得されます。セッション ID は `String` 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、このオプションを選択します。
オブジェクト参照	パラメータを Web サービス内に定義されているターゲットオブジェクトまたは別の XML オペレーションによって共有されるオブジェクトにマップする場合に、このオプションを選択します。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。 このオプションを選択すると、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。この参照を再構成して、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、17 ページの「オブジェクトのインスタンス化と参照の解決」および 81 ページの「オブジェクトの共有」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。
メソッドの戻り値	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	有効になっているコンボボックス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。 パラメータの型が `java.util.Locale` である場合は、コンボボックスに 1 つのオブジェクト ClientLocale が表示されます。 パラメータの型が `java.security.Principal` である場合は、コンボボックスに 1 つのオブジェクト UserPrincipal が一覧表示されます。 パラメータの型が `java.lang.String` である場合は、コンボボックスに次の 2 つのオブジェクトが一覧表示されます。 • UserName • Session ID

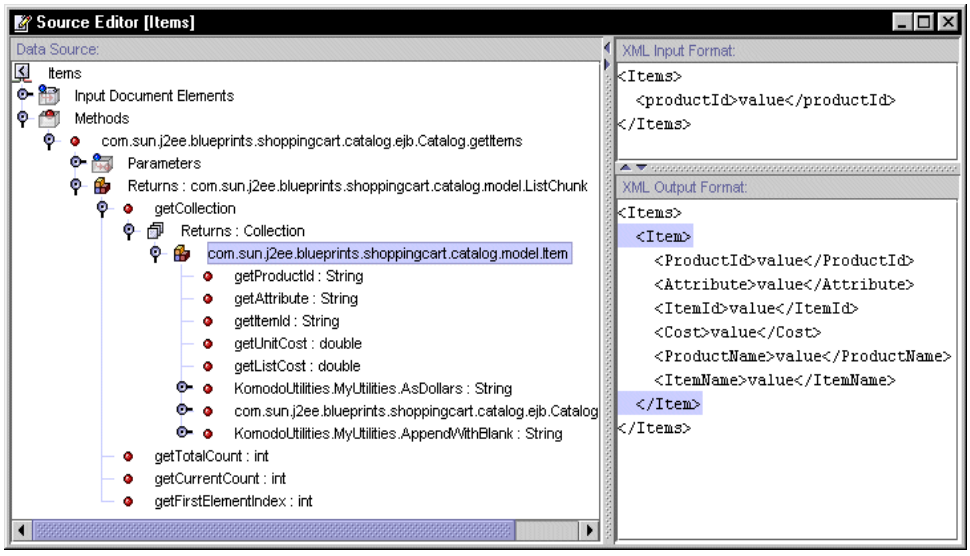
5. 「了解」をクリックします。

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」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、次の図は、「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。



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 `Collection`、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、「展開」を選択します。

   クラス上のすべての取得メソッドは、クラスノードにサブノードとして追加され、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。

キャッシングおよびナビゲーションのサポート

XML オペレーションが配列または `Collection` (「結果セット」) を取得することによって要求に応答すると、デフォルトの動作によって XML 出力ドキュメントにすべての結果セットが配置され、クライアントに返されます。「ナビゲーションオプション」ダイアログで、この動作をカスタマイズすることができます。

このダイアログ内のオプションを使用して、XML オペレーションに結果セットをキャッシュして指定した部分 (「チャンク」) のみをクライアントに返すように指示することができます。クライアントは、キャッシュされた結果セット内をナビゲートして、後続の要求それぞれでチャンクを取得することができます。クライアントは、いくつかのナビゲーションパラメータを要求に含めることによって、チャンクを返すように指定します。

注 - XML オペレーションごとに 1 つの結果セットに対してのみナビゲーションオプションを有効にすることができます。

基本サポート

XML オペレーションは、「ナビゲーションオプション」ダイアログで選択したオプションに依存して、ナビゲーションのいくつかのモデルをサポートできます。最も基本的なサポートレベルは次のように構成されます。

- 結果セットのキャッシング
- クライアントによって提供されるナビゲーションパラメータに従って、要求ごとに結果セットのチャンクを戻す (Web サービスは、これらのパラメータを XML オペレーションに入力ドキュメント要素として渡す)
- 開発者が要求ごとに返される行 (オブジェクト) 数を指定できる (つまり、開発者がチャンクサイズを設定できる)

基本サポートでは、クライアントプレゼンテーション開発者が結果セットの「Next」、「Previous」、「First」、「Last」のチャンクを返す静的ハイパーリンクを Web ページに組み込むことができます。基本サポートは、「基本リンク」オプションを選択することによって XML オペレーションに提供することができます。

拡張サポート

基本サポートを選択した後に、クライアントプレゼンテーション開発者に、より強力なハイパーリンクを組み込むことができる機能を提供する追加サポートオプションを選択することができます。追加オプションは次のとおりです。

- ブックマーク可能なリンク
- ブックマーク可能なデータ

これらのオプションは、どちらか、または両方を選択できます。どちらのオプションも XML 出力ドキュメントの方法でクライアントにナビゲーションデータを返すように XML オペレーションに指示します。このデータは、クライアントプレゼンテーション開発者が、ブックマークされたときに、クライアントのセッションが終了してキャッシュされた結果セットが参照できなくなった後でも有効な動的ハイパーリンクを構築するために使用できます。(基本サポートを利用する静的ハイパーリンクは、正しくブックマークすることはできません。) この 2 つのオプションの違いは、クライアントに返されるデータの型です。

注 - ブックマークされたハイパーリンクが共有オブジェクトを使用する XML オペレーションを起動する場合は、XML オペレーションが実行されるときに共有オブジェクトが存在することを確認する必要があります。共有オブジェクトについての詳細は、81 ページの「オブジェクトの共有」を参照してください。

ブックマーク可能なリンク

「ブックマーク可能なリンク」オプションでは、ハイパーリンクの構築に使用できる URL フラグメントが返されます。これらのフラグメントは、オリジナルの照会を指定して結果セットがブックマークされたハイパーリンクから再構築できるようにするデータを含んでいます。また、結果セットの「Next」、「Previous」、「First」、「Last」のチャンクを返すハイパーリンクを構築するために使用できるデータも含んでいます。

ブックマーク可能なデータ

「ブックマーク可能なデータ」オプションでは、次の項目を指定する raw データを返します。

- オリジナルの照会パラメータ

- 結果セット内の行の合計数
- 要求ごとに返される行の数 (チャンクサイズ)
- 現在の行位置 (現在のチャンクの最初の行を結果セット全体との関係で識別します)

これらのデータを使用して、プレゼンテーションサーバーの開発者は、コンテキスト依存ハイパーリンクを作成する論理を構築できます。たとえば、データを コンテキスト依存ナビゲーションツールバーを構築する JavaScript 機能に渡すことができます。この機能は通常、「First」、「Next」、「Previous」、「Last」のリンクを表示しますが、結果セットの最初のチャンクが表示されている場合は、「First」および「Previous」リンクは抑制されます。

XML オペレーションによって提供されるナビゲーションサポートを利用する JSP ページを開発する方法についての詳細は、77 ページの「プレゼンテーションサーバー内でのナビゲーションサポートの使用」を参照してください。

XML 入力ドキュメントおよび出力ドキュメント内のナビゲーション要素

選択するナビゲーションオプションは、XML 出力ドキュメントに追加されるナビゲーション要素を決定します。ナビゲーション出力要素は次のように追加されます。

ナビゲーションオプション	XML 出力ドキュメントに追加されるナビゲーション要素
基本	なし

ブックマーク可能なリンク	`action-next`
	`action-previous`
	`action-first`
	`action-lastaction-input`

ブックマーク可能なデータ	`action-items`
	`action-current-item`
	`action-chunk-size`
	`action-input`

どのナビゲーションオプションが選択された場合でも、同じ 2 つの XML 入力ドキュメント要素を使用して返される結果セットチャンクが選択されます。要素は次のとおりです。

入力ドキュメント要素	使用
`action`	返されるチャンクをキーワード (「Next」、「Previous」、「First」、「Last」) または絶対レコード番号を使用して指定します。現在のチャンクサイズを新しい値に設定することもできます。
`action-chunk-size`	現在のチャンクサイズを新しい値に設定するオプション要素。

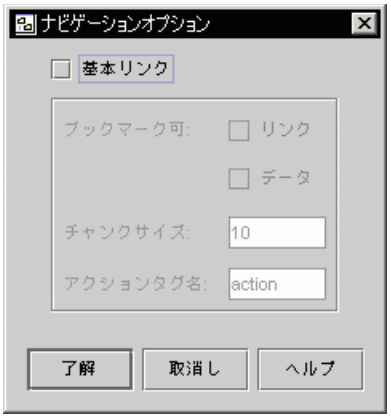
これらの要素の使用方法を示している例については、77 ページの「プレゼンテーションサーバー内でのナビゲーションサポートの使用」を参照してください。

キャッシングおよびナビゲーションのサポートの有効化

キャッシングおよびナビゲーションのサポートを有効にする手順は、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. 配列または `Collection` を表すノードをマウスの右ボタンでクリックして、「ナビゲーションオプション」を選択します。

   「ナビゲーションオプション」ダイアログが開きます。



3. 「基本リンク」チェックボックスをクリックします。

   このオプションを選択すると、XML オペレーションで結果セットをキャッシュしてこれらの要素が XML 入力ドキュメントに追加されます。

   - `action`
   - `action-chunk-size`

   追加されたこれらの要素により、クライアントが結果セットをナビゲートすることができます。

4. 任意の追加サポートオプションを選択します。

   追加のナビゲーションサポートを提供する場合は、次のチェックボックスのいずれか、または両方をクリックします。

   - ブックマーク可能なリンク

   このオプションでは、次のナビゲーション要素を XML 出力ドキュメントに追加します。

   - `action-next`
   - `action-previous`
   - `action-first`
   - `action-last`
   - `action-input`

   これらの要素には、`action` 入力要素の値として使用して結果セットのナビゲーションを可能にするデータが含まれます。`action-next`、`action-previous`、`action-first`、および `action-last` タグ値はそれぞれ、「Next」、

「Previous」、「First」、「Last」チャンクにナビゲートするために使用できます。これらのタグにも結果セットを再構築するために必要な情報を含むため、ハイパーリンクをブックマークして後から結果セットを再構築して適切なチャンクを表示するために使用することができます。

- ブックマーク可能なデータ

このオプションを選択すると、クライアントプレゼンテーション内でコンテキストに依存した、ブックマーク可能なハイパーリンクを使用することができます。このオプションでは、次の要素を XML 出力ドキュメントに追加します。

- `action-items`
- `action-current-item`
- `action-chunk-size`
- `action-input`

これらの要素は引数として JavaScript 機能に渡すことができ、コンテキスト依存ナビゲーションハイパーリンクを構築するために使用できます。

これらのオプションについての詳細は、72 ページの「拡張サポート」および 73 ページの「XML 入力ドキュメントおよび出力ドキュメント内のナビゲーション要素」を参照してください。

5. チャンクサイズを指定します。

「チャンクサイズ」フィールドに整数値を入力します。XML オペレーションは、要求ごとに指定した数の行 (オブジェクト) を返します。すべての行を返すには、0 を入力します。

6. アクションタグ名を指定します。

ナビゲーションサポートが XML 入力および出力ドキュメントに追加する要素の名前をカスタマイズするには、「アクションタグ名」フィールドに名前を入力します。たとえば、`navigation` と入力すると、`action` および `action-chunk-size` 入力要素の名前が `navigation` および `navigation-chunk-size` にそれぞれ変更されます。

7. 「了解」をクリックします。

プレゼンテーションサーバー内でのナビゲーションサポートの使用

XML オペレーションでナビゲーションが有効になっている場合、`action` パラメータを要求 URL に含めることによって結果セットナビゲーションをプレゼンテーションに追加することができます。利用可能ナビゲーション出力ドキュメント要素に依存しているので、いくつかの方法で `action` パラメータ用の値を派生することができます。

ここでは、キャッシュされた結果セットのナビゲーションに使用される入力および出力ドキュメント要素の意味論および構文について説明します。

action パラメータ

このパラメータは、返されるキャッシュされた結果セットのチャンクを識別します。有効な値は、next、previous、first、last、current、または結果セットのインデックス位置を示す整数です。オプションでこの値にコロンと整数を付加して返される行数 (新しいチャンクサイズ) を示すことができます。要求に対して明示的にチャンクサイズを設定すると、後続の結果セットのナビゲーションでチャンクサイズが変更されます。

action パラメータの値は、次のように結果セットナビゲーションに影響します。

URL	結果セットナビゲーションへの影響
my.jsp?action=next	結果セットの現在のチャンクサイズを使用して、結果セットの次のチャンクを返します。
my.jsp?action=10	行 10 から開始するチャンクを返します。
my.jsp?action=previous:2	チャンクサイズを 2 に変更して前の 2 行を返します。
my.jsp?action=1:0	チャンクサイズを 0 に変更して、ナビゲーションをオフにして結果セットすべてを返します。

action-chunk-size パラメータ

オプションの action-chunk-size パラメータは、結果セットのチャンクサイズを変更するために使用されます。この機能は、新しいチャンクサイズを action パラメータの値に付加した場合と同一です。action-chunk-size パラメータは、プレゼンテーションがチャンクサイズの指定を HTML 形式で許可している場合に役立ちます。例:

```
<form action="categories.jsp" method="GET">
  <input name="action" type="hidden" value="first" />
  <input name="action-chunk-size" type="text" size="4" />
  <input type="submit" name="Update" value="Update" />
</form>
```

ブックマーク可能リンク出力ドキュメント要素のマッピング

ブックマーク可能リンクサポートが設定されている結果セットでは、プレゼンテーション開発者によって action パラメータにマップできる出力ドキュメント要素が返されます。出力要素は次のとおりです。

- action-next
- action-previous
- action-first
- action-last

これらの各要素の値は、適切な動作値と結果セットを作成するために使用されたすべての入力パラメータが含まれます。結果セットが利用できなくなった場合 (クライアントセッションがタイムアウトした場合など) に、この情報によって XML オペレーションが結果セットを再構築することができます。

JSP タグライブラリを使用して、これら要素の値をプレゼンテーションの action パラメータにマップできます。次の JSP フラグメントは、Forte ESP タグライブラリを使用してこれを行う方法を示しています。

```
<a href="products.jsp?action=previous">
  <xsl:attribute name="href">
    <forte:expand value="products.jsp?action={Products/action-
     previous}" />
  </xsl:attribute>
  Previous
</a>
```

この JSP フラグメントは、ハイパーリンクを action-previous 出力要素からの action パラメータ値を使用して表します。

ブックマーク可能データからの動的リンクの構築

ブックマーク可能データサポートが設定されている結果セットでは、JavaScript 機能などのスクリプト記述機能に入力として渡すことができる出力ドキュメント要素が返されます。そのような機能では、ナビゲーションに適した action パラメータを生成できます。

ブックマーク可能データ結果セットに対して返される出力ドキュメント要素は、次のとおりです。

出力要素	値
action-items	整数。結果セット内の行の合計数を指定します。
action-current-item	整数。現在のチャンク内の最初の行の位置インデックスを指定します。
action-chunk-size	整数。現在のチャンクサイズを指定します。
action-input	結果セットを構築するために使用された入力パラメータのリストを含みます。

JSP タグライブラリを使用して、これら要素の値を JavaScript 機能に渡すことができます。次の JSP フラグメントは、Forte ESP タグライブラリを使用してこれを行う方法を示しています。

```
<script language="JavaScript">
    CreateNavigationScrollbar(
        <forte:expand value="{Products/action-current-
         item}">1</forte:expand>,
        <forte:expand value="{Products/action-
         items}">2</forte:expand>,
        <forte:expand value="{Products/action-chunk-
         size}">3</forte:expand>,
        '<forte:expand value="{Products/action-
         input}">p4</forte:expand>',
        'products.jsp');
    </script>
```

CreateNavigationScrollbar 機能では、適切なナビゲーションハイパーリンクを関数の入力に基づいて action パラメータを使用して構築します。

XML オペレーション結果セットナビゲーションの完全な例については、Komodo Pet Store サンプルアプリケーションを参照してください。これは Forte for Java ポータルの http://www.sun.com/forte/ffj で入手できます。

オブジェクトの共有

XML オペレーションメソッド呼び出しによって返されるオブジェクトは通常、呼び出しを行なった XML オペレーション内で要求が持続している間だけ、アクセス可能です。ただし、これらの返されたオブジェクトを明示的に共有することができます。オブジェクトを共有すると、オブジェクトはクライアントのセッションが持続している間、Web サービス内のすべての XML オペレーションで利用できます。

共有オブジェクトは次のとおり使用できます。

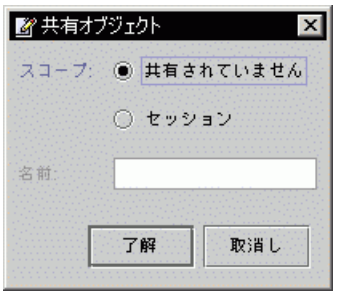
- XML オペレーションでメソッド呼び出しへのパラメータとして使用する (66 ページの「メソッドパラメータのソースへのマッピング」を参照)
- 「オブジェクト参照を解決」ダイアログでオブジェクト参照を解決するために使用されるターゲットオブジェクトとして使用する (17 ページの「オブジェクトのインスタンス化と参照の解決」を参照)
- ターゲットオブジェクトをインスタンス化するために使用されたメソッドへのパラメータとして使用する (20 ページの「ターゲットオブジェクト定義を編集する」の手順 6 を参照)

共有オブジェクトは、明示的に共有する XML オペレーションが実行されたときに作成されます。その XML オペレーションが実行される前には共有オブジェクトは存在せず、他の XML オペレーションで参照することはできません。したがって、アプリケーションロジックでは、オブジェクトを作成する XML オペレーションを実行した後、別の XML オペレーションでそのオブジェクトが参照される必要があります。

XML オペレーションでメソッド呼び出しによって返されるオブジェクトを共有する手順は、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. 返されたオブジェクトを表すノードをマウスの右ボタンでクリックし、「オブジェクトの共有」を選択します。

   「共有オブジェクト」ダイアログが開きます。



3. 「セッション」ラジオボタンをクリックします。
4. 「名前」フィールドにオブジェクトの名前を入力し、「了解」をクリックします。

   その他の XML オペレーションは、セッションが有効な間、この名前でオブジェクトを参照できます。

システム共有オブジェクト

Web サービスは、統合プレゼンテーションサーバーを介して 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保持します。データはシステム共有オブジェクトに格納されます。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てクライアントセッションに関連付けられます。システム共有オブジェクトは、Web ブラウザクライアントに関する次の種類のデータを保持します。

- **ユーザー名。**ユーザー名は、J2EE セキュリティメカニズムから取得されます。ユーザー名は `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。
- **ロケール。**クライアントロケールは、HTTP 要求から取得されます。ロケールは `java.util.Locale` オブジェクトとして提供されます。
- **セッション ID。**セッション ID 番号は、HTTP 要求から取得されます。セッション IDは `String` オブジェクトとして提供されます。

`java.util.Locale` 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 ClientLocale が表示されます。

`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、コンボボックスには次の 2 つのオブジェクトが表示されます。

- UserName
- Session ID

Static ユーティリティメソッド

ビジネスコンポーネントの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。

   これを実行する方法についての詳細は、58 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。メソッドを再配置するには、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。

      「メソッドパラメータソース」ダイアログが開きます。

   c. 「メソッドの戻り値」を選択します。

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、66 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します。

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。したがって、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。

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。

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、62 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。

Static ユーティリティクラスの例

Static ユーティリティクラスの例は、Forte for Java, Enterprise Edition とともに提供され、*FFJ_installation_directory*/modules/webservices-util.jar 内に置かれています。このユーティリティクラスを使用するには、JAR ファイルをエクスプローラにマウントして、XML オペレーションの com.sun.forte4j.webdesigner.utilities.WebServicesDesigner クラスからメソッドを取り込みます。

次の表は、このユーティリティクラスによって定義されるメソッドを示しています。

表 3-1　サンプル Static ユーティリティクラスに定義されているメソッド

メソッド名	説明
Capitalize	String をパラメータとして使用します。最初の文字を大文字にし、残りの文字を小文字にします。String を返します。
Identity	String をパラメータとして使用します。入力 String を返します。
Append	2 つの String パラメータを使用します。連結された String を返します。
AppendWithBlank	2 つの String パラメータを使用します。間にスペースを入れて 2 つを連結します。連結した String を返します。
AsDollars	double を String に変換して US ドルとしてフォーマットします。

第4章

Web サービス開発シナリオ

Web Services モジュールでは、いくつかのアプリケーション開発モデルをサポートしています。Web サービスアプリケーションを開発する方法は、開発するアプリケーションの種類と開発組織の構成によって異なります。

この章では、Web サービスアプリケーションを開発する 2 つのシナリオを説明します。各シナリオにおいて、Web サービスアプリケーションには 1 つの統合プレゼンテーションサーバーが含まれています。プレゼンテーションサーバーを使用すると、Web ブラウザで Web サービスにアクセスできます。プレゼンテーションサーバーについての詳細は、2 ページの「Web サービスインタフェース」および 8 ページの「統合クライアントのアクセス」を参照してください。

単一開発者シナリオ

このシナリオでは、1 人の開発者が Web サービスと統合プレゼンテーションサーバーを構築します。

背景

XYZ Entertainment は、ビデオと DVD を電話で販売するアウトレットショップです。この会社では、顧客がオンラインアクセスできるようにしようと考えました。顧客は Web ブラウザでカタログを参照してビデオや DVD を注文し、注文の状況を確認できるようになります。

この会社ではすでにイントラネット上で稼動する電話販売担当者用のオーダーエントリシステムを実装しています。オーダーエントリシステムでは EJB コンポーネントを使用してビジネスデータを表しています。会社の現在のタスクは、顧客がインターネットを利用してこのデータにアクセスできるようにする Web アプリケーションを構築することです。

開発マネージャの Dave は、Forte for Java, Enterprise Edition を開発プラットフォームとして選択しました。Dave は、Janice をこのプロジェクトのただ 1 人の開発担当者に決定しました。Janice は Java コンポーネントと HTML の両方の知識を持っているので、1 人でこの作業を遂行できるからです。

開発フロー

1. アプリケーションの計画。

Dave は Janice にアプリケーションに配置するページ、各ページ上のデータ、および顧客がページを移動する方法など Web サイトがどのように構成されるべきか説明します。Janice はすでに Catalog、Customer、Order、および Inventory EJB の各コンポーネントについての知識があり、必要なデータはすべて EJB インタフェースから利用できることがわかっています。顧客の購入プロセスを管理するために、別のアプリケーションからショッピングカート EJB コンポーネントを再利用します。

Janice は、アプリケーションを構築する最も簡単な方法はビジネスコンポーネントを Web サービスとして発行し、プレゼンテーションサーバーを構築して Web ブラウザによるアクセスを提供することであると判断しました。

2. XML オペレーションの開発。

Janice は IDE を起動します。アプリケーションをページごとに開発することから開始します。各ページで、関連する動的データのブロックを識別します。次にこれらのブロックを表す XML オペレーションを作成します。動的データの各ブロックは 1 つの XML オペレーションで表され、いくつかの XML オペレーションは複数の Web ページで使用されます。

各 XML オペレーションの開発は、EJB コンポーネントおよび Java クラスでメソッドを選択し、それらを XML オペレーションによって呼び出されたメソッドのリストに追加することによって行います。これらのメソッドによって返されたデータの形式が Web ページに表示するのに適していない場合があります。その場合には、Static ユーティリティーメソッドにコールを追加してデータを再フォーマットします。

作業を進めていくうちに、各 XML オペレーションが入力として要求し、出力として返すデータがソースエディタに表示されます。入力および出力データは XML ドキュメントとして表示されます。これらを検証して、オペレーションがページが要求するデータを正確に返すことを確認します。また、どの入力パラメータを HTML リンクに指定する必要があるかも確認します。

Janice は、ショッピングカートを返す XML オペレーションの開発を開始し、ShoppingCart EJB コンポーネントがアイテム番号と注文数量しか記録しないことに気づきます。ページにはアイテムの名前、サムネール画像、および在庫があるかどうかも表示される必要があります。幸いソースエディタでは、任意の複雑な操作を構築できます。単一のコード行を記述しないで複数の EJB メソッド呼び出しを追加し、呼び出しを使用して、データを Static ユーティリティーメソッドに変換し、必要なデータセットを生成します。

3. Web サービスの構築。

いくつかの XML オペレーションを構築した後で、テストを行います。IDE では、新しい Web サービスを作成して XML オペレーションを参照として追加します。次に Web サービスの実行時クラスおよびファイルを生成します。クラスは自動的にコンパイルされます。デフォルトで、生成コマンドにより、各 XML オペレーション用に開始 HTML 画面と 1 つの JSP ページを含む Documents フォルダも作成されます。Janice はこれらのページを使用して Web サービスをテストし、後からプレゼンテーションサーバーを構築します。

4. Web サービスアプリケーションの構築と配備。

「テンプレートウィザードから新規作成」ウィザードを開始して新しい J2EE アプリケーションを作成します。Web サービスを追加して、次に、すべての EJB ビジネスコンポーネントを EJB モジュールに収集してモジュールをアプリケーションに追加します。「配備」をクリックします。

IDE では Enterprise Application Archive (EAR) ファイルを作成します。このファイルには、すべてのアプリケーションコンポーネントが含まれ、次にデフォルトアプリケーションサーバーの配備サービスが呼び出されます。Web サービスアプリケーションが配備され、実行されます。

5. Web サービスのテスト。

Web ブラウザを使用して、Web サービスの生成開始画面を開きます。アプリケーションを移動して、必要に応じて入力データを入力し、表示される出力データを参照します。バックグランドで、生成された JSP ページは入力データを提供して XML オペレーションを呼び出します。XML オペレーションは EJB メソッドを呼び出し、戻り値を XML にフォーマットします。XML 出力ドキュメントは HTML にマージされて出力 Web ページを生成します。

6. Webサービス開発の完了。

IDE に戻ります。追加の XML オペレーションを作成して Web サービスに追加します。さまざまなポイントで、Web サービス J2EE アプリケーションを配備してテストします。(実行時クラスを再生成したり J2EE アプリケーションを組み立て直す必要はありません。これらのタスクは IDE によって自動的に実行されます。)

7. JSP ページのカスタマイズ。

Web サービスが正しく動作していることを確認します。ここで、高品質なユーザーインタフェースの作成を行い、ディレクトリを作成して生成したテストクライアントページをコピーします。HTML WYSIWYG オーサリングツールを起動してページのカスタマイズを開始します。

Forte ESP タグライブラリは XML に準拠しているので、このオーサリングツールは Web Services モジュールで生成された JSP ページをフォーマットして編集できます。(WYSIWYG ツールでのオーサリングは、専用タグまたは Java スクリプト記述コードを含む JSP ページで使用できない場合があります。)

これで、独自の JSP ページのカスタマイズセットができあがりました。参照をフォルダに追加して、JSP ページを Web サービスに含めます。次に Web サービスを構成して、それ以上 JSP ページを生成しないようにします。Web ページはオーサリングツールで管理されたページを使用します。次にアプリケーションを構築して配備したときに、その中にカスタマイズ JSP ページが配置されます。

8. セキュリティの追加。

ページおよび XML オペレーションにセキュリティ制約を追加することにします。また、開始画面とデフォルトのエラーページを定義します。再度アプリケーションを構築し、配備してテストします。Web サービスアプリケーションが完了しました。

9. Web サービスのメンテナンス。

Web サービスはフィードバックおよび新しい要求に基づいて発展し続けます。Web ページプレゼンテーションへの変更を処理するために、ページを HTML オーサリングツールで編集して Web サービス J2EE アプリケーションを再配備します。追加の動的データを必要とする変更を処理するために、適切な XML オペレーションおよび対応する JSP ページを編集して Web サービス J2EE アプリケーションを再配備します。Web ページと Java コンポーネント間の XML 規約によってメンテナンスタスクが制限されます。

開発チームシナリオ

このシナリオでは、開発者のチームが Web サービスと統合プレゼンテーションサーバーを構築します。

背景

XYZ Entertainment の IT Manager である Margaret は、IT 運用コストを削減する方法を模索していて、 内部の人的リソース (HR) システムに注目します。HR 部門では、従業員がデータを用紙に記入して提出するようにしています。HR 担当者は従業員データをシステムに入力します。このプロセスで発生する冗長性と転記エラーに関しては、コストがかかります。そこで、従業員が個人情報を自分で入力できるようにするシステムを実装することにします。

Margaret は開発マネージャである Dave に新しい HR システムを構築するために必要な予算を与えます。これは大きな仕事であるため、開発者のチームが必要になります。

Janice の成功プロジェクト (87 ページの「単一開発者シナリオ」を参照) に基づいて、Dave は Forte for Java, Enterprise Edition を開発プラットフォームとして使用して HR システムを Web サービスアプリケーションとして統合プレゼンテーションサーバーとともに実装することにします。

開発フロー

1. 開発タスクの割り当て。

スタッフの専門知識を最大限に利用するために、アプリケーション開発を EJB コンポーネント、Web サービス、および HTML プレゼンテーションの 3 つの部分に分けます。

開発者もこれに合わせてチームに分割します。Java コンポーネントと XML の知識がある Ed を Web サービスの開発担当者にします。Dreamweaver の知識がある Pablo に HTML プレゼンテーションの開発作業を割り当てます。

Java プログラマであるその他の開発者を EJB コンポーネントチームに入れます。

2. プロジェクトインタフェースの設計。

開発者は、各チームのサブプロジェクト間のインタフェースを設計するためにミーティングを開きます。IT 設計者は、アプリケーションに配置するページ、各ページ上のデータ、および従業員がページを移動する方法について説明します。

Pablo は、ページの見本を作成して Ed と一緒に各ページで必要な関連動的データのブロックを識別します。彼らは共同でこれら動的データのブロックを表す XML ドキュメントの構造を設計します。これらの XML ドキュメントの構造は、Web サービスとプレゼンテーションサーバー間のインタフェースを定義し、Pablo と Ed との作業コントラクトを形成します。

Ed と EJB コンポーネントチームは、EJB コンポーネントを Home および Remote インタフェースを通じて展開する方法を決定します。これらのインタフェースは、ビジネスコンポーネントと Web サービス間の規約を定義します。

3. HTML プレゼンテーションの設計。

Macromedia Dreamweaver オーサリングツールを使用して、Pablo はアプリケーション用 Web ページの設計を開始します。アプリケーションで必要な各 Web ページに対して、HTML ファイルを作成します。(HTML ファイルは後から JSP ページに変換します。)

この時点では、静的サンプルデータを動的データが必要な領域に挿入します。サンプルデータを使用すると、これらの領域をフォーマットでき、後で動的データを挿入するための可視ターゲットを作成できます。また、静的 URL をページを相互にリンクするハイパーリンクに挿入します。これらの URL には、後から動的データを挿入します。ページを業界標準に準拠するようにフォーマットします。イメージが豊富で JavaScript スクリプト記述言語を拡張して使用しています。

さまざまなポイントで、Pablo はページを IT 設計者と HR の代表者に見せます。彼らの提案に基づいてページを改訂していきます。いくつかの変更では、追加の動的データが必要になります。その場合には、Pablo は Ed と相談して XML の定義を改訂します。

4. EJB ビジネスコンポーネントの構築。

Pablo が HTML プレゼンテーションを設計している間に、EJB コンポーネントチームは次のことを行います。

- 会社の従業員データにアクセスするセッションおよびエンティティ Bean を作成します。
- 従業員データを表示および更新するために必要なビジネスメソッドをコード化します。
- J2EE セキュリティを実装して、承認されたマネージャのみが管理機能を実行できるようにします。

5. Web サービスの作成。

ビジネスコンポーネントが完了したら、Ed は IDE を使用して論理 Web サービスを作成します。論理 Web サービスは、エクスプローラ内に別のノードとして表示されます。彼は XML オペレーション、JSP ページ、HTML ファイル、JAR ファイルなどを論理 Web サービスノードに追加して Web サービスアプリケーションの機能を開発します。

6. XML オペレーションの開発とテスト。

Ed は、クライアントと Web サービス間の XML の定義に従ってXML オペレーションの開発を開始します。各 XML オペレーションは、特定のクライアントの要求に対する Web サービスの応答を表します。

各 XML オペレーションを作成するにつれて、Ed は、呼び出す XML オペレーション用の EJB メソッドを選択します。XML オペレーションが作成されると、オペレーションは自動的にソースエディタ内で開きます。ここでは、次のことを行います。

- EJB メソッドの呼び出しを追加します。
- ユーティリティメソッドの呼び出しを追加して EJB メソッドによって返される値を再フォーマットします。
- 名前を変更するか、要素を削除して XML 出力ドキュメントの構造を調整します。

Ed が最初の XML オペレーションを完了した後に、それを論理 Web サービスノードに追加して Web サービス用の実行時コードを生成します。次に J2EE アプリケーションを作成して Web サービスを追加し、そのアプリケーションを IDE 内で実行中の RI アプリケーションサーバーに配備します。

Ed が Web サービスの実行時コードを生成すると、テストクライアントも生成されます。デフォルトで、クライアントは配備されたアプリケーション内に含まれています。Ed は生成されたこのクライアントを使用して XML オペレーションをテストします。テストクライアントの構成は次のとおりです。

- XML オペレーションへの要求を発行し、その応答を表示する JSP ページ
- パラメータの入力と JSP ページへの呼び出しを可能にする HTML フォームを含む HTML 開始画面

Ed は Web ブラウザ内で開始画面を開き、フォームのフィールド内に値を入力して「送信」ボタンをクリックします。フォームは、JSP ページにパラメータを渡し、それにより XML オペレーションが呼び出されて応答が表示されます。

Ed が別の XML オペレーションを完了するたびに、それを論理 Web サービスノードに追加して Web サービスアプリケーションを再配備します。実行時コードとテストクライアントは、これを行うたびに自動的に再生成されます。テストクライアントは、Web サービス内の各 XML オペレーション用の開始画面上に常に 1 つの JSP ページと 1 つの HTML フォームを持ちます。

7. EJB コンポーネントへのアクセス。

EJB コンポーネントチームは疑問を持ちます。過去のプロジェクトでは、EJB コンポーネントにアクセスするには JavaBeans コンポーネントを構築してビジネスデータをプレゼンテーションデータに変換する必要がありました。しかし、Ed は追加メソッドまたは Bean を要求していません。XML オペレーションの抽象化により、コードを記述することなくビジネスデータをプレゼンテーションフォーマットに変換できます。その結果、EJB コンポーネントは純粋なビジネスコンポーネントのままになり、再利用性が向上します。EJB コンポーネントチームは予測よりも短期間でプロジェクトの担当部分を完了できます。

8. Web サービスの発行。

この時点で、HTML ページの設計が完了し、XML オペレーションが作成されてテストされています。Pablo は静的 HTML ページを動的 JSP ページに変換する準備ができています。

Ed は Web サービスの XML オペレーションを Forte for Java Enterprise Server Presentation (ESP) レジストリに発行します。オペレーションを発行すると、Dreamweaver 内で Forte ESP 拡張に対して XML データソースとして利用可能になります。

9. 動的データの HTML プレゼンテーションへの追加。

Pablo は Dreamweaver 内で Forte ESP 機能拡張を使用して各 HTML ページの静的データを動的データにマップします。マウスを使用して HTML ページ上のフィールドを選択してから、それぞれを XML オペレーションのいずれか 1 つの出力ドキュメントの要素にマップします。Forte ESP ウィザードでは、XML ドキュメント、挿入された ESP タグ、および実行時データソースの詳細を非表示にします。

Pablo はページ内のハイパーリンクも編集して、ターゲット XML オペレーションで要求される入力データを提供します。

Pablo がページを完了した後、Dreamweaver 内の基本的な表示を変えずに、また、ページの操作に影響を与えずに、それ以降の HTML 編集ができるようになります。各ページを .jsp 拡張子を付けて保存して JSP ページとして認識されるようにします。HTML プレゼンテーションが完了しました。

10. プレゼンテーションの Web サービスアプリケーションへの追加。

HTML プレゼンテーションが完了したら、Web サービスアプリケーションに追加します。Ed は Pablo の JSP ページを含む開発ディレクトリを自身の Forte for Java エクスプローラにマウントします。次にそのディレクトリを論理 Web サービスノードに直接追加します。これを行うと、IDE によって Web サービスに対して実行時コードが生成されるたびにテストクライアントページの生成を継続するかどうかが尋ねられます。Ed はテストクライアントページの生成を停止するよう選択します。

次に Web サービスアプリケーションを再配備します。新しいプレゼンテーション JSP ページは自動的にアプリケーションの WAR ファイル内に含まれます。

Web サービス J2EE アプリケーションは、EJB コンポーネントチームの EJB モジュール、Ed の XML オペレーション、および Dreamweaver で作成した Pablo のプレゼンテーションで構成されます。アプリケーションは完全なテストに対する準備ができました。

11. バグ修正の調整。

テスト中に発見されるバグのほとんどはサブプロジェクト間 (つまり、Home と Remote EJB インタフェース、および XML オペレーションの入力と出力ドキュメント) のインタフェースに影響しません。担当チームはこれらの切り離されたバグを修正します。追加のデータを XML 出力ドキュメントに追加することが必要なバグは、Pablo と Ed によって XML の定義が調整され、変更される必要があります。

付録 A

Web サービスの組み立て、配備およびテスト

この付録では、Web サービスを J2EE アプリケーションとして組み立て、J2EE Reference Implementation (RI) サーバーに配備して、テストする方法を説明します。このタスクは、次の下位レベルタスクで構成されています。

1. Web サービスを組み立てる準備ができていることを確認します。

このタスクを実行する方法についての詳細は、98 ページの「Web サービスの準備ができていることを確認する」を参照してください。

2. RI アプリケーションサーバーが構成され、稼動していることを確認します。

構成および稼動されていない場合は、インストールされている J2EE RI サーバーを構成して起動する必要があります。このタスクを実行する手順についての詳細は、98 ページの「RI サーバーの構成と起動」を参照してください。

3. Web サービスと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し、組み立てます。

このタスクを実行する手順についての詳細は、100 ページの「J2EE アプリケーションの組み立て」を参照してください。

4. J2EE アプリケーションを配備します。

このタスクを実行する手順についての詳細は、103 ページの「Web サービスの配備」を参照してください。

5. Web サービスをテストします。

このタスクを実行する手順についての詳細は、103 ページの「Web サービスのテスト」を参照してください。

Web サービスの準備ができていることを確認する

Web サービスアプリケーションを組み立てて配備する前に、次の点について確認します。

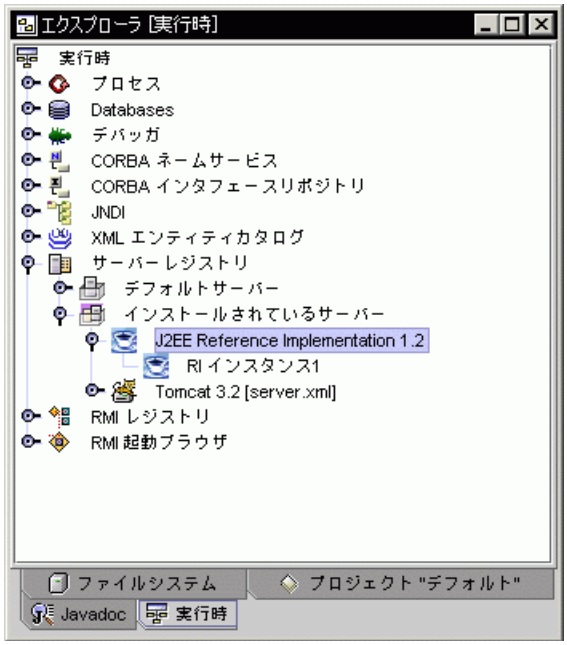
- Web サービスの実行時クラスを生成していること。このトピックについての詳細は、35 ページの「実行時クラスとクライアントページ」を参照してください。
- Web サービスによって参照されるすべての EJB コンポーネントが、EJB モジュールに組み立てられていること。
- Web サービスによって参照されるすべての EJB コンポーネントおよびそれを含む EJB モジュールが正しく構成されていること。EJB コンポーネントおよび EJB モジュールについての詳細は、Forte for Java オンラインヘルプを参照してください。

RI サーバーの構成と起動

RI アプリケーションサーバーを構成して起動する手順は、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 Reference Implementation サーバーインスタンスを追加します。

   J2EE Reference Implementation 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」を選択します。



4. J2EE Reference Implementation のプロパティシートを開きます。

   J2EE リファレンス実装ノードを再度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。

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/default.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。)

7. RI アプリケーションサーバーを起動します。

   a. コマンド行コンソールを開きます。

   b. J2EE SDK インストールの `bin` ディレクトリに移動します。

   c. `j2ee -verbose` と入力して、Enter キーを押します。

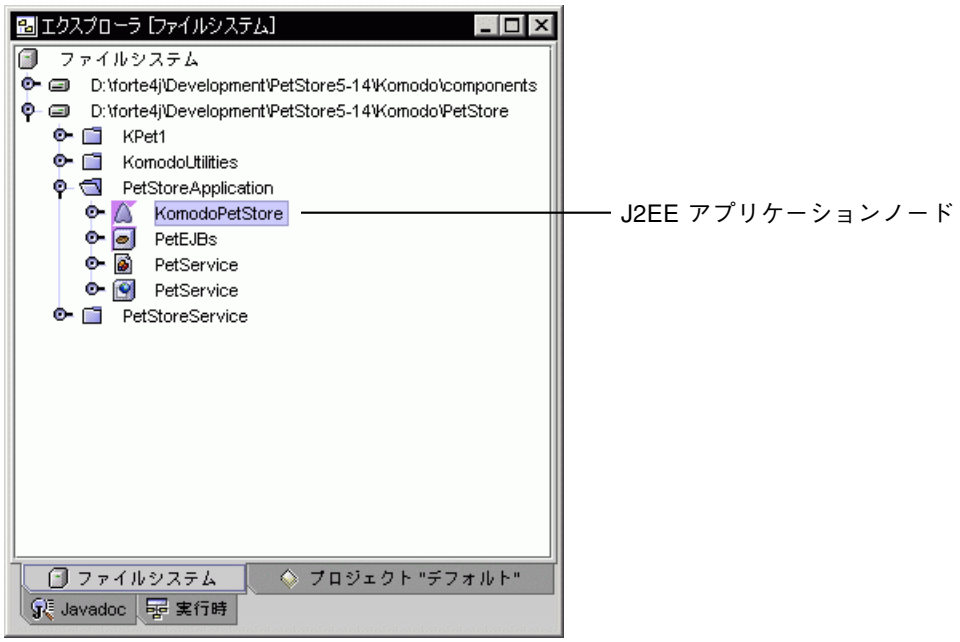
J2EE アプリケーションの組み立て

Web サービスと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. エクスプローラウィンドウの「ファイルシステム」タブをクリックします。
2. 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」>「Application」を選択します。

   「テンプレートウィザードから新規作成」ウィザードが表示されます。
3. 「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。



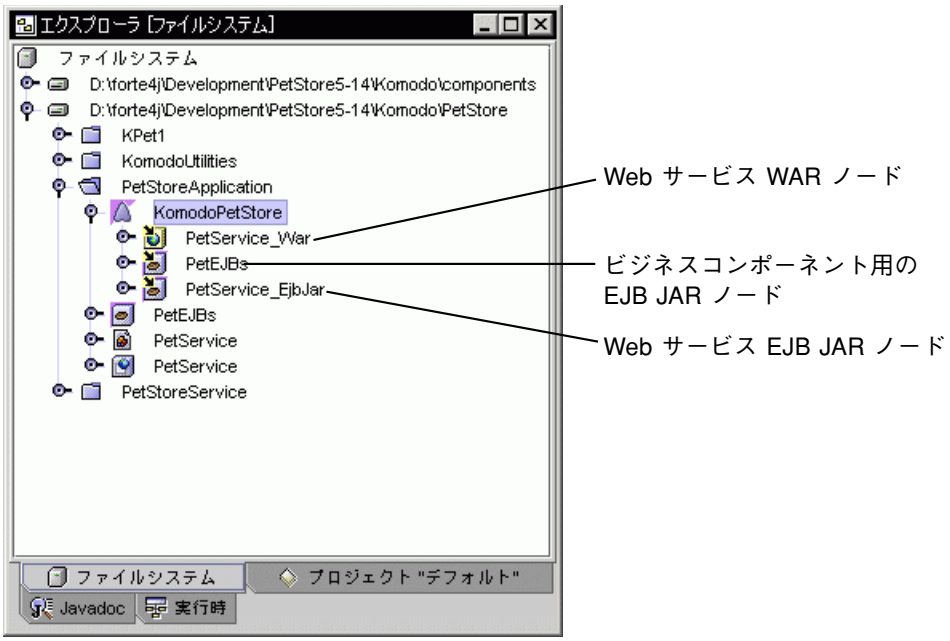
4. アプリケーションノードをマウスの右ボタンでクリックし、「モジュールの追加」を選択します。

   「アプリケーションにモジュールを追加」ダイアログが開きます。

5. 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。

   次のノードが、J2EE アプリケーションノードの下に作成されます。

   - EJB JAR ノード *webservice-name*_EjbJar。*webservice-name* は、Web サービスの名前です。このノードは、Web サービスの生成された EJB モジュールに対応します。
   - WAR ノード *webservice-name*_War。*webservice-name* は、Web サービスの名前です。このノードは、Web サービスの生成された Web モジュールに対応します。
   - 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。



6. 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 のルートを決定します。たとえば、次のコードは、Web コンテキストプロパティが `myWebService` に設定された Web サービスにアクセスするために使用される URL を示します。

   ```
   http://localhost:8000/myWebService/myPage.jsp
   ```

   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。

Web サービスの配備

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、パッケージ処理が開始されます。アプリケーションのパッケージ処理が完了すると、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。

   すべての処理が終了するまでに数分間かかります。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、配備の進捗状況が示されます。

   配備が完了すると、Web サービスが実行され、テストできる状態になります。

Web サービスのテスト

Web サービスのテストは、次のクライアントの 1 つからアクセスすることによって行います。

- **生成されたテストクライアント。**このクライアントは、いくつかの JSP ページと開始 HTML 画面で構成されます。これらのファイルは、Web サービスのクライアントの生成プロパティが有効になっている場合は、自動的に生成され、Web サービスとともに配備されます。テストクライアントには、Web ブラウザを使用してアクセスします。テストクライアントは、Web サービスの XML 出力を Web ブラウザで表示できる HTML に変換する統合プレゼンテーションサーバーです。
- **独自のカスタマイズクライアント。**このクライアントは、生成されたテストクライアントをカスタマイズして自身で配備した JSP および HTML ページで構成されます。これは生成されたテストクライアントと機能的に同等で、同じ方法でアクセスできます。カスタマイズしたクライアントページを Web サービスに追加してから配備する必要があります。

- **HTTP クライアント。**XML オペレーションは、Web ブラウザを使用して直接呼び出すことができます。この場合、Web サービスは Web ブラウザに XML を返します。XML の表示方法は、Web ブラウザによって異なります。

統合プレゼンテーションサーバーを使用したテスト

生成されたテストクライアントまたは独自のカスタマイズクライアントなどの統合プレゼンテーションサーバーを使用して Web サービスをテストする手順は、次のとおりです。

1. クライアントの開始ページを Web ブラウザで開きます。

   次の構文を使用して URL をフォーマットします。

   `http://`*host*`:`*port*`/`*webContext*`/`*welcome.html*

   host。*host* は、Web サービスアプリケーションが配備されているマシンの名前に置き換えます。

   port。*port* は、Web サービスが利用可能なポートの数に置き換えます。

   webContext。*webContext* は、Web サービスの Web モジュールの Web コンテキストプロパティの値に置き換えます。URL のこの部分は、大文字と小文字を区別します。

   welcome.html。*welcome.html* は、開始画面の名前に置き換えます。(生成されたテストクライアントの開始画面は、論理 Web サービスノードの名前と同じ名前になります。) URL のこの部分は、ページが Web モジュールの配備記述子内で開始画面として指定されている場合は、オプションです。

   たとえば、次のように想定します。

   - Web サービスアプリケーションは、デフォルトの RI インストールに配備されます。
   - Web サービスの Web モジュールの Web コンテキストプロパティは、`myService` に設定されています。
   - 開始画面は、`homePage.html` という名前が付けられます。

   次の URL は、開始画面を開きます。

   ```
   http://localhost:8000/myService/homePage.html
   ```

2. テストする XML オペレーションを呼び出します。

   生成されたクライアントを使用してテストしている場合は、開始ページの HTML フォームを使用して XML オペレーションを呼び出します。各 XML オペレーションには、対応するフォームがあります。フォームを使用すると、XML オペレーションに入力パラメータを指定できます。

   カスタマイズクライアントを使用してテストしている場合は、テスト手順はページの設計により、異なります。

HTTP クライアントを使用したテスト

HTTP クライアントを使用して Web サービスをテストする手順は、次のとおりです。

1. Web ブラウザを起動します。
2. テストする XML オペレーション別に呼び出します。

   次の構文を使用して URL をフォーマットします。

   `http://`*host*`:`*port*`/`*webContext*`/komodo?rootElement=`*XMO*`&`*inputDocElements*

   host。*host* は、Web サービスアプリケーションが配備されているマシンの名前に置き換えます。

   port。*port* は、Web サービスが利用可能なポートの数に置き換えます。

   webContext。*webContext* は、Web サービスの Web モジュールの Web コンテキストプロパティの値に置き換えます。URL のこの部分は、大文字と小文字を区別します。

   XMO。 *XMO* は、テストする XML オペレーションの名前に置き換えます。URL のこの部分は、大文字と小文字を区別します。

   inputDocElements。*inputDocElements* は、XML オペレーションで要求される XML 入力ドキュメント要素に置き換えます。これらの要素は、URL 照会パラメータとしてフォーマットし、名前と値の組をアンパサンド記号 (&) で区切ります。URL のこの部分は、大文字と小文字を区別します。

   たとえば、次のように想定します。

   - Web サービスアプリケーションは、デフォルトの RI インストールに配備されます。
   - Web サービスの Web モジュールの Web コンテキストプロパティは、`myPortal` に設定されています。
   - XML オペレーションの名前は、`AddUserXMO` です。
   - XML オペレーションの入力ドキュメントには、`userName` と `password` の 2 つの要素があります。

   次の URL は、`Micky` と言う名前の `let_me_in` というパスワードを持つ新規ユーザーを追加します。

   ```
   http://localhost:8000/myPortal/komodo?rootElement=AddUserXMO&userName=Micky&password=let_me_in
   ```

付録 B

Web サービスアプリケーションのアーキテクチャ

この付録では、Web サービスアプリケーションの実行時アーキテクチャについて説明します。次の 2 つの種類の Web サービスアプリケーションについて記載しています。

- B2B クライアントによって HTTP 接続でアクセスされる Web サービス
- 統合 ESP プレゼンテーションサーバーを持つ Web サービス

B2B クライアントによってアクセスされる Web サービス

図 B-1 は、B2B クライアントによる要求に応答する Web サービスを示しています。B2B クライアントは、HTTP 接続を使用して Web サービスにアクセスします。

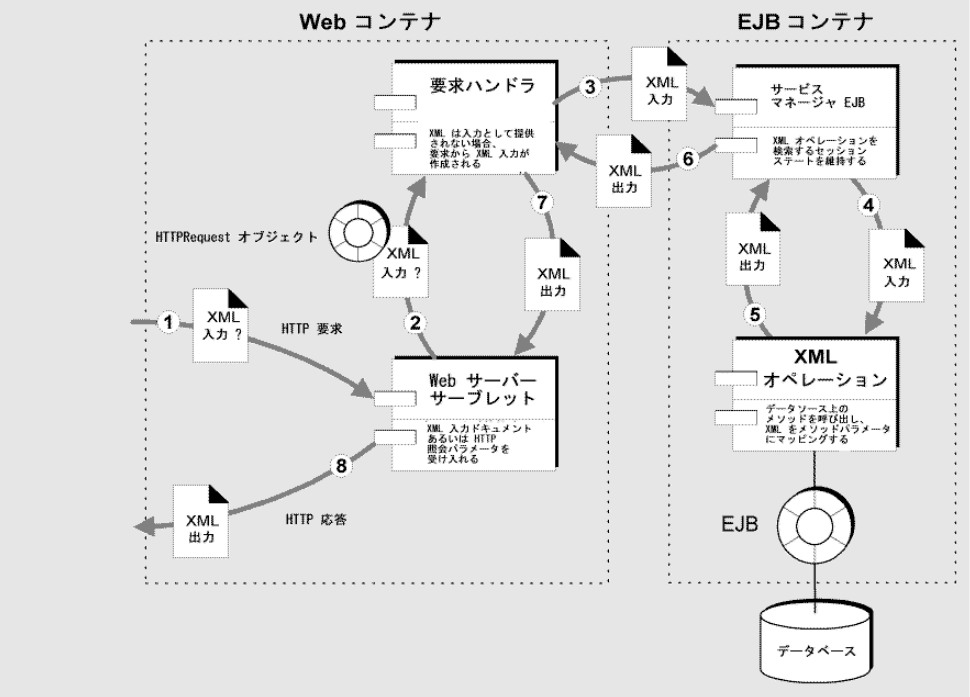


図 B-1　B2B クライアント要求に応答する Web サービス

次の手順は、図 B-1 の実行時の対話を説明しています。手順番号は図中の番号に対応しています。

1. B2B アプリケーションは、Web サービスサーブレットに対して HTTP 要求を行います。要求では HTTP `get` または `post` メソッドを使用します。ターゲット XML オペレーションおよびその入力パラメータは、URL 内で照会パラメータとして指定されます。ターゲット XML オペレーションは、`rootelement` という名前のパラメータ内に指定されます。

2. Web サービスサーブレットは `HTTPRequest` オブジェクトを要求ハンドラに渡します。Web コンテナはすでに URL を解析して URL 照会パラメータを `HTTPRequest` オブジェクトに配置しています。

   (Web サービスサーブレットは、汎用であり、Web Services モジュールにより提供されます。)

3. 要求ハンドラが HTTPRequest パラメータを XML タグにマッピングすることによって XML 入力ドキュメントを作成する場合、 ターゲット XML オペレーションを指定する rootelement タグの値は、XML ドキュメントのルート要素にマップされます。

   次に要求ハンドラは、現在のクライアントセッションに関連付けられているサービスマネージャ EJB を見つけて callWithXML メソッドを呼び出し、XML 入力ドキュメントをパラメータとして渡します。

   (要求ハンドラは、スタートアップ時にインスタンス化される Web Services モジュールによって提供される汎用クラスのインスタンスです。Web コンテナごとに 1 つだけインスタンスがあります。サービスマネージャは、設計時に Web サービス定義から生成されるステートフルセッション EJB コンポーネントです。各クライアントセッションは独自のサービスマネージャ EJB を持ちます。)

4. サービスマネージャは、ターゲット XML オペレーション (XML 入力ドキュメントのルート要素によって指定される) を指定するか、またはインスタンス化します。次に XML オペレーションの callWithXML メソッドを呼び出し、XML 入力ドキュメントをパラメータとして渡します。

   (設計時に XML オペレーション定義から XML オペレーションクラスを生成します。)

5. XML オペレーションは、次のことを行います。
   - XML 入力ドキュメントの要素をメソッドパラメータ名にマップします。
   - XML オペレーションに定義されているメソッドを呼び出します。
   - 戻り値を XML 出力ドキュメントにフォーマットします。
   - 戻り値が Collection で、XML オペレーションがクライアントナビゲーション用に構成されている場合には、キャッシュを管理します。
   - XML 出力ドキュメントを返します。

6. サービスマネージャ EJB は、XML 出力ドキュメントも返します。

7. 要求ハンドラも、XML 出力ドキュメントを返します。

8. Web サービスサーブレットは、XML ドキュメントを HTTP 応答内でクライアントに返します。

プレゼンテーションサーバーを使用した Web サービス

図 B-2 は、ESP プレゼンテーションサーバーを統合クライアントとして持つ Web サービスを示しています。プレゼンテーションサーバーは、Web サービスと同じコンテナ内で実行します。HTTP 要求を Web サービスの代わりに受け取り、Web サービスの応答を HTML、WML、または XML など、自身のクライアントに適切なフォーマットに変換します。

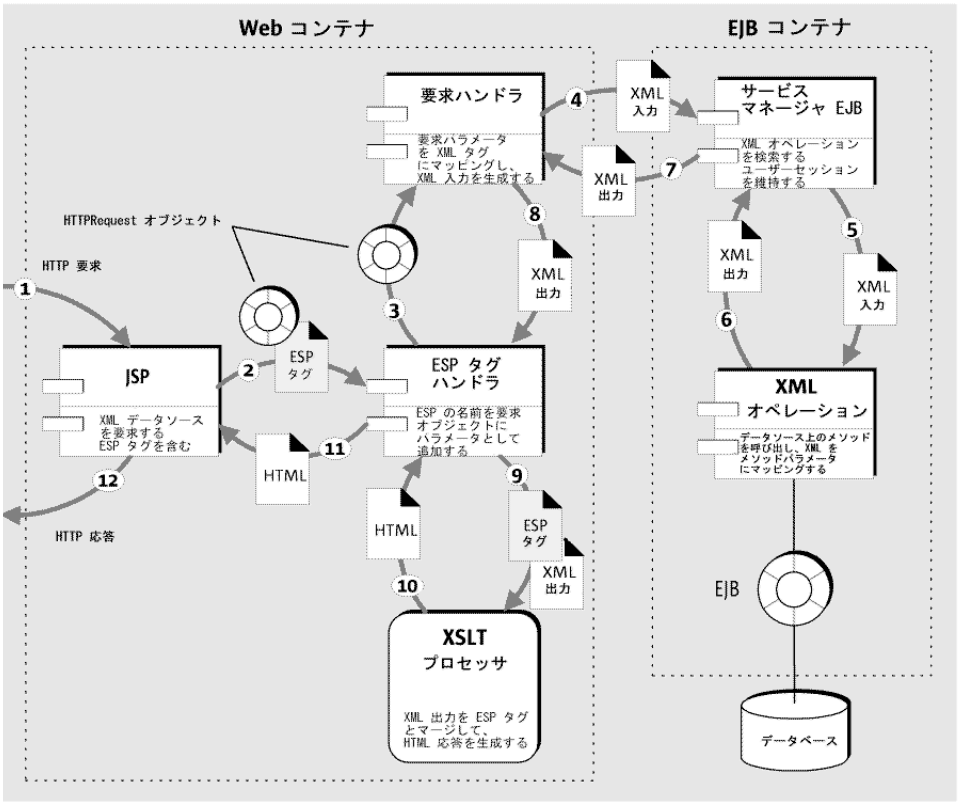


図 B-2　統合 ESP プレゼンテーションサーバーを使用した Web サービス

次の手順は、図 B-2 の実行時の対話を説明しています。手順番号は図中の番号と対応しています。

1. ブラウザを使用してページを表示しているユーザーが、Web サーバー上で配備される ESP タグを含む JSP ページをターゲットとして HTTP `GET` または `POST` を発行する URL をクリックします。入力パラメータが名前と値の組として照会パラメータまたは `POST` データのどちらかを使用して渡されます。

2. JSP ページが実行されます。ESP タグライブラリがロードされます。処理中に、ESP タグが検出されます。ESP タグ内のタグブロックと `HTTPRequest` オブジェクトが ESP タグハンドラに渡されます。

   (この JSP ページは、Web サービスから設計時に生成できます。各 XML オペレーションに対して 1 つが生成されます。)

3. ESP タグハンドラは、Web サービスのデータソースを要求しているタグを検出します。ハンドラは要求された XML オペレーションの名前を ESP タグから抽出し、`HTTPRequest` オブジェクトにパラメータとして追加します。次に `HTTPRequest` オブジェクトを Web サービスの要求ハンドラに渡します。

4. 要求ハンドラは `HTTPRequest` パラメータを XML タグにマッピングすることによって XML 入力ドキュメントを作成します。XML オペレーションの名前を XML 入力ドキュメントのルート要素として使用します。次に、現在のユーザーのセッションに関連付けられているサービスマネージャ EJB コンポーネントを指定して `callWithXML` メソッドを呼び出し、XML 入力ドキュメントをパラメータとして渡します。

   (要求ハンドラは、起動時にインスタンス化される Web Services IDE モジュールによって提供される汎用クラスのインスタンスです。Web コンテナごとに 1 つのみのインスタンスがあります。サービスマネージャは、設計時に Web サービス定義から生成されるステートフルセッション EJB コンポーネントです。各ユーザーセッションは独自のサービスマネージャ EJB を持ちます。)

5. サービスマネージャは、適切な XML オペレーション (XML 入力ドキュメントのルート要素によって指定され、現在のユーザーのセッションに関連付けられている) を指定するか、またはインスタンス化します。次に XML オペレーションの `callWithXML` メソッドを呼び出し、XML 入力ドキュメントをパラメータとして渡します。

   (設計時に XML オペレーション定義から XML オペレーションクラスを生成します。)

6. XML オペレーションは、次のことを行います。
   - XML 入力ドキュメントの要素をメソッドパラメータ名にマップします。
   - XML オペレーションに定義されているメソッドを呼び出します。
   - 戻り値を XML 出力ドキュメントにフォーマットします。
   - 戻り値が `Collection` で、XML オペレーションがクライアントナビゲーション用に構成されている場合には、キャッシュを管理します。
   - XML 出力ドキュメントを返します。
7. サービスマネージャ EJB は、XML 出力ドキュメントも返します。
8. 要求ハンドラも、XML 出力ドキュメントを返します。
9. ESP タグハンドラは、XML 出力ドキュメントと ESP タグのコンテンツを XSLT プロセッサに渡します。
10. XSLT プロセッサは ESP タグのコンテンツを XML 出力ドキュメントにマージし、その結果 (HTML、WML、XML) を ESP タグハンドラに返します。
11. ESP タグハンドラは HTML を JSP ページに返します。
12. JSP ページは HTML を Web ページの動的領域として HTTP 応答に含んでいます。

    JSP プロセッサが別の ESP タグを検出した場合、処理手順が繰り返されます。処理が完了したときに、生成される出力マークアップ (たとえば、HTML、WML、XML) ストリームが Web サーバーに返されてから、Web ブラウザに返されます。

索引

B

B2B クライアント, 3

E

EJB コマンドからのオペレーションの生成, 57
EJB コンテナ, 4, 8
ESP レジストリ、公開, 40

J

JAR
  web サービスへの追加, 11
Javadoc
  Forte for Java での使用, xvi
JSP ページ
  生成, 36

U

URL、要求データを渡すために使用, 7

W

Web コンテナ, 4, 8
web サービス
  ESP レジストリへの公開, 40
  HTTP クライアントによるアクセス, 7
  XML オペレーションおよびファイルの追加, 11
  アーキテクチャ, 4
  アプリケーションの組み立て, 100
  アプリケーションの配備, 103
  開発ワークフロー, 9
  クライアントタイプ, 3
  クライアントへのインタフェース, 2
  作成方法, 10
  実行時環境, 8
  説明, 1
  テスト, 103
  内部変換, 3
  ビジネスコンポーネントへのインタフェース, 3
  プレゼンテーションサーバーによるアクセス, 8
  利点, 1
web サービスアプリケーションの組み立て, 100
web サービスアプリケーションのテスト, 103
web サービスアプリケーショの配備, 103
web サービスサーブレット, 7
web サービスでのオブジェクトのインスタンス化, 17
web サービスのアーキテクチャ, 4
web サービスの実行時環境, 4
web サービスへの HTTP アクセス, 7

X

XML オペレーションによって返されるクラス
  展開と縮小, 51
XML オペレーション
  説明, 4
  EJB から生成, 57
  web サービスへの追加, 11
  開発, 47
  開発ワークフロー, 53
  返されたオブジェクトの共有, 81
  作成, 54
  入力ドキュメント要素を追加する, 60
  パラメータのマッピング, 50
  編集, 58
  メソッド呼び出しの追加, 58
  戻りデータの指定, 50
XML オペレーションによって返されるクラスの
  展開, 51
XML オペレーションでのメソッド呼び出し
  並び替え, 65
  削除, 52, 65
  実行順序, 50
  説明, 50
  追加, 58
XML オペレーションによって返されるクラス
  縮小, 70
  展開, 70
XML オペレーションによって返されるクラスの
  展開, 70
XML オペレーションの実行, 4
XML 出力ドキュメント
  要素の除外, 69
  要素を取り込む, 70
「XML 出力フォーマット」区画, 48
「XML 入力フォーマット」区画, 48

あ

アプリケーション、web サービス用の組み立
  て, 100
アプリケーションサーバー, 4

う

上へ移動コマンド, 65

え

エラーページ
  生成, 36
  設定, 39

お

応答メカニズム, 4
オブジェクト
  web サービスでのインスタンス化, 17
  別の XML オペレーションと共有, 52, 81
オブジェクト参照
  解決, 17
オブジェクト参照の解決, 17
オブジェクト参照
  ターゲットオブジェクトの指定, 18
オブジェクト参照を解決コマンド, 18, 20, 21
「オブジェクト参照を解決」ダイアログ, 18, 21
オブジェクトの共有, 52, 81
  コマンド, 81
  ダイアログ, 81
オペレーション、XML オペレーションを参照, 4

か

開始画面
  生成, 36
  設定, 39
開発シナリオ
  単一開発者, 87
  チーム, 91
開発ワークフロー
  web サービス, 9
  XML オペレーション, 53
環境エントリ
  概要, 27
  追加, 27

き

「基本リンク」オプション, 71
キャッシュされたデータタセットのナビゲーション, 53
キャッシュされたデータのナビゲーション, 71
キャッシュデータ、ブックマーク, 72

く

クライアント
  web サービスへのアクセス, 7, 8
  タイプ, 3
  統合, 8
クライアントページ
  生成, 35
  生成の制御, 37
「クライアントを生成」プロパティ, 37
クラスの縮小, 51, 70

こ

コマンド
  EJB からのオペレーションの生成, 57
  上へ移動, 65
  オブジェクト参照を解決, 18, 20, 21
  オブジェクトの共有, 81
  参照の追加, 14
  下へ移動, 65
  縮小, 70
  出力からタグを省略, 69
  出力にタグを取り込み, 70
  セキュリティ, 30, 32
  ソースの変更, 66
  展開, 70
  ナビゲーションオプション, 74
  入力ドキュメント要素の追加, 60
  メソッドを追加, 59

さ

サーバー
  RI の構成と開始, 98
  アプリケーション, 4
  プレゼンテーション, 3
サーブレット
  ESP レジストリ, 40
  web サービス, 7
削除
  web サービスからのリファレンス, 12
参照
  web サービスからの削除方法, 16
  web サービスへの追加, 14
  解決, 17
参照の追加コマンド, 14

し

下へ移動コマンド, 65
実行時クラス、生成, 35
実行時クラスとクライアントページの生成, 35
縮小コマンド, 70
出力からタグを省略コマンド, 69
出力にタグを取り込みコマンド, 70

す

スコープ
  セッション, 21
  メッセージ, 21

せ

セキュリティ
  定義, 29
  ドキュメントへのアクセスの制御, 29
  ドキュメントへのアクセスを制限する方法, 32
  認証、説明, 29
  認証の定義, 30
セキュリティコマンド, 30, 32
「セキュリティ」ダイアログ, 30, 32
セッションのスコープ, 21

そ

ソースの変更コマンド, 66

た

ターゲットオブジェクト, 17
　オブジェクト参照のターゲットとして指定, 18
　定義の編集, 20
ダイアログ
　「オブジェクト参照を解決」, 18, 21
　オブジェクト参照を解決, 18
　オブジェクトの共有, 81
　セキュリティ, 30, 32
　ナビゲーションオプション, 71, 74
　入力ドキュメント要素の追加, 60
　メソッドの選択, 59
　メソッドパラメータ, 23
　メソッドパラメータソース, 66
　ロールを追加, 33

つ

追加
　web サービスへのリファレンス, 11

て

データ
　XML オペレーションによって返される, 50
　クライアントへの戻り値を少なくする, 52
　展開または縮小したデータの取得, 51
　ブックマーク, 72
「データソース」区画, 48
データのキャッシュ, 53, 71
展開コマンド, 70

な

ナビゲーションオプションコマンド, 74
「ナビゲーションオプション」ダイアログ, 71, 74

に

入力ドキュメント要素
　並び替え, 65
入力ドキュメント、要素を追加する, 60
入力ドキュメント要素
　削除, 65
入力ドキュメント要素の追加
　コマンド, 60
　ダイアログ, 60
入力ドキュメント要素フォルダ, 49
認証
　説明, 29
　定義, 30

は

配備記述子
　web サービス用に生成, 43
　伝達される IDE フィールド, 43
　表示, 43
　編集, 43
配備記述子に伝達される IDE フィールド, 43
パラメータ
　web サービスでのマッピング, 23
　XML オペレーションでのマッピング, 50
　XML オペレーションでのマップピング, 23
　ソースタイプ, 51
　マッピング, 50, 66
パラメータのマッピング, 66
パラメータフォルダ, 50

ふ

ファイル
　web サービスへの追加, 11
フォルダ
　web サービスへの追加, 11

ブックマーク可能なデータ
説明, 72
有効化, 74
ブックマーク可能なリンク
説明, 72
有効化, 74
ブックマークのキャッシュデータ, 72
プレゼンテーションサーバー, 3
web サービスへのアクセス, 8
プロパティ
「クライアントを生成」, 37

ほ

ポストメソッド、要求データを渡すために使用, 7

め

「メソッドの選択」ダイアログ, 59
メソッドパラメータ
マッピング, 66
「メソッドパラメータソース」ダイアログ, 66
「メソッドパラメータ」ダイアログ, 23
メソッドフォルダ, 49
メソッドを追加コマンド, 59
メッセージのスコープ, 21

ゆ

ユーティリティメソッド、Static
概要, 83
構成, 83
サンプルクラス, 86
使用, 83

よ

要求応答メカニズム, 4
要求データ
クライアントから web サービスへ渡す, 7
要素
XML 出力ドキュメントからの除外, 69
XML 出力ドキュメントに取り込む, 70
XML 入力ドキュメントへの追加, 60
XML 出力ドキュメントからの除外, 52
要素、入力ドキュメント
並び替え, 65
削除, 65

ろ

「ロールを追加」ダイアログ, 33
ログインページ
生成, 36