Novell exteNd Composer™

4.2

www.novell.com

ユーザガイド

Novell®

保証と著作権

米国 Novell, Inc.
1800 South Novell Place
Provo, UT 85606

www.novell.com

Novell exteNd Connect ユーザガイド
2003 年 1 月
000-000000-000

オンラインマニュアル： この製品およびその他の Novell 製品のオンラインマニュアルや更新情報については、**http://www.novell.com/documentation** を参照してください。

目次

このガイドについて

目的

このガイドでは、Web サービスなどの B2B 統合アプリケーションを作成するために視覚的な設計環境を提供する Novell exteNd Composer の使用方法について説明します。ここでは、Composer の設計時の機能の使用に関する情報を取り扱います。ランタイムの機能については、『Composer Enterprise Server ガイド』で詳しく説明されています。

対象読者

このガイドは、exteNd Composer を使用して J2EE アプリケーション (Web サービスを含む) を作成するアプリケーション設計者を対象としています。

前提条件

XML 関連の標準 (スキーマ、XSL、および XPath を含む)、ドキュメントオブジェクトモデル、およびファイルのパッケージ (JAR/EAR/WAR ファイル) に関する基本的な J2EE の概念について精通している必要があります。また、ECMAScript に関する知識があるとさらに便利です。Web サービスを作成する場合は、WSDL および関連する標準について熟知している必要があります。

追加のマニュアル

Novell exteNd Director の完全なマニュアルのセットは、Novell マニュアルの Web サイト (**http://www.novell.com/documentation-index/index.jsp**) を参照してください。

構成

このガイドは、次のように編成されています。

章	説明
第 1 章、「exteNd Composer へようこそ」	exteNd Composer の概要、機能、および設計理念について説明します。
第 2 章、「アプリケーションの計画」	XML 統合アプリケーションの設計と作成に必要な準備事項について説明します。
第 3 章、「exteNd Composer をお使いになる前に」	製品の起動と Composer 環境の要素について説明します。

章	説明
第 4 章、「プロジェクトの作成と管理」	プロジェクトとその要素、およびこれらの作成方法について説明します。
第 5 章、「XML テンプレート」	XML テンプレート、サンプルドキュメント、DTD、XSL スタイルシート、および XML カテゴリに加えて、これらを使用する場合とその方法について説明します。
第 6 章、「XML Map コンポーネントの作成」	XML Map コンポーネント、コンポーネントを作成するための XML サンプルドキュメントの使用、DOM、および DOM の使用方法とその理由について説明します。
第 7 章、「基本的なアクション」	Map アクション、Log アクション、さまざまなフロー制御アクションなどを含む、Composer のすべてのコンポーネントエディタで使用可能なコアアクションについて説明します。段階を追った手順が各アクションの作成方法に対して示されています。
第 8 章、「高度なアクション」	Declare Group、Repeat for Group、Process XSL、Repeat While、Raise Error、Try/On Error、および XML Interchange を含む高度なアクションについて説明します。
第 9 章、「リソース」	スキーマリソース、WSDL、コードマップ、コードマップテーブル、接続、およびカスタムスクリプトを含む、Composer リソースのさまざまなタイプについて説明します。
第 10 章、「exteNd Composer でのカスタムスクリプト作成および XPath 論理」	Composer の組み込み ECMAScript 機能を使用したカスタムスクリプト作成について説明します。また、XPath、DOM、およびJavaと組み合わせてスクリプトを使用する方法についても言及します。Composer の組み込み ECMAScript 拡張に対する API ガイドも記載されています。
第 11 章、「共通タスクへのアクションの適用」	要素とデータのマッピング、リーフ要素のマッピング、ディープコピーのマッピング、コードテーブルと関数を使用した要素の変換、Loop アクションの実行、および集約計算の実行について説明します。
第 12 章、「アニメーションツールを使用したテスト」	アニメーションツールと、それらをサービスおよびコンポーネントのテストで使用する方法について説明します。
第 13 章、「サービスの操作」	サービスの概要、サービスを作成またはインポートする方法、サービスエディタの概要、およびサービスをコンポーネントとともに作成する方法について説明します。また、この章では、WSDL リソースに含まれる情報に基づいて外部 Web サービスを呼び出す方法の例も詳しく示します。

章	説明
第 14 章、「レジストリの操作」	UDDI レジストリの検索方法、UDDI レジストリへの発行方法、レジストリからの WSDL の取得方法などを含む、Composer の［Registries］タブに関連付けられている機能の使用方法について説明します。
第 15 章、「配備に対するプロジェクトの準備」	アプリケーションサーバへの Composer サービスの配備に関する基本的な事項について説明します。
付録 A、「XCCLASSPATH を使用した exteNd への Java 拡張の追加」	exteNd CLASSPATH で Java クラスパスをアクセス可能にする方法について説明します。
付録 B、「予約語」	予約語をリストします。予約語は、ユーザ定義の変数名またはラベルに使用することはできません。
付録 C、「用語集」	このガイドで使用されている重要な用語を定義します。

PDF マニュアルについて

このガイドを Acrobat Reader で表示する際は、さまざまなナビゲーション機能を使用できます。

- (ウィンドウの左側にある)［ブックマーク］フレームには、ガイドのコンテンツが章名、見出し、および小見出し別にリストされます。コンテンツツリーにリストされている各トピックは、クリックできるリンクです。ツリーノードでサブツリー全体を開くには、<Ctrl> キーを押しながらペアレントノードをクリックします。［ブックマーク］フレームの表示レベルを切り替えるには、<F5> キーを押します。
- ガイドのインデックスの各項目は、クリックできるリンクです。クリックすると、説明テキストに直接移動します。
- Web サイトのアドレス (URI) が表示されている場合は、アドレスをクリックすると、通常は、そのサイトがブラウザに表示されます。URI は、青色でない場合や下線が引かれていない場合でも、一般的にホットリンクです。ホットリンクであるかどうかについては、マウスを URI の上に移動することによって確認できます。ホットリンクである場合は、カーソルが矢印からの指の形に変わります。
- 章内および章間の相互参照もクリックできます。
- ガイド内の特定のページに移動するには、<Ctrl>+<N> キーを使用します。ページ番号の入力を求めるダイアログボックスが表示されます。

- PDF テキストは、通常の方法 (<Shift> キーを押しながらドラッグしてテキストを選択し、<Ctrl>+<C> キーを使用) でクリックボードにコピーできます。多くのプログラムでは、クリップボードのコンテンツを、特定のフォーマット機能を保持したまま、RTF (リッチテキスト形式) として貼り付け (または「形式を選択して貼り付け」) することができます。複数の PDF ページにまたがっている大きなテキスト部分を選択するには、まず、Acrobat ウィンドウの最下部にあるボタンバーで、「［連続］ページモード」アイコンをクリックします。次に、<Shift> キーを押しながらドラッグしてテキストを選択し (または、<Ctrl>+<A> キーを使用してすべてを選択し)、コピーします。

1 exteNd Composer へようこそ

Web サービスによって、企業が情報を交換したり、ビジネストランザクションを実行したりする方法が根本的に変わりつつあります。「Web サービス」という用語には多くの意味が考えられますが、どのような定義であれ、ビジネスパートナー (以前であれば相互に作業できなかったパートナーを含む) の間で Web によって可能になる操作を目的とした、さまざまな新しい技術、標準、および方法が、この用語の概念の中に含まれるという点では、ほとんどのアナリストの意見が一致します。

オンライントランザクションに対する Web サービスという手段は、次の観点から必要です。

- インターネットを通じた、リアルタイムでの市場へのグローバルなアクセス。
- さまざまなパートナー間のシームレスな相互運用性。これは、明確に定義された、プラットフォームに依存しないインタフェース (XML、SOAP、および WSDL を使用) によって可能になります。標準のインタフェースを使用することで、参加企業はお互いの実装の詳細を把握する必要がなくなるため、B2B アプリケーションの開発が大幅に簡略化されます。
- UDDI ベースの Web サービスプロバイダの発見。これにより、新しいビジネスパートナーの発見およびパートナーへのアクセスが可能になります。
- ジャストインタイムでの統合。Web サービスベースのトランザクションのエンドポイントおよびプロトコルは、ランタイム時に動的にバインドされます (またはバインドするよう設定できます)。
- レガシーシステムのより容易な Web 対応。これは、データ交換に XML を使用し、サービスリクエストを SOAP エンベロープ内にラップすることによって可能になります。
- 開発サイクルの短縮。これは、さまざまなチームが同時に開発できる、適切にカプセル化され細分化された再利用可能なコンポーネントとリソースによって可能になります。
- SOAP を使用した RPC 交換のより容易な実装 (CORBA と比較した場合)。

このような明白な利点はありますが、Web サービスベースのビジネスの構築は、たいへんな業務です。成功させるためには、企業は、XML や SOAP などの重要な標準を十分に活用した、洗練された Web 対応 IT アーキテクチャを構築すると同時に、製品化までの期間に対する要求が厳しさを増す中で、パフォーマンス、セキュリティ、スケーラビリティ、および信頼性の厳密な要件を満たす必要があります。

Web サービスベースのアーキテクチャへの移行を成功させる上で最も重要な要因は、世界的な水準の開発ツールが使用可能かどうかという点です。このようなツールは、理想的には次のようなものである必要があります。

1. Web サービス開発用に新しく専用に作成されている。
2. 無関係の構成要素の寄せ集めではなく、設計上密接に統合されている。
3. ビジネスアナリストからシステム管理者そしてソフトウェアエンジニアまで、さまざまなチームのユーザが同時開発環境でただちに生産的に作業できるよう、「習得と使用が容易」である。
4. すべての重要な Web サービス技術に関して、標準に完全に対応している。このような技術には、XML (スキーマ、XPath、および XSL を含む)、SOAP、WSDL、UDDI、セキュリティ / 証明書、HTTPS などが含まれます。
5. CICS、3270/5250 端末データストリーム、およびリレーショナルデータベースシステムに関する強力な統合アプリケーションを短期間で作成できるよう、レガシーシステムに対応している。
6. 特定のベンダのアプリケーションサーバだけでなく、さまざまなアプリケーションサーバでの開発を対象とし、それらのアプリケーションサーバに直接アプリケーションを配備できる。
7. さらに、プログラミング言語が Java の場合、IDE (統合開発環境) が完全に J2EE に対応している。つまり、IDE では、EJB、サーブレット、JSP ベースの開発に加え、JAR/EAR/WAR パッケージなども「認識」し、これらを中心にしたさまざまな便利な機能を提供する必要があります。

Novell の exteNd 製品シリーズは、これらの条件をすべて満たしています。

Novell exteNd ファミリ

Novell exteNd は、J2EE (Java) アプリケーションサーバプラットフォーム上で Web サービスオブジェクトを迅速に開発するための Web サービス開発製品のファミリです。

exteNd スイートは、密接に統合された次の 3 つのコンポーネントで構成されます。

- **exteNd Workbench** – J2EE (Java 2 Platform, Enterprise Edition)向けの Web サービスとアプリケーションの迅速な設計、開発、パッケージ化、および配備のために特別に作成された統合開発環境 (IDE)。

- **exteNd Director** – ワイヤレス技術 (WAP)、Web ブラウザ、電子メール、キオスク、または「フロントエンド」と呼びうるその他のあらゆる機能を通じてユーザに Web サービス (および関連コンテンツ) を配信するために必要なコンテンツ管理、ワークフロー管理、および e ポータル機能を含む「配信層」開発環境。

- **exteNd Composer** – Web サービスおよびXMLに対応したバックエンド統合アプリケーションを設計およびテストするための視覚的な開発環境。設計時 IDE およびランタイムエンジン (Composer Enterprise Server) の他に、Composer には、XML 統合アプリケーションでさまざまなデータソース (EDI、JMS、JDBC、Telnet、CICS RPC、3270/5250 端末データストリーム、および通常の HTML-over-HTTP を含む) を使用できるようにする一連の専用コネクタもあります。

すべての exteNd 製品は、J2EE 標準に厳密に準拠しているため、一般的なほとんどの Java/J2EE アプリケーションサーバ上で実行できます。exteNd Composer の場合、明確にサポートされているランタイム環境としては、Novell exteNd Application Server、IBM の WebSphere、および BEA WebLogic が含まれます。

exteNd Composer スイート

Novell exteNd Composer は、Web サービスおよび XML 統合アプリケーション (つまり、さまざまなバックエンドシステムやデータソースに接続できるアプリケーション) を迅速に設計および配備するために設計された開発 (およびランタイム) 環境です。Composer スイートは、次の製品で構成されます。

- **exteNd Composer** – Web サービスおよびXMLに対応したバックエンド統合アプリケーションの作成とデバッグのための視覚的な設計時ツール。

- **exteNd Composer Enterprise Server** – exteNd Composer で作成したアプリケーションを実行および管理するランタイムエンジン (アプリケーションサーバ上で使用)。

- **exteNd Connect ファミリ** – exteNd Composer および exteNd Server の機能を拡張し、専用のデータソース (EDI、CICS RPC、3270/5250 端末データストリーム、Telnet、JMS など) に依存するシステムの XML 対応を可能にする個々のアドイン製品 (リレーショナルデータベースとの通信を可能にする exteNd Composer JDBC Connect は、コアの Composer インストールスイートにバンドルされています。その他の Connect 製品は個別に購入できます)。

すべての exteNd Composer 製品は、Novell exteNd Application Server、IBM の WebSphere、および BEA WebLogic で動作することが認証されており、Windows NT および Windows 2000 から、Solaris、AIX、および HP-UX にいたるまで、さまざまなオペレーティングシステムをサポートしています。

注記： exteNd Composer Enterprise Server および exteNd Composer Enterprise Connect の各製品には、専用のマニュアルがあります。このガイドでは、設計時製品 (つまり、exteNd Composer) のみを説明します。

Composer とは

exteNd 製品ファミリの設計時アプリケーションは、exteNd Composer です。exteNd Composer には、迅速なアプリケーション開発のために、マウス操作だけで使用できる柔軟で直感的な GUI (グラフィカルユーザインタフェース) が備わっており、最小限の時間で強固な XML 統合アプリケーションを作成するための強力なツールがビジネスアナリストやアプリケーション開発者に提供されます。

Composer では、特に次の機能を備えています。

- オンボードの XML エディタ。
- ドラッグアンドドロップに対応した XML マッピングエンジン。スキーマ、DTD、XSL、XPath、および DOM レベル 2 をサポートしています。
- Java プログラミングに関する広範な専門知識なしに、標準の制御フロー構成要素、エラートラップ、ログなどを実装するための直感的で視覚的な編集環境。
- リアルタイムステップイン / ステップオーバのデバッグおよびアニメーション。開発環境内からリアルタイムでアプリケーションをテストできます。
- マルチドキュメントインタフェース (MDI)。一度に複数のドキュメントやコンポーネントで作業できます。
- リアルタイムレジストリ参照。UDDI レジストリを使用した WSDL の公開 / 取得がサポートされています。
- WSDL の自動生成および「スマート」編集。
- ビジネス論理やデータ操作に関して細かい制御を必要とするユーザ用の、組み込まれた ECMAScript サポート (オンボードのカスタムスクリプトエディタを含む)。
- 3270、5250、Telnet、JMS、JDBC、CICS RPC、EDI など用の exteNd Composer Connect アドインを通じたバックエンドシステムとの接続。
- プロジェクトを直接アプリケーションサーバに配備するための Deployment Wizard (Novell、WebSphere、または WebLogic アプリケーションサーバに対するコンテキスト駆動型のカスタマイズ)。
- exteNd Workbench との統合 (いつでも Workbench に切り替えて、Composer プロジェクトを任意の WAR ファイルにインポートすることが可能)。

Composer を使用できるユーザ

Novell exteNd Composer (または、簡潔に Composer) は、ビジネスアナリスト、IT マネージャ、Java 開発者、および Web サービスの設計と開発に携わる必要があるその他のユーザを対象としています。

Composer は、さまざまなスキルレベルのユーザが利用できます。たとえば、プログラミング経験がほとんどまたはまったくないビジネスアナリストでも、Composer のドラッグアンドドロップによる XML マッピング機能を使用すれば、複雑なデータ変換を短時間でマスターすることができます。一方、Java 開発者は、Composer を使用して、ECMAScript、SQL、COBOL 統合、カスタム Java オブジェクト、または専用のパッケージ (WAR/EAR/JAR ファイル)、あるいはこれらすべてに大きく依存する可能性がある、より洗練されたサービスやコンポーネントを開発できます。

Composer の非常に視覚的な GUI、強力なアニメーションツールとデバッグツール、および「例外によるプログラミング」という開発モデルにより、高レベルの開発から低レベルの開発への快適な移行を実現できます。さまざまなタイプのユーザがどのように Composer の機能を使用できるかは、次の図のとおりです。

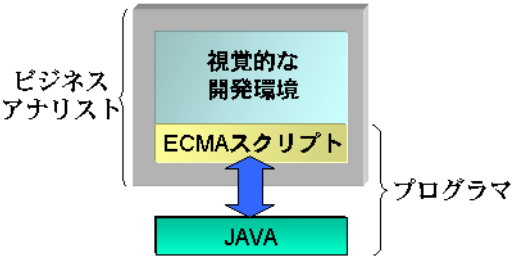


図 1-1

コンポーネントおよびサービス

Composer によるアプリケーション設計は、「コンポーネント」と「サービス」という 2 つの主な処理構成要素を含む「アクションモデル」アーキテクチャに基づきます。コンポーネントは、ユーザの設定および具体的な統合のニーズによって多かれ少なかれ細分化された、作業の実行可能な単位です。たとえば、標準的な JDBC コンポーネントでは、着信した XML 要求ドキュメントを検証して、ドキュメントの重要な情報を SQL 問い合わせにマップし、SQL 結果セットを XML 応答ドキュメントにマップします。データ取得機能は、すべてコンポーネントレベルでカプセル化できます。

一方、サービスでは、通常、コンポーネントを「結合」して、コンポーネント間のデータの流れを調整します。代表的なサービスには、入力 XML ドキュメントの受信、洗練されたドキュメントマッピング / 変換の実行、バックエンドデータソースからの情報の収集、メインフレームおよび AS/400 上でのトランザクションの実行、エラー条件の処理、電子メールまたは JMS による状況依存型の通知の送信、または元のリクエスタへの XML 応答ドキュメント (1 つまたは複数) の返信、あるいはこれらすべてを行う一連のコンポーネントが含まれる場合があります。サービスのタスクを独立したコンポーネントに分割することで、テスト、デバッグ、コードの保守、およびコードの再利用に関して重要な利点を実現できます。

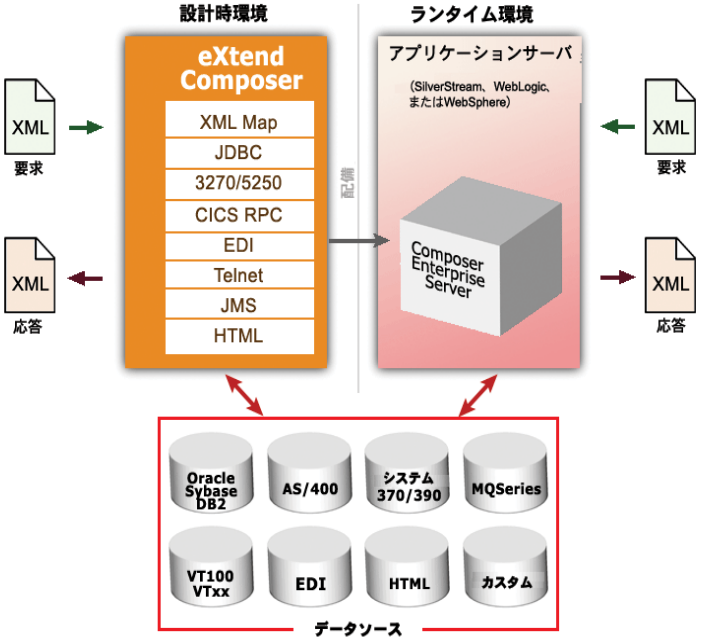


図 1-2

通常は、Composer を使用して、データの取得と変換に関する B2B 統合タスクを、XML 技術 (オプションとして、SOAP および Web サービス技術を含む) で実行するコンポーネントおよびサービスを作成します。これらのコンポーネントやサービスは、J2EE アプリケーションサーバ環境に配備します。この環境では、exteNd Composer Enterprise Server (Composer のランタイムの部分) によって、コンポーネントやサービスの実行が実現されます。

Composer 自体と同様に、exteNd の各 Connect 製品には、企業ビジネスシステムの XML 対応と、それらのアプリケーションの操作のランタイム管理を両方とも提供するよう設計された、設計時およびランタイムの構成要素があります。

全体として、exteNd 製品スイートでは、次の機能が有用です。

- 異種ドキュメントマッピング

  XML に対応した任意のアプリケーションから XML 形式のデータを受信します。続いて、受信したデータを異なる XML ドキュメントタイプにマップまたは変換し、生成された XML 形式のデータを、XML に対応したその他のアプリケーションに送信します。

- 統合制御処理

  イベントと、構成要素、条件、エラー、ログ、カスタム ECMAScript 関数などの繰り返しとグループ化を含む、統合アプリケーションに関連付けられている処理動作をすべて管理します。

- 端末データインタフェースを使用したホストアプリケーションの XML 対応

  XML ドキュメントからデータを読み込み、そのデータを端末トランザクションに直接マップ、変換、および転送するか、あるいは端末トランザクションの「結果」からデータを直接読み込んで、そのデータを XML ドキュメントにマップします。一般的な端末トランザクションタイプの例は、3270、5250、および VT100 です。端末データインタフェースの XML 対応は、exteNd の Connect 製品を通じて提供されます。

- トランザクションベースおよびメッセージベースのプログラミングインタフェースを使用した、ホストアプリケーションの XML 対応

  XML ドキュメントからデータを読み込み、そのデータを COBOL/CICS Procedure Division に直接マップ、変換、または転送するか (例 : COMMAREA やメッセージキューを使用)、あるいは COBOL Procedure Division からデータを読み込んで、そのデータを XML ドキュメントにマップします。トランザクションベースおよびメッセージベースのプログラミングインタフェース (JMS (Java Messaging Service) インタフェースなど) の XML 対応は、exteNd の Connect 製品を通じて提供されます。

- JDBC を使用したデータベースの XML 対応

  XML ドキュメントからデータを読み込み、そのデータをデータベーストランザクションに直接マップ、変換、または転送するか (JDBC を使用)、あるいはデータベーストランザクションの「結果」からデータを直接読み込んで、そのデータを XML ドキュメントにマップします。JDBC データソースの XML 対応は exteNd Composer の JDBC Connect を通じて提供され、バッチ処理およびプリペアドステートメントが完全にサポートされています。

- exteNd Composer EDI Connect を使用した電子ドキュメント交換の XML 対応

  交換時またはドキュメントレベルで EDI ファイル (ANSI X12 または Edifact) を読み込み、EDI ペイロードを XML に解析します。データが XML に変換された後、(その他の exteNd コネクタを使用して) バックエンド統合プロセスのタイプを任意の数だけ利用できます。逆に、他のコネクタによって XML 形式で取得されたデータは、EDI に変換して、HTTP または VAN を通じて転送できます。

- Composer の SAP Connect を使用した SAP 対応

  コンポーネント内で SAP 機能を実行します。

- Java の XML 対応

  Java オブジェクトを開発して、統合アプリケーションに直接組み込みます。XML データは、これらのオブジェクトに渡して Java で処理し、Composer に送信して、さらに操作やマッピングを行うことができます。また、Java オブジェクトからは、exteNd Composer Enterprise Server のフレームワーク API にアクセスして、カスタムドキュメント管理やイベント処理などの高度な操作を実行できます。

- SOAP 対応

  XML テンプレートおよび Map コンポーネントを使用して、exteNd Composer で直接 Simple Object Access Protocol ベースのサービスを作成します。

- レガシーシステムの Web 対応

  定期的にアップデートされるレガシーデータベースから製品の説明、画像、価格、および在庫情報を取得して、Web ブラウザに表示します。オペレーショナルシステムから最新の製品情報を取得したり、データベースから静的情報 (画像など) を取得したりして、個別の情報ソースの情報をマージしてから、ユーザに対して表示できます。これにより、内部ユーザと外部ユーザの両方に同じ最新情報が提供されます。

- Web ページの「スクリーンスクレーピング」を使用したホストアプリケーションの対応

  exteNd HTML Connect を使用すると、任意の Web URL からデータを取得し、そのデータの一部を XML DOM または XHTML にマップしたり変換したりできます。

これらの機能の他に、Composer の *Process Manager* を使用して、Composer のサービスを「相互接続」し、自動化されたビジネスプロセスを作成することもできます。Process Manager では、設計時のアニメーション / デバッグ機能を使用して、洗練された、長時間実行されるプロセス (ワークフロー) を設計、配備、および管理でき、プロセスの配備前に完全なプロセスをテストできます。Process Manager 設計環境は Composer の内部で動作するため (異なる Connect 製品用の異なるコンポーネントエディタでもすべて同様)、1 つの環境内ですべての設計作業を行うことができます。

Composer で作成できるアプリケーションの種類

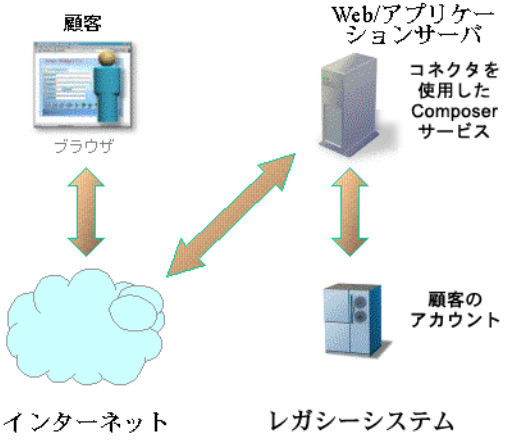
Composer では、アプリケーションのタイプを任意の数だけ作成できますが、通常は、サーブレット、EJB、カスタム Java オブジェクト、または JMS メッセージキュー上の着信メッセージによってトリガされる XML 統合アプリケーションを作成します。場合によっては、アプリケーションを単にアプリケーションサーバ上でローカルに使用して、ローカルプロセス全体にサービスを提供し、外部環境には公開しないこともあります。また、別の場合には、URI を通じてインターネットユーザにアプリケーションを公開することもあります。「外部に公開した」アプリケーションとのインタフェースには、SOAP または WSDL が含まれたり含まれなかったりします。

一般的に、Composer では、データ入力とデータ出力に XML を使用する任意の種類のアプリケーションを実装できます。

Composer 用の JMS Connect アドインを購入した場合、入力および出力に対して「メッセージ」を使用するサービスを作成することもできます。

注記： (IBM の MQSeries のようなメッセージ指向ミドルウェアに関する) メッセージングは、それ自体で強力なデータ共有の例で、XML 以外のペイロードを使用できます。Composer と JMS Connect を使用すると、入力にはメッセージ、出力には XML を使用するアプリケーション、入力側では XML、出力にはメッセージを使用するアプリケーション、または XML-in/XML-out アプリケーション内でメッセージングを使用するアプリケーションなどを作成できます。

次に示す簡単な例では、XML と Composer を使用して、購入者と供給者が各ビジネスシステムをインターネット上で接続しています。



Composer を使用すると、次の 1 つまたは複数のタイプのアプリケーションを作成できます。

1 「**内部アプリケーション統合サービス**」。さまざまなソースのデータを交換する多くのアプリケーションを持っている場合があります。たとえば、Oracle の財務アプリケーション、SAP の製造アプリケーション、および自社開発した受注処理アプリケーションを相互に接続したいとします。Composer は、このような目的の実現に役立ちます。

2 「**外部 Web サービスアプリケーション**」。サービスを Web 上で取引パートナー (またはその他のユーザ) に公開しなければならない場合があります。Composer を使用すると、SOAP サービスおよび WSDL ベースの Web サービスを短期間で簡単に構築できます。Composer でサービスを設計すると、サービスの WSDL が Composer によって実際に自動生成されます (さらに、必要に応じて、UDDI レジストリへの公開までも実行されます)。

3 「**データウェアハウジングアプリケーション**」。Composer の重要な機能は異種ソースからのデータをマップすることなので、Composer は、データマイニング技術およびウェアハウジング技術と効果的に連携します。

自動化されたビジネスプロセス管理 (ワークフロー)

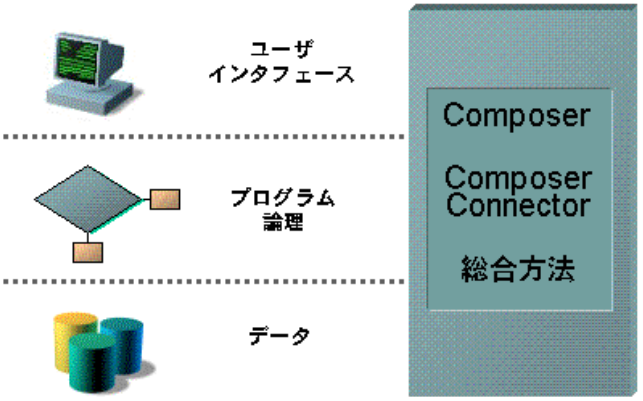
ビジネスアプリケーションを Web サービスとしてパッケージ化すると、ワークフローを自動化するための新しい機会が生まれます。Web Services Flow Language (ワークフロー自動化用の複数の新しい標準の 1 つ) によって、次世代の BPM ソフトウェアが作成される標準が提供されます。この次世代技術の基礎は、「Web サービス」上に構築されるワークフローです。SOAP と XML は、将来のワークフローシステムの重要な技術的基盤となります。

Composer は、自動化されたワークフローに関する新しい標準や技術に従って動作します。次世代のワークフローエンジンでは、タイムアウトや再試行などの BPM の概念、サービス間の条件付きリンク、並列実行に関する個々のサービス間の制御フローなどに依存して、Web サービス (外部または内部) を「接続」し、洗練された、長時間実行されるアプリケーションを作成できます。また、Web サービスに関する複雑な処理が実現可能になります(実現可能な処理の一部については、「RosettaNet Partner Interface Processes」で説明されています)。Composer は、WSFL 対応アプリケーションを作成する場合に便利なツールです。

exteNd Composer Enterprise Connect 製品シリーズについて

exteNd は、単純なハブ & スポークアーキテクチャに基づいて構築されています。ハブは、XML ドキュメントを受け付けてドキュメントを処理し、XML ドキュメントを返す、強力な XML 変換エンジンです。スポーク (つまり、Connect 製品) は、ネイティブでは XML 対応でないデータのソースを「XML に対応させる」プラグインモジュールで、データをハブに送信して XML として処理します。これらのデータソースには、レガシー COBOL/VSAM で管理されている情報から、HTML ページに対するメッセージキューまで何でも使用できます。

さまざまな Connect 製品は、情報ソースを XML に対応させるために各製品で使用されている統合方法に従って分類できます。統合方法は、インターネットベースのコンピュータアーキテクチャに対する現在のシステム設計において使用される主要な区分を反映したものです。exteNd では、B2Bi のニーズに応じて、ユーザインタフェースレベル、プログラム論理レベル、またはデータレベルでビジネスシステムを統合できます。



JDBC (Composer に付属するコアの Connect) に加え、exteNd Composer では、次の領域において Connect 製品が有用です。

- **JMS** — Java Message Service 標準を使用する Java ベースのメッセージング。この Connect 製品は、Composer アプリケーションと任意の JMS 対応メッセージングシステムとの間の接続を提供します。
- **3270** および **5250** — 最も一般的な 2 つの端末データストリームタイプとのシームレスな接続。
- **CICS RPC** — CICS を使用してリモートプロシージャコールを通じて COBOL システムを操作する透過的な機能。
- **Telnet** — Telnet データストリームをスクリーンスクレープおよび操作します。
- **HTML** — Web ページのスクリーンスクレーピング、または HTML データの XML データへの再マッピング、あるいはその両方を行います。

- **EDI** — EDI に対応した XML 統合アプリケーションを作成します。

注記： このガイドでは、exteNd Composer の基本的な機能を説明しています。各 Connect を追加すると、Composer で使用可能な機能が増えます。これらの追加機能は、各 Connect に付属する別のユーザガイドで説明されています。

Connect 製品をインストールすると、Composer の GUI が次の項目でアップデートされます。

- 新しい xObject カテゴリ
- 新しい Connect エディタ (コンポーネントエディタと呼びます)
- 新しい接続タイプ
- 新しいアクションタイプ
- 新しいメニュー項目、および関連するダイアログボックスとウィザード

これらの拡張や追加は Connect のインストールとともに自動で行われ、既存の Composer 環境にシームレスに統合されます。

ライセンスのアップデート

Composer または Connect 製品に関連付けられているライセンス文字列をアップデートする必要が生じた場合は、**UpdateLicense.bat** ファイル (**exteNdComposer\bin** ディレクトリにあります) を使用してこれを実現できます。コマンドラインから、次のコマンドを実行します。

```
updateLicense product newLicense [Composer/Server]
```

ここで、*product* には (ライセンスをアップデートする) 特定の製品の名前を指定し、*newLicense* にはライセンス文字列を指定します。最後の引数 (*Composer* または *Server* の一方) は、目的の製品の設計時バージョンまたはランタイムバージョンのどちらをアップデートするかを指定します。

インストールされている製品のリストは、次のコマンドを実行して参照できます。

updateLicense -L

さらに詳しいヘルプの参照先

多くの場合、Composer を理解するための最適な方法は、実際に動作している状態で確認することです。Composer には、実用的なビジネス業務がアプリケーションによってどのように処理されるかを順に確認できる、完全に動作するプロジェクト (チュートリアルと呼びます) が含まれています。このプロジェクトファイルは **Tutorial.spf** で、メインの exteNd Composer のインストールフォルダの下層にある［**Samples**］フォルダに、その他のサンプルファイルとともにあります (65 ページ「プロジェクトを開く」を参照)。

2 アプリケーションの計画

XML 統合アプリケーションとは

XML 統合アプリケーションは、データ転送媒体として XML を使用することによって特定のビジネスタスクを実行する 1 つまたは複数の関連サービスの集合です。これらのサービスでは、バックエンドシステムの情報ソースにアクセスしたり、システムからシステムに移動する際に情報を転送したり、1 つまたは複数の転送プロトコルを介してネットワークで情報を配信したり、あるいはこれらすべてを行わなければならない場合があります。サービスは、URI で Web サービスとして開示されることもあれば、完全に内部で使用されることもあり、また、他のサービスによって呼び出される場合もあります。

Novell exteNd Composer を使用すると、Web サービスとして配備できる強力な XML 統合アプリケーションを作成することができます。Composer で作成したアプリケーションは、さまざまなバックエンドシステムが、通信プロトコル、ファイル形式、またはオペレーティングシステム、あるいはこれらすべてに関して異なっていても、効率的に XML に対応したデータソースであるこれらのシステムを結び付けることができます。

Composer サービスには、Java または J2EE 技術で処理できるあらゆる種類のビジネス論理を実行しながら、XML コンテンツをマップおよび転送するコンポーネントや、その他の操作 (電子メールの送信など) を含むことができます。

Composer アプリケーションに接続できる、異なるタイプのバックエンドシステムの数は、インストールした Composer Connect 製品の数によって異なります。Composer のコアインストールには、データベースシステムに接続するための JDBC Connect が含まれています。他の Connect 製品では、3270/5250 システムでのデータの交換、CICS RPC 操作の利用、JMS メッセージングの使用、Telnet セッションの確立などを実行できます。また、EDI データの使用または SAP 関数の実行、あるいはその両方を行うことも可能です。

Composer には、統合アプリケーションを作成し、強力でインタラクティブな「ステップスルー型」のデバッグ機能を使用して設計時にそのアプリケーションをテストするための、直感的で視覚的なインタフェースがあります。単純なドラッグアンドドロップ操作によって、Java コードを 1 行も記述しなくても、非常に洗練された XML 統合アプリケーションを短時間で作成できます。完了すると、IBM WebSphere や BEA Weblogic などの主要な J2EE アプリケーションサーバに、アプリケーションを迅速に配備できます。

Composer でアプリケーションを設計および作成する方法

Composer を使用する場合は、あらゆるプロジェクトを開始する場合と同様に、要件を把握して、この要件を満たすために使用できる構成要素を理解することから始めます。

Composer で使用する構成要素は、次のとおりです。

- サービス
- コンポーネント
- リソース
- テンプレート

コンポーネントは、作業の小さな単位を実装したものと考えることができ、これらが集約されてサービスとなります。リソースは、1 つまたは複数のコンポーネントで作業を行うために実行時に必要となる可能性のある XML スキーマ、カスタムスクリプトライブラリ、および接続プロファイルなどのことです。テンプレートは、通常は、コンポーネントやサービスが必要とする XML スタブドキュメントです。

最大限の再利用を実現できるよう、コンポーネントの設計時および作成時には注意が必要です。たとえば、共通 XML ドキュメントを使用するコンポーネントを作成して、レガシーデータソースの情報にアクセスし、各リクエストに対してそのコンポーネントを呼び出すことができます。コンポーネントは、他のコンポーネントでコンポーネントごとに同じことを実行しなくてすむように、着信リクエストを特定の形式ニーズに対してあらかじめ処理するよう設計することができます。

xObject とは

「xObject」という単語は、このガイド全体で頻繁に使用されています。xObject は単に、Composer で作成されたサービス、コンポーネント、またはリソースのことです。サービス、コンポーネント、またはリソースで使用されるデータや命令は、すべて XML 形式でディスクに保持されます。Composer では、保持されたメタデータを介して、対応するランタイムオブジェクトが作成されます。このように作成されたオブジェクトが、xObject です。

xObject「自体」の低レベルの内部については、特に考慮する必要はありません。そうした処理は、Composer がすべて行うためです。用語的な観点からは、xObject は、Composer アプリケーションを構成する、XML に保存可能なオブジェクトと考えることができます。

サービスとは

Composer の「サービス」は、作業の論理単位を実行するために設計された、1 つまたは複数の「コンポーネント」を呼び出す xObject です。サービスでは、1 つの XML ドキュメントを入力として受け付け、XML データで動作するコンポーネントを使用し、1 つの出力 XML ドキュメントを生成します。また、サービスでは、XML ドキュメントレベルでデータソース間のデータのマッピング、変換、または転送も行います。サービスは、企業のスケーラブルなアプリケーションサーバ環境に統合される、ランタイム時に配備できる単位です (配備方法の詳細については、『*exteNd Server ガイド*』を参照してください)。サービスでは、他のサービスやコンポーネントを実行することもできます。作成できるサービスの例は、次のとおりです。

- XML 要求に基づいた取引パートナーへのステータス情報の送信
- Web ブラウザのリクエストに応答したレガシーデータソースからのデータの取得
- 内部データソース間での情報の交換

コンポーネントとは

Composer の「コンポーネント」は、XML ドキュメント要素の処理または非 XML データソースとの通信、あるいはその両方を行うための命令やアクションのセットです。コンポーネントでは、1 つまたは複数の XML ドキュメントを入力として受け付け、要素レベルでアクティビティを実行し、1 つまたは複数の XML ドキュメントを生成します。異なるタイプの単純または複雑なコンポーネントを作成して互いにリンクさせると、完全なビジネス操作を実行できます。また、コンポーネントでは、XML 要素レベルで XML ドキュメント間のデータのマッピング、変換、または転送も行います。さらに、XML ドキュメントと外部データソース (ライブ 3270 トランザクションや SQL データベースなど) 間でデータを移動することもできます。コンポーネントでは、他のコンポーネントやサービスを実行することもできます。

コンポーネントは、共通のプロセスがサービス間で共有されるように、個々のプロセスを実行するよう設計する必要があります。コンポーネントの例は、次のとおりです。

- 共通の標準への入力リクエストのマッピング
- 共通の標準に基づいたリレーショナルデータベースへのアクセス
- ある標準から別の標準への XML ドキュメントの変換

リソースとは

後に説明するように、コンポーネントとサービスには、XML ドキュメント内でデータのマッピング、変換、および転送を行うアクションモデルが含まれています。ただし、必要な操作が、アクションモデルの機能を超えるほど特殊で複雑な場合もあります。リソースは、このような場合に使用します。リソースには、アクションモデルは含まれず、入力 XML ドキュメントや出力 XML ドキュメントも含まれません。リソースは、コンポーネントやサービスでタスクを実行できるようにするユーティリティのように機能します。

Composer のリソースは、次のとおりです。

- 「**コードテーブル**」— コードテーブルには、一般的に使用されるビジネスコードのテーブルが格納されます (例 : 都道府県テーブル、地域テーブル)。
- 「**コードテーブルマップ**」— コードテーブルマップでは、コードテーブルに存在するあるコードセットを、別のコードセットに変換します (例 : 都道府県から地域へのマッピング)。
- 「**接続**」— 接続では、Connect トランザクション環境でデータの特定のソースとの通信を確立します (例 : JDBC 接続)。
- 「**カスタムスクリプト**」— カスタムスクリプトでは、ECMAScript または Java 言語を使用して、ユーザ開発の関数のライブラリを表します (例 : 文字列操作、Java ビジネスオブジェクトへのアクセス)。

XML テンプレートとは

XML テンプレートには、コンポーネントへの入力と出力を設計およびテストする際に便利なサンプルドキュメント、定義、およびスタイルシートが含まれています。Composer では、作成するコンポーネントの入力および出力として、XML テンプレートを使用します。XML テンプレートは設計時にのみ使用され、exteNd Server では、実際のサービス実行中にはライブドキュメントを使用することに注意してください。

Composer サービスを開発するための基本的な手順

アプリケーション開発プロセスでは、次の基本的な手順を考慮に入れる必要があります。

- Composer を使用する前にサービスを計画し、必要なサンプル XML ドキュメント、定義、およびスタイルシートを収集する
- Composer でサービスを作成およびテストする
- サーバにサービスを配備する

第 1 部 : サービスの計画 (Composer の使用前)

Composer アプリケーションは、XML ドキュメントの処理に基づきます。アプリケーションを計画する場合は、サービスを設計する前に要件を書き出し、分析することが推奨されます。必要な作業は、次のとおりです。

- 入力 / 出力要件を決定する (データの受信元、送信先、およびデータの形式)
- 既存の XML ドキュメントを収集する (取得可能な場合は、同業種グループやビジネスパートナーからの標準の XML ドキュメントも含む)
- 入力 XML ドキュメントおよび出力 XML ドキュメントを作成する (必要な場合)

要件の書き出し

要件を書き出す場合は、次の質問に回答すると役立ちます。

入力ドキュメントの外観、業界の標準に準拠するかどうか、自分で定義する必要があるかどうか

出力ドキュメントの外観、サンプルの取得場所、DTD または XSL スタイルシートが必要かどうか

必要な処理コンポーネント、XML データを変換するために XML Map コンポーネントがアプリケーションに必要かどうか、1 つまたは複数のデータベースに接続するために JDBC コンポーネントがアプリケーションに必要かどうか、すでに作成されているコンポーネントまたはリソースを再使用できるかどうか

アプリケーションに必要な追加のリソース、カスタム関数が必要かどうか

要件の分析

設計段階では、考慮に入れなければならないプロジェクトの要素が多数あります。

まず、接続する必要のあるデータソースを把握しておかなければなりません。また、必要なデータ、データの受信元、および転送モードも認識しておかなければなりません。その他にも考慮に入れる必要のある項目は、次のとおりです。

- 「**認証**」— 接続する予定のデータソースで認証情報 (ユーザ ID、パスワードなど) が必要かどうか、IT グループからの認証が必要かどうか、他の部署と連携する必要があるかどうか
- 「**セキュリティ**」— セキュリティ上の問題やファイアウォールがあるかどうか
- 「**担当者**」— データソースに接続するために特別なサポートが必要かどうか、必要な外部データソース接続を理解、作成、およびトラブルシューティングまたはデバッグするために欠かせない技術を身に付けている担当者が組織内にいるかどうか

- 「**レガシーアプリケーション**」— 端末データストリーム、リレーショナルデータベース、メッセージキューに対処する必要があるかどうか
- 「**使用可能性**」— 接続したいデータソースがいつでも使用可能かどうか、何度でも接続可能かどうか
- 「**トランザクション制御**」— ロールバック / コミット論理をアプリケーションに組み込む必要があるかどうか
- 「**XML ドキュメント**」— 同業種グループ、標準の組織、またはビジネスパートナーから取得しなければならない既存の XML ドキュメントまたはスキーマがあるかどうか
- 「**ログおよび通知**」— 進行状況の追跡やエラーの監視に関する特別なニーズがアプリケーションにあるかどうか、特定の状況が発生した場合に通知 (電子メールまたは JMS メッセージング経由) が送信される必要があるかどうか、クレジットの制限、金額、サプライチェーンなどに関する問題が発生した場合、明確に定義されたエスカレーション手順にサービスが従う必要があるかどうか
- 「**取引パートナー要件**」— 取引パートナーによってサービスが使用されるかどうか、アプリケーションの設計を制限する可能性のあるセキュリティ、監査証跡、タイムアウト / 再試行、またはその他の要件、あるいはこれらすべてが取引パートナーにあるかどうか

サービスの設計

要件を分析した後、サービスを設計します。これで、Composer の構成要素を考慮に入れて設計を開始できます (次の図を参照)。

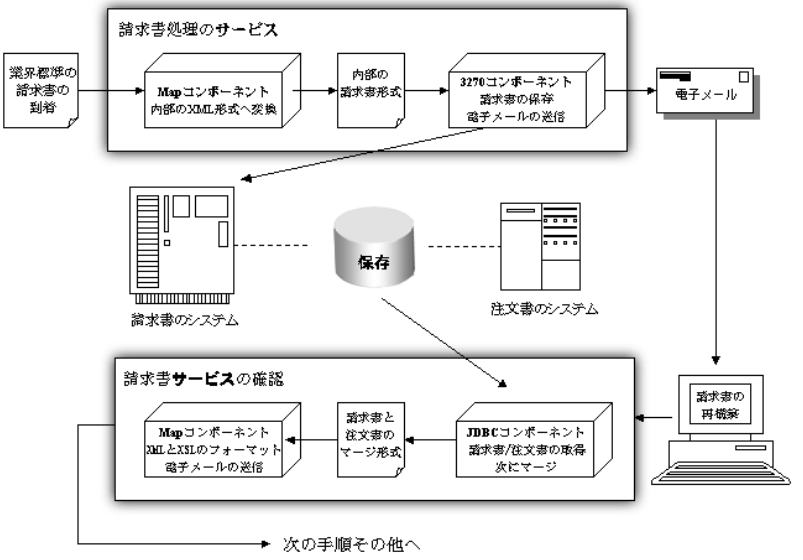


図 2-1

前に説明したように、サービスは、複数の「コンポーネント」で構成され、exteNd Server での配備の単位です。サービスの一部として作成する必要があるコンポーネントの数については、この時点ですでに把握しているはずです。たとえば、ある XML ドキュメントのデータを別の XML ドキュメントのデータにマップして、コードテーブル変換を実行する必要がある場合は、このタスクを実行するコンポーネントが必要となります。さらに、JDBC データベースへの接続を確立してデータを抽出する必要がある場合は、この作業を実現するコンポーネントも必要となります。

第 2 部 : サービスの作成

サービスを作成する際に必要な作業は、次のとおりです。

- XML テンプレートを作成する
- サービスに必要なリソース (スキーマや WSDL リソースなど) を作成する
- サービスの実行可能構成要素 (コンポーネントと呼ばれる) を作成し、サービスに固有なデータ取得および XML 変換のさまざまな段階をカプセル化する
- 構成要素を使用してサービスを作成する
- サービスをテストする
- サービスをドキュメント化する (必要な場合)

第 3 部 : サービスの配備

サービスの作成とテストは完了しました。次に、サービスをアプリケーションサーバに配備して、動作させます。Composer には、設計時のサービスを exteNd Server で実行できるようにするプロセスを順に説明する Deployment Wizard が備えられています。また、exteNd Workbench には、大きな J2EE プロジェクトのサブプロジェクトとして WAR ファイルに Composer ランタイムオブジェクトをパッケージ化するための機能も含まれています。

注記 : 配備における基本的 (しかし不可欠) な考慮事項については、このガイドの第 15 章で説明します。配備関連の問題および手順の詳細については、アプリケーションサーバに該当する『*Composer Enterprise Server ユーザガイド*』を参照してください。

サービスの実行時におけるデータの処理方法

サービスとコンポーネントでは、XML 入力 DOM および XML 出力 DOM (ドキュメントオブジェクトモデル : ここでは、XML ドキュメントのコンテンツと構造を表すメモリ内のオブジェクトを意味する) によってランタイム処理中に情報が相互に渡されます。あるコンポーネントまたはサービスの XML 出力は、別のコンポーネントまたはサービスの XML 入力になることがあります。ただし、サービスとコンポーネントでは、実際にはこのような XML ドキュメントはディスクファイルとして渡されず、ファイルの「メモリ内の」DOM イメージが渡されます。これらの DOM は、ディスクに書き込まれたり、ディスクファイルを実際に変更したりすることなく、データ統合の目標を実現するために処理中に破棄、変更、および再作成できるため、これは重要な相違点です。XML ドキュメントの DOM は、メモリにロードされたら、コンポーネントエディタのさまざまなマッピング、変換、および転送機能と、それぞれによってアクセスされるトランザクション環境によって操作できます。

3 exteNd Composer をお使いになる前に

exteNd Composer の起動

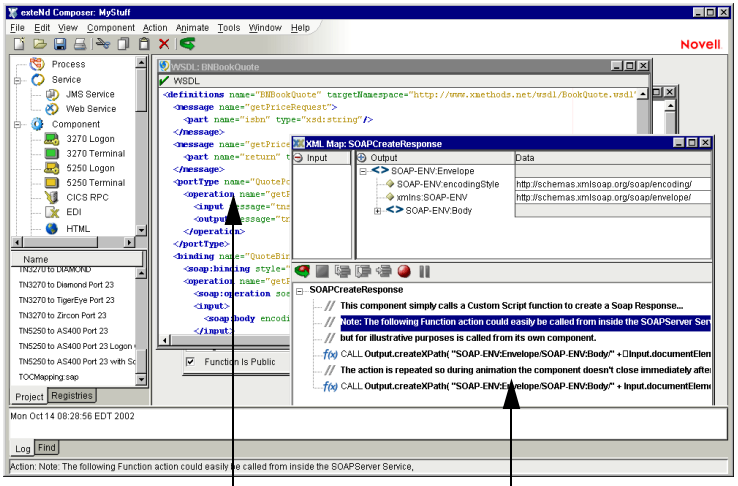
exteNd Composer は、次の方法のいずれかを使用して起動できます。

- デスクトップの［**exteNd Composer**］アイコンをクリックします。
- Windows の［**スタート**］メニューから、［**プログラム**］、［**exteNd Composer**］の順にクリックします。
- Windows エクスプローラを開き、**\exteNd\Composer\bin** ディレクトリに移動して、**XC.exe** をダブルクリックします。

exteNd Composer 環境の概要

Composer は、XML ベースの B2B 統合サービスを作成するために使用するアプリケーションです。作成したサービスは、Java アプリケーションサーバ (Novell のサーバ、または別の J2EE サーバ) に配備され、exteNd Composer Enterprise Server によって実行されます。Composer では、配備するすべてのオブジェクトを作成、整理、および準備できます。

Compser 内には、他のアプリケーションが開発環境に含まれます。このようなアプリケーションには、さまざなま Compser オブジェクトエディタ (例 : 該当するコンポーネントのタイプに関するコンポーネントエディタ)、カスタムスクリプトエディタ、および XML エディタがあります (次を参照)。



さまざまなデータソースにアクセスしたり、XML 構造やデータをマップまたは変換したりできる異なるタイプのコンポーネントを作成するには、Composer コンポーネントエディタを使用します。Composer 内からは、exteNd Developer Workbench だけでなく、そのすべての機能にもアクセスできます。

使用方法

作成するサービスが比較的単純な XML 統合サービスである場合でも、複雑な Web サービスである場合でも、作成プロセスは Composer を使用する際の次の基本手順に従います。

1 「**Composer 環境**」(この章で説明されています) について理解します。

2 「**プロジェクト**」を作成します。プロジェクトでは、作成しているアプリケーションのオブジェクトがすべて保持されます。プロジェクトは、「spf」というファイル拡張子で保存されます。

注記： 単一の .spf ファイルまたはプロジェクトには、さまざまなタイプのコンポーネントやリソースだけでなく、それらを使用するサービスも多数含めることができます。プロジェクトは、Composer から直接配備したり、Developer Workbench を使用して EAR/WAR ファイルにインポートしたりできます。

3 アプリケーションを作成してテストするのに使用するサンプルドキュメントを整理する方法を表す「**XML カテゴリ**」を作成します。

4 サンプルドキュメントを含める「**XML テンプレート**」を作成します。

5 プロジェクトに必要な「**リソース**」(例 : 接続、カスタムスクリプト、XSD または WSDL リソースなど) を作成します。

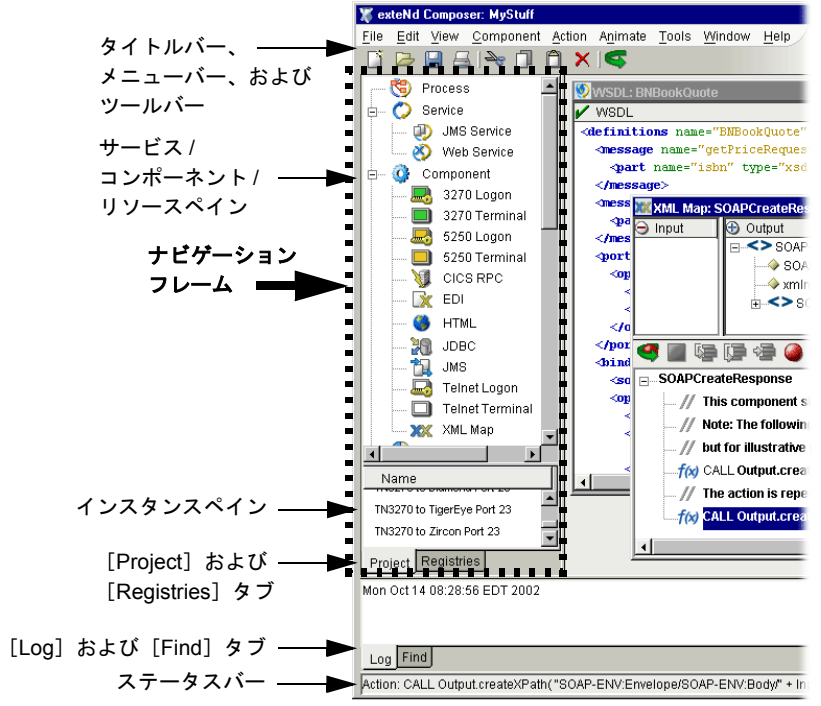
6 テンプレートおよびリソースを使用する「**コンポーネント**」を作成します。

7 コンポーネントを実行する「**サービス**」を作成します。

8 「**配備**」に対してプロジェクトを準備します。

Composer 環境について

オブジェクトを作成したり整理したりするには、Composer メインウィンドウを使用します。ウィンドウの各部分は、次の図のとおりです (ナビゲーションフレームは、点線のボックスで囲まれています)。



ナビゲーションフレーム、メッセージフレーム、およびコンテンツフレーム

Compser ウィンドウには、デフォルトで、ナビゲーションフレーム (左側)、メッセージフレーム (下部)、およびコンテンツフレーム (右側) という 3 つのフレームが表示されます。各フレームのサイズは、それぞれを区切っているセパレータバーをドラッグすることによって、他のフレームと相対的に調節できます。また、メインウィンドウのサイズも、通常の方法 (最大化、最小化、アイコン化、および引き伸ばし) で調節できます。「画面領域」を管理する際に最大限にフレキシブルにするため、ナビゲーションフレームとメッセージフレームを個々に非表示にすることもできます。ナビゲーションフレームの表示レベルは、<**Ctrl**>+<**Shift**>+<**N**> キーを使用して切り替えることができます。また、メッセージフレームの表示レベルは、<**Ctrl**>+<**Shift**>+<**O**> キーを使用して切り替えることができます。

ナビゲーションフレーム

ナビゲーションフレームには、[Project] と [Registries] という 2 つのタブが下部にあります。[Project] タブでは、カテゴリ (上のペイン) ビューおよびインスタンス (下のペイン) ビューで Composer オブジェクトを表示できます (前の図を参照)。[Registries] タブでは、UDDI タイプのレジストリの中からレジストリエントリを検索して表示することができます。この機能の詳細については、第 14 章を参照してください。

メッセージフレーム

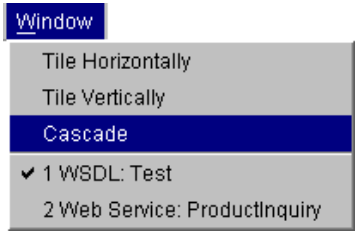
Composer ウィンドウの下部にあるメッセージフレームには、[Log] と [Find] という 2 つのタブがあります。[Log] タブでは、Composer セッション中にリアルタイムでエラーメッセージや Log アクション出力を表示できます。これにより、ログファイルを手動で開いたり、システムコンソールメッセージをチェックしたりする必要がなくなります。[Find] タブでは、検索結果を表示できます。[Find] コマンドの使用の詳細については、72 ページ「xObject またはテキストの検索」を参照してください。

コンテンツフレーム

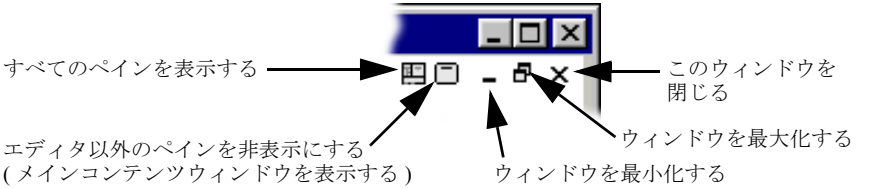
コンテンツフレーム (右上角) には、DOM ツリー、アクションモデル、およびネイティブ環境ペインを含む、コンポーネントエディタのコンテンツが表示されます。

Composer の MDI ウィンドウ環境の操作

Composer には、MDI (「マルチドキュメントインタフェース」) が備えられており、複数のエディタウィンドウを同時に開く (および表示する) ことができます (前の図を参照)。個々のウィンドウは、他のウィンドウと同様に、最小化、最大化、および閉じることができます。また、ウィンドウは、並べて表示したり、重ねて表示したり、クリックアンドドラッグで任意に配置したりすることができます。開かれている複数のウィンドウの配置を制御するには、Composer メインメニューバーの［Window］コマンドを使用できます。



さらに、メイン画面の右上角で該当するアイコンをクリックすると、エディタ以外のペイン (Composer のナビゲーションペインやメッセージペインなど) を非表示にすることもできます。これは、アクションモデルで作業する場合や、［Log］ペインやナビゲーションツリーなどを表示する必要がない場合に便利です。



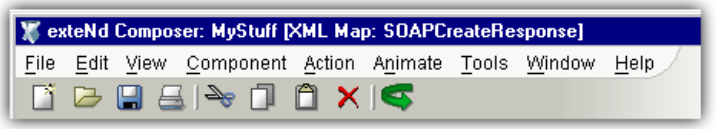
注記：すべてのペインを表示 / 非表示にするアイコンは、現在のエディタウィンドウが最大化されている場合にのみ有効になります。ただし、ナビゲーションペインとメッセージペインは、［View］メニューのコマンドを使用して、いつでも個々に表示または非表示にできます (次の説明を参照)。

エディタ以外のペインをすべて非表示にすると、現在のエディタウィンドウが表示され、(ツールバーとメニューを除く) Composer ウィンドウ全体が利用されます。これは、多数の DOM を含む複数のサブペインがアクションモデルにある場合や、カスタムスクリプトエディタで作業しており「領域」がさらに必要な場合に便利なことがあります。

［View］メニューには、Composer メインウィンドウ設定を調節するための追加のコマンド (［Navigator Tabs］と［Output Tabs］) があります。［Navigator Tabs］を選択すると (<**Ctrl**>+<**Shift**>+<**N**> キーを入力した場合も同じ)、Composer メインウィンドウのナビゲーションフレームの表示レベルが切り替えられます。また、［Output Tabs］を選択するか、あるいは <**Ctrl**>+<**Shift**>+<**O**> キーを入力すると、ウィンドウの下部にあるメッセージペインの表示レベルが切り替えられます。

タイトルバー、メニュー、ツールバー、およびステータスバーの使用

Composer では、標準の Windows メニューバーとツールバーを使用してオブジェクトを操作できます。Composer を初めて開いたときに表示されるタイトルバー、メインメニュー、およびツールバーは、次の図のとおりです。



タイトルバー

タイトルバーには、開いている現在のプロジェクトの名前が表示されます。「プロジェクト」は、まとめて開発、保守、および配備される exteNd Composer サービスの集合です。

メニュー

使用できるオプションは、次のとおりです。

表 3-1

Composer メニューコマンド	説明
［File］メニュー	
［New xObject］	新しいサービス、コンポーネント、リソース、XML テンプレート、および XML カテゴリを作成するときに使用されます。リソースには、コードテーブル、コードテーブルマップ、接続、およびカスタムスクリプトがあります。68 ページ「xObject の作成」を参照してください。
［Open xObject］	詳細ペインで選択されたオブジェクトを開きます。オブジェクトは、ダブルクリックして開くこともできます。69 ページ「xObject を開く」を参照してください。

Composer メニュー コマンド	説明
［Recent xObjects］	最近開いた xObject のリストを表示します。このリストからは、開く xObject を選択できます。
［Delete xObject］	選択したオブジェクトを Composer ウィンドウから削除し、関連付けられているすべてのファイルもディスクから削除します。オブジェクトは、選択して <**Delete**> キーを押すことにより、削除することもできます。
［New Project］	新しいプロジェクトのファイル名およびディレクトリを作成したり、現在のプロジェクトを終了したり、新しいプロジェクトを現在のプロジェクトにしたりします。
［Open Project］	既存のプロジェクトを開きます。
［Recent Projects］	最近開いたプロジェクトのリストを表示します。このリストからは、開くプロジェクトを選択できます。
［Delete Project］	選択したプロジェクトをディスクから削除します。
［Import xObject］	プロジェクトに xObject を追加します。68 ページ「xObject の作成」を参照してください。
［Deploy］	配備に対してプロジェクトを準備します。
［Properties］	選択したプロジェクトのプロパティを表示します。プロパティには、オブジェクトのヘッダ情報（名前と説明）に加えて、オブジェクトタイプに固有な情報もあります。71 ページ「xObject のプロパティの表示」を参照してください。
［Print］	選択したプロジェクトの詳細を印刷します。71 ページ「xObject のプロパティの印刷」を参照してください。
［Exit］	Composer アプリケーションを終了します。保存されていないコンポーネントが開いている場合、変更内容を保存または無視するかを尋ねるメッセージが表示されます。
［Edit］メニュー	
［Undo］	最後の操作を削除し、開いているオブジェクトをその操作を行う前の状態に戻します。［Undo］オプションは、コンポーネントエディタのアクションモデルペインでのみ使用できます。105 ページ「XML Map コンポーネントの作成」を参照してください。
［Cut］	選択したオブジェクトまたはアクションを Composer ウィンドウから削除し、Windows クリップボードに配置します。
［Copy］	選択したオブジェクトまたはアクションのコピーをWindowsクリップボードに配置します。

Composer メニュー コマンド	説明
［Paste］	Windows クリップボードのコンテンツを Composer ウィンドウにコピーします。
［Delete］	選択したオブジェクトまたはアクションを Composer ウィンドウから削除し、オブジェクトに関連付けられているファイルも削除します。
［Find］	オブジェクト内で特定の文字列の最初のインスタンスを検索します。［Find］オプションは、xObject が開いているときはいつでも使用できます。［**Tools**］メニューの［**Find**］も参照してください。
［Find Next］	［Find Text］ダイアログボックスに入力した文字列の次のインスタンスを検索します。［Find Next］オプション (<F3> キー) は、xObject が開いているときはいつでも使用できます。
［Replace］	特定の文字列を、入力した新しい文字列で置換します。［Replace］オプションは、コンポーネントエディタのアクションモデルペインでのみ使用できます。105 ページ「XML Map コンポーネントの作成」を参照してください。
［View］メニュー	
［Navigator Tabs］	Composer メインウィンドウの左側にあるナビゲーションフレームの表示を切り替えます。
［Output Tabs］	Composer メインウィンドウの下部にあるメッセージフレームの表示を切り替えます。
［Tools］メニュー	このメニューのオプションは、選択するオブジェクトのタイプによって異なります。
［Find］	名前、含まれる文字列、使用する XML テンプレート、またはコンポーネントを使用する場所によって、プロジェクト内で xObjects を検索します。
［Configuration］	プロジェクトグローバル変数を設定し、使用する XML エディタ、ログファイル、Web ブラウザ、および配備パッケージを確立する設定を管理します。プロキシサーバ設定を入力することもできます。
［Window］メニュー	開いているすべてのウィンドウを表示します。
［Help］メニュー	
［Help Topics］	Composer のオンラインヘルプを表示します。
［Novell on the Web］	World Wide Web 上のヘルプへのリンクがあるサブメニューを表示します。

Composer メニューコマンド	説明
[My Project]	プロジェクト用に作成した HTML ヘルプファイルを表示します。
[About exteNd Composer]	Composer に関するプログラムとバージョンの情報を表示します。

ツールバー

メニューオプションに加え、ツールバーには次のボタンがあります。

表 3-2

ボタン	説明
	[New] - このダイアログボックスでは、作成するコンポーネントのタイプを選択できます。
	[Open] - このダイアログボックスでは、開くコンポーネントのタイプと名前を選択できます。
	[Cut] - このボタンをクリックすると、オブジェクトが Composer ウィンドウから削除され、Windows クリックボードに配置されます。
	[Copy] - このボタンをクリックすると、選択したオブジェクのコピーが Windows クリックボードに配置されます。
	[Paste] - このボタンをクリックすると、Windows クリックボードのコンテンツが選択したオブジェクトにコピーされます。
	[Delete] - このボタンをクリックすると、選択したオブジェクトが Composer ウィンドウから削除され、(オブジェクトに) 関連付けられているファイルも削除されます。

ステータスバー

Composer ウィンドウには、メニューとツールバーのほかに、ステータスバーもあります。ステータスバーはウィンドウフレームの下部にあり、現在選択されているオブジェクトの状態を表示します。ステータスバーに「READY」と表示されていれば、オブジェクトで操作を実行できます。

Composer アイコンの概要

Composer では、異なるオブジェクトタイプを示すのにアイコンを使用します。アイコンとそれらのタイプは、次のリストのとおりです。

表 3-3

アイコン	オブジェクトタイプ
	サービスグループ
	Web サービス
	コンポーネントカテゴリ
	XML Map コンポーネント
	リソースグループ
	コードテーブル
	コードテーブルマップ
	接続
	カスタムスクリプト
	XML テンプレートカテゴリ
	XML テンプレートグループ
	XML テンプレート

ナビゲーションフレーム

Composer メインウィンドウには、左側にナビゲーションフレームがあります。ナビゲーションフレームは、下部で選択したタブに基づいて、2 つの異なるモードで使用できます。モードを制御する 2 つのタブには、それぞれ［Project］、［Registries］というラベルが付けられています。

［Project］タブ

［Project］タブを選択すると、ナビゲーションフレームには、「サービス、コンポーネント、およびリソース」のペイン (上部) と「インスタンス」ペイン (下部) が含まれます。

注記：２つのペインの相対的なサイズは、(ペインの間に表示される) 小さな水平境界線を上下にドラッグすることにより調節できます。

下部ペインのコンテンツは、上部ペインで異なる項目を選択すると変わります。たとえば、上部ペインで Web Service 項目をクリックした場合、下部ペインには、現在のプロジェクトにおける既存の Web サービスの名前が表示されます。

サービス / コンポーネント / リソースペイン

サービス / コンポーネント / リソースペインには、Composer で作成するオブジェクト (xObject としても知られる) の 4 つの主なカテゴリであるサービス、コンポーネント、リソース、およびテンプレートが含まれます。

サービス

サービスは、プロジェクトを運用システムに配備した後にアプリケーションサーバで発生する作業やビジネスパートナートランザクションの高レベルの単位を表します。これらは、作成するさまざまなコンポーネントを結合して、アプリケーションサーバ環境内に作業論理を作成するために使用されます。また、サービスは、exteNd Composer Enterprise Server によって実際に実行される、プロジェクト内のプライマリオブジェクトでもあります。サービスは、主に配備関連の問題と関係があり、受信する入力 (URL パラメータまたは XML ドキュメント)、実行をトリガするオブジェクトのタイプ (Servlet または EJB)、および返す出力 (XML または HTML ドキュメント) によって区別できます。

コンポーネント

コンポーネントは、1 つまたは複数の XML ドキュメントを入力として受け付け、入力で動作するアクションの集合を使用し、XML ドキュメントを出力として返すオブジェクトです。通常、コンポーネントはサービスによって呼び出され、アクションまたはその他のコンポーネントへの呼び出しを含むことができます。コンポーネントは、外部データリソースを XML に対応させる機能によって区別されます。基本的な XML Map コンポーネントでは、XML 対応アプリケーションを有効にできます。たとえば、JDBC コンポーネントでは、JDBC を介してリレーショナルデータベースシステムを XML に対応させることができ、3270 端末コンポーネント (3270 Connect によって個別にインストールされる) では、3270 端末データストリームを介してメインフレームトランザクションを XML に対応させることが可能です。

リソース

リソースは、専用の操作を実行する xObject です。これらは、タスクを実行できるようにするためにサービスとコンポーネントによって使用されます。リソースのタイプには、コードテーブル、コードテーブルマップ、接続、およびカスタムスクリプトがあります。

テンプレート

XML テンプレートには、コンポーネントを設計およびテストする際に便利なサンプルドキュメント、定義、およびスタイルシートが含まれています。まず、類似の XML テンプレートを含む XML カテゴリを作成します。次に、作成するコンポーネントの入力および出力として使用される XML テンプレートを作成します。

xObject の操作

オブジェクトは、［**File**］メニューの［**New xObject**］オプションを使用して、4 つの xObject カテゴリのうちの 1 つに追加できます。また、オブジェクトは、コンテキストメニュー (次を参照) の［**Delete**］オプションを使用して、カテゴリから削除することができます。xObject ペインでは、メインカテゴリの削除や追加を行うことはできません。

各カテゴリには、プラス記号またはマイナス記号が付いています。この記号は、ツリーでのアイコンの状態を示します。プラス記号が表示されている場合、クリックしてカテゴリを展開すると、該当するカテゴリの下層にあるすべてのチャイルドノードが表示されます。同様に、マイナス記号がアイコンの隣に表示されている場合は、クリックしてカテゴリを閉じると、すべてのチャイルドノードが非表示になります。

コンテキストメニューの使用

上部ペインには、ペイン内でマウスを右クリックすることによってアクセスできる独自のコンテキストメニューがあります (次の図を参照)。

コンテキストメニューを使用すると、新しい xObject の作成、xObject のインポート、および Windows クリップボードにコピーされている xObject の貼り付けを実行できます (これらのトピックについては、他の節で個別に説明されています)。

インスタンスペインについて

インスタンスペインには、特定の xObject カテゴリに属するユーザ作成のオブジェクトがすべてリストされます。上部ペインにあるアイコンをクリックすると、そのインスタンスオブジェクトが下部 (インスタンス) ペインに表示されます。

詳細ペインの表示を切り替える

1 カテゴリペインで、コンテンツを表示するアイコンを選択します。

2 ［View］メニューで、表示オプションを選択します。オプションは、［Icons］と［List］です (44 ページ「［View］メニュー」を参照)。

インスタンスペインのコンテキストメニューの使用

詳細ペインの xObject には、コンテキストメニューがあります (次を参照)。

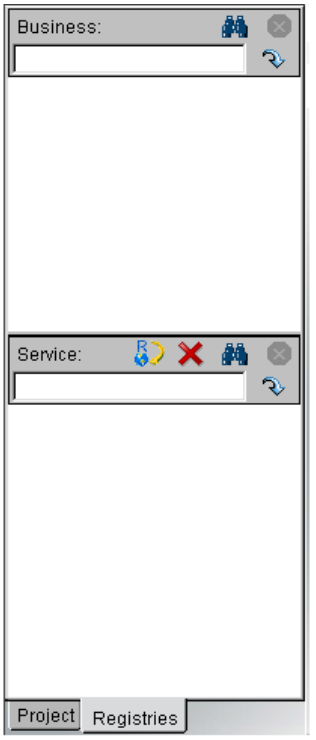
コンテキストメニューのオプションの一部は、メインメニューにもあります。コンテキストメニューに追加として含まれているオプションは、次のとおりです。

表 3-4

インスタンスペインのコンテキストメニューコマンド	説明
［Rename］	選択したオブジェクトの名前を変更します。
［Find Where Used］	コンポーネントおよび XML テンプレートの場合、この選択肢によって［Find］ダイアログボックスが開かれ、選択したオブジェクトを参照する他のオブジェクトがプロジェクト内で自動的に検索されます。
［Edit Sample］	XML テンプレート専用。選択された XML テンプレートに含まれている XML ドキュメントのリストを表示し、XML エディタでそれらを編集できます。このオプションは、XML テンプレートが選択されている場合にのみ使用できます。
［Edit DTD］	XML テンプレート専用。DTD (Document Type Definition) ファイルのリストを表示し、それらを編集できます。このオプションは、XML テンプレートが選択されている場合にのみ使用できます。

［Registries］タブ

ナビゲーションフレームの下部にある［Registries］タブを選択すると、フレームの外観は次のようになります。



［Business］および［Service］というラベルの付いた 2 つのペインがあります (この 2 つのペインの間には調節可能な境界線があります)。これらのペインは、UDDI レジストリに含まれる情報を検索および取得するために使用されます。詳細については、第 14 章を参照してください。

Composer の環境の設定

Composer は、設計時の要件を満たすよう、さまざまな方法で設定できます。変更可能なすべてのプログラム設定は、xconfig.xml という名前の XML ファイル (Composer の \bin ディレクトリにある) 内にあります。このファイルには、プロパティを次のように設定する要素が含まれています。

- XML エディタのファイルと場所
- Composer システムログのファイルと場所

- ヘルプを実行するための優先 HTML ブラウザのファイルと場所
- Composer を起動するたびにシステムログを上書きするかどうか
- システムメッセージを管理する「ログレベル」(重要度しきい値)
- Composer のトランザクションエミュレーションモード
- 独自のカスタムプロジェクト変数の名前と値
- プロキシサーバとそのアドレスを使用するかどうか
- 詳細なプロキシ設定

xconfig.xml は、必要に応じて直接編集することができますが、**xconfig.xml** を最も便利で迅速に変更する方法は、[Configuration] ダイアログボックスを使用することです。この方法は、ほとんどの場合に適用します。

[Configuration] ダイアログボックス ([Tools] メニューから選択) には、次の 4 つのタブがあります。

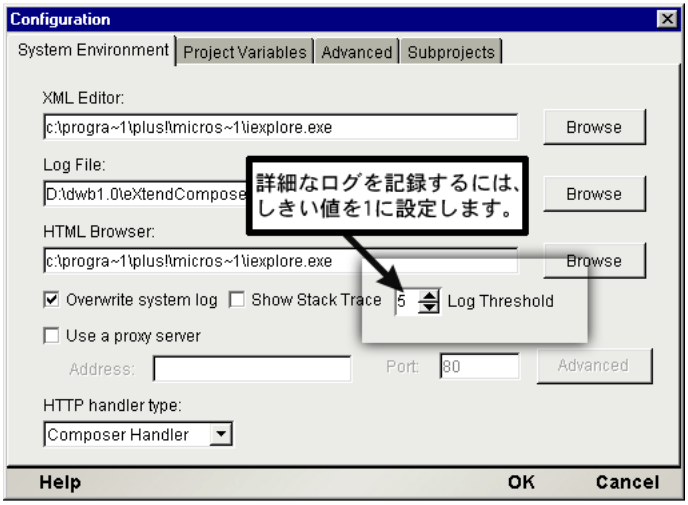
- [**System Environment**] — デフォルトの Web ブラウザや XML エディタの指定、ログファイルのパス名の設定、ログファイルの「チャットレベル」への設定、およびプロキシサーバ設定の指定を実行できます。
- [**Project Variables**] — 使用するすべてのグローバル変数の名前と初期値をプロジェクト内スコープで指定できます (これらの変数は、配備済みのプロジェクトだけでなく、設計時にも適用されます)。変数は、実際には、$PROJECT/USERCONFIG の XPath にある XML ドキュメントに保存されます (プロジェクト変数の設定の詳細については、75 ページ「プロジェクト変数の作成」を参照してください)。
- [**Advanced**] — 設計時に対してトランザクションエミュレーションモード(コンテナ管理とサーブレット /Bean 管理) を設定できます。
- [**Subprojects**] — 現在のプロジェクトで使用できる他の Composer プロジェクト (ある場合) を指定できます。この機能を使用すると、すべてのファイルの新しいコピーを作成することなく、他のプロジェクトのオブジェクトとリソースを現在のプロジェクトにインポートできます。

これらの各タブについては、この後のページで詳しく説明します。

システム環境設定

➢ 設定ファイルでシステム環境設定を指定する

1 [Tools] メニューから、[**Configuration**] を選択します。デフォルトで [System Environment] タブが表示された [Configuration] ダイアログボックスが現れます。



2 選択した XML エディタの完全修飾パスを入力するか、または［**Browse**］をクリックしてアプリケーションを検索します。

3 ［**Log File**］と［**HTML Browser**］に対して、手順 2 を繰り返します。

4 Composer を起動するたびにシステムログをクリアする場合は、［**Overwrite system log**］をオンにします。

5 ［**Display Stack Trace**］をオンにし、スタックトレースをログファイルに出力する機能を有効にします。

6 上下 (スピン) コントロールを使用して、「Composer のログレベル」を 1 から 10 までの値に設定します。この値は、コンポーネント内で実行される Log アクション「および書き出されるシステムメッセージ」を制御するしきい値です。優先度設定がこの値以上の Log アクションのみが実行されます (152 ページ「基本的なアクション」という章に記載されている節 :「ログ優先度レベル」を参照してください)。

注記 : すべてのシステムメッセージ (エラーメッセージ) を表示する場合は、この値を 10 に設定します。最小限のシステムメッセージのみを表示する場合、少ない値に設定します。

7 ［**Use a proxy server**］チェックボックスは、Composer でのコンポーネント実行時の XML Interchange アクションに関するオプションです。アクションで参照される URL がプロキシサーバを通過するよう設定する場合は、［**Use a proxy server**］チェックボックスをオンにし、プロキシサーバの「**アドレス**」と「**ポート**」を入力します。

注記： サービスが配備されると、プロキシサーバの設定は、Composer の **\bin** ディレクトリ (設計側) にある **xconfig.xml** ファイルに保存されているため、無視されます。これらの設定を「アプリケーションサーバ」環境でアクティブにするには、PROXYSERVERINFO 要素の分岐を設計側の **xconfig.xml** ファイルから切り取って、Composer Enterprise Server で使用されるサーバ側の **xconfig.xml** ファイルに貼り付ける必要があります。

8 ［**Advanced**］をクリックして、プロキシ設定を指定します。説明については、次を参照してください。

9 必要に応じて、プルダウンメニューから別の「**HTTP ハンドラタイプ**」を選択します。この設定では、Composer セッション中に使用される HTTP ストリームハンドラのタイプを指定できます。ここでの主な考慮事項は、XML Interchange アクションを使用して FTP PUT 操作を実行するかどうかということです (高度なアクションについての章を参照)。Novell ハンドラでは、FTP プロトコルによる PUT 操作と GET 操作を有効にすることができますが、他のハンドラでは、PUT をサポートできない可能性があります。

- FTP PUT 操作を使用する場合は、［**Novell Handler**］を選択します。
- XML Interchange アクションで FTP GET 操作のみを行う場合は、［**Composer Handler**］または［**JRE Default Handler**］を選択します。

各 HTTP ハンドラでは、詳細な機能やパフォーマンスが若干異なっています。そのため、特別な状況 (たとえば、JRE のデフォルトハンドラが期待どおりに動作しない場合や、HTTP セッション中にメモリ不足の状況が発生する場合など) では、この設定オプションを再度変更しなければならないことがあります。

注記： この設定に加えられた変更は、次回 exteNd Composer を起動するまで反映されません。

10 ［**OK**］をクリックし、新しい設定を保存して閉じます。

詳細なプロキシ設定の入力

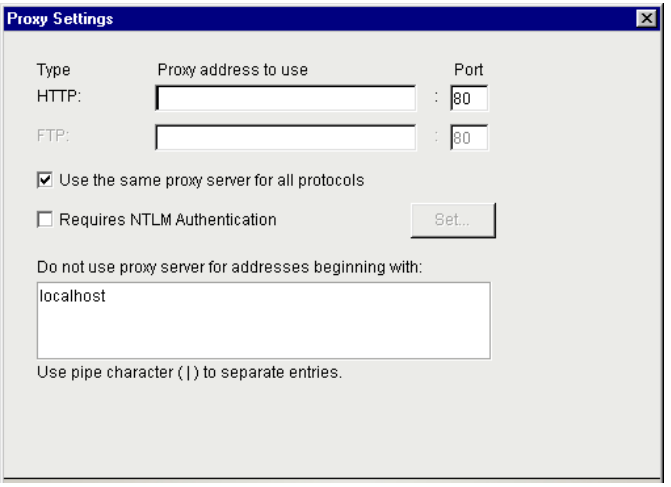
［Configuration］ダイアログボックスで［**Use a proxy server**］をオンにした場合、詳細なプロキシ設定を入力できます。これらの設定では、HTTP サーバおよび FTP サーバへの接続が確立され、特定のアドレスによってプロキシサーバが使用されないよう指定することができます。

➢ 詳細なプロキシ設定を入力する

1 ［Configuration］ダイアログボックスを開きます。

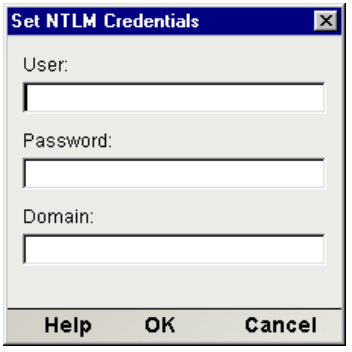
2 ［**Use a proxy server**］がオンになっていることを確認します。

3 ［**Advanced**］をクリックします。［Proxy Settings］ダイアログボックスが表示されます。



4 HTTP サーバおよび FTP サーバに対して「**アドレス**」と「**ポート**」を入力します。HTTP サーバと FTP サーバが同じ場合は、一方のサーバに対してのみ入力し、［**Use the same proxy server for all protocols**］をオンにします。

5 NTLM 認証を要求するサイトにアクセスする場合は、［**Requires NTLM Authentication**］チェックボックスをオンにします。次に、［**Set**］ボタンをクリックします。新しいダイアログボックスが表示されます。



ユーザ ID、パスワード、およびドメインに対して適切な情報を入力し、［**OK**］をクリックしてダイアログボックスを閉じます。

6 プロキシサーバを使用しないアドレスを入力します。アドレスはパイプ文字 (|) で区切ります。

7 ［**OK**］をクリックして、［Configuration］ダイアログボックスに戻ります。

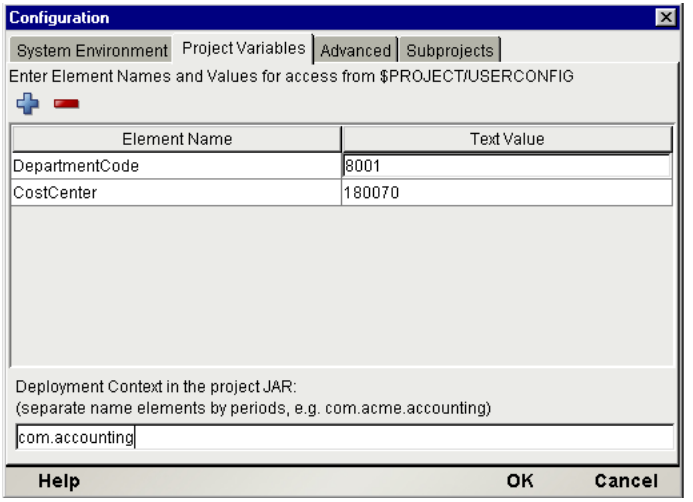
プロジェクト変数

プロジェクト変数は、プロジェクトスコープ付きのグローバル変数と考えることができます。これらは、配備時に他のプロジェクトのリソースとともに配備される独自の XML ドキュメントにそれぞれ保存されます。

➢ プロジェクト変数を作成する

1 ［Tools］メニューから、［**Configuration**］を選択します。［Configuration］ダイアログボックスが表示されます。

2 ［**Project Variables**］タブを選択します。



3 ダイアログボックスの小さなツールバーにあるプラス記号のアイコンをクリックして、変数を追加します。選択した変数を削除するには、マイナス記号のアイコンをクリックします。

4 新しい変数を追加し、［**Element Name**］に名前を、［**Text Value**］に初期値をそれぞれ入力します (プロジェクト変数には文字列値が必要です)。

5 ダイアログボックスの下部にあるテキストフィールドに、変数の配備コンテキストを入力します。これは、ピリオドで区切られたラベルであれば、いくつでも指定することができます (前の図を参照)。

注記： コンテキスト文字列には、*protected*、*default*、*int*、*new*、*try* などの Java キーワードを使用しないでください。予約語の完全なリストについては、付録 B「予約語」を参照してください。

6 ［**OK**］をクリックして、ダイアログボックスを閉じます。

注記： プロジェクト変数の詳細については、75 ページ「プロジェクト変数の作成」を参照してください。

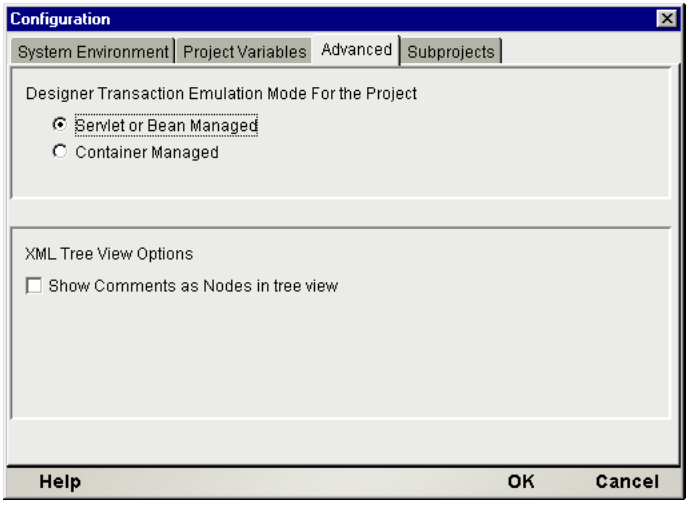
トランザクションエミュレーションモードの設定

［Configuration］ダイアログボックスの［Advanced］タブは、Composer のトランザクションエミュレーションモードをテストを目的として制御できます (これは、設計時のみにおける設定です。ランタイムトランザクションの動作は、各アクションおよび配備ウィザードで制御します)。また、このタブからは、Composer のドキュメントツリー内のツリービューに XML コメントをノードとして表示するかどうかを制御することもできます。

➢ トランザクションエミュレーションモードを設定する

1 ［Tools］メニューから、［**Configuration**］を選択します。［Configuration］ダイアログボックスが表示されます。

2 ［**Advanced**］タブをクリックします。次のペインが表示されます。



3 トランザクションエミュレーションモードを設定するには、次のラジオボタンの中から 1 つ選択します。

［**Servlet or Bean-Managed**］— 処理されるアクションモデルまたはコードブロックは、Transaction アクションによって行われる明示的な呼び出しに基づいてロールバックまたはコミットされることを意味します。

［**Container-Managed**］— トランザクションスコープが EJB コンテナによって管理されることを意味します。コンポーネント内で Begin、Commit、または Rollback コマンドが発行されると、IllegalStateException がスローされます。

4 XML ツリービューでコメントノードを表示または非表示にするには、ダイアログボックスの下半分にあるチェックボックスのオン / オフを希望に応じて切り替えます。

5 ［**OK**］をクリックして、ダイアログボックスを閉じます。

配備済みのサービスに適したトランザクション制御の詳細については、『*Composer Enterprise Server ガイド*』を参照してください。

サブプロジェクト

［Configuration］ダイアログボックスの［Subprojects］タブは、サブプロジェクト (Composer 作成の他の **.spf** プロジェクト) を現在のプロジェクトに追加 (インポート) したり、現在のプロジェクトから削除したりすることができる場所です。このトピックについては、第 4 章「プロジェクトの作成と管理」で詳しく説明されています。［Configuration］ダイアログボックスの［Subprojects］タブの使用方法については、該当する節を参照してください。

Composer オンラインヘルプ

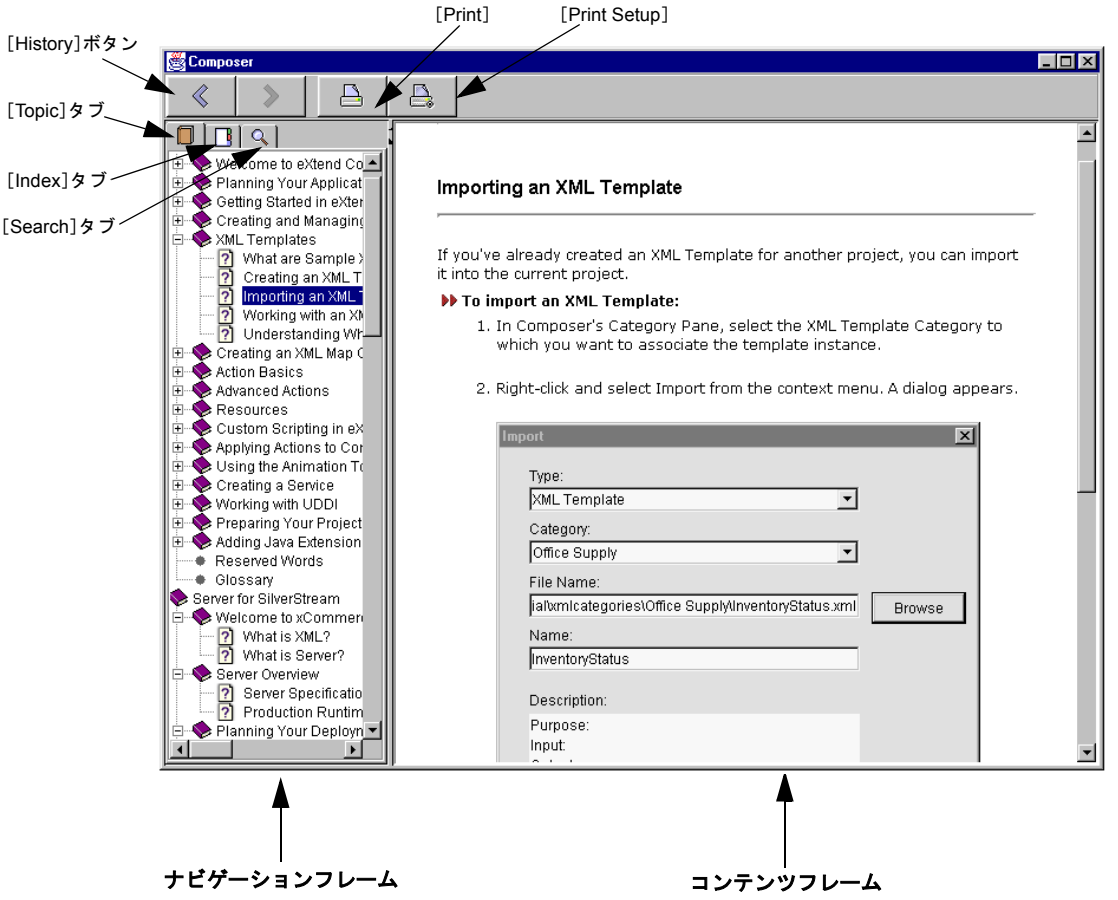
Composer には、プログラムの使用時に役立つオンラインヘルプが複数の形式で備えられています。すべてのオンラインヘルプには、［**Help**］メニューからアクセスできます。このメニューで使用できるコマンドは、次のとおりです。

表 3-5

ヘルプタイプ	説明
［Help Topics］	現在作業しているタスクに関するヘルプを表示します。［Help Topics］へは、3 つの方法のうちの 1 つを使用してアクセスできます。［**Help**］メニューから［**Help Topics**］を選択して、オンラインヘルプの目次を表示します。目次では、表示するトピックを選択できます。また、**<F1>** キーを押すことにより、現在選択されているダイアログボックスのヘルプをいつでも表示することができます。さらに、ダイアログボックスの左下角にある［Help］ボタンをクリックする (または、ダイアログボックスを表示中に <F1> キー押す) と、状況に応じたヘルプを表示できます。
［Novell on the Web］	World Wide Web 上にあるヘルプリソースのリストを表示します。ヘルプページを選択すると、デフォルトのブラウザが開き、選択された URL がブラウザにロードされます。
［My Project］	プロジェクトのカスタムドキュメントを HTML ファイルの形式で作成できます。カスタムドキュメントを作成すると、このオプションにより、プロジェクトのカスタムヘルプファイルのリストが表示されます。

［**My Project**］を選択すると表示されるオンラインヘルプファイルは、Composer の設定ファイル (Composer の **\bin** ディレクトリにある **xconfig.xml** ファイル) 内で設定されます。

注記： ヘルプシステムを初めて呼び出す場合は、システムがロードされてメモリにキャッシュされるまでの間、わずかな遅延が生じるのに気付くことがあります。この遅延は、ヘルプシステムに今後アクセスする場合には発生しません。



オンラインヘルプの使用

オンラインヘルプは、<**F1**> キーをクリックすることによって、いつでも呼び出すことができます。自由に動かすことのできる新しいモードレスウィンドウが表示されます (前の図を参照)。

注記： <F1> キーを押したときにモーダルダイアログボックスがアクティブになっていた場合、ヘルプウィンドウのコンテンツペインには、「状況依存ヘルプ」が通常は表示されます。

Composer のオンラインヘルプシステムのコンテンツは、製品マニュアルに由来しており、基本的には PDF 版のマニュアルと同じものですが、HTML 形式で表示されます。HTML ファイルは、Composer インストールの **\Composer\Help** ディレクトリに、製品別に整理されています。特定製品のヘルプの HTML ファイルを検索するには、**\Composer\Help\[product]\HlpTopic** にアクセスします。ここで、[product] には、Composer、JDBC、Server などの中から該当するものを入力します。これらの HTML ファイルは、好きな Web ブラウザを使用して表示できます。Composer のビューアを使用する必要はありません。

Composer のオンラインヘルプビューアでは、1 つの統合ヘルプセットで、インストールされているすべての Composer 製品 (コネクタを含む) を網羅したすべてのヘルプトピックにアクセスできます。つまり、使用しているコンポーネントエディタやウィザードパネルなどにかかわらず、インストール済みの製品全体に関する統合ヘルプを常に使用できます。

オンラインヘルプでの移動

Composer のヘルプシステムには、左側の (ナビゲーション) フレームの上部に [Topic] タブ、[Index] タブ、および [Search] タブによって示される 3 つナビゲーションオプションがあります (前の図を参照)。

コンテンツの表示

[Topic] タブを選択すると、インストールされているすべての Composer 製品を網羅した全ヘルプトピックの完全なリストが表示されます。

「本」のアイコンはフォルダを示しています。本のアイコンをダブルクリックすると、そのフォルダレベルの下層にあるツリーが展開します (本のアイコンが選択されている場合、コンテンツペインには有用な情報は表示されません。詳細なコンテンツを表示するには、本のアイコンの下にあるトピックを選択します)。

(疑問符が中に表示されている)「ページ」のアイコンは、使用できる詳細なヘルプの各トピックを示しています。ページアイコンをシングルクリックすると、ビューアのコンテンツフレームに関連するコンテンツが表示されます。

注記 : ナビゲーションフレームでは、上下の矢印を使用してトピック間を迅速に移動できます。また、<Return> キーを押して、フォルダ (本のアイコン) を展開することもできます。

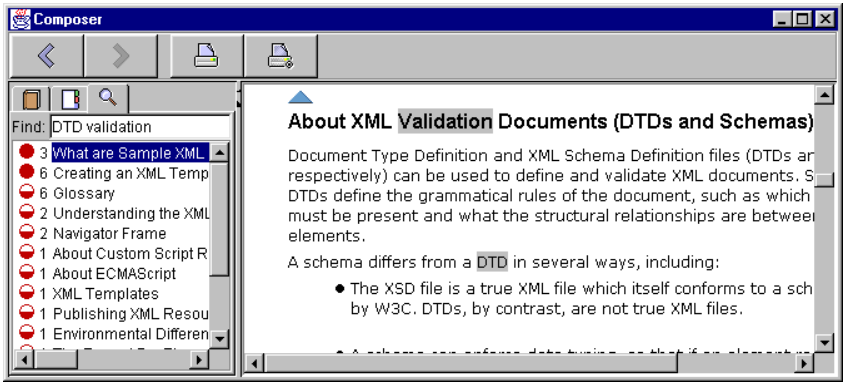
インデックス

[Index] タブを選択すると、ヘルプウィンドウのナビゲーションフレームに、トピックのインデックスが五十音順で表示されます。リストのトピックをシングルクリックすると、そのコンテンツが表示されます。

フレームの上部にあるテキストフィールドにキーワード (またはその一部) を入力すると、インデックスリスト内で入力内容に最も一致するトピックを参照しているものが表示されます。この方法では、手動でスクロールするよりも高速に検索できます。

フルテキスト検索エンジンでは、検索で返された一致を「緩和ルール」によって関連度順にランク付けする自然言語検索技術を使用します。検索の最初の列にある赤い円は、関連度のランクを示します。赤い円が完全に塗りつぶされている場合は、関連度が最も高いことを表しており、塗りつぶしの領域が少ない場合は、関連度が低いことを表しています。円の隣にある数字は、3 番目の列にリストされているトピックに対して検索エンジンによって見つけられた一致の数を示しています。3 番目の列には、一致を含むトピックの名前が表示されます (次の図を参照)。検索クエリがより詳細で、さらに多くの情報が含まれている場合、関連度のランクが上がります。

ナビゲーションフレームで任意の「ヒット」をシングルクリックすると、関連するコンテンツがコンテンツフレームに表示されます。テキストの関連セクションは、ビューアによって自動的にスクロールされ、「ヒットした語句」がグレーで表示されます (次の図を参照)。



検索エンジンでは、共通のルーツを持つ語句を検索するために、単語モーフィング技術も使用します。たとえば、検索文字列に「build」という単語が含まれている場合、「build」、「builder」、「building」、および「builds」を含む一致が返されます。

ヘルプでは、部分テキスト検索も行われます。たとえば、［Index］の［Find］ボックスに「x」という文字を入力すると、*examples*、*execution errors*、および *XML Integration* が検索されます。また、ヘルプでは、語句全体に対して近い一致が検索されるため、*execute* を検索すると、「execute」および「executes」が返されます。

ヘルプの印刷

上部ツールバーにある［Print］ボタンおよび［Print Setup］ボタンを使用すると、現在のヘルプトピックを印刷できます。

4 プロジェクトの作成と管理

プロジェクトとは

「プロジェクト」は、XML ベースの B2B 統合サービスを実行するために設計された Composer オブジェクトの集合です。プロジェクトでは、作成しているアプリケーションのオブジェクトがすべて保持されます。プロジェクトは、exteNd Server に配備される単位です。アプリケーションサーバに配備できるプロジェクトの数に制限はありません。

プロジェクトの各部分は、次の図のとおりです。

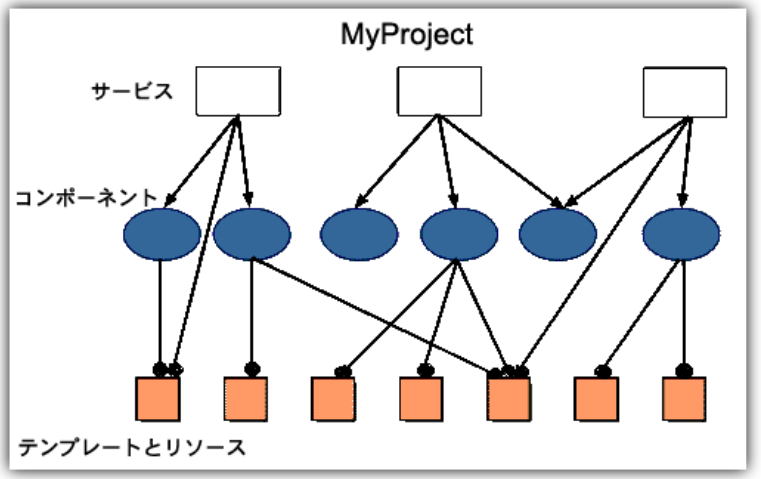


図 4-1

サービスについて

サービスは、作成するさまざまなコンポーネントを結合して、exteNd Server に対する配備の論理単位を作成するために使用されます。サービスは、exteNd Server 内で実際に実行されるオブジェクトです。Web サービスは、サービストリガオブジェクトにより開始され、XML ドキュメントを入力として受け付け、出力として XML ドキュメントを返します。JMS サービスは、メッセージを入力として受け付け、メッセージがキューに入るとトリガされます。詳細については、321 ページ「新しいサービスの作成」を参照してください。

コンポーネントについて

「コンポーネント」は、1 つまたは複数の XML ドキュメントを入力として受け付け、入力で動作するアクションの集合を使用し、XML ドキュメントを出力として返す xObject です。

通常、コンポーネントは「サービス」内で呼び出され、アクションまたはその他のコンポーネントへの呼び出しを含むことができます (サービスは、基本的にコンポーネントの集合です)。

コンポーネントの機能、作成方法、および設計上で基盤となる原理の詳細については、105 ページ「XML Map コンポーネントの作成」を参照してください。

リソースについて

リソースは、専用の操作を実行する xObject です。これらは、タスクを実行できるようにするためにサービスとコンポーネントによって使用されます。リソースのタイプには、コードテーブル、コードテーブルマップ、接続、およびカスタムスクリプトがあります。

XML テンプレートについて

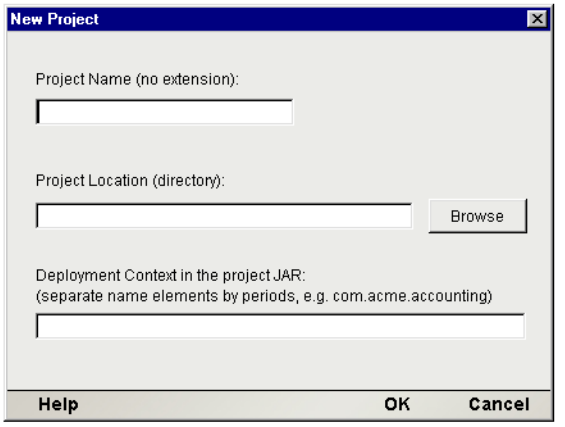
XML テンプレートには、コンポーネントを設計およびテストする際に便利なサンプルドキュメント、スキーマ、およびスタイルシートが含まれています。まず、類似の XML テンプレートを含む XML カテゴリを作成します。次に、XML テンプレートを作成し、作成するコンポーネントの入力および出力としてそのテンプレートを使用します。詳細については、87 ページ「XML テンプレートについて」を参照してください。

新しいプロジェクトの作成

Composer を初めて起動すると、チュートリアルと呼ばれるサンプルプロジェクトがロードされます。独自のアプリケーションを起動する場合は、新しい空のプロジェクトを作成して起動します。

➢ 新しいプロジェクトを作成する

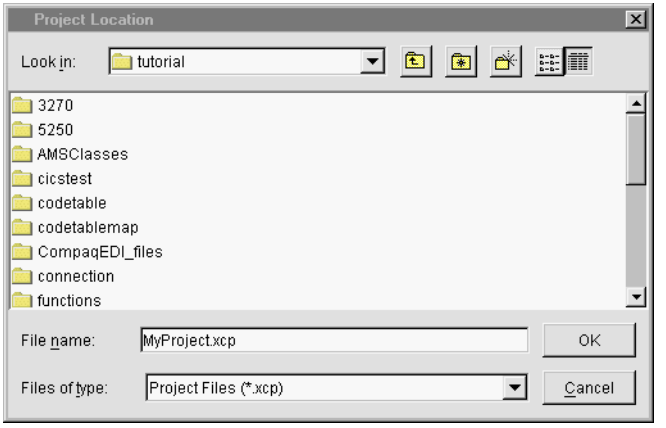
1 ［**File**］、［**New Project**］の順にクリックします。［New Project］ダイアログボックスが表示されます。



2 ［**Project Name**］にプロジェクト名を入力します。これは必須のフィールドです。Composer により、プロジェクト名の拡張子である「.spf」が追加されます。

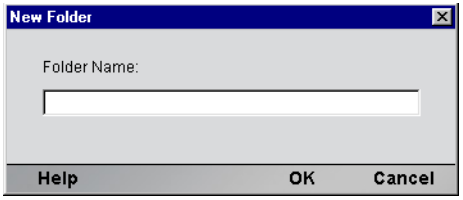
3 ［**Browse**］を選択して、プロジェクトを保存するフォルダを検索します。［Project Location］ダイアログボックスが表示されます。

注記： プロジェクトが開いている場合、［Project Location］ダイアログボックスには、開いているプロジェクトが保存されているフォルダがデフォルトで表示されます。



4 プロジェクトを保存するフォルダに移動します。

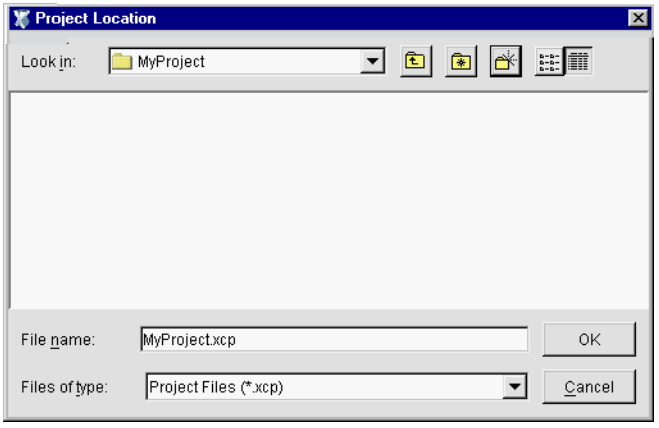
5 新しいプロジェクトを保存する新しいフォルダを作成するには、［**New Folder**］アイコンをクリックします。 ［New Folder］ダイアログボックスが表示されます。



注記： プロジェクトのフォルダ名は、この手順の説明に従って手動で作成する必要があります。

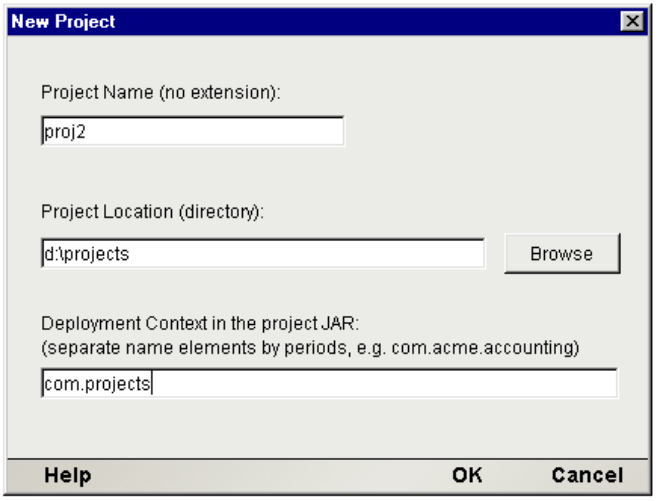
6 「**フォルダ名**」を入力します。

7 ［**OK**］をクリックします。新しく作成したフォルダが［**Look In:**］フィールドに表示された［ファイルのディレクトリ］ダイアログボックスが現れます。



注記： ［Project Location］ダイアログボックスの［**File Name**］には、手順 2 で指定したプロジェクト名がデフォルトで表示されます。

8 ［**OK**］をクリックします。新しく作成した「**プロジェクト名**」と「**プロジェクトの場所**」が表示された［New Project］ダイアログボックスが現れます。



9 ダイアログボックスの最下部にあるテキストフィールドに、「配備コンテキスト」文字列を入力します。この文字列には、「com.server.apps」のような、ピリオドで区切られたラベル (スペースなし) を含める必要があります。

注記 : コンテキスト文字列には、*try*、*catch*、*finally*、*int*、*for* などの Java 言語キーワードを含むことはできません。Java キーワードの完全なリストについては、付録「予約語」を参照してください。

10 [**OK**] をクリックします。タイトルバーで作成したプロジェクトの名前が表示された Composer ウィンドウが現れます。

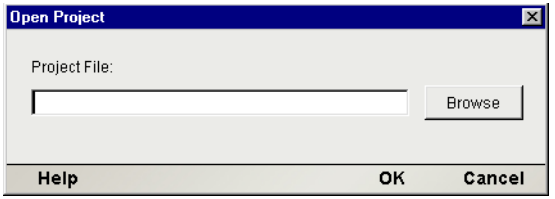
プロジェクトを開く

プロジェクトは、次の方法で開くことができます。

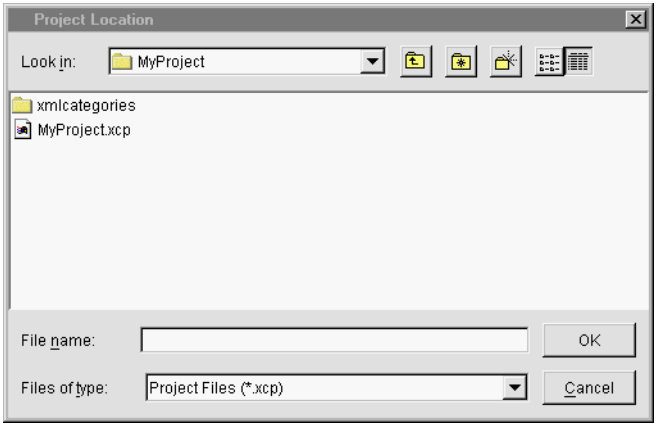
Composer 内からプロジェクトを開く

➢ **既存のプロジェクトを開く**

1 [**File**]、[**Open Project**] の順にクリックします。[Open Project] ダイアログボックスが表示されます。



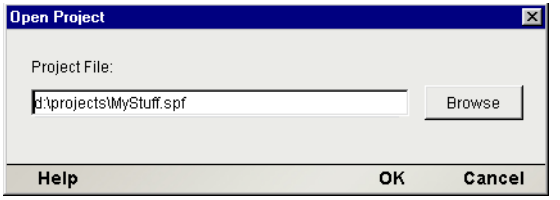
2 ［**Browse**］ボタンをクリックします。また、開くプロジェクトのパスを入力することもできます。［Project Location］ダイアログボックスが表示されます。



3 開くプロジェクトが保存されているディレクトリに移動します。

4 プロジェクトを選択します。

5 ［**OK**］をクリックします。選択したプロジェクトのパスが［**Project File**］フィールドに表示された［Open Project］ダイアログボックスが現れます。



6 ［**OK**］をクリックします。タイトルバーで開いたプロジェクトの名前が表示された Composer ウィンドウが現れます。

注記： プロジェクトは、［File］メニューの［**Recent Projects**］リストから選択して開くこともできます。

コマンドラインから Composer を起動するときに特定のプロジェクトを開く

起動オプションとして、Composer は、コマンドラインモードで **XC.exe** を実行して起動し、次のようなプロジェクト名パラメータを指定することができます。

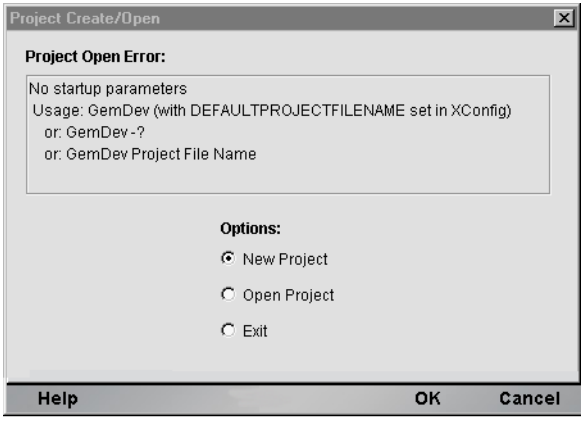
```
C:\exteNd\Composer\Bin\xc myproject
```

この例では、*myproject.spf* という名前のプロジェクトファイルで **XC.exe** が実行されます。

最近のプロジェクトが見つからないときにプロジェクトを開く

Composer を起動すると、最後に作業したプロジェクトが自動的にロードされます。プロジェクトファイルを移動した場合や、アクセス不可能な別のユーザのプロジェクトにアクセスしようとした場合は、Composer によってプロジェクトは見つけられません。

起動時に、Composer では、開くプロジェクトファイルの名前の最初のコマンドラインパラメータを使用します。コマンドラインパラメータが省略されている (または不正である) 場合、Composer では、開くプロジェクトの名前として、xconfig.xml の DEFAULTPROJECTFILENAME を使用します。コマンドラインオプションが省略されているか無効であり、DEFAULTPROJECTFILENAME も省略されているか無効である場合、Composer によって［Project Create/Open］ダイアログボックスが表示されます。



［Project Create/Open］ダイアログボックスでは、新しいプロジェクトを作成したり、既存プロジェクトを開いたり、Composer を完全に終了したりすることができます。また、このダイアログボックスには、初期プロジェクトを開けなかった場合のエラー情報も表示されます。

➢ 起動時にプロジェクトを検索する

1 ［Project Create/Open］ダイアログボックスで、次のいずれかを実行します。

- ［**New Project**］ラジオボタンをオンにして、［New Project］ダイアログボックスを表示します。
- ［**Open Project**］ラジオボタンをオンにして、プロジェクトを検索します。
- ［**Exit**］ラジオボタンをオンにして、終了します。プロジェクトファイルの保存場所の詳細については、73 ページ「プロジェクトファイルの保存場所について」を参照してください。

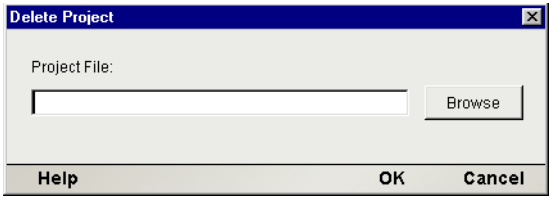
2 ［**OK**］をクリックします。

プロジェクトの削除

exteNd Composer を使用すると、プロジェクト全体とそのオブジェクトをすべてディスクドライブから削除できます。また、プロジェクトは通常のディレクトリ構造で保存されているため、Windows の標準の削除機能を使用して削除することもできます。いずれの場合も、削除を行った場合、プロジェクトは完全に破棄され、exteNd で復元することはできません。

➢ プロジェクトを削除する

1 ［**File**］、［**Delete Project**］の順にクリックします。［Delete Project］ダイアログボックスが表示されます。



2 ［Delete Project］ダイアログボックスで、次のいずれかを実行します。

- 削除するプロジェクトの完全パスとプロジェクトの .spf ファイルを指定します。
- ［**Browse**］ラジオボタンをオンにして、削除するプロジェクトを検索します。

3 ［**OK**］をクリックします。［Confirm Project Delete］ダイアログボックスが表示されます。

4 ［**Yes**］を選択します。

xObject の管理

xObject は、すべての XML 統合サービスの構成要素です。プロジェクトコンポーネントは、次のいずれかの方法で作成できます。

- xObject を作成する
- 既存の xObject を開く
- xObject を他のプロジェクトからインポートする
- これらの 3 つを組み合わせた操作を行う

xObject の作成

xObject は、メニューバーから作成し、コンポーネントで使用できます。

➢ xObject を作成する

1. ［**File**］メニューから、［**New xObject**］を選択します。
2. 作成する xObject のタイプを選択します。選択肢は、［**Service**］、［**Component**］、［**Resource**］、［**XML Template**］、および［**XML Template Category**］です。
3. ［**Component**］を選択した場合は、コンポーネントタイプを選択します。選択肢は、インストールした Connect 製品によって異なります。
4. ［**Resource**］を選択した場合は、リソースタイプを選択します。選択肢は、［**Code Table**］、［**Code Table Map**］、［**Connection**］、および［**Custom Script**］です。
5. xObject の名前を入力します。
6. xObject 定義画面の残りを完了します。［**Finish**］をクリックして完了し、xObject を保存します。

xObject は、タイプに応じて Composer 内の適切なカテゴリに配置されます。

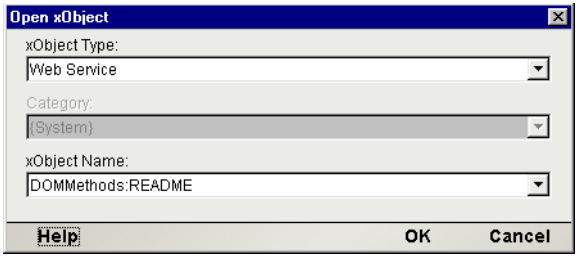
注記： 作成した xObject は、XML ファイルとしてハードディスクに保存されます。保存場所は、xObject が属するカテゴリと同じ名前のディレクトリとなります。

xObject を開く

現在のプロジェクトに既存の xObject がある場合、Composer メインウィンドウからそれらの xObject を開いて、コンテンツを編集または表示できます。

➢ xObject を開く

1. ［**File**］メニューから、［**Open xObject**］を選択します。［Open xObject］ダイアログボックスが表示されます。



2. 開く xObject のタイプを選択します。
3. xObject を選択します。
4. ［**OK**］をクリックします。

xObject が独自のウィンドウで開きます。

xObject のインポート

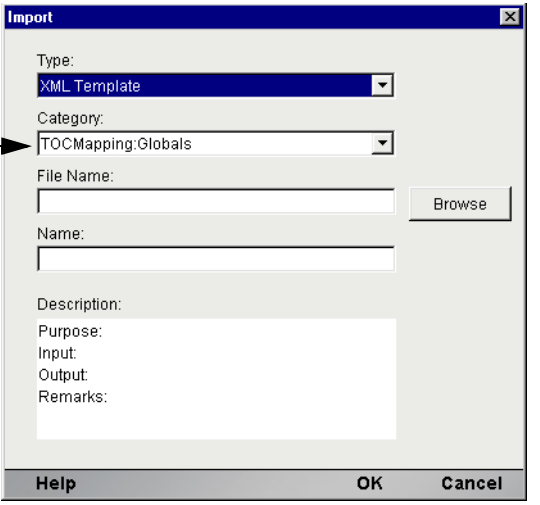
xObject は、開くだけでなく、別のプロジェクトや場所からインポートすることもできます。

注記： 作成した xObject は、XML ファイルとしてハードディスクに保存されます。保存場所は、xObject が属するカテゴリと同じ名前のディレクトリとなります。xObject をインポートするには、次の手順に従って xObject の XML ファイルを選択します。

➢ xObject をインポートする

1 ［**File**］メニューから、［**Import xObject**］を選択します。［Import xObject］ダイアログボックスが表示されます。

［Category］は、［XML Template］を選択した場合にのみ有効になります。



2 インポートする xObject のタイプを選択します。

3 ［**XML Template**］を選択した場合は、カテゴリを選択します。

4 xObject のパスとファイル名を入力するか、または［**Browse**］をクリックして xObject を検索します。

5 xObject の名前を入力するか、または元の名前をそのまま使用します。

6 オプションとして、説明するテキストを入力するか、または元のテキストをそのまま使用します。

7 ［**OK**］をクリックします。

xObject は、タイプに応じて Composer 内の適切なカテゴリに配置されます。

xObject のプロパティの表示

すべての xObject には、プロパティが関連付けられています。プロパティには、名前、説明 (ヘッダ情報)、および xObject タイプに固有なその他の情報があります。

➢ xObject のプロパティを表示する

1. 詳細ペインで xObject を選択します。
2. ［**File**］メニューから、［**Properties**］を選択します。
3. ［Properties］ダイアログボックスのタブをクリックし、［Header］および［XML］のプロパティを表示します。

注記： マウスを右クリックして、コンテキストメニューから［Properties］を選択することもできます。

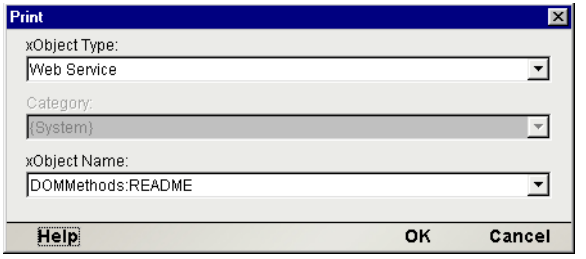
xObject のプロパティの印刷

xObject のプロパティは、表示するだけでなく印刷することもできます。xObject を印刷すると、印刷日時の他に、xObject の名前、およびすべてのヘッダと xObject タイプに固有なその他の情報が含まれます。

コンポーネントのプロパティを印刷すると、コンポーネントの DOM 構造 (107 ページ「DOM とは」を参照) やアクションモデル (125 ページ「アクションモデルペインについて」を参照) に関するデータがすべて含まれます。

➢ xObject のプロパティを印刷する

1. ［**File**］メニューから、［**Print**］を選択します。［Print］ダイアログボックスが表示されます。



2. xObject タイプを選択します。
3. xObject を選択します。
4. ［**OK**］をクリックします。
5. 任意のプリンタオプションを設定し、［**OK**］をクリックします。

最初に xObject を選択してから印刷することもできます。

➢ **選択した xObject を印刷する**

1 詳細ペインで xObject を選択します。

2 マウスを右クリックし、コンテキストメニューから［**Print**］を選択します。

3 任意のプリンタオプションを設定し、［**OK**］をクリックします。

xObject の名前変更

xObject の名前を変更するには、xObject を右クリックし、［**Rename**］を選択します。テキストフィールドに新しい名前を入力します。

注記： ［Properties］ページで xObject の名前を変更すると、［**Save As**］操作が行われるため、元の xObject はそのまま維持され、新しい名前を持つコピーが作成されます。xObject の名前のみを変更するには、コンテキストメニューの［**Rename**］オプションを使用します。

xObject の削除

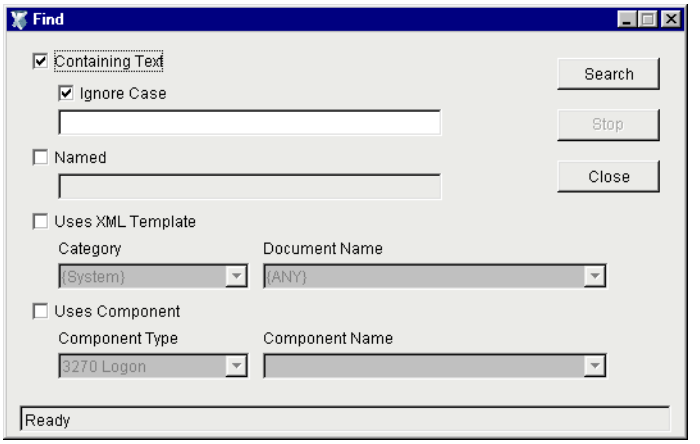
xObject を削除するには、xObject を右クリックし、［**Delete**］を選択します。続けて、xObject を削除することを確認します。

xObject またはテキストの検索

プロジェクトには、作成時に多数のサービス、コンポーネント、リソース、および XML テンプレートを含めることができます。［Find］ダイアログボックスを使用すると、xObject、オブジェクト内のテキスト、または特定の XML テンプレートやコンポーネントを参照するオブジェクトを検索できます。

➢ **プロジェクト内の xObject を検索する**

1 ［**Tools**］、［**Find**］の順にクリックします。［Find］ダイアログボックスが表示されます。



2 検索基準を選択して指定します。

- xObject の名前 (または名前の冒頭)
- xObject に含まれる可能性のあるテキスト文字列
- xObject を使用する XML テンプレート
- xObject を使用するコンポーネント
- これらの検索基準の組み合わせ

3 ［**Search**］をクリックします。

見つかった場合、ターゲット xObject は、Composer メインウィンドウの［Find］タブにリスト形式で表示されます。検索結果リストに表示されたコンポーネントやオブジェクトは、ダブルクリックして開くことができます。ナビゲーションフレームのカテゴリペイン内の場合のように、xObject の隣にあるアイコンは、そのタイプ (コンポーネント、サービス、リソース、または XML テンプレート) を示しています。

システムメッセージの表示

コンポーネントの実行中、特定のメッセージ (例 : Composer からの内部システムメッセージ、Log アクションで指定されたテキスト) がログファイルの **xcsyslog.txt** に書き込まれます。このファイルの場所は、**xconfig.xml** ファイル (Composer の設計時インストールの **\bin** ディレクトリにあります) のコンテンツを変更することによって指定できます。**xconfig.xml** の <LOGFILE> 要素を検索し、そのコンテンツを希望するパス名に変更します。

注記 : ログファイルパスを簡単に変更する方法は、［Configuration］ダイアログボックスの［System Environment］タブで新しいパスを入力することです。このダイアログボックスは、［Tools］メニューの［Configuration］コマンドを使用して表示します **(51 ページ「設定ファイルでシステム環境設定を指定する」を参照)****。**

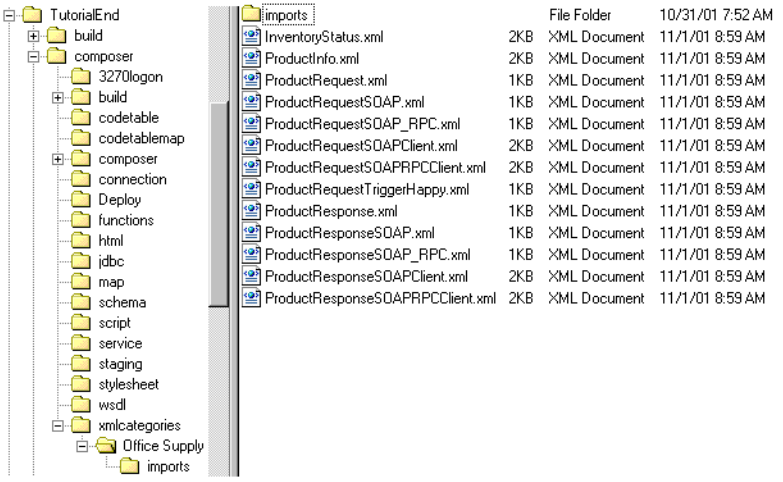
アニメーション時、または Composer でコンポーネントを実行している場合、システムメッセージ (および Log アクション出力) が Composer メインウィンドウの下部にあるメッセージペインにリアルタイムで表示されます。ウィンドウの下部にある［Log］タブを選択します (メッセージペインが非表示になっている場合は、メインメニューから［**View**］>［**Output Tabs**］の順に選択します)。

プロジェクトファイルの保存場所について

Composer で作成したすべての exteNd オブジェクト (プロジェクト、XML カテゴリ、XML テンプレート、コンポーネント、リソース、サービスなど) は、オブジェクトタイプに一致する名前のフォルダに保存されます。

プロジェクトを作成すると、プロジェクトファイル (例 : myproject.spf) が、そのプロジェクトに由来して名付けられたフォルダに保存されます (プロジェクトに対して新しいフォルダが手動で作成されていることを前提とします)。XML テンプレート、リソース、およびコンポーネントを作成することによってアプリケーションを作成する際に、作成されたオブジェクトは、プロジェクトフォルダのサブフォルダ内に XML ファイルとして保存されます。したがって、「AcceptInvoice」という名前のサービスを作成すると、そのサービスによって実行されるアクションがすべて含まれる「AcceptInvoice.xml」という名前の XML ファイルが作成されます。すべての XML テンプレートカテゴリは、カテゴリの名前別に［XMLCategories］に保存されます。すべての XML テンプレートは、テンプレートの名前別に［category］に保存されます。あるカテゴリのすべての XML サンプル (つまり、複数のテンプレートが存在する場合がある) は、サンプルドキュメントの名前別に［Imports］に保存されます。

ファイルの保存場所の例は、次の図のとおりです。



設計時および配備済みのプロジェクトファイルについて

すべての XML ドキュメントとサポートファイルは、プロジェクトの一部です。プロジェクトを配備すると、プロジェクトファイルは、Java Archive ファイル (JAR ファイル) に保存されます。

プロジェクトを構成するファイルは、次の表のとおりです。

表 4-1:

プロジェクトファイル名	説明
[プロジェクト名].spf	exteNd Composer プロジェクトファイル。プロジェクトの起動情報が保存されます。このファイルは、新しいプロジェクトの作成時に作成されます。
PROJECT.xml	これは、Composer によって作成されるオプションのファイルです。このファイルには、ユーザが定義するプロジェクト変数が含まれます。詳細については、75 ページ「プロジェクト変数の作成」を参照してください。
[プロジェクト名].jar	配備中に作成される Java Archive。詳細については、『*exteNd Server ガイド*』を参照してください。
.xml、.xsl	アプリケーションの設計時に使用するすべての XML サンプル、定義、およびスタイルシートは、プロジェクトフォルダの下層にあるフォルダに保存されます。

プロジェクト変数の作成

「プロジェクト変数」を使用すると、要素の値を指定し、作成するすべてのコンポーネントおよび機能でその指定した要素をグローバルに使用できます。ECMAScript の「globals」とは異なり (globals は、globals を使用するコンポーネントにスコープされる)、プロジェクト変数はサービスのセッションにスコープされるため、Composer サービス内で実行されているコンポーネントであればいくつでも、プロジェクト変数を使用できます。

プロジェクト変数は、$PROJECT と呼ばれるメモリ内 DOM に保存される値として実装されます。次に、この DOM は、Composer によって各プロジェクトに作成される **PROJECT.xml** というファイルから派生されます (このファイルは、後でサーバに配備されます)。

注記： ランタイム時に発生したプロジェクト変数を変更しても、サービスの呼び出し全体では適用されません。運用環境では、PROJECT.xml は読み取り専用です (「持続的」な globals を作成するには、XML Interchange アクションを使用して独自のスクラッチファイルの読み書きする必要があります)。

プロジェクト変数はスコープにおいてグローバルであるため、プロジェクトの設計時開発中と配備後のランタイム保守の両方で重要な機能を実行できます。

設計時には、プロジェクト変数により、プロジェクト内の複数の場所で使用しなければならない可能性のある、プロジェクト全体の値を容易に集中管理できます。配備時には、プロジェクト変数ファイル (**PROJECT.xml**) により、プロジェクトの静的変数を容易に更新できます。また、配備後は、アプリケーションサーバ上に配備された **PROJECT.xml** ファイルを更新するだけで、複数の配備済みコンポーネントやサービスの動作を簡単に変更できます。

プロジェクト変数として最適に保存される可能性のある項目は、次のとおりです。

- 項目のコンポーネント内で参照される URL
  - ログファイルのパス
  - DTD およびスキーマのパス
  - XSL スタイルシート
  - XML 交換 URL
- メール送信 — メールサーバ ID
- データベースまたはバックエンドシステムとの接続を確立するために必要な認証情報
- メッセージキュー名
- サービスおよびコンポーネントに該当するバージョン情報

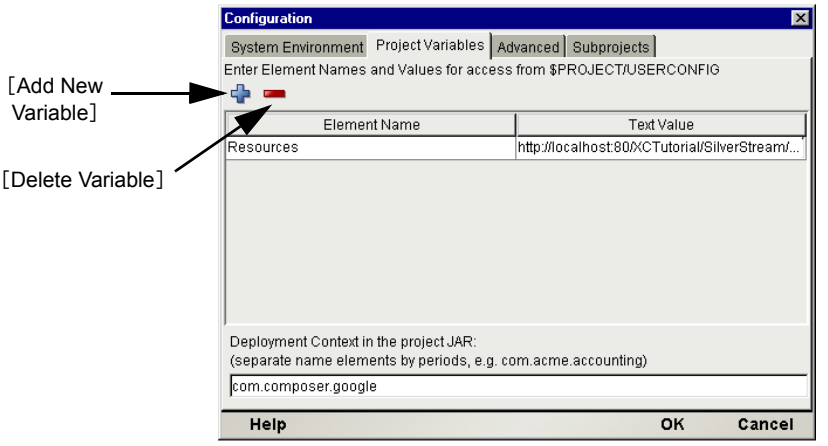
PROJECT.xml ファイルの更新プロセスについては、『*exteNd Server ガイド*』で詳しく説明されています。

プロジェクトへのプロジェクト変数の追加

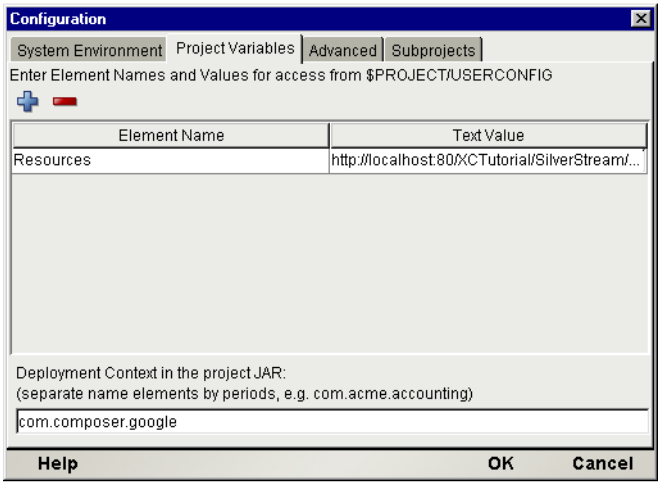
特定の値にマップするプロジェクト変数の名前を作成します。特定の情報をプロジェクト変数から間接的に参照すると、データを 1ヶ所で変更するだけで、そのデータが使用されているすべての場所で同じく新しい値が取得されるようにすることが可能です。

➢ プロジェクト変数を追加する

1. Composer ウィンドウで、［**Tools**］、［**Configuration**］の順に選択します。［Configuration］ダイアログボックスが表示されます。
2. ［**Project Variables**］タブを選択します。［Project Variables］ウィンドウが表示されます。



3 ［**Add New Variable**］ボタンをクリックします。［Project Variables］ウィンドウに空のフィールドが表示されます。



4 空の［**Element Name**］フィールドをクリックし、要素名を入力します。たとえば、「CompanyName」のように要素を入力します。スペースは使用しないでください。

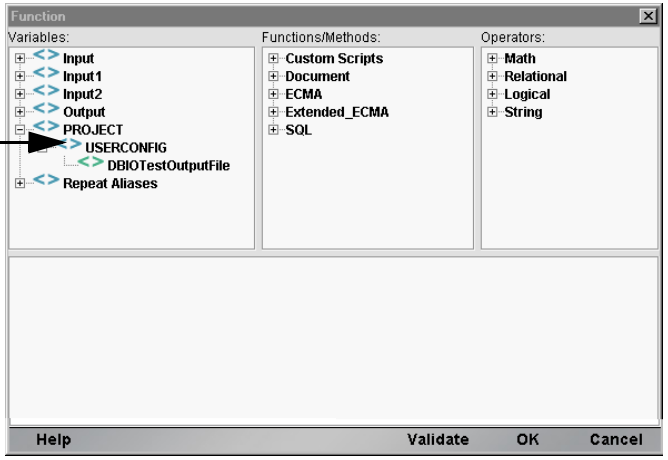
5 空の［**Text Value**］フィールドをクリックし、テキスト値を入力します。たとえば、「ABC Corporation」と入力します。

6 ［**OK**］をクリックします。

これで、作成した要素名とテキスト値は、PROJECT.xml というプロジェクト XML ファイルに保存されます。変数値を変更する必要がある場合は、プロジェクトの配備後にこのファイルを手動で編集できます。

要素名と値は、コンポーネントの作成に使用できるよう、Composer のダイアログボックスに自動的に追加されます。たとえば、変数は関数で使用することができます。

プロジェクト変数は、コンポーネントの作成時に使用できます。



プロジェクト変数の動的な作成

プロジェクト変数は、恒久的 (静的) に作成できるだけでなく、コンポーネントまたはサービス内で動的に作成することもできます。

$PROJECT DOM は、[Map action] ダイアログボックスに表示される DOM リスト (ドロップダウンメニュー) に常に存在します。要素と要素の値を他の DOM にマップする場合と同じ方法 (ドラッグアンドドロップ操作を含む) で $PROJECT DOM にマップできるため、プロジェクト変数を作成して値を割り当てることは非常に簡単です。

注記： $PROJECT DOM のコンテンツは、Composer の［**View**］メニューから［**Window Layout**］を選択し、$PROJECT DOM を表示させることによって、ツリー、テキスト、または様式化された形式でいつでも表示できます (このガイドの 116 ページ「コンポーネントエディタでの［Window Layout］および［Show/Hide］の使用」での説明を参照)。

PROJECT.xml ファイルの構造を見ると、USERCONFIG というルート要素があることが分かります。ユーザ定義変数は、このノードに子要素として追加されています。子要素の文字列値は、要素名に対応するプロジェクト変数の値です。

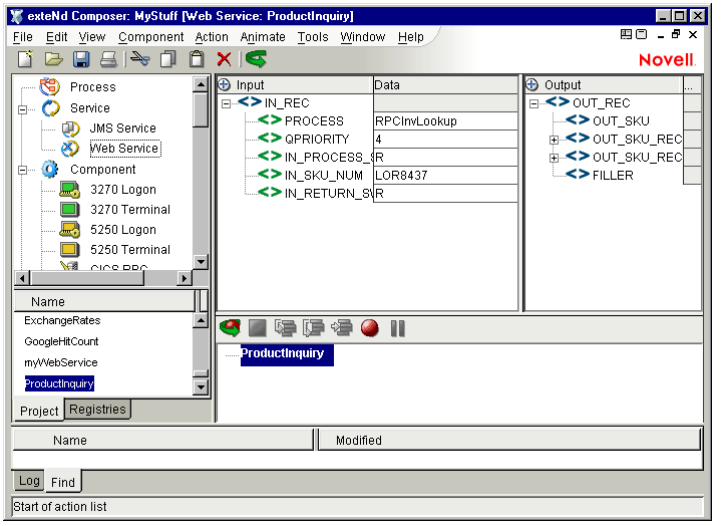
USERCONFIG の下層には、ユーザ定義プロジェクト変数名に加え、Composer 定義要素もあります。これは、**PROJECT.xml** ファイルを使用して特定のプロジェクトの初期設定値が Composer で保持されるためです。

動的に作成されたプロジェクト変数は、当然のことながら、一時的なものです。動的なプロジェクト変数は、実行中のサービスのライフタイムの間使用できます (その後、サービスによって、このサービスを使用する多くのコンポーネントが呼び出される場合があります)。サービスの実行が終了すると、動的な変数は破棄されます。これは、動的な変数がメモリ内で作成されたためです。

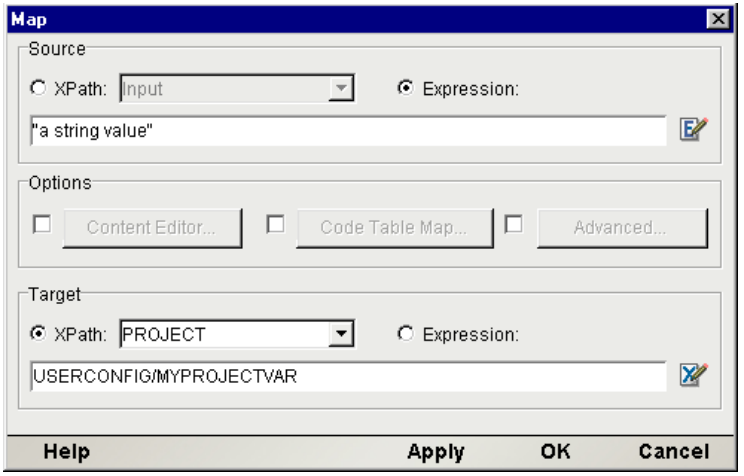
プロジェクト変数には、$PROJECT DOM のノードにマップすることによって、値を何度でも再割り当てできます。この例は、次のとおりです。

➢ 動的なプロジェクト変数を作成して値をマップする

1 インスタンスペインでサービスをダブルクリックします。サービスエディタウィンドウが表示されます。



2 メインメニューバーで、[**Action**]、[**Map**] の順にクリックします。[Map] ダイアログボックスが表示されます。



3 ［Map］ダイアログボックスの［**Source**］セクションで、［**Expression**］ラジオボタンをオンにします。

4 ［**Source**］フィールドに、プロジェクト変数の値を入力します。文字列の値を入力する場合は、値を二重引用符で囲んでください。

5 ターゲット式は、次の 2 種類の方法で入力できます。

- ダイアログボックスの［**Target**］セクションで［**Expression**］ラジオボタンをオンにし、［**Target**］フィールドに次を入力します。

```
PROJECT.createXPath(“USERCONFIG/MYPROJECTVAR”)
```

ここで、*MYPROJECTVAR* は、作成するプロジェクト変数の名前です。

- ［Target］で［**XPath**］ラジオボタンをオンにし、ドロップダウンメニューから［PROJECT］を選択した後、下にある［Target］フィールドに次を入力します。

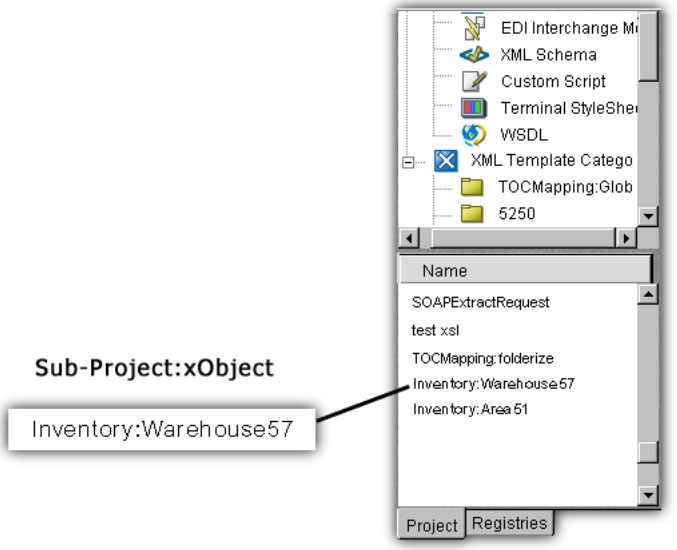
```
USERCONFIG/MYPROJECTVAR
```

ここで、*MYPROJECTVAR* は、作成するプロジェクト変数の名前です (完全なダイアログボックスの状態については、前の図を参照してください)。

6 ［**OK**］をクリックします。これで、作成した動的変数は、サービスエディタウィンドウのアクションペインに表示されます。

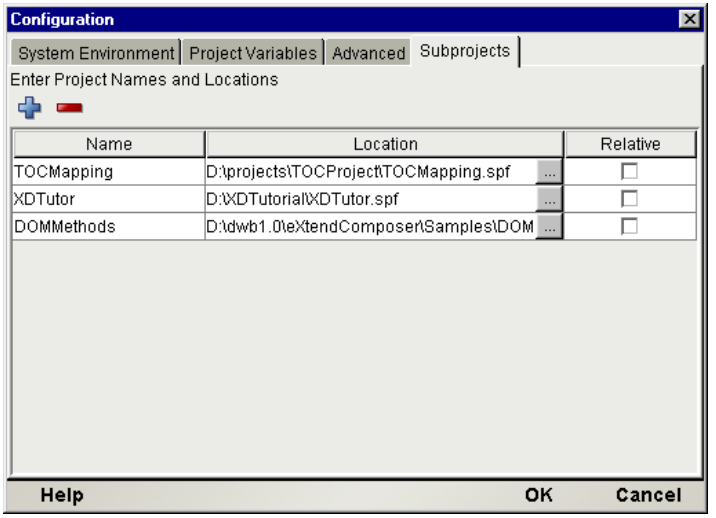
プロジェクト内のサブプロジェクト

現在のプロジェクト内には、他の Composer プロジェクトを追加することができます。これは、既存の xObject を再利用することによってアプリケーションの開発を迅速に行うための機能です。こうした目的で再利用する場合、外部プロジェクトは「サブプロジェクト」と呼ばれます。サブプロジェクトの xObject は、名付けられたサブプロジェクトのオブジェクトであることを識別するためにプロジェクトのプリフィックスが各プロジェクトの名前の前に表示されることを除いて、他は通常どおりに現在のプロジェクトのカテゴリペインとインスタンスペインに表示されます。例は次のとおりです。



➢ Composer プロジェクトにサブプロジェクトを含める

1. Composer のメインメニューバーにある［**Tools**］メニューで、［**Configuration**］を選択します。［Configuration］ダイアログボックスが表示されます。
2. ［Configuration］ダイアログボックスの［**Subprojects**］タブを選択します (次を参照)。



3 ダイアログボックスの左上角にある「**プラス記号**」アイコンをクリックして、サブプロジェクトを追加します。ダイアログボックスが表示され、ファイルシステムを参照することができます。Composer で作成された **.spf** ファイルを選択します。選択したファイルは、サブプロジェクトのリストに表示されます。

注記： 選択する **.spf** に独自のサブプロジェクトがすでに含まれている場合、サブプロジェクトを含むサブプロジェクトを追加することはできないことを通知するエラーダイアログボックスが表示されます。

4 サブプロジェクトの場所を (メインプロジェクトの .spf への) 相対パスに変更する場合は、［**Relative**］チェックボックスをオンにします。メインプロジェクトのドライブとは異なるドライブにプロジェクトがある場合、［Relative］チェックボックスは無効になります。

5 サブプロジェクトを削除するには、そのサブプロジェクトを選択し、「**マイナス記号**」アイコンをクリックします。

6 手順 3 を繰り返し、任意の数のサブプロジェクトを追加します。

7 ［**OK**］をクリックしてダイアログボックスを閉じます。サブプロジェクトの xObject が、Composer のナビゲーションフレームの詳細ペインに表示されます。xObject は、各 xObject 名にネームスペースまたはコロンのプリフィックスがあるかどうかによって区別できます。

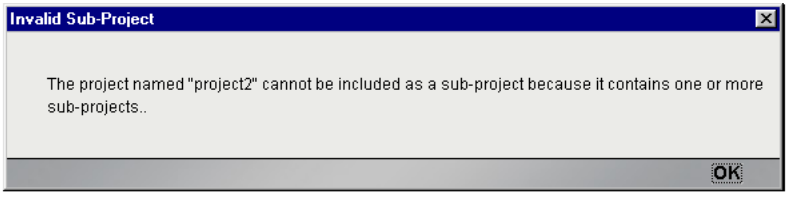
インポートされた xObject とサブプロジェクト

オブジェクトの再利用を実現するために、サブプロジェクトを活用する以外に、xObject を 1 つずつ特定のプロジェクトに直接インポートすることができます (この章の「xObject のインポート」を参照してください)。ただし、xObject の「インポート」には、元のオブジェクトの基盤となる XML ファイルが現在のプロジェクトに「コピーされてしまう」という短所があります。これにより、元の xObject の変更または更新が必要になるという点や、オブジェクトをインポートしたプロジェクトに存在する可能性のあるそのオブジェクトのすべての「コピー」において、コード保守の問題が生じます。サブプロジェクトの使用では、この問題が生じることはありません。外部プロジェクトを現在のプロジェクト内に含めても、サブプロジェクトのソースファイルのコピーは作成されません。すべての「ソースコード」は 1ヶ所に保存されたままとなるため、保守が簡単になります。

サブプロジェクトのネスト

1 レベルを超えたサブプロジェクトのネストは、サポートされていません。1 つの特定のプロジェクトには、サブプロジェクトをいくつでも含めることができますが、すべてのサブプロジェクトは同じレベル (1 レベル) に存在する必要があります。これは、1 つまたは複数のサブプロジェクトを含むプロジェクトを別のプロジェクトのサブプロジェクトにすることはできないことも意味します。たとえば、プロジェクト A にプロジェクト B という名前のサブプロジェクトが含まれているとします。この場合、3 つ目のプロジェクトであるプロジェクト C では、プロジェクト A をサブプロジェクトとして使用することはできませんが、プロジェクト B をサブプロジェクトとして使用することは可能です。

現在のプロジェクトにサブプロジェクトを追加しようとする際に、そのサブプロジェクトに独自のサブプロジェクトが含まれている場合は、次のような警告メッセージが表示されます。



サブプロジェクトでの xObject と変数のスコープおよび表示レベル

xObject と変数をプロジェクトやサブプロジェクトの間で共有する場合、注意すべき特定のスコープルールによって制限されます。

1. 「**xObject**」: プロジェクトからはサブプロジェクトのコンポーネント (および他の xObject) にアクセスできますが、サブプロジェクトからはペアレントプロジェクトのオブジェクトにアクセスできません。たとえば、プロジェクト A にプロジェクト B がサブプロジェクトとして含まれている場合、プロジェクト B (「チャイルド」プロジェクト) のコンポーネントでは、プロジェクト A (ペアレント) のコンポーネントまたはリソースを扱うことはできません。
2. 「**プロジェクト変数**」: $PROJECT DOM (この章の前半の「プロジェクトへのプロジェクト変数の追加」を参照) から派生された変数は、「それらの変数を作成したプロジェクト」に属します。プロジェクト A のコンポーネントとサービスでは、プロジェクト B に属するプロジェクト変数を「認識」することはできず、また、プロジェクト B のコンポーネントとサービスでは、プロジェクト A に属するプロジェクト変数を認識することはできません。
3. 「**ECMAScript の変数と関数**」: スクリプト変数のライフタイムは、常にコンポーネントにスコープされます。コンポーネントがスコープ外になると、そのコンポーネントによって使用された可能性のある ECMAScript 変数もスコープ外になります。一方、サブプロジェクト内のカスタムスクリプトリソースでは、`Projects` オブジェクトという組み込まれた Composer ECMAScript 拡張により、メインプロジェクトにアクセスできます。たとえば、現在のプロジェクトであるプロジェクト A に *MyOtherProject* というサブプロジェクトが含まれており、`salesTax()` という関数を含むカスタムスクリプトが MyOtherProject に含まれているとします。この場合、プロジェクト A 内のコンポーネントでは、次を呼び出すことによって salesTax()関数を使用できます。

   ```
   Projects.MyOtherProject.salesTax()
   ```

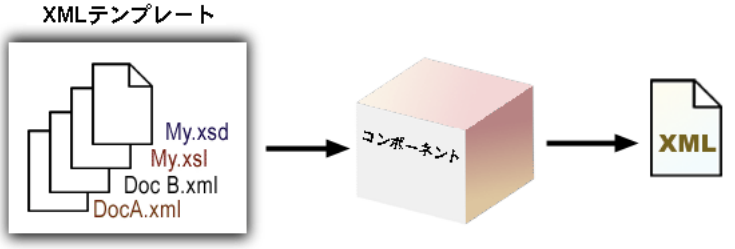
5 XML テンプレート

Novell exteNd Composer では、XML、XSL、DTD、または XSD、あるいはこれらすべての関連グループを、名前を付けた「テンプレート」に整理するという概念に大きく依存しています。

サンプル XML ドキュメント、ドキュメント定義、XSL スタイルシート、およびテンプレート

サンプル XML ドキュメント、ドキュメント定義、および XSL スタイルシートは、Composer のメインナビゲーションフレームのカテゴリペイン (上部ペイン) にリストされたコアオブジェクトタイプの 1 つである XML テンプレートに整理できるファイルのセットで構成されています。XML テンプレートの目的は、単に関連ドキュメントを単一の機能グループに整理することです。

XML テンプレートは、作成した後で、アクションモデルの作成時やコンポーネントのテスト時に使用します。標準の XML ドキュメント、作成した新しい XML ドキュメント、ビジネスパートナーからのサンプルデータドキュメント、XML ドキュメント定義、および (オプションで) XSL スタイルシート、DTD、またはスキーマファイル、あるいはこれらすべてをテンプレートに含め、コンポーネントの入力および出力を定義することができます。



入力 XML テンプレートと出力 XML テンプレートの両方を提供する

サンプル XML ドキュメントについて

サンプル XML ドキュメントは、コンポーネントまたはサービスで処理されるデータの視覚的なモデルで、同じ要素とデータが含まれます。たとえば、アプリケーションで企業 ABC の請求書が処理される場合、アプリケーションの作成時にサンプル請求書を使用します。サンプルには、処理される請求書と同じ XML 構造とデータが含まれます。

サンプルドキュメントは、作成するコンポーネントの入力と出力の両方に対して使用されます。アプリケーションを計画および設計する場合は、コンポーネントを作成し始める前に必要な入力サンプルドキュメントと出力サンプルドキュメントをすべて決定する必要があります。

必要なサンプルドキュメントのタイプは、次のとおりです。

- 処理するデータの要素と構造がすでに含まれている、標準の組織 (例 : cXML、OAG、および OFX) によって提供された XML ドキュメント。
- ユーザまたはビジネスパートナーにより作成された新しい XML ドキュメント。開始タグと終了タグのコンテンツ、およびこれらのタグ間のコンテンツから成る、階層構造に整理された XML 要素を設計します。

XML 検証ドキュメント (DTD およびスキーマ) について

DTD (ドキュメントタイプ定義) ファイルおよび XSD (XML スキーマ定義) ファイルは、XML ドキュメントを定義したり検証したりするために使用できます。スキーマおよび DTD では、必要な要素や要素間の構造関係など、ドキュメントの文法ルールが定義されます。

スキーマは、次を含む多くの点で DTD とは異なります。

- XSD ファイルは、W3C によって定義されたスキーマにそれ自体が準拠する真の XML ファイルです。逆に、DTD は、真の XML ファイルではありません。
- スキーマでは「データ入力」を強制できるため、たとえば、CCYY-DD-MM 形式の日付データが要素で必要な場合、この要件を指定 (および厳密に強制) することができます。
- スキーマではネームスペース宣言を使用できるため、特定のドキュメントボキャブラリに属するものとして要素を一意に識別できます。
- スキーマは、再利用性を最大限にするために、さらに明確になるよう設計されています。
- スキーマは、作成者が同じドキュメント内である文法ルールは厳密に実行し、他の文法ルールは緩慢に実行するよう指定できる点において、柔軟であるといえます。

- スキーマは、拡張可能であるため、作成者はまったく新しいカスタムデータのタイプを定義できます。

このような理由 (ただし、これらだけに限定されません) により、スキーマ (XSD ファイル) は、XML ドキュメントの定義および検証において、DTD ファイルに代わって次第に使用されるようになってきています。

XSL スタイルシートについて

コンポーネントで使用するファイルセットの一部として、XSL スタイルシートを含めることができます。XSL スタイルシートでは、XML ドキュメントの表示プロパティが定義されます。スタイルシートは、Composer の外部で作成または取得します。スタイルシートは、Web ブラウザに表示するページを作成しているアプリケーションのコンポーネントに対して役立つことがあります。

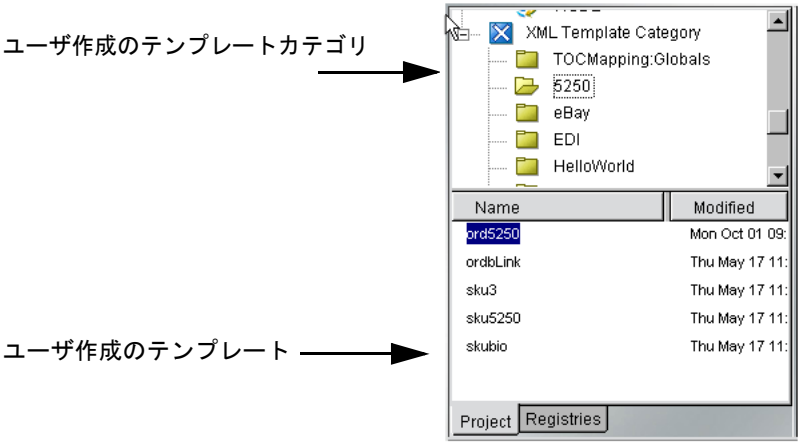
XML テンプレートについて

XML テンプレートは、Composer で作成され、前に説明したように、コンポーネントで使用できるファイルのセットを構成するサンプルドキュメント、ドキュメント定義、および XML スタイルシートが含まれます。XML テンプレートは、コンポーネントの設計段階の早い時点で作成し、作成するコンポーネントの入力および出力を指定するために使用します。

XML テンプレートは、多くのタイプのサンプルデータを使用およびテストできるよう、主に存在しています。エラーを発生させることなく、同じコンポーネントによって処理しなければならない、構造の異なった XML ドキュメントを 2 つ存在させることは可能です。たとえば、業界標準の注文書ドキュメントを入力として使用しているが、顧客のビジネスでは若干異なるバージョンのドキュメント (例 : オプションの要素の一部が欠けている) が使用される場合、この顧客のドキュメントをテストを目的としてコンポーネントにロードすることができます。コンポーネントでは、異なるドキュメントのバージョンを処理できなければならず、コンポーネントの入力として機能するテンプレートにすべてのサンプルを収集することで、複数の状況をテストすることが可能です。

テンプレートカテゴリについて

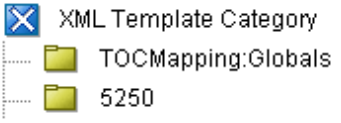
XML テンプレートのインスタンスは、テンプレートカテゴリに収集されます。テンプレートカテゴリにはユーザ割り当ての名前があり、このカテゴリは、Composer のナビゲーションフレームのカテゴリペインに表示されます。特定のカテゴリを構成するインスタンスは、カテゴリペイン下のインスタンスペインに表示されます (次を参照)。



アプリケーションには多数の入力ドキュメントや出力ドキュメントを含めることができるため、XML テンプレートカテゴリ内で整理が必要な場合があります。XML テンプレートカテゴリ内では、アプリケーションに最も適した方法でテンプレートを整理できます。たとえば、次に対してフォルダを作成することが可能です。

- 特定のビジネスプロセス (例 : 注文書の受け付け、請求書の送信、請求書の受信)
- 業界標準の XML ドキュメント

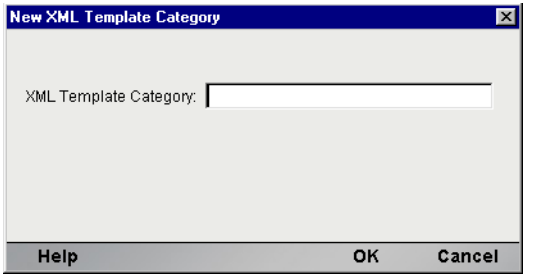
整理されたスキーマの例は、次のとおりです。



フォルダの目的は、サンプル XML ドキュメント、スキーマ、および XSL スタイルシートを含む可能性のある XML テンプレートを保存することです。

➢ XML テンプレートカテゴリを作成する

1. Composer のカテゴリペインで［**XML Template Category**］という名前を選択します。
2. マウスを右クリックして、［**New**］を選択します。

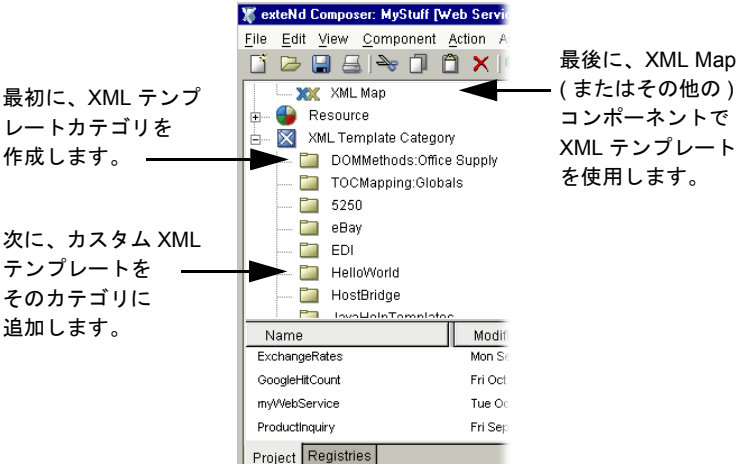


3 カテゴリの名前を入力し、[**OK**] をクリックします。

XML テンプレートの作成

XML テンプレートは、XML テンプレートカテゴリ内に存在します。XML テンプレートは単に関連ファイルをグループ化したもので、グループ化は再利用されるよう設計されています (1 つのテンプレートは、複数のコンポーネントで使用できます)。たとえば、テンプレートは、あるコンポーネントの入力、および別のコンポーネントの出力として使用できます。

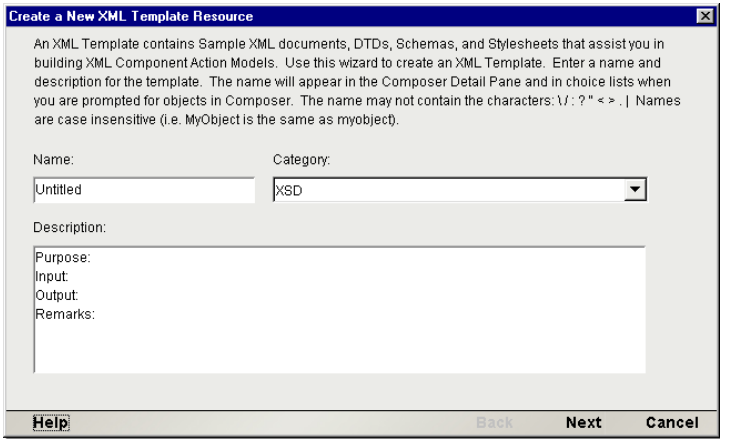
注記： 1 つのコンポーネントからの XML 出力は、サービス内の次のコンポーネントの入力として使用されることがあります。



➢ XML テンプレートを作成する

1 カテゴリペインで XML テンプレートカテゴリを選択します。

2 カテゴリでマウスを右クリックして、[**New**] を選択します。**Create a New XML Template** ウィザードが表示されます。

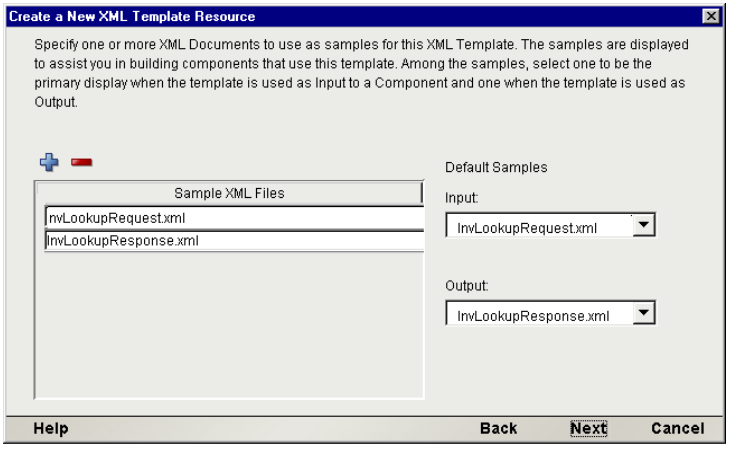


3 このテンプレートの「**名前**」を入力します。

4 ［Category］の下にあるプルダウンメニューで、すでに作成した既存の XML テンプレートカテゴリの中からカテゴリを 1 つ選択します (上の「XML テンプレートカテゴリを作成する」を参照)。

5 ［**Description**］に、テンプレートの使用目的の説明をプレーンテキスト形式で入力します (オプション)。

6 ［**Next**］をクリックします。ウィザードのドキュメント選択パネルが表示されます。

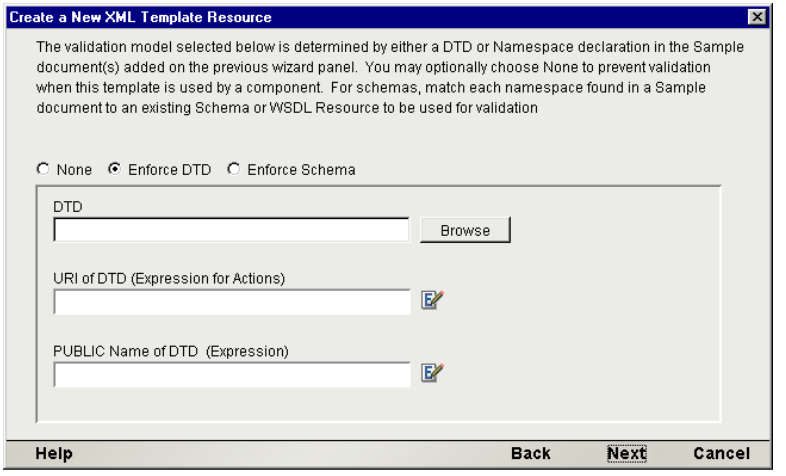


7 青緑色の［+］アイコンをクリックします。ファイルのナビゲーションダイアログボックスが表示されます。このダイアログボックスを使用して、このテンプレートに追加する XML ファイルをディスク上で指定します。この手順を繰り返して、XML ファイルを必要な数だけ追加します (［-］アイコンをクリックすると、特定のファイルがリストから削除されます)。

8 ［**Default for Input**］プルダウンメニューを使用して、このテンプレートを使用するコンポーネントのデフォルトの入力 DOM として表示するファイルを選択します (プルダウンメニューには、前の手順で作成したリストに表示されているファイルの名前が表示されます)。

9 ［**Default for Output**］プルダウンメニューを使用して、このテンプレートを使用するコンポーネントのデフォルトの出力 DOM として表示するファイルを選択します (プルダウンメニューには、左側のリストに表示されているファイルの名前が表示されます)。

10 ［**Next**］をクリックします。ウィザードのドキュメント検証パネルが表示されます。



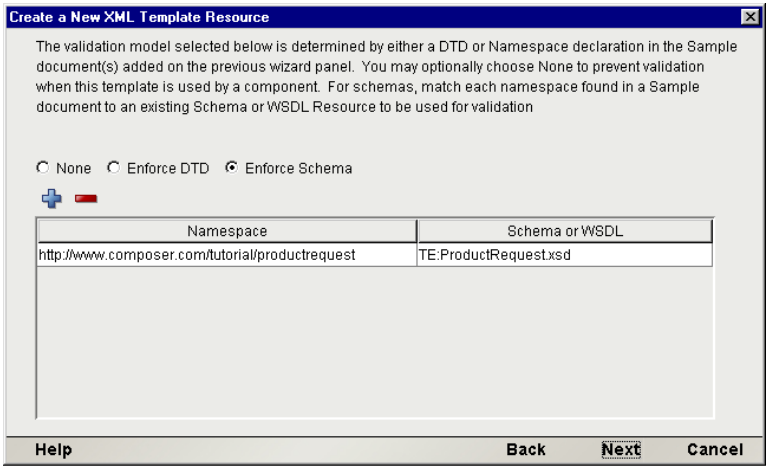
11 テンプレートドキュメントで実行するドキュメント検証のタイプを示すには、［**None**］、［**Enforce DTD**］、または［**Enforce Schema**］という 3 つのラジオボタンのいずれかをオンにします。ダイアログボックスの外観は、アクティブにしたボタンに基づいて変わります。Composer では、前のダイアログボックスで指定した XML テンプレートドキュメント (複数可) の検査に基づいて、正しいラジオボタンを設定しようとします。Composer により選択された内容は、いつでも上書きできます。ラジオボタンをオンにすると得られる結果は、次のとおりです。

［**None**］— XML ドキュメントの検証がアプリケーションで必要ない場合や、テンプレートドキュメントで指定した DTD 情報または XSD 情報を上書きする場合は、このオプションを選択します。

［**Enforce DTD**］— テキストフィールドで名前または URI、あるいはその両方が指定された DTD に対して、ドキュメントが検証されます。

注記： ランタイム時に DTD が動的に決定される場合、ECMAScript 式として URI を指定できます。プロジェクトを配備した後で PUBLIC DTD/ スキーマを使用する予定の場合は、［PUBLIC Name of DTD］フィールドに入力しなければなりません。

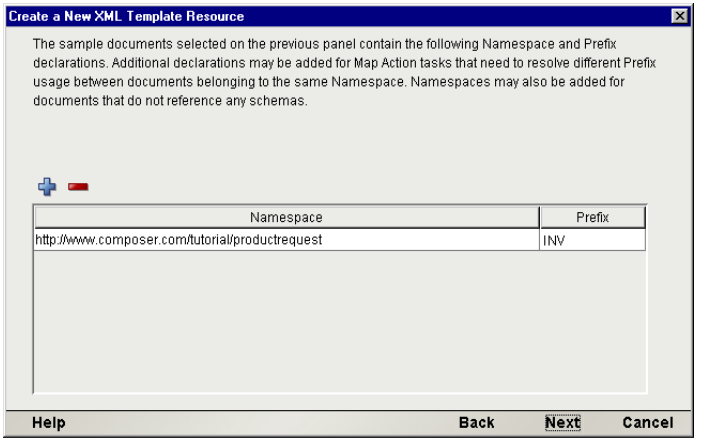
［**Enforce Schema**］— 指定された XSD ファイルまたは WSDL ファイルに対して、ドキュメントが検証されます (次の図を参照)。



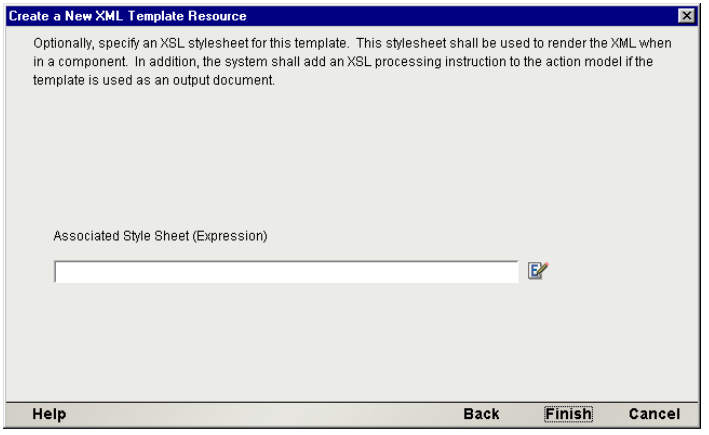
注記： Composer では、サンプルドキュメントを自動的に検索して、サンプルドキュメント内で宣言されたネームスペース (該当する場合) と、参照する **.xsd** ファイルをすべて見つけます。ネームスペースとそれらに関連するスキーマは、上のダイアログボックスに自動的に表示されます。ほとんどの場合、自分でダイアログボックスに入力する必要はありません。正しいスキーマリソースの隣にネームスペースが表示されない場合は、右側のプルダウンメニューから適切なスキーマリソースを選択します (つまり、プルダウンメニューを使用して、正しいスキーマと正しいネームスペースを関連付けます)。

12 ［**Next**］をクリックして、次のパネルに移動します。

13 使用しているドキュメントにネームスペース情報が含まれている場合は、ネームスペースとそれに対応するプリフィックスがこのダイアログボックスで要約されます。ネームスペース宣言をさらに追加する必要がある場合は (たとえば、スキーマを参照しないドキュメントに対して)、プラス記号 (+) のアイコンを使用してこれを行います。



14 ［**Next**］をクリックします。ウィザードのスタイルシート選択ペインが表示されます。



15 (オプション) 出力ドキュメントに関連付けるスタイルシート (**.xsl** ファイル) の名前を入力します (この情報がランタイム時に式から決定される場合は、適切な ECMAScript 式を入力します)。ファイル名は、ECMAScript 文字列として処理されるため、引用符で囲む必要があります。

16 XSL スタイルシートを指定すると、次が実行されます。

- 新しいサービスまたはコンポーネントを作成し、出力 DOM に対してテンプレートが使用される場合は、Composer によって、新しいコンポーネントのアクションモデルに Function アクションが自動的に追加されます。Function アクションでは、出力 XML ドキュメントに *processing instruction* を追加し、ドキュメントの XSL スタイルシートを指定します。

- processing instruction で参照されるスタイルシートは、この XML テンプレートで指定したスタイルシートです。

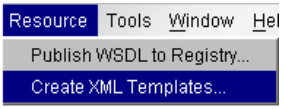
17 ［**Finish**］をクリックして、XML テンプレートを作成します。

WSDL からの XML テンプレートの作成

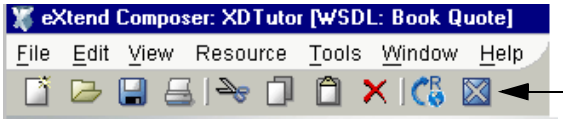
exteNd Composer に外部 WSDL がダウンロードされたら、作業コンポーネントを作成するために、WSDL のメッセージに対応する XML テンプレートが通常は必要となります。また、WSDL に対して検証できる XML サンプルを作成する必要があります (このようなテンプレートは、その後、WS 交換で使用されるアクションを作成するためにコンポーネントで使用できます)。Composer を使用すると、この操作を実行することが可能です。サービスに対する WSDL リソースがある場合は、単に WSDL リソースを開きます。Composer のツールバーとメニューが変更され (次の図を参照)、ボタンをクリックすると、XML スタブドキュメント (テンプレートドキュメント) を作成できるようになります。

開いている WSDL リソースから XML テンプレートドキュメントを生成するには、次の 2 つの方法があります。

- ［**Resource**］メニューから、［**Create XML Template**］を選択します。



- ツールバーの［**Create XML Templates**］ボタンをクリックします。

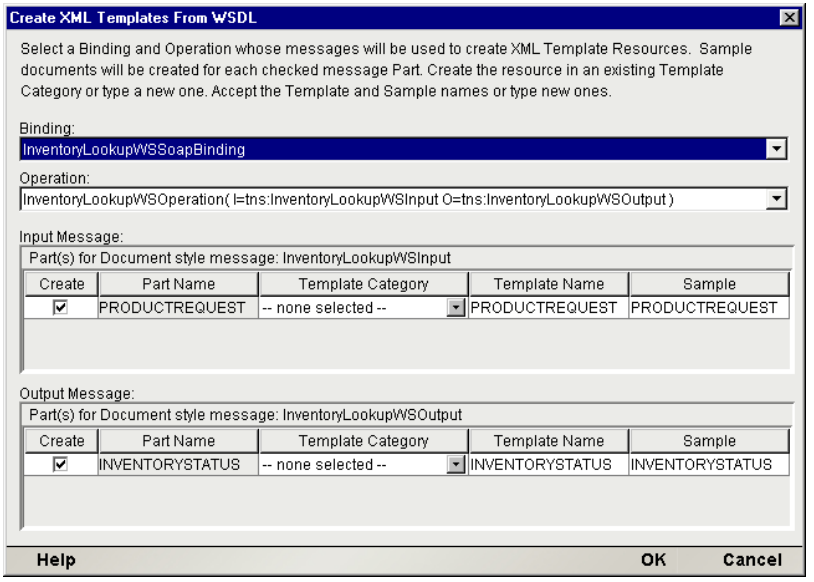


注記： 作成したサンプルには、要素データは含まれていません。このため、テストを目的として、さまざまな要素にサンプルデータを入力しなければならない場合があります。**##any** ネームスペースまたは **##other** ネームスペースを要素が参照する場合、サンプルは不完全となるため、手動でサンプルを完了する必要があることにも注意してください。

テンプレートドキュメントは、RPC バインドだけでなく、ドキュメントスタイルに対してもこのように作成できます。

➢ WSDL から XML テンプレートを作成する

1 メインメニューから［**Resources**］>［**Create XML Templates**］の順にクリックするか、またはツールバーのボタンをクリックします。ダイアログボックスが表示されます。



2 XML テンプレートを作成するためのソースとして、ドロップダウンリストから［**Service/Port or Binding**］を選択します。

3 XML テンプレートを作成するためのソースとして、ドロップダウンリストから［**Operation**］を選択します。

4 ダイアログボックスの下部は、入力と出力の「**サンプル**」に分割されます。名前は、デフォルトでメッセージ名から決定されます。デフォルトの名前が希望のサンプル名でない場合は、新しい名前を入力します。

5 ドロップダウンリストから、「**カテゴリ**」のタイプを選択します。また、新しいカテゴリを作成することもできます。

6 ドロップダウンリストから、「**テンプレート**」のタイプを選択します。また、新しいテンプレートを作成することもできます。

7 ［**OK**］をクリックして終了します。

XML テンプレートのインポート

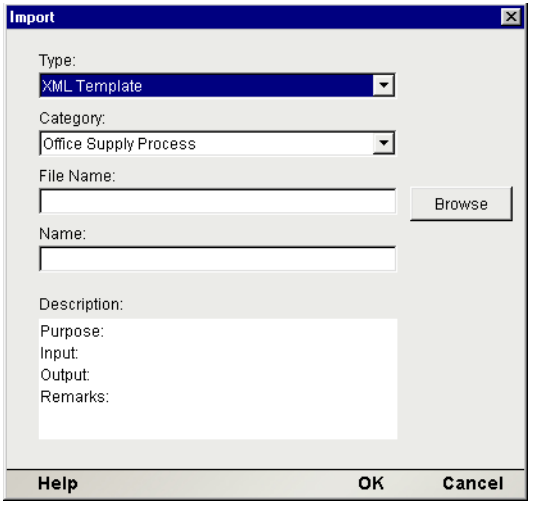
別のプロジェクトに対して XML テンプレートをすでに作成した場合は、この XML テンプレートを現在のプロジェクトにインポートできます。

➢ XML テンプレートをインポートする

1 Composer のカテゴリペインで、テンプレートインスタンスに関連付ける「XML テンプレートカテゴリ」を選択します。

2 マウスを右クリックして、コンテキストメニューから［**Import**］を選択します。ダイアログボックスが表示されます。

注記： ［Import］コマンドが選択されていない場合、これは、サブプロジェクトに属するテンプレートカテゴリをすでに選択していることに起因しています。そのため、この操作を実行することはできません。テンプレートドキュメントをサブプロジェクトにインポートする必要がある場合は、現在のプロジェクトを閉じてください。そして、サブプロジェクトを開き、そのサブプロジェクトにテンプレートを追加して保存した後、閉じます。その後、作業していた元のプロジェクトに戻ります。



3 ［**Type**］で［**XML Template**］を選択します。

4 XML テンプレートカテゴリを選択します。

5 ［**Browse**］を使用して、「**ファイル名**」の場所を選択します。

6 「**名前**」を入力します。

7 オプションとして、「**説明**」を追加します。

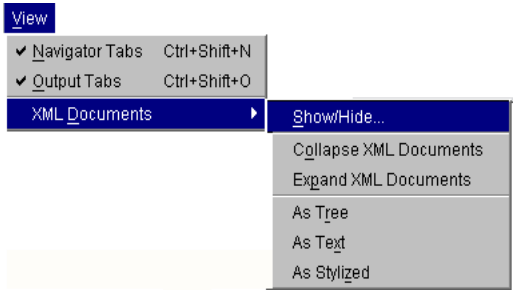
8 ［**OK**］をクリックします。

XML ドキュメントの表示および非表示

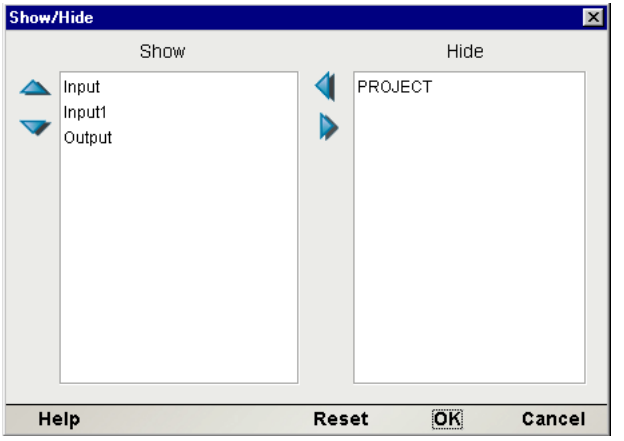
Composer のメインウィンドウで作業する際には、XML ドキュメントの表示レベルを切り替えると便利です。

➢ XML ドキュメントの表示レベルを切り替える

1 Composer のメインメニューバーで、［**View**］>［**XML Documents**］>［**Show/Hide**］の順に選択します。



ダイアログボックスが表示されます。



2 ［**Show/Hide**］ダイアログボックスには、XML ドキュメントの名前が表示されます。

注記： 「入力」XML ドキュメントおよび「出力」XML ドキュメントは、デフォルトで［**Show**］列に表示されます。Component アクションの結果として作成された DOM は、デフォルトで非表示になります。

3 ［**Hide**］列で、「表示する」XML ドキュメントを選択し、「**左矢印**」ボタンをクリックします。反対に、［**Show**］列で、「非表示にする」XML ドキュメントを選択し、「**右矢印**」ボタンをクリックします。

4 「トップドキュメント」として表示する XML ドキュメントを選択し、［**Show**］列の最上層ドキュメントとしてドキュメントが表示されるまで［**Promote**］ボタンをクリックします。

5 引き続き［**Show**］列で XML ドキュメントを選択し、上向き三角形と下向き三角形のボタンを使用して、XML ドキュメントを希望の順序に移動します。

6 ［**OK**］をクリックします。ダイアログボックスが閉じ、コンポーネントエディタのデータペインが順に並べ替えられます。

XML テンプレートエディタ

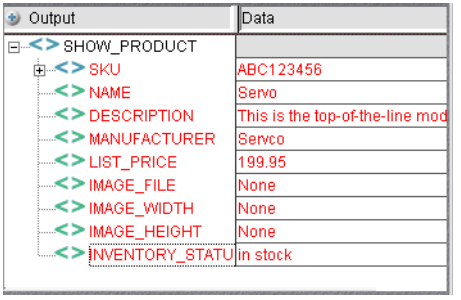
XML テンプレートエディタを使用すると、外部エディタを使用するのではなく、Composer でテンプレートを編集できます。

テンプレートエディタとコンテキストメニューでのドキュメントの表示

メインメニューバーの［View］オプションを使用すると、DOM で XML 情報を表示する方法を選択できます。表示方法は、ツリー、テキスト、または様式化された形式の中から選択できます。各ビューには、マウスを右クリックするとアクセスできる独自のコンテキストメニューがあります。

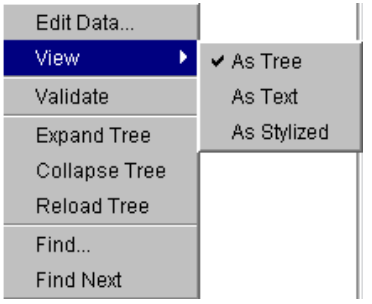
ツリービューおよびコンテキストメニューオプション

デフォルトのビューでは、DOM がツリーとして表示されます (次を参照)。



このビューでは、要素と属性の「値」(つまり、ドキュメントデータ) を編集できますが、DOM 構造は編集できません。

マウスを右クリックしてアクセスできるコンテキストメニューコマンドは、次の図のとおりです。



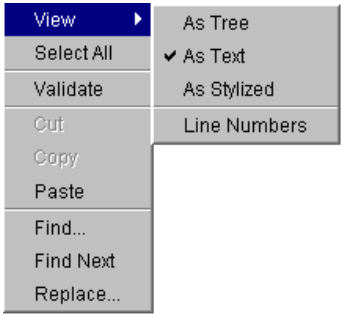
テキストビューおよびコンテキストメニューオプション

テキストビューでは、構造要素を含む完全な XML ファイルを表示したり編集したりできます。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SHOW_PRODUCT>
  <SKU DEFAULT="12345">ABC123456</SKU>
  <NAME>Servo</NAME>
  <DESCRIPTION>This is the top-of-the-line model
  <MANUFACTURER>Servco</MANUFACTURER>
  <LIST_PRICE>199.95</LIST_PRICE>
  <IMAGE_FILE>None</IMAGE_FILE>
  <IMAGE_WIDTH>None</IMAGE_WIDTH>
  <IMAGE_HEIGHT>None</IMAGE_HEIGHT>
  <INVENTORY_STATUS>in stock</INVENTORY_STATUS>
</SHOW_PRODUCT>
```

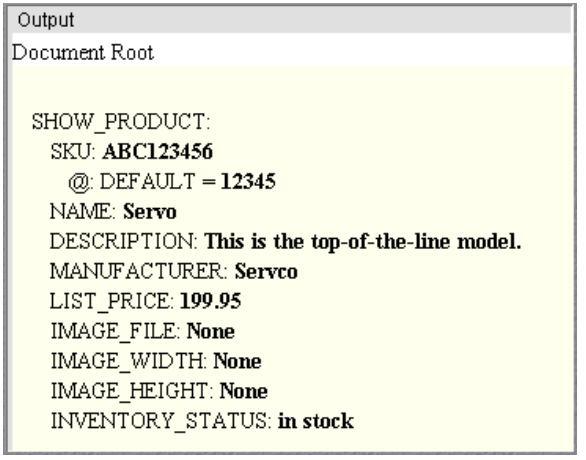
テキストビューでは、comment、processing instruction、DOCTYPE declaration などの、入力 DOM、一時 DOM、または出力 DOM の非コンテンツモデル部分を簡単に検査できます。

マウスの右ボタンを使用してアクセスできるコンテキストメニューオプションは、次の図のとおりです。



様式化されたビューおよびコンテキストメニューオプション

様式化されたビューが選択されたペインである場合、次のような DOM コンテンツが表示されます。

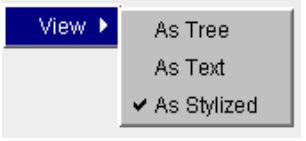


このビューでは、XML コンテンツの「レポート」スタイルの概要が表示されるため、すべての属性や要素のコンテンツを一目で確認できます。このビューでは、次のアルゴリズムを使用して XML を表示します。

このドキュメントコンポーネントに関連付けられたスタイルシートが存在する場合は、式を評価してその式を使用します。

これに失敗した場合は、デフォルトのスタイルシートである **com/sssw/b2b/dt/default.xsl** を使用します。

マウスの右ボタンを使用してアクセスできるコンテキストメニューオプションは、次の図のとおりです。



変更の保存およびドキュメントの印刷

ドキュメントに行った変更内容は、メインメニューバーから［**File**］>［**Save**］の順に選択した場合にのみ反映されます。

XML ドキュメントを印刷するには、メインメニューバーから［**Select File**］>［**Print**］の順に選択します。ドキュメントコンポーネントがテンプレートに基づいてフォーマットされます。

XML テンプレートの操作

作成した XML テンプレートは、それぞれ XML テンプレートカテゴリ内に存在します。XML テンプレートの名前と作成日を表示するには、XML テンプレートカテゴリを選択します。カテゴリの XML テンプレートは、すべて Composer の詳細ペインにリストされます。各テンプレートには、テンプレートを操作できるようにするコンテキストメニューがあります。

XML ドキュメントの表示

各 XML テンプレートには、1 つまたは複数のサンプルドキュメントが含まれています。サンプルドキュメントは、XML エディタ (Composer の外部の別のアプリケーション) で開くことができます。

➢ XLM エディタでサンプルドキュメントを表示する

1 XML テンプレートでマウスを右クリックします。

2 ［**Edit Sample**］を選択し、編集するサンプルドキュメントを選択します。

XML テンプレートの編集

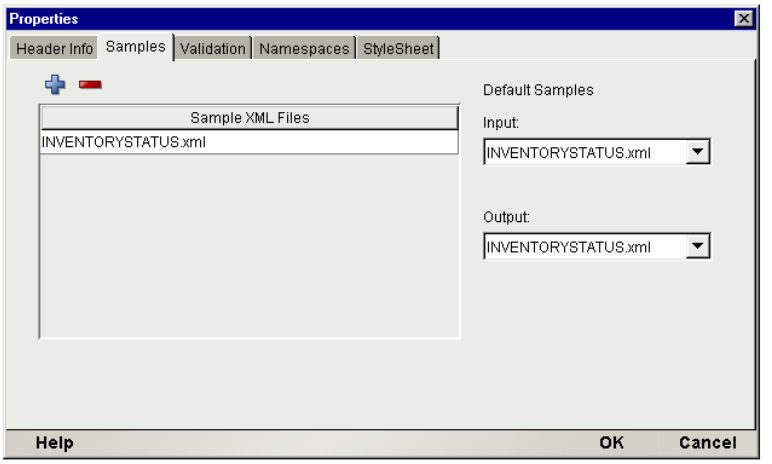
XML テンプレートは、サンプルドキュメント、スキーマ、および XSL スタイルシートを追加したり削除したりすることによって変更できます。

➢ XML テンプレートを編集する

- XML テンプレートインスタンスをダブルクリックして、［**Properties**］ページを開きます。

  または

- XML テンプレートインスタンスをシングルクリックしてマウスを右クリックし、コンテキストメニューから［**Properties**］を選択します。



注記： ［Properties］ページでテンプレートの名前を変更すると、［**Save As**］操作が行われるため、元のテンプレートはそのまま維持され、新しい名前を持つコピーが作成されます。XML テンプレートの名前のみを変更するには、コンテキストメニューの［**Rename**］オプションを使用します。

XML テンプレートの削除

XML テンプレートを作成すると、Composer では、元の XML、ドキュメント定義、および XSL ファイルのコピーが作成され、適切な XML カテゴリの［Imports］ディレクトリに配置されます。XML テンプレートを削除すると、元のファイルではなく、コピーが削除されます。XML テンプレートを削除するには、XML テンプレートを右クリックし、［**Delete**］を選択します。

異なるカテゴリへの XML テンプレートの移動

➢ **あるカテゴリから別のカテゴリに XML テンプレートを移動する**

1. 移動するテンプレートを選択します。
2. マウスを右クリックし、［**Cut**］または［**Copy**］のいずれかを選択します。
3. Composer のカテゴリペインで、別の XML カテゴリをクリックします。
4. 詳細ペインでマウスを右クリックし、［**Paste**］を選択します。

XML テンプレートの名前変更

➢ **XML テンプレートの名前を変更する**

1. 名前を変更するテンプレートを選択します。
2. マウスを右クリックして、[**Rename**] を選択します。
3. 新しい名前を入力します。
4. [**OK**] をクリックします。

注記： 前の手順に従ってテンプレートの名前を変更するようにしてください。[**Properties**] ページでテンプレートの名前を変更すると、[**Save As**] 操作が行われるため、元のテンプレートはそのまま維持され、新しい名前を持つコピーが作成されます。

ハードドライブ上で保存される XML テンプレートの場所の概要

XML テンプレートは、プロジェクトの一部として保存されます。プロジェクトファイルの保存場所の詳細については、73 ページ「プロジェクトファイルの保存場所について」を参照してください。

注記： テンプレートで使用されるサンプル、定義、および XML スタイルシートのコピーが、フォルダ内に保存されます。このため、元のドキュメントは変更されません。

6 XML Map コンポーネントの作成

Composer XML Map コンポーネントは、いろいろな意味において、Composer のコンポーネントタイプの中でも最も単純でありながら重要であるものです。このコンポーネントは、出力ドキュメントへの入力ドキュメントの XML 変換を実行するために使用します。有用な Composer サービスを作成する場合は、XML Map コンポーネント (および関連リソース) の機能について理解することが必要です。

この章では、XML Map コンポーネントについて紹介し、exteNd Composer サービス内での XML Map コンポーネントの動作についても説明します。この章を読むと、XML Map コンポーネントの構成要素と機能、および XML Map コンポーネントの設計 / 作成 / 使用方法を習得できます。

XML Map コンポーネントとは

XML Map コンポーネントは、1 つまたは複数の XML ドキュメントを入力として受け付け、入力で動作するアクションの集合を使用し、XML ドキュメントを出力として返すオブジェクトです。

図 6-1

XML Map コンポーネントでは、1 つの XML ドキュメントから別の XML ドキュメントへのデータのマッピングや転送など、単純なデータ操作を実行できます。また、データおよびドキュメント構造の両方の変換など、洗練された操作を実行することもできます。さらに、XSL の処理、メールの送信、および HTTP プロトコルを使用した XML ドキュメントのポストと受信を行う XML Map コンポーネントを作成することも可能です。

コンポーネントの背後にある概念は、1 つまたは複数の XML ドキュメントを入力として渡し、これらの入力を処理して、1 つの出力 XML ドキュメントを返すことです。その後、出力 XML ドキュメントは、他のコンポーネントに対する入力として使用されたり、サービスの最終出力として返されたりします。このようにして、サービス内で連携し、完全な B2B ソリューションを提供するコンポーネントを作成できます。

XML テンプレートサンプルドキュメントを使用した XML Map コンポーネントの作成

XML Map コンポーネントを作成する際に使用する XML ドキュメントは、実行されているアプリケーションで処理される実際のドキュメントのサンプルです。サンプルドキュメントは、Composer の XML テンプレートオブジェクトに追加されます。処理されるドキュメントとまったく同じ構造およびデータ表記となるドキュメントのサンプルを使用します。作成を目的として使用するドキュメントと、コンポーネントによって実際に処理されるドキュメントの相違点は、次の図のとおりです。

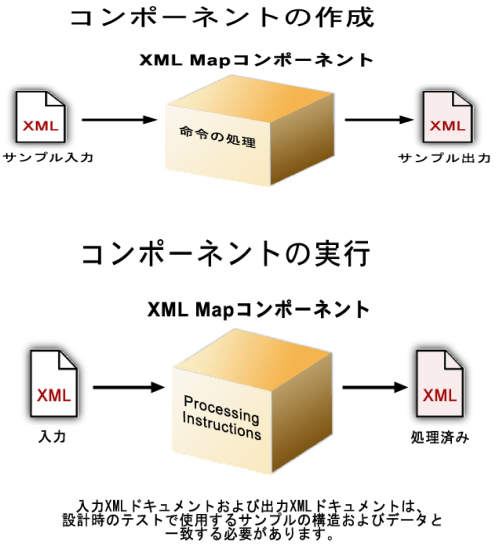


図 6-2

コンポーネントの作成に使用されたサンプルは、ランタイム時に実際に使用されたり、その名前で参照されることはありません。これらは、単に操作される構造とデータを表したテンプレートです。サンプルは、正しいランタイム操作を実行する処理アクションを作成する上で一時的に役立ちます。XML データを処理するためにランタイム時に exteNd によって実際に使用されるのは、DOM と呼ばれる XML ドキュメントをオブジェクトで表したものです。

DOM とは

XML Map コンポーネントエディタでは、サンプルドキュメントは、DOM (ドキュメントオブジェクトモデル) として知られている、W3C 推奨の形式で表されます。DOM は、ソフトウェアプログラムのメモリ内のオブジェクトとして作成された XML ドキュメントです。これにより、オブジェクトを操作するための標準的な方法が提供されます。DOM を使用すると、Composer では、XML ドキュメントの作成や XML ドキュメント構造内での移動、および要素やコンテンツの追加 / 変更 / 削除を行うことができます。XML ドキュメント内にあるものは、すべて DOM メソッドを使用して操作できます。Composer では、DOM バインド仕様に対して W3C ECMA 推奨の DOM メソッドをすべてサポートしています (**http://www.w3.org/** を参照)。

注記 : 一部のダイアログボックスでは、DOM はメッセージと呼ばれます。

DOM 構造の概要

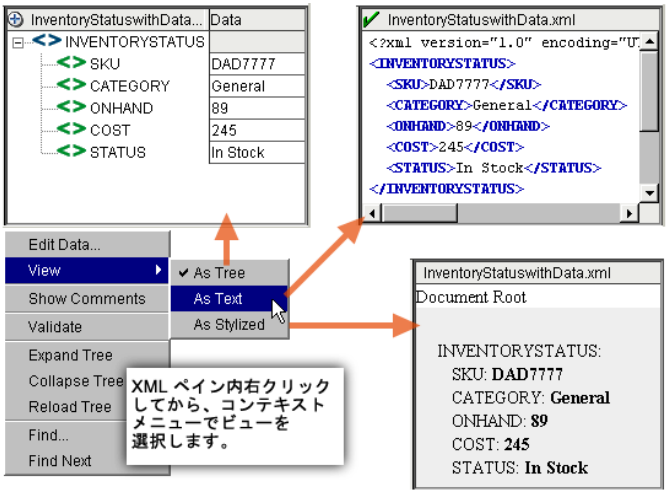
XML Map コンポーネントエディタがアクティブになっていると、すべてのサンプルドキュメントが DOM に変換されます。

DOM は、次の理由のために使用されます。

- DOM では、要素を簡単、明瞭に選択できるよう、XML 構造要素の名前付けおよび整理には一般的な方法を使用している
- DOM の構造要素は、操作することができる
- DOM は、ランタイム時に作成したり操作したりできる構造である

DOM は、階層的に整理されます (つまり、ツリー構造を形成しています)。特定の DOM が構成される方法を理解するには、異なる視点から DOM を表示できると便利な場合があります。たとえば、ある状況では、DOM の基盤となる XML ドキュメントの加工されていないテキストを表示すると便利です。また、別の状況では、ドキュメントで (要素名や属性名ではなく) データの要約ビューを表示したい場合もあります。Composer では、ツリー、テキスト、および要約による表示を切り替えるために、必要に応じてビューを変更することが可能です。

ビューを変更するには、単に任意の DOM ウィンドウ内でマウスを右クリックして、希望のビューを［View］サブメニューから選択します。使用できる 3 つのビュータイプは、次のとおりです。



DOM の要素は、XML ドキュメントでタグによって定義されます。たとえば、前の例では、<INVENTORYSTATUS> と呼ばれる XML ドキュメントに要素タグがあります。このタグには、終了タグ (</INVENTORYSTATUS) があります。<INVENTORYSTATUS> および </INVENTORYSTATUS> 内のすべての構造要素 (<SKU> など) は、DOM ツリーのさらに下層 (または、DOM 用語では「子孫」) のレベルで表されます。

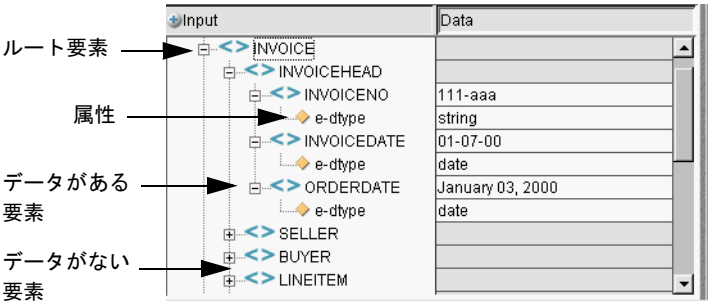
すべての要素名では、「大文字と小文字が区別」されます (つまり、<INVOICENO> と <InvoiceNo> は同じではありません)。

DOM ツリーの要素は、「ノード」と呼ばれます。ノードの集合は階層で表され、次の命名規則によって言及されます。

表 6-1

ノードタイプ	説明
ルート	DOM ツリーでの最上位の要素。他のすべての要素は、この要素の子孫です。1 つの要素のみを使用できます。
子孫	別のノードの下層にある (別のノード内に含まれる) ノードすべて。
チャイルド	あるノードの直接の子孫。
兄弟	同じペアレントノードを共有するノードすべて。
先祖	別のノードの上層にある (別のノードを含む) ノードすべて。
ペアレント	あるノードの直接の先祖。
リーフ	子孫のないノードすべて。

DOM の各要素タイプには、独自のアイコンがあります (次を参照)。



ランタイム時の DOM の使用

コンポーネントでは、実行中のアプリケーションで、DOM を通して互いにデータの受け渡しを行います。ランタイム時には、コンポーネントが実行されると DOM が渡されます。渡された DOM は、操作できる入力 DOM となります。コンポーネントのマッピングアクションをそれぞれ実行すると、要素ごとに出力 DOM が作成されます。

ランタイム中の DOM の動作

コンポーネントを初めて開いた場合、元のサンプルが入力 DOM および出力 DOM にロードされます。アニメーションを開始すると、入力 DOM はそのままの状態となりますが、一時 DOM および出力 DOM は、このような DOM に最初に含まれていたデータからクリアされます。実行の最後では、データがすべての DOM に表示されます。

異なる DOM タイプの作成

XML Map コンポーネントを作成し、この XML Map コンポーネントに対する入力 XML テンプレートと出力 XML テンプレートを選択します。ただし、コンポーネントエディタ内では、次の操作を実行することもできます。

- テンプレートを使用せずに出力DOMを作成する (127ページ「テンプレートを使用しない出力 DOM の作成」を参照)
- 一時 DOM を作成する (128 ページ「一時 DOM の作成」を参照)
- XML Interchange アクションを使用して、外部 XML ドキュメントから DOM を動的に作成する

XML Map コンポーネントの作成

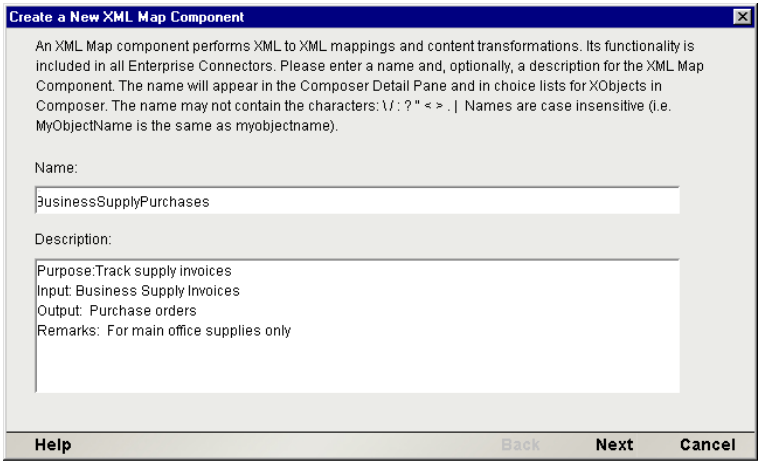
XML Map コンポーネントを作成する場合の最初の手順は、コンポーネントの XML テンプレートを指定することです。詳細については、89 ページ「XML テンプレートの作成」を参照してください。

XML テンプレートを指定すると、コンポーネントによって処理される入力および出力を表すテンプレートのサンプルドキュメントを使用して、コンポーネントを作成できます。

注記： 他のさまざまなコンポーネントタイプ (JDBC コンポーネント、JMS コンポーネントなど) については、該当する Enterprise Connect 製品のユーザガイドで詳細に説明されていますが、すべてのコンポーネントの作成時や編集時に使用される基本的な原理は同じです。また、XML Map コンポーネントで使用できるさまざまな基本的なアクション (次の章を参照) も、その他すべての Composer コンポーネントタイプで利用できます。

➢ XML Map コンポーネントを作成する

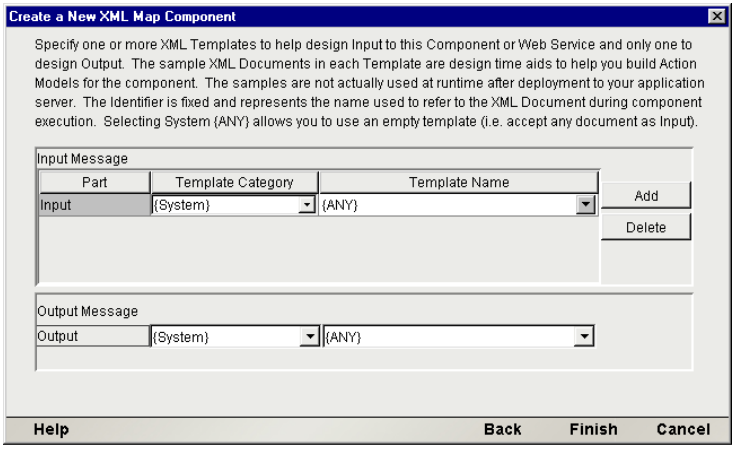
1 Composer の［File］メニューから、［**New xObject**］、［**Component**］、［**XML Map**］の順に選択します。［New xObject］ダイアログボックスが表示されます。



2 コンポーネントの「**名前**」を入力します。

3 オプションとして、「**説明**」情報を入力します。

4 ［**Next**］をクリックします。新しいパネルが表示されます (次を参照)。



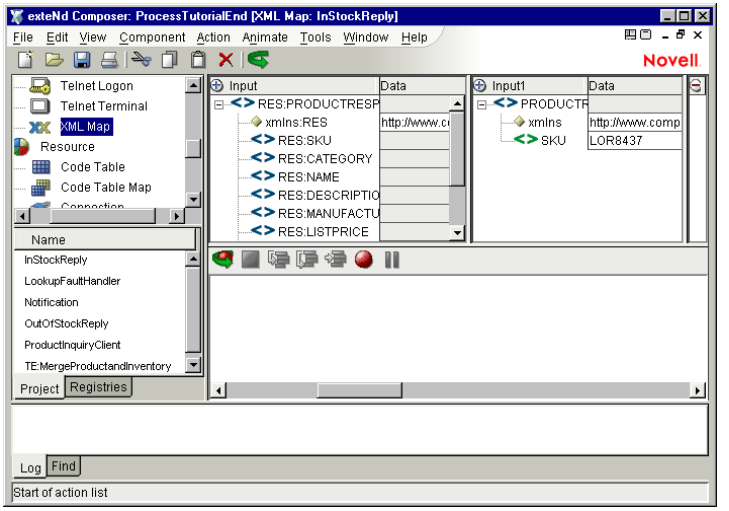
5 入力テンプレートおよび出力テンプレート(メッセージとも呼ばれます)を指定します。

- デフォルトのカテゴリと異なる場合は、[**Template Category**]を選択します。
- 選択した[**Template Category**]にある XML テンプレートのリストから[**Template Name**]を選択します。
- 入力 XML テンプレートをさらに追加するには、[**Add**]をクリックして、手順 2 から 4 を繰り返します。
- 入力 XML テンプレートを削除するには、エントリを選択して[**Delete**]をクリックします。

6 XML テンプレートを出力として選択します (出力 DOM の名前は「Output」です)。

注記： 出力テンプレートとして {ANY} を選択すると、構造が含まれない出力 XML テンプレートを指定できます。詳細については、127 ページ「テンプレートを使用しない出力 DOM の作成」を参照してください。

7 [**Finish**]をクリックします。コンポーネントが作成され、XML Map コンポーネントエディタが表示されます。



ネームスペースおよび出力 DOM

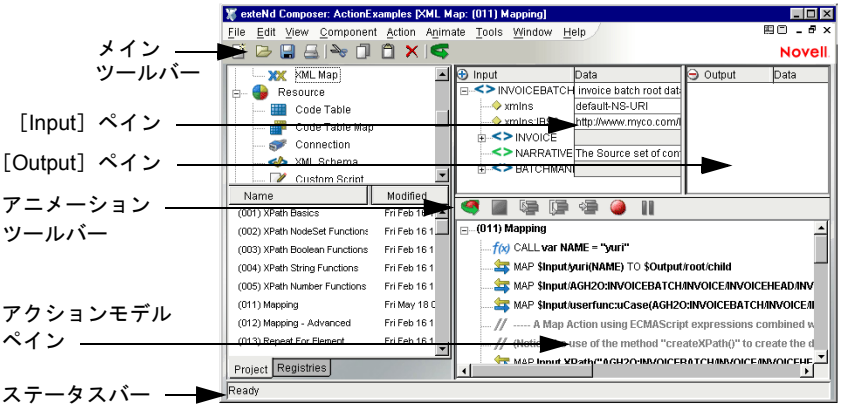
新しいXML Mapコンポーネントを作成し、その出力テンプレートでネームスペースが使用される場合は、ネームスペース URI が出力 DOM の属性にマップされ、新しい「Map アクション」がアクションリストに自動的に配置される点に注意してください。この Map アクションは、「削除」したり、誤って上書きしたりしないようにしてください。Component アクションを namespace-URI Map アクションの「ダウンストリーム」に配置し、Component アクションによって結果が出力 DOM に返されると、上書きが実行される可能性があります。この場合の解決法は、元の Map アクションを、Component アクションのスポットダウンストリームに (切り取りと貼り付けを使用して) 移動することです。

XML Map コンポーネントエディタの概要

XML Map コンポーネントエディタは、すべての入力と出力の構造やデータのマッピング、変換、および転送を指定する場所です。

XML Map コンポーネントエディタを使用すると、コンポーネントの入力、出力、およびアクションに対する論理的な作業環境が提供されます。XML Map コンポーネントエディタは、複数のマッピングペインと 1 つのアクションモデルペインで構成されています。マッピングペインには、サンプル入力ドキュメントとサンプル出力ドキュメントの DOM が表示されます。アクションモデルペインには、マッピングペインで動作するアクションが表示されます。

メニューとツールバー、1 つの入力 DOM、1 つの出力 DOM、および複数のアクション (アクションモデルペイン内) が表示された XML Map コンポーネントエディタは、次の図のとおりです。



メニューおよびツールバーについて

XML Map コンポーネントエディタには、独自のメニューおよびツールバーオプションがあります。Composer メニューのオプションに加え、XML Map コンポーネントエディタメニューには、次のコンポーネント固有のオプションもあります。

表 6-2

XML Map コンポーネントメニュー	オプション
［File］	**Composer の［File］メニューからと同じように、この［File］メニューからは、任意のタイプのコンポーネントを作成したり、開いたり、削除したりできます。** さらに、XML Map コンポーネントの［File］メニューからは、次の操作も実行できます。 ［**Save**］：コンポーネントで入力、出力、およびアクションを保存します。 ［**Save as...**］：新しい名前でコンポーネントを保存し、入力、出力、およびアクションを変更できます。132 ページ「コンポーネントの保存」を参照してください。 ［**Save XML Output As...**］：出力 DOM 構造を XML ドキュメントに保存できます。133 ページ「XML ドキュメントとしての DOM の保存」を参照してください。 ［**Load XML Sample...**］：コンポーネントをテストするために、テンプレートから DOM に他のサンプルドキュメントをロードできます。131 ページ「サンプルドキュメントのロード」を参照してください。 ［**Properties**］：コンポーネントのテンプレートおよびその他の情報を表示できます。135 ページ「コンポーネントのプロパティの表示」を参照してください。 ［**Print**］：コンポーネントを印刷できます。
［Edit］	アクションを元に戻したり、コンポーネントエディタの任意の場所でテキストの切り取り、コピー、および貼り付けを行ったりすることができます。さらに、次の操作も実行できます。 ［**Find**］：マッピングペインで要素を検索したり、アクションモデルでテキストを検索したりします。123 ページ「DOM 要素の検索」を参照してください。 ［**Find Next**］：マッピングペインで次の要素を検索したり、アクションモデルで次のテキストを検索したりします。124 ページ「次の DOM 要素の検索」を参照してください。 ［**Replace**］：選択したテキストをアクションモデルのみで置換します。125 ページ「アクションモデルでのテキストの置換」を参照してください。

XML Map コンポーネントメニュー	オプション
［View］	［**Navigator Tabs**］: Composer メインウィンドウの左側にあるナビゲーションフレームの表示レベルを切り替えます。ナビゲーションフレーム全体を表示または非表示にする場合に、このコマンドを使用します。 ［**Output Tabs**］: Composer メインウィンドウの下部にあるメッセージフレームの表示レベルを切り替えます。メッセージフレーム全体を表示または非表示にする場合に、このコマンドを使用します。 ［**Expand XML Documents**］: すべての DOM ノードをすべてのマッピングペインで表示します。 ［**Collapse XML Document**］: ルートノード以外のすべての DOM ノードをすべてのマッピングペインで非表示にします。 ［**Expand XML Documents**］: すべての DOM ノードをすべてのマッピングペインで表示します。 ［**XML Documents**］: DOM ペインでの DOM コンテンツの表示方法のコントロール (［**As Tree**］、［**As Text**］、［**As Stylized**］) を表示します。 ［**Window Layout**］: ［Component］ウィンドウ内で表示する DOM や、DOM ペインとアクションモデルペインの整列方法を指定できます。116 ページ「コンポーネントエディタでの［Window Layout］および［Show/Hide］の使用」を参照してください。
［Component］	［**Reload XML Documents**］: DOM ペインに現在表示されている内容をクリアして、入力テンプレートおよび出力テンプレートから元のサンプルを再ロードします。130 ページ「XML ドキュメントの再ロード」を参照してください。 ［**Execute**］: テストを目的として、コンポーネントを最初から最後まで実行します。
［Action］	このメニューには、アクションモデルに追加できるアクションがすべて含まれています。
［Animate］	このメニューには、コンポーネントをテストするために使用できる処理アニメーションツールがすべて含まれています。ツールバーには、アニメーションツールを実行できるようにするボタンがあります。詳細については、301 ページ「アニメーションツールを使用したテスト」を参照してください。

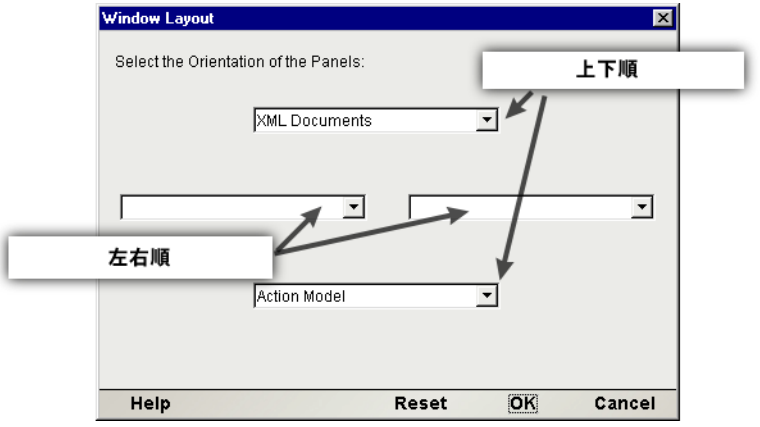
メニュー項目の多くは、ツールバーにあります。ツールバー項目の上にマウスを置くと、その項目の簡単な説明が表示されます。

コンポーネントエディタでの［Window Layout］および［Show/Hide］の使用

コンポーネントエディタのペインは、コンテンツを操作しやすくするために、表示、非表示、位置変更、およびサイズ変更することができます。［Window Layout］オプションおよび［Show/Hide］オプションは、［View］メニューから使用します。

➢ コンポーネントエディタのペインを整列する

1 ［View］メニューから、［**Window Layout**］を選択します。［Window Layout］ダイアログボックスが表示され、ウィンドウ内でパネルの位置を調節できます。



2 XML ドキュメントおよびアクションモデルの方向を次のように選択します。

- ペインを上下方向にする場合は、一番上にあるプルダウンメニューから［XML Document］または［Action Model］のいずれかを選択します。上部ペインに対して［XML Document］を選択した場合、一番下にあるプルダウンメニューから［Action Model］を選択します。上部ペインに対して［Action Model］を選択した場合は、下部ペインに対して［XML Document］を選択します。
- ペインを左右方向にする場合は、左端にあるプルダウンメニューから［XML Document］または［Action Model］のいずれかを選択します。左ペインに対して［XML Document］を選択した場合、右端にあるプルダウンメニューから［Action Model］を選択します。左ペインに対して［Action Model］を選択した場合は、右ペインに対して［XML Document］を選択します。

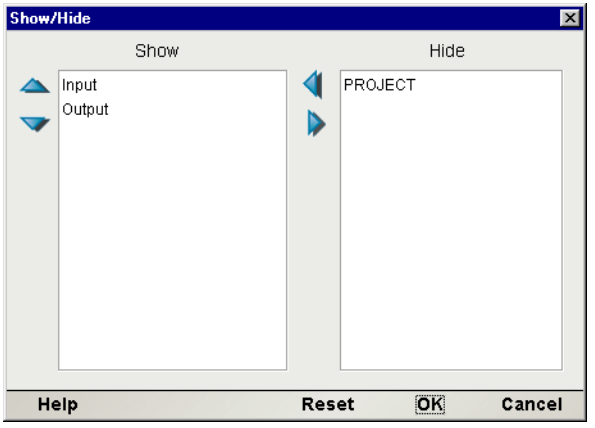
3 ［**OK**］をクリックします。

4 結果に満足できないときは、［**View**］、［**Window Layout**］の順に選択して、［**Reset**］ボタンをクリックします。ペインはデフォルト設定に戻ります。

➢ XML ドキュメントの表示レベルを設定する

1 ［**View/XML Documents**］>［**Show/Hide**］の順に選択します。［Show/Hide］ダイアログボックスに、XML ドキュメントの表示ステータスが表示されます。

注記： 入力 XML ドキュメントおよび出力 XML ドキュメントは、デフォルトで［**Show**］列に表示されます。Component アクションの結果として作成された DOM は、デフォルトで「非表示」になります。



2 ［**Hide**］列から、表示する XML ドキュメントを選択し、「**左向き三角形**」のボタンをクリックします。反対に、［**Show**］列から、表示しない XML ドキュメントを選択し、「**右向き三角形**」のボタンをクリックします。

3 「トップドキュメント」として表示する XML ドキュメントを選択し、［**Show**］列の希望のレベルにドキュメントが表示されるまで「**上向き三角形**」のボタンをクリックします。

注記： 前の手順 (「コンポーネントエディタのペインを整列する」を参照) で左右方向を選択した場合、XML ドキュメントの順序は左から右へと表示されます (優先順位の高いドキュメントは左側に表示されます)。

4 引き続き［**Show**］列で XML ドキュメントを選択し、XML ドキュメントが希望の順序になるまで、必要に応じて上向き三角形と下向き三角形のボタンを使用します。

5 ［**OK**］をクリックします。［Window Layout］ダイアログボックスが閉じ、コンポーネントエディタが順に並べ替えられます。

マッピングペインについて

デフォルトの XML Map コンポーネントエディタでは、次のウィンドウペイン設定になっています。

- 1 つまたは複数の入力マッピングペイン (各ペインには、それぞれの XML テンプレートの 1 サンプルに対する DOM が表示される)、1 つの出力マッピングペイン
- 1 つのアクションモデルペイン
- 一時 DOM マッピングペイン、または Component アクションを使用して別のコンポーネントを実行した結果として返された DOM

マッピングペインにはカラーコーディングの特徴があることに注意してください。赤は、要素の最初の直接マッピングを示します。このため、出力ツリーでマップ済みの要素を確認する場合は、赤という色によってその要素を識別できます。緑は、繰り返し別名が定義されている要素が初めて現れたことを示します。

注記： XML Interchange アクションによって動的に作成されたマッピングペインなど、動的に作成された DOM を表示するには、前に説明した［Window Layout］を使用する必要があります。

入力マッピングペインについて

入力マッピングペインには、入力 XML テンプレートから 1 つのサンプルドキュメントが DOM として表示されます (コンポーネントに複数のテンプレートが含まれている場合は、各入力 DOM がそれぞれのペインに表示されます)。ペインは、境界線を上下または左右にドラッグすることによって、サイズを調整できます。

コンポーネントに複数の入力 XML テンプレートが含まれている場合、これらのテンプレートは次の名前で表示されます。

表 6-3

表示順	DOM 名
最初のテンプレート	Input
2 番目のテンプレート	Input1
追加のテンプレート	Input*n*: (n = テンプレート順 - 1。たとえば、最後の入力テンプレートが5番目の入力テンプレートである場合、DOM 名は Input4 となります。)

テンプレートの表示順は、コンポーネントの作成時に XML テンプレートを選択したときに指定した順序によって決定されます。この順序は、テンプレートのプロパティを使用して変更できます。

➢ テンプレートの表示順を変更する

1 コンポーネントを開きます。

2 ［**File**］メニューから、［**Properties**］を選択します。

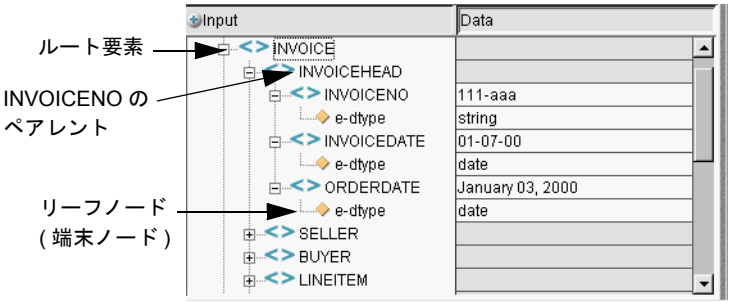
3 ［**XML Property Info**］タブを選択します。

4 入力識別子の名前を変更します。

5 ［**OK**］をクリックします。

DOM 要素およびデータ値について

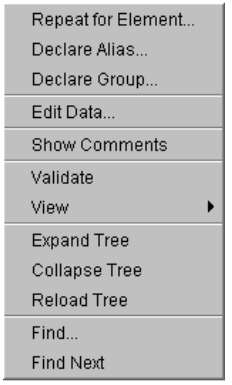
それぞれの［Input］ペインは、DOM ツリーとデータ値の 2 つの領域で構成されます。複数の要素を含む入力 DOM と、それらの要素のデータは、次の図のとおりです。



DOM 要素を選択すると、その要素名は、完全修飾要素名を表示するステータスバーに表示されます。(XML ドキュメントから) DOM 要素に関連付けられているデータは、すべて［Data］領域に表示されます。データは、ランタイム時のコンポーネントによる処理では使用されませんが、コンポーネントのセットアップやテストを行う場合には役立ちます。DOM 要素のデータは、そのままにしておいたり変更したりすることができます。データの変更の詳細については、120 ページ「DOM 要素の値の編集 (［Edit Data］コマンド)」を参照してください。

入力マッピングペインのコンテキストメニューについて

入力マッピングペインには、入力 DOM でタスクを実行するためにアクセスできるコンテキストメニューがあります。コンテキストメニューにアクセスするには、ペインの任意の場所でマウスを右クリックします。コンテキストメニューは、次の図のとおりです。



Repeat for Element、Declare Alias、または Declare Group アクションの作成

入力 DOM で繰り返す要素には、Repeat for Element、Declare Alias、または Declare Group アクションを作成できます (148 ページ「Declare Alias アクション」を参照)。

また、サンプルドキュメントで元の位置とは異なる構造に DOM 要素を変換できるようにする DOM 要素のグループを作成することもできます。グループを作成すると、グループに対して集約操作を実行することができます。たとえば、DOM 要素をアメリカの州 (アラバマ、アリゾナなど) 別にグループにまとめて、各州ごとに総売上高を計算することが可能です。

グループは、常に Repeat for Group アクションと組み合わさって機能します。グループの作成および使用の詳細については、203 ページ「Repeat for Group アクション」を参照してください。

DOM 要素の値の編集 (［Edit Data］コマンド)

入力 DOM の要素を選択し、値を設定することができます。これは、アニメーションツールを実行してコンポーネントをテストする場合に便利です。

注記 : 値は、編集セッションに対して一時的です。

➢ **DOM 要素の値を編集する**

1 入力 DOM 要素を選択します。

2 ［Input］ペインでマウスを右クリックします。

3 ［**Edit Data**］を選択します。

4 選択した DOM 要素がダイアログボックスに表示されます。要素に対して設定する値を入力します。

5 ［**OK**］をクリックします。値が、選択した要素の隣の［Data］領域内に表示されます。

［Show Comments］

このコンテキストメニューコマンドを使用すると、ソース XML ファイルでコメントの表示レベルを切り替えることができます。コンテンツの部分をラップした <!-- および --> というマーカで示されるコメントは、DOM ノードを構成しますが、常に表示する必要があるわけではありません。表示レベルは、コンテキストメニューの［Show Comments］コマンドを使用して制御されます。

DOM の検証

コンテキストメニューから［**Validate**］を選択すると、DTD またはスキーマ定義ファイルに対して DOM を検証できます。検証は、コンポーネントの構築およびテストの際に便利です。

DOM ツリーの展開

マウスを右クリックして［Expand Tree］を選択すると、DOM ツリー内のすべての要素を表示できます。

DOM ツリーを展開する別の方法として、ペインの上部 (または DOM ツリーフォルダアイコンの左側) の DOM 名 (「Input」など) の左側にある［+］アイコンをクリックすることもできます。

DOM ツリーを閉じる

マウスを右クリックして［**Collapse Tree**］を選択すると、DOM ツリー内のすべての要素を非表示にできます。

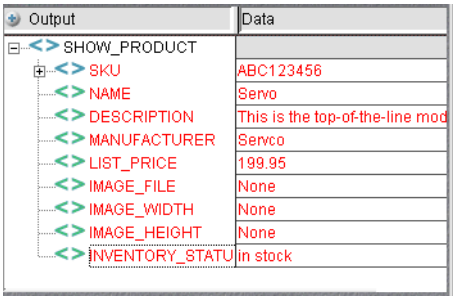
DOM ツリーを閉じる別の方法として、ペインの上部 (または DOM ツリーフォルダアイコンの左側) の DOM 名 (「Input」など) の左側にある［-］アイコンをクリックすることもできます。

DOM の表示

DOM は、ツリー、テキスト、または様式化された形式 (XSL スタイルシートを使用) で表示できます。

ツリービュー

デフォルトのビューでは、DOM がツリーとして表示されます (次を参照)。



このビューでは、要素と属性の「値」(つまり、ドキュメントデータ) を編集できますが、DOM 構造は編集できません。

テキストビュー

テキストビューでは、構造要素を含む完全な XML ファイルを表示したり編集したりできます。

(DOM ペインの任意の場所で) マウスを右クリックして、[**View**]、[**Text**] の順に選択すると、DOM をテキストとして表示できます。DOM は、プレーンテキスト形式で表示されます (次を参照)。

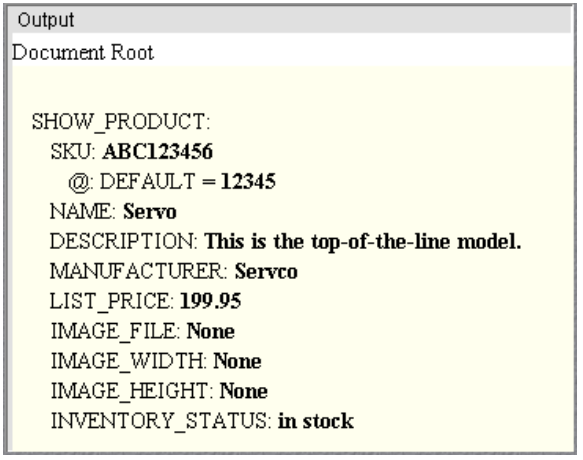
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SHOW_PRODUCT>
  <SKU DEFAULT="12345">ABC123456</SKU>
  <NAME>Servo</NAME>
  <DESCRIPTION>This is the top-of-the-line model
  <MANUFACTURER>Servco</MANUFACTURER>
  <LIST_PRICE>199.95</LIST_PRICE>
  <IMAGE_FILE>None</IMAGE_FILE>
  <IMAGE_WIDTH>None</IMAGE_WIDTH>
  <IMAGE_HEIGHT>None</IMAGE_HEIGHT>
  <INVENTORY_STATUS>in stock</INVENTORY_STATUS>
</SHOW_PRODUCT>
```

テキストビューでは、comment、processing instruction、DOCTYPE declaration などの、入力 DOM、一時 DOM、または出力 DOM の非コンテンツモデル部分を簡単に検査できます。

注記： テキストビューは、DOM ビューの場合と同様に、アニメーション中に動的に更新されるため、個々の Map アクションの実行時にその結果を表示することができます。

様式化されたビュー

(DOM ペイン内で使用できるコンテキストメニューで、マウスを右クリックして) 様式化されたビューを選択すると、DOM コンテンツが次のように表示されます。

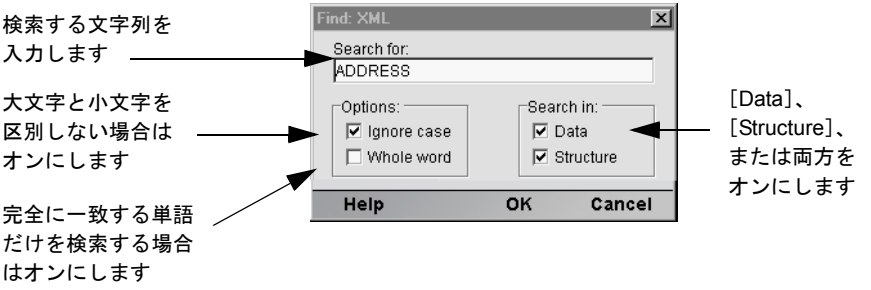


このビューでは、XML コンテンツの「レポート」スタイルの概要が表示されるため、すべての属性や要素のコンテンツを一目で確認できます。

注記 : このビューの作成において Composer が依存するデフォルトの XSL スタイルシートは、**\bin** ディレクトリの **xcd-all.jar** ファイル内に **default.xsl** という名前で存在します。このファイルは、jar ファイルから解凍 (展開) して、jar ファイルの同じ場所に編集したバージョンを再挿入することにより、編集または置換できます。

DOM 要素の検索

(DOM ウィンドウのコンテキストメニューに表示される) [**Find**] コマンドを使用すると、要素名および要素データを検索できます。[Find] ダイアログボックスでは、値を入力して、DOM ツリーを検索できます。検索する際は、単語の一部または完全に一致する単語だけを検索したり、大文字と小文字を区別せずに検索したりすることが可能です。[Find Text] ダイアログボックスは、次のとおりです。



次の DOM 要素の検索

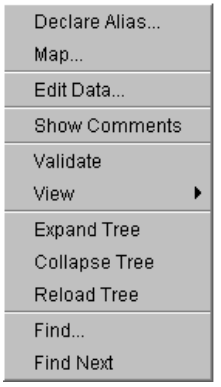
以前に検索した単語または文字列が次に現れた場合を検索することができます。［**Find Next**］を選択した場合、ダイアログボックスは表示されません。代わりに、Composer では、最後に検索したものが次に現れる場所を検索します。一致するものがない場合は何も起こりません。

DOM ツリーの再ロード

コンテキストメニューを表示して［**Reload Tree**］を選択すると、特定の DOM をリセットできます。これによって、XML Map コンポーネント内で個別の DOM ツリーを再ロードできます。テスト中にアニメーションを中止して DOM を未完成のままの状態にしておいた場合は、DOM ツリーの再ロードが必要になることがあります。

出力マッピングペインについて

出力マッピングペインには、出力 DOM が表示されます。出力マッピングペインにもコンテキストメニューがあります (次を参照)。



出力マッピングペインのコンテキストメニューにあるオプションは、入力マッピングペインのものに類似していますが、次に説明する部分が異なります。

出力要素への入力要素のマッピング

マウスを右クリックすると表示されるコンテキストメニューから［Map］を選択すると、Map アクションを使用できます。

値の設定

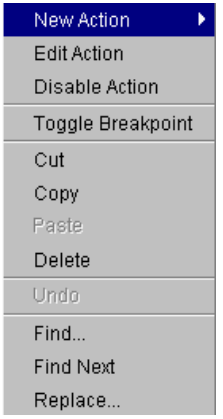
出力 DOM の［Edit Data］オプションを使用すると、ノードの値を検査することはできますが、変更はできません。

アクションモデルペインについて

すべてのコンポーネントには、単一のアクションモデルがあります。アクションモデルは、マッピング、変換、およびランタイム処理中に XML ドキュメントで実行されるその他のアクションを表します。また、アクションモデルペインは、XML Map コンポーネントエディタウィンドウ内でサイズを変更できます。アクションモデルペインで行われるアクティビティのほとんどは、アクションの追加および編集に関連しています。

アクションモデルのコンテキストメニューについて

アクションモデルで右クリックすると、次のようなメニューが表示されます。



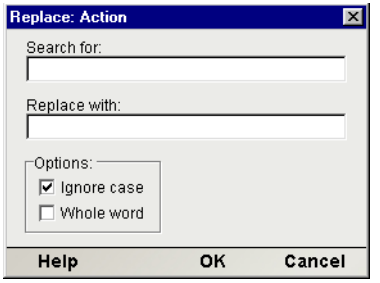
このメニューからは、アクションを選択したり他のタスクを実行したりできます。

アクションモデルでのテキストの置換

マウスを右クリックすると表示されるメニューまたはコンポーネントエディタの［Edit］メニューにある［Replace］オプションを使用すると、単語または文字列を置換できます。

➢ **テキストを置換する**

1 アクションモデルでマウスを右クリックし、［**Replace**］を選択します (または、アクションを選択し、［**Edit**］メニューから［**Replace**］を選択します)。



2 検索テキストを入力して、［**OK**］をクリックします。

3 Composer では、入力したテキストが最初に現れる場所を検索し、置換を確認するメッセージが表示されます。その後、検索テキストが次に現れた場合、または検索テキストが現れる箇所すべてで置換することができます。

コンポーネントへのアクションの追加

入力テンプレートおよび出力テンプレートを指定すると、XML Map エディタが開き、アクションの追加を開始する準備が整います。アクションは、コンポーネント内で行われる処理手順です。アクションの詳細については、後のトピックで取り扱います。

コンポーネント内では、DOM 要素をマップするアクションの追加、ファイルからのデータの読み書き、電子メールの送信、および他の共通タスクを実行します。アクションの集合は、アクションモデルと呼ばれます。

アクションモデル内のアクションは行として表示され、アクションタイプを示すアイコンとアクションの簡略定義が含まれます。一部のアクションは、他のアクションに対するサブオーディネートです。たとえば、ループ処理を制御する Repeat アクションを作成し、ループ内にアクションを追加できます。ループ内のアクションは Repeat アクションに対するサブオーディネートで、下層にインデントされて表示されます。これらのアクションでは、Repeat アクションが true の場合、処理を行います。

➢ アクションモデルにアクションを追加する

1 アクションモデルペインで、次のアクションを挿入する場所の上にカーソルを置きます。

2 次の方法のいずれかを使用して、アクションを追加します。選択した行の下に新しいアクションが挿入されます。

- 「**ドラッグアンドドロップ**」：入力DOMから出力DOMまたは一時DOMに要素をドラッグアンドドロップすると、Map アクションを追加できます。入力 DOM の要素を単にクリックし、出力 DOM または一時 DOM の上にドラッグしてください。

- 「**コピーして貼り付け**」: アクションモデルペインのアクションをコピーして、そのペイン内のどこかに貼り付けたり、別のコンポーネントのアクションモデルペインに貼り付けたりすることができます。
- 「**[Action] メニュー**」: アクションモデルペインで行を選択し、[**Action**] メニューからアクションを選択します。選択した行のすぐ下に新しいアクションが配置されます。
- 「**アクションモデルペインのコンテキストメニュー**」: アクションモデルペインの任意の場所でマウスを右クリックすると、コンテキストメニューが表示されます。
- 「入力マッピングペインおよび出力**マッピングペインのコンテキストメニュー**」: DOM ツリー要素を選択し、それぞれのコンテキストメニューからアクションを選択すると、アクションモデルペインにアクションを追加できます。

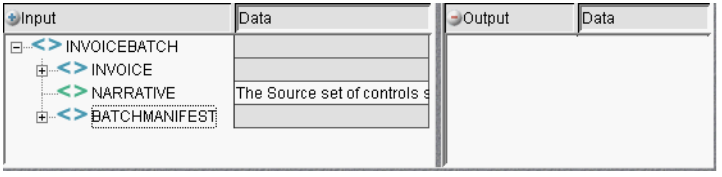
注記： アクションモデルペインのアクションは、新しい場所にドラッグして並べ替えることができます。

アクションモデルを作成した後、ライブデータで処理する前に、コンポーネントをテストする必要があります。テストは、Composer のアニメーションツールを使用して実行します。アニメーションツールを使用すると、ブレークポイントの設定、アニメーションの開始、アクションのステップインとステップオーバ、およびアニメーションの一時停止を実行できます。

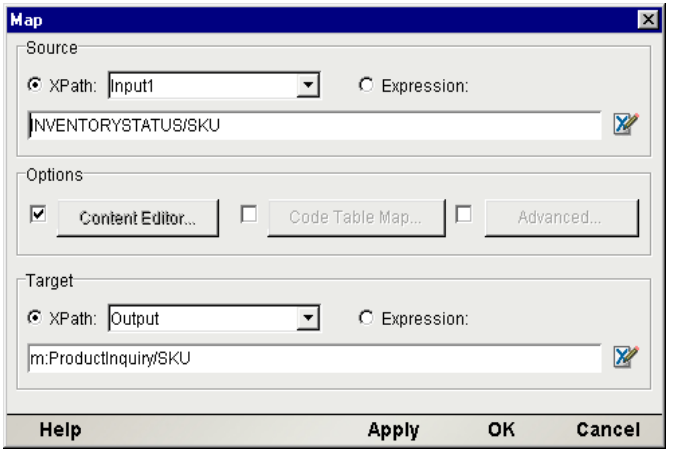
テンプレートを使用しない出力 DOM の作成

コンポーネントの作成時に、出力テンプレートとして `{SYSTEM}{ANY}` を選択すると、構造が含まれない出力 XML テンプレートを指定できます。次に、入力 DOM 要素を、まだ存在しない出力 DOM 要素にマップして、出力 DOM を動的に作成できます。

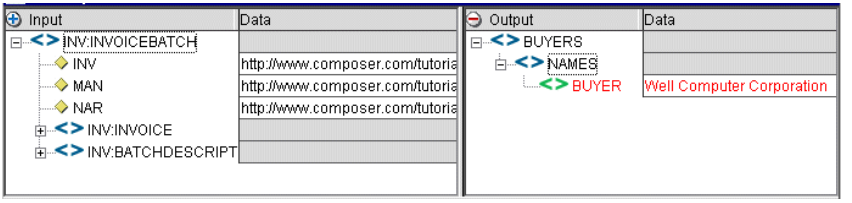
たとえば、要素が含まれる入力 DOM と要素が含まれない出力 DOM が 1 つずつ存在するコンポーネントは、次の図のとおりです。



出力 DOM には要素がありません。出力 DOM を動的に作成するには、入力 DOM 要素を、まだ存在しない出力 DOM の構造にマップします。次の図では、入力 DOM の請求書が出力 DOM の品目にマップされます。



結果の DOM は、次の図のとおりです。



入力 DOM 要素を出力 XPath にマップすると、出力 DOM 構造を作成できます。ただし、完全修飾 DOM 名にマップするようにしてください。

注記：実際には、出力 DOM と一時 DOM は、常に動的に作成されます。サンプルドキュメントは、単にアクションを定義するときに役立つものです。

作成した出力 DOM を他のコンポーネントの作成に使用できる場合は、XML ドキュメントとして保存し、XML テンプレート内でサンプルとして使用できます。詳細については、133 ページ「XML ドキュメントとしての DOM の保存」を参照してください。

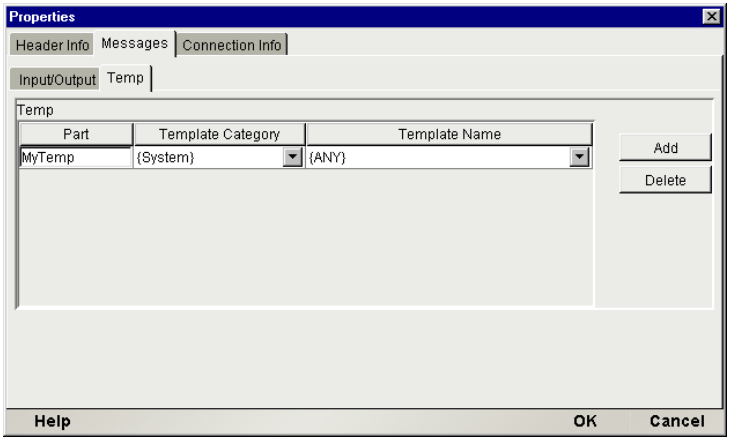
一時 DOM の作成

入力 DOM と出力 DOM のほかに、一時 DOM を作成することもできます。この DOM は、入力 DOM と出力 DOM の複雑な操作を実行するための作業領域として使用されます。一時 DOM ペイン内では、5 つのマッピング方法のいずれかを使用して、どの入力 DOM からでも要素を追加できます (125 ページ「アクションモデルペインについて」を参照)。

一時 DOM は、許可されているアクションの方向が入力 DOM および出力 DOM とは異なります。一時 DOM は、マッピングアクションのソースとターゲットの両方に指定することができますが、入力 DOM はソースとしてのみ、出力 DOM はターゲットとしてしか指定できません。一時 DOM は、複数追加することができ (追加できる数はメモリによってのみ制限されます)、いつでも削除できます。また、［**Temp**］ラベルを上書きして、一時 DOM に独自の名前を割り当てることもできます。

➢ 一時 DOM を作成する

1 ［**File**］メニューから、［**Properties**］を選択します。［Properties］ダイアログボックスが表示されます。



2 ［**Temp Documents**］タブをクリックします。

3 右端にある［**Add**］ボタンをクリックします。識別子、テンプレートカテゴリ、およびテンプレート名に対する選択肢が使用可能になります。

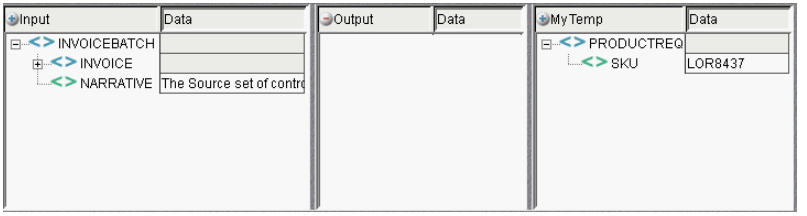
4 一時 DOM の「**識別子**」(独自のラベル) を入力します。

5 ［**Template Category**］ドロップダウンメニューから、XML カテゴリを選択します。

6 ［**Template Name**］ドロップダウンメニューから、XML テンプレートを選択します。

7 ［**OK**］をクリックします。

XML Map コンポーネントエディタに、一時 DOM ペインが表示されます (次の図を参照)。

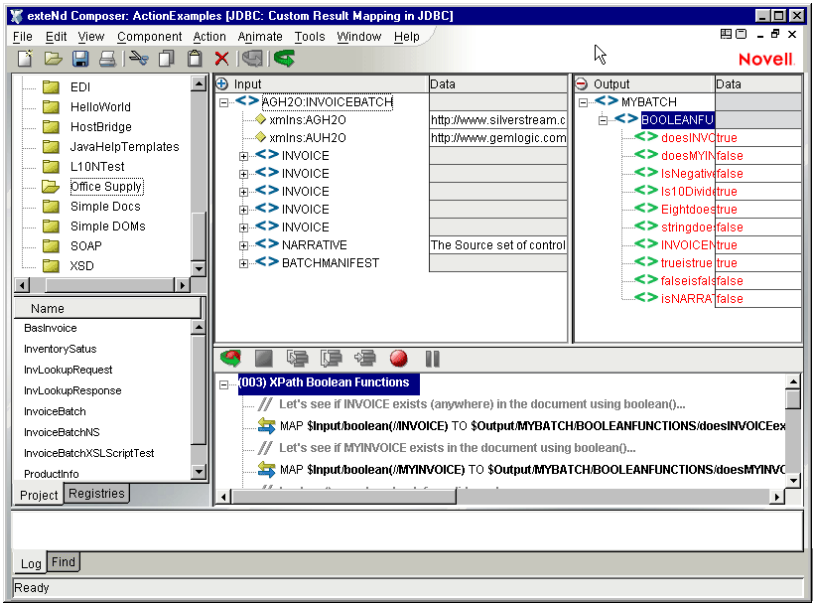


注記： 127 ページ「テンプレートを使用しない出力 DOM の作成」で説明されているとおり、DOM は、テンプレートを使用しないで動的に作成することもできます。

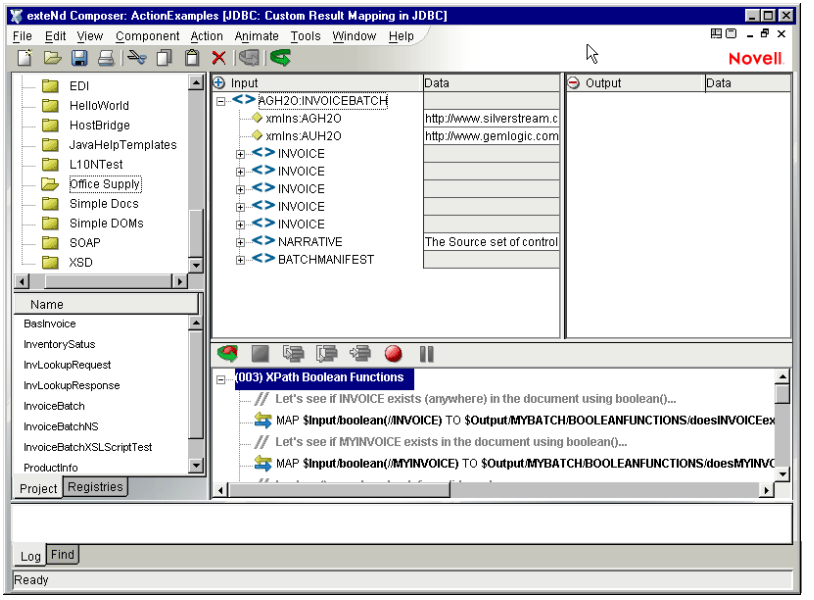
XML ドキュメントの再ロード

マッピングによって DOM 構造を変更した後で、DOM を元の状態に戻したい場合は、XML ドキュメントを再ロードできます。XML ドキュメントを再ロードすると、一時 DOM (作成されている場合) を含むすべての DOM が、入力 XML ドキュメントおよび出力 XML ドキュメントによって定義されている状態に戻されます。ただし、アクションモデルペインの Map アクションはそのまま保持されることに注意してください。つまり、コンポーネントを実行する予定であった場合、すべての Map アクションが実行されます。

入力 DOM、出力 DOM、および一時 DOM で反映される Map アクションがいくつか含まれているコンポーネントは、次の図のとおりです。



アクションモデルペインの DOM の詳細と Map アクションに注目してください。XML ドキュメントが再ロードされた後の同じ画面は、次の図のとおりです。



DOM は元の状態に戻されましたが、アクションモデルペインはそのままの状態で保持されています。XML ドキュメントを再ロードするには、［**Component**］メニューから［**Reload XML Documents**］を選択します。

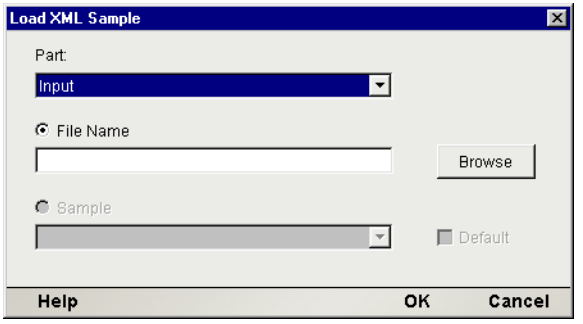
サンプルドキュメントのロード

異なるサンプルドキュメントを任意の DOM にロードし、要素のマッピングまたはアクションモデルのテストに対して新しい DOM 構造を使用できます。テンプレートから異なるサンプルドキュメントをロードすると、ランタイム時にコンポーネントによって受信される可能性のあるすべての XML ドキュメントのケースをアクションモデルで処理できるかどうかをテストできます。

XML サンプルをロードすると、DOM は変更されますが、アクションモデルはそのまま保持されます。サンプル XML ドキュメントでのテストが完了すると、［Load XML Sample］の手順を繰り返して、元の XML ドキュメントを再ロードできます。

➢ サンプルドキュメントをロードする

1 ［**File**］メニューから、［**Load XML Sample**］を選択します。［Load XML File］ダイアログボックスが表示されます。



2 ［**Part**］ドロップダウンボックスから、新しいサンプルドキュメントのロード先となる DOM を選択します。

3 元の XML テンプレートに含まれていないサンプルドキュメントをロードする場合は、［**File Name**］をクリックしてファイルの名前を入力します (または、［**Browse**］をクリックしてファイルを検索します)。

4 元の XML テンプレートに含まれている XML ファイルをロードする場合は、［**Sample**］をクリックして XML ドキュメントを選択します。

5 選択したサンプルを、このコンポーネント内でのみ選択した DOM に対してデフォルトの XML ドキュメントにする場合は、［**Default**］をクリックします (これは、ファイル名オプションには適用されません)。

6 ［**OK**］をクリックします。

コンポーネントの保存

ハードウェアまたはソフトウェアの障害によって作業が失なわれるのを防ぐため、コンポーネントは頻繁に保存してください。また、コンポーネントに異なる名前を付けて保存すると、バックアップコピーを作成することもできます。別の名前で保存すると、XML プロパティ (入力 XML テンプレートおよび出力 XML テンプレートを含む) も変更できます。

➢ 新しい名前でコンポーネントを保存する

1 ［**File**］メニューから、［**Save As**］を選択します。

2 ［**Name**］フィールドに、新しい名前を入力します。

3 入力XML ドキュメントおよび出力XML ドキュメントを変更するには、［**XML Property Info**］タブをクリックします。

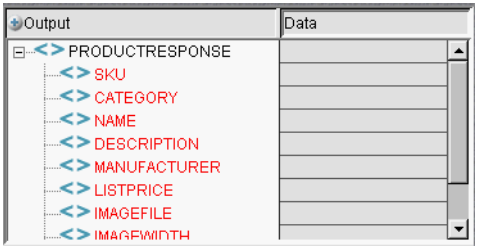
4 入力 XML ドキュメントを変更または追加します。

5 出力 XML ドキュメントを変更します。

6 ［**OK**］をクリックします。

XML ドキュメントとしての DOM の保存

DOM は、XML ドキュメントとして保存することもできます。これにより、DOM の構造とデータを含む XML ドキュメントが作成 (または上書き) されます。出力 DOM と結果の XML ドキュメントは、次の図のとおりです。



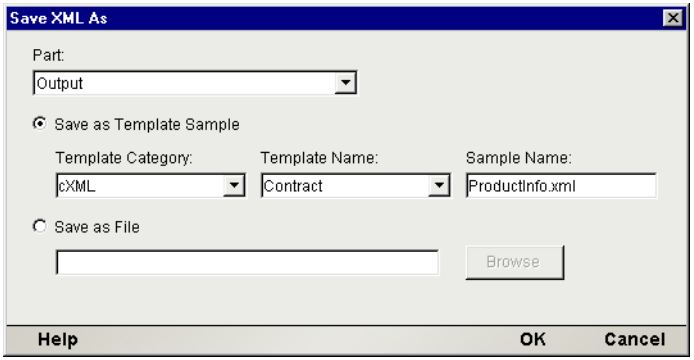
データを含む出力 DOM

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- edited with XML Spy v2.5 NT  - http://www.xmlspy.com -->
<!DOCTYPE PRODUCTRESPONSE SYSTEM "PRODUCTRESPONSE.dtd">
<PRODUCTRESPONSE>
  <SKU/>
  <CATEGORY/>
  <NAME/>
  <DESCRIPTION/>
  <MANUFACTURER/>
  <LISTPRICE/>
  <IMAGEFILE/>
  <IMAGEWIDTH/>
  <IMAGEHEIGHT/>
  <INVENTORYSTATUS/>
</PRODUCTRESPONSE>
```

結果の XML ドキュメント

➢ メモリ内 DOM を XML ファイルに保存する

1 ［**File**］メニューから、［**Save XML As**］を選択します。［Save XML As］ダイアログボックスが表示されます。



2 ［**Part**］の下にあるプルダウンメニューを使用して、ディスクに保存するソース DOM を選択します。前の例では、［Output］が選択されています。

3 ［**Save as File**］ラジオボタンをオンにします。

4 XML ドキュメントのパスと名前を入力するか、または［**Browse**］をクリックしてパスを選択します。

注記： 既存の XML ドキュメントを選択すると、ソース DOM の構造とデータで上書きされます。

5 ［**OK**］をクリックします。

テンプレートとしての XML ファイルの保存

コンポーネントエディタに表示される DOM は、(最初に DOM をファイルに保存してからテンプレートにインポートするのではなく) 直接 XML テンプレートとして保存できます。ターゲットテンプレートは、その場で作成できるため、あらかじめ存在している必要はありません。

➢ DOM を XML テンプレートに保存する

1 ［**File**］メニューから、［**Save XML As**］を選択します。［Save XML As］ダイアログボックスが表示されます (前の図を参照)。

2 ［**XML Document**］の下にあるプルダウンメニューから、テンプレートとして保存するソース DOM を選択します。

3 ［**Save as Template**］ラジオボタンをオンにします。

4 カテゴリをその場で作成する場合は、「***カテゴリ***」の名前を入力します (または、プルダウンメニューから既存のカテゴリを選択します)。

5 テンプレートをその場で作成する場合は、［**Template Name**］に名前を入力します (または、プルダウンメニューから既存の XML テンプレートの名前を選択します)。

6 この XML ドキュメントの「***サンプル名***」を入力します (これは、ディスク上でのファイル名になります。ファイルは、プロジェクトディレクトリの **\xmlcategories\[CategoryName]\imports** に保存されます)。

7 ［**OK**］をクリックします。Composer のナビゲーションフレームにあるインスタンスペインに、新しい XML テンプレートが表示されます。

この方法を使用して XML テンプレートをその場で作成している際に、新しいテンプレートに関連付ける追加の情報 (スキーマ名や XSL スタイルシートなど) を入力するよう求められることはありません。新しいテンプレートの検証、スタイルシート、またはその他のプロパティを検査したり編集したりする場合は、次に示す手順に従います。

XML テンプレートのプロパティの検査および編集

XML テンプレートを作成すると、そのプロパティを検査したり変更したりできます (101 ページ「XML テンプレートの編集」を参照)。

メモリ不足の問題の回避

大きな DOM を操作している場合は、ランタイム時でのメモリに関するエラーを回避するため、**xconfig.xml** ファイルに次の行を追加します (次の **xconfig** サンプルファイルの抜粋を参照)。

追加 (または編集する) 行 : `<VM_PARAMS>-Xms64m-Xmx128m</VM_PARAMS>`

コンポーネントのプロパティの表示

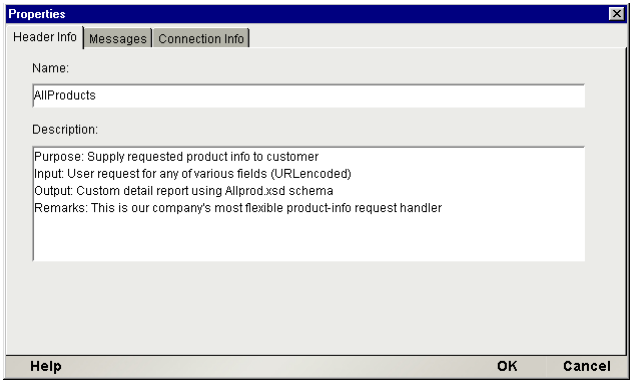
コンポーネントのさまざまなプロパティは、いつでも検査 (および、場合によっては編集) できます。

➢ コンポーネントのプロパティを表示または変更する

1. Composer の［**File**］メニューから、［**Properties**］を選択します。［Properties］ダイアログボックスが表示されます。このダイアログボックスには、次の 3 つのタブがあります。

   - ［**Header Info**］— これは、コンポーネントを最初に作成したときに入力した (または入力しなかった) 説明的なコメントです。

- ［**Messages**］— これは、New XML Map Component ウィザードで 2 番目に表示されるダイアログボックスに相当し、このコンポーネントで使用されるテンプレートとテンプレートのカテゴリが表示されます。
- ［**Connections Info**］— このタブは、特定の接続リソースに関連付けられているコンポーネントでのみ表示されます (たとえば、JDBC コンポーネントでは、［プロパティ］ダイアログボックスにこのようなタブがあります)。単純な XML Map コンポーネントの場合、この情報が含まれないため、このタブは表示されません。



2 このコンポーネントの説明的なコメントを表示または編集するには、［**Header Info**］タブをクリックし、希望の情報を入力します。

3 XML テンプレートの選択肢を表示または変更するには、［**Messages**］タブをクリックします。テンプレートドキュメントまたはテンプレートカテゴリ、あるいはその両方は、必要に応じて追加または削除できます。

4 コンポーネントで特別な接続リソースが使用されている場合は、［**Connections Info**］タブをクリックして、このコンポーネントの接続リソース情報を表示します (通常の XML Map コンポーネントには適用されません)。

5 ［**OK**］をクリックして、ダイアログボックスを閉じます。

6 コンポーネントを「**保存**」します。

コンポーネントの印刷

コンポーネントのコンテンツは印刷することができます。印刷物には次の項目が含まれます。

- コンポーネントを印刷した日時
- コンポーネントの名前と説明
- 入力DOM、出力DOM、および一時DOMを構成しているすべてのXMLドキュメント
- アクションモデルのすべてのアクション

➢ コンポーネントを印刷する

1. ［**File**］メニューから、［**Print**］を選択します。
2. プリンタを選択します。
3. ［**OK**］をクリックします。

コンポーネントの設計、テスト、および実行

コンポーネントを設計、テスト、および実行する際にサンプルドキュメントが使用される方法については、次の表のとおりです。

表 6-4

DOM	Composer での設計時	Composer でのアニメーションツールの使用時	サーバでの実行時
入力	サンプルは、アクションとテストデータを作成するために設計時に役立つものとしてロードおよび使用できます。	デフォルトのサンプルドキュメントは、ランタイム入力DOMをシミュレートするためにロードおよび使用されます。	DOMデータは、別のコンポーネント、サービス、またはサービストリガによって渡されます。
一時	サンプルは、アクションとテストデータを作成するために設計時に役立つものとしてロードおよび使用できます。	サンプルドキュメントはロードされません。DOMは、アクションモデルによって作成されます。	DOMは、アクションモデルによって作成されます。
出力	サンプルは、アクションを作成するために設計時に役立つものとしてロードおよび使用できます。	サンプルドキュメントはロードされません。DOMは、アクションモデルによって作成されます。	DOMは、アクションモデルによって作成されます。

7 基本的なアクション

これまでの章では、XML テンプレートおよび入力と出力にテンプレートを使用する XML Map コンポーネントの作成方法について説明しました。この章では、実際に行う作業について説明します。ここからはアクションが登場します。

注記：この章では、XML コンポーネントで使用できる基本的なアクションについて説明します。次の章では、より強力なアクションを、また、第 11 章「共通タスクへのアクションの適用」では、これらのアクションをいくつか使用した詳細な例について取り扱います。

アクションとは

「アクション」は、プログラミングステートメントに類似しており、パラメータの形式で入力を受け付け、特定のタスクを実行します。たとえば、Send Mail アクションでは、受信者の電子メールアドレスをパラメータの 1 つとして入力することにより、電子メールが送信されます。

個々のアクションについて学ぶ前に、まず exteNd のアクションモデルを理解する必要があります。これまでの章で、コンポーネントは、XML ドキュメントを処理したり、XML 以外のデータソースと通信したりするための手順のセットであるということを説明しました。この手順のセットは「アクションモデル」と呼ばれます。Composer では、アクションモデルは、データマッピング、データ変換、およびコンポーネントとサービス内でのデータ転送をすべて実行します。

アクションモデルは、アクションのリストから構成されています。アクションモデル内のすべてのアクションは相互に機能します。たとえば、あるコンポーネントのアクションモデルでは、請求書のデータをディスクから読み取り、電子メールアドレスを請求書から取得し、請求書が受信されたことを通知する電子メールメッセージを受信者に送信します。

このアクションモデルの例は、いくつかのアクションから構成されています。そのアクションは次のとおりです。

- 請求書のドキュメントを開き、請求書のデータをメモリに読み込む
- 請求書から電子メールアドレスを抽出する

- 電子メールを作成し、送信する
- メールが送信されたことを表示して請求書の記録を更新し、ファイルを閉じる

Composer アクションの使用

Composer では、基本の XML Map コンポーネントでアクションを実行します。これらのアクションは、JDBC コンポーネントや JMS コンポーネントなどの「他のすべてのコンポーネントタイプ」で使用できます。アクションは、［Action］メニューで、基本的なアクションと高度なアクションに分けられています。この 2 種類のアクションについては、次の表のとおりです。

表 7-1

基本的なアクション	説明
Comment	アクションモデルを記録します。特に、アクションモデルに Decisions または Repeats、あるいはその両方が使用されている場合、コメントを使用して処理を明確にすることができます。
Component	別のコンポーネントまたはサービスを実行し、渡されるランタイム DOM を指定し、呼び出されたコンポーネントから受信します。
Decision	指定した条件に基づいて、アクションの 2 つのセットから 1 つを実行できます。コンポーネントの実行で指定した条件がどのように解決されるかによって、True または False へのパスの分岐を処理します。
Declare Alias	Xpath への任意のラベルを割り当て、操作を便利にすることができます。割り当てたラベルは、ランタイム時またはアニメーション時に完全な Xpath に拡張されます。
Function	ECMAScript スクリプト関数または以前に作成したカスタムスクリプトのいずれかを実行します。カスタムスクリプトは、Composer のカスタムスクリプトリソースエディタを使用して作成できます。
Log	コンポーネントに指定されているさまざまなログファイルに情報を書き込みます。ログのタイプには、システム出力、システムログ、およびユーザログの 3 種類があります。
Map	要素のデータをある XML DOM から別の XML DOM へ転送したり、オプションで変換したりします。
Send Mail	コンポーネントの実行中、指定した電子メールアドレスに自動的に電子メールを送信します。

基本的なアクション	説明
Switch	入力値とケースの値との一致に基づいて、プログラムの制御をアクションの特定のブロックに分岐させることができます。これは、長く、読み取りが困難な if またはその他 (Decision アクション) のチェーンを排除するときに使用できる、基本的に便利なアクションです。

アクションの作成

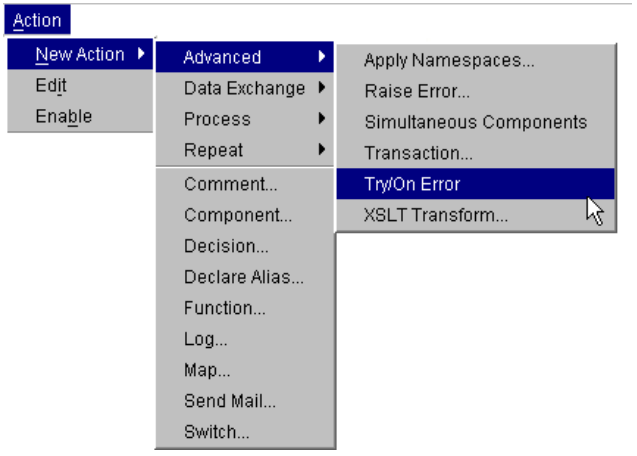
使用するアクションを作成するは、メインメニューバーの［**Action**］メニューを使用する方法と、Composer エディタで、コンテキストメニューを使用する方法の 2 種類があります。どちらの場合も、アクションを選択する前にコンポーネントを開く必要があります。

➢ ［Action］メニューを使用したアクションの作成

1 コンポーネントを開きます。

2 アクションモデルで、新しいアクションを配置したい場所のすぐ上の行でマウスをクリック (つまり、ハイライト表示または「選択」) します。選択した行の下に新しいアクションが挿入されます。

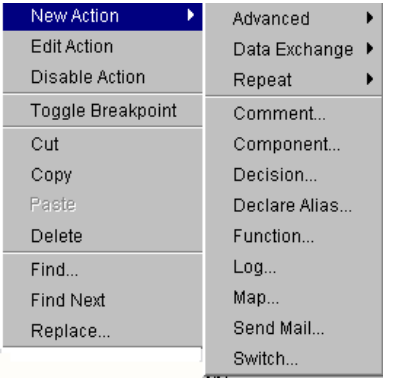
3 Composer の［**Action**］メニューから、［**New Action**］を選択し、次に作成するアクションのタイプを選択します。



4 ダイアログボックスが表示されたら、必要に応じてアクションに関するパラメータを入力または選択します (これらについては、後続のトピックで説明します。以降の説明を参照してください)。次に、該当する場合、ダイアログボックスを閉じます。

➢ コンテキストメニューを使用したアクションの作成

1. アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
2. マウスの右ボタンをクリックし、**コンテキスト**メニューを表示します。



3. コンテキストメニューから、アクションを選択します。
4. 表示されるダイアログボックスの操作を行います。
5. ダイアログボックスを閉じます。

アクションの追加に加え、アクションモデルの既存のアクションを編集したり、無効にしたりすることもできます。無効にすると、そのアクションは実行できなくなりますが、アクションモデルにはそのまま保持されるため、あとで有効にすることもできます。

➢ アクションを編集する

1. アクションモデルで任意のアクションをダブルクリックし、編集します。
2. または、アクションモデルペインでアクションを選択します。
3. ［**Action**］メニューから、［**Edit**］を選択します。アクションタイプのダイアログボックスが表示されます。
4. アクションに必要な変更を加えます。
5. ［**OK**］をクリックします。

➢ アクションを無効にする

1. アクションモデルペインで、アクションを選択します。
2. ［**Action**］メニューから、［**Disable**］を選択します。アクションがグレー表示になります。
3. アクションを再び有効にするには、［**Enable**］を選択して手順 1 と 2 を繰り返します。

➢ アクションを切り取る、コピーする、または貼り付ける

1. アクションモデルペインで、アクションを選択 (クリック) します。
2. メインメニューバーの［Edit］メニューまたはマウスの右ボタンで使用できるコンテキストメニューから、［**Cut**］、［**Copy**］、［**Paste**］、または［**Delete**］を必要に応じて選択します。
3. 操作を取り消すには、<Ctrl>+<Z> キーを入力 (または［Edit］メニューから［Undo］を選択) します。

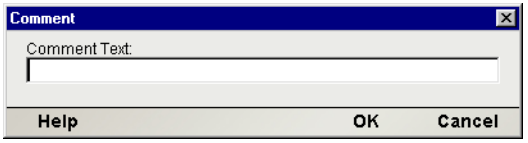
この章の残りの部分では、基本的なアクションとその使用方法の例を示します。

Comment アクション

Comment アクションを使用すると、アクションモデルを記録して実行する処理を明確にすることができます。コメントはアクションモデル内のどこにでも追加できます。コメントそれ自体は処理を行いません。

➢ Comment アクションを追加する

1. コンポーネントを開きます。
2. コメントを配置するアクションモデルの行を選択します。選択した行の下に新しいコメントが挿入されます。
3. ［**Action**］メニューから、［**New Action**］、［**Comment**］の順に選択します。［Comment］ダイアログボックスが表示されます。



4. コメントを入力します。
5. ［**OK**］をクリックします。

Component アクション

Component アクションでは、指定したランタイムの入力と出力で別のコンポーネントまたはサービスを呼び出し、実行します。プロジェクト内のどのコンポーネントでも呼び出すことができます。別のコンポーネントを呼び出すには、アクションに対して次の 4 種類のパラメータを指定する必要があります。

- Component Type
- Component Name
- Passed IDs

- Returned ID

［Component Type］には、呼び出すコンフリクトのカテゴリを指定します。コンポーネントタイプは、コンポーネントの見出しの付いた、Composer のカテゴリペインにリストされるタイプとは一致しません。次に挙げる文字列は有効な値であり、大文字小文字が区別されます。

- service
- Map
- jdbc
- 3270
- 5250
- cicsrpc
- html
- jms
- vt100

これらは、そのコンポーネントタイプを実装する接続がインストールされているかどうかによって異なります。

［Component Name］には、呼び出すコンポーネントまたはターゲットコンポーネントの名前を指定します。［Component Type］で選択したタイプに存在するコンポーネント名でなければなりません。

［Passed ID］には、現在のコンポーネントまたはサービス内のドキュメント名を指定します。ターゲットコンポーネントに渡すドキュメントを、1 つまたは複数指定できます。何も指定しないことも可能です。ここで指定するドキュメント名は、入力ドキュメントとしてターゲットコンポーネントに渡されます。

［Returned ID］には、ターゲットコンポーネントの結果を受け取る、現在のコンポーネントまたはサービス内のドキュメント名を指定します。既存のドキュメント名を使用するか、存在しない名前を指定して新しいドキュメントを作成することもできます。

これらのパラメータは、［*Predefined*］または［*Dynamic*］のいずれかの方法で指定できます。定義済みの Component アクションでは、プロジェクトの現在の状態における値を使用した 4 種類のパラメータを指定します。一度指定されると、これらの値は手動で変更されない限り、アクションのすべての実行に対して適用されます。動的な Component アクションでは、ランタイム時の 4 種類のパラメータは、作成した式によって計算された値で指定します。この場合、Component アクションの動作は柔軟になり、実行ごとにランタイムでの条件に対応できるようになります。1 度の Component アクションで、さまざまなランタイムの条件に基づいて異なるコンポーネントを実行する、異なる入力ドキュメントを渡す、または異なるドキュメントで結果を受け取ることができます。

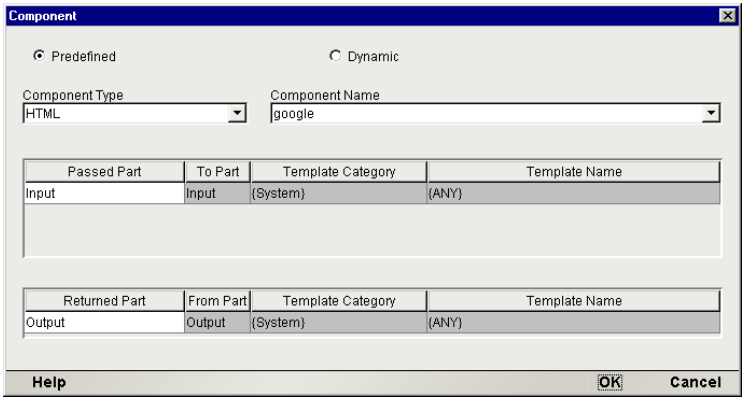
➢ 定義済みの Component アクションを追加する

1 コンポーネントを開きます。

2 コンポーネントを呼び出すアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。

3 ［**Action**］メニューから、［**New Action**］、［**Component**］の順に選択します。［Component］ダイアログボックスが表示されます。

4 ［Predefined］ラジオボタンが選択されていない場合、クリックしてオンにします。



5 ［Component Type］ドロップダウンリストから該当するコンポーネントタイプを選択します。

6 ［Component Name］から、実行するコンポーネント名を選択します (コンポーネントのリストは、［Component Type］で選択したコンポーネントタイプに対応しています)。

7 ［**Passed ID**］フィールドで、ソースコンポーネント DOM を選択します。

8 ［**Passed ID**］フィールドで、呼び出されるコンポーネントが出力を返すソース DOM を選択します。新しい DOM を作成する場合、［**Returned ID**］フィールドに名前を入力します。

9 ［**OK**］をクリックします。

➢ 動的な Component アクションを追加する

1 コンポーネントを開きます。

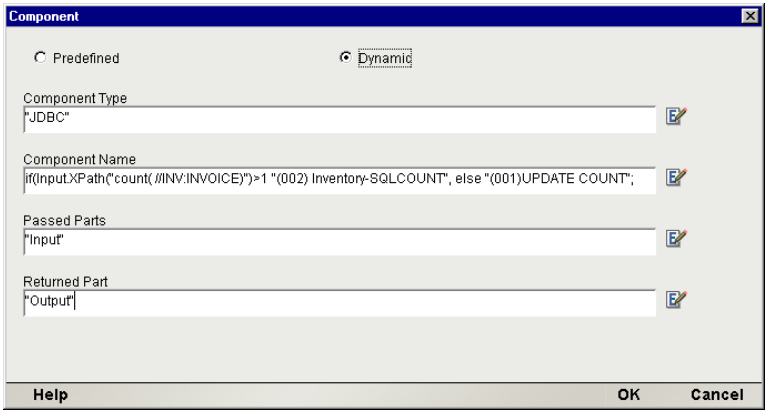
2 コンポーネントを呼び出すアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。

3 ［**Action**］メニューから、［**New Action**］、［**Component**］の順に選択します。［Component］ダイアログボックスが表示されます。

4 ［Dynamic］ラジオボタンが選択されていない場合、クリックしてオンにします。

5 有効なコンポーネントタイプ (map、service、JDBC、3270、5250、CICSRPC、JMS、HTML) のうちの 1 つを返す ECMAScript 式を作成します。

注記：コンポーネントタイプは、そのタイプを実装している接続が、使用している Composer にインストールされている場合にのみ有効です。



6 プロジェクト内の有効なコンポーネント名またはサービス名を返す ECMAScript 式を作成します。

7 現在のコンポーネントまたはサービス内のランタイム時における有効なドキュメント ID を返す ECMAScript 式を作成します。このドキュメントは、入力ドキュメントとしてターゲットコンポーネントまたはサービスに渡されます。複数のドキュメントを渡す場合、式は、ドキュメント ID のコンマで区切られたリスト (たとえば、Input、Input 1、Temp、MyDoc) を含む単一文字列を返す必要があります。

8 ターゲットコンポーネントの結果を受け取るドキュメント ID を返す ECMAScript 式を入力します。

Decision アクション

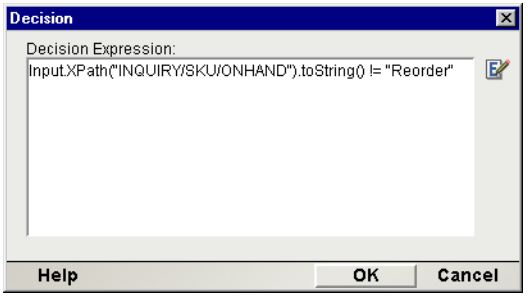
Decision アクションでは、アクションまたはアクションのグループに分岐する *if... then* ステートメントが作成されます。Decision アクションは、指定した条件に基づいて、ある分岐か別の分岐を選択するために使用します。条件には、==、<、>、!、>=、<=、(&)、OR (||)、または <> のような、ECMAScript 比較演算子を使用する必要があります。式はブール true ステートメントまたはブール false ステートメントを解決する必要があります。たとえば、請求書がある日付よりも古いかどうかを確認し、そうである場合に電子メールを送信できます。

➢ Decision アクションを追加する

1 コンポーネントを開きます。

2 アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。

3 ［**Action**］メニューから、［**New Action**］、［**Decision**］の順に選択します。［Decision］ダイアログボックスが表示されます。



4 任意の ECMAScript 比較演算子を使用して式を入力するか、［**Expression Builder**］ボタンをクリックして、ランタイム時に *true* または *false* を返す Decision スクリプト (ECMAScript 式) を作成します。

5 ［**OK**］をクリックします。アクションモデルには、次の Decision アクションが表示されます。これにより、INVOICE ノードの存在が確認されます。

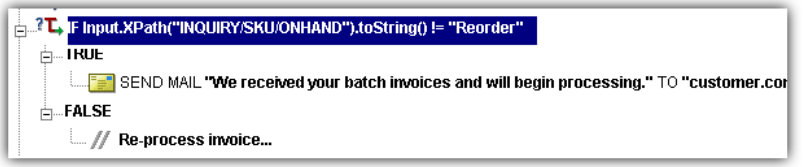
6 アクションモデルペインで、［**TRUE**］アイコンを選択します。

7 式が true である場合に実行する 1 つまたは複数のアクションを追加します。true 分岐の外側から内側へ cut/copy アクションをドラッグアンドドロップできます。

8 ［**FALSE**］アイコンを選択します。

9 式が false である場合に実行する 1 つまたは複数のアクションを追加します。

Decision アクションの **TRUE** 分岐または **FALSE** 分岐、あるいはその両方の内側に他の Decision アクションをネストできます。アクションモデルペインの完全な決定は、次の図のとおりです。

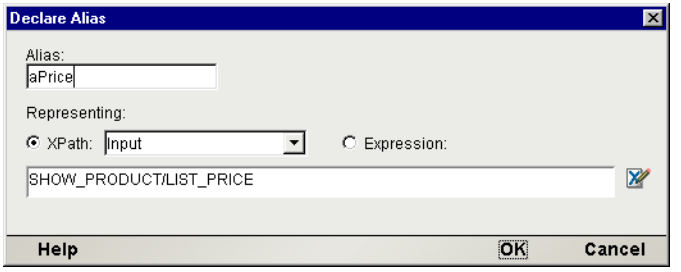


Declare Alias アクション

Declare Alias アクションでは、特定の Xpath 式に独自のカスタムラベルを適用できます (特定のアクションモデルのスコープ内でのみ有効です)。このアクションを使用すると、アクションモデルを認識しやすくでき、また入力の手間を省くことができます。

➢ Declare Alias アクションを追加する

1. コンポーネントを開きます。
2. Declare Alias アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action**］、［**Declare Alias**］の順に選択します。［Declare Alias］ダイアログボックスが表示されます。



4. ［**Alias**］テキストフィールドに、使用する名前を入力します。
5. ［**XPath**］ラジオボタンまたは［**Expression**］ラジオボタンのいずれかを選択します。
6. ［XPath］ラジオボタンを選択した場合、ドロップダウンメニューから、ターゲット DOM (ターゲット Xpath を含むドキュメントを示す) を選択します。次に、下のテキストフィールドに、ターゲットノードへの XPath を入力します。
7. ［Expression］ラジオボタンを選択した場合、テキストフィールドに、ターゲット XPath の ECMAScript での適切な表現を入力するか、またはテキストフィールドの右にある［Expression Builder］アイコンをクリックして、Expression Builder ピックリストを使用して式を作成します。

8 ［**OK**］をクリックします。アクションモデルに新しいアクションが追加されます。

図の例では、入力 DOM に SHOW_PRODUCT/LIST_PRICE というノードがあります。XPath 式への別名 (任意の名前) を割り当てると、アクションモデルを全体を通して「$Input/SHOW_PRODUCT/LIST_PRICE」と繰り返し入力する必要がなくなるため便利です。この例では、$Input/SHOW_PRODUCT/LIST_PRICE に、「aPrice」という別名が割り当てられています。別名を割り当てると、アクションモデル全体を通して「$Input/SHOW_PRODUCT/LIST_PRICE」ではなく「aPrice」を使用できます。ランタイム時には、別名は完全な XPath に拡張されます。

Function アクション

Function アクションでは、ECMAScript 関数またはカスタムスクリプトリソースエディタで作成したカスタムスクリプト関数のいずれかを実行できます。DOM の要素を操作するには、Function アクションで呼び出すスクリプトが現在のコンポーネント内の完全に識別できる DOM 要素の名前を参照している必要があります。

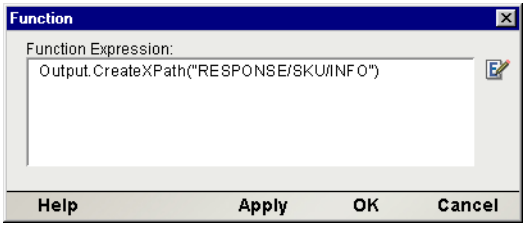
作成してアクションモデルに追加するカスタムスクリプト関数は、どの DOM ツリー要素に対しても使用できます。たとえば、日付要素の形式を変更する関数を作成したり、要素のコンテンツで math 関数を実行する関数を作成したりすることができます。DOM とのやり取りのない system 関数、database 関数、または URL 関数を実行することもできます。Function アクションは、カスタムスクリプトリソースで登録した Java メソッドを呼び出すために使用することもできます。これにより、複雑 (および単純) な Java の処理をアクションモデルに視覚的に直接統合することができます。

➢ Function アクションを追加する

1 コンポーネントを開きます。

2 Function アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。

3 ［**Action**］メニューから、［**New Action**］、［**Function**］の順に選択します。［Function］ダイアログボックスが表示されます。



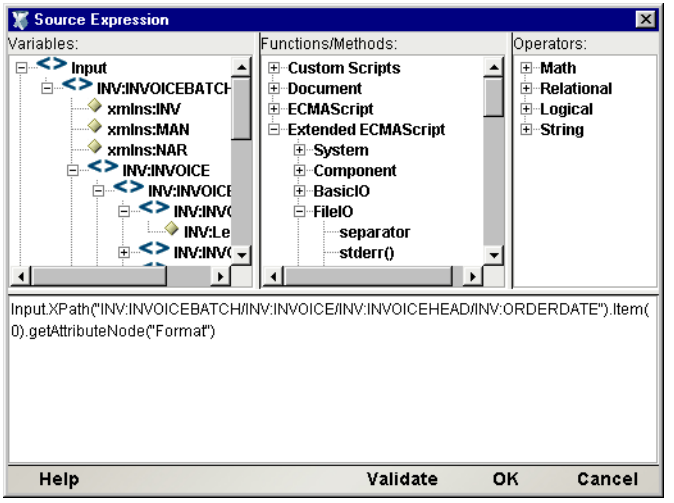
4 ［**Function Call**］フィールドに関数を入力するか、［**Expression Builder**］ボタンをクリックして ECMAScript 式を作成します (後の説明を参照)。関数の呼び出しでは、大文字小文字が区別されます。また、関数にパラメータが必要な場合、関数の呼び出しにそれらを含めることを忘れないでください。

5 ［**OK**］をクリックします。［Apply］ボタンをクリックすると、ダイアログボックスを閉じずに Function アクションの結果を表示できます。これにより、Function アクションを繰り返し編集しても、結果を迅速に確認できます。

➢ Expression Builder を使用する

1 前の節での手順に従って、新しい function アクションを追加します。

2 ［**Expression Builder**］ボタンをクリックし、［Function Expression Builder］ダイアログボックスを開きます。



3 変数、関数 / メソッド、または演算子をダブルクリックし、それらを関数に挿入します。関数に直接入力することもできます。

注記： 関数が ECMAScript の標準に準拠しているかどうかを確認します。準拠していない場合、関数は正しくコンパイルまたは動作しません。通常、関数をカスタムスクリプトリソース内で作成し、使用する前にテストすると、効率がよくなります。Function アクションの作成時には、カスタムスクリプトの関数名を参照し、それをパラメータとして指定できます。

4 ［**Validate**］をクリックして、保存する前にスクリプトを検証します。

5 ［**OK**］をクリックして、スクリプトを保存します。

6 ［**OK**］をもう一度クリックして、Function アクションを保存します。

注記： ECMAScript は解釈済みの言語であるため、[**Validate**] では、有効な ECMAScript 構文に準拠しているかどうかは確認されますが、ランタイムの従属式についてはチェックされません。

Log アクション

Log アクションは、カスタマイズ可能なレポート機能 (設計時、ランタイムとも) を Composer アプリケーションに配備できます。ログレベル設定 (後の節で説明) を使用すると、レポートのレベルを詳細に制御できます。Log アクションを単に「オン」または「オフ」にする必要はありません。

Log アクションを使用する場合の例は、次のとおりです。

- Try On Error 条件が満たされたときに、あるエラー情報をオペレータのコンソールに書き出す
- デバッグを助ける (Log メッセージは ECMAScript 式として作成できるため、ランタイム時にのみ使用される値を持つ変数または DOM のコンテンツに関する情報を記録できます)
- Repeat for Element ループの各サイクルから特定の情報をキャプチャする
- 開発時に自己記録型コンポーネントの作成を助ける

ログファイルの場所

Log アクションでは、Composer および exteNd Composer Enterprise Server の外部のさまざまな場所に情報を記録できます。実際の場所はアクションによって指定されます。ログ出力の場所には、システム出力、システムログ、およびユーザログの 3 種類があります (次を参照)。

システム出力

[System Output] オプションでは、[Log Expression] フィールドに指定するメッセージが設計時の Java 仮想マシンの処理ウィンドウまたはランタイム時のアクションサーバコンソールに書き出されます。

ログメッセージを作成するには、有効な ECMAScript 式を作成したり、Expression Builder を使用してログの式を生成したりすることができます。書き出されるメッセージの前には、日付と時間のスタンプおよびログを実行しているコンポーネントの情報が記録されます。これらのメッセージは、Composer メインウィンドウのメッセージフレームにも表示されます。

システムログ

［System Log］オプションでは、［Log Expression］フィールドに指定するメッセージが Composer の設定ファイル **xconfig.xml** の <LOGFILE> 要素に指定されたファイル名に書き出されます。ログファイルの名前と場所は、Composer メニューバーから、［**Tools**］>［**Configuration**］を選択して変更できます。

ユーザログ

［User Log］オプションでは、［Log Expression］フィールドに指定するメッセージが、［Log Action］ダイアログボックスの［User Log File］フィールドに指定するファイルに書き出されます (後を参照)。

ログメッセージを作成するには、静的な文字列を入力したり、有効な ECMAScript 式を作成したり (または Expression Builder を使用してログの式を生成したり) することができます。ログの式の結果は、テキストとしてログに書き出されます。書き出されるメッセージの前には、日付と時間のスタンプおよびログを実行しているコンポーネントの情報が記録されます。

ユーザログファイルを作成するには、有効な ECMAScript 式を作成してファイル名を生成したり、［**Expression Builder**］ボタンをクリックして、Expression Builder を使用したりすることもできます。

ログ優先度レベル

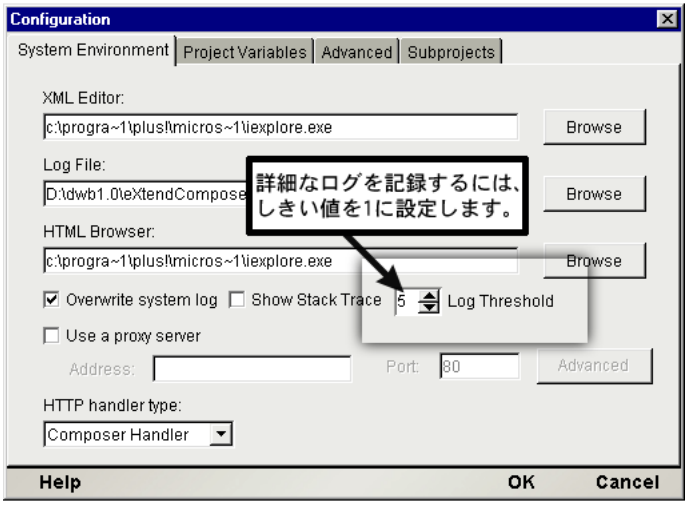
個々の Log アクションには、優先度レベル (1 から 10 まで) を割り当てることができます。ランタイム時の Log アクションの優先度は、［Tools］メニューの［Configuration］ダイアログボックスで設定した、レポートのしきい値と比較されます。優先度がレポートのしきい値と等しいまたはレポートのしきい値より高い Log アクションが実行され (つまり、メッセージは必要に応じてシステム出力またはディスクに記録されます)、優先度の低い Log アクションのメッセージは記録されません。

個々の Log アクションの優先度レベルは、［Log Action］ダイアログボックスで設定できます。記録のしきい値は、［Configuration］ダイアログボックスの［System Environment］タブで設定します (次の説明を参照)。しきい値を設定すると、それと等しいかそれより高い優先度の Log アクションが実行されます。たとえば、Log Action A の優先度が 4 に、Log Action B の優先度が 9 に、それぞれ設定されており、［Configuration］ダイアログボックスのしきい値が 8 に設定されている場合、ランタイム時には Log Action B が実行されます。Log Action A は無視されます。

注記：レポートのレベルは、プロジェクトの配備後に、exteNd Composer Enterprise Server のコンソール画面で調整することもできます。詳細については、Composer Enterprise Server のマニュアルを参照してください。

➢ ログのレポートしきい値を設定する

1 ［**Tools**］メニューにアクセスし、［**Configuration**］を選択します。［Configuration］ダイアログボックスが表示されます。



2 ［System Environment］タブで、「**Composer のログレベル**」を 1 から 10 までの値に設定します。ここで設定する値が、「しきい値」です。優先度がこの値と等しいまたはこの値よりも高い Log アクションのみが実行されます。

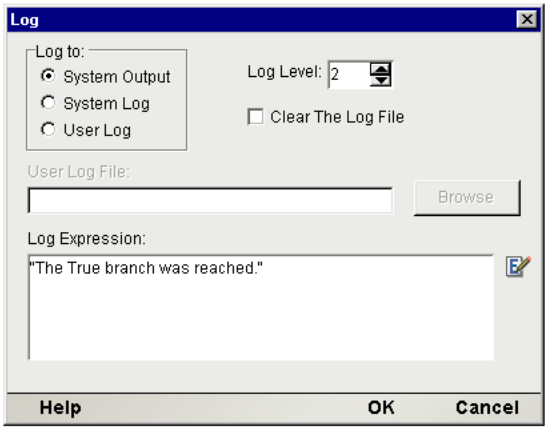
3 ［**OK**］をクリックして、ダイアログボックスを閉じます。

➢ Log アクションを作成する

1 コンポーネントを開きます。

2 アクションの行を選択します。

3 ［**Action**］メニューから、［**New Action**］、［**Log**］の順に選択します。［Log］ダイアログボックスが表示されます。



4 ［**Log to**］ラジオグループで、メッセージを書き込む場所を選択します (前に扱った場所についての説明を参照)。

5 ［**Log Level**］スピンコントロールを使用して、この Log アクションの優先度レベル (1 から 10 まで) を選択します。デフォルトでは 5 に設定されています。一般的に、重要度が高いメッセージには、高い値を割り当てます。ここで割り当てる優先度は、前の節で選択したしきい値と比較されます (前の節を参照)。優先度がしきい値と等しいまたはしきい値より高い場合、メッセージは記録されます。それ以外の場合では、メッセージは記録されません。

6 ［**Log Expression**］テキストフィールドに、文字列または ECMAScript 式を入力します (テキストフィールドの右側にある小さなアイコンをクリックして Expression Builder にアクセスし、ピックリストの選択肢を使用して式を作成できます)。

7 コンポーネントの実行ごとにログファイルのデータをクリアする場合は、［**Clear the Log File**］をオンにします。

ログファイルの詳細については、73 ページ「システムメッセージの表示」を参照してください。

Map アクション

Map アクションは、DOM ノードの入力 / 出力のマッピングです。これにより、あるドキュメントのコンテキストのデータが別のドキュメントのコンテキストに転送 (または、オプションで変換) されます。コンテキストには 2 つの部分があります。通常、最初の部分で DOM を特定し、2 つ目の部分で DOM 内の場所を特定します。Composer の基本ドキュメントコンテキストは、XPath 式を通して特定される要素の場所 (場所として参照) と結合された DOM 名として表されます。通常 DOM 名は、入力、入力 1、入力 (n)、一時、出力、またはコンポーネントにロード済みの名前の付いた DOM です。DOM 内の場所を特定する XPath 式には、「/」で区切られたパス要素があります。

注記： Composer のコンテキストは、それ自体が XPath 式の単なる別名または省略形であるグループ名である場合もあります。

XPath および ECMAScript 式について

Map アクションを作成すると、XPath および ECMAScript という 2 種類のメソッドから選択して、DOM 内での場所を特定できます。デフォルトは XPath です。これは基本の特定メソッドです。

基本メソッド : Xpatht の単独使用

XPath を使用する主な目的は、XML ドキュメントの部分 (つまり、要素および属性) を特定または検索することです。XPath は文字列、数値、およびブール値を単純な式構文を通して操作するための基本機能も備えています。XPath は、要素ノード、属性ノード、およびテキストノードを含む、DOM ノードを特定します。

XPath はパターンの一致に基づいています。ターゲットドキュメントのノードを解決する要素名のパターンを入力します。ほとんどの場合、XPath によって、入力したパターンと一致する特定ノードを含む「ノードリスト」が返されます (多くの XPath 式では、1 つのノードのみが返されますが、複数のノードが返されることもあります)。その他の場合、XPath により、プリミティブ値 (文字列、数値、またはブール値) が返されることもあります。

XPath 式を取るすべての Composer ダイアログボックスでは、Expression Builder のピックリストを使用して式を作成できます (後の「ECMAScript を使用して式を作成する」を参照)。

完全な XPath の仕様については、**http://www.w3.org/TR/xpath** を参照してください。

注記： XPath の仕様は、Composer の \Doc ディレクトリの下層にもあります。

他のメソッド : ECMAScript 内での XPath の使用

DOM 内の場所をアドレスするもう 1 つのメソッドは、XPath で ECMAScript を使用することです。このメソッドは、厳密な XPath アドレッシングの範囲を超える場合に使用します。ECMAScript は、ホスト環境 (つまり Composer) でオブジェクトを操作するための、オブジェクト指向のスクリプト言語です。ECMAScript (ECMA-262 および ISO/IEC 16262) は、JavaScript (Netscape) および JScript (Microsoft) の両方の基盤となる標準ベースのスクリプト言語です。ECMAScript は、Composer のグラフィカルユーザインタフェースのような、ホスト環境における既存の機能を補完および拡張する目的で開発されました。ホスト環境として、Composer は ECMAScript にさまざまなオブジェクト (XML ドキュメントを含む) へのアクセスを提供し、処理できるようにします。代わりに、ECMAScript は、これらのオブジェクトで動作できる Java のような言語を提供します。

Composer の組み込み ECMAScript インタープリタは、XPath() と呼ばれるカスタムメソッドを認識します。これにより、次のような式を使用できます。

```
Input.XPath("Inventory/Books/Engineering")
```

このタイプの式の作成は、Composer の Expression Builder 機能を使用することにより大幅に容易になります。(後の「ECMAScript を使用して式を作成する」を参照)。

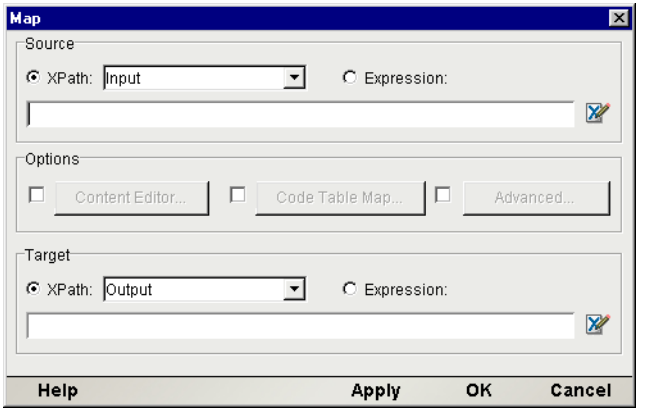
Map アクションの追加

➢ Map アクションを追加する

1 コンポーネントを開きます。

2 Map アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。

3 [**Action**] メニューから、[**New Action**]、[**Map**] の順に選択します。[Map] ダイアログボックスが表示されます。



4 ［**Source**］のタイプは、［XPath］です。プルダウンメニューから、DOM (［nput］、［Output］、または［Temp］) を選択し、次に、マップする要素を検索する XPath 式を入力します。

注記：［**Expression Builder**］をクリックして、XPath 式の作成に Composer を利用することもできます (161 ページ「XPath Expression Builder の使用」を参照)。

DOM 名と XPath はともに、Map アクションのソースコンテキストを示します。

5 ［**Target**］で、手順 4 から 5 を繰り返します。

6 ダイアログボックスの中央にある［**Options**］で、［**Content Editor**］、［**Code Table Map**］、または［**Advanced**］、あるいはこれら全部もしくはこのうちの 2 つをオンにし、マッピングの詳細な設定を行います。

注記：［**Content Editor**］オプションおよび［**Code Table Map**］オプションの詳細については、285 ページ「要素の変換」を参照してください。［**Advanced**］オプションについては、後の節で説明します。

7 ［**OK**］をクリックします。アクションモデルペインに Map アクションが表示されます (次の図を参照)。

注記：［Apply］ボタンを押すと、ダイアログボックスを閉じずに Map アクションの結果を表示できます。これにより、Map アクションを繰り返し編集しても、結果を迅速に確認できます。

デフォルトのマッピング動作

Map アクションを使用して DOM 内で要素と属性をマップすると、あるデフォルトの動作が行われます。デフォルトの動作は、次の表のとおりです。

表 7-2

マップのタイプ	デフォルトの動作
リーフ要素からリーフ要素へ	要素のデータのみを転送します。
リーフ要素から分岐要素へ	要素のデータのみを転送します。
分岐要素からリーフ要素へ	分岐の下層にあるすべてのチャイルド要素と属性のデータを含む分岐全体を転送します。
分岐要素から分岐要素へ	ターゲットの現在の分岐を削除した後、前のタイプと同様に分岐全体を転送します。

マップのタイプ	デフォルトの動作
リーフ要素のリストにある特定のリーフ要素から要素へ	選択したリーフ (または要素のインスタンス) からターゲット要素へ要素のデータを転送します。
属性から属性へ	属性のデータのみを転送します。
要素から属性へ	要素のデータを属性のデータへ転送します。
属性から要素へ	属性のデータのみを転送します。

これらの動作の多くは、[Advanced] マッピングダイアログボックスでのオプションの使用によって、アクションごとに変化します (次の節を参照)。

マークアップを含むリーフ要素

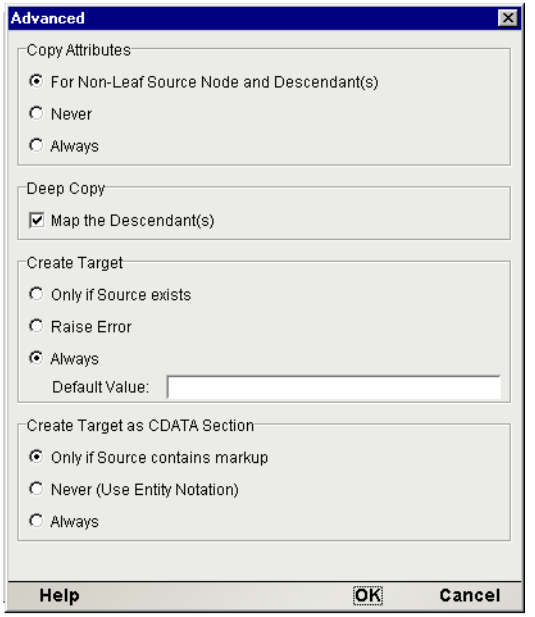
ランタイム時に Java または ECMAScript 操作を使用して要素をマップすると、ある特別な状況が発生することがあります。要素は、マークアップ、つまり < および > などの無効な文字を含む文字列を含むデータを受け取る可能性があります。これにより、Composer がそのような要素の加工されていないコンテンツをそのまま出力 DOM のノードにマップした場合に出力ドキュメントに異常が発生するというマッピングの問題が生じます。

Composer では、ターゲットドキュメントにその場で作成される CDATA セクションにマークアップを含むデータをマッピングすることでこの問題を解決します。

注記 : 設計時にマークアップを手動で入力すると、少々異なる動作が起こります。設計時に、(マウスボタンを右クリックし、[**Edit Data**]) を選択して) マークアップデータをノードに入力すると、マークアップ文字はその場でエンティティ化されます。加工されていない XML をテキストビューで表示すると、手動で入力した「<」文字は、`<` に変換されていることなどがわかります。次に、エンティティ化されたデータは出力に直接マップされます。

詳細なマッピングオプション

[Map Action] ダイアログボックスで [Advanced] チェックボックスをオンにすると、次のダイアログボックスが表示されます。このダイアログボックスで設定するオプションは、現在の Map アクションにのみ適用され、次のアクションには適用されません。



このダイアログボックスのオプションにより、入力 DOM ノードの出力 DOM へのマッピングを詳細に制御できます。

［Copy Attributes］

このコントロールのグループでは、属性のマップ方法を指定します。このグループには、次の 3 種類のラジオボタンがあります。

- ［**For Non-Leaf Root Nodes and Dependents**］— デフォルトでオンに設定されているこのボタンでは、Composer の標準 (デフォルト) のマッピング動作を指定します。非端末 (非リーフ) 要素を出力にマップすると、要素 (属性を除く) とそのチルドレン要素が出力にマップされます。チルドレン要素の属性データは含まれますが、元の (ペアレント) 要素の属性データは含まれません。
- ［**Never**］— このオプションでは、(ペアレントまたはリーフノードにかかわらず) 属性データがマッピングにより継承されないことを指定します。
- ［**Always**］— すべてのノードのすべての属性データが出力にマップされます。

［Deep Copy］

Composer はデフォルトで、分岐全体 (つまり、ターゲットノードとすべてのチャイルドノード) を一度にマップします。場合によっては、この「ディープコピー」動作をオフにして、チャイルド要素を除いてペアレント要素のみをコピーします。Composer の標準のディープコピー動作を無効にする場合、［Map the Dependents］というチェックボックスをオフにします。

［Create Target］

［Create Target］オプションでは、ソースノード (または分岐) がソース DOM に存在するかどうかによって、［Map Action］ダイアログボックスの［Target］で指定したマップ先のノード (または分岐) をオプションで作成できます。デフォルトの動作では、Composer はマッピングのソース XPath で指定したノードがランタイムのソース DOM に含まれているかどうかにかかわらず、ターゲットを作成します。

例 : ［Map Action］ダイアログボックスで、次のようなソース XPath を指定したとします。

```
$Input/Root/MySourceElement
```

一方、［Target］では、次のように指定したとします。

```
$Output/Root/MyParentNode/SomeOtherElement
```

使用する入力ドキュメントが `Root/MySourceElement` に対応するノードを持たない場合、Composer はあくまでも (デフォルトで) 出力 DOM に空の `Root/MyParentNode/SomeOtherElement` ノードを作成します。しかし場合によっては、この状況を望まないこともあります。このような場合、［Advanced］マッピングダイアログボックスにあるこのラジオボタンを使用すると、デフォルトの動作を変更できます。

注記 : ［Map Action］ダイアログボックスで［Code Table Map］が選択されている場合、［Create Target］オプションは無効です。

このラジオボタンループで使用できるオプションは次のとおりです。

- ［**Only if Source Exists**］— ソース XPath で指定したノードが入力ドキュメントに存在しない場合、Map アクションはスキップされます (出力 DOM にターゲットノードは作成されません)。
- ［**Raise Error**］— このボタンをオンにすると、ソース XPath で指定したノードが入力ドキュメントに存在しない場合、ランタイム時にはエラーであるとみなされます。必要に応じて、Try/OnError ブロック内で Map アクションをラップすると、エラー処理に備えることができます。
- ［**Always**］— デフォルトの動作です (常にターゲットノードが作成されます)。このボタンが選択されている場合、近くにある［Default Value］テキストフィールドが有効になり、ターゲット要素のデフォルト値をオプションで入力できます。

CDATA セクションとしてのターゲットの作成

このラジオボタンクループでは、要素のデータを CDATA セクションにマップする方法を制御できます。オプションは次のとおりです。

［**Only if source contains markup**］— このオプションでは、ソースデータが XML タグ、HTML タグ、または「無効な」文字が使用されているその他のタイプのマークアップを含む場合、データはターゲット DOM の CDATA セクションにそのまま配置されます。これは Composer のデフォルトの動作です。

［**Never**］— このオプションを設定すると、ソースデータが出力の CDATA セクションにラップされることはありません。ソースデータに含まれている無効な文字は、たとえば、> の場合、**>** になるなど、適切にエスケープされたエンティティに出力側で変換されます。

［**Always**］— ソースデータは、その形式にかかわらず出力の CDATA セクションにラップされます。

XPath Expression Builder の使用

［Map Action］ダイアログボックスでは、該当するテキストフィールドの右端にある［Expression Builder］ボタンを選択して、独自の XPath 式を作成できます。表示される［XPath Expression Builder］ダイアログボックスには、クリックするだけで有効な XPath 構文を作成できるピックリストがあります。これは、基本の XPath アドレッシングの範囲を超えて、XPath のより強力な機能のいくつかを使用する場合に特に便利です。

exteNd Composer では、W3C で採用されてるアドレッシング構文を使用します。XPath 構文は、基本的な概観では URI アドレスと同様ですが、XML のアドレッシングおよび操作のための詳細で強力な機能を多く備えています。より一般的な構文ルールのいくつかは、次の表のとおりです。

表 7-3

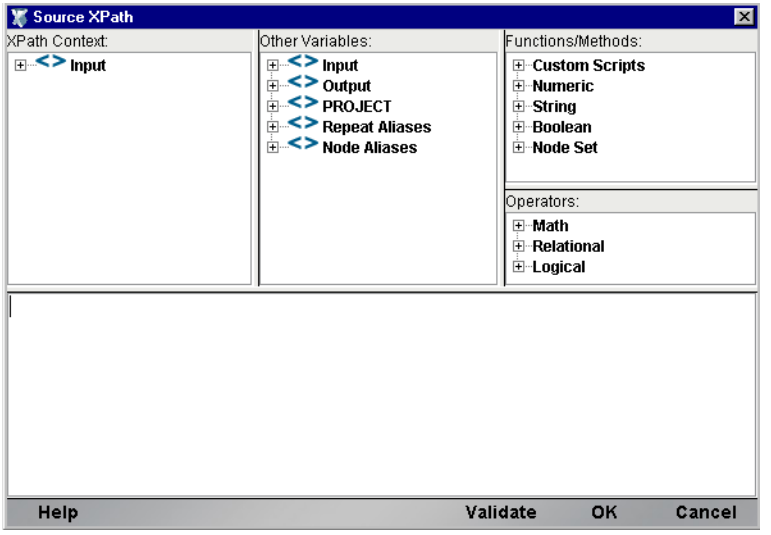
XPath 構文	説明
/	単一のスラッシュは要素への絶対パスを示します。/ABC では ABC のルート要素が選択されます。
//	二重スラッシュはパス内のすべての要素を示します。//ABC では、ABC が現れるすべての箇所が選択されます。//ABC//DEF では、ABC のチャイルド要素であるすべての DEF 要素が選択されます。
*	アスタリスクでは、前のパスで指定されたすべての要素が選択されます。*ABC/DEF では、要素 ABC/DEF で囲まれたすべての要素が選択されます。//* では、すべての要素が選択されます。

XPath 構文	説明
[ ]	角括弧は特定の要素を示します。/ABC[3] では、ABC の 3 番目の要素が選択されます。これはフィルタ (SQL での Where 節と同様) として使用できます。//ABC["Table"] では、「Table」というコンテンツを持つすべての要素が選択されます。
@	アット記号では、特定の属性を持つ要素が選択されます。/ABC@name では、name という属性を持つ、ABC 内のすべての要素が選択されます。
|	縦線では、複数のパスを指定できます。//ACB|//DEF では、ABC および DEF 内のすべての要素が選択されます。
$	ドル記号では、現在のドキュメント以外の他のドキュメントを参照できます。INVOICEBATCH/INVOICE[SELLER/NAME= $PROJECT/USERCONFIG/COMPANYNAME]
function()	XPath には、XPath のアドレスに追加できる多くの関数があります。たとえば、//*[count(*)=2] では、2 つのチャイルド要素を持つすべての要素が選択されます。
math operator()	XPath には、XPath のアドレスに追加できる多くの算術演算子があります。たとえば、/ABC|position() mod 2 = 0] では、ABC 内のすべての偶数要素が選択されます。

演算子の完全なリストについては、**http://www.w3.org/TR/XPath** を参照してください。

➢ XPath を使用して式を作成する

1. コンポーネントを開きます。
2. ［**Action**］メニューから、［**Map**］アクションを選択します。
3. ［**XPath**］ラジオボタンがオンになっていることを確認します。
4. ［**Expression Builder**］ボタンをクリックします。［Source XPath］ダイアログボックスが表示されます。



5 ペインでアイテムをダブルクリックして式を作成します。

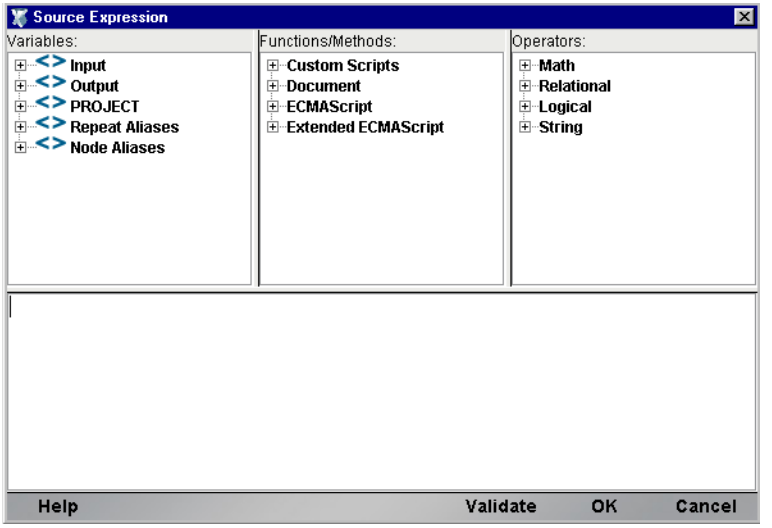
6 式の構文が正しいかどうかを検証します (［**Validate**］ボタンを使用します)。

7 ［**OK**］をクリックします。

ECMAScript Expression Builder の使用

Map アクションの［ECMAScript］ラジオボタンをオンにすると、［ECMAScript Expression Builder］が表示され、有効な ECMAScript 構文を簡単に作成できます。これは、厳密な XPath アドレッシングの範囲を超えて、Composer による ECMAScript アドレッシングのより強力な機能をいくつか使用する場合に便利です。

ECMAScript Expression Builder は、次の図のとおりです。



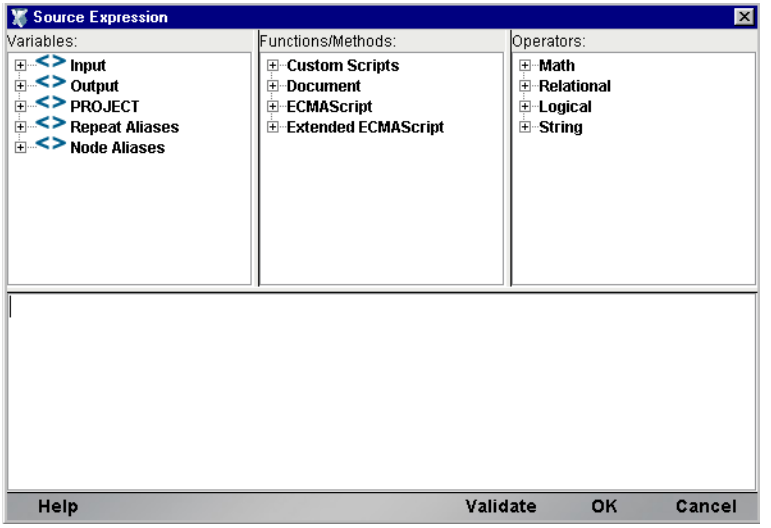
ピックリストのオブジェクトは、使用頻度が高いものから順に表示されます。オブジェクトのすべてのプロパティおよびメソッドにも順序があります。常にプロパティが最初にアルファベット順に表示され、その後にすべてのオブジェクトのメソッドがアルファベット順に表示されます。

［**Functions/Methods**］ピックリストおよび［**Operators**］ピックリストのすべてのアイテムには、関連付けられたツールヒントがあります。ツールヒントを表示するには、該当するアイテムの上にカーソルを合わせます。［**Variables**］ピックリストのアイテムにカーソルを合わせると、アイテムに関連付けられたデータが表示されます。

注記： 複雑な ECMAScript 式を作成できますが、それらの式は、DOM と DOM 内のアドレスを構成するドキュメントコンテキストを返す必要があります。

➢ ECMAScript を使用して式を作成する

1. コンポーネントを開きます。
2. ［**Action**］メニューから、［**Map**］アクションを選択します。
3. ［**Expression**］の隣にあるラジオボタンをオンにします。
4. ［**Expression Builder**］ボタンをクリックします。［Source Expression］ダイアログボックスが表示されます。



5 ペインでアイテムをダブルクリックして式を作成します。

6 ［**Validate**］をクリックすると、式の構文が正しいかどうかをオプションで検証できます (これによって式が実行されることはありません。単に解析されるだけです)。

7 ［**OK**］をクリックします。

Send Mail アクション

Send Mail アクションは、コンポーネントの実行中に、電子メールメッセージを作成したり送信したりします。Send Mail アクションの作成では、電子メールメッセージのルーティングとコンテンツに使用する情報を Composer に指示する式を作成します。電子メールのデータには、電子メールの受信者のアドレス、送信者の電子メールアドレス、電子メールメッセージの件名、電子メールメッセージの本文、および電子メールを送信する SMTP メールサーバの名前が含まれます。

Send Mail アクションを使用する場合の例は、次のとおりです。

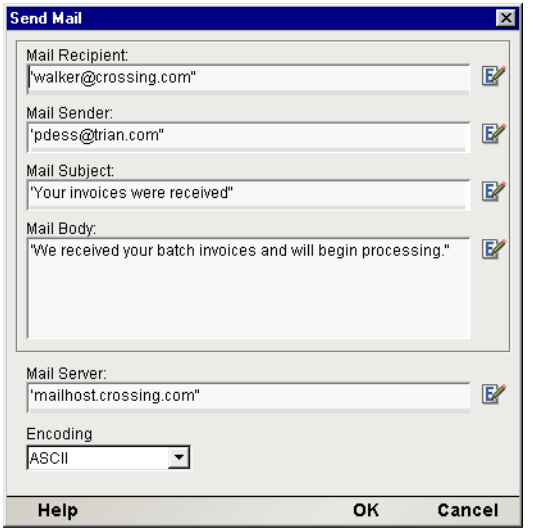
- 取引先からの請求書を転送して経理担当者に人為的な処理を促すサービスまたはコンポーネントにおいて。
- 早急な措置を必要とする、致命的なエラーの状況を直ちに通知する場合。電子メールはポケットベルにもルーティングできます。
- Web での発注に対するお礼のメッセージを生成する場合。

➢ Send Mail アクションを追加する

1 コンポーネントを開きます。

2 Send Mail アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。

3 ［**Action**］メニューから、［**New Action**］、［**Send Mail**］の順に選択します。［Send Mail］ダイアログボックスが表示されます。



4 ［**Mail Recipient**］フィールドに、有効な ECMAScript 式を入力して受信者の電子メールアドレスを指定します。ハードコード化しているテキスト値がある場合は、結果のテキストが引用符で囲まれていることを確認してください。

5 ［**Mail From**］フィールドに、有効な ECMAScript 式を入力して送信者の電子メールアドレスを指定します。自分が送信者である場合、自分の電子メールアドレスを入力します。形式は "`name@domain.extension`" である必要があります。ハードコード化された定数値がある場合は、結果のテキストが引用符で囲まれていることを確認してください。

6 ［**Mail Subject**］フィールドに、有効な ECMAScript 式を入力して電子メールの件名を指定するか、件名を入力します。ハードコード化された定数値がある場合は、結果のテキストが引用符で囲まれていることを確認してください。たとえば、コンポーネント名、日付 / 時間、およびエラーを示す式の連結から、エラーメッセージの件名を生成できます。

7 ［**Mail Body**］フィールドに、有効な ECMAScript 式を入力して電子メールの本文テキストを指定するか、本文テキストを入力します。ハードコード化された定数値がある場合は、結果のテキストが引用符で囲まれていることを確認してください。

8 ［**Mail Server**］フィールドに、有効な ECMAScript 式を入力して SMTP メールホストの名前を指定するか、電子メールメッセージを送信する SMTP メールホストの名前を入力します。ハードコード化された定数値がある場合は、結果のテキストが引用符で囲まれていることを確認してください。

9 必要があれば、ダイアログボックスの下部近くにある［**Encoding**］プルダウンメニューから正しい文字エンコードを選択します。

10 ［**OK**］をクリックします。

Switch アクション

Switch アクション (Java および C 言語の switch ステートメントにより呼び出される) は、特定の入力変数の値または XPath 式に基づいてアプリケーションを適切なカスタム論理へ分岐させるためのものです。Switch アクションは、ネストされた Decision アクションの連続を使用する必要性を回避できる便利なアクションです。このアクションを使用すると、複数のアクションを除外したり、それらを一貫した、わかりやすく読みやすい単一のアクションに統合したりすることによって、アクションモデルの読みやすさを大幅に向上させることができます。

ケースについて

Switch アクションでは、一連の値または選択肢 (「ケース」) を入力値と比較します。値または選択肢は、静的または動的のいずれかです。入力値と選択肢のうちの 1 つに完全一致が起こると、実行は選択肢の下層にリストされているアクションに分岐します。ケースは、一連の if/else ステートメントと同様に「リストされた順序で」テストされ、一致が起こると、一致論理の実行は Switch アクション内の他の実行を除外します。

ケースに関連付けられたカスタム論理は、単一のアクションまたはアクションのブロックで構成でき、アクションには Composer の標準 (基本的または高度な) アクションと特定の接続に固有のアクションを含むことができます。

Switch の例

受信する XML ドキュメントが商品の小売り注文であり、アプリケーションで実行する必要のあるタスクの 1 つが顧客の住所に基づく発送方法の決定であるとします。Switch アクションへの入力には、次のような XPath 式が考えられます。

```
$Input/Order/Customer/Address/Country
```

Switch アクションのケースの値および各選択肢に対して関連付けられた論理は、次のような場合があります。

```
CASE:"USA"
   CALL shipMethod = (weight < 10) ? "FedEx" : "UPS";
```

```
CASE:"ANGOLA"
   CALL shipMethod = "Air Gemini";
CASE:"ARGENTINA"
   CALL shipMethod = "International First Services";
CASE:"AUSTRALIA"
   CALL shipMethod = "Ansett International";
.
.
.
DEFAULT:
   CALL shipMethod = "UPS";
```

ランタイム時には、Order/Customer/Address/Country の入力 DOM 要素の値は、一致が起こるまで、"USA" を始め、後続の各ケースの値と比較されます。この例では、"ANGOLA" で一致で起こった場合、"Air Gemini" を (ECMAScript) 変数 shipMethod に割り当てる Function アクションが実行され、その後直ちに Switch アクションは終了し、次に実行は、Switch アクションの後に最初のアクション (存在する場合) へと続きます。

注記： Java および C 言語のケースステートメントに組み込まれている「フォールスルー」動作は、Composer の Switch アクションではないため、明示的な Break アクションをケースのアクショングループに挿入する必要はありません。一致が起こると、次のケースへのフォールスルーが起こることはありません。

前の例は、Decision アクションの連続として作成することもできます。Decision アクションの連鎖の擬似論理は、次のようになります。

```
country = inputValue
if (country == USA)
   ship via A or B
   else if (country == ANGOLA)
      ship via C
      else if (country == ARGENTINA)
         ship via D
         else if (country == AUSTRALIA)
            ship via E
[etc]
               else ship via Default shipper
```

Switch を使用すると、この種類のコードに特有の階段状のインデントと if/else 論理の繰り返しを使用する必要がなくなります。また、読みやすく、保守の容易なコードを作成することにもつながります。一般に、長い条件文を取り扱う場合は、Switch アクションの使用を考慮する必要があります。

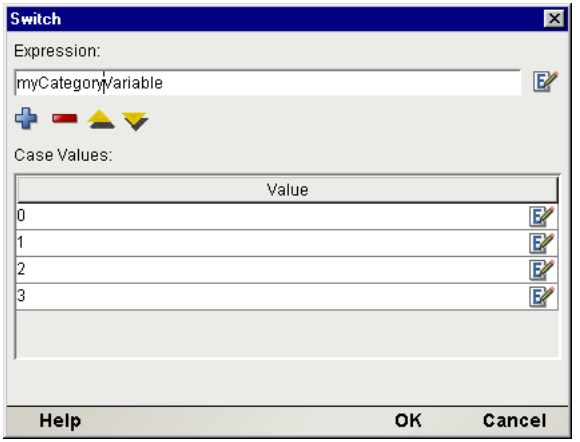
デフォルトのケースについて

各 Switch アクションの下の最後の「ケース」には、常に Default というラベルが付きます。この行は自動的に生成され、削除することはできません。Default の下に配置されるアクションは、Switch アクションがランタイム時に、ケースのリストで一致するケースと遭遇しなかった場合に限り、実行されます。

注記： Default の下にアクションを配置することを要求されていない場合、Log アクションまたは Raise Error アクションであっても、「一致なし」の場合に備えて何らかの予備論理を用意しておくことがプラグラム作成上、望まれます。

➢ Switch アクションを追加する

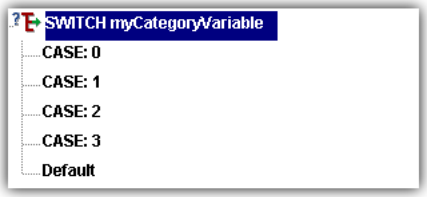
1. コンポーネントを開きます。
2. Switch アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
3. [**Action**] メニューから、[**New Action**]、[**Switch**] の順に選択します。[Switch] アクションダイアログボックスが表示されます (次のダイアログボックスに指定されているテキスト値はデフォルトではありません。これらの値は単に例として示されているものです)。



4. ダイアログボックスの上部にある [**Expression**] に、XPath または ECMAScript 式を入力します。これは、Switch アクションに対する入力値です。
5. コンボボックスに、前の手順で指定した入力値と比較される、静的な文字列値または ECMAScript 式を入力します。各ケースの値は、ランタイム時には、リストした順序で比較されます。(ヒント：パフォーマンスを最適にするには、可能性の高い一致から順にリストします)。

   注記： デフォルトでは、ケースの新しいエントリは既存のリストの最後に追加されます。ただし、選択肢を選び、必要に応じて [上] ボタンと [下] ボタンをクリックすると、選択肢の順序を変更できます。

6 ［**OK**］をクリックします。ダイアログボックスが閉じ、アクションモデルに新しい Switch アクションが表示されます (次の図を参照)。



アクションモデルに新しい Switch アクションを追加すると、ケースの値のリストが表示されます。ケースに独自のカスタム論理を関連付けるには、ケースをクリックし、次に新しいアクションを必要に応じて 1 つずつ追加します。新しいアクションを追加するには、マウスの右ボタンをクリックし、コンテキストメニューから［New Action］を選択します。アクションブロックに含めることができるアクションの数 (およびタイプ) に制限はありません。

➢ カスタムケース処理論理を追加する

1 アクションモデルで、処理論理の追加先となるケースを探します。

2 該当するケースの下にあるアクションの行をクリックします。

3 マウスを右クリックして、コンテキストメニューを表示します。［**New Action**］を選択し、サブメニューの使用可能なアクションから任意のアクションを選択します。

4 前の手順を繰り返し、必要なだけアクションを追加します。

Switch アクションの編集

Switch アクションの編集に使用する主なツールは、［Switch］アクションダイアログボックスです。ここでは、入力式を編集したり、ケースの順序を変更したり、ケースの式を編集したり、ケースの値を追加または削除したりすることができます。このダイアログボックスには、アクションモデルで任意の Switch アクションをダブルクリックするだけでアクセスできます。

アクションモデル自体 (設定のダイアログボックスを開かない場合) では、編集できる内容に制限があります。この制限は次のとおりです。

- Switch アクション (一番上の行) 自体で、切り取り、コピー、削除、および貼り付けの操作を行うと、すべての一致と関連付けられたアクションのリストを含む、Switch ブロック全体が切り取られたり、コピーされたりします。
- アクションモデルの選択されたケースの値を切り取るまたは削除することはできますが、(貼り付けによて) 新しいケースの値を「追加」することはできません。

- 切り取りまたは削除の操作では、ケース自体だけでなく、関連付けられたすべてのアクションが切り取られたり、削除されたりします。

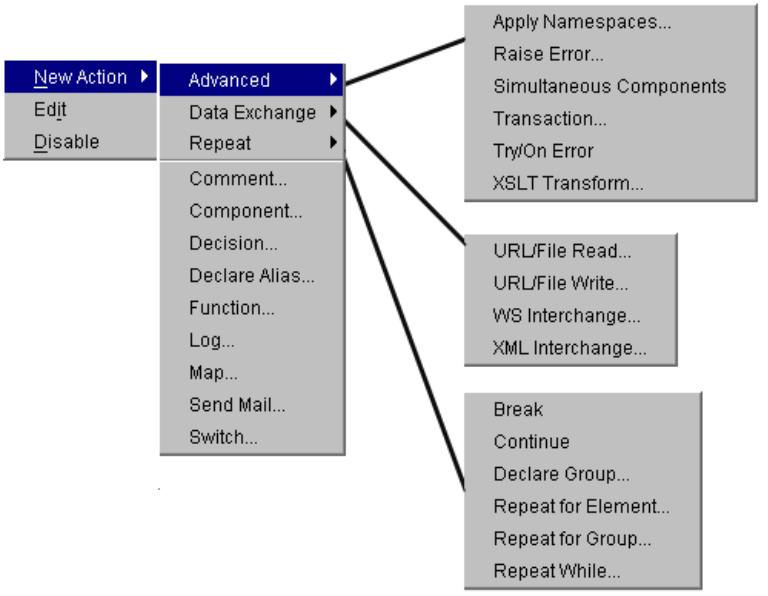
5 ケースのアクションリストにあるアクションは通常の方法で編集できます。

8 高度なアクション

前章では、コンポーネントを作成するときに使用できる基本的なアクションについて説明しました。この章でとりあげるアクションは、前述のものよりも高度な機能を備えており、入出力関連アクション、フロー制御構成要素、およびその他の追加アクションが含まれます。

この章で説明されるアクションは、[Action] メニューでネストされているサブメニューの下のコマンドを使用して作成できます。このサブメニューには、[**Advanced**]、[**Data Exchange**]、および [**Repeat**] が含まれます。また、アクションモデル内で右クリックしてアクセスすることもできます。

メニュー構造は、次のとおりです。



次の表に、高度なアクションを要約します。

表 8-1

高度なアクション	説明
Apply Namespaces	ネームスペースのプリフィックスをオーバーライドし、新しいネームスペースを宣言するか、すべてのネームスペースを無視します。
Raise Error	条件を評価し、その条件が真の場合は ERROR というグローバル変数に式の内容を書き込みます。単独で使用されると、例外が発生し、コンポーネントを停止し、サービスのコントロールを返します。Try On Error アクションの Execute 分岐内で使用されると、評価され、On Error 分岐のアクションにコントロールが渡されます。
Simultaneous Components	2 つ以上のコンポーネントを同時に実行できます (すなわち、マルチスレッド形式)。
Transaction	非コンテナ管理のサービスの一部として配備されるコンポーネントの［*User Transaction*］コマンド (*begin*、*commit*、*rollback* など)、またはコンテナ管理の EJB 配備の一部となるコンポーネントの *setRollbackOnly* を呼び出すことができます。
Try On Error	アクションのセットを実行することにより、エラーが発生するアクションに応答します。Try On Error アクションは、基本的にエラートラップ / ソリューションアクションです。
XSLT Transform	XSL ファイルの指示に従って XML ファイルを変換します。出力は、一般的に Web ブラウザで XML ファイルをレンダリングするために使用されます。

注記 : 一部のアクションの使用例については、第 11 章「共通タスクへのアクションの適用」を参照してください。

Apply Namespaces アクション

コンポーネントでは、常に有効な XML ドキュメント (つまり、スキーマに対して検証されたドキュメント) を受信し、データを適切にマップおよび変換して、有効な XML ドキュメントを送信することが理想的です。しかし、現実の世界では、必ずしもそうであるとは限りません。したがって、XML ドキュメントの検証のための何らかの手段を持つことが重要です。

スキーマは、ネームスペースと結合することによって、検証を実行するメカニズムを提供します。しかし、スキーマ、ネームスペース、およびプリフィックスは、XML 変換を実行中に容易に問題を起こす可能性があります。ドキュメントの検証とマーシャリングを含む単純な処理の場合、Composer のスキーマサポート、XML テンプレート機能、およびドラッグアンドドロップによるマッピングは、ネームスペースとネームスペースプリフィックスを管理する上で通常は心配がないことを意味します。しかし、次の状況では、ドキュメントでネームスペースやネームスペースプリフィックスの特別な処理が必要となる可能性がある、ドキュメントの変換を含む場合が多くあります。

- ビジネスパートナーは、同じネームスペースに属する有効なドキュメントを交換しますが、それぞれが別のネームスペースプリフィックスを使用します。1 つのパーティが他のパーティのドキュメントを検証または操作するためには、各パートナーのネームスペースプリフィックスをドキュメントで宣言する必要があります。

- XML テンプレートは、ドキュメントのネームスペースにプリフィックスを解決するために必要ではありません (つまり、入力 XML テンプレートが System {Any})。しかし、Map アクションを正しく機能させるには、Map のソースとターゲットで使用されるプリフィックスをネームスペースで解決できなければなりません。

また、ネームスペースを単にすべて無視したい場合など、その他の場合もあります。このような場合や、他の多くの XML 処理の場合では、Composer のデフォルトのスキーマと XML テンプレートのサポートにより提供されるプリフィックスとネームスペース処理を変更または上書きする方法が必要です。

Apply Namespaces アクションを使用すると、コンポーネントのアクションモデル内の XML ドキュメントに対して有効なネームスペースとネームスペースプリフィックスを管理するためのメカニズムが提供されます。このアクションでは、1 箇所でドキュメントに対するすべてのネームスペースとプリフィックス宣言を統合するだけでなく、コンポーネントによって使用される XML テンプレートで宣言されたものを上書きしたり、ネームスペースをすべて無視したりすることができます。

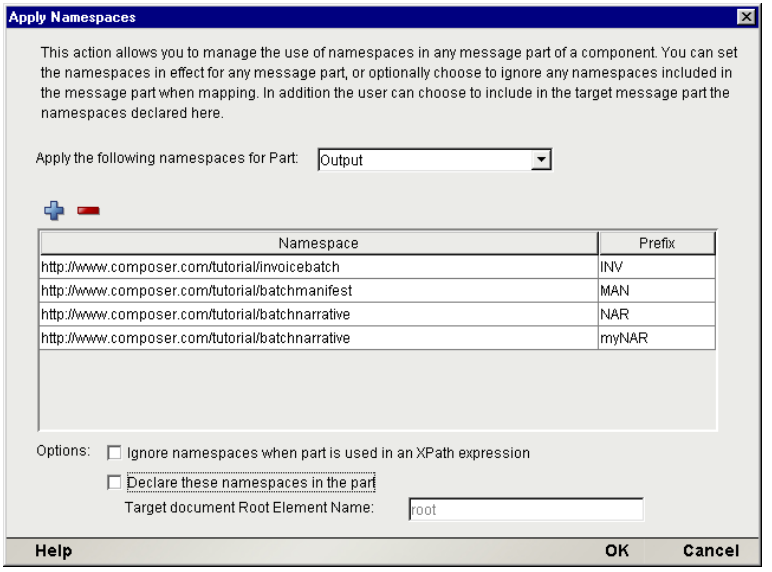
Apply Namespaces アクションは、任意のメッセージパート (Input、Input1、Temp、Temp1、または Output) に適用できます。また、アクションモデルで指定された条件に基づいて有効なネームスペースを効果的に変更することにより、単一のメッセージパートに対して複数の Apply Namespaces アクションを持つこともできます。任意のパートに対して宣言されたネームスペースは、アクションモデルの最後に達するか、またはそのパートの別の Apply Namespaces アクションが実行されるまで有効になります。つまり、最新の Apply Namespaces アクションのみが任意の単一のパートに対して有効になります。

新しいコンポーネントの作成時に、XML テンプレートによってネームスペースが宣言される場合、出力パートに対して Apply Namespaces アクションが自動的に作成されます。コンポーネントの作成後、任意またはすべてのメッセージパートに対して追加の Apply Namespaces アクションを手動で作成できます。いずれの場合も、アクションダイアログボックスを初めて開いたときに最初に指定されたネームスペースとプリフィックスは、XML テンプレートから直接取得されます。その後、このアクション内では、必要に応じてネームスペースやプリフィックスを追加、変更、または削除できます。

➢ Apply Namespaces アクションを作成する

1 Namespace アクションを適用するコンポーネントを開きます。

2 [**Action**] メニューから、[**New Action/Advanced**]、[**Apply Namespaces Action**] の順に選択します。[Apply Namespaces] ダイアログボックスが表示されます (次を参照)。



3 [**For Part**] ドロップダウンリストから、ネームスペース (つまり、出力) を適用する場所を選択します。このコントロールにより、ネームスペース宣言のリストを適用できるメッセージパートが表示されます。

4 [+] アイコンをクリックして行を追加し、[-] アイコンをクリックして行を削除します。「ネームスペース」を追加するときは、URI と「プリフィックス」を表示された列に入力します。

注記： プリフィックス表には、[For Part] コントロールに表示されたドキュメントに有効なネームスペース宣言がすべて表示されます。新しい Apply Namespaces アクションを作成した後、表には、選択したパートの宣言のリストが含まれる場合と含まれない場合があります。宣言のリストは、XML テンプレートのネームスペース宣言パネルで定義された宣言から最初は作成されます。パートの XML テンプレートが System{Any} である場合、またはスキーマベースではない場合、宣言がテンプレートのネームスペース宣言パネルに追加されない限り、リストは空になります。

注記： 単一のメッセージパートの宣言リスト内において、プリフィックスは固有でなければなりません。しかし、固有なプリフィックスに関連付けられている場合は、重複したネームスペース URI エントリを持つことができます。これにより、同じネームスペース URI に関連付けられたプリフィックスを複数宣言することが可能になります。

5 **オプション**: Map アクションソース XPath で XML ローカル名のみを使用して要素を検索する場合、ドキュメントが Map アクションの［Source］オプションで使用されるときに［**Ignore Namespaces**］のチェックボックスをオンにします。

注記： これは、Map アクションの指定の制限を緩和し、一部の処理状況において Map アクションに間違ったプリフィックスが含まれていたり、それらの Map アクションのソース指定にプリフィックスがない場合に役立ちます。これにより、入力メッセージにプリフィックスが含まれるかどうかをテストする Apply Namespaces アクションを Decision アクション内に配置したり、別のドキュメントに入力をマップするために Map アクションの 1 つのセットを保持したりできます。つまり、このコンポーネントでは、入力にプリフィックスが含まれていることを通常は期待するため、すべての Map アクションはプリフィックス名を付けて設計します。プリフィックスのない入力があった場合は、Decision アクションにより、いずれの場合でも Map アクションが機能するために入力のネームスペースを無視するよう定義された Apply Namespaces アクションがアクティブになります。

注記： このオプションは、任意のパート (つまり、Input.setSkipNameSpaces(true)) で使用された setSkipNameSpaces() メソッドと同じ機能を果たします。このメソッドと Apply Namespaces アクションのうち、アクションモデルで最後に実行されたものが有効になります。

6 **オプション**: Action モデルで作成した出力ドキュメントのルート要素のネームスペースのセットを宣言するときにドキュメントがマップターゲットで使用される場合は、［**Declare These NameSpaces**］のチェックボックスをオンにします。このオプションは、出力で作成されたプリフィックス要素が Map アクションの結果として適切なネームスペースに解決されるよう、ほとんど常に［Output］をオンにしておきます。

注記： これにより、出力の受信者がドキュメントを正しく検証できるようになります。このオプションをオンにすると、Apply Namespaces アクションは、すでに宣言を含んでいる既存のドキュメントに新しい宣言を追加するためにも使用できます。

7 ［Target Document Root Element Name］は、ネームスペース宣言属性を含むルート要素の名前を指定します。ターゲットのメッセージパートがスキーマ検証で XML テンプレートに基づく場合、このコントロールは、Composer により自動的に書き込まれます。ターゲットのメッセージパートがスキーマ検証で XML テンプレートでない場合は (例 : System{Any})、値を入力しなければなりません。

8 ［**OK**］をクリックすると、新しいアクションがコンポーネントの Map アクションペインに追加されます。

Map アクション、XML テンプレート、ネームスペース、およびプリフィックス

XML テンプレートと、コンポーネントごとに処理される XML ドキュメントのネームスペースとプリフィックスは、すべて期待どおりに Map アクションが機能するかどうかに影響します。デフォルトでは、Map アクションを機能させるために、ソース XPath のプリフィックス / 要素名の組み合わせがフルネームに拡張されます。そして、同様のプロセスが、Map アクションによって参照されるメッセージパートで発生します。ソース指定とメッセージパートの間でマッチが見つかると、データまたはコンテンツモデルが Map アクションのターゲットにマップされます。最も重要な要素は、Map アクションのソースが XML メッセージパートと比較されるときに、プリフィックスがネームスペースに拡張されるかどうかです。ネームスペース解決が実行されない場合 (つまり、オフになっている)、Map アクションは常に機能します。

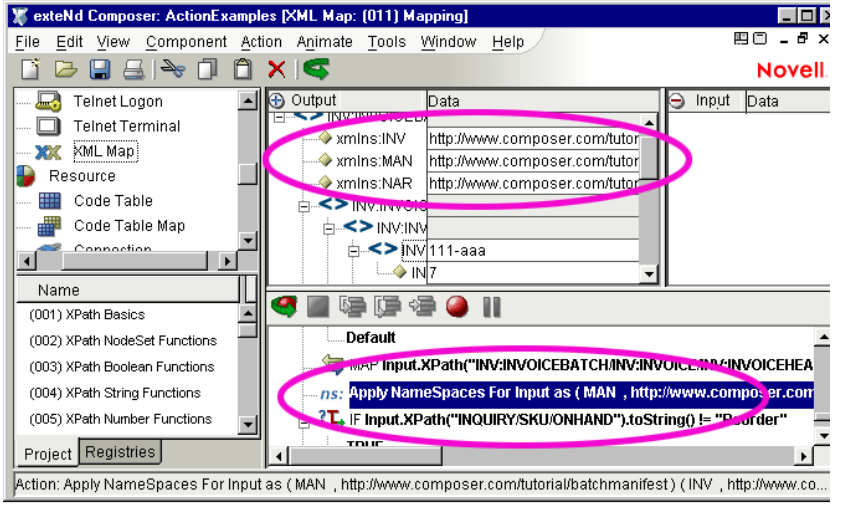
デフォルトでは、Composer によってネームスペース解決が実行されます。しかし、Composer でネームスペース解決が実行されないようにする方法が 2 つあります。最初の方法は、`Input.setSkipNameSpaces(true)` の場合と同様に、`setSkipNameSpaces()` メソッドを使用することです。2 番目の方法は、ドキュメントが Map Action のソースコントロールで使用される場合に、Apply Namespaces アクションを追加し、［Ignore Namespaces］をオンにすることです。

ネームスペース解決が実行されるときは、Map アクションを機能させるために 2 つの追加条件が満たされることが必要です。Map アクションで使用されるプリフィックスと、ドキュメントに存在するプリフィックスでは、(1) ネームスペース URI に解決されるということと、(2) ネームスペース URI が一致しなければならないということを満たしていなければなりません。最初の条件は、2 番目の条件の前提条件です。

最初の条件では、Map アクションソースで使用されるプリフィックス (受信することが期待されるもの) とランタイムドキュメントの要素に使用されるプリフィックス (実際に受信するもの) が拡張され、ネームスペース URI に解決できることが必要とされます。いずれかを解決できない場合、Map アクションの実行は失敗します。Map アクションを機能させるためには、ソース指定プリフィックスの拡張形式がマップされる XML ドキュメントの要素の拡張形式と一致していなければなりません (2 番目の条件)。Map アクションプリフィックスは、XML テンプレートまたは Apply Namespaces アクションで指定されたネームスペース URI に解決することにより拡張されます。XML ドキュメントの要素のプリフィックスは、XML ドキュメントで宣言されたネームスペース URI に解決することにより拡張されます (つまり、ルート要素の **xmlns:someprefix="someURI"** 属性)。Map アクションの期待されたネームスペース URI がドキュメントからの実際のネームスペース URI に一致しない場合、Map アクションの実行は失敗します。

例 : 出力メッセージへのネームスペース宣言の割り当て

新しいコンポーネントが作成され、出力メッセージがネームスペース宣言を含む XML テンプレートに基づく場合、Composer では、Apply Namespaces アクションをアクションモデルに自動的に追加します。コンポーネントを実行すると、このアクションによって、適切なルート要素、ルート要素のネームスペースプリフィックス、およびルート要素ネームスペース宣言属性が出力 XML メッセージに作成されます。通常、このアクションは、アクションモデルの開始時に適宜残されます。さらに、このアクションを使用すると、XML テンプレートで宣言されていない出力メッセージに新しいネームスペース宣言を追加できます。一般的な出力メッセージに対して Declare Namespaces アクションが定義される方法は、次の図のとおりです。



このコンポーネントの出力を受け取るコンポーネントまたはプログラムが同じネームスペースで機能するように設計されている場合、Apply Namespaces アクションを使用して出力メッセージで代替のネームスペースプリフィックスを追加できます。これを行うには、単に Apply Namespaces アクションを開いて［Add］ボタンを押すだけです。別のプリフィックスと関連付けるネームスペース URI をコピーし、新しい行に貼り付けます。次に、代替のプリフィックスを指定します。

注記： Temp ドキュメントを Map アクションのターゲットとして使用する場合は、同様の Apply Namespaces アクションを定義する必要があります。Temp ドキュメントは Map アクションのソースとターゲットの両方に設定できるため、Composer では目的がわからないため、アクションを自動的に作成しません。

例 : Ignoring Namespaces

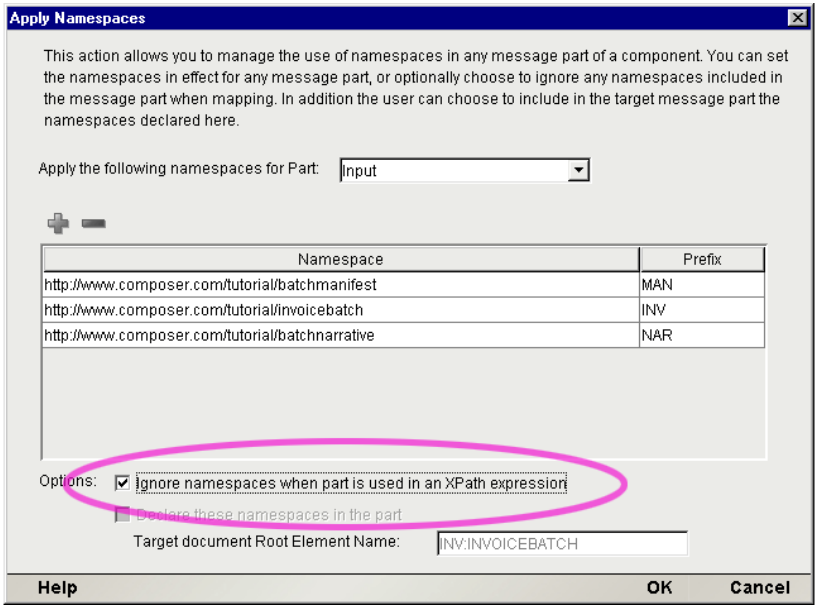
場合によっては、ネームスペースと、それらに関連付けられているプリフィックスは、コンポーネントのマッピングや変換の目的とは関係がないことがあります。恐らく、ドキュメントは、すでに検証されていても、バックエンドデータストアに挿入される前に (例 : JDBC コンポーネントを経由するリレーショナルデータベースや CICS/RPC コンポーネントを経由する CICS トランザクション) 再構成する必要があります。この場合、Map アクションでは、ドキュメントのローカル名のみを参照することにより容易かつ迅速に設計できる別の階層に入力 XML メッセージを再構成することのみを行います。この場合、ネームスペースをすべて無視する Apply Namespaces アクションを追加できます。これによって、定義したソース XPath のネームスペースプリフィックスを省略する Map アクションを作成できます。したがって、Map アクションのソースは、次のとおりではなく、

```
INV:INVOICEBATCH/INV:INVOICE/INV:INVOICEHEAD/INV:INVOICENO
```

次のように記述することができ、

```
INVOICEBATCH/INVOICE/INVOICEHEAD/INVOICENO
```

こちらの方がもっとわかりやすくなります。



Raise Error アクション

Raise Error アクションでは、コンポーネントの実行を中断し、ERROR という名前の ECMAScript グローバル変数に書き込む情報を指定できます。Raise Error は、指定した条件が真である場合にのみ実行されます。実行すると、Raise Error アクションによって、ERROR という名前のグローバル変数にその式のコンテンツが書き込まれます。Raise Error 条件は、次の 3 つの方法からいずれかを選択して指定できます。

- **［Raise Error Action］ダイアログボックスで指定されたエラー条件を使用する。**アクションが実行されると、エラー条件が評価され、真の場合は、エラー式のコンテンツが ERROR グローバル変数に書き込まれ、コンポーネント実行が中断され、実行が破棄されます。エラー条件が偽の場合は、アクションモデルの次のアクションが実行されます。
- **Decision アクションの決定式の使用**：エラー条件は、Decision アクションの決定式に条件を入力して指定できます。次に、Decision アクションの True 分岐に Raise Error ステートメントを配置します。ここでは、［Raise Error］ダイアログボックスのエラー条件で追加条件を指定したり、空白のまま残して変数 ERROR に書き込むエラー式を単に指定したりできます。アクションが実行され、すべての条件が真の場合、エラー式のコンテンツが ERROR グローバル変数に書き込まれ、コンポーネント実行が中断され、実行が破棄されます。Decision アクションまたは Raise Error アクションのエラー条件が偽の場合は、アクションモデルの次のアクションが実行されます。

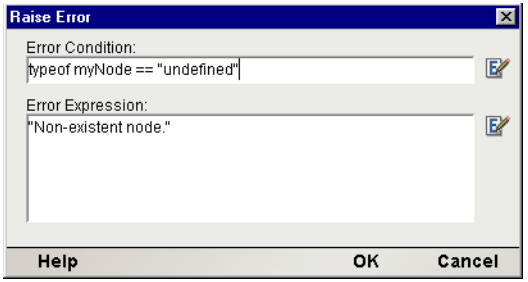
- **Try/On Error アクションの内部** : Try/On Error アクションの Execute 分岐の内部に前のいずれかのメソッドを配置することにより、コンポーネントで実行が中断されないようにし、エラーに応答したりエラーから回復したりできるようにします。出力をテストする他のアクションが正しく機能した後で、Try/On Error アクションの Execute 分岐の内部に前の 2 つのメソッドの 1 つを使用してエラー条件を作成します。Raise Error アクションを実行すると、コンポーネントの実行を中断する代わりに、Try/On Error アクションの On Error 分岐にコントロールが渡されます。ここでは、エラーを解決するか、またはエラーに応答するよう他のアクションを指定できます。

エラー条件を処理するのが適切であるか、または Raise Error を適宜使用するのが適切であるかを判断できます。たとえば、ランタイム時にエラー条件に遭遇したときにコンポーネントで実行を停止させる (および、実行中にサービスにコントロールを戻す) 場合は、Raise Error を単独で使用します。アクションによって例外が発生し (Composer のダイアログボックスとして表示されます)、アプリケーションサーバのコンポーネントが停止します。

一方、サービスによって Try/On Error アクション内から別のコンポーネントが呼び出されるとします (特に、**Try** 分岐の下)。他のコンポーネント内で、Decision アクションにより、XML ドキュメントの一部のデータを検査されます。データが有効である場合、コンポーネントでは実行を続行します。データが有効でない場合は、Raise Error アクションが実行され、ERROR 変数にメッセージが配置され、コンポーネントでは実行が停止し、サービスにコントロールが返されます。Try/On Error では、Raise Error が発生し、論理が Try/On Error アクションの On Error 分岐に渡されたことを検出します。On Error 分岐で、ログファイルへのメッセージの書き込みなど、ERROR 変数の操作を自由に行うことができます。

➢ Raise Error アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルで、Raise Error アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**Raise Error**］の順に選択します。［Raise Error Action］ダイアログボックスが表示されます。



4 真の場合はアクションにエラーを発生させる有効な ECMAScript 式を入力します (また、[**Expression Builder**] ボタンをクリックして式を作成することもできます)。この手順はオプションです。

5 ERROR グローバル変数に書き込むデータを決定する有効な ECMAScript スクリプト式を入力するか、または [**Expression Builder**] ボタンをクリックして式を作成します。

6 [**OK**] をクリックします。

Simultaneous Components アクション

Simultaneous Component アクションでは、2 つ以上のコンポーネントを同時に実行できます (つまり、独自の異なる実行スレッドで)。応答に比較的時間がかかるレガシーシステムへの問い合わせに依存する XML 統合アプリケーションでは、これは重要な機能です。たとえば、サービスで 2 つのデータソースから CICS RPC と JDBC を経由して情報を取得する必要があるとします。CICS の問い合わせにかかるラウンドターン時間は 5 秒で、JDBC の問い合わせには 4 秒かかるとします。2 つの問い合わせが順に行われた場合、データの待ち時間は 9 秒になります。しかし、両方のバックエンドシステムに同時に問い合わせることができる場合、合計待ち時間は約 5 秒に短縮されます。これは、大幅なパフォーマンスの向上になります。

Simultaneous Components アクションでは、任意の数のコンポーネント (またはその他) のアクションを下に挿入できる「Simultaneous Components」ヘッダ行をアクションモデルに配置します。

前の図では、「Simultaneous Components」の下のアクションリストに、3270 コンポーネントへの呼び出し、JDBC コンポーネントへの呼び出し、および Send Mail アクションが含まれています。2 つの Component アクションが別のスレッドに作成され、その後、Send Mail アクションがただちに実行されます (3270 および JDBC コンポーネントが返されたかどうかにかかわらず)。

注記： Simultaneous Components アクションの下にあるリストには、任意のタイプのアクション (Map、Decision など) を含めることができます。しかし、リストのアクションは、いずれも Component アクションからの戻り値には依存しません。これは、ブロックの他のアクションが実行される前に Component アクションによって返されることが保証されていないためです。

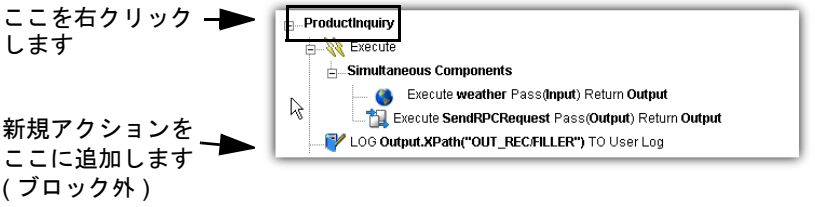
Simultaneous Components ブロックの「外」にあるダウンストリームアクションは、作成されたコンポーネントの戻り値に依存させることができます。これは、Simultaneous Components アクションでは、すべての作成されたコンポーネントによって返されるまで、ダウンストリームアクションにコントロールを渡さないためです。同期は、Simultaneous Components ブロックを超えて実行が継続される前に、必ず行われます。

➢ Simultaneous Components アクションを作成する

1. コンポーネントを開きます。
2. アクションモデルで、Simultaneous Components アクションを配置する行を選択します。
3. [**Action**] メニューから、[**New Action/Advanced**]、[**Simultaneous Components**] の順に選択します。Simultaneous Components ヘッダ行がアクションモデルに表示されます (上の図を参照)。
4. ヘッダ行の下に任意の数のアクションを配置します (ヘッダ行を右クリックし、コンテキストメニューからアクションを選択するか、または Simultaneous Components ブロックにアクションを貼り付けます)。

   注記： Components アクション以外のアクションは、新しいスレッドとして作成されません。

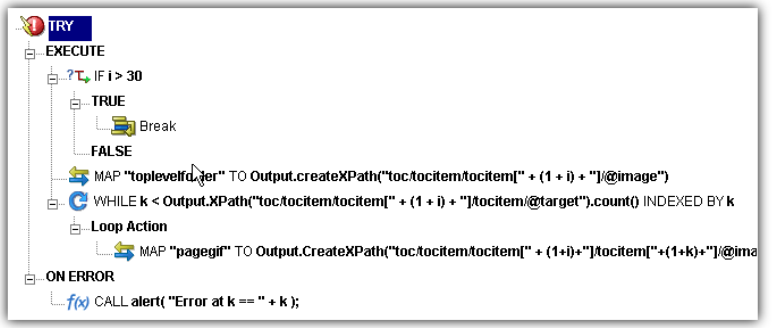
Simultaneous Components ブロック (のダウンストリームの) 外に新しいアクションを配置するには、Simultaneous Components ヘッダ行の上にある行を右クリックして、新しいアクションを選択します。新しいアクションが Simultaneous Components ブロックの下に追加されます (次を参照)。



Transaction アクション

Transaction アクションでは、アクションモデルに begin、commit、または rollback コマンドを挿入し、トランザクションを使用するコンポーネント内でトランザクション境界区分に対して低レベルのコントロールを実行できます。

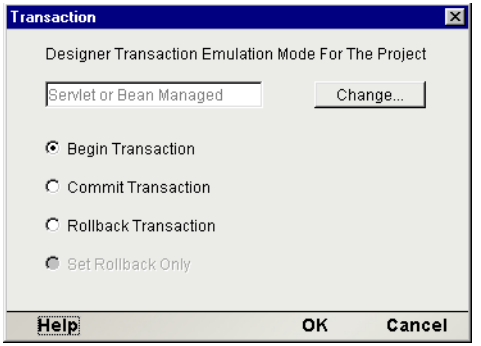
アクションリストに配置するすべての Transaction アクションでは、サービスのアプリケーションメタデータで生成される、適切な対応 Java パススルーを作成します。これがどのようにして発生するかについては、ここでは取り扱いません。Novell exteND アプリケーションサーバについては、『*Novell extend Composer ユーザガイド*』の「Transaction Management」の章を参照してください (別のアプリケーションサーバを使用している場合は、該当する『*exteNd ユーザガイド*』を参照してください)。



注記： Transaction アクションを正しく使用するには、Java トランザクションモデルについて十分に理解している必要があります。exteNd Composer で作成するサービスは、Servlet トリガまたは Enterprise Java Bean (EJB) トリガを使用して配備することができます。配備モードの選択肢には、トランザクション管理に関する重要な意図が含まれています。

➢ Transaction アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルで、Transaction アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action /Advanced**］、［**Transaction**］の順に選択します。［Transaction］ダイアログボックスが表示されます。

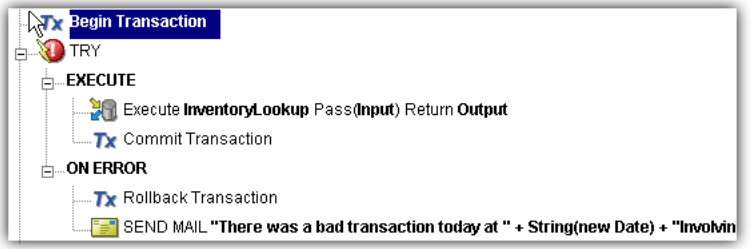


4 使用可能なトランザクションコマンドタイプの 1 つを選択します。

注記： ラジオボタンは、[**Tools**]、[**Configuration**]、[**Advanced**] の順に選択して指定したトランザクションモードにより、有効またはグレー表示されています。たとえば、前の図では、最初の 3 つのラジオボタンは有効になっていますが、[**Set Rollback Only**] ボタンはグレー表示されています。これは、現在のトランザクションエミュレーションモードが［Servlet or Bean Managed］になっているためです。[**Set Rollback Only**] ボタンは、コンテナ管理の EJB 配備のコンテキストでのみ使用可能です。つまり、Servlet/Bean 管理の EJB 配備には適用されません。エミュレーションモードを変更する (および、対応する変更により［Transaction］ダイアログボックスでラジオボタンを有効にする) には、[**Change**] ボタンをクリックします。

5 ［**OK**］をクリックします。

アクションモデルペインに表示される Transaction アクションのペアは、次の図のとおりです。



アクションモデルで Transaction アクションを生成すると、Composer のコンポーネントを実行することにより (または、アニメーション / デバッグセッションの一部としてアクションリストを通して実行することにより) テストできます。適切なエラーメッセージは、アクションモデルでトランザクションコマンドを使用したために存在する可能性があるすべての問題に基づいて表示されます。たとえば、介入する commit コマンドなしにアクションリストで 2 つの begin コマンドを使用した場合、ネストされたトランザクションがサポートされていないという事実に基づいて警告ダイアログボックスが表示されます。

Try/On Error アクション

Try/On Error アクションでは、Try/On Error アクションの Execute 分岐内でエラーが発生すると、アクションのセットを実行します。Try/On Error アクションを使用すると、予測されるエラーをトラップし、他のアクションを実行してエラーを解決したり報告したりできます。たとえば、Try/On Error は、ファイルを検索できなかった XML Interchange アクションに応答するために使用することができます。

Try/On Error アクションを追加すると、アイコンがアクションモデルペインに配置されます。このアイコンには、2 つの分岐である **Execute** と **On Error** があります。アクションによって引き起こされる可能性のあるエラーに気付いた場合は、**Execute** 分岐の下にそれらのアクションを配置します。その後、**On Error** 分岐の下にエラー処理アクションを配置します。エラーが発生した場合、**On Error** 分岐の下のアクションが実行されます。

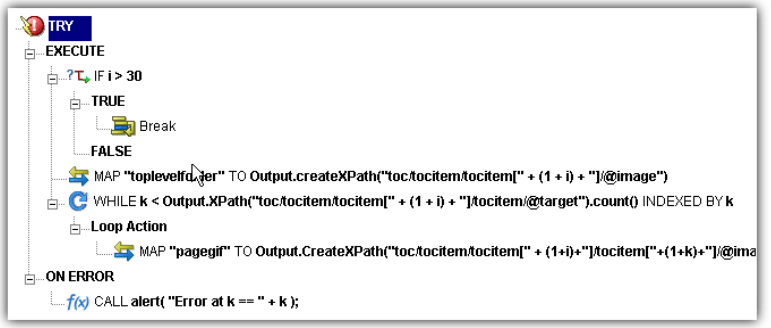
前に挙げた例の後に、XML I/O アクションでエラーが予測される場合は、Execute 分岐の下にアクションを配置します。On Error 分岐の下には、別の場所からファイルを読み込もうとする 2 番目の XML I/O アクションを追加できます。

注記： このアクションに関連付けられているダイアログはありません。

➢ Try/On Error アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルで、Try/On Error アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**Try/On Error**］の順に選択します。**Execute** と **On Error** 分岐のある Try On Error アクションアイコンが、アクションモデルペインに表示されます。
4. **Execute** 分岐の下に、潜在的なエラーを発生させる可能性のあるアクションを追加します。
5. **On Error** 分岐の下に、エラーを解決するアクションを追加します。

アクションモデルの完全な Try/On Error アクションは、次の図のとおりです。



注記： アクションモデル全体で Try/On Error アクションを自由に使用することは、プログラミング上望ましい方法です。

XSLT Transform アクション

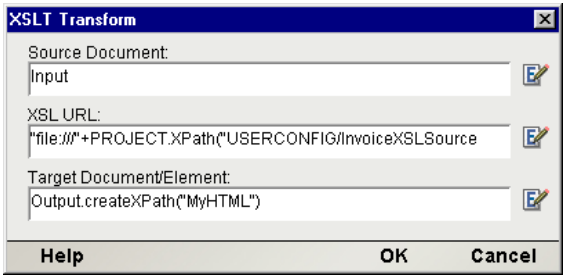
XSLT Transform アクションでは、入力として指定する DOM と XSL スタイルシートを取り出し、コンポーネントの別の DOM に出力を送信します。このプロセスは、サーバ側 XSL 処理とも呼ばれます。

XSL 出力を作成するには、アクションの 3 つのパラメータを指定する必要があります。ソースドキュメント式は、DOM またはドキュメントハンドル (Input など) の名前となる有効な ECMAScript 式です。XSL URL 式は、XSL スタイルシートを指す有効な ECMAScript 式です。このパラメータは、XSL スタイルシートを指定する XSL 処理命令が DOM にすでにある場合にはオプションです。XSL スタイルシートが DOM で指定されない場合、このパラメータを指定しなければなりません。このパラメータを指定し、XSL スタイルシート処理命令が DOM にもある場合は、指定したパラメータによってこの処理命令が上書きされます。

Target Document/Element Expression は、XSL 処理の結果を受信する DOM を指定します。

➢ XSLT Transform アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルで、XSLT Transform アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**XSLT Transform**］の順に選択します。［XSL Process］ダイアログボックスが表示されます。



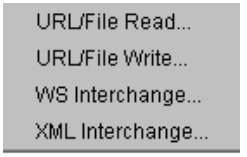
4. レンダリングする「**ソースドキュメント**」の名前を入力するか、または［**Expression Builder**］ボタンをクリックして有効な DOM を解決する ECMAScript 式を作成します。
5. 変換に使用する XSL スタイルシートの名前を［**XSL URL Expr**］フィールドに入力するか、または［**Expression Builder**］ボタンをクリックして有効なスタイルシートを指す ECMAScript 式を作成します。
6. 使用する「**ターゲットドキュメント / 要素**」の名前を入力するか、または［**Expression Builder**］ボタンをクリックして DOM を指定する ECMAScript スクリプト式を作成します。

7 ［**OK**］をクリックします。

アクションモデルの完全な XSLT Transform アクションは、次の図のとおりです。

Data Exchange アクション

このサブメニューには、ファイルの読み書きおよび Web サービスと XML のデータの交換に関するアクションが含まれています。



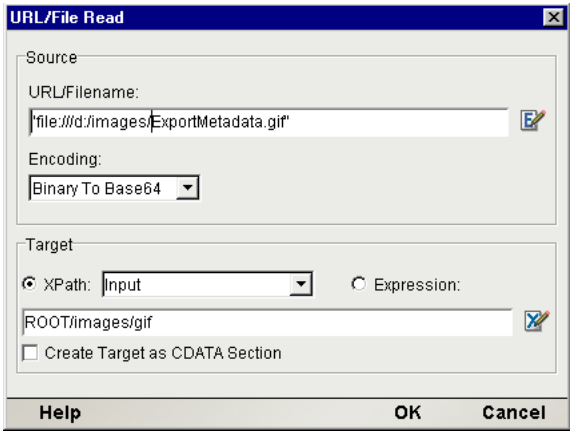
Data Exchange アクション	説明
URL/File Read	XML でないファイル形式を Composer に読み込むことができます。
URL/File Write	XML から別の形式にファイルを書き込むことができます。
WS Interchange	WSDL リソースで定義されたメッセージと操作を使用して Web サービスを実行します。
XML Interchange	外部 XML ドキュメントをコンポーネントの DOM に読み込んだり、コンポーネントの DOM を外部 XML ドキュメントに書き込んだりします。Read/Write メソッドには、File、FTP、HTTP、および HTTPS プロトコルを使用する Get、Put、Post、および Post with Response が含まれます。

URL/File Read

ファイルが XML 以外の形式の場合、このアクションを使用して、ファイルを XPath の場所に読み込みます。

➢ 新規 URL/File Read アクションを作成する

1 ［**Action**］メニューから、［**New Action/Advanced**］、［**URL/File Read**］の順に選択します。次のダイアログボックスが表示されます。



2 画面の［**Source File**］の部分に、ファイルの URL を入力します。これは ECMAScript 式であるため、URL 文字列は「引用符で囲む必要」があります。

3 ファイル形式に該当する場合は、ドロップダウンメニューから［**Encoding**］アルゴリズムを選択します。

注記： 一般的な使用例は、前のとおりです。該当するファイルはバイナリであることがありますが、その場合は、ドロップダウンリストから［Binary to Base64］を選択するのが適切です。正しい「デコード」方法は、URL/File Write アクション (次を参照) で指定することができます。

4 画面の［Target File］の部分で［**XPath**］、［**Input**］を選択し、ファイルのコンテンツの XPath 保存先を入力します。また、ラジオボタンをクリックして、式を選択することもできます。必要に応じて、右側にある式アイコンをダブルクリックし、Expression Builder を呼び出します。

5 ファイルのコンテンツを CDATA セクションでラップする場合には、［**Create Target as CDATA Section** ］チェックボックスをオンにします (これは、前の例で Base64 としてエンコードされるバイナリファイルに対しては必要ありません)。これにより、各括弧 (< >) などの文字は、開始タグまたは終了タグの一部として解釈されることなく、XML ドキュメント内で使用できます。

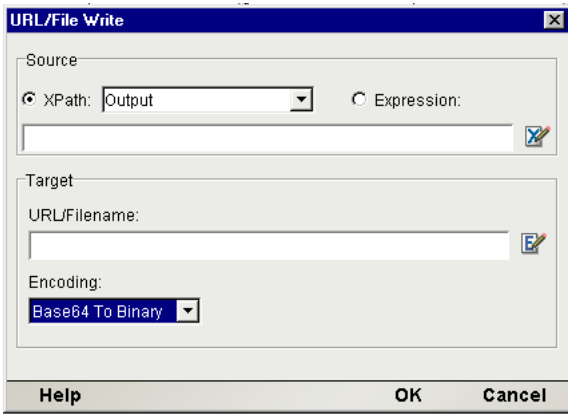
URL/File Write

ファイルが XML 以外の形式でなければならい場合、このアクションを使用して DOM またはメッセージパートからファイルを書き込みます。

注記： このアクションは、すべての点において、前に説明した URL/File Read アクションの機能的な補足です。

➢ 新規 URL/File Write アクションを作成する

1 ［**Action**］メニューから、［**New Action/Advanced**］、［**URL/File Write**］の順に選択します。次のダイアログボックスが表示されます。



2 画面の［Source File］の部分で、［**Source XPath**］、［**Output**］の順に選択します。

3 ファイルのコンテンツを含む XPath を入力します (または、［Expression］ラジオボタンを選択して、ファイルのコンテンツの場所を指定する ECMAScript 式を入力します)。

4 画面の［Target］の部分で、ファイルを保存する URL を入力します。

5 ファイル形式に該当する場合は、［Encoding］リストボックスから選択して、ファイルが書き込まれる前にデコードを指定します。

Web Service (WS) Interchange アクション

Web Service Interchange アクションを使用すると、コンポーネントでは、WSDL リソースで指定された呼び出し規則に従って Web サービスを呼び出すことができます (WSDL リソースの詳細については、236 ページ「WSDL リソースについて」を参照してください)。

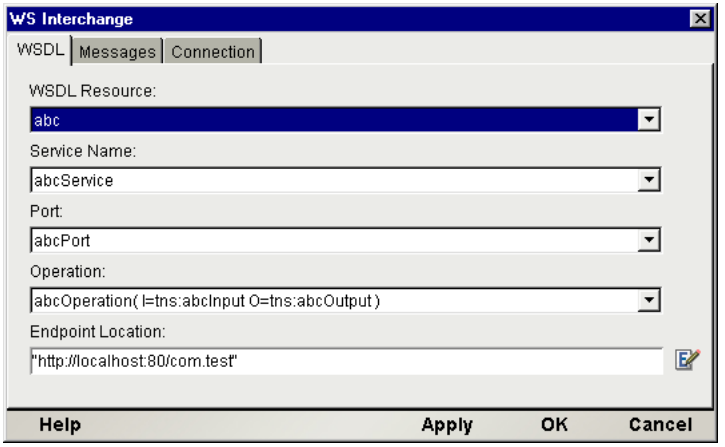
Web Service Interchange アクションを作成するには、サービスを記述する WSDL リソースが必要です。

➢ Web Service (WS) Interchange アクションを作成する

1 コンポーネントを開きます。

2 アクションモデルで、Web Service Interchange アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。

3 ［**Action**］メニューから、［**New Action/Advanced**］、［**Web Service Interchange**］の順に選択します。［Web Service Interchange］ダイアログボックスが表示されます。

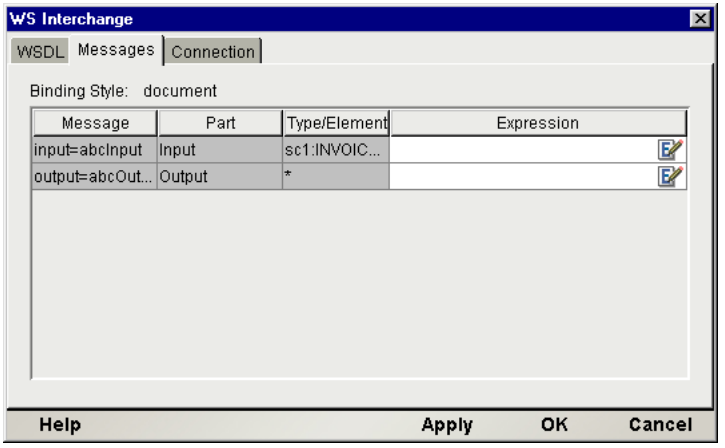


4 表示されるドロップダウンメニューから、希望の「**WSDL リソース**」、「**サービス名**」(存在する場合)、「**バインド**」、および「**操作**」を選択します (これらのメニューには、既存の WSDL リソースの情報から取り出された選択肢が表示されます)。

5 使用する Web サービスの「**エンドポイントの場所**」(通常は、サーブレットを指す URL) を入力し、引用符で囲みます (または、ランタイム時にエンドポイントの場所を評価する ECMAScript 式を入力します)。

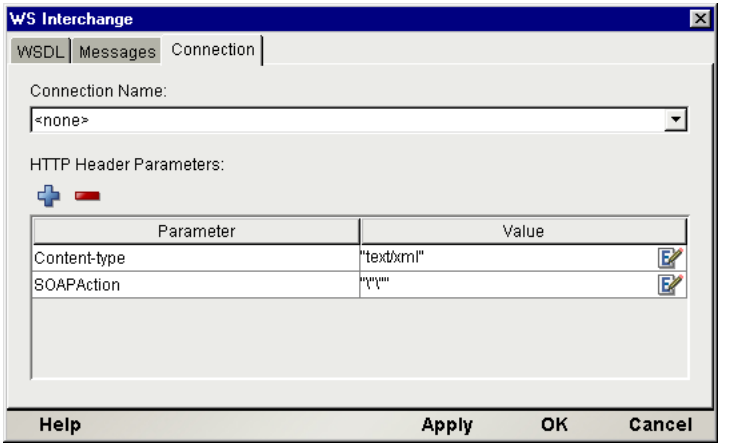
注記： これは、手動で入力しなければならない、ダイアログボックスの［WSDL］タブでの唯一のフィールドです。

6 ［**Message**］タブをクリックして、次のパネルを表示します。



7 呼び出す特定のサービスに対して入力と出力メッセージを指定します。［**Message**］、［**Part**］、および［**Type/Element**］フィールドが入力されます。［**Expression**］に、各メッセージのソースとターゲットを記述する ECMAScript 式を入力します。これは、通常は、入力 DOM または出力 DOM の XPath の場所を指定する式です。適切な式をポイントアンドクリックで簡単に作成できる［Expression Builder］ダイアログボックスの右端にある［Expression Builder］アイコンをクリックします。

8 ［**Connection**］タブをクリックして、次のパネルを表示します。



9 ［**Connection Name**］ドロップダウンメニューから、(必要に応じて) HTTP 接続リソースをクリックします。

注記： 通常の HTTP 接続の場合、ここでは <none> を指定できます。このフィールドの目的は、HTTP 接続リソースに保存されたユーザ ID とパスワード情報を使用して、セキュアサイトに HTTPS を経由して接続することです。

10 ［**Parameter**］と［**Value**］フィールドは、このダイアログボックスの他のタブで指定された操作とメッセージ情報に基づいて、すでに入力されています。［Value］フィールドが空の場合は、SOAP アクションのタイプまたは交換のコンテンツタイプ (MIME タイプ)、あるいはその両方に適切な文字列または式を入力します。

11 追加の HTTP ヘッダ情報を指定する場合は、コンボボックスの上にあるプラス記号をクリックして、新しい HTTP パラメータフィールドを追加します。

12 ［**Apply**］をクリックしてリアルタイムで Web Service アクションをテストするか、または［**OK**］をクリックしてダイアログボックスを閉じます。

XML Interchange アクション

XML Interchange アクションでは、コンポーネントの DOM に外部 XML ドキュメントを読み込み、コンポーネントの DOM から XML ファイルにデータを書き出します。XML Interchange アクションには、次の 4 つのタイプがあります。

- GET
- PUT
- POST
- POST with Response

Get 交換を使用する場合、コンポーネントに挿入する XML ドキュメントを指す URL を指定する必要があります。その後、「Get Document Handle」を指定するか、または XML を受信する DOM を指定しなければなりません。指定した DOM 名が存在しない場合は、自動的に作成されます。また、Get 交換タイプに使用する HTTP 接続リソースを指定することもできます。

Put 交換を使用する場合、XML ドキュメントを書き込む場所を指す URL を指定する必要があります。その後、「Put Document Handle」(つまり、コンポーネントの DOM の名前) を指定するか、または XML としてデータを送信する DOM を指定しなければなりません。また、Get 交換タイプに使用する HTTP 接続リソースを指定することもできます。PUT メソッドは、囲まれたエンティティが指定した Request-URI に保存されることを要求します。

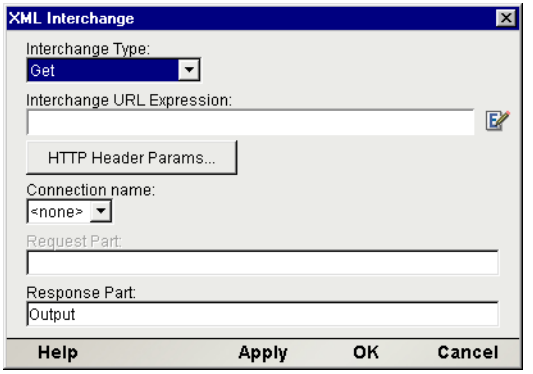
注記： すべての HTTP ハンドラが同様の方法で FTP の PUT プロトコルをサポートするわけではありません。Composer で現在使用しているものが FTP PUT をサポートしていない場合に別の HTTP ハンドラを選択する方法については、「exteNd Composer をお使いになる前に」という章の「システム環境設定」を参照してください。この設定は、［**Tools**］、［**Configuration**］ダイアログボックスの順に選択すると表示されるドロップダウンメニューコントロールを使用して変更できます。

Post 交換を使用する場合、XML ドキュメントを書き込む場所を指す URL を指定する必要があります。その後、「Put Document Handle」(つまり、コンポーネントの DOM の名前) を指定するか、または XML としてデータを送信する DOM を指定しなければなりません。また、Get 交換タイプに使用する HTTP 接続リソースを指定することもできます。POST メソッドは、Request-Line の Request-URI によって識別されるリソースの新しいサブオーディネートとしてリクエスト内で囲まれた要素発信元のサーバが受け付けるよう要求するために使用されます。POST メソッドによって実行される実際の機能はサーバによって決定され、通常は Request-URI に依存しています。

Post with Response 交換を使用する場合、Post と同じパラメータを指定する必要があります。さらに、Post with Response アクションから応答 XML ドキュメントを受信するために DOM も指定しなければなりません。発信元のサーバから応答 XML オブジェクトが返されることを XML Interchange アクションで期待する点以外は、**POST** と同じです。

➢ XML Interchange アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルで、XML Interchange アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**XML Interchange**］の順に選択します。［XML Interchange Action］ダイアログボックスが表示されます。

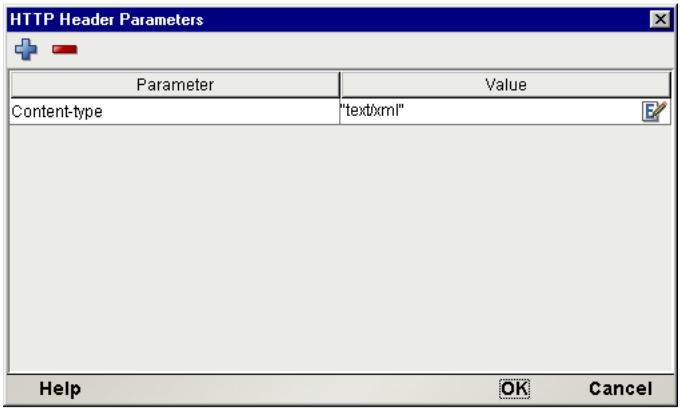


4. 「**交換タイプ**」を選択します。
5. ［**Interchange URL Expression**］フィールドに、次のサポートされているプロトコルのいずれかを使用して、XML ドキュメントの完全修飾 URL を定義する式を入力します。
   - file
   - ftp
   - http
   - https

   選択した交換タイプに応じて、この URL は、XML Interchange アクションの XML ファイルのソースまたは保存先になります。例は次のとおりです。

   file:///g:/xmldata/invoicebatch1.xml

   ftp://accounting:password@123.456.789.987:21/invoices
6. オプションとして、［**HTTP Header Parameters**］ボタンをクリックします。［HTTP Header Parameters］ダイアログボックスが表示されます。



7 プラス (+) アイコンをクリックして、新しいヘッダパラメータを追加します。「**パラメータ**」名と、対応する「**値**」を入力します。共通の HTTP ヘッダパラメータには、「Content-Type」、「Content-Length」、および「Keep-Alive」が含まれます。このダイアログボックスでは、任意の数のパラメータと値のペアを追加できます。

8 ［**OK**］をクリックして、［HTTP Header Parameters］ダイアログボックスを閉じます。［XML Interchange］ダイアログボックスが再表示されます。

9 ［**GET Document Handle**］フィールドに、XML としてコンテンツを送信する DOM ツリーの名前、または XML を受信する DOM ツリーの名前を入力します。

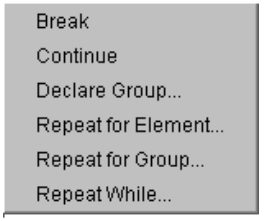
注記： ［*Request Part and Response Part*］に対しては、2 つのドキュメントハンドルを指定します。1 つは XML ドキュメントを送信するもので、もう 1 つは応答を受信するものです (XML ドキュメントの形式で)。

10 「**接続名**」を選択します。接続名は、［Resource］の［Connection Type］で指定したものです。

11 ［**OK**］をクリックします。または、［Apply］ボタンを押して、ダイアログボックスを閉じずに XML Interchange アクションの影響を確認することもできます。これにより、XML Interchange アクションを繰り返し編集して、結果を迅速に確認することが可能になります。

Repeat アクション

このサブメニューには、ルーピングとループ管理構成要素を実装するアクションが含まれています。



Repeat アクション	説明
Break	Repeat for Element、Repeat for Group、または Repeat While ループの実行を停止し、ループ外の次のアクションの実行を続行します。
Continue	Repeat for Element、Repeat for Group、または Repeat While ループの現在のループ反復の実行を停止し、次の反復と同じループの先頭から続行します。
Declare Group	複数回発生する要素に基づいて、グループを作成および名前付けすることができます。グループは、Repeat for Group アクションで使用されます。
Repeat for Element	指定された要素が DOM ツリーで見つかるたびに、1 つまたは複数のアクションを繰り返します。Repeat for Element アクションでは、複数回発生する要素に基づいてループを作成できます。
Repeat for Group	グループの各メンバーに対して 1 つまたは複数のアクションを繰り返します。Repeat For Group アクションでは、データを再作成して、データを集約計算できます。
Repeat While	ループを作成することにより、1 つまたは複数のアクションを繰り返します。While Repeat アクションでは、有効な ECMAScript 式の処理ループに基づくことができます。

Break アクション

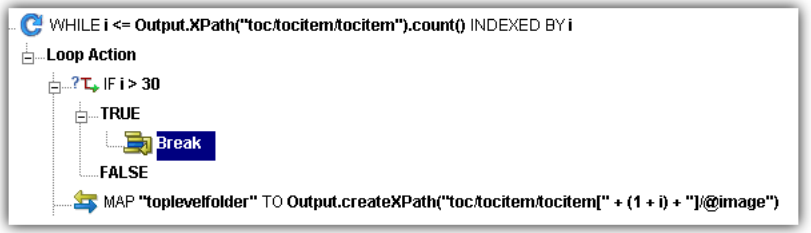
Break アクションを使用すると、Repeat for Element、Repeat for Group、または Repeat While ループの実行が停止します。アクションモデルでは、ループ「外」の次のアクションで実行を続行します。

たとえば、1 つの特定な項目のノードリストを検索するためにループを使用する場合には、Break の使用が適切です。ターゲット項目が見つかると、反復を続行する必要がなくなります。したがって、ループをただちに終了するために Break を使用できます。

注記： Break アクションは、一般的に、Decision アクションの 1 つの分岐 (ループ内) で発生します。Break アクションは、Decision アクションの True または False 分岐のいずれかに、適宜配置します。

➢ Break アクションを追加する

1. Break アクションを含めるために変更する Repeat アクションを含むコンポーネントを開きます。
2. Break アクションを挿入する位置をループ内で選択します。一般的に、これは 1 つの足で、もう一方の足は Decision アクションです (次を参照)。
3. ［**Action**］メニューから、［ **New Action/Advanced**］、［**Break**］の順に選択します。Break アクションが、アクションモデルにただちに表示されます。セットアップダイアログボックスはありません。(次を参照)。



Continue アクション

Continue アクションでは、Repeat for Element、Repeat for Group、または Repeat While ループの現在の反復の実行を停止し、ループの先頭から次の反復の実行を開始します。Continue アクションを使用すると、ループ内でダウンストリームアクションを短絡化する一方、次の反復にループを続行することができます。

Continue アクションは、たとえば、リストの 1 つの項目を何らかの理由でスキップする必要がある場合でも、ループの実行を続行しなければならないような状況において適切です。

注記： Continue アクションは、一般的に、Decision アクションの 1 つの分岐 (ループ内) で発生します。Continue アクションは、Decision アクションの True または False 分岐のいずれかに、適宜配置します。

➢ Continue アクションを追加する

1. Continue アクションを含めるために変更する Continue アクションを含むコンポーネントを開きます。
2. Continue アクションを配置する位置を Loop アクション内で選択します。これは、一般的に、Decision アクションのフォークのどれかの中にあります (次の図を参照)。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**Continue**］の順に選択します。Continue アクションがアクションモデルに表示されます。

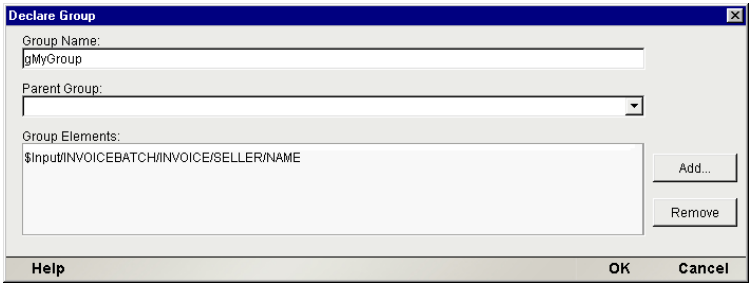
Declare Group アクション

Declare Group アクションでは、それぞれが DOM を参照する 2 つの特殊なリストを作成できます。これらのグループリストは、その後、Repeat for Group アクションのループの基礎として使用できます。リストを作成するには、グループ名を指定し、続けて XPath も指定します。Composer によって、XPath に一致するすべての要素の中で見つかった固有な値それぞれに対して 1 つのエントリを含むグループリストが作成されます。グループリストは、指定したグループ名別に参照されます。次に、グループのメンバーと同じ数のエントリを含むグループリストの固有なエントリそれぞれに対して、詳細リストが作成されます。詳細リストは、ラベル「(Detail)」が後に付けられた、指定したグループ名別に参照されます。

グループ化すると、入力 DOM で繰り返し要素を選択し、その繰り返し要素のすべてのインスタンス (兄弟) 全体で固有な値に基づき、より少数の要素を作成できます。したがって、複数の要素を持つ代わりに、出力 DOM の固有な要素それぞれに対して 1 つの要素を持つことになります。

➢ Declare Group アクションを追加する

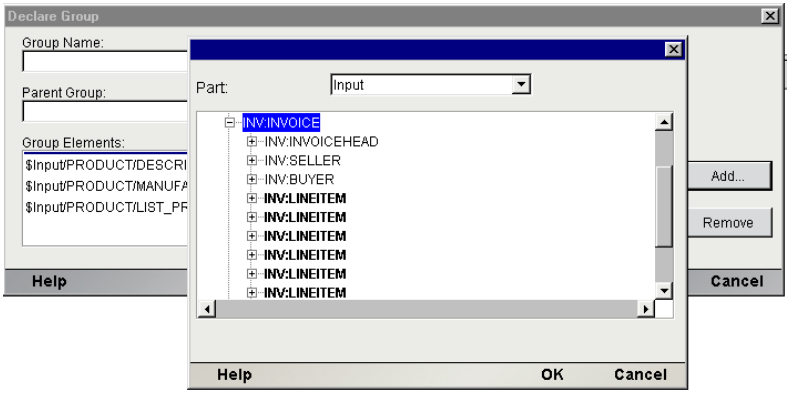
1. コンポーネントを開きます。
2. アクションモデルで、Declare Group アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**Declare Group**］の順に選択します。［Declare Group］ダイアログボックスが表示されます。



4 グループの名前を入力します。

5 オプションとして、ペアレントグループを選択します。これは、複数のグループレベルを作成する場合に使用します。

6 [**Add**]をクリックします。[Add Element]ダイアログボックスが表示されます。



7 ドキュメントハンドル (DOM) と要素を選択します。

8 [**OK**]をクリックします。

9 手順 6 から 8 を繰り返して、グループに複数の要素を追加します。

10 [**Remove**]をクリックして、グループから要素を削除します。

11 グループに希望の要素がすべて含まれている場合は、[**OK**]をクリックします。

注記： この例は、コンピュータにインストールされたアクション例サンプルプロジェクトで見つけることができます。

Repeat For Element アクション

Repeat アクションでは、アクションモデル内にループ構造を作成します。ループを作成すると、1 つまたは複数のアクションのセットを繰り返すことができます。ループには、Repeat For Element、Repeat For Group、および Repeat While の 3 つのタイプがあります。

XML では、ドキュメントの要素の複数のインスタンス (データベーステーブルの複数のレコードに類似) を使用できます。インスタンスの数は、ドキュメントごとに異なる場合があり、ドキュメントスキーマ (DTD または XML スキーマ) で定義されます。たとえば、請求書を含む XML ドキュメントを毎日受け取らなければならない場合があります。XML ドキュメントに含まれる請求書の数は、毎日異なります。XML ドキュメントに含まれる請求書のインスタンス数を把握していない場合、入力 XML ドキュメントの各請求書から出力 XML ドキュメントに請求書数を転送しようとすると、問題が生じます。しかし、Repeat for Element アクションを使用すれば、この問題が解決します。

Repeat for Element アクションでは、複数回発生する要素をマークできます。その後、マークされた要素の各インスタンスに対してそのようなインスタンスがなくなるまで 1 つまたは複数のアクションを実行する処理ループを設定します。前の例では、請求書番号を転送する Map アクションが処理ループに含まれており、このアクションは、すべての請求書番号がマップされるまで、受信する請求書の数に関係なく繰り返されます。

Repeat for Element アクションでも別名の概念を使用します。別名は、2 つの機能を実行します。これは、マークされた繰り返し要素の代替名または省略形で、長い XPath 式を再指定する必要がなくなります。場合によっては、繰り返し要素は、ドキュメント階層の数段階下になります。マークされた要素のチャイルド要素を転送する Repeat ループに Map アクションを作成する場合、別名を使用すると、長い XPath 式を入力し直すよりも早く処理できます。別名は、ループが処理されるたびに繰り返し要素の次のインスタンスを使用するための、Repeat ループ内の Map アクションに対するインジケータでもあります。別名を使用しない Repeat for Element ループ内の Map アクションでは、ソース DOM の要素の最初のインスタンスを常に参照します。

注記： ［Map］ダイアログボックスの繰り返し別名の上にマウスを移動すると、別名によって表される XPath を示すツールヒントが表示されます。

Repeat for Element アクションを使用すると、ループ内の複数のアクションを処理できます。最も単純な場合、Repeat ループには、入力 DOM から出力 DOM に現在の要素インスタンスの値を転送する Map アクションが 1 つだけ含まれます。処理ループでは、複数のアクション (現在の値を転送する Map アクションと、ファイルに書き込んで各転送の監査を作成する Log アクション) を定義することもできます。

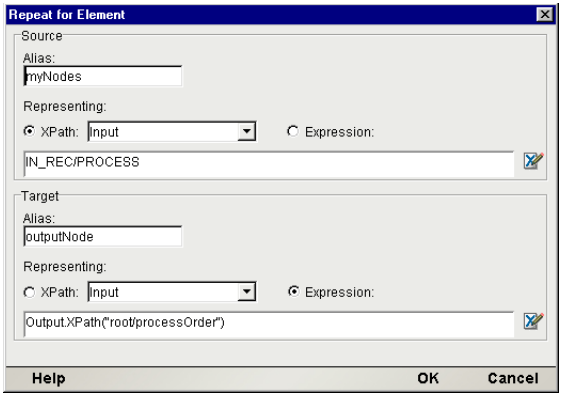
➢ Repeat For Element 処理ループを使用する

1. Repeat For Element アクションを作成します。
2. Repeat For Element 処理ループの Map アクションを作成します。

➢ Repeat For Element アクションを追加する

1. DOM ツリーで繰り返し要素の最初のインスタンスを選択します。

2 ［**Action**］メニューから、［**New Action/Advanced**］、［**Repeat for Element**］の順に選択します。［Repeat for Element］ダイアログボックスが表示されます。



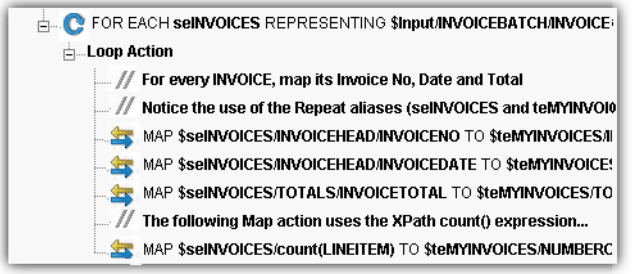
3 ［**Alias**］フィールドに入力します。別名の命名規則は、ソースまたはターゲットを示すプリフィックスが付けられた要素名と、「sreELEMENTNAME」などの Repeat アクションのタイプを使用することです。

4 XPath 式を入力するか、または［**Expression Builder**］ボタンをクリックして繰り返し要素の XPath 式を作成します。

5 ［**Target**］に対して手順 3 と 4 を繰り返し、マップされた要素を配置する場所を識別します。

6 ［**OK**］をクリックします。

アクションモデルの完全な Repeat For Element アクションは、次の図のとおりです。



Repeat for Group アクション

受信した XML ドキュメントの形式は、常にビジネスプロセスの条件を満たす形式であるとは限りません。たとえば、XML ドキュメントに異なる販売者からの請求書が含まれる場合があります。データは個々の請求書として受信されますが、B2B トランザクションのコンテキストでは、データを要約してマネージャにサマリデータを送信すると同時に、会計支払部署に請求書データを送信しなければならないことがあります。

Repeat For Group アクションでは、データを再作成したり、データを集約計算するためのフレームワークを確立したり、あるいはその両方を実行することができます。グループ化すると、入力 DOM で繰り返し要素を選択し、その繰り返し要素のすべてのインスタンス (兄弟) 全体で固有な値に基づき、より少数の要素を作成できます。これにより、請求書全体 (一部の請求書では販売者値が同じ) で複数の販売者要素を使用する代わりに、出力 DOM の固有な販売者値それぞれに対して 1 つの要素を使用することになります。

Repeat For Group アクションでは、Declare Group アクションにより作成される 2 つのリストのうちの 1 つに基づいた処理ループをセットアップします。ループは、使用するリストのエントリの数だけ実行されます (Group リストまたは Group (Detail) リストのいずれか)。前の例では、Group リストを使用し、販売者あたりに 1 つの要素を設定すると、処理ループに Map アクションを追加して、各販売者における請求書の数を計算できます。また、各販売者の下に個々の請求書数をリストすることもできます。Repeat For Group を Map コマンドと結合させると、元の XML ドキュメントとは構造とデータが異なる新しい XML ドキュメントを作成できます。

Repeat for Element アクションと同様に、Repeat for Group アクションでも別名の概念を使用します。［Repeat for Group］ダイアログボックスで使用されるソースグループの値は、Declare Group アクションによって作成されたリスト名です。リスト名は、2 つの機能を果たします。これらは、ループ内の任意の Map アクションの XPath ソースに対する別名と省略形です。これによって、長い XPath 式を再指定する必要がなくなります。Map アクションソースの DOM 名の代わりに使用されるグループリスト名は、ループによって処理が行われるたびにグループリストの次のインスタンスを使用するための、Repeat ループ内の Map アクションに対するインジケータでもあります。グループ名を使用しない Repeat for Group ループ内の Map アクションでは、ソース DOM の要素の最初のインスタンスを常に参照します。

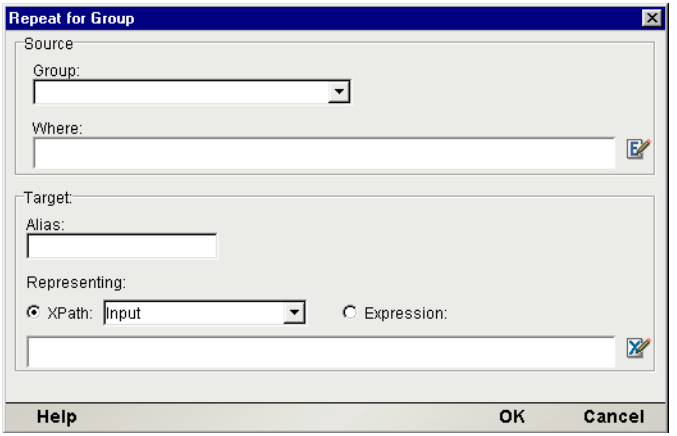
Repeat for Group アクションで作成されたターゲット別名も、2 つの機能を果たします。これらは、ループ内の任意の Map アクションの XPath ターゲットに対する別名と省略形です。これによって、長い XPath 式を再指定する必要がなくなります。DOM 名の代わりに使用されるターゲットの別名は、ターゲット DOM のソースの新しいインスタンスを作成するための、Repeat ループ内での Map アクションに対するインジケータでもあります。ターゲットの別名を使用しない Repeat for Group ループ内の Map アクションでは、ターゲット DOM で作成された最初のインスタンスをソースグループリストからの後続のインスタンスで常に上書きします。

Repeat For Group アクションを作成するには、次の 3 つのタスクを完了する必要があります。

- Declare Group アクションを作成して、グループリストを作成します。
- 使用するグループリストを指定する Repeat for Group アクションを作成します。
- ループ内に Map アクションを作成します。

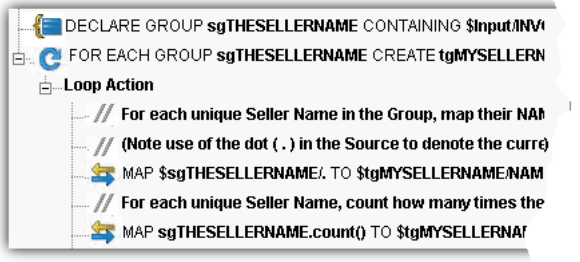
➢ Repeat For Group アクションを追加する

1. コンポーネントを開きます。
2. ［**Action**］メニューから、［**New Action/Advanced**］、［**Repeat for Group**］の順に選択します。［Repeat for Group］ダイアログボックスが表示されます。



3. Repeat for Group アクションループに基づくグループ名を選択します。
4. オプションとして、**Where** 句を入力してグループリストをフィルタするか、または［**Expression Builder**］ボタンをクリックして Where 式を作成します。
5. オプションとして、ターゲット式の Map アクションによって使用される別名を作成します。
6. 別名によって表わされる XPath を作成します。
7. ［**OK**］をクリックします。

アクションモデルの完全な Repeat For Group アクションは、次の図のとおりです。



Repeat While アクション

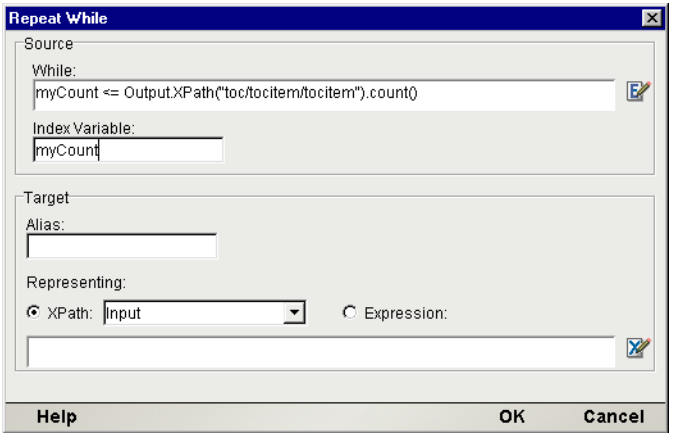
Repeat While アクションでは、指定した条件が真である限り、1 つまたは複数のアクションを繰り返します。たとえば、請求書内の品目からの総売上高を含む変数を作成できます。その後、請求書を読み込み、品目の合計を計算し、品目の合計が特定の金額に達したときに停止する Repeat While アクションを作成できます。

Repeat While アクションで作成されたターゲットの別名は、2 つの機能を果たします。これは、ループ内の任意の Map アクションの XPath ターゲットに対する別名または省略形です。これによって、長い XPath 式を再指定する必要がなくなります。Map アクションの DOM 名の代わりに使用されるターゲットの別名は、ターゲット DOM のソースの新しいインスタンスを作成するための、Repeat ループ内での Map アクションに対するインジケータでもあります。ターゲットの別名を使用しない Repeat for Group ループ内の Map アクションでは、ターゲット DOM で作成された最初のインスタンスをソースからの後続のインスタンスで常に上書きします。

注記： Repeat for Element や Repeat for Group と異なり、Repeat While は、DOM ツリーのデータに基づく必要はありません。ループは、DOM ツリーのデータとは独立して操作できます。

➢ Repeat While アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルで、**Repeat While** アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. ［**Action**］メニューから、［**New Action/Advanced**］、［**Repeat While**］の順に選択します。［Repeat While］ダイアログボックスが表示されます。



4 While ループをテストするための式を入力するか、または［**Expression Builder**］ボタンをクリックして式を作成します。

5 ループの条件をトラッキングする変数の名前を入力します。

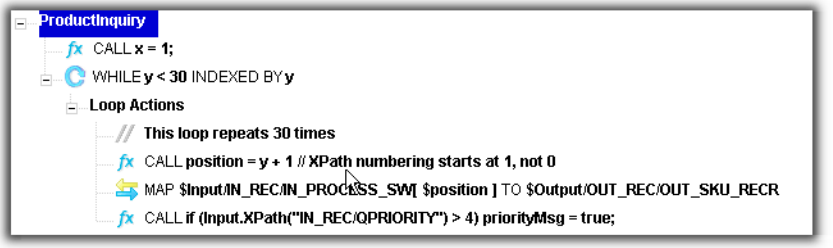
6 「**ターゲット**」要素の別名がわかっている場合は、［**Alias**］フィールドに入力します。

7 別名がわからない場合は、XPath と DOM 要素、または式のいずれかを選択し、有効な式を入力します。

8 条件ステートメントを入力するか、または［**Expression Builder**］ボタンをクリックして式を作成します。

9 ［**OK**］をクリックします。

アクションモデルペインの完全な Repeat While アクションは、次の図のとおりです。



9 リソース

リソースとは、コンポーネントがタスクを実行するために必要となる可能性のある、再使用可能な xObject です。たとえば、ほとんどの XML 統合アプリケーションは何らかの「バックエンド」システムと通信し、このためには、通常ポートの仕様、ドライバの場所、ユーザ ID およびパスワードなどに関わる「接続」を確立することが必要となります。この種の情報は、再使用可能なオブジェクトに保存でき、ランタイム時にコンポーネントによってアクセスされます。これは、リソース xObject によって実現されます。

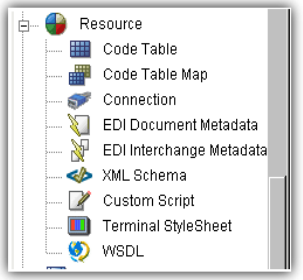
Composer で使用可能なコアリソースのタイプは、次のとおりです。

- コードテーブル
- コードテーブルマップ
- 接続
- カスタムスクリプト
- WSDL
- XML スキーマ

これらのコアタイプに加え、各種の Composer 製品では、コネクタに固有となる、追加のリソースタイプを使用しています。たとえば、EDI コネクタを使用すると、EDI ドキュメントメタデータおよび EDI 交換メタデータのリソースを指定できます。

注記： コネクタに固有のリソースタイプを作成する方法については、コネクタのドキュメントで説明しています。次の節では、Composer コアリソースのタイプのみについて説明します。

リソースタイプには、それぞれ独自のアイコンがあり、Composer のメインナビゲーションフレームのカテゴリペインに表示されています。リソースのカテゴリおよび関連アイコンは、次のとおりです。



注記：端末スタイルシートなどの一部のリソースタイプは、特定のEnterprise Connect製品専門です。

リソースの操作

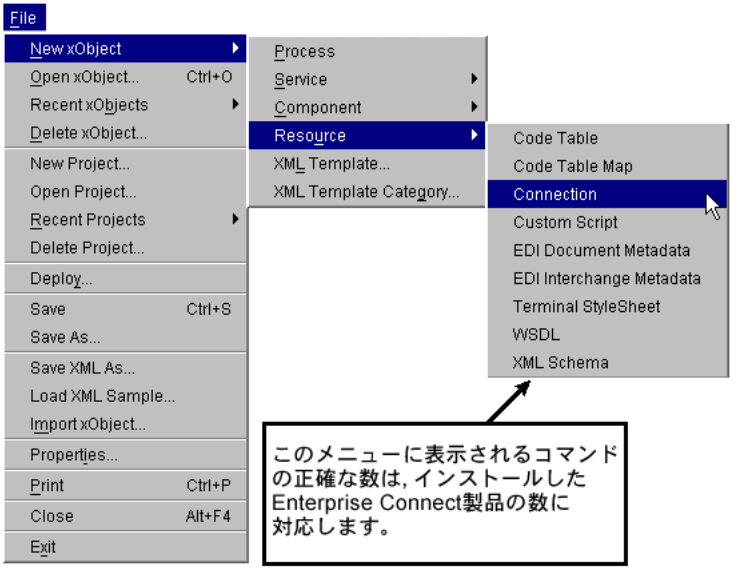
あるタイプでカスタム作成されたリソースは、カテゴリペインで特定のリソースカテゴリ (コードテーブル、接続など) を選択 (ハイライト) するとインスタンスペインに表示されます。各リソースインスタンスは、現在のプロジェクトでさまざまなコンポーネントまたはサービス、あるいはその両方で再使用でき、他のプロジェクトにもインポートできます。

コンポーネントの作成時には、コンポーネントのウィザードにより、コンポーネントで使用するリソースの名前を入力するよう求めるプロンプトが表示されます。つまり、「最初に」そのリソースを作成して、コンポーネントでそのリソースが使用できるようにする必要があります。

すべてのリソースは、同じ基本的な手順を使用して作成されます (次を参照)。

➢ プロジェクトに新しいリソースを追加する

1 Composer の［**File**］メニューから、［**New xObject**］、［**Resource**］、希望のリソースカテゴリの順に選択します。次の図を参照してください。



2 ウィザードが開き、カスタムリソースの名前、および作成しているリソースのタイプに関する他の情報を求めるプロンプトが表示されます。ウィザードが要求した情報を入力します。

3 ウィザードの最後の画面で、［**OK**］をクリックします。該当するリソースカテゴリに新しいリソースが追加され、インスタンスペインにリソースの名前が表示されます。

コードテーブルについて

Composer アプリケーションを作成する際には、受け取ったデータを繰り返し変換するための要件に直面することが多々あります。この変換タイプの代表的な例には、分類目的で、州コード (例 : アラバマ、イリノイ) を地域に変更したり、これらがシステム間で移動するごとにコードを説明することが含まれます。Composer では、このタイプの変換を実行できるようにする機能が提供されています。たとえば、書店では「コード 1」で小説のカテゴリを表したり、デパートでは「コード M」を使用して、男性用衣類を表す場合があります。

他の説明は含まずに、出力 XML に「コード 1」または「コード M」のみが含まれるようなアプリケーションを設計する場合、あいまいで紛らわしい結果になる可能性があります。このような場合に、コードテーブルが役立ちます。コードテーブルには、一般的に使用されるビジネスコードテーブルが保存され、コードテーブルマップと同時に動作し、出力を受け取る担当者やビジネスプロセスにとってより意味のある出力 XML ドキュメントが生成されます。書店の場合、「コード 1」を含む入力 XML は、コードテーブルを使用してマップされ、カテゴリが「小説」である出力 XML が生成される場合があります。

コードテーブルエディタについて

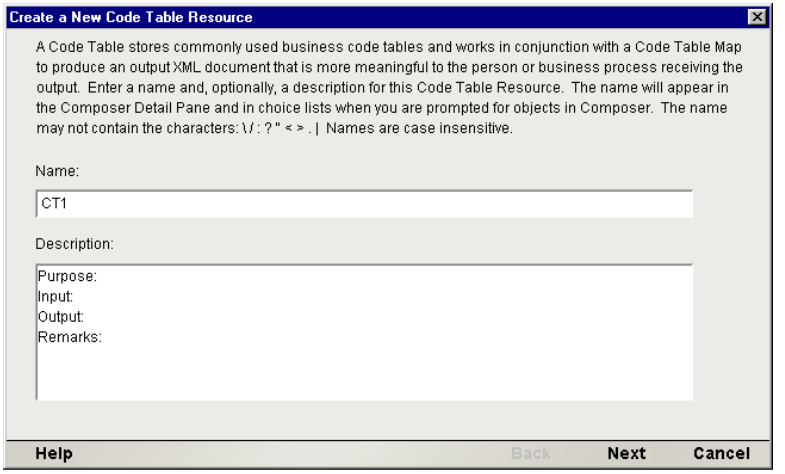
コードテーブルエディタには、メニューオプションとツールバーの両方があります。メニューオプションに加え、コードテーブルエディタには、次のボタンから構成されたツールバーが含まれます。

表 9-1

ボタン	説明
	[Save] - このボタンをクリックすると、開いているコードテーブルに変更が保存されます。
	[Cut] - このボタンをクリックすると、選択したデータがコードテーブルエディタから削除され、Windows のクリップボードにコピーされます。
	[Copy] - このボタンをクリックすると、選択したデータが Windows のクリップボードにコピーされます。
	[Paste] - このボタンをクリックすると、Windows のクリップボードのコンテンツがカーソル位置に貼り付けられたり、選択したテキストが置換されたりします。
	[Delete] - このボタンをクリックすると、データがコードテーブルエディタの現在アクティブな (または選択した) セルから削除されます。
	[Add Row] - このボタンをクリックすると、コードテーブルエディタに新しい空白の行が追加されます。
	[Delete Row] - このボタンをクリックすると、現在アクティブな (または選択した) 行がコードテーブルエディタから削除されます。

➢ コードテーブルを作成する

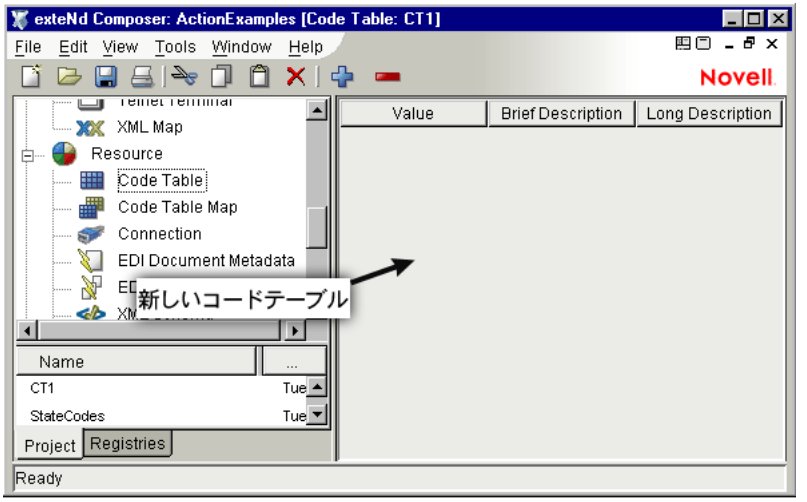
1 ［**File**］>［**New xObject**］>［**Resource**］>［**Code Table**］の順に選択します。Create a New Code Table xObject ウィザードが開きます。



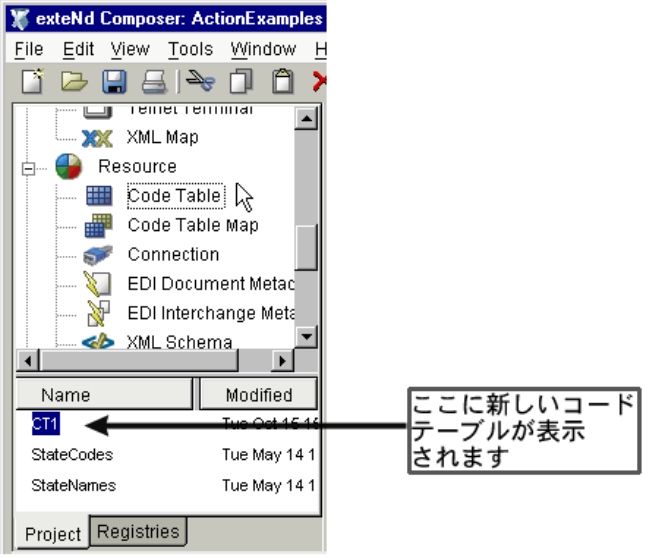
2 ［**Name**］に名前を入力します。

3 オプションとして、［**Description**］に説明情報を入力します。

4 ［**Next**］をクリックします。タイトルバーに空白のコードテーブルの名前が表示された、コードテーブルエディタが表示されます。

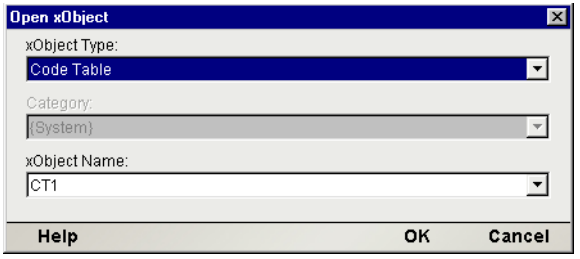


コードテーブルエディタを閉じると、新しいコードテーブルの名前が、Composer ウィンドウのリソースのカテゴリで、［Code Table］の下に表示されます (次の図を参照)。



➢ コードテーブルを開く

1 ［**File**］>［**Open xObject**］の順に選択します。［Open xObject］ダイアログボックスが表示されます。



2 ［**xObject Type**］ドロップダウンリストから、［**Code Table**］を選択します。

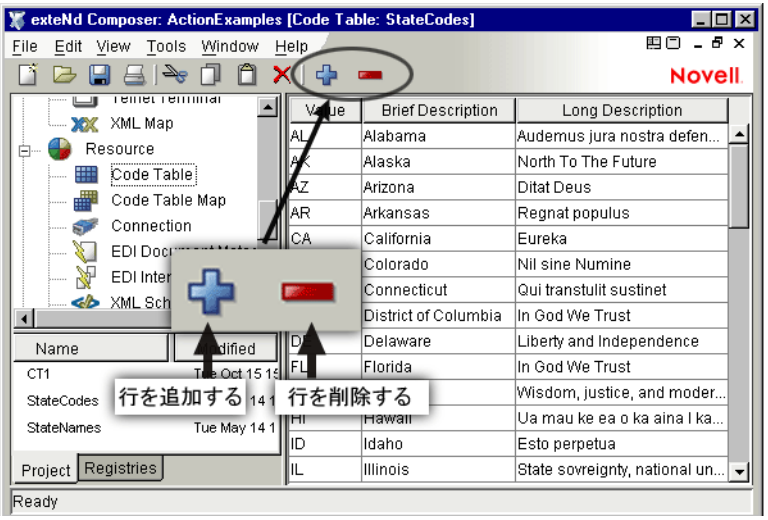
3 ［**xObject**］ドロップダウンリストから、開くコードテーブルを選択します。

4 ［**OK**］をクリックします。コードテーブルエディタで選択したコードテーブルが開きます。

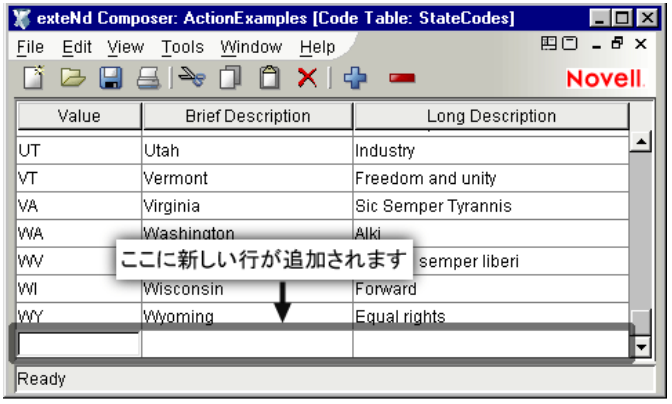
注記： オプションとして、Composer ウィンドウのカテゴリペインでコードテーブルを選択して、詳細ペインからコードテーブルをダブルクリックすることもできます。

➢ コードテーブルにデータを追加する

1 データを追加するコードテーブルを開きます。開いたコードテーブルがコードテーブルエディタに表示されます。



2 ［**Add Row**］ボタンをクリックします。コードテーブルエディタウィンドウに空白の行が表示されます。



3 データを追加するセルをクリックします。

4 新しいデータを次のように入力します。

- ［**Value**］フィールドには、使用している XML サンプルから要素データを入力します。
- ［**Brief Description**］フィールドには、簡単な説明を入力します。
- ［**Long Description**］フィールドには、完全な説明を入力します。

5 手順 3 と 4 を繰り返して、すべてのデータを追加します。

6 ［**File**］>［**Save**］の順に選択するか、［**Save**］ボタンをクリックします。

注記： データをより迅速に入力する別の方法は、スプレッドシートからデータをコピーして、コードテーブルに貼り付けることです。その場合、3 列が選択されているか確認してください。最初の列には、データが含まれていなければならず、2 番目および 3 番目の例はオプションです。スプレッドシートを開き、この 3 列と必要な数の行をコピーします。コードテーブルを開き、ただちに［Paste］ボタンを押します。同じ方法で、Microsoft Word® ドキュメントの表からデータをコピーすることもできます。

➢ コードテーブルを編集する

1. 編集するコードテーブルを開きます。
2. 編集するデータをハイライトします。
3. ［**Edit**］メニューまたはコードテーブルエディタのツールバーボタンを使用して、選択したデータを切り取り、貼り付け、またはコピーできます。
4. 編集が完了した後、［**Save**］ボタンをクリックします。

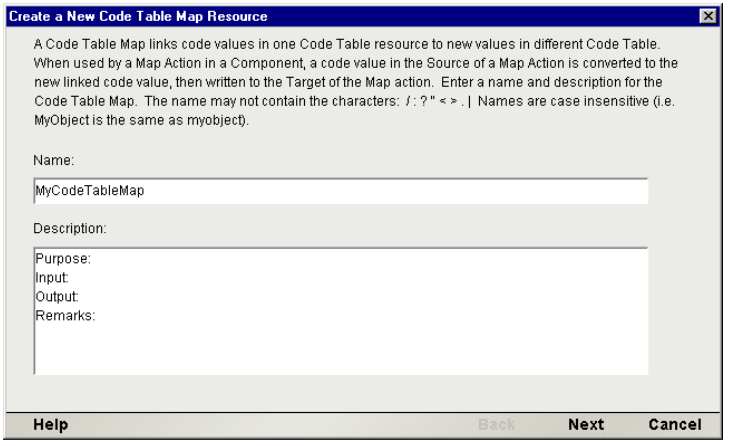
コードテーブルマップについて

「コードテーブルマップ」とは、あるコードのセットを別のコードのセットに自動的に変換するために使用するリソースです。これらのマップは、コンポーネント内の XML サンプル間でデータを変換および交換する際に便利です (ある企業では数値コードを使用してステータスフィールドを保存しており、別の企業ではアルファベットコードを使用してステータスフィールドを保存している場合など)。

注記： コードテーブルは、同じコードテーブルにマップできないため、コードテーブルマップを作成する前には、2 つの個別のコードテーブルを作成する必要があります (209 ページ「コードテーブルについて」を参照)。

➢ コードテーブルマップを作成する

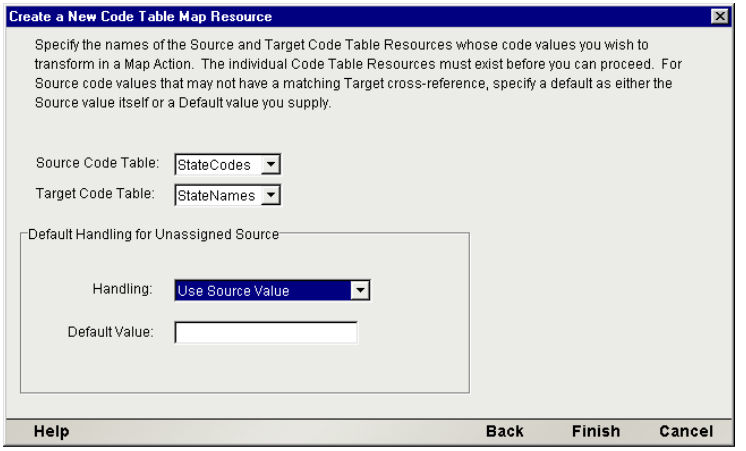
1. ［**File**］>［**New xObject**］>［**Resource**］>［**Code Table Map**］の順に選択します。Create a New Code Table Map xObject ウィザードが開きます。



2 ［**Name**］に名前を入力します。

3 オプションとして、［**Description**］に説明情報を入力します。

4 ［**Next**］をクリックします。Create a New Code Table Map xObject ウィザードの 2 番目のページが表示されます。



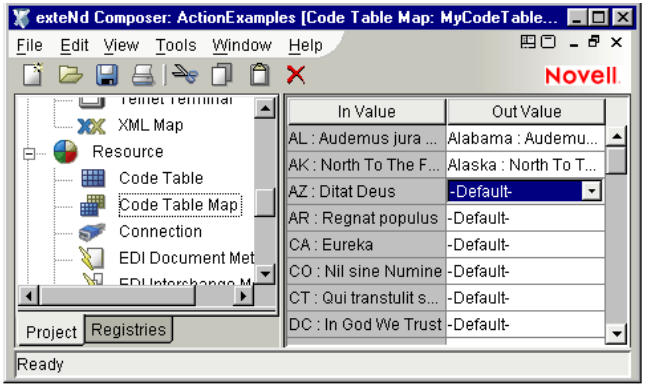
5 ［**Input Code Table**］を選択します (これらのコードは、コンポーネントに受信されるとおりのデータコンテンツを表します)。

6 ［**Output Code Table**］で出力コードテーブルを選択します (これらのコードは、希望のコード値を表します)。

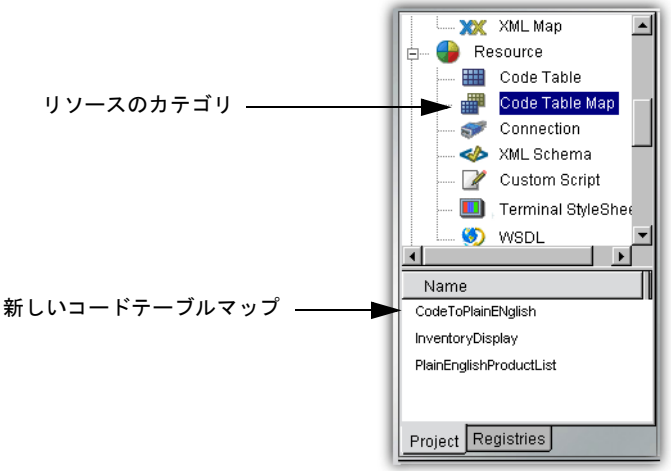
7 ［**Handling**］で、処理方法を選択します。この機能では、マップしている出力コードテーブルに対応する値を持たない入力コードテーブルからの値を処理する方法を Composer に命令できます。たとえば、コードテーブル 1 には値が 6 つあり、コードテーブル 2 には値が 5 つしかない場合、追加の値を処理する方法を、Composer に命令する必要があります。次の 2 つの選択肢があります。

- ［**Use Source Value**］— この選択肢では、単純に出力値として入力値を使用します。たとえば、「Warehouse1」の入力は、単純に「Warehouse1」の出力値にマップされます。
- ［**Use Default Value**］— この選択肢では、デフォルトで［**Default Value**］フィールドで設定した値を使用します。たとえば、［**Default Value**］フィールドに「Not Applicable」と入力できます。

8 ［**Finish**］をクリックします。新規に作成したコードテーブルマップが表示されます。



コードテーブルマップエディタを閉じると、新規に作成したコードテーブルマップが、インスタンスペインのリソースのカテゴリで、［Code Table Map］の下に表示されます (次の図を参照)。

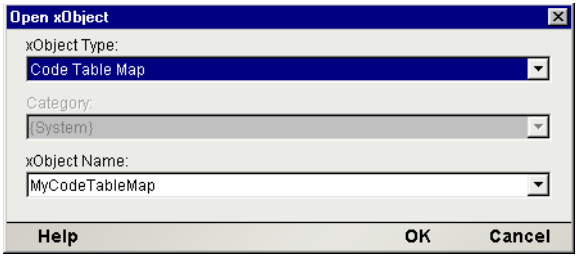


コードテーブルのマップ

入力コードテーブルおよび出力コードテーブルを選択した後、値をマップする必要があります。コードテーブルマップには、最初に［**Out Value**］フィールドでデフォルト設定にマップされた「**In 値**」が表示されます。In 値は編集できないため、グレー表示になっています。デフォルトのフィールドをクリックすると、ドロップダウンリストを使用して、In 値を［Out Value］フィールドのいずれかの値にマップできます。これにより、1 つ以上の In 値を同じ Out 値にマップできます。

➢ コードテーブルマップを開く

1 ［**File**］>［**Open xObject**］の順に選択します。［Open xObject］ダイアログボックスが表示されます。



2 ［**xObject Type**］ドロップダウンリストから、［**Code Table Map**］を選択します。

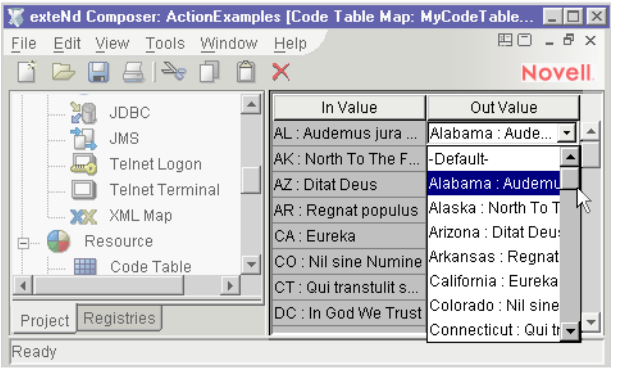
3 ［**xObject**］ドロップダウンリストから、開くコードテーブルマップを選択します。

4 ［**OK**］をクリックします。選択したコードテーブルマップが開きます。

➢ コードテーブルマップで値をマップする

1 値をマップするコードテーブルマップを開きます。

2 最初のレコードで、［**Out Value**］フィールドをクリックします。出力コードテーブルから使用できる値がすべて表示されたドロップダウンリストが表示されます。



3 ドロップダウンリストから希望の値を選択します。

4 すべてのレコードに対して、手順 2 と 3 を繰り返します。

5 ［**File**］>［**Save**］の順に選択するか、ツールバーで［**Save**］ボタンをクリックします。

➢ コードテーブルマップを編集する

1. 編集するコードテーブルマップを開きます (上の「コードテーブルマップを開く」を参照)。
2. 編集する［**Out Value**］セル内をクリックします。
3. ドロップダウンリストから新しい値を選択します。
4. ［**File**］、［**Save**］の順に選択するか、ツールバーで［**Save**］ボタンをクリックします。

コードテーブルマップの使用

コードテーブルマップを作成し、コンポーネントの作成時にこれを使用します。たとえば、XML Map コンポーネントエディタでは、入力 DOM からの要素を、コードテーブルマップを使用して出力 DOM にマップするアクションを作成できます。アクションは、次のようになります。

Map アクションでコードテーブルマップを使用すると、入力データを転送するだけでなく、出力に配置する前に変換することができます。

詳細については、第 7 章「基本的なアクション」を参照してください。

接続について

「接続」とは、JDBC データソースとの通信、または HTTP 認証を使用するセキュリティ保護されたサーバとの通信を確立するために使用するリソースです。Composer Enterprise Connect 製品をインストールしている場合 (JMS 用 Connect、3270、5250 など)、各 Connect では、それぞれ関連するデータソースまたはトランスポート層に適切となる、独自のタイプの専用データ接続が使用されます。接続リソースには、ドライバ、タイムアウト、ユーザ認証、トランスポートプロトコル、または特定のデータストリームタイプと接続を設定するために必要となる可能性のあるエンドポイント仕様、あるいはこれらすべてに関する重要な情報が格納されます。

接続リソースは、さまざまなデータソース (データベース接続など) だけではなく、Composer のコアアクションの 1 つである XML Interchange アクションに対しても必要となります。XML Interchange アクションでは、HTTP または HTTPS を使用して XML ドキュメントを交換できます。HTTP 接続リソースには、HTTPS セッションの設定に必要なユーザ認証およびセキュリティの情報が格納されます。

ユーザにより提供された接続リソースの情報の一部は、ECMAScript 式を使用してランタイム時に動的にバインドできます (次の説明を参照)。接続リソースのすべてのユーザ情報が、静的である必要はありません。

接続リソースでは、該当するデータストリームまたはエンドポイントに対する詳細なアクセス情報が指定されるため、一般的に、コンポーネントまたはサービスで使用するすべてのデータソースに対して、1 つの接続リソースを作成する必要があります。たとえば、アプリケーションで異なる 3 つのデータベースと通信する必要がある場合、おそらく異なるサーバ上でそれぞれ異なるデータベースにアクセスするような、3 つの JDBC 接続リソースを作成しなければならない場合があります (ただし、接続リソースは、複数のコンポーネントによって再使用できる実際の「リソース」です)。

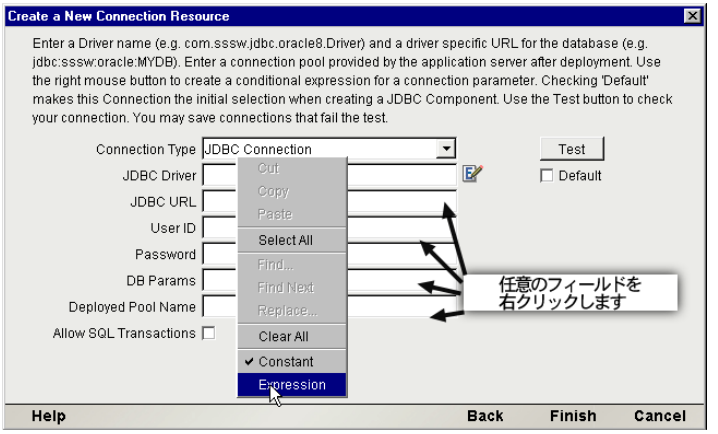
定数駆動型および式駆動型の接続について

接続パラメータの値は、定数または式としての方法のうちいずれかを使用して指定できます。定数ベースのパラメータでは、接続が使用されるたびに［Connection］ダイアログボックスに入力した値を使用します。式ベースのパラメータでは、ランタイム時に接続が使用されるたびに異なる値となりえる、プログラム的な式を使用して値を設定できます。この場合、接続の動作は柔軟になり、接続の使用ごとにランタイム時の条件に対応できるようになります。

たとえば、HTTP 接続における式駆動型のパラメータの非常に単純な使用の 1 つは、ユーザ ID とパスワードを PROJECT 変数 (例 : PROJECT.XPATH("USERCONFIG/MyDeployUser")) として定義することです。このようにすると、プロジェクトを配備する際に、Deployment Wizard で PROJECT 変数を最終配備環境に適切な値に更新できます。それとは正反対に、アプリケーションサーバで Java ビジネスオブジェクトを照会するカスタムスクリプトを使って、使用するユーザ ID とパスワードを決定することもできます。

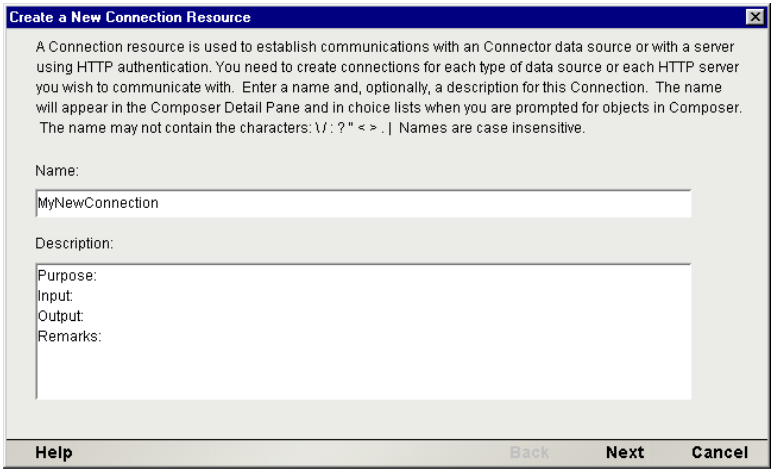
➢ 定数駆動型から式駆動型にパラメータを切り替える

1. 変更するパラメータフィールドでマウスを右クリックします。
2. コンテキストメニューから［**Expression**］を選択すると、エディタボタンが表示されるか、または有効になります。
3. ボタンをクリックしてから、ランタイム時に有効なパラメータ値を返す式を作成します。



➢ 接続を作成する

1 ［**File**］>［**New xObject**］>［**Resource**］>［**Connection**］の順に選択します。Create a New Connection xObject ウィザードが表示されます。

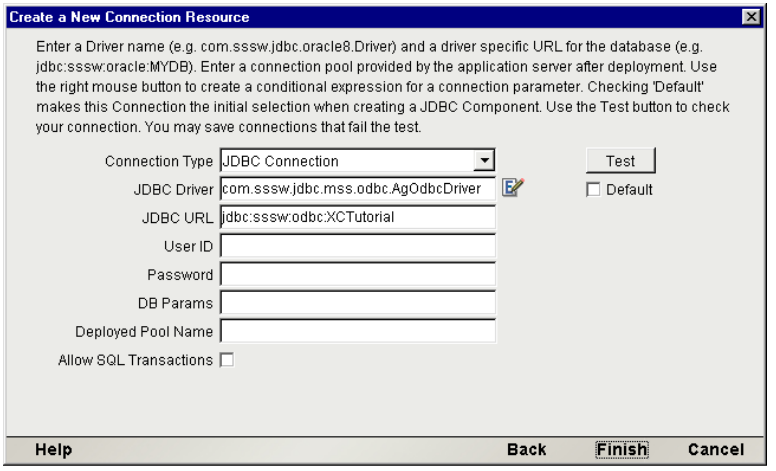


2 「**名前**」を入力します。

3 オプションとして、［**Description**］に説明テキストを入力します。

4 ［**Next**］をクリックします。Create a New Connection xObject ウィザードの 2 番目のページが表示されます。

5 ドロップダウンリストから HTTP 基本認証を選択します。この認証は、HTTP 接続を使用する XML 交換と同時に使用されます。



6 「**ユーザ ID**」と「**パスワード**」を入力します。これらは、接続の確立中に実際には送信されず、単にここで定義されます (パスワードは暗号化されます)。ユーザは、ECMAScript 式からユーザ ID およびパスワード変数にアクセスして、画面内の値として、ユーザ ID とパスワードをマップできるようになります。このため、パスワードが誰かに表示されることは決してありません。

7 ［**Browse**］ボタンをクリックしこのサービス接続で使用する証明書ファイルを選択して、「**クライアント証明書**」を選択します。

8 ［**Browse**］ボタンをクリックし、セキュリティ用のクライアントキーファイルを選択して、「**クライアントプライベートキー**」を選択します。

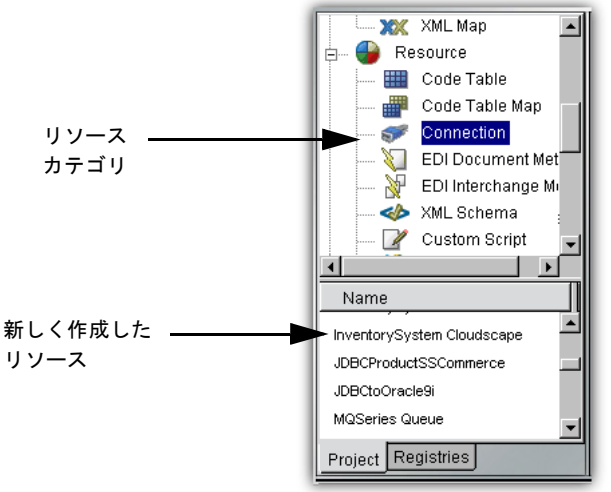
9 「**プライベートキー**」の「**パスワード**」を入力します。プライベートキーは、クライアントプライベートキーの所有者のための異なるレベルのセキュリティです。

10 「**接続タイムアウト**」の値を秒単位で入力します。

11 適切な Component ウィザードにこの接続をデフォルト接続として表示する場合は、［**Default**］チェックボックスをオンにします。

12 ［**Finish**］をクリックすると、接続が作成されます。

注記： 接続リソースの詳細については、各 Connect ガイドの「はじめに」の節を参照してください。



カスタムスクリプトリソースについて

カスタムスクリプトリソースとは、ECMAScript プログラミング言語で作成されたユーザ開発関数のライブラリです。関数をコンポーネント全体や他の関数内で使用できるようにすることができます。カスタムスクリプトを使用すると、次の操作を実行する関数を開発できます。

- 独自のカスタマイズに加えて、基本的な XML Map コンポーネントとほとんど同様の機能を使用する
- 文字列、日付、数値、正規表現などに関するデータを操作する
- W3C ECMAScript-to-DOM バインドメソッドを使用して XML ドキュメントを操作する
- 標準 Java クラスまたはカスタム Java クラスと統合する

注記： カスタム関数を作成するには、ECMAScript 言語を十分に理解しておく必要があります。次の節では、スクリプト作成のチュートリアルではなく、一般的なガイドラインのみを示します。Composer でのスクリプト作成の詳細については、次の章を参照してください。

カスタム関数の整理および使用

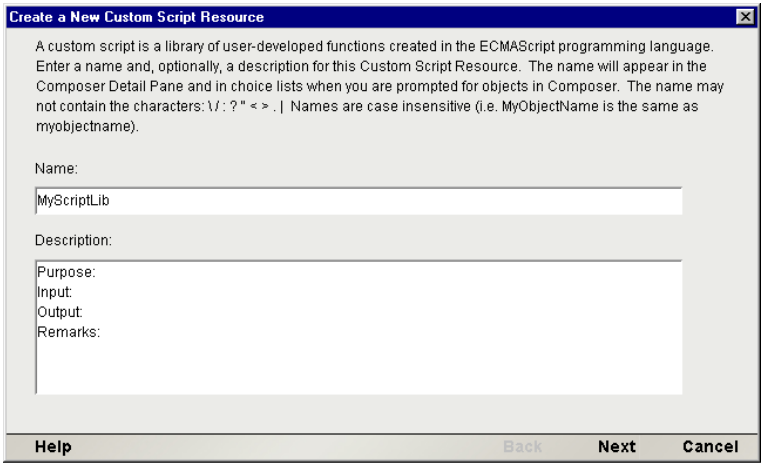
関数を異なるライブラリに整理したい場合があります。たとえば、アプリケーションに必要な math 関数、string 関数、または database 関数が複数あるとします。類似した関数を分類する (つまり、同じライブラリにすべての string 関数を作成する) と、カスタムスクリプトエディタを使用して、同じライブラリ内ですべての関数によって使用可能なグローバル変数を宣言することもできます。

関数を作成、検証する際に、Composer では、Component アクション内のすべての Expression Editor でこれらの関数が使用できるようにします。

たとえば、10 個の関数を含み、「String」と呼ばれるカスタム関数ライブラリを作成した場合、他の標準的な関数とともに、カスタムスクリプトというラベルで Expression Editor に表示されます。

➢ カスタムスクリプトを作成する

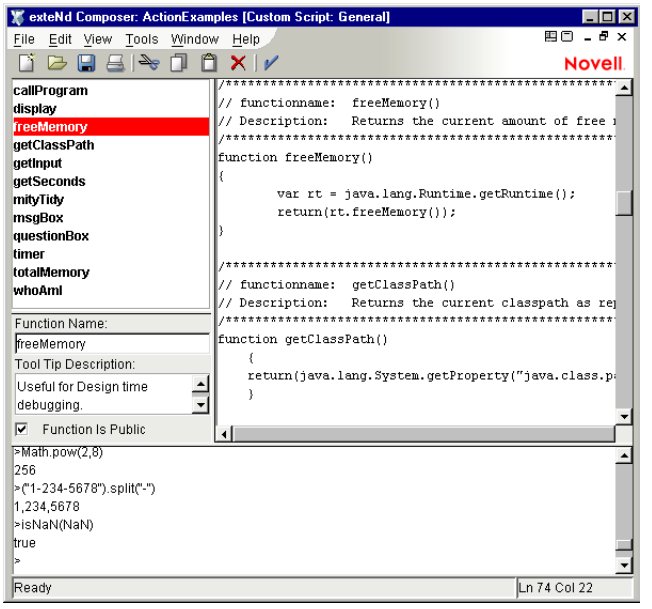
1 ［**File**］>［**New xObject**］>［**Resource**］>［**Custom Script**］の順に選択します。Create a New Custom Script xObject ウィザードが表示されます。



2 「**名前**」を入力します。

3 オプションとして、［**Description**］に説明情報を入力できます。

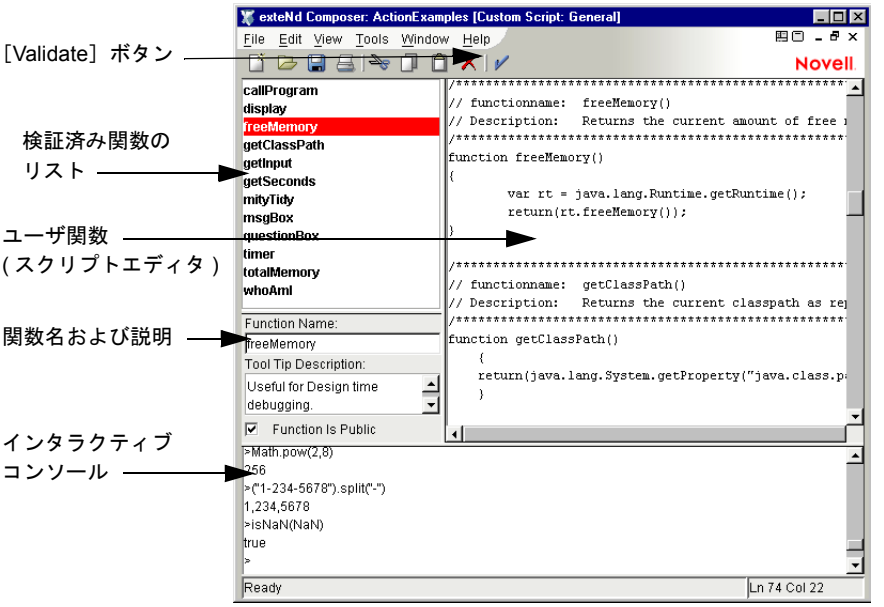
4 ［**Next**］をクリックします。カスタムスクリプトエディタが開き、新しく作成されたカスタムスクリプト名がタイトルバーに表示されます。



カスタムスクリプトエディタウィンドウについて

カスタムスクリプトエディタウィンドウは、複数のペインに分かれています。ペインのビューを切り替えて、必要なコンテンツを含めることができます。

複数の関数を追加した後のエディタは、次の図のとおりです。



注記 : 上のビューでは、[Output] (エラーメッセージ) ペインは、<Ctrl> -<shift> -<O> キーで非表示になっており、ナビゲーションフレームは、<Ctrl> -<shift> -<N> キーで非表示になっています。

関数の作成および検証

関数を入力して最初から作成します。関数を作成する際には Expression Builder を使用することもできます。詳細については、234 ページ「Expression Editor を使用した関数の作成」を参照してください。

➢ 関数を作成および検証する

1. 関数作成領域に、*function* という単語を入力します。
2. 同じ行で、*function* という単語の後に関数名を入力します。
3. 同じ行で、関数パラメータをカンマで区切って入力し、括弧で囲みます。
4. 左丸括弧を入力して、<Enter> キーを押します。
5. 関数ステートメント (複数可) を入力します。
6. 右丸括弧を入力して、<Enter> を押します。
7. 希望する場合は、コメントを関数に追加します。

関数は、次の例のようになります。

```
/*****************************************************************************/
// functionname: chr(aiCharCode)
// Description:  this function will return the ansi character for an integer value
// aCharCode:    is required, any integer value
// Returns:      ansi character for the value of the integer
// Note:         Unicode Values 0x20 to 0x7E and 0xA0 to 0xFF correspond to ASCII
/*****************************************************************************/
function chr(aiCharCode)
{
    return String.fromCharCode(aiCharCode)
}
```

➢ 関数の構文を検証する

- ［**Validate**］ボタンをクリックします。

関数が有効である場合、Composer により、検証された関数リストに関数名が追加されます。関数にエラーが含まれる場合は、Composer により、詳細なエラーメッセージが表示されます。

➢ 関数をテストする

1. テスト領域に有効なパラメータを完全に備えた関数名を入力します。
2. <Enter> を押します。

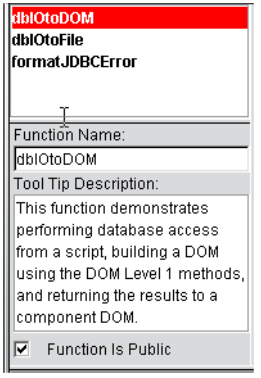
関数をテストする前には、前の節で説明した構文の検証が渡される必要があります。

関数のツールヒント説明の追加

関数を検証して、検証済み関数のリストに追加すると、関数の説明を記述できます。説明は、Composer 全体で Expression Builder に関数が表示されている場合に、マウスを関数名に合わせると「ツールヒント」として表示されます。

➢ 説明を追加する

1. 関数を作成および検証します。
2. ［Description］テキストボックスで、関数のテキストでデフォルトの式を上書きします (次の図を参照)。



3 希望する場合は、［**Function is Public**］チェックボックスをオンにします。このチェックボックスがオンの場合、次の 2 つの処理が実行できます。

- 任意のアクションの任意の Expression Builder から、関数を使用できます。
- ［Custom Scripts］見出しで、Expression Builder の選択リストに関数が表示されます。

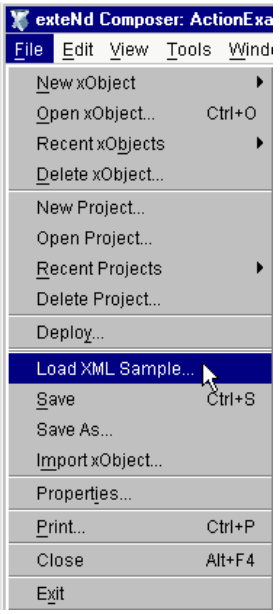
スクリプトエディタ内での DOM ツリーの表示

作成したカスタムスクリプト関数の多くで、XML ドキュメント内で特定の XPath の場所にあるデータを参照したり、データと直接動作させなければならない場合があります。この操作を簡単に行えるよう、カスタムスクリプトエディタでは、エディタ内で (ツリービュー、テキストビュー、または様式化されたビューの 3 つの使用可能なビューのうちいずれかで) XML ドキュメントを表示できます。これにより、関数定義の本文に XML 要素をドラッグアンドドロップできるため、XPath の参照をさらに簡単に指定できます。

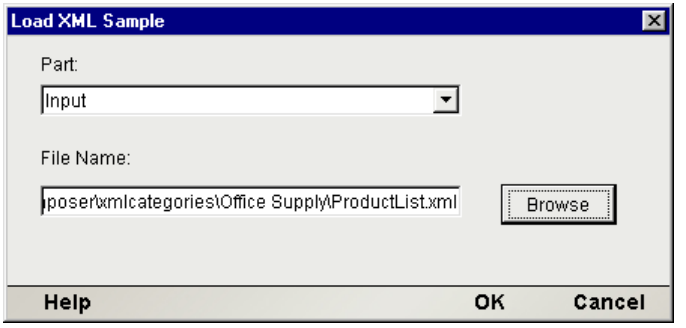
➢ XML ドキュメントを専用のペインで表示する

1 カスタムスクリプトを開き、スクリプトエディタ環境であることを確認してください。

2 Composer のメインメニューで、［**File**］、［**Load XML Samples**］の順に選択します (次の図を参照)。



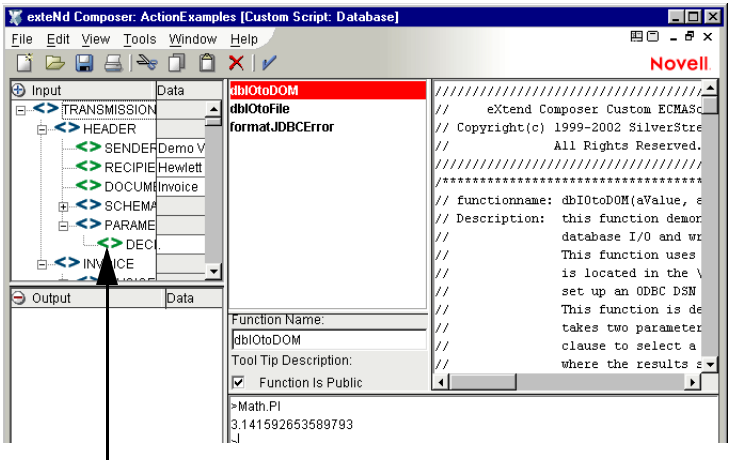
このコマンドを選択すると、［Load XML Sample］ダイアログボックスが表示されます。



3 ［**Part**］というラベルの付いたプルダウンメニューで、ファイルに関連付ける DOM (入力 DOM または出力 DOM) を選択します。

4 ［**Browse**］ボタンを使用して、ファイルのナビゲーションダイアログボックスを表示します。ロードする XML ファイルに移動して、ナビゲータを閉じます。

5 ［**OK**］をクリックして、［Load XML Sample］ダイアログボックスを閉じます。選択したファイルが専用のペインに表示されます。



XML ドキュメント
専用のペインでの表示

6 ファイルを使用、選択する XML ドキュメントのディレクトリに移動します。入力マップペインと出力マップペインが表示されます。

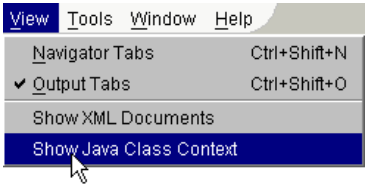
注記： XML テンプレートから XML ドキュメントを使用する場合、プロジェクトの「XMLCategories」ディレクトリで該当する「Imports」ディレクトリ (例 : **/Tutorial/XMLCategories/OfficeSupply/Imports**) に移動します。

Java クラスとカスタムスクリプトとの統合

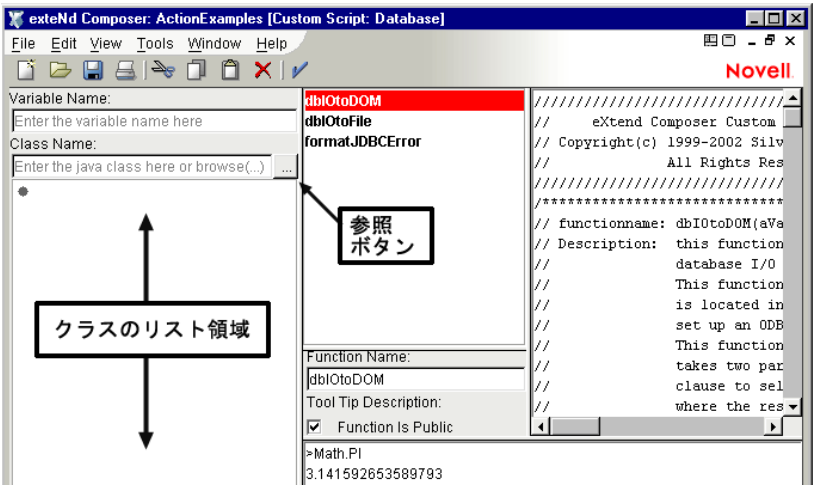
Java クラスを統合する必要があるカスタムスクリプトを作成している場合、カスタムスクリプトエディタウィンドウのビューを展開して、必要な情報を表示できます。作成したカスタムスクリプト関数の多くで、Java クラスを参照したり、Java クラスと直接動作させて、XML ドキュメントでデータを操作しなければならない場合があります。この操作が簡単に実行できるよう、カスタムスクリプトエディタでは、現在の CLASSPATH をスキャンして、検索したクラス、メソッド、およびプロパティを表示する Java クラスブラウザが提供されています。これにより、Java コンストラクタ、メソッド、およびプロパティを関数定義の本文、または関数をテストするテスト (コンソール) 領域にドラッグアンドドロップできるため、これらの項目をさらに簡単に指定、使用できます。

➢ Java クラスを使用する

1 カスタムスクリプトエディタのメニューバーから［**View**］>［**Show Java Class Context**］の順に選択します。

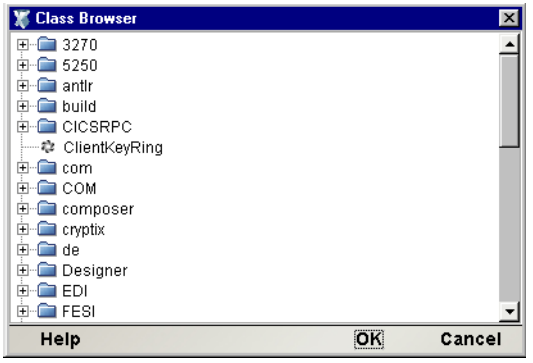


このコマンドを選択すると、メインエディタに Java クラスパネルが表示されます。



2 Java クラスパネルで、次の操作を実行します。

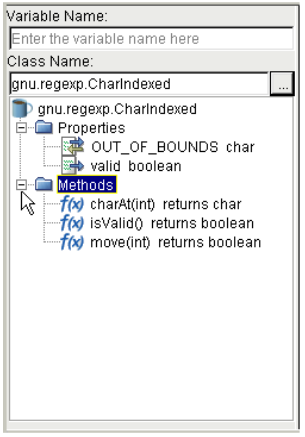
- ［**Class Name**］フィールドに名前を入力します。または、
- ［**Browse**］ボタンをクリックします。しばらくすると、［Class Browser］ダイアログボックスが開き、Composer CLASSPATH に使用可能な Java パッケージが表示されます。



3 使用するクラスに達するまでコンテキストツリーを移動します (コンテキストノードの左側にある小さなマイナス記号をクリックすると、ノードが展開されます)。

4 使用するクラスを選択して、[**OK**] をクリックすると、ダイアログボックスが閉じます。エディタのクラスリスト領域に、クラスが表示されます。

5 (クラスをダブルクリックするか、横にあるプラス記号をクリックして) クラスを展開し、コンストラクタ、メソッド、およびプロパティを表示します (次の図を参照)。



6 (オプション) 個々のメソッドまたはプロパティを、エディタペインまたはコンソールにドラッグアンドドロップして、該当するプロパティを使用します。

希望する場合は、Composer CLASSPATH に独自のクラスを追加するか、Composer の CLASSPATH を拡張して独自のクラスを含めることで、[Class Browser] に独自のクラスを追加できます。

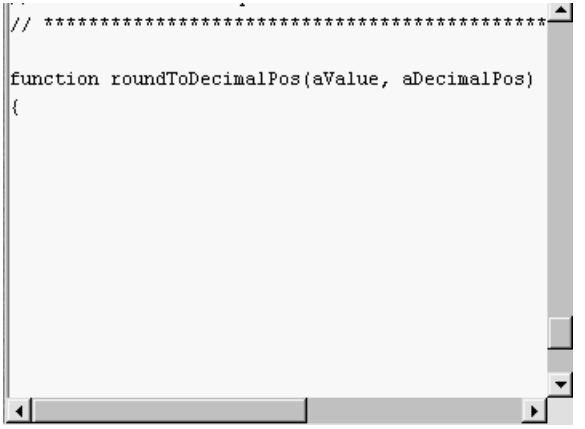
注記 : カスタム Java クラスを使用している場合は、アプリケーションの配備時にアプリケーションサーバにクラスをインストールする (または EAR/WAR ファイルに含める) ようにしてください。その場合には、exteNd Developer Workbench を使用できます。詳細については、Workbench マニュアルを参照してください。

ECMAScript での Java クラスの操作

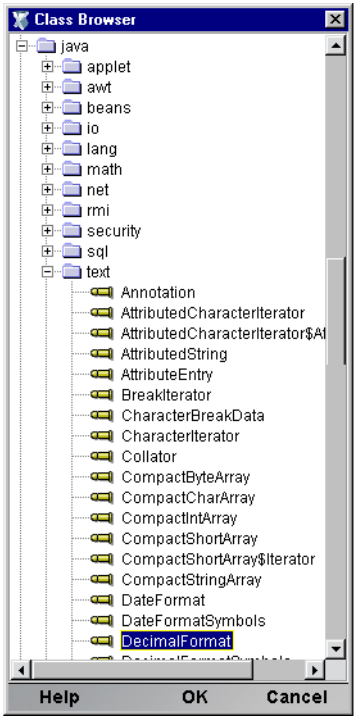
Java の `DecimalFormat` クラスを使用する、`RoundToDecimalPos()` と呼ばれるスクリプト関数を作成する方法は、次の例のとおりです。この例では、関数で、切り上げ / 切り捨てする数値、および切り上げ / 切り捨てする箇所の数という 2 つのパラメータを受け入れます。

➢ Java を使用するスクリプト関数を作成する

1 Java クラスパネルが表示された状態で、関数ペインに関数シグネチャを入力します (次を参照)。

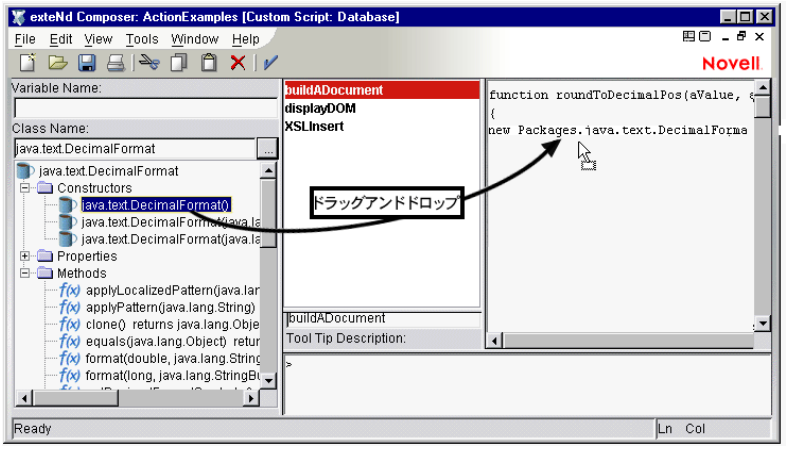


2 ［**Class Name**］コントロールで、参照ボタンを選択します。［Class Browser］ダイアログボックスが表示されます。



3 ［**Java**］>［**Text**］>［**DecimalFormat**］の順に移動します (前の図を参照)。

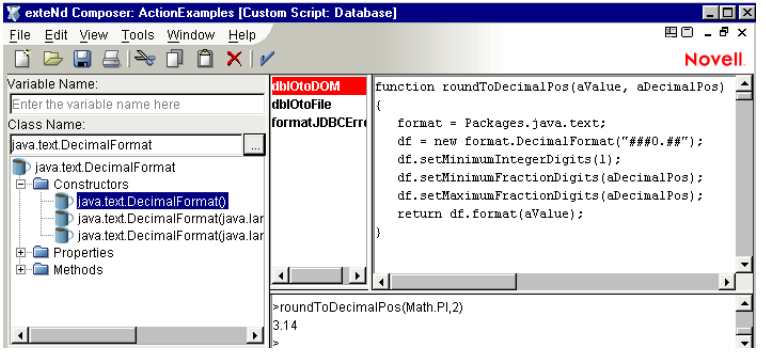
4 ［**OK**］をクリックします。カスタムスクリプトウィンドウの［**Class Name**］フィールドが自動的に入力されます。



5 (オプション)［**Variable Name**］フィールドに名前を入力します。

6 希望する「**コンストラクタ**」を関数ペインにドラッグアンドドロップします。コンストラクタのパラメータを入力します。

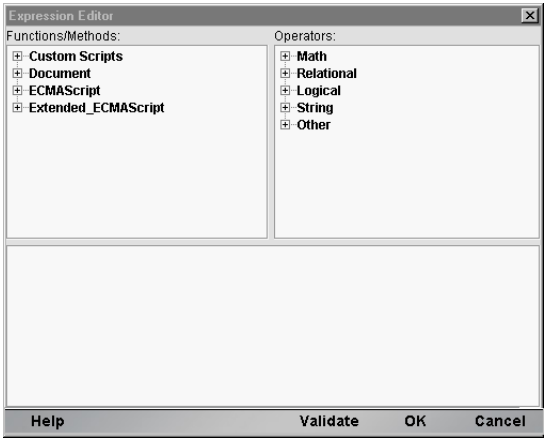
7 希望する場合は、ECMAScript 関数を編集します。関数の一例は、次のとおりです。



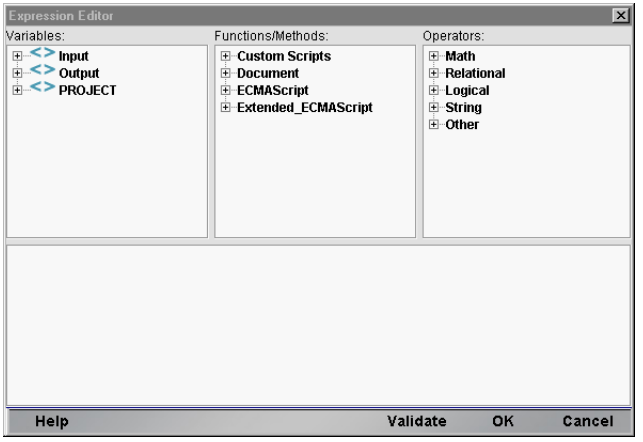
Expression Editor を使用した関数の作成

関数を最初から作成する代わりに、Expression Editor を使用して関数を作成することができます。この機能を使用する場合の利点は、Expression Editor では、マウス操作だけで使用できる選択リストを使用することで、DOM メソッド、Composer 拡張、組み込み ECMAScript メソッド、および DOM ノードターゲットがほとんどすべて明らかになります。選択リストを使用して式を作成すると、操作が簡単で便利なだけではなく、入力ミスをする恐れが少なくなります。また、すべての使用可能なメソッドに対する構文の呼び出しは、選択ツリーで各リーフノードのロールオーバツールヒントに表示されるため、役に立つ参照にもなります。

カスタムスクリプトエディタには、選択したビューに基づき、［Expression Editor］内で 2 つの異なるビューが表示されます。基本的なビューには、独自の関数を作成するために使用できる ECMAScript オブジェクトと演算子のリストが表示されます (次を参照)。

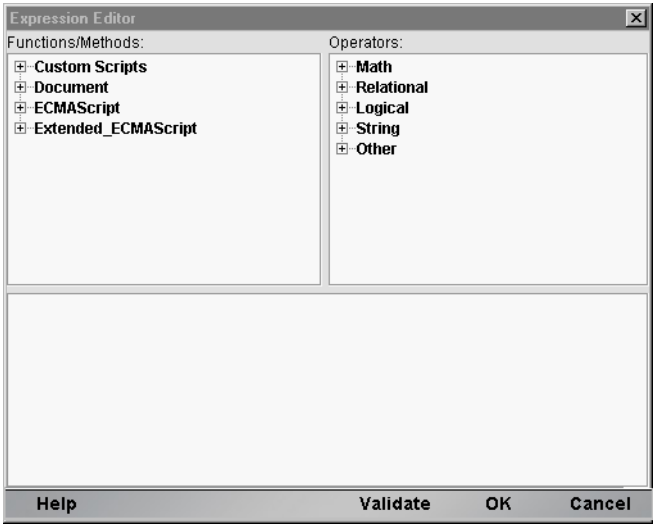


カスタムスクリプトエディタで、［**View**］、［**Show XML Documents**］の順に選択すると、［Expression Editor］が開き、DOM で要素を選択するための追加の選択リストが表示されます (次を参照)。



➢ Expression Editor を使用する

1 メインメニューバーで、［**Tools**］、［**Expression Editor**］の順に選択します。［Expression Editor］が表示されます。



2 ［**Variables**］ペイン、［**Functions/Methods**］ペイン、または［**Operators**］ペインでツリーを展開し、要素をダブルクリックして関数を作成します。要素をダブルクリックすると、一番下の式ペインに表示されます。

または、マウスの右ボタンを使用して、関数の編集ペインまたはテスト領域のいずれかに Expression Builder を表示することもできます。

注記： ECMAScript の詳細については、このガイドの次の章「exteNd Composer でのカスタムスクリプト作成および XPath 論理」を参照してください。

WSDL リソースについて

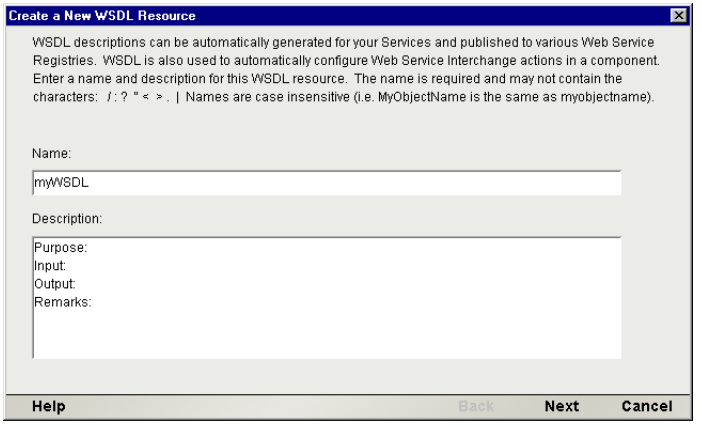
WSDL (Web Services Description Language) とは、Web サービスを記述するための XML ボキャブラリです。WSDL を使用すると、ビジネスがオンラインでやりとりするために十分詳細なレベルで、インタフェース、プロトコルのバインド、および Web ベースのサービスに関するさまざまなタイプの情報を、(標準の方法で) 記述することができます。完全な標準については、**http://www.w3.org/TR/wsdl** を参照してください。

WSDL リソースを生成するには 3 つの方法があります。1 番目の方法は、Composer の XML エディタを使用して WSDL ドキュメントを最初から作成することです。2 番目の方法は、Composer で WSDL を生成します (Composer では、プロジェクトに追加した任意の Web サービスに対して WSDL ファイルを自動生成できます)。これら 2 つの方法に対する手順は、次のとおりです。

また、WSDL をプロジェクトに直接ダウンロードすることで、レジストリ (UDDI パブリックレジストリなど) から WSDL を取得することもできます。この方法は、次に詳しく説明します。

➢ 既存のサービスから WSDL リソースを生成する、または XML エディタで WSDL リソースを作成する

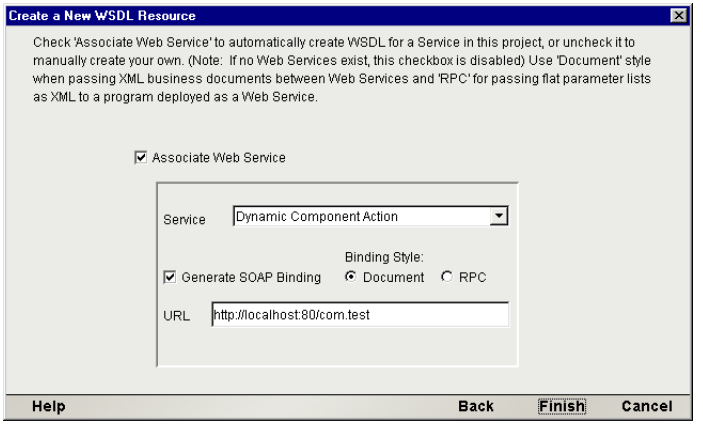
1. Composer の［**File**］メニューから、［**New xObject**］、［**Resource**］、［**WSDL**］の順に選択します (または、カテゴリペインで［WSDL Resource］アイコンを右クリックして、［**New**］を選択します)。WSDL Resource ウィザードの最初のペインが表示されます。



2 リソースの「**名前**」を入力します (この名前は、WSDL の **/service** 要素の name 属性にも表示されます)。

3 オプションとして、説明情報を入力します。

4 ［**Next**］をクリックします。新しいペインが表示されます。



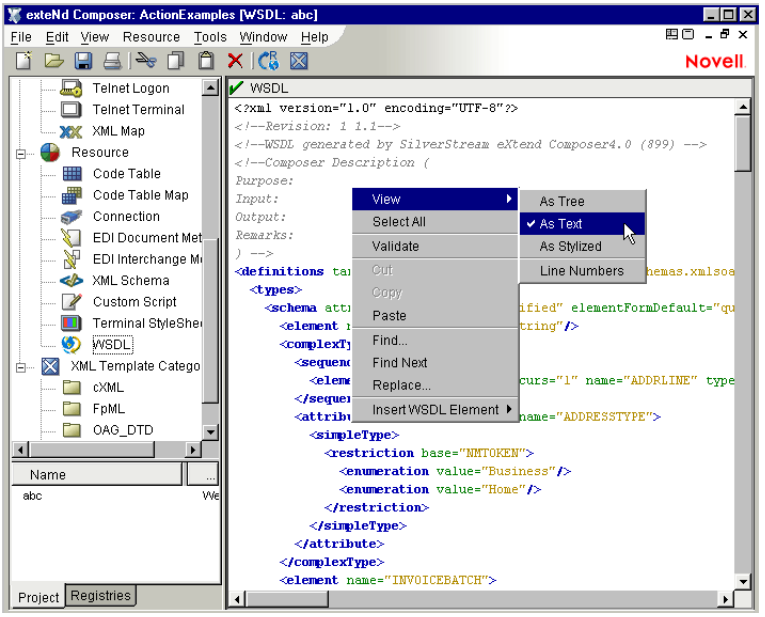
5 プロジェクトでの既存の Web サービスに基づき WSDL を作成する場合は、［**Associate Web Service**］チェックボックスをオンにします (代わりに、XML エディタ環境で独自の WSDL を手動で作成する場合は、このチェックボックスをオフのままにして、ダイアログボックスを閉じてから［**Finish**］をクリックします。次に、Composer のコンテンツペインで右クリックして、コンテキストメニューから［View As Text］を選択し、入力していきます)。

6 ［**Service**］プルダウンメニューからサービスを選択します。

7 Composer で、WSDL での SOAP バインド情報を自動的に作成する場合は、［**Generate SOAP Binding**］チェックボックスをオンにします。［**Document**］および［**RPC**］というラベルの付いたラジオボタンから、使用するバインドスタイルを選択します。

8 WSDL の **/service/port/address** 要素の location 属性に表示する URI を入力します。

9 ［**Finish**］をクリックします。新しく作成した WSDL が、Composer のコンテンツウィンドウで DOM ツリーとして表示されます。DOM を右クリックして、［**View**］>［**As Text**］の順に選択して、WSDL ドキュメントのテキストビューを表示します。この WSDL ドキュメントは、必要に応じて後で手動で編集できます (次を参照)。



➢ レジストリブラウザを使用して外部サービスから WSDL を取得する

1. Composer のナビゲーションフレームで［**Registries**］タブをクリックします。
2. 335 ページ「レジストリ参照」で説明されているとおり、(ビジネスまたはサービスのいずれかを) 検索します。
3. ［Service］ペインに詳細情報を表示できるサービスを選択します。
4. 346 ページ「レジストリからの WSDL の取得」で説明されているとおり、そのサービスの WSDL を取得します。取得した WSDL のツリービューが、コンポーネントエディタのコンテンツペインに自動的に表示されます (他のビューを選択するには、コンテンツペイン内で右クリックして、コンテキストメニューから［**View As**］を選択します)。
5. Composer の［File］メニューから、［**Save As**］を選択します。リソースの名前を入力し、［**OK**］をクリックします。
6. 取得した WSDL に基づき、新しい WSDL リソースが Composer のナビゲーションフレームのインスタンスペインに表示されます (また、WSDL は、この時点でディスクに保持されます)。

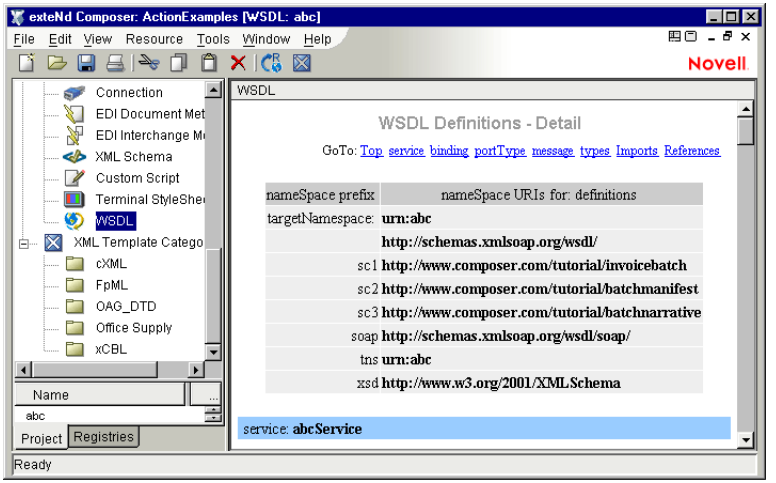
様式化されたビューの取得

デフォルトでは、WSDL リソースをはじめて開くと、ドキュメントは構文に色が付けられたテキスト編集ビューで表示されます。ドキュメントに XSL スタイルシートを適用して作成した、WSDL ドキュメントの様式化されたビューを表示することもできます。

➢ WSDL ドキュメントの様式化されたビューを表示する

1 WSDL リソースを開きます。

2 WSDL エディタペインで右クリックして、コンテキストメニューから［**View**］>［**As Stylized**］の順に選択します。しばらくすると、様式化されたビューに切り替わります。

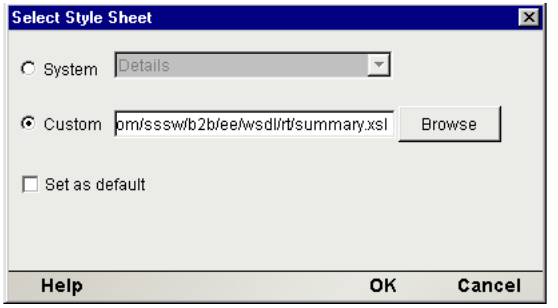


ここでは、要約スタイルシートがドキュメントに適用されています。希望する場合は、代わりにカスタムスタイルシートを適用できます (次の手順を参照)。

➢ 様式化されたビューのカスタムスタイルシートを選択する

1 様式化されたフォームに WSDL ドキュメントがすでに表示された状態で、ペイン内を右クリックします。コンテキストメニューが表示されます。［**Select Style Sheet**］をクリックします。

次のダイアログボックスが表示されます。



2 様式化されたビューの基礎として、既存の標準スタイルシート ([Details] または [Summary]) のいずれかを選択する場合は、[**System**] ラジオボタンをオンにします。

- [**Details**] では、(XML タグを持たない) WSDL ドキュメントの詳細なプレーンテキストビューを表示できます。
- [**Summary**] では、WSDL コンテンツのより簡潔なビューを表示できます。

3 それ以外の場合は、[**Custom**] ラジオボタンをオンにして、選択したスタイルシートへのパスを入力します (または、[Browse] ボタンを使用して、標準のファイルのナビゲーションダイアログボックスを表示します)。

4 様式化されたビューで選択したスタイルシートをデフォルトとして適用する場合は、[**Set as default**] チェックボックスをオンにします。この設定は、Composer セッション全体で適用されます。

WSDL ドキュメントへの要素の追加

WSDL エディタを使用すると、他のテキストエディタと同様に、通常のテキストの挿入、および切り取り、貼り付けを使用した編集が行えます。標準の WSDL ドキュメント要素を簡単かつ迅速に作成できるよう設計された特別なコンテキスト機能を利用することもできます。

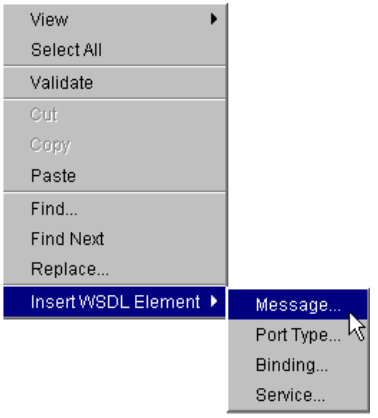
WSDL ドキュメントには、message、portType、binding、および service という 4 つの標準の要素タイプが含まれる場合があります。さらに、これらの要素は、カスケーディング参照を使用して相互に構成されているため、ダイアログボックスを使用しないで WSDL ファイルを作成する際には、message セクションを最初に作成してから、portType セクション、binding セクションの順に作成し、最後に service セクションを作成することをお勧めします。WSDL エディタでは、ダイアログボックスベースでこれら 4 つのタイプを個々に作成できます。

メッセージ要素の追加

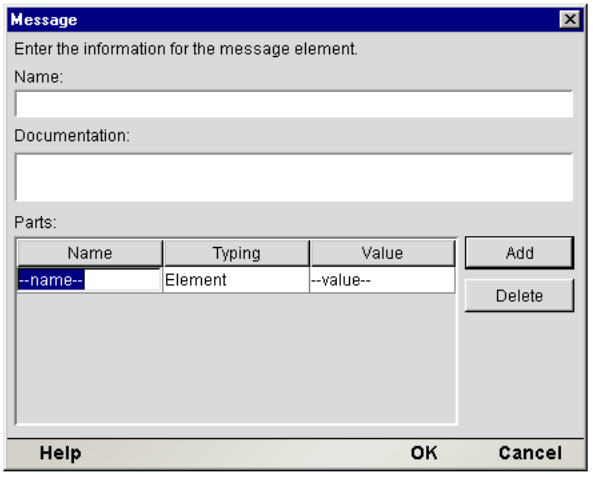
WSDL では、メッセージとは交換されるデータの抽象的な、型定義です。ランタイム時には、実際のメッセージは DOM として表されます。

➢ 新しいメッセージ要素を作成する

1. WSDL リソースが開いていない場合は、WSDL リソースを開きます。
2. WSDL ドキュメントがテキストビューモードになっていることを確認してください (ドキュメントの任意の場所を右クリックして、［**View**］>［**As Text**］の順に選択します)。
3. テキストビューペイン内でマウスを右クリックします。コンテキストメニューが表示されます。



4. ［**Insert WSDL Element**］>［**Message . . .**］の順に選択して、新しいメッセージの挿入ダイアログボックスを表示します。



5. ［**Add**］ボタンをクリックして、［Parts］テーブルに空白の行を追加します。

6 ［**Name**］テキストフィールドに、メイン <message> 要素の name 属性の値を入力します。

7 ［**Documentation**］フィールドに、定義要素に関連付ける読みやすいコメントまたは記述言語を入力します (オプション)。

8 ［Parts］で、［**Name**］列に、メッセージセクションの最初の <part> 要素に対する name 属性を入力します。

9 プルダウンメニューから［**Typing**］の値に［Element］または［Type］を選択します。

10 ［**Value**］で、このパートの element 値を入力します。

11 ［Add］ボタンをクリックして、このメッセージへの別のパートエントリを追加します。

12 エントリを削除するには、最初にエントリをクリックして該当する行を選択してから、［**Remove**］ボタンをクリックしてそのエントリを削除します。

13 ［**OK**］をクリックします。ダイアログボックスが閉じ、ドキュメントに新しいセクションが追加されます。

```
<message name="GetLastTradePriceOutput">
     <part name="body" element="xsd1:TradePriceResult"/>
</message>
```

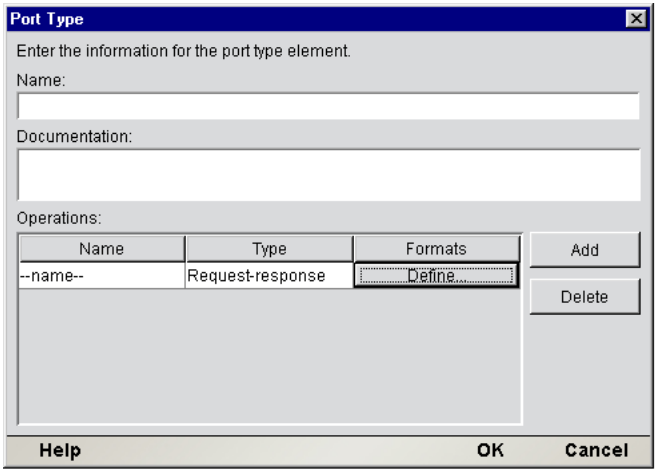
ポートタイプ要素の追加

WSDL ポートタイプは、サービスおよびサービスで使用される通信モード (一方向、リクエスト - 応答など) によってサポートされた操作の抽象的な定義です。

➢ WSDL ドキュメントに新しいポートタイプを追加する

1 エディタのテキストビューペインにマウスを合わせ、マウスを右クリックします。コンテキストメニューが表示されます。

2 ［**Insert**］>［**Port Type . . .**］の順に選択して、新しいポートタイプの挿入ダイアログボックスを表示します。



3 ［**Add**］ボタンをクリックして、［Parts］テーブルに空白の行を追加します。

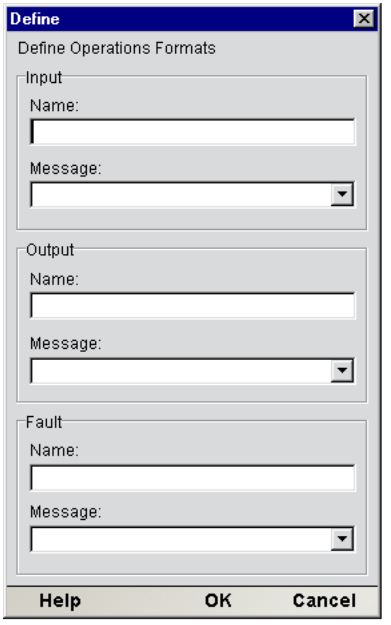
4 ［**Name**］フィールドに、作成している <portType> 要素の name 属性の値を入力します。

5 ［**Documentation**］フィールドに、定義要素に関連付ける読みやすいコメントまたは記述言語を入力します (オプション)。

6 ［Operations］で、この操作の「**名前**」を入力します。

7 ［**Type**］列で、プルダウンメニューから［One-Way］、［Request-Response］、［Solicit-Response］、または［Notification］を適宜選択します。

8 ［**Formats**］で、入力メッセージおよび出力メッセージを入力するか、操作の編集ダイアログボックスを使用して適切なメッセージを作成します。操作の編集ダイアログボックスを開くには、行の最後で［**Set. . .**］ボタンをクリックします。新しいダイアログボックスが開きます。



9 操作の編集ダイアログボックスには、複数のコントロールグループがあります。該当する操作 (リクエスト - 応答、依頼 - 応答など) に適切なコントロールのみが、有効になります。たとえば、前の手順 6 で［Type］列から［Notification］を選択した場合、出力コントロールグループのみが有効になります。有効になった各グループでは、操作に適切な［**Name**］および［**Message**］が、［**Input**］および［**Output**］に対して必要となります。ただし、［**Fault**］グループは、必須ではなくオプションです。

10 ［**OK**］をクリックして、操作の編集ダイアログボックスを閉じます。

11 ［**Add**］をクリックして、現在のポートタイプセクションにさらに操作を追加します。

12 操作を削除するには、削除する操作を選択してから、［**Delete**］ボタンをクリックします。

13 ［**OK**］をクリックして、新しいポートタイプの挿入ダイアログボックスを閉じます。WSDL ドキュメントに新しいセクションが追加されます。

```
<portType name="StockQuotePortType">
    <operation name="GetTradePrice">
        <input name="input"
message="tns:GetLastTradePriceInput"/>
        <output name="output"
message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>
```

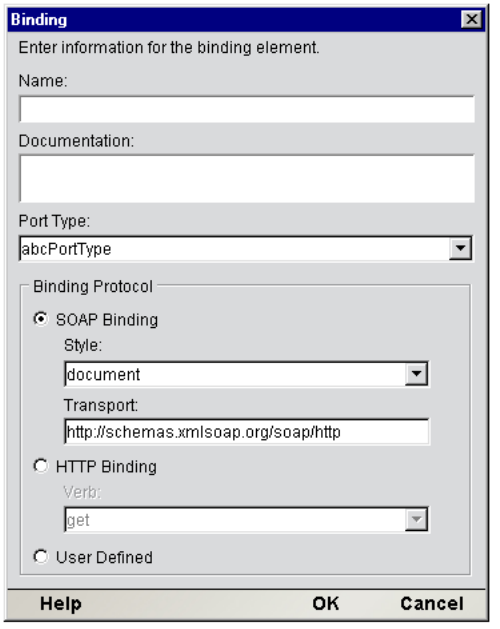
バインド要素の追加

バインドでは、特定のポートタイプによって定義された操作とメッセージに対する具体的なプロトコルおよびデータ形式の仕様が指定されます。

➢ WSDL ドキュメントに新しいバインドを追加する

1 エディタのテキストビューペインにマウスを合わせ、マウスを右クリックします。コンテキストメニューが表示されます。

2 ［**Insert**］>［**Binding . . .**］の順に選択して、新しいバインドの挿入ダイアログボックスを表示します。



3 ［**Name**］フィールドに、作成している <binding> 要素の name 属性の値を入力します。

4 ［**Documentation**］フィールドに、定義要素に関連付ける読みやすいコメントまたは記述言語を入力します (オプション)。

5 ［**Port Type**］の横にあるプルダウンメニューを使用して、このバインドに対する適切なポートタイプを選択します。プルダウンメニューには、このドキュメントに対してこれまでに作成した (該当する場合) ポートタイプの名前が含まれます (前の「ポートタイプ要素の追加」を参照)。

6 WSDL ドキュメントで SOAP バインドを指定する場合、[**SOAP Binding**] チェックボックスをオンにしてから、[**Style**] のプルダウンメニューで [RPC] または [Document] を選択して、[**Transport**] に値を入力します (または、デフォルト値を受け入れます)。

7 HTTP バインドを使用する場合は、[**HTTP Binding**] チェックボックスをオンにして、[**Verb**] に適切な動詞 ([GET] または [POST]) を入力します。

8 [**OK**] をクリックして、ダイアログボックスを閉じます。WSDL ドキュメントに新しいバインドセクションが追加されます。

```
<binding name="StockQuoteSoapBinding"
type="tns:StockQuotePortType">
    <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
        <soap:operation
soapAction="http://example.com/GetLastTradePrice"/>
        <input>
            <soap:body use="literal"
namespace="http://example.com/stockquote.xsd
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output>
            <soap:body use="literal"
namespace="http://example.com/stockquote.xsd"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
    </operation>
</binding>
```

サービス要素の追加

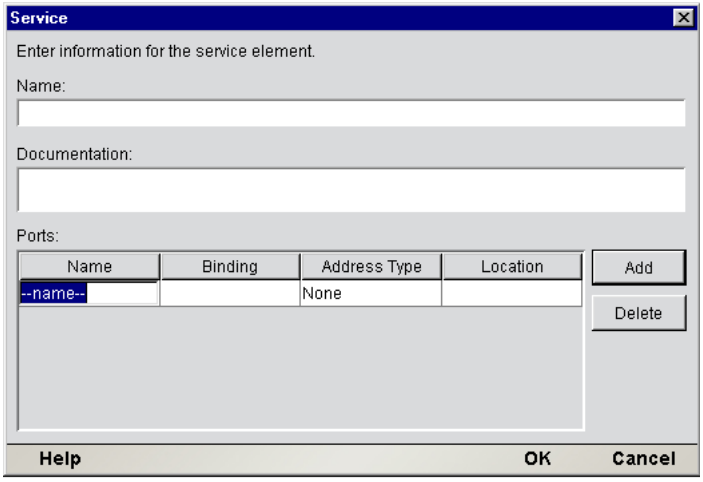
サービス要素では、該当するサービスのエントリポイントアドレス (複数可) を命名します。これらのアドレスは、URI の形式で、「ポート」を構成します。

➢ WSDL ドキュメントに新しいサービスを追加する

1 エディタのテキストビューペインにマウスを合わせ、マウスを右クリックします。コンテキストメニューが表示されます。

2 [**Insert**] > [**Service . . .**] の順に選択して、新しいサービスの挿入ダイアログボックスを表示します。

3 [**Add**] ボタンをクリックして、サービステーブルに空白の行を追加します。



4 ［**Name**］フィールドに、作成している <service> 要素の name 属性の値を入力します。

5 ［**Documentation**］フィールドに、このサービス要素に関連付ける読みやすいコメントまたは記述言語を入力します (オプション)。

6 ［Ports］セクションの［**Name**］で、この <port> 要素の名前を入力します。

7 ［**Binding**］列で、プルダウンメニューから既存のバインドを選択します。［Binding］セクションには、このドキュメントに対してすでに作成されている (該当する場合) 使用可能なバインドが反映されます。

8 ［**Address Type**］列で、プルダウンメニューを使用して［None］、［SOAP］、または［HTTP］を適宜指定します。

9 ［**Location**］で、サービスが開始する URI を入力します。

10 ［Add］をクリックして、サービスにさらに行 (追加のポートエントリ) を追加します。

11 エントリを削除するには、エントリを選択してから、［**Delete**］ボタンをクリックします。

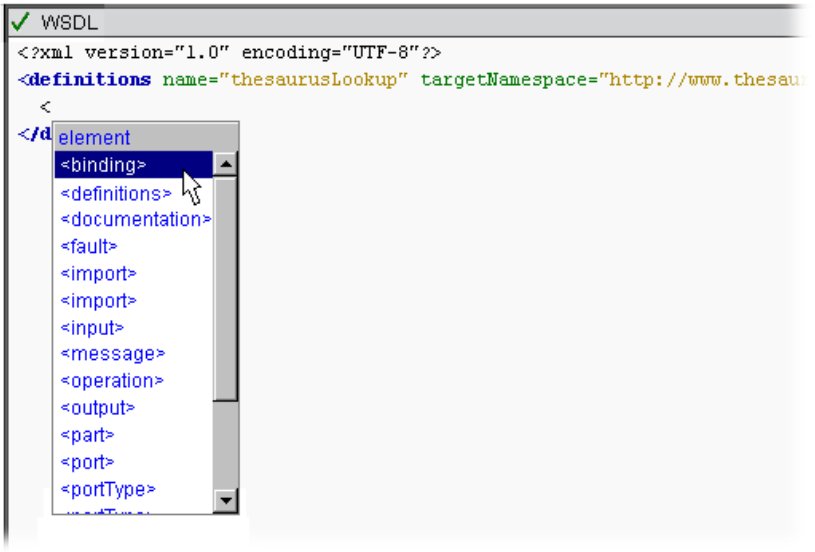
12 ［**OK**］をクリックして、ダイアログボックスを閉じます。WSDL ドキュメントに新しいサービスエントリが追加されます。

```
<service name="StockQuoteService">
    <port name="StockQuotePort"
binding="tns:StockQuoteBinding">         <soap:address
location="http://example.com/stockquote"/>
    </port>
</service>
```

先読み入力 (コードの完了)

WSDL エディタには、< 記号 (つまり要素タグの開始) を入力すると有効になる「スマート先読み入力」機能が組み込まれています。コンテキストメニューが自動的に表示され、ファイルで指定したスキーマ、およびドキュメントの場所に基づき、使用できるタグ名の選択肢が表示されます。

たとえば、手動で WSDL ドキュメントを作成しており、ドキュメントの上部に近づいた場合、「<」を入力するとカーソルの場所にメニューが開き、次の選択肢が表示されます。



このメニューの要素名は、WSDL での有効なタグ名に対応していることにご注意ください。メニュー項目を選択するには、単純にダブルクリックします。

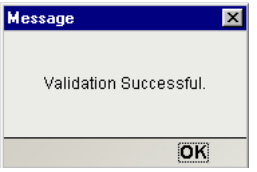
メニューの選択肢は、状況依存型です。つまり、要素ツリー内の深い階層で、< 記号を入力した場合、先読み入力メニューに表示される選択肢は、入力しているXPath コンテキストで有効となる値に制限されます。たとえば、WSDL ファイルの任意の場所にある <documentation> ノード内で、「<」を入力した場合、この時点で作成できる唯一有効なタグは、終了タグとなるため、</documentation> という選択肢が 1 つだけ表示された先読み入力メニューが開きます (WSDL スキーマでは、documentation 要素内でチャイルド要素は許可されていません)。

もちろん、常に先読み入力メニューは無視して、状況に応じて任意の項目を入力することはできます (例 : コメントなど)。

注記： 先読み入力のヒントは、ドキュメントに適用されるスキーマに基づきます。明らかに、ドキュメントでネームスペースまたはスキーマが指定されない場合、エディタで有効なタグの選択肢を「認識」する方法はなく、先読み入力メニューは表示されません。

WSDL ドキュメントの検証

WSDL ドキュメントが開いており、コンテンツがエディタに表示されている場合、WSDL ドキュメントウィンドウの左上隅にある小さな緑色のチェックマークのアイコンをクリックして、WSDL ドキュメントを検証できます。ドキュメントが正常に検証されると、ダイアログボックスが表示されます (次を参照)。



それ以外の場合は、エラーが表示され、ドキュメントの不正なステートメントを特定する情報が示されます。

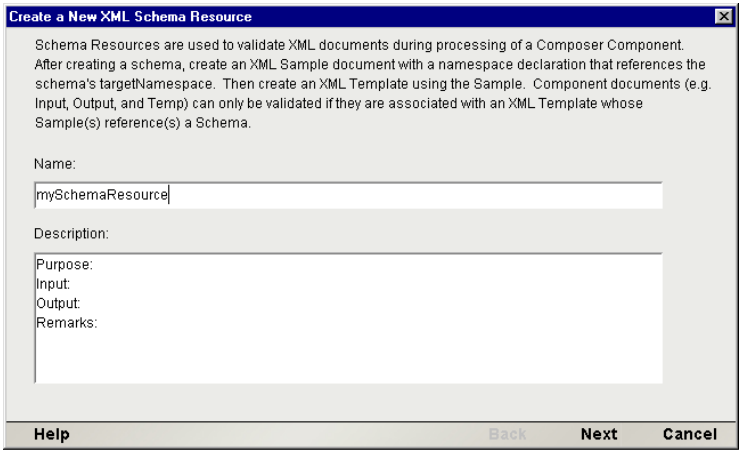
注記： ドキュメントの検証が正常に行われた場合でも、慎重に WSDL を確認する必要があります。W3C WSDL 仕様では、WSDL ドキュメントのすべてのレベルで拡張要素が許可されています。そのため、ダイアログボックスを使用せずにドキュメントを作成したり、他のソースから大幅に切り取り、貼り付けを行った場合、ドキュメントは有効としてテストされますが、必ずしも希望に添わない場合があります。

XML スキーマリソースについて

XML スキーマ定義ファイル (XSD) は、さまざまなコンポーネント、サービス、および Composer プロジェクトによって再使用でき、また XML スキーマ定義ファイルを使用するすべてのプロジェクトやコンポーネントに一度に再度インポートせずに編集、変更できるよう、専用のリソースタイプで指定されます。

➢ スキーマリソースを追加する

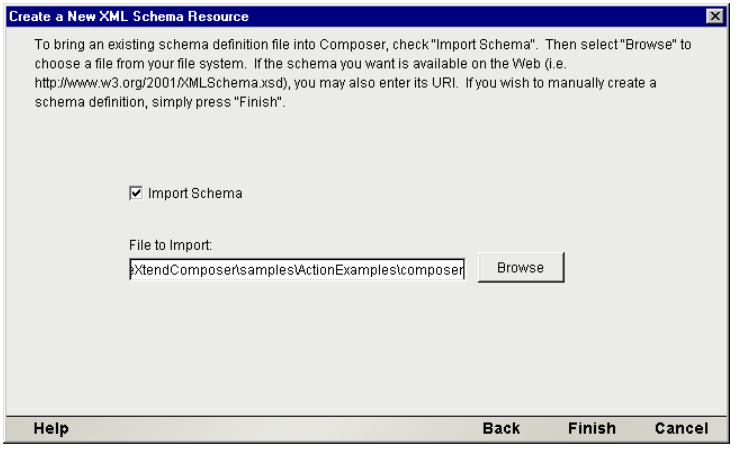
1 Composer の［**File**］メニューから、［**New xObject**］、［**Resource**］、［**XML Schema**］の順に選択します (または、カテゴリペインの［XML Schema Resource］アイコンを右クリックして、［**New**］を選択します)。XML Schema Resource ウィザードの最初のペインが表示されます。



2 リソースの「**名前**」を入力します。

3 オプションとして、リソースに関する説明情報を入力します。

4 ［**Next**］をクリックします。ウィザードの次のペインが表示されます。



5 (前から存在するファイルを使用せずに)スキーマ定義ファイルを手動で作成する場合は、単純にウィンドウ下部の［**Finish**］をクリックします。Composer のコンテンツウィンドウでスキーマを作成できます。

6 既存のスキーマ **(.xsd)** ファイルをディスク上で指定する場合は、［**Import Schema**］チェックボックスをオンにします。

7 ファイルシステムを移動して、スキーマファイルを検索する場合は、[**Browse**] ボタンをクリックします。すでにファイルの名前と場所が分かっている場合は、[**File to Import**] テキストフィールドにスキーマファイルの URI を入力します (ここで入力する文字列は、完全修飾 URI にもできます)。

8 [**Finish**] をクリックしてダイアログボックスを終了します。XML スキーマリソースが、インスタンスペインに追加されます。

9 オプションとして、コンテンツペインを右クリックし [**View As**] > [**Text**] の順に選択して、XML エディタに移動します。

10 exteNd Composer でのカスタムスクリプト作成および XPath 論理

exteNd Composer には、Composer アプリケーションの機能をさまざまな方法で拡張できるオンボードの EMCAScript インタプリタが組み込まれています。たとえば、スクリプトを作成して、次の処理を実行できます。

- DOM Level 2 メソッドを使用して、XML データを直接操作する
- プログラムによって Composer コンポーネントを実行する
- Java を直接呼び出す
- ファイル I/O 処理を実行する
- カスタム処理論理を使用してアクションモデルを拡張する
- 一般的なデータ操作タスクを実行するための独自のライブラリを開発する
- データ接続をランタイム時に動的にバインドする
- デバッグで alert( ) 関数を使用する
- 迅速にプロトタイプを作成して、最終的には Java で実装する可能性のある概念をテストする

XPath 言語でも、Composer コンポーネントでカスタム論理を開発できます。XPath 仕様には、XML データのフィルタ、修飾、集約、または配置、あるいはこれらすべてのために使用できる 20 以上の定義済み関数が含まれています。

この章では、Composer におけるカスタム ECMAScript または XPath 論理、あるいはその両方の使用に適用できる手法と機能をいくつか説明します。また、さまざまな W3C 標準と、XPath および ECMAScript との関係についても説明します。

ECMAScript とは

ECMAScript は、カスタム論理を使用できるようにすることによって多様なホスト環境 (Web ブラウザ、エディタ、IDE など) の機能を拡張するための軽量型オブジェクト指向スクリプト言語です。ECMAScript は、exteNd Composer をはじめとするホストプログラムの既存の機能を補完または拡張するためのものです。Web ブラウザの環境では、ECMAScript は、多くの場合、JavaScript または JScript と呼ばれます。

ECMAScript は、次のような理由から特に Java ホスト環境に適しています。

1. Java によく似た構文を使用するオブジェクト指向言語である
2. ECMAScript で作成したスクリプトから、Java のコンストラクタとメソッドを直接呼び出せる

ECMAScript の拡張性、強力な文字列処理ツール (「正規表現」を含む)、DOM バインディング、および Java とのブリッジを提供する機能により、ECMAScript は、exteNd Composer によって使用されるプログラミング構成要素と標準を拡張するための理想的な言語となります。

注記： ECMAScript に関する詳細は、次に示す European Computer Manufacturers Association (ECMA) の Web サイトで参照できます。**http://www.ecma.ch/**

ECMAScript が備える機能

スクリプトを使用すると、微調整したカスタム論理をアクションモデルに組み込むことができるだけでなく、データ操作の柔軟性も大きく増します。これは、Composer の ECMAScript 拡張には、DOM および XPath に関連するさまざまなオブジェクトとメソッドが用意されているためです。また、「拡張可能な」言語として、ユーザ定義カスタムオブジェクトを ECMAScript 内でその場で作成したり、Composer のコンポーネントやサービスで使用したりすることもできます。

ECMAScript の有用性は、メモリ内 DOM を扱う場合に特に明白です。Composer では、XML ドキュメントは、W3C DOM Level 2 仕様に基づいて、メモリ内オブジェクトとして作成されます。DOM-2 仕様では、DOM ツリーの内容への容易なアクセスを提供するさまざまなメソッドやプロパティとともに、ECMAScript バインディングが定義されています (**http://www.w3.org/TR/DOM-Level-2-Core/ecma-script-binding.html** を参照)。Composer の標準の DOM (Input、Input1、Input(n)、Temp、および Output) は、Composer 内で ECMAScript によって認識されるオブジェクトで、DOM に適用される、W3C で定義された ECMAScript 拡張に Composer からアクセスできます。

ECMAScript は、XPath などの他の式言語とのブリッジも提供します。Composer の場合は、これにより、Novell が提供するメソッド `XPath()` を DOM で使用して、ドキュメント構造内のさまざまな要素のアドレスを指定できます。

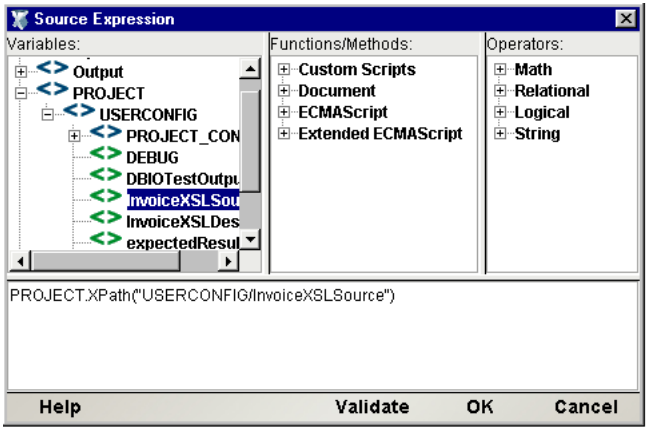
Composer の ECMAScript バインディングのもう 1 つの便利な面は、「ファイル I/O 拡張」(コア言語の一部ではありません) が含まれるところです。カスタムスクリプトを使用して、スクラッチファイルを簡単に読み書きしたり、情報をディスクに保存したり、他の一般的なファイルアクセスタスクを実行したりできます。

Composer の ECMAScript バインディングには、プログラムから JDBC を通じてデータベースにアクセスできるようにするデータベース拡張も含まれます。SQL ステートメントを文字列として渡し、接続を定義できる任意のデータベースに対して実行できます。

Composer のユーザインタフェースでのスクリプトの表示方法

このガイド全体で説明するように、Composer では、コンポーネントエディタユーザインタフェースの多くの部分から、ECMAScript へアクセスできるようになっています。最も一般的なアクセスの方法は、Expression Builder を使用する方法です。次のアイコンが表示されていれば、いつでも Expression Builder に移動できます。

このアイコンは、［Map Action］ダイアログボックスや［Connection Resource］ダイアログボックスなど、Composer の多くのダイアログボックスに表示されます。このアイコンをクリックすると、次のようなダイアログボックスが表示されます。



［Expression Builder］ダイアログボックスでは、最上部のペイン (すべてサイズを変更可能) に、使用可能なオブジェクト、メソッド、およびプロパティの選択リストと、ECMAScript ステートメントの作成に役立つロールオーバのツールヒントが提供されます。選択ツリーで任意の項目をダブルクリックすると、ウィンドウ下部の小さな編集ペインに、対応する ECMAScript ステートメントが表示されます。この例では、PROJECT に対応する DOM 選択ツリーを［Variables］ペインで開き、次の位置にあるノードをダブルクリックしています。

```
USERCONFIG/PROJECT_CONFIG/DESIGNER_EMULATION_MODE
```

PROJECT DOM 内のこのノードのコンテンツにアクセスできる ECMAScript 式が、自動的に編集ペインに表示されます。

ウィンドウのボタンバーには、［Validate］ボタンがあります。このボタンをクリックすると、ECMAScript インタプリタによって式の構文がリアルタイムでチェックされます。ECMAScript の構文に関する問題がある場合は、ただちにエラーダイアログボックスが表示され、必要に応じて、式を編集して再検証できます (ただし、検証はオプションです)。

注記：検証処理では、式は実行されず、構文のチェックのみが行われます。

動的なパラメータ値の式

通常、Composer の各アクションには、アクションの実行に使用する 1 つまたは複数のパラメータが必要です。可能な場合、これらのパラメータの代わりに ECMAScript 式を使用できます。静的な文字列、式、またはセミコロンで区切った一連の式を入力できます。式はランタイム時に評価されるため、実行時までパラメータ値の選択を延長できます。このようなパラメータ値の遅延バインディングは、入力値が事前にわからない場合に重要です。

例 : Send Mail アクションの Recipient パラメータに対して、静的な文字列をハードコードするよう選択できます。しかし、ECMAScript を使用して、着信した XML ドキュメント内にあるデータから電子メールアドレスを作成し、ランタイム時の情報に基づいてカスタマイズできる、データ駆動型の柔軟なアクションを作成することもできます。

Composer のほとんどのアクションでは、パラメータ値として ECMAScript 式を受け付けます。たいていの場合、XPath 式も受け付けられます。通常は、［XPath］および［Expression］というラベルの付いた 2 つのラジオボタンから選択できます。ECMAScript Expression Builder にアクセスするには、［Expression］ラジオボタンをオンにして、パラメータ値が表示されるテキストフィールドの横にある小さな［Expression Builder］アイコンをクリックします。

カスタムスクリプトライブラリ

ECMAScript は、カスタムスクリプトと呼ぶ一般的なリソースとしても Composer に統合されています。カスタムスクリプトリソースは、エディタ内にあるコマンドライン評価プログラムで実行およびデバッグできるカスタム ECMAScript 関数を作成するための編集環境を提供します。スクリプトエディタは、サンプル XML ドキュメント (DOM ツリー) へのアクセスも提供しており、カスタム Java コードとのブリッジとして機能するスクリプトを容易に作成できるように、Java クラスブラウザを備えています。カスタムスクリプトのライブラリをカスタムスクリプトリソースとして保存できます。ライブラリは、Composer のナビゲーションフレームのインスタンスペインに表示されます。また、カスタム関数をカスタムスクリプトリソースに組み込むと、[Expression Builder] ダイアログボックスのすべての選択リストに自動的に表示されます。

カスタムスクリプトリソースとスクリプトエディタの詳細については、222 ページ「カスタムスクリプトリソースについて」を参照してください。

Function アクション

ECMAScript の機能を Composer を表示するもう 1 つの方法は、すべてのコンポーネントエディタで使用可能なコアアクションの 1 つである Function アクションを使用する方法です。Function アクションをアクションモデルの任意の場所に挿入して、変数の初期化、カスタム関数の呼び出しなどを行うことができます。Function アクションの最も便利な使い方の 1 つは、デバッグ支援です。組み込みの `alert()` 関数を任意の文字列引数 (DOM ノードの内容など) とともに呼び出し、1 つのアクションまたは複数のアクションのブロックの前後でパラメータ値の内容を検査できます。`alert()` 関数により、該当する文字列を示すダイアログボックスが表示されます。

注記： `alert()` **の呼び出しを配備の前に無効にしておく必要があることに注意してください。これは完全に設計時メソッドであるため、サーバ環境には適用できません。**

Function アクションの作成および使用の方法については、149 ページ「Function アクション」を参照してください。

XPath から ECMAScript へのアクセス

一部のダイアログボックスフィールドには、XPath 式が必要です。しかし、一部のインスタンスでは、XPath 上での ECMAScript の大幅な速効性を好んだり、ロジックの要件を XPath が関連する組み込み関数の制限付きのセットに入らないことがあります。このような場合でも、ECMAScript を引き続き、次のような方法で使用できます。`userfunc` ネームスペース経由、XPath を必要とするすべてのフィールドで ECMAScript にアクセスできるようにする。

たとえば、`getTotal()` と呼ばれるカスタム ECMAScript 関数を独自に定義したとします。

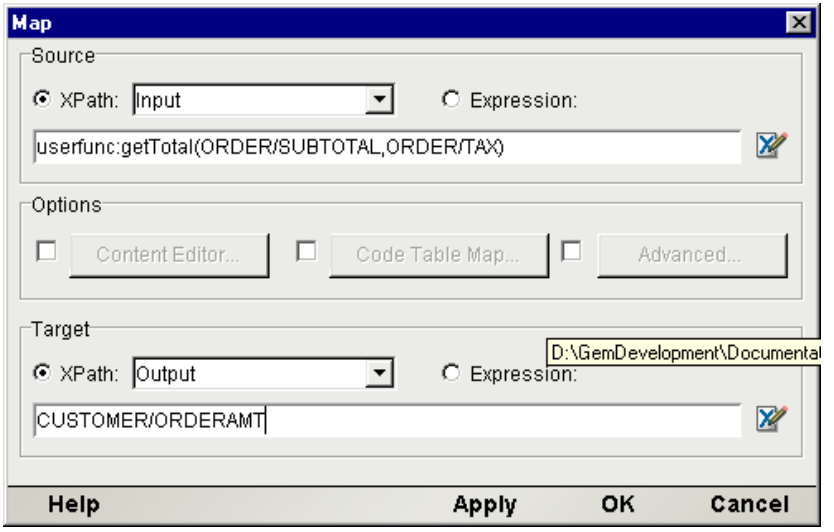
```
function getTotal(a,b) {
    return Number(a) + Number(b);
}
```

この関数はプロジェクトのカスタムスクリプトリソース、または関数 Action の中のいずれかで定義できます。

ORDER/SUBTOTAL および ORDER/TAX によりこの関数を XPath ステートメントから呼び出すことができると仮定します。XPath は次のように書きます。

```
userfunc:getTotal(ORDER/SUBTOTAL,ORDER/TAX)
```

この呼び出しを実行すると、XML の［Map］アクションダイアログは次のように表示されます。



XPath Access from ECMAScript

XPath から ECMAScript 関数に達することができるように、ノードオブジェクト、ノードデータ値などを ECMAScript 経由で取得することもできます。Composer は DOM 要素の操作のために種々の ECMAScript 拡張を備えています (後で詳しく説明します)。これらの拡張が最も頻繁に使用されるのは、`XPath()` メソッドで、単独の引数として XPath 形式のパス文字列を使用します。

```
var taxNode = Input.XPath("ORDER/TAX");
var taxAmt = taxNode.toString() * 1;
```

DOM ルートのペアレントである `XPath()` メソッド (この場合は Input) は、常にノード値ではなく、「ノードオブジェクト」を返します。ノードのデータ値を取得するためには、コア言語 ECMAScript メソッド `toString()` をそれに適用します。結果文字列の値が数値として使用される場合、ECMAScript の `Number()` コンストラクタにそれをラップするか、1 を乗じることにより、数値にキャストします (例を参照)。

注記： `XPath()` メソッドを使用するときの最も一般的なエラーは、「ノードオブジェクト」または「ノードリスト」を返すとき、データ値 (文字列、数値など) を返すことであると想定されることです。`toString()` を使用して、ノードオブジェクトからデータ値を取得してください。

カスタムスクリプト関数とスクリプト変数の範囲

カスタムスクリプトリソースに保存された関数は、アクションモデルのどこででも、プロジェクトのすべてのコンポーネントまたはサービスで使用できます (しかし、カスタム関数を作成した後で、コンポーネントで関数を使用できるようにするには、関連付けられたカスタムスクリプトリソースを保存しなければなりません)。

カスタムスクリプトリソース内のグローバル変数 (すなわち、カスタム関数の「外部」で宣言された変数) は、変数を使用するカスタムスクリプトリソース関数にのみ表示されます。言い換えると、変数を宣言する場合、**myFunctions** 内の唯一の関数である **myFunctions** というカスタムリソースの中の `myVariable` は、`myVariable` を表示し、使用することができます。

コンポーネントのアクションモデル内で宣言された変数は、コンポーネントにスコープされます。すなわち、(Function アクションの) アクションモデルの先頭で宣言された変数は、宣言の任意のアクションのダウンストリームに表示され、コンポーネントのライフタイムでは有効ですが、外部コンポーネントでは変数を使用できません。

変数の「サービス間」スコープを達成するには、274 ページ「Component (xObject)」という節で説明されている `putSessionValue()` と `getSessionValue()` メソッドを使用します。

ECMAScript の例の参照

任意のカスタム関数の本文の中では、DOM を ECMAScript オブジェクトとして取り扱い、`toString()` などの DOM を文字列テキストとして書き出すなどのオブジェクトに対する有効なメソッドを呼び出すことができます。

注記： カスタム関数に加えて、すべての標準的な組み込み ECMAScript オブジェクト (配列、ブール値、日付、関数、数値演算、オブジェクト、数値、RegExp、文字列、およびトップレベルグローバルオブジェクト)、およびメソッドとプロパティを式からアクセスすることができます。

Function アクションで使用するカスタム ECMAScript 式の例は次のとおりです。

```
var onHand = Input.XPath("INVENTORYSTATUS/ONHAND");
if (Number(onHand) < 10)
   Output.XPath("PRODUCTRESPONSE/INVENTORYSTATUS") =
        "Time to reorder";
```

このスクリプトでは、INVENTORYSTATUS/ONHAND 要素ノードの入力 DOM の値をチェックし、10 未満の場合、「Time to reorder」という文字列を要素 PRODUCTRESPONSE/INVENTORYSTATUS の出力 DOM にマップします。

ECMAScript の記述方法に従うと、ローカル変数 `onHand` の宣言にはデータ型ラベルを含める必要はありません。しかし、`onHand` から呼び出される値は「文字列」であることが頻繁にあります。その文字列を数字にキャストするためには、それにコア ECMAScript `Number()` 関数を適用する必要があります。その際に、条件の中で小なり演算子を使用することができます。

onHand が最終的に数値にキャストできない値 (空の文字列など) を割り当てられる可能性はもちろんあります。その場合、`Number()` は問題のある値 NaN を返し、条件式に例外を生成させることになります。例外を生成せずにこの可能性を取り扱うには、次のように処理することができます。

```
if ( !isNaN(Number(onHand) ) ) ?
    if (Number(onHand) < 10)
        [ code here ]
```

`isNaN()` メソッドは「数値であること」をチェックする ECMAScript 言語のコアメソッドです。

`isNaN()` 戦術の代わりに、*try/catch* ステートメントに例外コードをラップして、*catch* ブロックの例外を処理することができます。(*try/catch* の作成は、ECMAScript によりサポートされています。)

注記： その他の ECMAScript の例については、「Expressions」という Composer プロジェクトのサンプルに含まれている任意のカスタムスクリプトリソースを開いて (またはプロジェクトにインポートして) ください。

パフォーマンスについて

ECMAScript は、解釈済みの言語です。つまり、実行前に式のスクリプトの各行が解析され、同義の Java に変換される必要があります。これにより、コードに対して著しいオーバーヘッドがかかるため、純粋な Java に比べてスクリプトの実行速度が全体的に遅くなります。ECMAScript をコンポーネントやサービスで広く使用する前に、パフォーマンスに影響が生じる可能性があることも考慮に入れる必要があります。

次のガイドラインは、コンポーネントやサービスで最適のパフォーマンスを達成する上で役立ちます。

- Composer の組み込みアクションタイプの 1 つを使用してロジカルタスクを実行できる場合はいつでも、ロジックの大半を Java で実行できるように、通常のアクションの観点でタスクを実行する必要があります。
- アクションを使用してタスクを達成できない場合、(ECMAScript から呼び出せる) カスタム Java クラスを使用して達成できるかどうか考慮してください。
- タスクをアクションとして実行できない場合、またはスクリプト作成により提供される細かいコントロールが必要である場合は、ECMAScript を使用してください。

パフォーマンスを改善するには、常に適切な実装が重要です。つまり、正しいアルゴリズムを選択し、変数の再使用に注意することなどが考えられます。実行速度の遅い言語で書かれた適切なコードは、実行速度の速い不適切なコードで書かれたコードよりも効率がよい場合が多くあります。Java で書いたとしても、ECMAScript で書かれた同等のロジックよりも実行速度が速いとは限りません。これは Java にはコンストラクタ呼び出しチェーンなど、固有のオーバーヘッドの制約が含まれているためです (コンストラクタから他のオブジェクトから継承した Java オブジェクトを呼び出させた場合、すべての祖先オブジェクトのコンストラクタも呼び出されます)。

ECMAScript のコアオブジェクト (文字列、配列、日付など) には、データ操作、フォーマット、解析、ソート、文字列や配列の相互変換などの便利な多くの組み込みメソッドがあります。これらのメソッドは、インタープリタの中で高度に最適化された Java コードで実装されています。データ解析またはフォーマット機能を「独自にロールダウン」するより、これらのメソッドをできる限り使用すると有利です。たとえば、長い文字列を区切り文字が存在するたびに複数の下位文字列に分解するとします。文字列メソッドの `indexOf()` と `substring()` を使用するループを作成して、下位文字列に解析し、配列のスロットにそれらを割り当てることができます。しかし、次のようにできる場合、この方法は非常に効率の悪い方法であるといえます。

```
var myArrayOfSubstrings = bigString.split( delimiter );
```

ECMAScript の文字列メソッド `split()` は、指定した区切り文字の値に基づいて下位文字列の配列に文字列を分解します。これは、ネイティブ Java で実行され、インタープリタにスクリプトの 1 行のみを解読するように指示します。`indexOf()` と `substring()` を繰り返し呼び出すループで同じことをしようとすると、必要のない大きなインタープリタと関数呼び出しのオーバーヘッドを含むことになり、関係者のパフォーマンスに影響します。

ECMAScript の組み込みメソッドを有効に使用することは、パフォーマンスを改善する上で重要なことです。スクリプトを広く使用する場合は、パフォーマンス上の欠点を排除する上で役立つため、ECMAScript 言語の細かい点について時間をかけて学習してください。

ECMAScript のドキュメントリソース

- ECMAScript の詳細については、**http://www.ecma.ch/** を参照してください。
- ECMA-262 のドキュメントは標準的な Composer のインストールの際に提供されます。**Composer\Docs** ディレクトリにあります。

XPath とは

XPath は XML ドキュメント内のコンテンツのアドレス指定と検索のための構文を記述する W3C 標準です。XPath はまた、文字列、式、ブール値の操作のための簡易「表現言語」も備えているため、ユーザは XML データの構築と集積を細かくコントロールできます。

XPath は、ペアレントとチャイルドを持つ「ノード」のツリーとして、XML ドキュメントをモデルにしています。このノードには、要素ノード、属性ノード、およびテキストノードがあります。XPath は、ペアレントとチャイルドをスラッシュで分割する、一部のファイルシステムのディレクトリ / ファイルパス指定規則に類似するアドレス指定スキームを使用します。次のような一般的な構造が適用されます。

- / (スラッシュ) – ツリーのペアレント要素とチャイルド要素の間の区切り文字
- . (ドット) – ツリーの現在の位置
- .. (二重ドット) – ツリーのペアレントの位置

XPath アドレスは式とも呼ばれ、「コンテキスト」を参照して評価されます。Composer のコンテキストは通常、Input、Input1、Input(n)、Temp、または Output などの DOM です。Composer のコンテキストにはまた、XPath 式の単に別名または省略形であるグループ名を使用することもできます。

XPath の対象者

XPath は XML ドキュメントの処理に必要なほとんどすべてのタスクに対して Composer のすべてのユーザにより使用されることを目的としています。プログラマには、XPath のアドレス指定機能が不十分であると感じる場合があるでしょう。そのような場合、XML ドキュメントのアドレス指定のために代わりにより細かい DOM メソッド (277 ページ「DOM について」の説明を参照) を使用することができます。XPath と DOM の両方が不適切である場合は、XML ドキュメントを Java プログラムで直接処理することもできます。

XPath を使用するタイミング

XML ドキュメントで要素 (または属性) または要素 (属性) のグループを参照する場合はいつでも、XPath 式を使用できます。特に、XPath 式を Map アクションで頻繁に使用して、XML ドキュメント間のデータ転送のための入出力を指定します。また、Group 宣言 (XPath 式のツリーノードマッチングのリストを作成する) と Repeat for Element アクションで XPath を使用して、ドキュメントの要素の反復パターンに別名を作成することもできます。

作成するカスタム ECMAScript 式の XPath 式を使用することもできます。Composer は、ECMAScript 関数内の XPath 式を使用する `XPath()` という特別なブリッジメソッドを備えています。一般的な構文は次のとおりです。

```
Input.XPath("ROOT/PARENT/CHILD")
```

`XPath()` メソッドは DOM オブジェクトのペアレントです。この例では Input という名前が付けられています。また、`XPath()` の引数は文字列です。(リテラル、スタティック文字列、または文字列変数のいずれも使用できます。)

XPath が Composer に組み込まれる方法

XPath は Composer の基本的なアドレス指定メカニズムです。それは、Map、Repeat for Element、および Group (さらに、その他の多く) などのアクションのダイアログを経由して Composer に直接、組み込まれます。XPath は、これらのアクションで、「コンテキスト」と「式」の 2 つの部分として指定されます。XPath コンテキストは、「ベースアドレス」を表し、それに関連して式の残りの部分の評価が行われます。ほとんどの場合、これには XML ドキュメントのルート (ドキュメントオブジェクトなど) を表す DOM の単純な名前 (Input、Input1、Temp、Output など) が使用されます。

XPath の「式」の部分は上から下への順番で、処理されるノード (またはノードのリスト) に導く連続する要素を指定します。

XPath は、ドラッグアンドドロップを通じて作成された Map アクションによって Composer に自動的に作成されます。XPath 式は、有効な XPath ステートメントの選択リストを示す XPath Expression Builder を使用して Map アクションのダイアログで指定することができます。ダイアログで［XPath］ラジオボタンが選択されているときはいつでも、［Expression Builder］ボタン (次の図を参照) を押すことによって XPath Expression Builder にアクセスできます。


［Expression Builder］アイコン

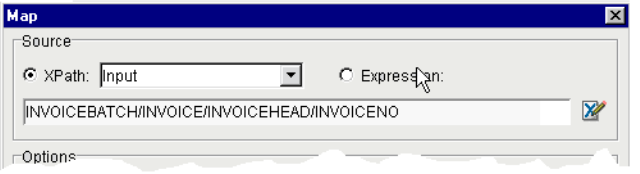
Composer は特別なメソッド **.XPath()** により EMCAScript に XPath を統合します。これは、ECMAScript 言語内の XPath 構文を使用して XML ドキュメントの部分をアドレス指定することができます。

Composer にはまた、XPath に関連した「グループ」の概念もあります。グループ名を宣言するとき、ドキュメントに複数ある XPath パターンに関連付けられます。このため、ツリーには 2 つの特別なノードのリストが表示されます。最初のリストはパターンに基づいた XML ドキュメントで見つかったそれぞれの固有ノードの値に対する 1 つのエントリを含むグループです。その後で、各グループのアクションを一度処理する Repeat for Group ループを設定することができます。

2 番目のリストは各グループ (固有または固有でない) の各メンバーに対して 1 つのエントリを含むグループ (詳細) です。その後で、各グループメンバーに対してアクションを一度処理する Repeat for Group ループを設定することができます。

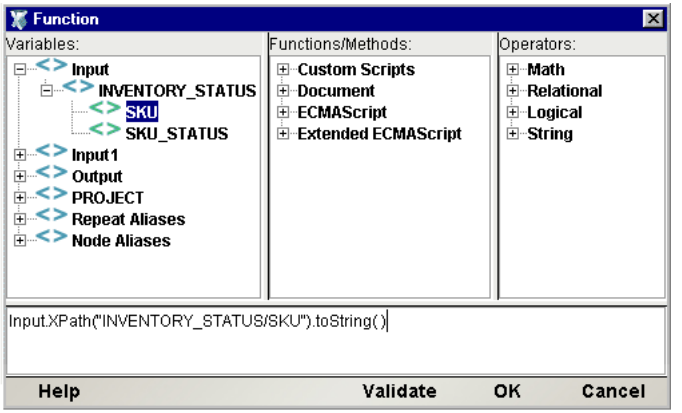
XPath の例の参照

Map アクションの XPath



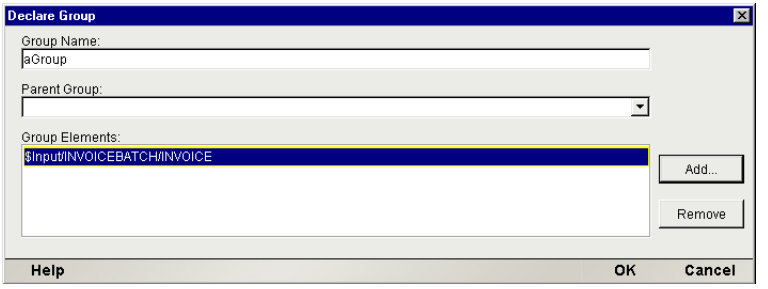
上記の例で、コンテキストは、「Input」DOM です。XPath 式は、「INVOICEBATCH/INVOICE/INVOICEHEAD/INVOICENO」で、INVOICE NO の要素の場所を INVOICEHEAD のチャイルドとして指定し、INVOICEHEAD は INVOICE のチャイルドで、INVOICE は INVOICEBATCH のチャイルドになります。

ECMAScript の XPath



上記の例で、コンテキストはメソッド「.XPath( )」を使用して INVENTORY_STATUS/SKU の場所を指定し、それをテキスト文字列 (ソース XML) に変換する XML ドキュメントオブジェクト「Input」です。その後、このテキスト文字列オブジェクトは ECMAScript メソッドを使用して操作できます。

グループの XPath



上記の例で、グループ名「sgSELLERNAME」は XPath 「$Input/INVOICEBATCH/INVOICE/SELLERNAME」の固有のデータ値に基づいてノードのリストを作成します。固有のノードのこのリストは、各グループの各メンバーの個々の値の代わりに、固有のグループ値に基づいてデータをマップする Repeat for Group ループアクションにより処理できます。

XPath 関数

XPath のリテラルアドレス指定機能を増大していく過程で、XPath の設計者は「表現言語」を仕様の中に作成し、洗練されたフィルタリング、内観、ノードセットの集積を許可します。XPath は事実上、文字列、数値、ブール値、およびノードセットの 4 つのデータ型をネイティブに認識する 2 ダース以上の便利な関数 (表 10-2 を参照) を事前定義しています。これらの関数を通常の XPath アドレス指定と合わせて使用すると、XML 開発者にとって XML データ操作のための強力なツールになります。

表 10-2 これらの関数のすべては、ロールオーバー (ツールヒント) ヘルプと共に Composer の Expression Builders にあります。

XPath 関数
ノードセット関数
number last()
number position()
number count(node-set)
node-set id(object)
string local-name(node-set)
string namespace-uri(node-set)
文字列関数
string name(node-set)
string string(object)
string concat(string, string, string*)
boolean starts-with(string, string)
boolean contains(string, string)
string substring-before(string, string)
string substring-after(string, string)
string substring(string, number, number)
number string-length(string)
string translate(string, string, string)
ブール関数
boolean boolean(object)

表 10-2 これらの関数のすべては、ロールオーバー (ツールヒント) ヘルプと共に Composer の Expression Builders にあります。

XPath 関数
boolean not(boolean)
boolean true()
boolean false()
boolean lang(string
数値関数
number number(object)
number sum(node-set) .
number floor(number)
number ceiling(number)
number round(number)

XPath 関数の詳細な説明は、このガイドの範囲を超えていますが (代わりに、**http://www.w3.org/TR/xpath** の完全な XPath の仕様書を参照してください)、XPath 表現言語の機能を示すいくつかの例を示します。

表 10-3

XPath 式	意味
//*	ドキュメントのすべてのノードから構成されるノードセット
count(//*)	ドキュメントのノード数
count(//*[contains(name(),'myNode')])	名前に (下位) 文字列「myNode」を含むドキュメントのノード数
name(//*)	すべてのノードのセットから、ドキュメントの最初のノード名をドキュメントの順序で検索します。(すなわち、ルートノード名を検索します。)
//*[name()='myNode']/@*	すべてのノードのセットから開始して、「myNode」という名前のノードを検索し、そのノードの下の最初の属性の値を、ノードの順序で取得します。

表 10-3

XPath 式	意味
name(//*[name()='myNode']/@*)	ノード「myNode」で見つかった最初の属性ノードの「名前」を取得します
concat(//*[name()='myNode4']/@*,' is what was found')	要素「myNode4」の下の最初の属性に保存されている「値」を「is what was found」という文字列と結合します。

より詳細な XPath の例については、出荷時に Composer に添付された「Action Examples」プロジェクトを参照してください。

XPath のドキュメントリソース

- XPath に関する詳細な情報は、Web サイト **http://www.w3.org/TR/xpath** を参照してください。
- W3C XML Path Language (XPath) のドキュメントもまた、インストール時に exteNd Composer の \Docs ディレクトリに保存されます。

XSL について

次の節では、XSL を使用するカスタムスクリプトの作成について説明します。

XSL とは

Extensible Stylesheet Language は、XML ドキュメントを他の種類のドキュメントに変換するための言語です。スタイルシート言語と同様、XSL はフォーマットの指定に XML ボキャブラリを含みます。

HTML と異なり、XML の要素名では直感的にわかる表記法は使用されていません。スタイルシート使用しないと、XML の配信プロセスでは区別できない文字列として以外に XML ドキュメントの内容をレンダリングする方法を決定できません。XSL は XML 構文を使用して理解可能なスタイルシートを作成するための包括的なモデルとボキャブラリを備えています。

XSL の機能は XSLT (XSL Transformations) で増大します。これは XML 構造を操作するためのプレゼンテーションを目的としない変換言語です。XSLT はソース XML ドキュメントまたはスタイルシートの作成者のいずれかから提供された文字列をフィルタリング、条件付処理、および生成するための要素を選択するために XPath により定義される表現言語を利用します。

XSL の対象者

XSL に興味をもつユーザは、Web マスター、eCommerce サイト作成者、ポータル作成者、および B to B 取引の一部として XML ドキュメントのグラフィック表示が必要な人々です。

XML ドキュメントを使用すると、設計者は XSL スタイルシートを使用して構造化したコンテンツを提示する方法を指定できます。つまり、Web ブラウザや PDA などのウィンドウ、またはカタログ、レポート、パンフレット、または書籍の物理的なページのセットなどのプレゼンテーション媒体でソースコンテンツのスタイルを指定し、レイアウトするか、ページ指定する、またはそれらのすべてを行う方法を指定できます。

XSL を使用するタイミング

XSL は XML 配信デバイスが人間が認識できる方法で XML を表示することを許可するように設計されています。XML データ交換は、Web ショッピング、データ監査、通知、またはデータのグラフィック表示を必要とするその他のユーザ操作を含む場合が頻繁にあります。要約すると、XML プレゼンテーションを使用可能にする必要があるたびに XSL を使用します。

XSL が Composer に組み込まれる方法

XSL はすべてのコンポーネントで使用可能な XSL Transaction Action を使用して Composer に組み込まれます。アクションを使用するためには、ソース DOM、XSL スタイルシート、および保存先 DOM (例、Temp または Output) のパラメータを指定する必要があります。説明については、次の節を参照してください。

Composer には、カスタムスクリプトまたは関数アクションに使用する特別な XSL メソッドもあります。

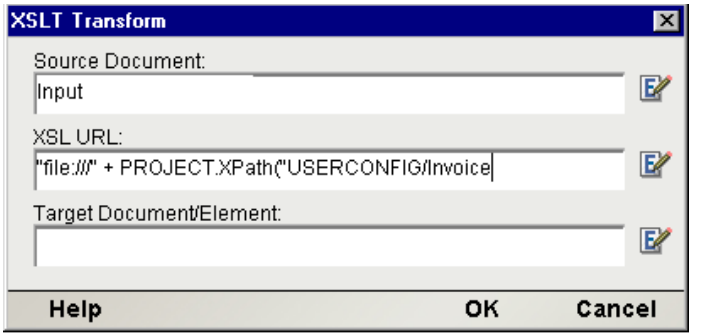
```
transformNodeViaDOM()
transformNodeToObject(,)
transformNodeViaXSLURL()
```

これらのメソッドの詳細については、次の API の説明を参照してください。

exteNd を使用して作成する Web サービスをまた、XSL を変換した XML を HTML に直接出力するようにセットアップすることもできます。処理命令を使用した HTML への配備に関する詳細は、特定のアプリケーションサーバプラットフォームの exteNd Composer Enterprise Server に関するガイドで、配備を説明した章を参照してください。

XSL の例を参照する

次に示す Process XSL アクションは、［XSL URL］フィールドで指定された XSL スタイルシートを使用して、入力 DOM を変換し、出力ドキュメントの「MyHTML」と呼ばれる XML 要素に結果を挿入します。



XSL の使用法に関する詳細は、Composer のインストールの「Action Examples」プロジェクトを参照してください。

XSL のリソース

- XSL の詳細については、次の Web サイトを参照してください。
  http://www.w3.org/TR/xsl
- XSL ドキュメント (W3C から) はまた、Composer のインストール階層の **\Docs** ディレクトリにも保存されます。
- Composer の作業例については、Composer のインストールの「Action Examples」プロジェクトを参照してください。

Novell Scripting 拡張について

Novell の ECMAScript の拡張は、xObject、DOM、その他の Composer オブジェクトを含む汎用スクリプト作成のための便利な方法のセットから構成されています。すべてのメソッドは、Expression Builder 選択リストに示されています。API の概要は次に示すとおりです。

汎用拡張

汎用拡張は、操作するオブジェクトの種類により分類され、次のものから構成されます。

ノード

`XML` — このプロパティは、DOM を示す文字列を返します。

`createXPath(XPathType asPattern)` — XPath パターンを作成します。

`getXML ()` — このプロパティは DOM を示す文字列を返します。

ドキュメント

`text` — このプロパティは、その下のすべてのテキストノード (コンテンツ) の連結された文字列を返します。

`setDTD(node RootElementName, object PublicName, object URL)` — ドキュメントの DTD ファイルをセットします。

`setValue(Object aValue)` — 渡されたオブジェクトからドキュメントの値を設定し、それが別のドキュメントにある場合は、このメソッドがチャイルドノード (要素と属性) をコピーします。渡されたオブジェクトがテキストの場合、DOM を作成するように解析されます。

`toString()` — DOM ドキュメントを XML 形式の文字列に変換します。

`transformNodeViaDOM(XSLDOM)` — XSLDOM に従ってドキュメントを変換し、文字列を返します。パラメータ XSLDOM は XSL スタイルシートであり、XML Interchange アクションによりコンポーネントに読み込まれている可能性があります。このメソッドは Map アクションのソースで使用することができるか、Server Framework クラス IGXSXSLProcessor にそれを呼び出します。

`transformNodeToObject(XSLDOM, OutputDOM)` — XSLDOM に従ってドキュメントを変換し、出力 DOM に結果を返します。パラメータ XSLDOM は XSL スタイルシートであり、XML Interchange アクションによりコンポーネントに読み込まれている可能性があります。パラメータ Output DOM は、結果のターゲット DOM です。このメソッドは、コンポーネントから Function アクションに挿入できます。また、カスタムスクリプトからは、すべての 3 つの DOM が揃ったときに一度使用することができるか、Server Framework クラス IGXSXSLProcessor を使用して Servlet で呼び出すことができます。

`transformNodeViaXSLURL(XSLURLLocation)` — XSLURLLocation に従ってドキュメントを変換し、文字列を返します。パラメータ XSLURLLocation は、XSL スタイルシートです。このメソッドは、Map アクションに挿入されるか、またはカスタムスクリプトからは、DOM が揃ったら一度使用することができるか、Server Framework クラス IGXSXSLProcessor を使用して Servlet で呼び出すことができます。

validate() — XPathTypes はタイプ NodeList、String、Number、または Boolean のいずれでもかまいません。通常、XPath パターンと一致するノードリストを返すために使用されます。リストから特定のノードを選択するためにはブラケットを使用します [例、Input.XPath("INVOICE/LINEITEM[1]") または Input.XPath("INVOICE/LINEITEM[last()]")]。属性によりノードを選択する場合は @ を使用します (例、Input.XPath("INVOICE/LINEITEM[@myattr]")。属性値により選択する場合 ...Input.XPath("INVOICE/LINEITEM[@myattr='abc']")。

Nodelist XPath(XPathType asPattern) — XPathType はタイプ NodeList、String、Number、または Boolean のいずれでもかまいません。通常、XPath パターンと一致するノードリストを返すために使用されます。リストから特定のノードを選択するためにはブラケットを使用します [例、Input.XPath("INVOICE/LINEITEM[1]") または Input.XPath("INVOICE/LINEITEM[last()]")]。属性によりノードを選択する場合は @ を使用します (例、Input.XPath("INVOICE/LINEITEM[@myattr]")。属性値により選択する場合 ...Input.XPath("INVOICE/LINEITEM[@myattr='abc']")。

要素

text — このプロパティは、その下のすべてのテキストノードの連結されたテキストを返します。

booleanValue() — 可能な場合、このオブジェクトのブール値 (true | false) を返します。

countOfElement(String propertyName) — 指定されたチャイルドの数を返します。

doubleValue() — 可能な場合、このオブジェクトの 2 倍の値を返します。

exists(String propertyName) — 指定されたチャイルドの存在をチェックします。

getIndex() — 現在のインデックスを返します。

getParent() — ペアレント要素を返します。

setIndex(int aiIndex) — この要素のイテレータインデックス値を設定します。

setText(String asText) — この要素に関連付けられたテキストノードを設定します。

setValue(Object aValue) — 渡されたオブジェクトから要素の値を設定します。別の要素である場合、このメソッドではチャイルドノード (要素と属性) もコピーされます。

toNumber() — テキストノードを取得し、それを数値に変換します。

toString() — この要素に関連付けられたテキストノードを取得します。

`Nodelist XPath(XPathType asPattern)` — XPathType はタイプ NodeList、String、Number、または Boolean のいずれでもかまいません。通常、XPath パターンと一致するノードリストを返すために使用されます。リストから特定のノードを選択するためにはブラケットを使用します [例、Input.XPath("INVOICE/LINEITEM[1]") または Input.XPath("INVOICE/LINEITEM[last()]")]。属性によりノードを選択する場合は @ を使用します (例、Input.XPath("INVOICE/LINEITEM[@myattr]")。属性値により選択する場合 ...Input.XPath("INVOICE/LINEITEM[@myattr='abc']")。

属性

`text` — このプロパティは属性のテキスト値を返します。

`setValue(Object aValue)` — 渡されたオブジェクトから属性の値を設定します。

`toString()` — この属性に関連付けられたテキストノードを取得します。

NodeList

`avg(NodeList)` — NodeList の平均値に等しい数値を返します。タイプ XPath の NodeList パラメータ。パラメータが指定されない場合、現在の NodeList/GroupName が使用されます。

`count(NodeList)` — NodeList のノードの数と等しい数値を返します。タイプ XPath のオプションの NodeList パラメータ。パラメータが指定されない場合 (通常の場合)、現在の NodeList/GroupName が使用されます。

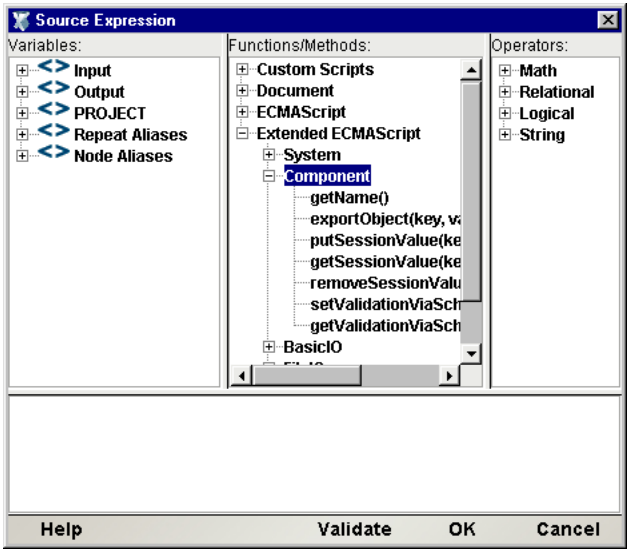
`min(NodeList)` — NodeList の最小値と等しい数値を返します。タイプ XPath の NodeList パラメータ。パラメータが指定されない場合、現在の NodeList/GroupName が使用されます。

`max(NodeList)` — NodeList の最大値と等しい数値を返します。タイプ XPath の NodeList パラメータ。パラメータが指定されない場合、現在の NodeList/GroupName が使用されます。

`sum(NodeList)` — NodeList の値の合計と等しい数値を返します。タイプ XPath の NodeList パラメータ。パラメータが指定されない場合、現在の NodeList/GroupName が使用されます。

`where(XPathType asPattern)` — XPath パターンと一致するノードの NodeList を取得します。

theComponent というオブジェクトは、Expression Builder を使用して各 Composer コンポーネントの中に示されます (Function アクション、Map アクション、またはアイコンが表示される他のダイアログの［Expression］アイコンをクリックすることにより、Expression Builder ウィンドウを開くことができます)。コンポーネントベースのメソッドは、Extended ECMAScript/Component の下の選択リストに示されます (次を参照)。



theComponent というコンポーネントには、次のメソッドがあります。

getName() — 現在実行中のコンポーネントの名前を返します。現在実行中のコンポーネントの名前を取得するには、次のように呼び出します。

```
theComponent.getName()
```

exportObject(key,value) — xObjects が使用できるように、グローバル変数の Java オブジェクトへの参照を保存できます。(保存しない場合、ユーザ変数は使用されるコンポーネントに限定されます。) たとえば、Java String オブジェクトを作成し、現在の exteNd セッションの他のコンポーネントで使用可能にする場合を想定します。

```
// create an instance of the String:
testString = new Packages.java.lang.String("hello");

// now publish it as a global, myExport:
theComponent.exportObject("myExport", testString)
```

この形式でエクスポートされたオブジェクトは、作成されたサービスインスタンスに限定されることを理解しておくことが重要です。サービスが終了すると、エクスポートされたオブジェクトはスコープ外になります。

変数のサービス間スコープセッションを達成するためは、次に示された `putSessionValue()` と `getSessionValue()` メソッドを使用します。

`putSessionValue(key,value)` — グローバル変数の Java オブジェクトへの参照を保存することを許可するため、同じサーブレットセッション (多くの HTTP ヒットにまたがることがあります) を実行中の他のサービスやコンポーネントから参照することができます。この形式でパブリッシュされたオブジェクトは、サーブレットレベルのスコープが設定されています。(セッションの寿命は、HTTP サーバセッションタイムアウトにより決まります。) 最初の引数は、パブリッシュされたオブジェクトの名前を表す文字列です。最初の引数は、パブリッシュされたオブジェクトに関連付ける名前を表す文字列です。2 番目の引数はオブジェクトです。(この構文は、先ほど示した `exportObject` の規則に従っています。)

注記： このメソッドは、EJB を使用して配備された Web サービスで使用される場合に例外を生成します。また、このメソッドは JMS サービスで使用することはできません。

`getSessionValue(key)` — `putSessionValue()` メソッドを使用して前もってパブリッシュされた Java オブジェクトへの参照を取得できます。このメソッドはキーと一致するオブジェクトが見つからない場合はヌルを返し、それ以外の場合はオブジェクトを返します。

注記： このメソッドは、EJB を使用して配備された Web サービスで使用される場合に例外を生成します。また、このメソッドは JMS サービスで使用することはできません。

`removeSessionValue(key)` — `putSessionValue()` メソッドを使用して前もってパブリッシュされた Java オブジェクトへの参照を破棄できます。

注記： このメソッドは、EJB を使用して配備された Web サービスで使用される場合に例外を生成します。また、このメソッドは JMS サービスで使用することはできません。

Connector 指定拡張

ここで説明されている以外の追加のカスタム ECMAScript オブジェクトとメソッドが、exteNd のほとんどの Connect 製品に関連して用意されています (たとえば、JMS 指定メソッドは、JMS Connector を使用してコンポーネントやサービスで使用するために用意されています)。

コネクタ指定の ECMAScript オブジェクトとメソッドの詳細については、該当する Connect のドキュメントを参照してください。

Novell 拡張を使用するタイミング

Composer の汎用拡張は、使用すると便利な場合、または XPath、DOM、または XSL の類似するメソッドよりも確かな場合、またはその両方の場合に使用します。

XML ファイルに関して反復するが、分散している共通データを要約するとき、Composer のグループ化または集積関連拡張の一部を使用する場合があります。たとえば、XML ファイルは組織のたった 7 部門で作成された 50 のランダムな請求書で送信されることがあります。Composer のグループ化機能とグループ指向のメソッドを使用して、50 の請求書を容易に部門ごとに整理し、各グループごとの「合計請求金額」を求めることができます。

Novell 拡張が Composer に組み込まれる方法

汎用拡張は ECMAScript に組み込まれ、Expression Builder の選択リストに他のオブジェクト、プロパティ、およびメソッドと共に表示されます。

Composer の Group アクションでは、グループの基礎を形成する XPath パターンを生成するための選択リストをクリックするだけでグループを指定します。Group または Group(Detail) の各メンバーに対してアクションのセットを処理する Repeat for Group アクションがあります。集計メソッドは［Map］ダイアログの ECMAScript Expression Builder で選択可能です。

ここに示されたアクションのタイプはすべて Connect コンポーネントタイプで使用できます。

グループ化の詳細

グループ化では、定義する共通の XPath パターンをもつノードの 2 つのリストに基づいて XML ドキュメントの情報を整理し、処理することができます。「Group」と呼ばれる最初のグループリストは、XPath パターンの XML ドキュメントをスキャニングすることにより作成されます。スキャンの後、1 つのグループが指定された XPath のそれぞれの固有の値に作成されます。その後、Repeat for Group アクションを使用して、グループに対して処理するアクションのセットを定義できます。Group 処理の共通のアクションには、グループに関する記述情報のマッピング、および数、合計、最大値と最小値、および平均を求める (各メソッドの詳細については、前の節を参照) ために集積拡張メソッドを使用することが含まれます。より高度なアクションには、グループに基づいてオリジナルの XML ドキュメントの構造変換も含まれます。

Group(Detail) という 2 番目のグループリストには、XPath パターンが互いにグループ化された共通の値と一致するノードが含まれます。その後、Repeat for Group アクションを使用して、各グループメンバーに対して処理するアクションのセットを定義できます。Group(Detail) 処理の共通アクションには、上位の要素と反対に階層内にある要素に従って XML ドキュメントを再構築することが含まれます。

拡張コードの例

ハードディスクの exteNd Composer のインストールの **\Samples** ディレクトリの「Action Examples」プロジェクトを参照してください。Novell 集積拡張で種々のグループ化、計算、および変換が可能な多くの Map コンポーネントがあります。

DOM について

DOM とは

Document Object Model は、プログラムおよびスクリプトが XML ドキュメントのコンテンツ、構造、およびスタイルをダイナミックにアクセスし、更新することができるインタフェースです。W3C の DOM (Document Object Model) は、ソフトウェアプログラム内の XML ドキュメント構造の標準的な内部表記であり、プログラマが容易に要素、属性、およびデータにアクセスし、その内容とスタイルを削除、追加、または編集できるようにすることを目的としています。

DOM の機能 とその重要な特長

DOM はオブジェクトとしてプログラム上で XML ドキュメントで作成および操作するための標準的なメソッドおよびプロパティのセットを定義します。要素、属性、テキスト、処理命令などを含む XML ドキュメントのすべてのパートを操作するためのメソッドを提供します。

DOM は、XML ドキュメントのノードのアドレス指定と検索のためのメソッドのセットも備えています。

DOM メソッドの対象者

DOM メソッドとプログラミングモデルは、DOM 操作に対して絶対的なコントロールを必要とするプロの開発者を対象としています。DOM メソッドを操作することにより、開発者は DOM を作成し操作するための初歩的な操作をコントロールできるようになります。たとえば、Composer の簡単な Map アクションは、10 行の ECMAScript/DOM 命令に変換することができます。

DOM メソッドを使用するタイミング

ECMAScript 機能と結合される基本 Composer アクションが XML ドキュメント処理のニーズを満たすとき、DOM メソッドを使用することができます。

DOM メソッドが Composer に組み込まれる方法

Composer DOM を操作するための DOM のメソッドとプロパティは、アクションのカスタムスクリプトエディタまたは ECMAScript Expression Builder でのみ使用できます。ダイアログで［Expression］ラジオボタンが選択されているときはいつでも、［Expression Builder］ボタン (次を参照) を押すことによって ECMAScript Expression Builder にアクセスできます。

DOM メソッドの例を参照する

- ディレクトリ ..\exteNd\Samples\CustomScripts の Database.es ファイルの「dbIOtoDOM()」というカスタムスクリプト関数を参照してください。
- すべての DOM メソッド / オブジェクトの完全な取り扱い方法については、サンプルプロジェクト「Expressions」を参照してください。

DOMS のドキュメントリソース

- DOM の詳細については、次の Web サイトを参照してください。
  http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/
- 完全なドキュメントオブジェクトモデル(DOM) レベル2仕様のドキュメントは、exteNd\Docs ディレクトリにあります。

Java の統合について

Java は単なるプログラミング言語ではありません。Java は同じソフトウェアが PC、UNIX ワークステーション、ワイヤレスデバイス、PDA、家電製品、および種々の埋め込み式システムなど、多くの種類のデバイスを動作させることができるように設計されたコンピューティングプラットフォームです。ネットワークを使用して、種々のデバイスを 1 つの稼動するアプリケーションに結合させる分散アプリケーションを作成することが可能です。

コンピューティングプラットフォームであることに加え、Java は Java 2 Enterprise Edition (J2EE) コンピューティングアーキテクチャを元に形成される強固でオブジェクト指向のコンピュータ言語です。これは、その適合性、頑強さ、プラットフォーム中立性および大企業で採用されているレコードトラック機能のため、IT 組織で多く採用されるようになりました。J2EE はまた、XML および Web サービスの分野の最新標準と深く結びついています。このことから、Java は理想的な企業型プログラミング言語であるといえます。

exteNd Composer での Java の使用

Java は、外部 Java オブジェクトへのダイレクトブリッジを備える ECMAScript スクリプト作成環境を通じて、Composer サービスに組み込まれています。Compower は、ドラッグアンドドロップ機能をもつカスタムスクリプトエディタの Java クラスブラウザを備え、Java オブジェクトを素早く組み込み、スクリプト内のコンストラクタ、プロパティ、およびメソッドを使用できるようにします。

Java を使用するタイミング

ほとんどの Composer ユーザは、Web サービスと XML の統合の目的を、カスタム Java クラスを使用することで Composer のネイティブ機能なしで達成することができます。しかし、Java オブジェクトを exteNd に組み込むことが望ましい場合があります。例 :

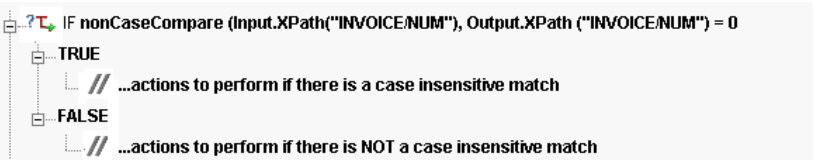
- 既存の Java 業務オブジェクトにアクセス (再使用) し、一部は他のコンピュータ環境やプログラムのデータをアクセスする場合
- 固有の Java 業務オブジェクトへの XML インタフェースを使用する場合
- Composer アクションまたは ECMAScript ではなく、Java でより効果的に処理される XML ドキュメントの複雑な操作を必要とする場合。
- ECMAScript を使用するコンセプトにプロトタイプを設定した後で、Java で同じコンセプトを実装しようとした場合 (再使用またはパフォーマンス、あるいはその両方の理由から)

Java 統合の例を参照する

XML アプリケーションの Java の簡単な使用例では、2 つの XML 要素でデータの大文字小文字を区別しない比較を実行します。この場合、次のように Java 文字列を使用してカスタムスクリプト関数を作成できます。

```
// Case Insensitive Compare, returns 0 if strings are equal,
non-zero if not…
function nonCaseCompare(string1,string2)
{
var s1 = new Packages.java.lang.String(string1);
var s2 = new Packages.java.lang.String(string2);
return s1.compareToIgnoreCase(s2);
}
```

その後、条件付きで異なるアクションを実行するために、Decision アクションの関数を使用します (図を参照)。



Java のドキュメントリソース

次の Web サイトでは、Java プラットフォームおよび Java プログラミング言語に関する詳細で的確な情報を見つけることができます。**http://java.sun.com**

11 共通タスクへのアクションの適用

アクションは、すべての Composer コンポーネントにおける作業の原子単位です。アクションでは、カスタムアプリケーションを使用可能にする制御フローと論理構成要素を統括します。一部のアクションは、他のアクションよりも頻繁に使用され、特定の設計パターンは、Composer を使って作成した Web サービスアプリケーションで何度も使用されます。この章では、Composer の使用時によく見られるアクションと設計パターンの一部について説明します。

この章での例について

exteNd Composer の設計側インストールにはサンプルプロジェクトが数個含まれており、そのうちの 1 つは ActionExamples.spf と呼ばれます。プロジェクトには、InvoiceBatch*.xml という名前のサンプルドキュメントが含まれています。この章の例は、XML Map コンポーネントへの入力として InvoiceBatch テンプレートを使用した場合に基づいています。

Composer の［File］メニューから ActionExamples.spf を開くと、例を参照しながら作業を進めることができます。ファイルは、次の場所に保存されています。

..\exteNd\Samples\ActionExamples\ActionExamples.spf

この章でのチュートリアルプロジェクトの場合と同様に、他のアクションモデルの例を参照することもできます。

要素とデータのマッピングについて

exteNd Composer の強力なツールの 1 つは、要素のマッピングです。異なる構造の DOM ツリー間で要素をマップし、XML ドキュメント間でデータを渡すことができます。

注記： Composer での基本的なマッピングの動作の概要については、157 ページ「マップのタイプ」の表を参照してください。

リーフ要素のマッピング

Composer で作成した要素のマッピングの多くは、2 つの DOM のリーフ要素 (端末ノード) 間で行われます。たとえば、入力 DOM の製品 SKU を出力 DOM の製品パーツ番号にマップする必要があるとします。この場合、サービスを実行すると、2 つの DOM 間で転送が行われ、入力 XML ドキュメントからの SKU が出力 XML ドキュメントのパーツ番号に書き込まれます。

2 つのリーフ要素をマッピングする 1 つの方法は、XML Map コンポーネントエディタの［Input］ペインと［Output］ペインでリーフ要素を選択し、Map アクションを追加することです。

注記：デフォルトでは、Composer の Map アクションによって「要素」データは転送されますが、属性データは転送されません。

➢ ［Action］メニューを使用してリーフ要素をマップする

1. コンポーネントを開きます。
2. アクションモデルペインで、Map アクションを配置する行を選択します。選択した行の下に新しい Map アクションが挿入されます。
3. ［Input］ペインで、マップするリーフ要素が表示されるまで入力 DOM を展開します。
4. リーフ要素を選択します。
5. ［Output］ペインで手順 3 と 4 を繰り返します。
6. ［**Action**］メニューから、［**New Action**］、［**Map**］の順に選択します。
7. ［Map］ダイアログボックスが表示されたら、［**OK**］をクリックします。入力要素から出力要素へのマッピングが自動的に作成されます。

次に説明するように、Composer の「ドラッグアンドドロップ」機能を使用して、入力リーフ要素を出力リーフ要素にマップすることもできます。

➢ ドラッグアンドドロップ機能を使用してリーフ要素をマップする

1. コンポーネントを開きます。
2. アクションモデルペインで、Map アクションを配置する行を選択します。選択した行の下に新しい Map アクションが挿入されます。
3. ［Input］ペインで、マップするリーフ要素が表示されるまで入力 DOM を展開します。
4. ［Output］ペインで、マップするリーフ要素が表示されるまで出力 DOM を展開します。
5. 入力リーフ要素を選択します。

6 マウスの左ボタンを押しながら、入力リーフ要素を出力リーフ要素の上にドラッグします。

7 マウスボタンを放します。Map アクションがアクションモデルペインに表示されます。

ペアレントとそのチャイルドのマッピング (ディープコピーのマッピング)

要素をマップする 2 番目の方法は、「ペアレント要素」をターゲット DOM にマップすることです。ペアレント要素がマップされると、そのチャイルド要素と属性もすべてマッピングに含まれます。たとえば、Line_Item というペアレント要素を選択し、Part_SKU、Part_Description、Part_Quanity、および Part_Cost を含むチャイルド要素がこの要素に含まれているとします。この場合に、Line_Item 要素を、PO_Line という名前の出力 DOM の要素にマップすると、結果として作成された Map アクションにより、元の分岐の構造は維持しながら、Line_Item 要素とそのチャイルド要素すべてが、出力の PO_Line 要素に転送されます。

注記 : Composer の Map アクションのデフォルトのマッピング動作は、無効にすることができます。

➢ **ペアレント要素とそのチャイルドをすべてマップする**

1 コンポーネントを開きます。

2 アクションモデルペインで、Map アクションを配置する行を選択します。選択した行の下に新しい Map アクションが挿入されます。

3 [Input] ペインで、マップするペアレント要素が表示されるまで入力 DOM を展開します。

4 ペアレント要素を選択します。

5 [Output] ペインで手順 3 と 4 を繰り返します。

6 [**Action**] メニューから、[**New Action**]、[**Map**] の順に選択します。

7 [Map] ダイアログボックスが表示されたら、[**OK**] をクリックします。

282 ページ「ドラッグアンドドロップ機能を使用してリーフ要素をマップする」で説明するように、Composer のドラッグアンドドロップ機能を使用して、入力ペアレント要素を出力ペアレント要素にマップすることもできます。

要素をマップする 3 番目の方法は、子孫要素なしでペアレント要素をマップすることです。基本的には、ハイレベルのデータをマップし、子孫要素のデータは無視します。たとえば、Invoice という要素をマップし、この要素に子孫要素が含まれている場合、出力 DOM では、Invoice 要素に関するデータのみを受信します。

➢ チャイルド要素なしでペアレント要素をマップする

注記： デフォルトの Map アクションの動作では、子孫要素が転送されるため、ECMAScript メソッドを作成して要素名に適用し、その要素のみをマップする必要があります。

1 コンポーネントを開きます。

2 アクションモデルペインで、Map アクションを配置する行を選択します。選択した行の下に新しい Map アクションが挿入されます。

3 ［Input］ペインで、マップするペアレント要素が表示されるまで入力 DOM を展開し、その要素を選択します。

4 ［Output］ペインで同じ操作を実行します。

5 ［**Action**］メニューから、［**New Action**］、［**Map**］の順に選択します。

6 ［Map］ダイアログボックスが表示されたら、［Source］で［Expression］を選択し、［Expression Builder］ボタンをクリックします。

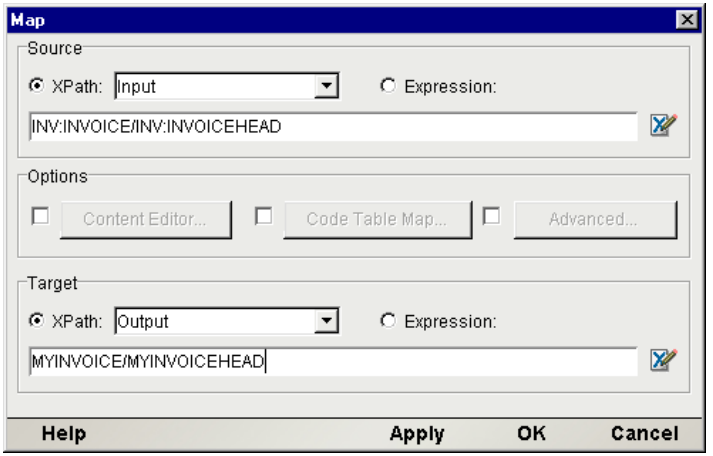
7 XPath フラグメントの前に、「`Input.XPath("`」と入力します。

8 XPath フラグメントを入力します。

9 XPath フラグメントの後ろに、「`").toString()`」と入力します。

10 ［**OK**］を 2 回クリックします。

チャイルド要素のないペアレント要素の Map アクションは、次の図のとおりです。



要素の変換

場合によっては、形式が異なる 2 つの要素をマップしなければならないことがあります。たとえば、入力 DOM の要素リーフは、4 つの数字と大文字 (1234CAT) から形成されており、出力要素リーフは、小文字と 6 つの数字 (cat001234) から形成されているような場合があります。

Composer には、DOM 間でデータを適切にマップできるよう、要素の形式を変換するための方法が 3 つあります。これらの 3 つの方法は、すべて Map アクションから使用できます。

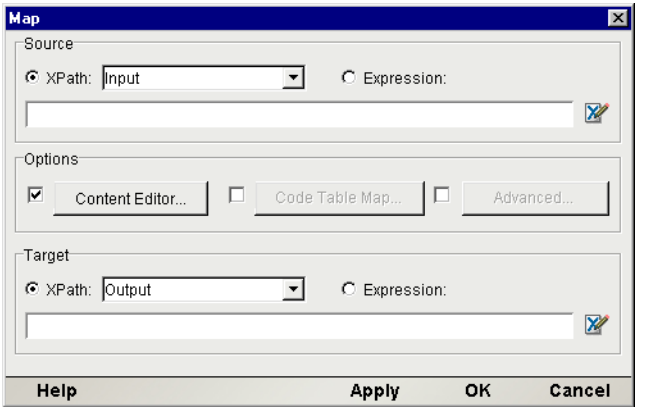
- コンテンツエディタ
- コードテーブルマップ
- 関数

コンテンツエディタを使用した要素の変換

コンテンツエディタを使用すると、入力要素の形式とコンテンツを変更して、出力要素に必要な形式とコンテンツに一致させることができます。コンテンツエディタでは、入力データを小さな部分にスライスしたり、それらの部分を互いに関連する異なる場所に移動したり、新しい部分を追加したり、一部の部分を省略したり、個々の部分に関数を適用したりできます。

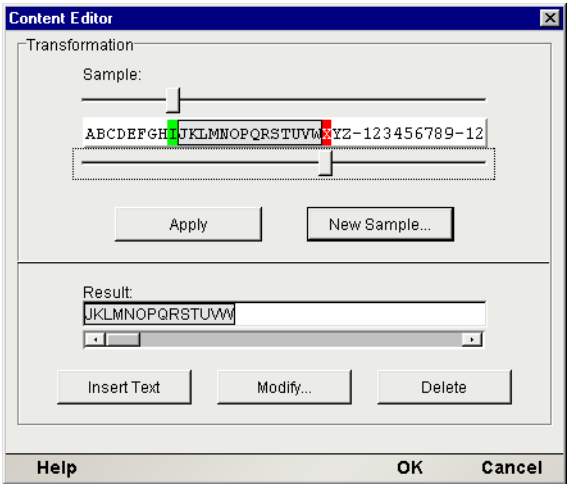
➢ コンテンツエディタにアクセスする

1. コンポーネントを開きます。
2. マップする 2 つの要素を異なる DOM からそれぞれ選択します。
3. ［**Action**］メニューから、［**New Action**］、［**Map**］の順に選択します。［Map］ダイアログボックスが表示されます。

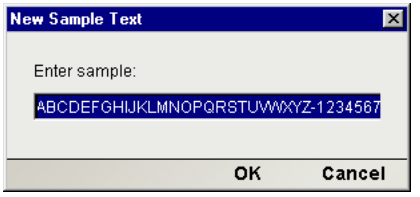


4 ［**Content Editor**］ボタンの横にあるチェックボックスをオンにします。これにより、［Content Editor］ボタンが有効になります。

5 ［**Content Editor**］ボタンをクリックします。Content Editor が表示されます。



6 または、［**New Sample**］ボタンをクリックして、同じ文字列を入力します。



ダイアログボックスを閉じます。

7 ［**Sample**］フィールドで、切り取りを開始する場所に上部スライダを移動し、切り取りを終了する場所に下部スライダを移動します。これらのスライダにより、入力データから下位文字列を切り取る方法が決定されます。

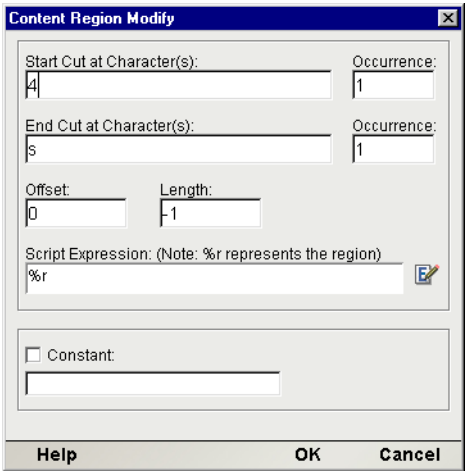
8 ［**Apply**］をクリックします。下位文字列は、個別のオブジェクトとして［**Result**］フィールドにコピーされます。

9 サンプルの各部分に対して、手順 6 から 8 を繰り返します (任意の順序で)。このようにして、元の入力の一部 (下位文字列) から新しい文字列を作成できます。

➢ ［Result］フィールドでオブジェクトの形式を変更する

1 オブジェクトを選択します。

2 ［**Modify**］をクリックします。［Content Region Modify］ダイアログボックスが表示されます。



注記：［**Start Cut at Character(s)**］フィールドには、文字列で切り取りを開始する場所にある文字が表示されます。最初の［**Occurrence**］フィールドには、切り取りが行われる場合が表示されます。前の図では、**T** の文字が初めて現れたときに、文字列の切り取りが開始されます。［**End Cut at Character(s)**］フィールドには、文字列で切り取りを終了する場所にある文字が表示されます。2 番目の［**Occurrence**］フィールドには、切り取りが行われる場合が表示されます。［Offset］フィールドには、オブジェクトが開始する元の文字列の最初から数えた文字数が表示されます。［Length］フィールドには、オブジェクトの長さが表示されます。

3 ［Script Expression］フィールドでは、ECMAScript Expression Builder をサポートしています。コンテンツエディタによって作成されたコンテンツ領域では、適用される Expression Builder の完全な機能を使用できます。

注記： %r は、関数を適用するコンテンツ領域を表すローカル変数です。たとえば、uCase() 関数をコンテンツ領域に適用する場合、uCase(%r) のようにスクリプト式を記述します。

4 オブジェクトを選択し、［**Constant**］チェックボックスをオンにしてから定数文字列を入力すると、そのオブジェクトに定数を割り当てることができます。

5 ［**OK**］をクリックして、形式の変更を適用します。

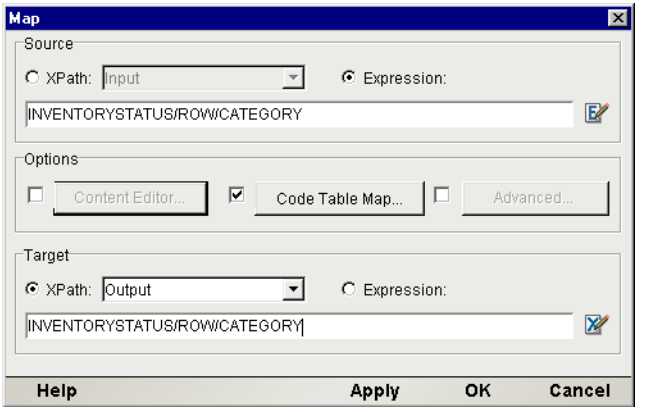
コンテンツエディタを使用した文字列形式のマッピングが完了した後、［**OK**］をクリックして変更内容を保存し、コンテンツエディタを閉じます。

コードテーブルを使用した要素の変換

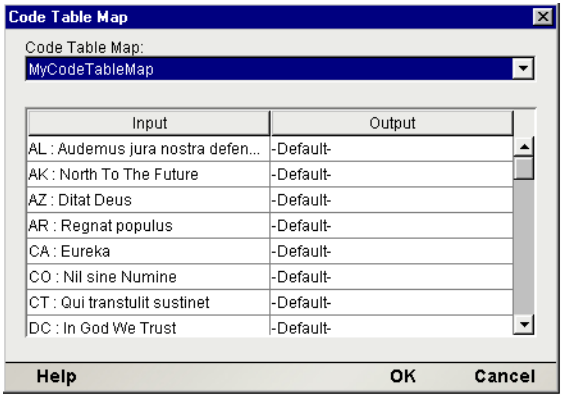
コードテーブルを使用してマップすると、入力 DOM で使用したあるコードセットを、出力 DOM で使用した別のコードセットに自動的に変換できます。コードテーブルを使用して要素を変換するには、コードテーブルとコードテーブルマップがすでに作成されている必要があります。

➢ コードテーブルを使用して要素を変換する

1. コンポーネントを開きます。
2. マップする 2 つの要素を選択します。
3. ［**Action**］メニューから、［**New Action**］、［**Map**］の順に選択します。［Map］ダイアログボックスが表示されます。



4. ［**Code Table**］ボタンの横にあるチェックボックスをオンにします。
5. ［**Code Table**］をクリックします。［Code Table Map］ダイアログボックスが表示されます。



6 コードテーブルマップを選択します。

7 ［**OK**］をクリックして、コードテーブルマップを割り当てます。

8 ［**OK**］をもう一度クリックして、Map アクションを保存します。

関数を使用した要素の変換

コンテンツエディタでは要素の形式構造を十分に変換できないという状況が生じる可能性があります。たとえば、日付形式 (例 : **5**/23) から月の数字を抽出して、月の名前 (**May** 23) に変換しなければならないことがあります。このような場合、ECMAScript および XPath カスタム関数を作成して、要素式に適用すると、カスタム変換を実行できます。

Composer には、サンプルカスタムスクリプト関数のライブラリが用意されており、次のカテゴリで編成されています。

- 文字列
- 数値演算
- ファイル
- 一般
- 日付
- データベース

関数のカテゴリは、**\exteNd\Samples\CustomScripts** サブディレクトリからインポートできます。

➢ ECMAScript カスタム関数を XPath 式に適用する

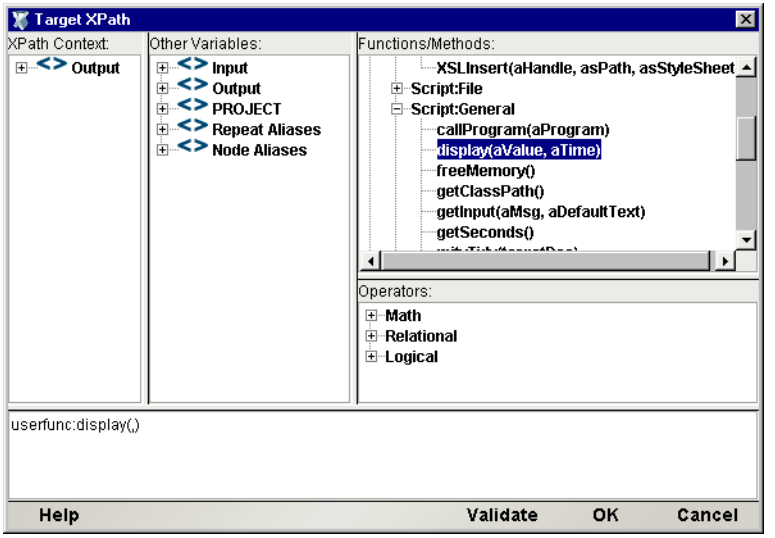
1 コンポーネントを開きます。

2 マップする入力要素と出力要素を選択します。

3 ［**Action**］メニューから、［**New Action**］、［**Map**］の順に選択します。［Map］ダイアログボックスが表示されます。

4 XPath Expression Builder を開きます。

5 選択リストを使用して、希望のカスタムスクリプト関数に移動し、ダブルクリックします。



6 必要に応じて式を編集し、文法的に正しくします。

7 ［**OK**］をクリックして、Map アクションを追加します。

注記： 関数を使用して Map アクション内で要素データを変換する場合は、関数の結果によって完全修飾 DOM 要素名が返されることを確認してください。

Map アクション外で要素のデータを変換する場合は、Function アクションを使用します (149 ページ「Function アクション」を参照)。

「Userfunc:」は、XPath 式で ECMAScript 関数を使用できるようにするブリッジ Novell 拡張メソッドです。また、XPath には、ノードセット、文字列、ブール、および数値として分類されるネイティブ関数の限定されたセットもあります。これらの関数では、userfunc: キーワードを使用する必要はありません。詳細については、exteNd/Docs/XPath ディレクトリに保存されている「XML パス言語 (XPath)」ガイドを参照してください。

アクションモデルでのループの使用

高度なアクションに関する章では、3 つの Repeat アクションと、アクションモデル内で反復処理を実行するためにRepeatアクションが使用される方法について学習します。この節では、Repeat アクションについて詳しく説明し、Repeat アクションを使用してデータ入力 DOM と出力 DOM の読み込み、マッピング、および書き込みを行う方法について示します。

Repeat アクションには、次の 3 種類のループがあります。

- Repeat for Element

- Repeat for Group
- Repeat While

Repeat for Element アクション

XML では、ドキュメントで要素のインスタンスを複数使用できます。インスタンスの数は、ドキュメントにより異なります。たとえば、請求書を含む XML ドキュメントを毎日受け取らなければならない場合があります。XML ドキュメントに含まれる請求書の数は、毎日異なります。XML ドキュメントに含まれる請求書のインスタンス数を把握していない場合、入力 XML ドキュメントの各請求書から出力 XML ドキュメントに請求書数を転送しようとすると、問題が生じます。しかし、Repeat for Element アクションを使用すれば、この問題が解決します。

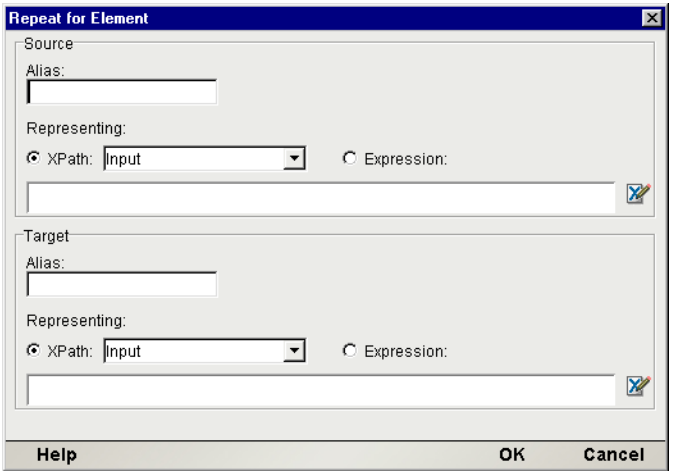
Repeat for Element アクションでは、複数回発生する要素をマークできます。その後、マークされた要素の各インスタンスに対して (そのようなインスタンスがなくなるまで) 1 つまたは複数のアクションを実行する処理ループを設定します。前の例では、請求書数を転送する Map アクションが処理ループに 1 つ含まれている可能性があります。

Repeat for Element 処理ループを使用すると、複数のアクションを処理できます。最も単純な場合、Repeat ループには、入力 DOM から出力 DOM に現在のインスタンスの値を転送する Map アクションが 1 つだけ含まれます。また、処理ループでは、現在の値を転送する Map アクションと、転送ごとに監査ログを作成してファイルに書き込む Log アクションなど、複数のアクションを設定することもできます。

Repeat for Element アクションを追加する最初の手順は、繰り返し処理を実行する場所にアクションモデルペイン内でカーソルを置くことです。

➢ Repeat for Element アクションを追加する

1. コンポーネントを開きます。
2. アクションモデルペインで、Repeat For Element アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
3. 入力 DOM で、繰り返す要素の最初のインスタンスを選択します。
4. コンテキストメニューを使用して、［**Repeat for Element**］を選択します。［Repeat for Element］ダイアログボックスが表示されます。



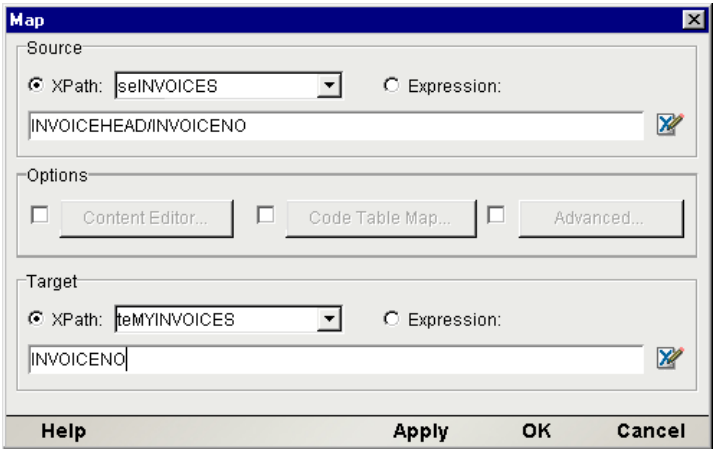
5 ［**Source**］の要素の別名を入力します。

6 デフォルトの XPath を受け入れるか、または［**Expression**］を選択して有効な式を入力します。

7 ［**Target**］に対して、手順 4 から 6 を繰り返します。

8 ［**OK**］をクリックします。

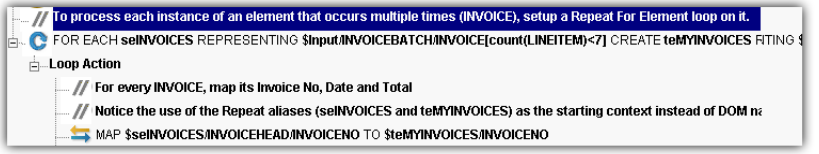
Repeat For Element アクションを作成すると、ループ内に Map (またはその他の) アクションを追加できます。たとえば、入力 XML ドキュメントから出力 XML ドキュメントに請求書数要素を単に転送するには、次の図に示すように Map アクションを定義します。



ここでは、XPath コンテキストとして繰り返し別名が使用されています。別名は Repeat アクションで定義され、実際の DOM 名とパスに解決されます。

［**Source**］フィールドでは、入力 DOM (seINVOICE/INVOICEHEAD/INVOICENO) の場所からのデータが出力 DOM (teINVOICE/INVOICENO) の場所に転送されるよう指定します。

Repeat for Element アクションと Map アクションは、アクションモデルペインに表示される必要があります (次の図を参照)。



Repeat for Group アクション

受信した XML ドキュメントの形式は、常にビジネスプロセスの条件を満たす形式であるとは限りません。たとえば、XML ドキュメントに異なる販売者からの請求書が含まれる場合があります。データは個々の請求書として受信されますが、B2B トランザクションのコンテキストでは、データを要約してマネージャにサマリデータを送信すると同時に、会計支払部署に請求書データを送信しなければならないことがあります。

Repeat For Group アクションでは、データを再作成して、データを集約計算するためのフレームワークを確立できます。グループ化すると、入力 DOM で繰り返し要素を選択し、その繰り返し要素のすべてのインスタンス (兄弟) 全体で固有な値に基づき、より少数の要素を作成できます。これにより、請求書全体 (一部の請求書では販売者値が同じ) で複数の販売者要素を使用する代わりに、出力 DOM の固有な販売者値それぞれに対して 1 つの要素を使用することになります。

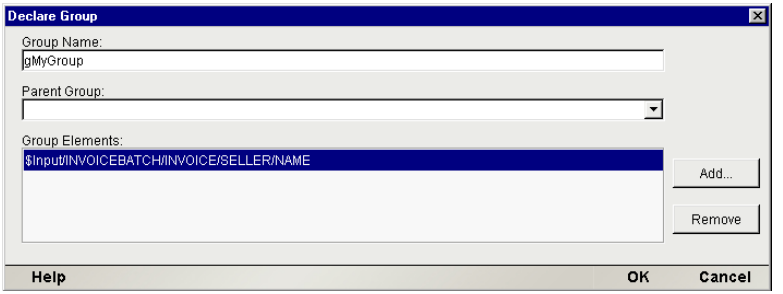
Repeat For Group アクションでは、グループの固有な値それぞれに対して実行する処理ループを設定します。販売者あたりに 1 つの要素を設定すると、処理ループに Map アクションを追加して、各販売者における請求書の数を計算できます。また、各販売者の下に個々の請求書数をリストすることもできます。Repeat For Group 処理ループを Map コマンドと結合させると、元の XML ドキュメントとは構造とデータが異なる新しい XML ドキュメントを作成できます。

Repeat For Group アクションを作成するには、次の 3 つのタスクを完了する必要があります。

- グループを作成して、繰り返し要素を識別する。
- Repeat For Group アクションを作成する。
- ループ内に Map アクションを作成する。

➢ グループを作成する

1. 入力 DOM で繰り返す要素を選択します。
2. マウスを右クリックして、［**Declare Group**］を選択します。［Declare Group Info］ダイアログボックスが表示されます。

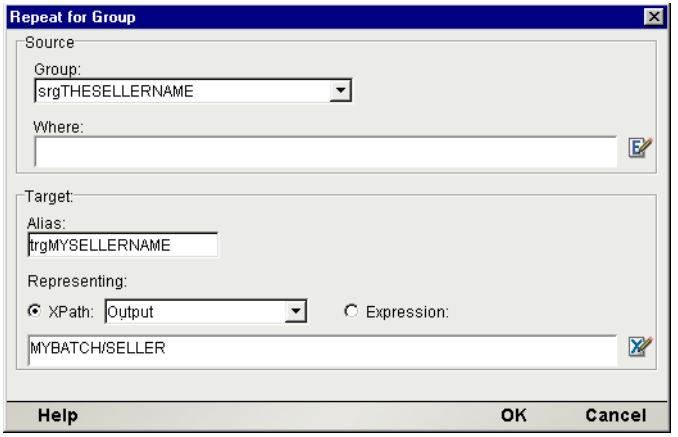


3. ［**Group Name**］フィールドに、Map アクションでグループの参照に使用する別名を入力します。
4. 複数のグループレベルを作成する場合は、［**Parent Group**］フィールドでグループを選択します。
5. ［**Group Elements/Attributes**］フィールドで、選択した要素の完全な名前を指定します。希望する場合は、このリストに別の要素を追加して、異なる要素の 2 つまたはそれ以上の値を連結させてグループを作成することもできます。
6. ［**OK**］をクリックして、グループを保存します。**Declare Group** 行がアクションモデルに表示されます。

入力 DOM 要素に基づいてグループを作成すると、Repeat For Group アクションを作成できます。

➢ Repeat for Group アクションを作成する

1. アクションモデルで、Repeat for Group アクションを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。
2. ［**Action**］メニューから、［**New Action/Advanced**］、［**Repeat for Group**］の順に選択します。［Repeat for Group］ダイアログボックスが表示されます。



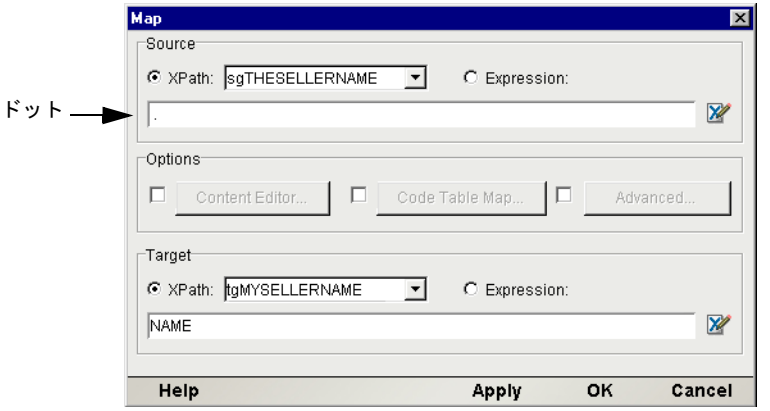
3 ［Source］フィールドで、処理ループの基礎を指定します。ループの基礎として使用するグループを選択します。

4 オプションの［**Where Script Expression**］フィールドでは、グループ処理から一部の繰り返し要素を選択的に除外できます。式を入力するか、または［**Expression Builder**］ボタンをクリックして、グループに含める要素を決定する ECMAScript 式を記述します。

5 オプションの［Target］フィールドでは、Repeat for Group アクション内でマップされたデータを配置する、出力 DOM 内での位置を指定できます。［Target］フィールドで別名を入力し、DOM を選択して XPath を指定します。この別名は、ループ内の Map アクションに対するターゲットコンテキストとして使用されます。

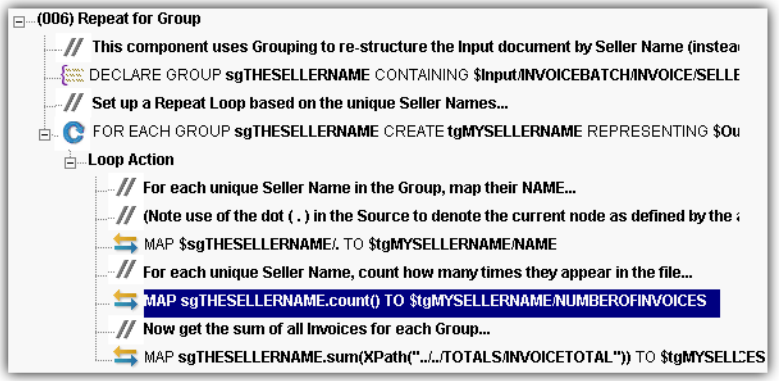
6 ［**OK**］をクリックして、Repeat for Group アクションを完了します。

Repeat for Group アクションを作成すると、ループ内に 1 つまたは複数の Map アクションを追加できます。入力 DOM 要素および出力 DOM 要素としてグループを使用する Map アクションは、次の図のとおりです。



ここでは、ドットが使用されています。これは、コンテキスト「sgTHESELLERNAME」で解決される現在の場所 (以前に、アクションモデルの Declare Group アクションで定義しました) を示しています。

Repeat for Group アクションと Map アクションは、アクションモデルペインに表示される必要があります (次の図を参照)。



Repeat While アクション

Repeat While アクションでは、定義する任意の条件に基づいて処理ループが作成されます。これによって、Repeat for Element アクションや Repeat for Group アクションの実行時とは異なる柔軟性が、繰り返しループの作成時に提供されます。この 2 つのアクションでは、両方ともループはドキュメントまたは DOM のデータに基づきます。しかし、Repeat While アクションを使用すると、処理ループを有効な XPath や ECMAScript 式に基づかせることができます。

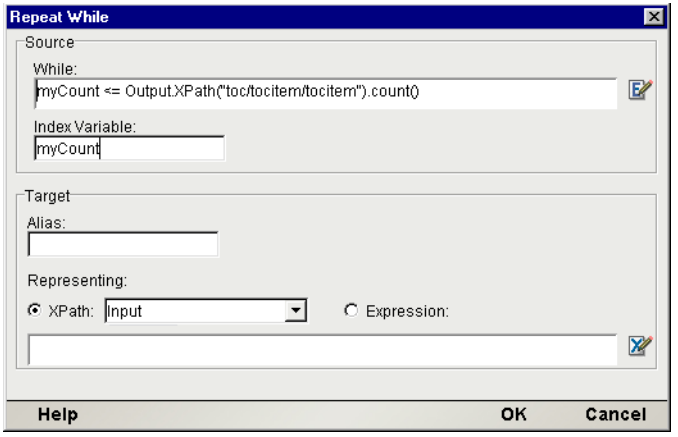
たとえば、ループの実行を、システムクロックを確認してループから抜け出すときを決定する ECMAScript 式に基づかせることができます。また、別の例では、ディレクトリ内でのファイルの有無にループを基づかせることができます。この場合、ループ内のアクションによってファイルが処理され、ファイルがなくなった場合にのみループが終了します。

Repeat While アクションを作成するには、次のタスクを実行する必要があります。

- アクションモデルペインで、Repeat While アクションを配置する場所を選択します。
- アクションを作成します。
- Repeat While アクション内で、1 つまたは複数のアクションを作成します。

➢ Repeat While アクションを追加する

1. アクションモデルで、Repeat While 処理ループを配置する行を選択します。選択した行の下にループが挿入されます。
2. ［**Action**］メニューから、［**New Action/Advanced**］、［**Repeat While**］の順に選択します。［Repeat While］ダイアログボックスが表示されます。

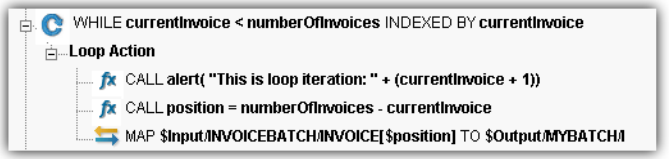


3. ［**While Script Expression**］フィールドは、ECMAScript 式を入力する場所です。false と評価された場合、ループの実行は停止します。［**Expression Builder**］ボタンをクリックして式を入力したり、事前に記述された式のリストから式を選択したりすることもできます。
4. ［**Repeat Index Script Variable**］フィールドでは、ループカウンタの名前を作成できます。このカウンタは、ループが実行されるたびに増分します。［**While Script Expression**］で値を取得し、ループ処理をさらに制御できます。

5 オプションとして、[**Target**] に情報を入力できます。別名を入力し、[**XPath**] と DOM 要素を選択するか、または [**Expression**] を選択して有効な式を入力します。また、[**Expression Builder**] ボタンをクリックして、式を作成することもできます。

6 [**OK**] をクリックして、Repeat While 処理ループを完了します。

Repeat While 処理ループを作成すると、ループ内に 1 つまたは複数の Map アクションを追加できます。2 つの Map アクションを使用した Repeat While ループは、次の図のとおりです。



集約計算の実行

集約計算には、アクション例プロジェクトの [008] 集約計算というコンポーネント内にある次の例が含まれます。

- 合計の計算
- 最大合計の検索
- 最大合計に対する特定の一致の検索

合計の計算

請求書を処理するコンポーネントがあり、請求書全体で全品目の合計 (課税前) を計算したいとします。

これには、XPath 構文を使用した単純な ECMAScript 式を使用します。まず、Map アクションを作成し、[Source] の [Expression] ラジオボタンをオンにします。そして、次の ECMAScript 式を入力します。

```
$Input.XPath("//LINETOTAL").sum()
```

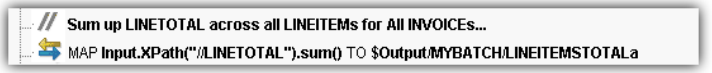
[Target] で [Output] を選択し、結果を受け取る XPath を指定します。

```
MYINVOICEBATCH/LINEITEMTOTALa.
```

[Source] の式では、親子関係に関係なく、LINETOTAL というラベルの入力ノードをすべて選択するために XPath の // パターンシンボルが使用され、その後、Novell 集約メソッドの `sum()` が適用されます。

注記： XML テンプレートは、[Output] に対して指定されていないため、動的に作成されます (つまり、Map アクションでは、「Map To」コントロールで指定した要素が検索されない場合に、この要素が作成されます)。

アクションの例は、次のとおりです。



最大合計の検索

請求書を処理するコンポーネントがあり、請求書金額の最大合計を検索したいとします。

複数の INVOICE 間で要素の最大を検索するには、[Source] の仕様で「max」メソッドを指定できます。これにより、`max()` 関数のコンテキストが確立されます。その後、最大を検索する要素が「TOTALS.INVOICETOTAL」に存在する DOM ツリーでのポイントまで、仕様を続行できます。

アクションの例は、次のとおりです。

最大合計に対する特定の一致の検索

前の請求書の例の続きとして、最大合計に一致する請求書を 1 つ選択したいとします。

すべての INVOICE を表示して、その中から 1 つだけ選択するには、[Source] の仕様で「where」メソッドを指定できます (where メソッドでは、各 INVOICE を処理することを意味します)。仕様は、各 INVOICE の TOTALS\INVOICETOTAL と、すべての TOTALS.INVOICETOTAL の「最大」を比較して続行されます。最大が検索された後、各 INVOICE の値に対して比較されます。一致が見つかると、仕様が続行され、INVOICENO が取得されます。

アクションの例は、次のとおりです。

12 アニメーションツールを使用したテスト

アニメーションツールとは

Composer のサービスエディタおよびコンポーネントエディタには、サービスやコンポーネント内でアクションをテストしたりトラブルシューティングしたりできるようにするアニメーションツールがあります。このため、サービスやコンポーネントのアクションモデルを通して段階的に作業し、各アクションの結果を監視することができます。また、アクションが失敗した場合に通知を受け取るだけでなく、データが計画どおりに動作したかどうかも確認できます。

アクションモデルの特定の 1 セクションをテストするには、アニメーションツールを使用して 1 つまたは複数のブレークポイントを設定できます。このようにすると、適切に動作するアクションを迅速に調べて、問題の発生したアクションでテストを停止した後、問題のあるアクションを 1 つずつ処理することができます。

アニメーションツールの使用

アニメーションツールは、サービスエディタとコンポーネントエディタのアクションモデルツールバーから使用できます。アクションモデルツールバーの他に、エディタには、［**Animate**］メニューのメニューオプションや、対応するキーボードコマンドもあります。さまざまなツールとその機能については、次の表で説明します。

表 12-1

アニメーションツールバーボタン	説明
	［Start Animation］— このボタンをクリックすると、アニメーション処理が開始します。オプションとして、［**Animate**］> ［**Start Animation**］の順に選択するか、または **<F5>** キーを押すこともできます。

アニメーションツールバーボタン	説明
	［End Animation］— このボタンをクリックすると、アニメーション処理が停止します。オプションとして、［**Animate**］>［**End Animation**］の順に選択するか、または <**Shift**>+<**F5**> キーを押すこともできます。
	［Step Into］— このボタンをクリックすると、現在選択されているアクションが実行され、次のアクションが選択されます。Component、Repeat、Decision、または Try/On Error アクションが現在選択されているアクションの場合、次に選択されたアクションがこれらのアクションの詳細になります。Repeat Loop アクションの場合、［Step Into］を押すと、ループの各アクションが実行され、各ループを通して反復します。Decision アクションの場合は、［Step Into］を押すと、True または False 分岐で次のアクションが処理されます。Try/On Error アクションの場合は、［Step Into］を押すと、実行内と、おそらく On Error 分岐で、次のアクションが処理されます。Component アクションの場合は、別のウィンドウが開き、そのコンポーネントに「ステップイン」します。オプションとして、［**Animate**］>［**Step Into**］の順に選択するか、または <**F7**> キーを押すこともできます。
	［Step Over］— このボタンをクリックすると、現在選択されているアクションが実行され、次のアクションが選択されます。［Step Into］ボタンとは異なり、このボタンをクリックしても、Component、Repeat、Decision、または Try/On Error アクションの詳細は選択されたり実行されたりしません。Component アクションの場合は、別のサービスやコンポーネントが呼び出されても、別のウィンドウは開きません。単に呼び出しが実行され、次のアクションに進むだけなので、つまりは、呼び出されたサービスやコンポーネント、あるいは Repeat、Decision、または Try/On Error アクションを基本的に「またぐ」ことになります。オプションとして、［**Animate**］>［**Step Over**］の順に選択するか、または <**F8**> キーを押すこともできます。
	［Toggle Breakpoint］— このボタンをクリックすると、アクションモデルで選択されているアクションがブレークポイントとして設定されます。ブレークポイントは、複数設定できます。オプションとして、［**Animate**］>［**Toggle Breakpoint**］の順に選択するか、または <**F2**> キーを押すこともできます。
	［Run To Breakpoint/End］— このボタンをクリックすると、アクションモデルの次のブレークポイントまたは最後のブレークポイントにアニメーションが実行されます。オプションとして、［**Animate**］>［**Run To Breakpoint/End**］の順に選択するか、または <**F9**> キーを押すこともできます。
	［Pause Animation］— このボタンをクリックすると、アニメーションが停止します。オプションとして、［**Animate**］>［**Pause Animation**］の順に選択するか、または <**F6**> キーを押すこともできます。

この章での例について

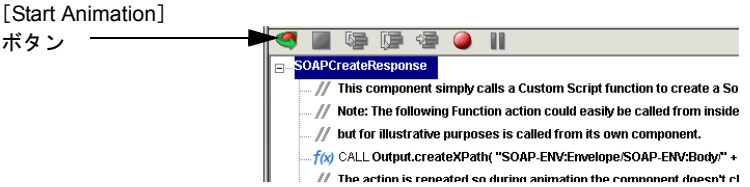
この章での例は、exteNd Composer に含まれるチュートリアルプロジェクト (Tutorial.spf) に基づいています。このプロジェクトを開いたら、例を参照しながら作業を進めることができます。

アニメーションの開始

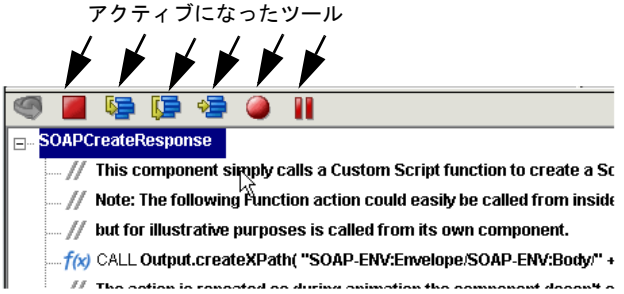
サービスまたはコンポーネントを初めて開いたときに使用できるのは、[Start Animation] ツールと [Toggle Breakpoint] ツールのみです。[Start Animation] ボタンをクリックすると、残りのアクションアニメーションツールが使用できるようになります。

➢ アニメーションを開始する

1 サービスまたはコンポーネントを開きます。サービスまたはコンポーネントが該当するエディタに表示されます。



2 アクションペインのツールバーで [**Start Animation**] ボタンをクリックするか、またはキーボードの <**F5**> キーを押します。現在では淡色表示されている [Start Animation] ボタンを除き、アクションツールバーのツールがすべてアクティブになります。



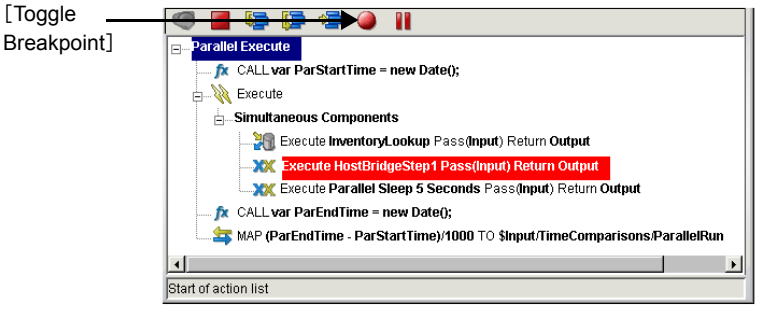
3 次のセクションの指示に従って、希望のアニメーションプロセスを実行します。

ブレークポイントの切り替え

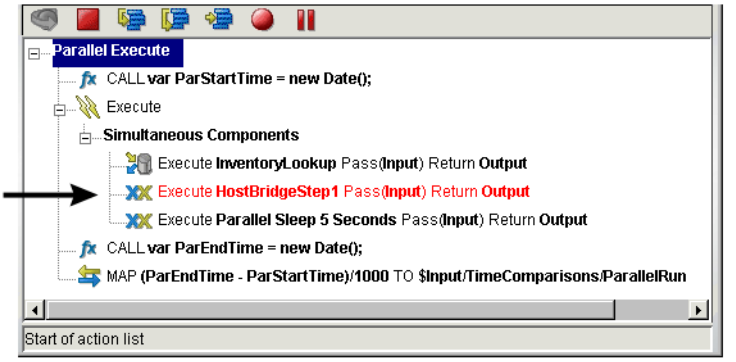
Toggle Breakpoint ツールを使用すると、プロセスを停止する場所となるブレークポイントをアクションモデルで設定できます。これは、適切に動作する長いセクションがある、時間のかかるアクションモデルである場合に特に便利です。問題が発生する各アクションの開始にブレークポイントを設定した後、アクションを 1 つずつトラブルシューティングすることができます。

➢ ブレークポイントを切り替える

1. サービスまたはコンポーネントを開きます。サービスまたはコンポーネントが該当するエディタに表示されます。
2. 必要に応じてアクションペインを調節し、そのコンテンツを表示します。
3. アクションペインで、ブレークポイントを配置するアクションを選択します。これは、アニメーションが停止する場所です。
4. アクションモデルツールバーで［**Toggle Breakpoint**］ボタンをクリックするか、またはキーボードの <**F2**> キーを押します。選択したアクションのバックグラウンドは赤色に、テキストは白色に変わります。



5. 必要に応じて手順 3 と 4 を繰り返し、追加のブレークポイントを選択します。
6. アクションモデルツールバーの［**Start Animation**］ボタンをクリックします (ボタンがグレー表示されていない場合)。［Start Animation］ボタンがグレー表示されている場合は、別のアニメーションツールを選択して、現在のブレークポイントからアニメーションプロセスを完了できます。［Start Animation］をクリックすると、次のように変更されます。
   - ［Start Animation］ボタンが非アクティブになる。
   - アクションモデルツールバーの残りのボタンがアクティブになる。
   - カーソルがアクションモデルの開始に移動し、「ProductInquiry」などのオブジェクトの名前が選択される。
   - ブレークポイントとして選択したアクションのバックグラウンドが白色に、テキストは赤色に変わる。



7 次のセクションの指示に従って、追加のアニメーションプロセスを実行します。

ブレークポイントまでの実行

［Run to Breakpoint］ツールは、［Toggle Breakpoint］ツールと組み合わせて使用することにより、アニメーションプロセスを制御できます。時間のかかるアクションモデルでは、アニメーションプロセスを実行する時間と停止するポイントを制御するための機能が役立ちます。この機能は、［Run to Breakpoint］を使用して実行できます。

➢ ブレークポイントまでアニメーションを実行する

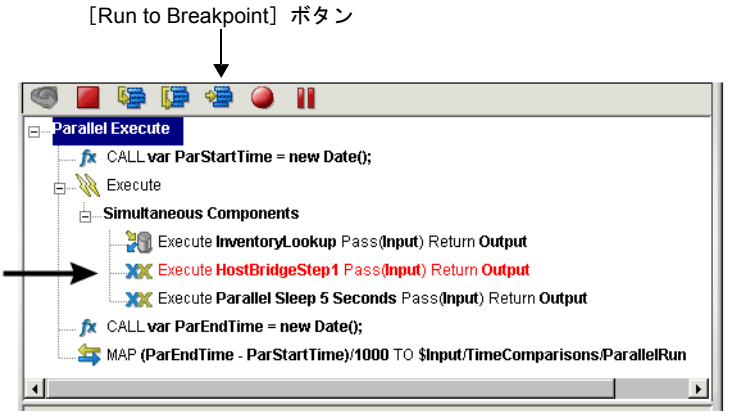
1 サービスまたはコンポーネントを開きます。開いたサービスまたはコンポーネントが該当するエディタに表示されます。

2 必要に応じてエディタのペインを調節し、アクションペインのコンテンツを表示します。

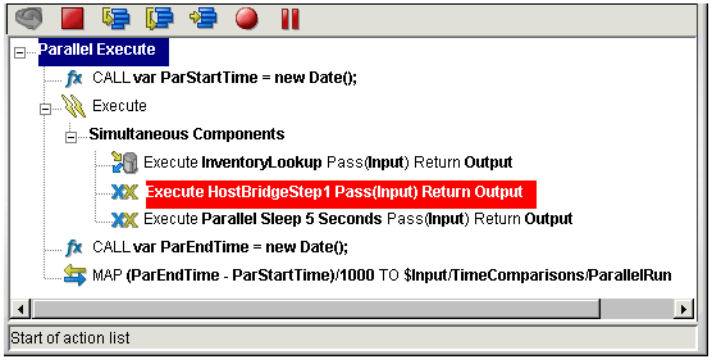
3 テスト用にブレークポイントとして設定するアクションを選択します。

4 アクションペインのツールバーで［**Toggle Breakpoint**］ボタンをクリックするか、またはキーボードの <**F2**> キーを押します。

5 アクションペインのツールバーで［**Start Animation**］ボタンをクリックするか、またはキーボードの <**F5**> キーを押します。サービスまたはコンポーネントのアクションペインは、次のようになります。



6 アクションペインのツールバーで［**Run to Breakpoint**］ボタンをクリックするか、またはキーボードの <**F9**> キーを押します。アニメーションプロセスにより、ブレークポイントの前までのアクションがすべて実行されます。プロセスはブレークポイントで停止し、ブレークポイントは赤色で選択されます (次を参照)。



7 次のセクションの指示に従って、アニメーションプロセスを続行します。

アクションへのステップイン

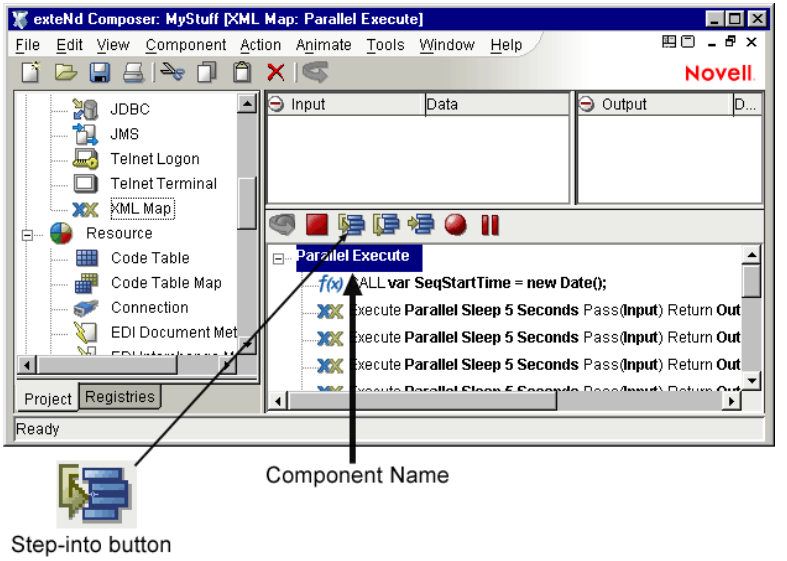
［Step Into］ツールを使用すると、アクションモデルで選択されているアクションを実行した後、次のアクションに移動します。Component、Repeat、Decision、または Try/On Error アクションが現在選択されているアクションの場合、次に選択されたアクションがこれらのアクションの詳細になります。Repeat ループの場合は、［Step Into］を押すと、True または False 分岐で次のアクションが処理されます。Try/On Error アクションの場合は、［Step Into］を押すと、実行内と、おそらく On Error 分岐で、次のアクションが処理されます。Component アクションの場合は、別のウィンドウが開き、そのコンポーネントに「ステップイン」します。このツールは、アクションモデル全体でアクションを 1 つずつ処理するために使用したり、［Run to Breakpoint］ツールと組み合わせて使用したりすることができます。［Run to Breakpoint］ツールを実行すると、このツールを選択して、ブレークポイントとして指定したアクションを実行できます。

これを実行するのに考えられるシナリオとして、10 個のアクションは適切に動作しているが、11 番目のアクションは適切に動作するかどうか確信がないような場合があります。このような状況では、11 番目のアクションをブレークポイントとして設定し、［Run to Breakpoint］ツールを実行してから、［Step Into］ツールを実行して 11 番目のアクションを対処することが可能です。

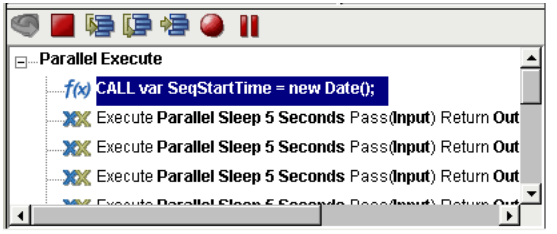
注記： サービスまたはコンポーネントがアクションモデルから呼び出され、呼び出されたオブジェクトを表示するために個別のエディタが開かれた場合、そのオブジェクトのアクションモデルを通して完了する必要があります。完了すると、エディタが閉じて、元のアクションモデルに戻ります。

➢ ［Step Into］ツールを実行する

1. サービスまたはコンポーネントを開きます。サービスまたはコンポーネントが該当するエディタに表示されます。
2. 必要に応じてペインを調節し、アクションモデルのコンテンツを表示します。
3. アクションペインのツールバーで［**Start Animation**］ボタンをクリックするか、またはキーボードの <**F5**> キーを押します。アニメーションツールがアクティブになり、オブジェクトの名前がアクションモデルで選択されます。



4 アクションツールバーで［**Step Into**］ボタンをクリックするか、またはキーボードの <**F7**> キーを押します。最初のアクションが選択されます。



5 ［**Step Into**］ボタンをもう一度クリックします。アクションが実行され、次のアクションが選択されます。

6 各アクションが実行され、次のアクションが選択されたら、［**Step Into**］ボタンをクリックすることによって、アクションモデルを通して作業を続行します。

7 別のサービスまたはコンポーネントを呼び出すアクションが選択されたら、［**Step Into**］ボタンをクリックします。次のような結果になります。

- 新しいウィンドウが開き、該当するエディタ (つまり、サービスまたはコンポーネント) が表示される。
- アクションペインに、(［Start Animation］を除く) ツールがすべてアクティブな状態で表示される。

8 呼び出されたコンポーネントのアクションペインのツールバーで、[**Step Into**] ボタンをクリックします。

9 引き続き [**Step Into**] ボタンをクリックして、呼び出されたコンポーネントのアクションをすべて実行します。アクションをすべて実行すると、ウィンドウが閉じ、元のアクションモデルで停止したポイントに戻り、次のアクションが選択されます。

10 引き続き [**Step Into**] ボタンをクリックして、元のサービスまたはコンポーネントのアクションをすべて実行します。完了すると、メッセージが表示されます。

注記： アクションモデルでは 1 つまたは複数のコンポーネントが呼び出されることがあり、各コンポーネントでも複数のコンポーネントが呼び出されることがあります。呼び出されたコンポーネントの各インスタンスでは、アニメーションツールはまったく同じように機能します。たとえば、呼び出されたコンポーネント内で [Toggle Breakpoint] 機能を実行した後、元のサービスまたはコンポーネントで [Run to Breakpoint] 機能を実行したいことがあります。このような場合、アクションモデルでは、アクションの実行を開始して、呼び出されたコンポーネントを開き、設定したブレークポイントでアクションを停止します。

アクションのステップオーバ

[Step Over] ツールは、Component、Repeat、Decision、または Try/On Error アクションの詳細にステップインしない場合に役立ちます。

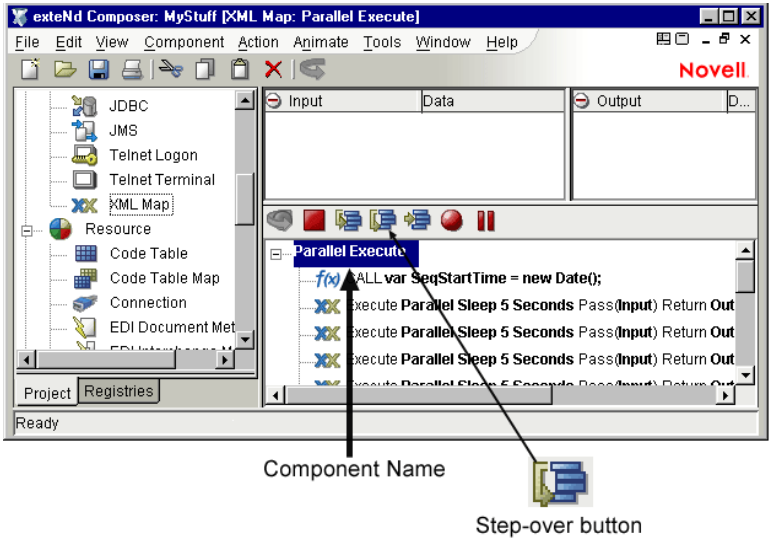
Component アクションの場合、[Step Over] ツールを使用すると、ターゲットコンポーネントを通してアニメーションを続行するのに長い時間を費やして別のエディタを開く可能性がなくなります。[Step Over] ツールでは、単にターゲットコンポーネントを実行した後、アクションモデルで次のアクションを選択します。

同様に、Try/On Error、Repeat、またはその他の制御フローアクションでラップされたコードのブロックの場合は、[Step Over] 機能を使用すると、(ループ内で厄介になることがあるため) 各アクションを個々に通さずに、コードのブロック全体を一度に実行できます。

［Step Over］ツールは、［Run to Breakpoint］ツールと組み合わせて使用すると便利な場合があります。たとえば、［Toggle Breakpoint］機能を使用し、［Run to Breakpoint］ツールを実行した後、［Step Over］ツールを使用して、ブレークポイントとして指定したアクションを実行することが可能です。［Step Over］ツールでは、個々のアクション (関係のないものが含まれる場合もある) を通して単調に実行する必要がないため、長いアクションモデルをテストする場合に時間を大幅に節約できます。

➢ ［Step Over］ツールを使用する

1. サービスまたはコンポーネントを開きます。サービスまたはコンポーネントが該当するエディタに表示されます。
2. 必要に応じてペインを調節し、アクションモデルのコンテンツを表示します。
3. アクションペインのツールバーで［**Start Animation**］ボタンをクリックします。アニメーションツールがアクティブになり、オブジェクトの名前がアクションモデルで選択されます。



4. ループ、またはインデントされたコードブロックに先行するコードの別の行に達するまで、［**Step Into**］ボタンを使用して、アクションモデルを通して作業します。
5. アクションツールバーで［**Step Over**］ボタンをクリックするか、またはキーボードの <**F8**> キーを押します。インデントされたコードのブロックの「後」にある最初のアクションが選択されます (インデントされたコードはすべて通常どおりに実行され、インデントされたアクションの行を個々に通して作業する必要なく、「アウトデント」された次のアクションに直ちに移動します)。

6 必要に応じて［**Step Over**］ボタンをクリックし、アクションモデルを通して作業を続行します。

7 引き続き［**Step Into**］ボタンまたは［**StepOver**］ボタン、あるいはその両方をクリックして、アクションモデルのアクションをすべて実行します。完了すると、メッセージが表示されます。

アニメーションの一時停止

［Pause Animation］ツールを使用すると、アクションモデルでアクションの実行を一時停止できます。これは、アクションモデルに長いループが含まれている場合に特に便利です。

➢ アニメーションを一時停止する

1 アクションの実行中に、アクションペインのツールバーで［**Pause Animation**］ボタンをクリックするか、またはキーボードの <**F6**> キーを押します。

2 アニメーションプロセスを再開するには、必要に応じて［**Step Into**］、［**Step Over**］、または［**Run to Breakpoint**］(ブレークポイントが設定されている場合) をクリックします。

アニメーションの停止

［Stop Animation］ツールでは、単にアニメーションプロセスを停止します。アニメーションは、停止すると、停止した場所から再開することはできません。このため、アクションモデルの開始から再開する必要があります。

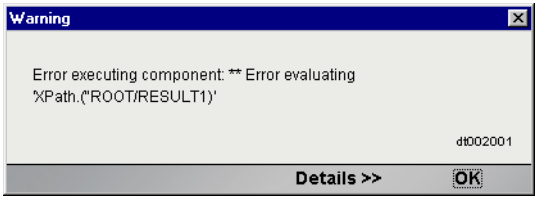
➢ アニメーションを停止する

1 アニメーションの実行中に、アクションペインのツールバーで［**Stop Animation**］ボタンをクリックするか、またはキーボードの <**Shift**>+<**F5**> キーを押します。次のメッセージが表示されます。

2 ［**OK**］をクリックします。

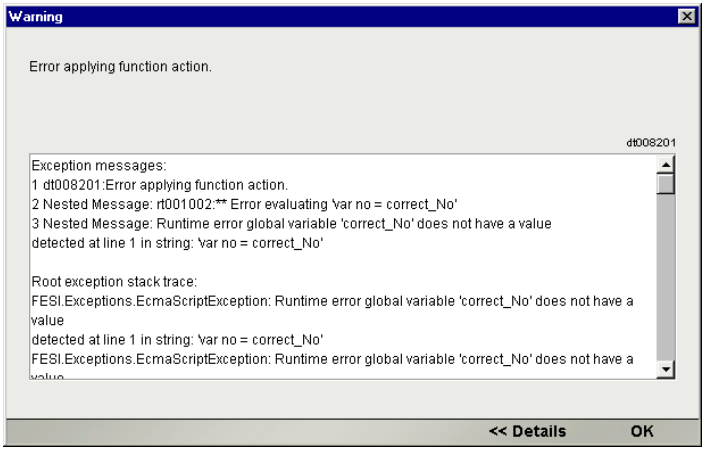
実行エラー

アクションが正しく実行されなかった場合、エラーメッセージが表示されます。



［**Details**］ボタンをクリックすると、発生した問題に関する詳細を表示できます。この機能により、完全な Java スタックトレースが生成されます。

すべてのエラーメッセージは、Composer の［View］メニューから表示できるシステムログにも書き込まれます。



注記： ログレベルを 1 に設定した場合（［**Tools**］>［**Configuration**］の順に選択）は、Composer のメインウィンドウの［Output］フレームに詳細なエラー情報が表示されます。このフレームが現在表示されていない場合は、<Ctrl>+<Shift>+<O> キーを使用して表示レベルを切り替えます。

テストのヒント

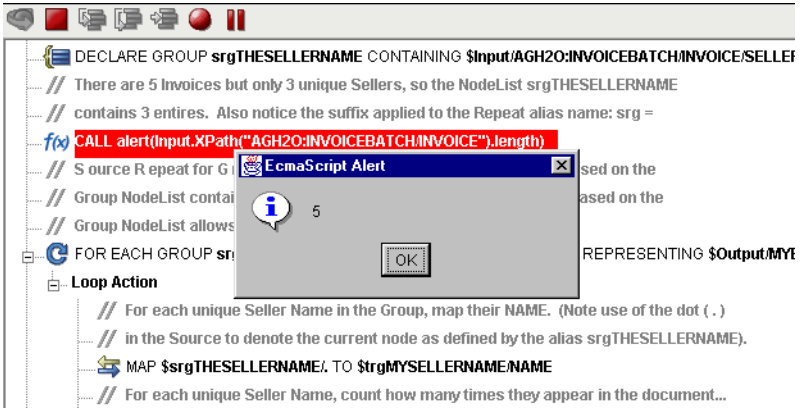
テストに役立つアクションをコンポーネントやサービスに作成すると便利な場合があります。Composer には、テストを援助する組み込み機能がいくつか含まれています。これらの機能は、次のとおりです。

- ECMAScript Alert() 関数
- プロジェクト変数

ECMAScript alert() 関数の使用

Map アクションやその他のタイプのアクションを実行する前または実行した後、あるいは両方の場合に、データ値を検査したいことがあります。このような状況では、ECMAScript `alert()` 関数を含む関数アクションを作成して検査することができます。alert 関数を使用すると、指定した値が表示されたメッセージボックスが現れます。

次の例では、特定の XPath ノードリストのノードカウントが表示されるよう、alert 関数アクションが作成されています。



注記： アプリケーションサーバ環境にプロジェクトを配備する前に、`alert()` 関数を使用するアクションをすべて無効にすることが推奨されます。`alert()` の機能は、設計時のテストの場合にのみ有用です。

プロジェクト変数を使用したデバッグのオン / オフの切り替え

コンポーネントまたはサーバにデバッグ関連のアクションが多数含まれている場合、プロジェクトの配備時にアクションが実行されないようにすることができます。1 つの方法は、Decision アクション内で使用できる「プロジェクト変数」を作成して、デバッグアクションを実行するかどうかを決定することです。その後、すべてのデバッグ関連アクションをコンポーネントの Decision アクション内に配置します。

別の方法は、コンポーネントの内部メソッドの 1 つに直接の呼び出しを使用して、現在の環境がランタイムまたは設計のいずれであるかを決定することです。

```
gDebugMode =
!Packages.com.sssw.b2b.rt.GNVXObjectFactory.isRuntime();
```

これは、カスタム Java メソッドを呼び出すために ECMAScript を使用する例です。ここでの結果は、コンポーネントが設計 (またはアニメーション) 時に Composer で実行されている場合は、(コンポーネントにスコープされた) 変数 `gDebugMode` に true が含まれ、コンポーネントがサーバ上に配備したプロジェクトで実行されている場合は、false が含まれます。

次に、`alert()` を呼び出してもよいかどうかを判断するためにこの「標識変数」を使用する例を示します。

注記： Log アクションも、デバッグにはとても重要で、[Configuration] ダイアログボックスの [System Environment] タブから簡単に制御できます (151 ページ「Log アクション」を参照)。

アニメーションテストと配備テストの環境的相違

Composer でのアニメーションテストと、配備テストには、大きな環境的相違があります。両方のタイプのテストは、作成したコンポーネントやサービスを適切に検証するために必要です。相違点については、次の表に詳しく説明します。

表 12-2

	Composer でのテスト	配備テスト
OS	Win98、WinNT、または Win 2000	WinNT または Sun Solaris
プラットフォーム	Java Runtime Environment (JRE)	フェイルオーバ、セキュリティ、接続 MGT などに対するJREサポートを完全に備えたアプリケーションサーバ
コンポーネントまたはサーバの起動	Composer から直接的	サービストリガによってのみ (つまり、配備サーブレットまたは EJB)
xObject アクセス	ディスクファイルから	アプリケーションサーバ内の JAR ファイルから
ランタイムコンテキスト	個々のコンポーネント、またはサービス内で実行されているコンポーネントをテスト	常にサービス内から

	Composer でのテスト	配備テスト
サービス入力およびコンポーネント入力	入力ドキュメントは、ローカルマシンのサンプル XML ドキュメントだけでなく、他のサービスやコンポーネントからの DOM に由来することが多い	入力ドキュメントは、サービストリガ、または他のサービスやコンポーネントからの DOM を介して、サービスとコンポーネント内に渡される
次に対するプロジェクト変数 : * ログファイルのパス * DTD URL * XSL URL * メール送信サーバ * XML 交換 URL	通常はローカルマシン上の場所を指す (ただし、サーバや Web 上の場所でも可)	プロダクションサーバおよび Web 上の場所を指す
テストツール	Log アクション以外に、ダイアログボックス (ECMAScript alert() 関数) を使用してランタイム値を表示することも可能	ダイアログボックスの使用は不可
JDBC 接続	サーバ接続プールを使用しない	サーバ指定の接続プールを使用する
HTTP 接続	ローカルマシンまたはテストサーバを指している可能性がある	テストサーバまたはプロダクションサーバを指している

13 サービスの操作

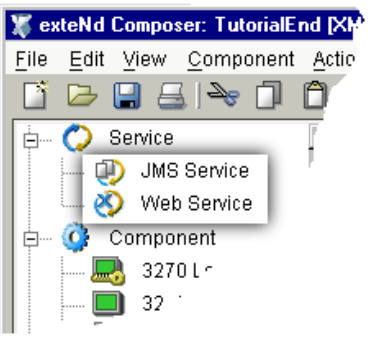
サービスとは？

「サービス」は、作成するさまざまなコンポーネントを結合して、アプリケーションサーバ環境 (リクエストで開始され、結果として応答を返す) 内に処理の論理単位を作成するために使用されます。サービスでは、さまざまなコンポーネントが連続的または条件的、あるいはその両方で通常は実行され、他のサービスを実行することもできます。その他のサービスレベルのタスクには、一般的なエラー処理またはログ記録の操作、あるいはその両方が含まれる場合があります。

サービスは、コンポーネントとほぼ同様の方法で、その他のあらゆる種類の実行可能な xObject を作成するのに使用するのと同じ一般的なコンポーネントエディタ環境 (および同じコアアクション) を使用して、Composer 内で作成されます。

使用可能なサービスタイプ

exteNd には、Web サービスと JMS サービスという 2 つの主なタイプのサービスがあります (JMS は Java Messaging Service の略で、メッセージ指向ミドルウェアに対する Sun 定義のインタフェースです)。WAR ファイルまたは EAR ファイルで配備されたプロジェクトには、これらのサービスのいずれかまたは「両方」が含まれていることがあります。2 つのサービスタイプは、Composer のナビゲーションフレームの［Service］という見出しの下に、2 つの異なるアイコンとして表示されます (次を参照)。



注記： Novell exteNd Composer JMS Connect がインストールされていない場合、JMS サービスタイプは Composer に表示されません。ただし、Web サービスカテゴリは常に表示されます。

JMS サービスの決定的な特徴は、そのトリガ方法にあります。企業メッセージングを使用するサービスが Web からトリガされる場合、そのサービスは、Web サービスとして作成される必要があります。また、メッセージの着信によってトリガされる場合は、JMS サービスとして作成される必要があります。

サービスとコンポーネントの違い

サービスは、実質的には exteNd コンポーネントの専用タイプです。サービスにはアクションモデルがあり、コンポーネントによって実行されるものと同じタスクを実行することができます。ただし、設計上の理由から、これらのタスクは例外ベースのみに制限し、サービスを構成する個別のコンポーネントでビジネス論理を処理できるようにする必要があります。「サービス」で使用する必要のある主なアクションは、Component、Log、Decision、Function、Try/On Error、および Raise Error です (サービスによるこれらのアクションの使用例については、326 ページ「コンポーネントを使用したサービスの作成」を参照してください)。

サービスは、アプリケーションサーバで実行される作業の単位、つまりアプリケーションであるという点で、コンポーネント (またはその他のすべての exteNd Composer オブジェクト) と異なっています。選択したアプリケーションサーバで実行されるよう登録できるのは、「サービス」のみです (コンポーネントではありません)。サービスは、コンポーネントやその他のサービスを「呼び出す」ことを目的としています。アプリケーションには、その他のサービスやコンポーネントをすべて呼び出すサービスが 1 つ含まれている場合や、アプリケーションが、アプリケーションサーバ上のプロセスによって個別に呼び出される「一連」のサービスから構成されている場合があります (さらに、さまざまなサービスが完全に異なる方法でトリガされることがあります)。

アプリケーションサーバでの処理の基本単位として「サービス」を使用することは、Composer アプリケーションの設計における主要な目的です。

Composer Web サービスおよび WSDL

Composer Web サービスは、URL で配備された WSDL 記述のサービスにすることができますが、必須ではありません。簡単に言うと、Composer Web サービスは、単に他のコンポーネントを呼び出すコンポーネントです。コンポーネントではなく「サービス」であるというのは、Web サービスの xObject をサーバにあるサーブレットまたは Java オブジェクトからトリガできることであり、それに対し、コンポーネントの xObject は、この方法ではトリガされません (Component は「サービスによって呼び出されます」)。

Composer Web サービスを使用すると、WSDL によって示される操作パターン (通知、一方向、リクエスト - 応答、または依頼 - 応答) を実装できます。また、パブリック URL に配備したり、ローカルアプリケーションとして内部で実行したりすることもできます。さらに、WSDL との関連付けや SOAP リクエストの受け付けを行ったり行わなかったりすることもできます。

Web サービスの例

Web サービスの部分と機能は、次の図のとおりです。その下には説明があります。

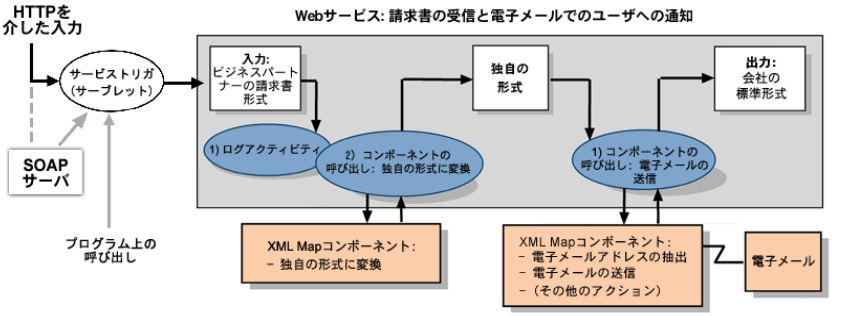


図 13-1

図の中で、グレーの大きな長方形は、Web サービスを表しています。数字が付いた影付きの楕円は、アクションモデルのアクションを表しています。入力 XML ファイルと出力 XML ファイル (正方形)、および呼び出されるコンポーネント (サービスの外にある小さな長方形) も表示されています。

このサービスの目的は、請求書 (業界標準の形式で) を受信し、請求書が受信されたことを送信者に通知することです。サービスの完了には、XML ドキュメントとして受信される請求書の操作がいくつか必要です。

サービスは、次のように機能します。

1 サービスは、サービストリガ (配備時に作成されるオブジェクトで、一部の外部イベントに応答してサービスを起動するよう設計されている) によって呼び出されます。サーブレットは、そのサーブレットに HTTP Post を発行するビジネスパートナーのアプリケーションサーバによって起動したり、(グレーの長い矢印で示されているように) ホストサーバの Java プロセスによってプログラムで起動したりできます。

2 この例の場合、サービスの最初のジョブは、ファイルを記述する Log アクションを実行して、サービスのアクティビティを実行時に記録することです。

3 次に、サービスでは、Component アクションを実行して、*Convert to My Format* コンポーネントを呼び出します。

4 *Convert to My Format* コンポーネントでは、業界標準の請求書形式を入力として使用し、企業の内部形式 (*My Format*) にフォーマットされた XML ファイルを出力として返します。

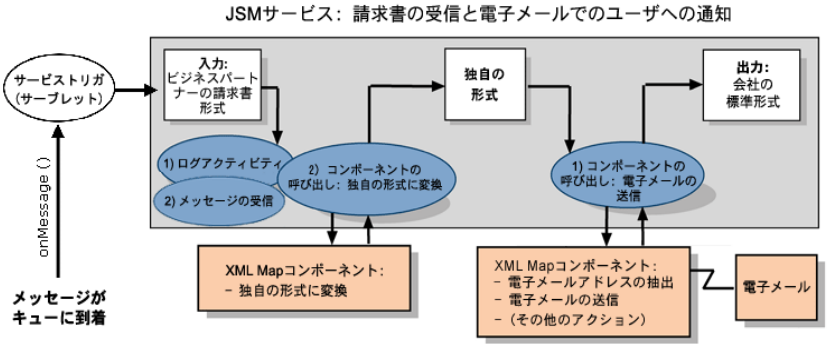
5 サービスでは、別の Component アクションを実行して、*Send Email* コンポーネントを呼び出します。*My Format* ファイルは、*Send Email* コンポーネントに対する入力です。

6 *Send Email* コンポーネントでは、複数のアクションを実行し (XML Interchange アクションを使用して請求書から電子メールアドレスを抽出したり、Send Mail アクションを使用して電子メールを送信したりする)、XML ファイルである *eMail* を返します。

7 企業標準の形式ファイルは、サービスによる出力です。

JMS サービスの例

JMS サービスの部分と機能は、次の図のとおりです。



JMS サービスは、前に説明した Web サービスとは実質的にあまり異なっておらず、主な相違点は、メッセージがキュー (または、パブリッシュ / サブスクライブに関する用語では「トピック」) に入ると JMS サービスが起動されることにあります。JMS サービスでは、リスナによって登録されているキューまたはトピックにメッセージが入ると `onMessage()` メソッドが自動的に呼び出される *MessageListener* オブジェクトを実装します。`onMessage()` メソッドでは、サービスを実行します。

JMS サービスには、本質的に、JMS Connect を使用して作成された Receive Message アクションが 1 つだけ含まれていなければなりません。Receive Message アクションを使用すると、サービスでは、受信メッセージのデータにアクセスしたり、その受信を適切に確認することができます。

このサービスに対するアクションモデルの残りは、先に説明した Web サービスの場合と同様です。

注記： この例は、Novell Composer JMS Connect を購入してインストールした場合にのみ該当します。

新しいサービスの作成

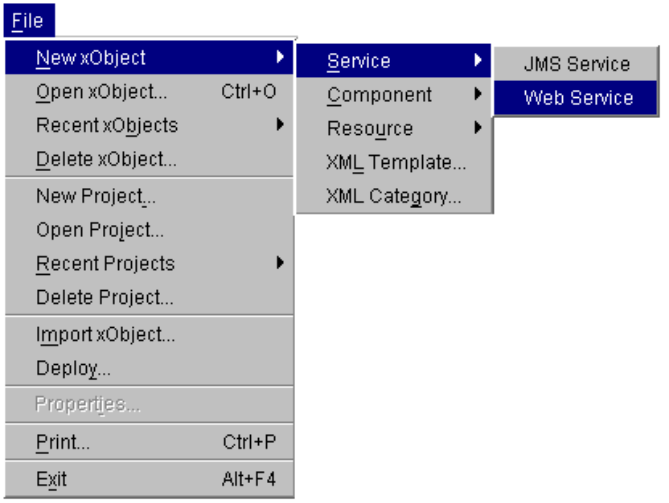
新しいサービスは、新しい XML Map コンポーネントを作成する場合と同様に作成します。XML Map コンポーネントをまだ作成していない場合は、サービスを作成する前に必要な XML テンプレートを作成する必要があります。詳細については、89 ページ「XML テンプレートの作成」を参照してください。

サービスに対する XML テンプレートの指定について

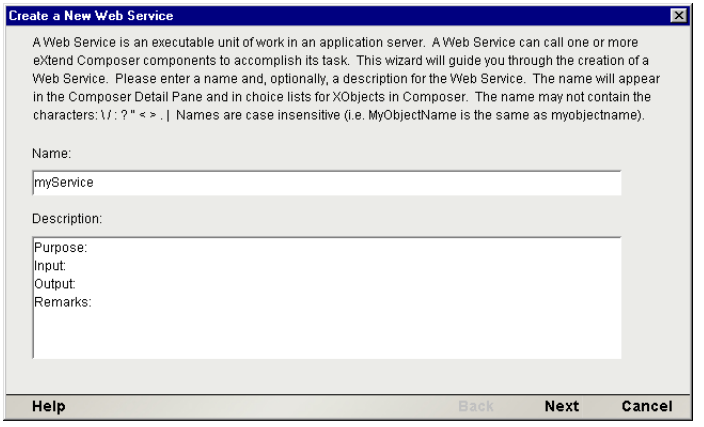
サービスを作成する場合は、コンポーネントの場合と同様に、入力テンプレートと出力テンプレートを指定します。サービスが、データを処理するようにではなく、コンポーネントを呼び出すよう設計されている場合、そのサービスに対して選択する入力テンプレートは、最初のコンポーネントによって使用されるテンプレートと同じになることがよくあります。また、出力テンプレートは、シーケンスの最後のコンポーネントに対する出力テンプレートと同じになることがよくあります。

➢ 新しい Web サービスを作成する

1 Composer ウィンドウの［File］メニューから、［**New xObject**］、［**Service**］、［**Web Service**］の順に選択します (次を参照)。



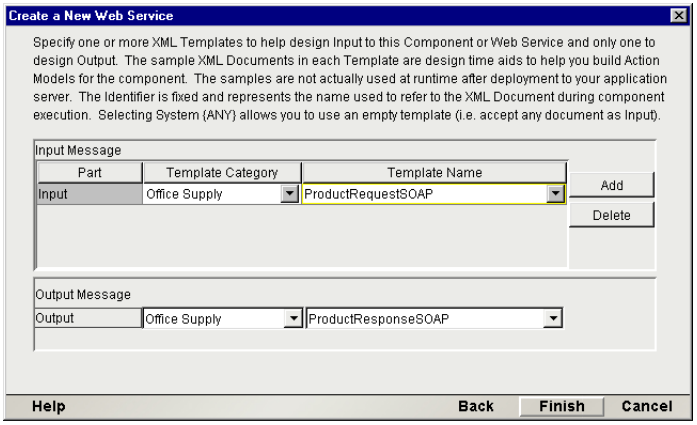
Create a New Web Service Component ウィザードが表示されます。



2 「**名前**」および「**説明**」(オプション) を入力します。

オプションの［Description］フィールドは、サービスによって実行されるタスクを説明するために使用できます。

3 ［**Next**］をクリックして、テンプレートパネルを表示します。

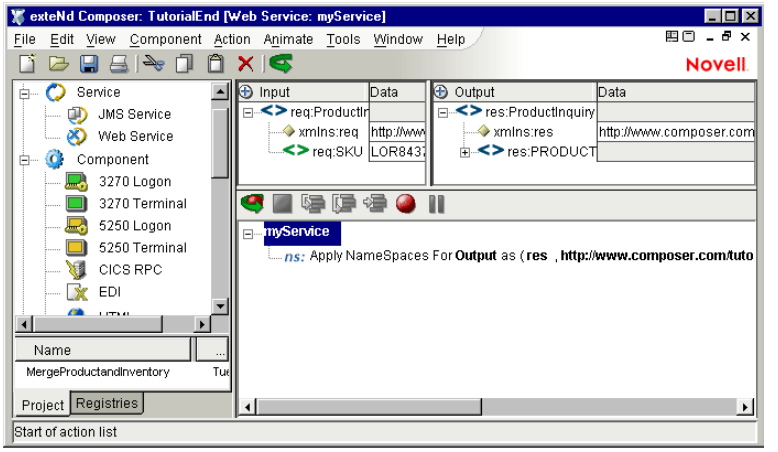


4 入力テンプレートおよび出力テンプレートを次のように指定します。ヒントについては、321 ページ「サービスに対する XML テンプレートの指定について」を参照してください。

- デフォルトのカテゴリと異なる場合は、［**Template Category**］を選択します。
- 選択した［**Template Category**］にある XML テンプレートのリストから［**Template Name**］を選択します。
- 入力 XML テンプレートをさらに追加するには、［**Add**］をクリックして、手順 2 から 4 を繰り返します。
- 入力 XML テンプレートを削除するには、エントリを選択して、［**Delete**］をクリックします。

5 出力として XML テンプレートを選択します。

6 ［**Finish**］をクリックします。コンポーネントが作成され、サービスエディタが表示されます。



テンプレートにネームスペース宣言が存在する場合、Composer によって、Declare Namesace アクションが新しいアクションモデルの上に自動的に生成されます。

JMS サービスの作成

JMS サービスの作成は、Web Service ウィザードと多くの共通点を持つウィザードを使用して行います。段階的な手順については、『*exteNd JMS Connect ユーザガイド*』を参照してください。

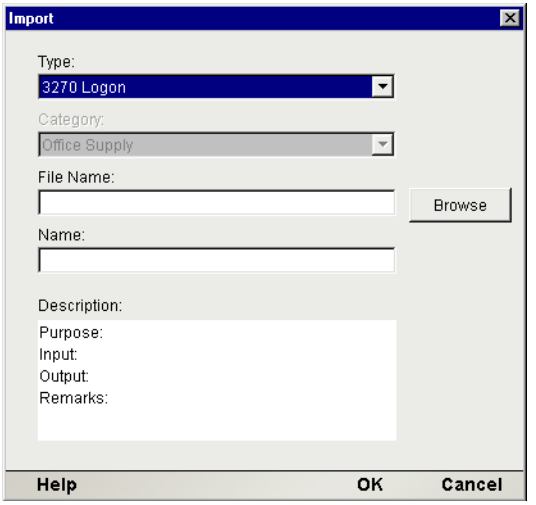
サービスのインポート

インポート機能を使用すると、別のプロジェクトで作成された Composer サービスのコピーを作成できます。インポートすると、現在のプロジェクト内で使用するためにサービスをカスタマイズすることが可能です。

➢ **サービスをインポートする**

1 exteNd Composer ウィンドウの［**Service**］項目を右クリックするか、またはメインの［**File**］メニューから［**Import xObject**］を選択して、［**Import**］を選択します。

［Import xObject］ウィンドウが表示されます。



2 ［**Type**］で［**Service**］を選択します (選択されていない場合)。

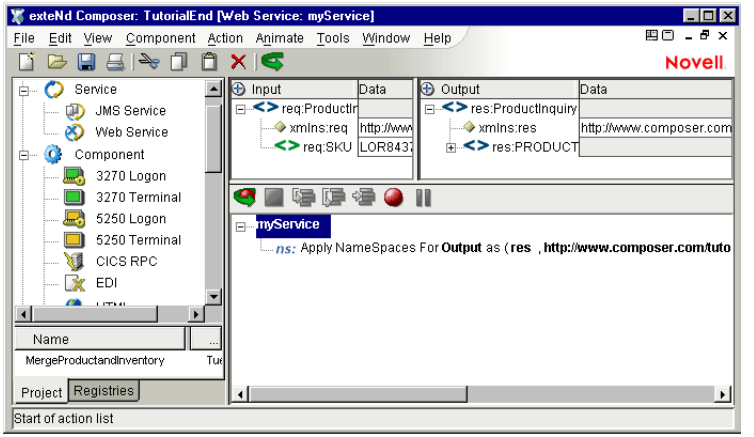
3 ［**File Name**］フィールドで、インポートするサービスの名前を入力するか、または［**Browse**］ボタンを使用してサービスを検索します。

4 ［**OK**］をクリックして、サービスをインポートします。

サービスエディタの概要

サービスエディタは、(通常は) コンポーネントやサービスの実行を指定したり、エラーログ、決定、および機能を実行したりする場所です。また、入力および出力の構造やデータをマップ、変換、および転送することもできます。

サービスエディタを使用すると、サービスの入力、出力、およびアクションを視覚化して操作するための論理的な作業環境が提供されます。サービスエディタは、複数のマッピングペインと 1 つのアクションモデルペインで構成されます。マッピングペインには、サンプル入力ドキュメントとサンプル出力ドキュメントの DOM が表示されます。アクションモデルには、DOM で動作するアクションが表示されます (この環境は、XML Map コンポーネントエディタの場合と本質的に同じです)。



サービスエディタの使用

サービスエディタには、XML Map コンポーネントエディタと同じ機能がすべてあります。サービスエディタの使用の詳細については、次のトピックを参照してください。

- 127 ページ「テンプレートを使用しない出力 DOM の作成」
- 128 ページ「一時 DOM の作成」
- 130 ページ「XML ドキュメントの再ロード」
- 131 ページ「サンプルドキュメントのロード」
- 132 ページ「コンポーネントの保存」
- 133 ページ「XML ドキュメントとしての DOM の保存」
- 135 ページ「コンポーネントのプロパティの表示」
- 137 ページ「コンポーネントの印刷」

コンポーネントを使用したサービスの作成

サービスは、通常は、1 つまたは複数の Component アクションで構成されます。それぞれのアクションでは、アプリケーション内で呼び出される次のコンポーネントまたはサービスで使用するデータのマッピング、変換、または転送、あるいはこれらすべてを行う特定のタスクを実行します。

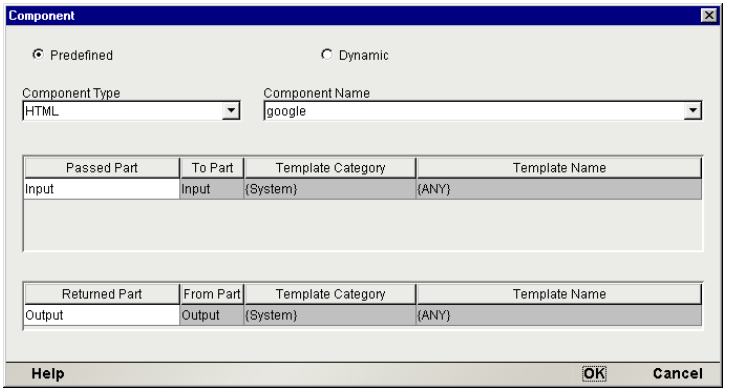
指定したランタイムの入力 DOM および出力 DOM が存在するコンポーネントまたはサービスを呼び出して実行するには、Component アクションを使用します。

➢ Component アクションを追加する

1 アクションモデルで、コンポーネントまたはサービスへの呼び出しを配置する行を選択します。選択した行の下に新しいアクションが挿入されます。

2 ［**Action**］メニューから、［**New Action**］、［**Component**］の順に選択します。［Action Component Information］ダイアログボックスが表示されます。

3 ラジオボタンをオンにして、［**Predefined**］を選択します (選択されていない場合)。定義済みおよびダイナミックの Component アクションの説明については、第 7 章を参照してください。



4 左上にあるプルダウンメニューから［**Component Type**］を選択します。

5 実行する「**コンポーネント**」の**名前**を選択します。

6 ［**Passed ID**］フィールドで、［DOM］を選択します。

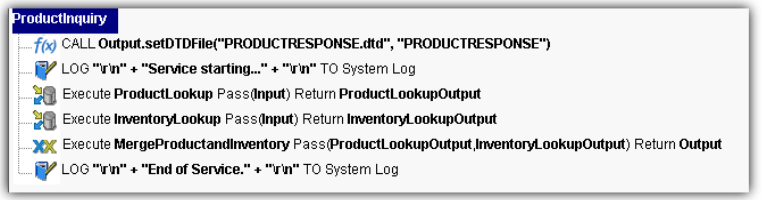
7 ［**Returned ID**］フィールドで、［DOM］に対し［Output］または［Temp］のいずれかを選択します。

8 ［**OK**］をクリックします。

サービスアクションモデルの例

Component アクションは、サービスに対して追加すると、サービスエディタのアクションモデルペインに表示されます。サービスのアクションモデルは、コンポーネントが呼び出される順序を表します。

アクションモデルの例は、次のとおりです。



アクションモデルの機能にはログ機能がいくつかあり、コンポーネントを次のように実行します。

1. 最初の Component アクションでは、コンポーネント (**ProductLookup**) を呼び出します。これによって、DOM は、コンポーネントがサービスからデータを受信する入力ドキュメントハンドル (**Input**) として渡されるよう指定され、また、コンポーネントの出力 (**ProductLookupOutput**) を受信するようにも指定されます。
2. 2 番目の Component アクションでは、コンポーネント (**InventoryLookup**) を呼び出します。これによって、DOM は、コンポーネントがサービスからデータを受信する入力ドキュメントハンドル (**Input**) として渡されるよう指定され、また、コンポーネントの出力 (**InventoryLookupOutput**) を受信するようにも指定されます。
3. 3 番目の Component アクションでは、コンポーネント (MergeProductAndInventory) を呼び出します。これによって、Input DOM および Input1 DOM としてコンポーネントがサービスからデータを受信する ProductLookupOutput および InventoryLookupOutput という DOM が渡され、サービス DOM はコンポーネントの出力 (Output) を受信するよう指定されます。

サービスに関する FAQ

異なるタイプのコンポーネント間ではデータをどのように渡すのですか？

exteNd には、異なるコンピューティング環境にアクセスできるさまざまな Connect コンポーネントが用意されています。すべてのコンポーネントタイプの入力および出力は、単に XML ドキュメントです。これは、異なるコンポーネントタイプ間での通信が直接的および簡単であることを意味します。

コンポーネント間でデータを渡すには、基本的な方法が 2 つあります。最初の方法では、個別に呼び出されるコンポーネントから入力および出力ドキュメントを渡したり受信したりするサービスを使用します。この方法では、コンポーネントは直接通信せず、代わりにサービスが連絡先として使用されます。2 番目の方法では、互いを直接呼び出すコンポーネントを使用します。選択する方法は、サービスの設計の仕方や、実行されるタスクのタイプによって異なります。

Composer サービスでは複数の入力ドキュメントを受け付けることができますか？

これは、サービスが配備されている方法によって異なります。サービスが SOAP サービスとして配備されている場合、SOAP サーバでは、複数の入力ドキュメントをサービスに渡すことができます (複数の入力がサービスの WSDL で指定されている場合)。4 つの標準的な Composer サービストリガタイプである Params (URL/ フォーム)、XML (MIM マルチパート)、XML (HTML フォームフィールド)、および XML (HTTP POST) を含む、その他のすべての状況では、入力として受け付けることができる XML ドキュメントの数は「1 つ」だけです。

配備の詳細については、『*Composer Enterprise Server ユーザガイド*』を参照してください。

サービスによって直接呼び出されないコンポーネントを実行することはできますか？

2 つのコンポーネントを呼び出す 1 つのサービスを含むプロジェクトを作成した場合、2 番目のコンポーネントを呼び出すサービスに出力を返す前に 3 番目のコンポーネントを呼び出すよう最初のコンポーネントを指定することは、設計上可能です。技術的には、3 番目のコンポーネントは「サービス内に含まれている」わけでも、そこから直接呼び出されるわけでもありません。サービスについて理解するのに重要なことは、アプリケーションサーバ上の「サービストリガ」オブジェクトによって呼び出されるのはサービスだけであるということです。コンポーネントはサービスに直接リンクされている必要はありませんが、一連のイベントにおいて何らかの理由によりコンポーネントが呼び出されない場合、そのコンポーネントは実行されません。

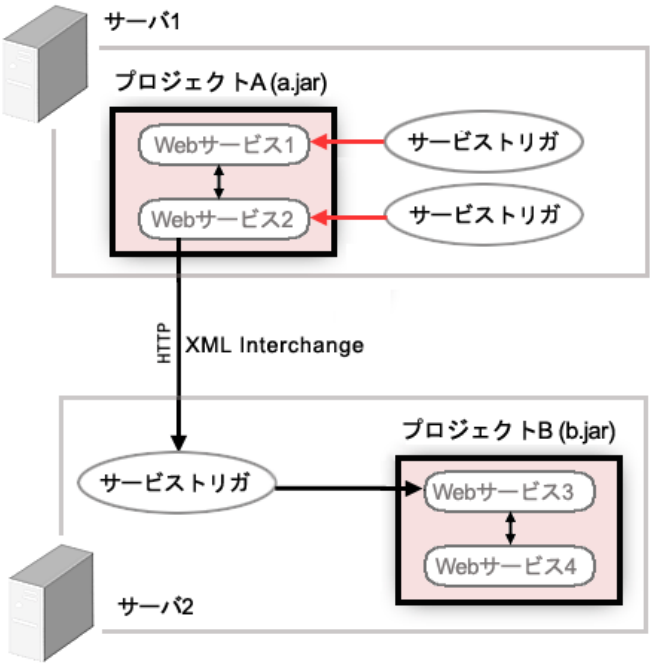
サーバトリガオブジェクトは、exteNd の配備フレームワークで作成する Java サーブレット、EJB、または MessageListener (JMS の場合) です。このオブジェクトは、Web ページに埋め込まれている URL、または Web 上の別のプログラムから呼び出される URL によってトリガされます。トリガされると、サーブレットまたは EJB によって Composer サービスが開始されます。

繰り返しますが、配備、サービストリガ、フレームワークオブジェクトなどの詳細については、『*Composer Enterprise Server ユーザガイド*』を参照してください。

別の JAR ファイルに配備されているサービスはどのように呼び出しますか？

プロジェクトは JAR ファイルとして配備され、通常は、プロジェクト内のサービスやコンポーネントで「他の」サービスまたはコンポーネント、あるいはその両方を呼び出す必要がある場合、その「呼び出される」サービス / コンポーネントは同じ JAR ファイルに保存されます。しかし、場合によっては、「異なる」JAR ファイル (つまり、配備された別のプロジェクト) に存在する別のサービスを呼び出すようサービスを指定すると便利 (または必要) なことがあります。この操作は、XML Interchange アクションにより実行できます。

XML Interchange アクションを使用すると、コンポーネントやサービスでは、HTTP GET、PUT、または POST プロトコルを介して XML ドキュメントを出力できます。また、別の Composer サービスに URL を提供すると、該当するサービスのサーブレットトリガが XML Interchange アクションによって起動されるようにすることができます (次の図を参照)。



この図において、プロジェクト A の Web サービス 2 では、プロジェクト B の Web サービス 3 を呼び出すことを希望しています。Web サービス 2 では、Web サービス 1 を直接呼び出すことはできますが、同じプロジェクト JAR に存在するため、Web サービス 3 に直接接続することはできません。このため、リモートサービスのサービストリガを起動する XML Interchange アクションを実行する必要があります。

1 つのサービス内から呼び出される各コンポーネントの単一ファイルでは、アクティビティのログをどのように出力しますか？

Log アクションでは、サービス内のコンポーネントのアクティビティに関する情報が書き込まれます。サービス内におけるすべてのコンポーネントのアクティビティを記録する単一ログファイルを作成するには、サービスと各コンポーネントで使用されるそれぞれの Log アクションに対して、［**Log to:**］フィールドに同じファイル名を単に指定します。Log アクションの詳細については、151 ページ「Log アクション」を参照してください。

注記： 複数の Log アクションに対して同じファイル名を指定する場合は、[**Clear the Log File**] チェックボックスがオフになっていることを確認してください。このチェックボックスをオンにすると、各 Log アクションによって書き込みが行われる前にログファイルが消去されてしまいます。ただし、サービス内に含まれる最初の Log アクションに対しては、このオプションを選択することができます。このオプションを選択すると、ログがクリアされ、ファイルの最後にメッセージを追加し続けることなく、アクションモデルを複数回トラブルシューティングしたり、アニメーション表示したりすることが可能になります。

サービステスト時のサンプルドキュメントのロード

コンポーネントを使用する場合と同様に、サービスに対して入力として使用する XML テンプレートには、複数のサンプルドキュメントを含めることができます。テストの実行中は、サービスのアクションを順に進みながら、適切なサンプルドキュメントをロードして、サービスで各インスタンスを処理できるかどうかを確認することができます。

詳細については、131 ページ「サンプルドキュメントのロード」を参照してください。

14 レジストリの操作

この章では、exteNd Composer に装備されているレジストリ参照機能について説明します。Web サービスを対象としたビジネスレジストリ標準は UDDI (Universal Description, Discovery and Integration) で、独自のサービスを説明したり、他の会社のサービスを見つけたり、リモート環境にあるパートナーと自動化または半自動化された方法により e ビジネスを行うために必要な手段を理解したりするのに統一された方法を企業に提供するために設計されています。UDDI は、次に説明されているレジストリ管理機能の基礎を形成しており、UDDI に精通していることをここでは想定しています。UDDI の詳細ついては、完全な標準を **http://www.uddi.org** から入手できます。

Registry Manager の機能

exteNd Composer には、Composer メインナビゲーションフレームの下部にある［Registries］タブからアクセス可能な Registry Manager が組み込まれています。また、［Profiles］機能 (Composer メインメニューバーで［**Tools**］>［**Profiles...**］の順に選択) を使用してレジストリを定義するための機能もあります。

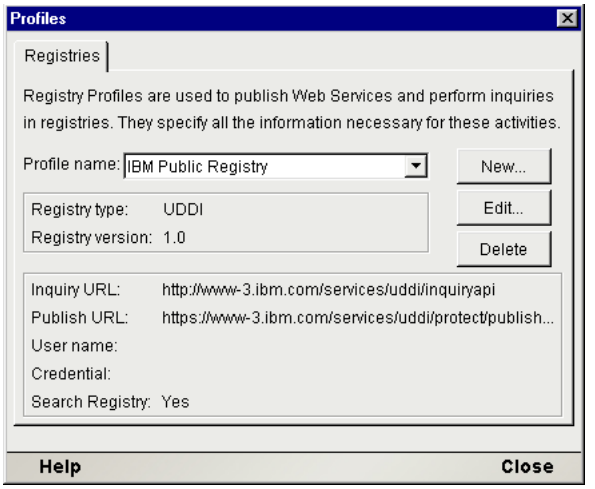
Registry Manager には、次の機能が含まれています。

- レジストリの追加 / 削除
- 検索プロセスに含めるレジストリの選択
- 特定のレジストリで選択した企業に関するビジネス情報の表示
- 特定の企業が提供している Web サービスに関する情報の表示
- (オプションとして、拡張クエリパラメータを使用することによる) レジストリまたはレジストリのグループ内での企業またはサービスの検索
- レジストリへの新しいサービスの公開

レジストリは URL によって指定され、ローカルまたは Web ベースにすることができます。レジストリは、e の［Profiles］ダイアログボックスから追加または削除できます。

➢ レジストリを編集または削除する

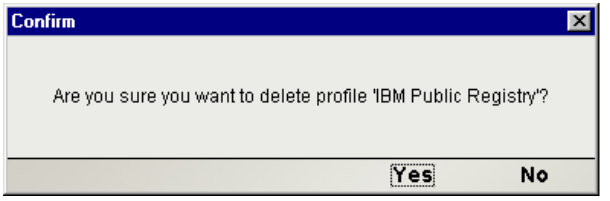
1 exteNd Composer メインメニューバーから、［**Tools**］>［**Profiles...**］の順に選択します。［Profiles］ダイアログボックスが表示されます。



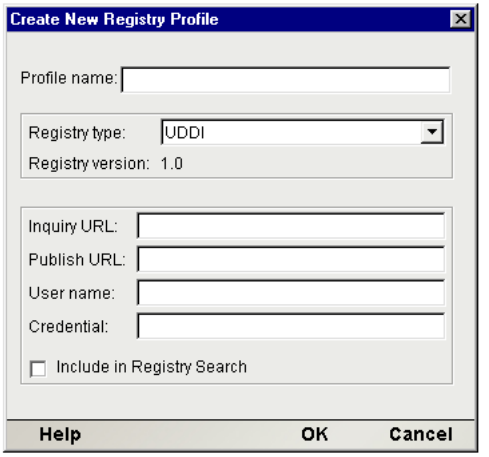
2 既存のエントリを編集する場合は、［Profile name］プルダウンメニューからそのエントリを選択し、［**Edit**］ボタンをクリックします。［Edit a Registry Profile］ダイアログボックスが表示されます (次を参照)。選択したエントリを編集し、［Close］をクリックして保存します。

注記： 編集中に名前を変更した場合、新しいレジストリが作成されます。古いレジストリを維持しない場合は、削除してください。

3 既存のエントリを削除する場合は、［Profile name］プルダウンメニューからそのエントリを選択し、［**Delete**］ボタンをクリックします。選択したエントリを削除してよいかどうか確認するメッセージが表示されます。選択したエントリを削除し、［Close］をクリックして保存します。



4 また、新しいエントリを作成する場合は、［**New...**］ボタンをクリックします。［Create a New Registry Profile］ダイアログボックスが表示されます。



5 ［**Profile name**］フィールドに、プロファイルの名前を入力します (必須)。

6 プルダウンメニューから［**Registry type**］を選択します。デフォルトは［UDDI］です。

7 ［**Inquiry URL**］フィールドに、レジストリを問い合わせることができる URL を入力します (必須)。

8 ［**Publish URL**］フィールドに、新しいサービスをレジストリに公開できる URL を入力します。

9 アクセスを公開するためにレジストリプロバイダによって割り当てられている場合は、「**ユーザ名**」と「**アカウント情報**」を入力します。

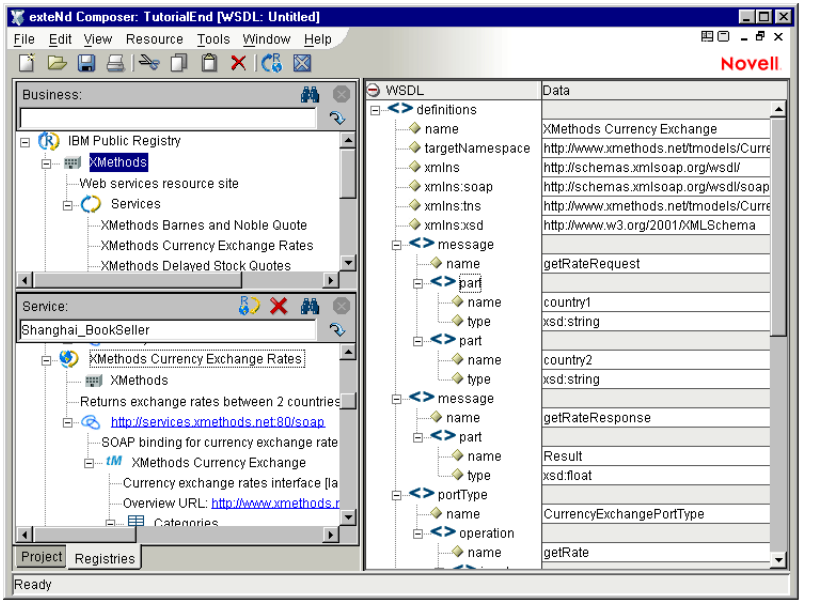
10 このレジストリがデフォルトの検索セットに自動的に含まれるようにする場合は、［**Include in Registry Search**］チェックボックスをオンにします。

11 ［**OK**］をクリックして、ダイアログボックスを閉じます。

このようにしてレジストリプロファイルを定義すると、Composer メインウィンドウのナビゲーションペインにある［Registry Browser］タブでレジストリプロファイルを使用できるようになります。また、サービスをレジストリに公開することも可能です。

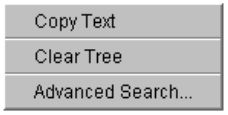
レジストリ参照

レジストリ参照機能は、Composer メインウィンドウのナビゲーションペインにある［Registries］タブから使用できます。ナビゲーションペイン内には、企業用 (上部) とサービス用 (下部) にそれぞれ 1 つずつ、合計 2 つのサブパネルがあります (図を参照)。



コンテキストメニュー項目

Registry Manager の各ペインに固有なコンテキストメニューは、Composer の使用時に利用可能です。［**Business**］のコンテキストメニューを表示するには、［Business］ペインのフィールドにカーソルを置き、マウスの右ボタンをクリックします。コンテキストメニューが表示されます (次を参照)。



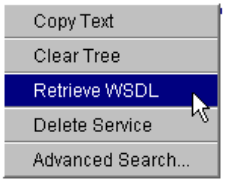
コンテキストメニュー項目の機能は、次のとおりです。

［**Copy Text**］— 現在選択されているビジネスツリーノードから別の領域またはファイルにテキストをコピーできます。

［**Clear Tree**］— 検索した結果として取得したビジネス情報のペインをクリアできます。

［**Advanced Search**］—［Set Browsing Criteria］ダイアログボックスで詳細な検索条件を設定できます。

「**サービス**」のコンテキストメニューを表示するには、[Service] ペインのフィールドにカーソルを置き、マウスの右ボタンをクリックします。コンテキストメニューが表示されます (次を参照)。



コンテキストメニュー項目の機能は、次のとおりです。

[**Copy Text**] — 現在選択されているツリーノードから別の領域またはファイルにテキストをコピーできます。

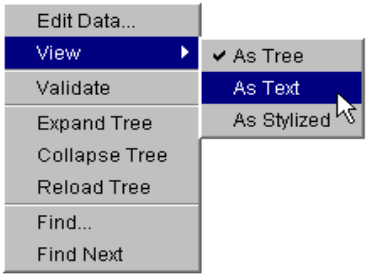
[**Clear Tree**] — 検索した結果として取得した情報の [Service] ペインをクリアできます。

[Retrieve WSDL] — レジストリから現在選択されているサービスに対して WSDL を取得できます。これは、[Retrieve] ボタンを使用して実行することも可能です。選択したサービスに WSDL 定義がない場合は、定義がないことを通知するメッセージが表示されます。

[Delete Service] — 企業またはサービスのレジストリで選択したサービスを削除できます。

[**Advanced Search**] — [Set Browsing Criteria] ダイアログボックスで詳細な検索条件を設定できます。

[**WSDL**] **コンテンツペイン**のコンテキストメニューを表示するには、ペインのフィールドにカーソルを置き、マウスの右ボタンをクリックします。コンテキストメニューが表示されます (次を参照)。



コンテキストメニュー項目の機能は、次のとおりです。

［**Edit Data**］— ペインに含まれている情報から別の領域またはファイルにテキストを変更できます。

［**View**］—［WSDL］ペインで情報を表示するための選択肢には、［**As Tree**］、［**As Text**］、および［**As Stylized**］の 3 つがあります。希望の選択肢をクリックすると、その情報がペインに表示されます。

［Expand Tree］— すべてのノードをペインに表示します。

［Collapse Tree］— ルートノード以外のすべてのノードをペインで非表示にします。

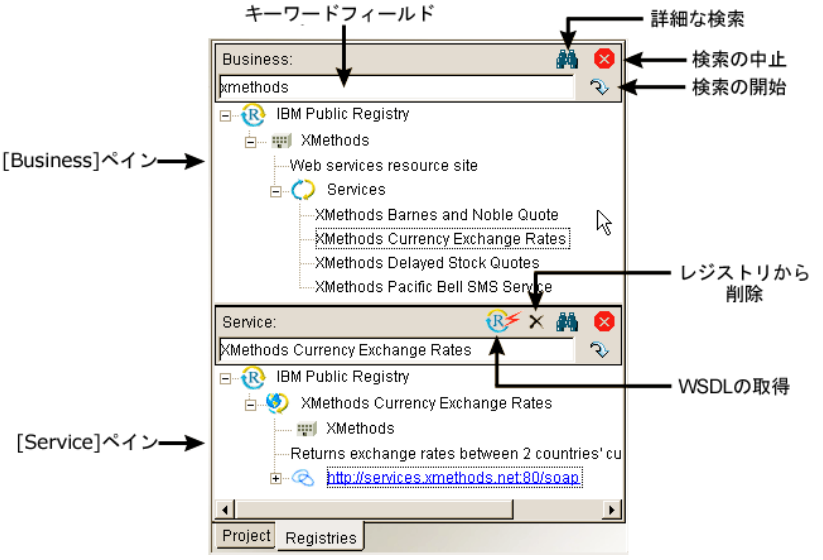
［Reload Tree］— 元のツリーをロードできます。

［Find］— ツリー内で特定の語または語の一部を、ダイアログボックスから検索できます。

［Find Next］— ツリー内で次の語または語の一部を、ダイアログボックスから検索できます。

アクションボタン

［Business］ペインおよび［Service］ペインのさまざまなアクションボタンの場所は、次の図のとおりです。



企業別に検索

企業 (1 つまたは複数) の検索は、[Business] の隣にあるテキストフィールドに完全または部分的な企業名を入力し、[Search] ボタン (または下向き矢印のような形の [Go] ボタン) をクリックするだけで行えます。一致した企業のリストは、ツリー表示形式で表示されます。ツリーのトップレベルにある各ノードはレジストリ、レジストリの各チャイルドは企業名、各企業の下は説明、カテゴリ、およびサービスで構成される詳細な情報です。また、複数の企業名を縦線 (パイプ文字) で区切って入力し、複数の企業グループを検索することもできます (例 : Silverton|Silicon)。

➢ キーワードで企業を検索する

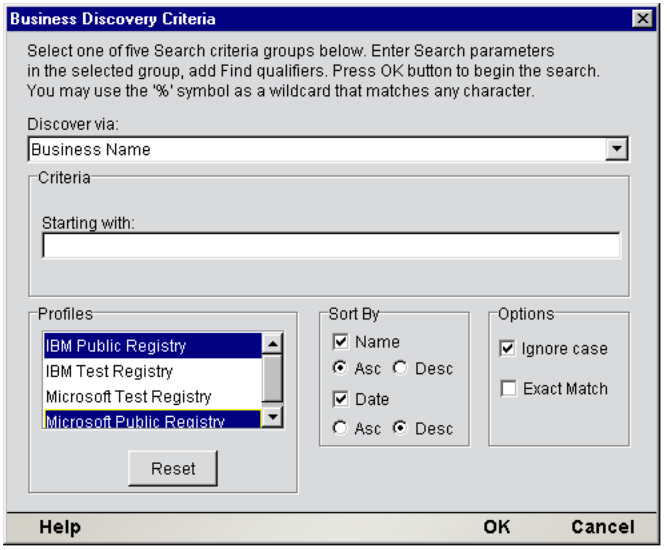
1. 企業名や部分的な名前 (または別のキーワード) による検索を行うには、キーワードフィールドにテキストを入力し、[Go] ボタン (下向き矢印のような形) をクリックします。検索が開始されます。検索が実行されている間、(通常はグレー表示されている) [Abort] ボタンは赤色になります。
2. 検索には、数分かかる場合があります。完了前に検索を中断する場合は、[**Abort**] ボタンをクリックします。部分的な検索結果が [Business] ペインに表示されます。
3. 検索が完了するまで待機します。結果が [Business] ペインに表示され、[Abort] ボタンの外観が通常のグレー表示 (無効) になれば、検索は完了です。

➢ 詳細な検索条件を設定する

1. 詳細な検索条件を設定する場合、テキストフィールドには何も入力せず、単に [Advanced Search] ボタン (双眼鏡のような形) をクリックします。

Advanced Search button

次のダイアログボックスが表示されます。



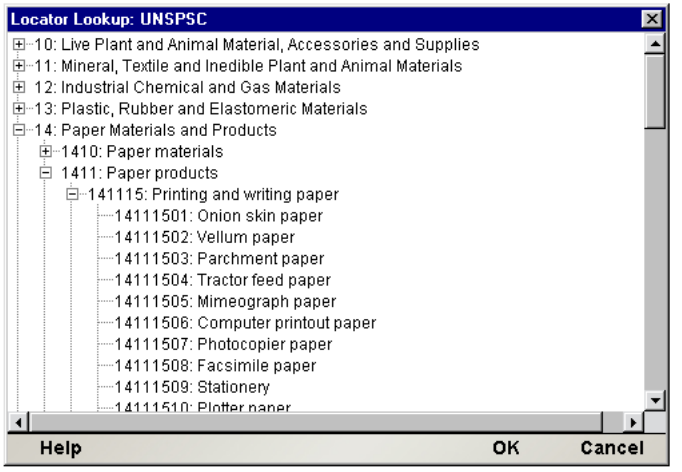
2 ラジオボタンがあることから示されるように、検索条件グループは一度に 1 つしか選択できません。使用できるオプションは、次のとおりです。

［**Business Name**］：［Business Name］の隣にあるラジオボタンをクリックします。［**Starting with**］の隣にあるテキストフィールドに、完全または部分的な企業名、あるいは縦線 (|) で区切った名前のリストを入力します。

［**Identifier**］：このオプションを選択した場合、新しいプルダウンメニューが表示されます。プルダウンリストから、［D-U-N-S］または［Thomas Register］(カタログ名) のいずれかを選択します。［**Starting with**］の隣にあるテキストフィールドに、(部分的または完全な) カタログからのキーを入力します。このエントリには、数値およびダッシュを含めることができます。

［Locator］：このオプションを選択すると、(ロケータと呼ばれる) 新しいプルダウンメニューが表示されます。プルダウンリストから、［NAICS］(North American Industry Classification System)、［UNSPSC］(United Nations Standard Products and Services Classification)、または［GEO］(geographical) のいずれかを選択します。［NAICS］または［UNSPSC］を選択した場合は、［Starting with］の隣にあるテキストフィールドに、(部分的または完全な) カタログからのキーを入力します。このエントリには、数値を含めることができます。［GEO］を選択した場合は、国 (地域) の省略形を入力します。

または、コントロールの右端にあるボタンをクリックして「キーピッカー」を表示し、入力済みリストで完全または部分的なキー名をダブルクリックします (次を参照)。



［Service Type Name］：これにより、特定の UDDI *tModel* に関連付けられている企業を検索できます。［Starting with］の隣にあるテキストフィールドに、この *tModel* に対するキーワードを入力します。

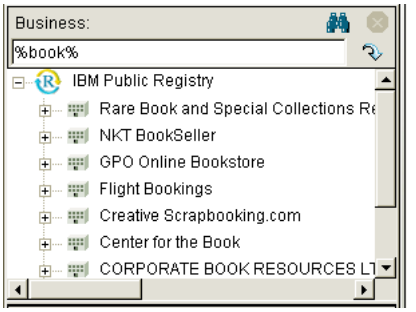
［Discovery URL］：［Starting with］の隣にあるテキストフィールドに、URL に対する IP アドレスまたは IP アドレスの一部を入力します。

3 ［Find Qualifiers］で、検索とソートのオプションを選択します。［検索オプション］では、適切なチェックボックスをオンにすることによって、［Ignore Case］または［Exact Match］、あるいはその両方を選択できます。［Sort Criteria］では、どのようにソートするか (名前別または日付別、Asc (昇順) または Desc (降順)) を選択できます。昇順で名前別 (アルファベット順)、または降順で日付別 (数字順) にソートする方法が最も一般的です。同じ名前の企業が含まれているグループ内では、日付別にソートすると便利です。

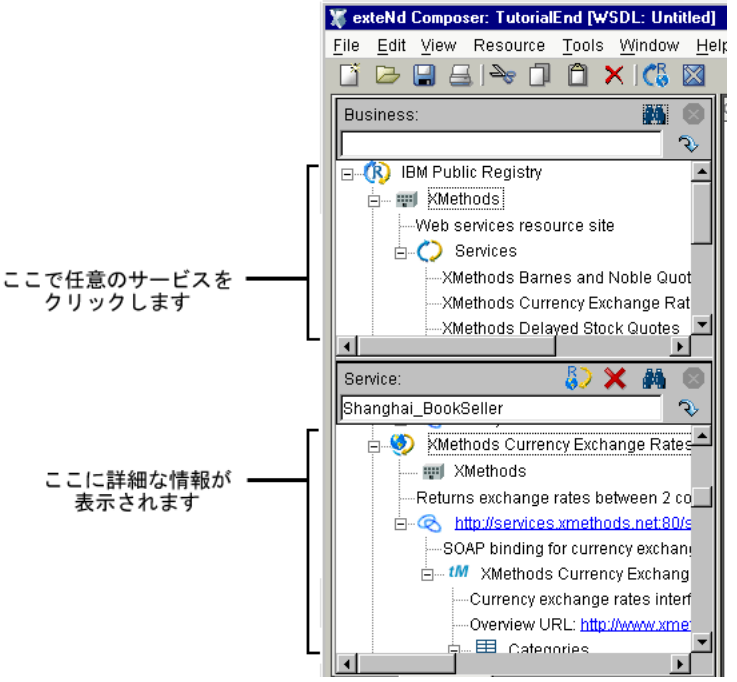
4 この検索に対して使用するレジストリを選択します。［Search Profiles］コンボボックスには、検索できるレジストリのリストが含まれています。すでに選択されているものは、［プロファイル］ダイアログボックスで選択した可能性のあるレジストリです (前の説明を参照)。ただし、リスト内でレジストリの 1 つまたはすべてを選択すると、これを上書きしてしまうことがあります。元 (デフォルト) のレジストリに戻す場合は、下にある［Reset］ボタンをクリックします。

5 ［OK］をクリックします。ダイアログボックスが閉じます。

6 ［Business］ペインの［Go］ボタンをクリックして、検索を開始します。



検索の後、一致した企業のツリーが［Business］ペインに作成され、［Service］サブペインはクリアされます。［Business］ツリーでサービスエントリをクリックすると、サービスの詳細情報 (バインディングなど) が下部にある［Service］ペインにツリー形式で表示されます (次を参照)。



サービス別に検索

サービス (または関連サービスのグループ) の検索は、［Service］の隣にあるテキストフィールドに完全または部分的なサービス名あるいはキーワードを入力し、［Search］ボタン (または下向き矢印のような形の［Go］ボタン) をクリックするだけで行えます。一致したサービスのリストは、ツリー表示形式で表示されます。ツリーのトップレベルにある各ノードは検索されたレジストリ、レジストリのすぐ下にある各チャイルドはサービス名で、サービスノードのチャイルドにはサービスに関連付けられている企業名、サービスの説明、およびサービスのバインディングで構成される詳細な情報が含まれます。

UDDI 検索でのワイルドカード

Composer レジストリ検索エンジンでは、ワイルドカードシンボルとしてパーセント記号 (%) の使用をサポートしており、この記号は 1 つまたは複数の任意の文字を意味します。これは、特定の語を「含んでいる」が、その語では「始まらない」企業名またはサービス名を検索する場合に特に便利です。

注記 : デフォルトの検索論理は「Start With」です。このため、「Books」を検索すると、「BooksRUs」は見つかりますが、「ABC Booksellers」や「Used Books」は見つかりません。この動作に優先する方法は、代わりに「%Books%」で検索することです。これらが 3 つとも見つかります。

Composer レジストリ検索エンジンでは、論理 OR シンボルとして縦線記号 (|) の使用もサポートしており、この記号の使用は「複数の語の組み合わせを含むヒットを検索する」ことを意味します。キーワードは、このようにして任意の数だけ連結させることができます。

```
%Booking% | %Travel% | %Airline%
```

この例では、これらの語のうち「少なくとも 1 つ」を含む名前が、名前内での語の位置にかかわらず、すべて返されます。

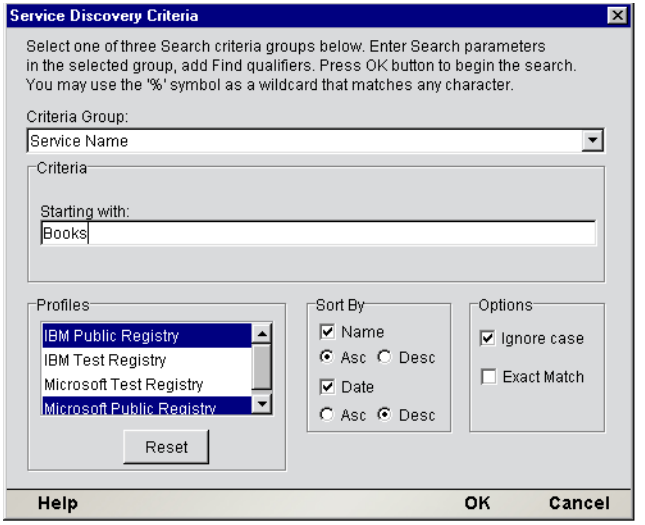
➢ キーワードでサービスを検索する

1. サービス名や部分的な名前 (または別のキーワード) による検索を行うには、キーワードフィールドにテキストを入力して、［**Go**］ボタン (下向き矢印のような形) をクリックします。検索が開始されます。検索が実行されている間、(通常はグレー表示されている) ［Abort］ボタンは赤色になります。
2. 検索には、数分かかる場合があります。完了前に検索を中断する場合は、［**Abort**］ボタンをクリックします。部分的な検索結果が［Service］ペインに表示されます。
3. 検索が完了するまで待機します。結果が［Service］ペインに表示され、［Abort］ボタンの外観が通常のグレー表示 (無効) になれば、検索は完了です。

➢ 詳細な検索条件を設定する

1 詳細な検索条件を設定する場合は、［Advanced Search］ボタン (双眼鏡のような形) をクリックします。

次のダイアログボックスが表示されます。



2 ラジオボタンがあることから示されるように、検索条件グループは一度に 1 つしか選択できません。使用できるオプションは、次のとおりです。

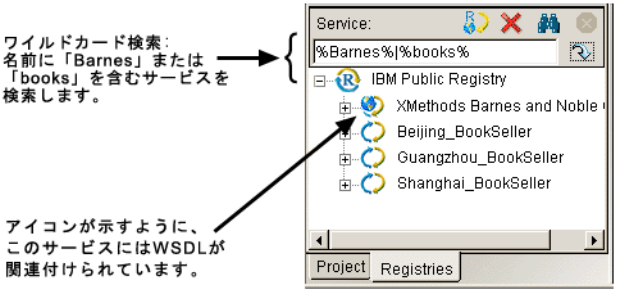
［**Service Name**］：［Service Name］の隣にあるラジオボタンをクリックします。［**Starting with**］の隣にあるテキストフィールドに、キーワードを入力します。

［Locator］：［Category］の隣にあるプルダウンリストから、［NAICS］(North American Industry Classification System)、［UNSPSC］(United Nations Standard Products and Services Classification)、または［GEO］(geographical) のいずれかを選択します。［NAICS］または［UNSPSC］を選択した場合は、［Starting with］の隣にあるテキストフィールドに、(部分的または完全な) カタログからのキーを入力します。このエントリには、数値を含めることができます。［GEO］を選択した場合は、国 (地域) の省略形を入力します。

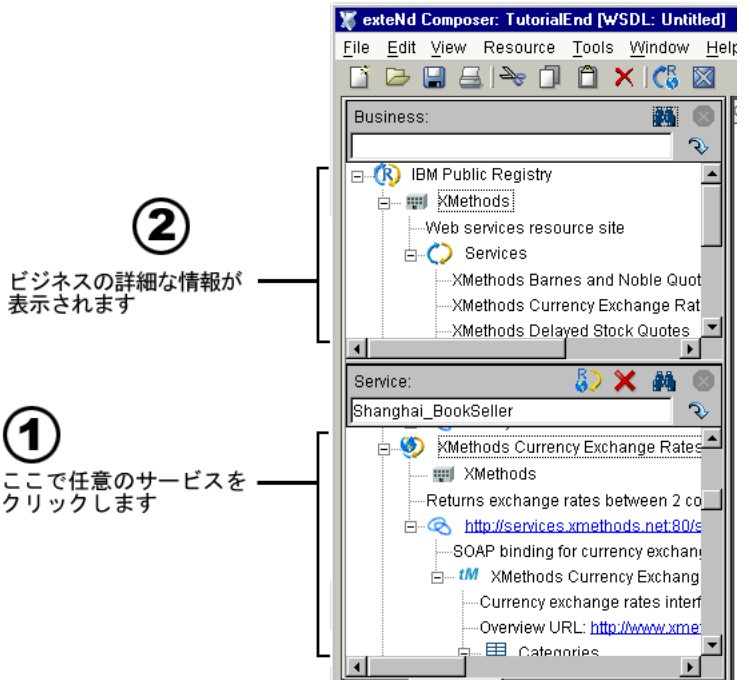
［Service Type Name］: 特定の tModel に関連付けられている企業を検索できます。［Starting with］の隣にあるテキストフィールドに、この tModel に対するキーワードを入力します。

3 ［Find Qualifiers］には、［Search Options］と［Sort Criteria］が含まれています。［Search Options］では、適切なチェックボックスをオンにすることによって、［Ignore Case］または［Exact Match］、あるいはその両方を選択できます。［Sort Criteria］では、どのようにソートするか (名前別または日付別、Asc (昇順) または Desc (降順)) を選択できます。昇順で名前別 (アルファベット順)、または降順で日付別 (数字順) にソートする方法が一般的です。同じ名前のサービスが含まれているグループ内では、日付別にソートすると便利です。

4 ［Search Profiles］には、検索できるレジストリのリストが含まれています。すでに選択されているものは、［Profiles］ダイアログボックスで選択した可能性のあるレジストリです。リスト内でレジストリの 1 つまたはすべてを選択すると、これを上書きしてしまうことがあります。選択されていた元のレジストリに戻す場合は、下にある［Reset］ボタンをクリックし、続けて［OK］をクリックします。



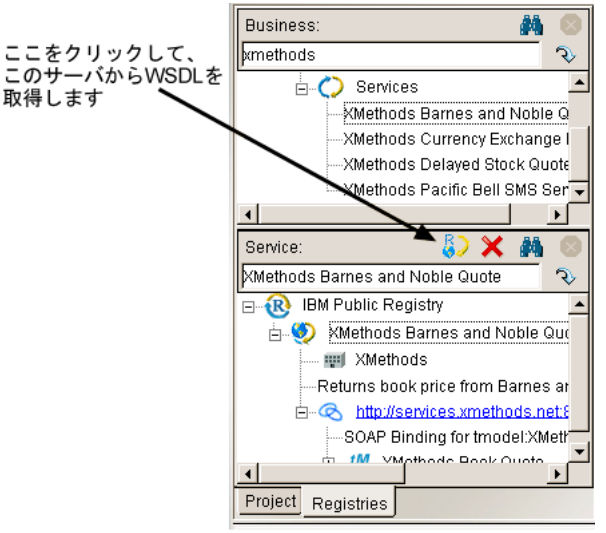
検索の後、一致したサービスのツリーが［Service］ペインに作成され、［Business］ペインはクリアされます。下部にある (［Service］) ツリーでサービスノードをクリックすると、企業の詳細情報が上部にある (［Business］) ペインにツリー形式で表示されます。



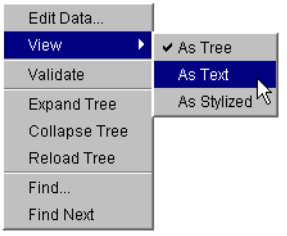
レジストリからの WSDL の取得

検索したサービスが見つかったら、レジストリからこのサービスの WSDL 定義を取得できます。希望のサービスノードを単に選択し、[Retrieve] ボタンをクリックします。サービスの定義が存在する場合、コンテンツペインに WSDL 情報がツリー形式で表示されます (図を参照)。サービスに対して WSDL が存在しない場合は、警告ダイアログボックスが表示され、その内容が報告されます。

注記： 特定のサービスリストに WSDL が存在するかどうかは、左側にあるサービスアイコンを確認することによって判断できます。中に地球が表示されているリングアイコンは、サービスに WSDL が存在することを意味します。また、地球が表示されていないリングアイコンは、WSDL Web サービスではないことを意味します。



マウスの右ボタンをクリックして、希望のビューを選択すると、コンテンツペインで情報をテキストまたは様式化された形式で表示できます。

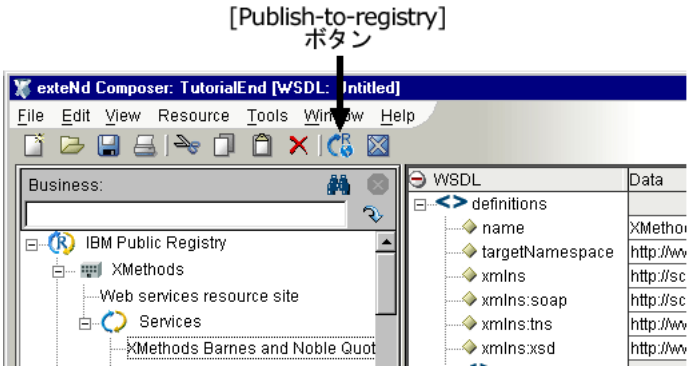


レジストリへの公開

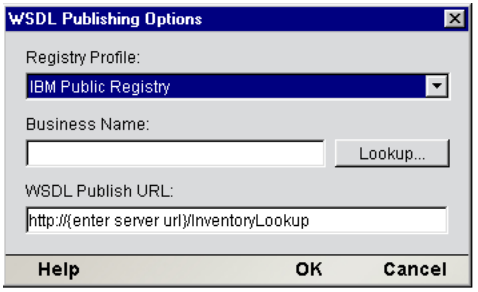
(Composer WSDL エディタを使用して) WSDL を作成すると、次に説明する手順に従って WSDL をレジストリに公開できます。

➢ レジストリに公開するには

1 ツールバーの［**Publish to Registry**］ボタンをクリックします (次を参照)。



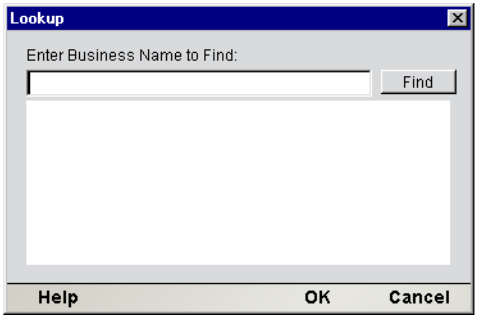
2 ダイアログボックス画面の［WSDL Publishing Options］が表示されます。



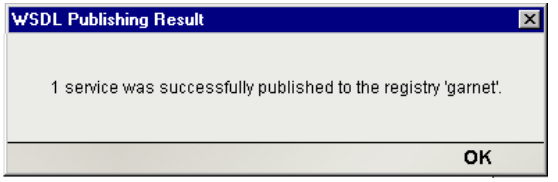
3 ［**Registry Profile**］: 公開するレジストリをプルダウンリストから選択します。

［**Business Entity**］: 企業をルックアップし、どの企業にサービスを関連付けるかを選択できます。［Lookup］ボタンをクリックすると、次のダイアログボックスが下に表示されます。

［**WSDL Publish URL**］: サービスが公開される URL を表示します。これは、必要に応じて編集できます。



4 ［OK］をクリックします。選択したレジストリにサービスが正しく入力された場合は、下に示すようなメッセージが表示されます。



15 配備のためのプロジェクトの準備

プロジェクトの設計、構築、およびテストを完了すると、次のステップはプロジェクトを実行するアプリケーションサーバに配備することです。

サポートされているアプリケーションサーバ環境での配備に関する特徴は、製品のバージョンおよびベンダにより多少異なります。従って、プロジェクトの配備の詳細については、別のドキュメントで説明します。Novell exteNd Application Server の『*Composer Enterprise Server ユーザガイド*』、WebSphere および WebLogic サーバの該当するガイドを参照してください。Composer サービスのカスタムサービストリガを作成する方法を含め、配備の詳細についてはこれらのガイドを参照してください。

注記： exteNd Workbench には、サーブレット、EJB、JSP、純粋な Java クラスなどさまざまなサービストリガの骨組みコードを自動的に生成できるウィザードがあります。この機能性については、該当するサーバガイドで詳しく説明されています。

この章では、配備の基本について簡単に述べます。手順ごとの説明など詳細については、該当するアプリケーションサーバに関する『*Composer Enterprise Server ユーザガイド*』を参照してください。

配備プロセス

exteNd Composer からサービスを配備する前に十分余裕を持って、以下を考慮する必要があります。

- **パッケージ要件** — サービスを Composer から直接アプリケーションサーバ環境に個別に配備するか、複数のサービスを含むプロジェクト全体を WAR ファイルまたは EAR ファイルにパッケージ化しますか？
- **トリガの必要性** — サービスの実行方法は？トリガオブジェクトとしてサーブレット、EJB、カスタム Java クラス、またはキューで受信される要求を「リスン」する (JMS の) MessageListener オブジェクトを選択できます。

- **着信データのパッケージ方法** — 着信データは、URI (HTTP GET) に追加された URL エンコードの param/value の対の形式にしますか？サービストリガにより、マルチパートの MIME 添付ファイルを使用して HTTP POST から着信 XML を処理しますか？複数のドキュメントを受け入れられるように SOAP サービスにしますか？
- **共有リソース** — ほかの Composer サービスで共有できるように、パブリックなリソースとして WSDL または XSD ファイルのようなリソースを配備しますか？

これらの考慮事項により、Composer で作成するサービスの配備方法が異なります。

配備オブジェクトのインストール方法

その他に、アプリケーションサーバで実際に配備を行う方法を考慮する必要があります。主なオプションとして、3 つの方法があります。

最初のオプションは Composer の Deployment Wizard を使用して、特定のプロジェクトの配備オブジェクトを自動的にインストールする方法です。この方法は、比較的簡素なスタンドアロンのアプリケーションを配備する場合に便利です。

2 番目のオプションはご使用のアプリケーションサーバベンダが推奨する手順に従って、手動で配備オブジェクトをインストールする方法です。この方法は、単に共有リソースを更新する場合など、何らかの理由で配備プロセスに対して低いレベルのコントロールが必要な場合に便利です。また、システム管理者が後からファイルを取得できるステージング領域でオブジェクトを作成する必要がある場合にも適しています。

3 番目のオプションは exteNd Workbench を使用して配備を行う方法です。この方法は、大きなプロジェクトの一部として JSP ファイルなどほかの Java オブジェクトと Composer プロジェクトを WAR ファイルにまとめる必要がある場合に便利です。

Web サービスのサービストリガについて

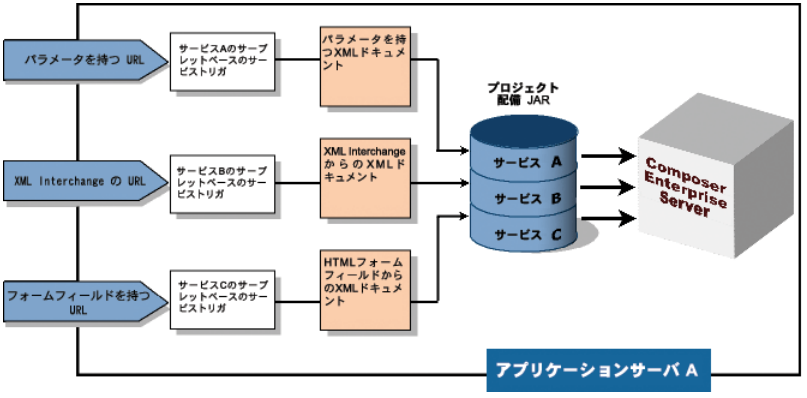
サービスは、アプリケーションサーバ環境における実行の基本単位です。サービスを含めすべての exteNd オブジェクトはメタデータの指示として作成され、配備 JAR の中にある XML ファイルに保存されます。配備の際、サービスは適当なサーバトリガメカニズムに接続する必要があります。Deployment Wizard で、Web サービスの主要な 2 つのサービストリガである Java サーブレットおよび Enterprise Java Beans を生成できます。さらに、exteNd Composer Enterprise Server のフレームワークを使用すると、サービストリガをカスタマイズしたり、手動で作成したりできます。

注記： トリガに関する以下の説明は、Web サービスの場合です。JMS サービスは Web サービスとは異なる方法で実行されます。ただし、単一の配備可能なプロジェクトに Web サービスまたは JMS サービス、あるいはその両方が存在する場合があるため、いずれのサービスでも同じ Deployment Wizard が使用されます。

サーブレットベースのサービストリガ

Java サーブレットはサービストリガとして使用されます。サービストリガにより URI と特定の Composer サービスが関連付けられ、入力データが HTTP 要求から XML ドキュメントに変換されます。サービスは XML ドキュメントを入力データとして受け入れることができ、最終的にサービスが実行されます。また、EJB からサービスが間接的にトリガされるようにサーブレットを生成することもできます。EJB からサービスをトリガするには、EJB ベースのトリガをまず生成して (355 ページ「EJB ベースのサービストリガ」を参照)、それによって呼び出されるサーブレットを生成する必要があります。

サーブレットベースのサービストリガは、exteNd Deployment Wizard により自動的に作成されます。必要な場合は、これらのサーブレットベースのサービストリガを変更したり、Composer により作成されたサービストリガとは別のものを定義したりできます。次の図では、Deployment Wizard により生成されたサーブレットタイプに関する一般的なシナリオが説明されています。



サーブレットタイプ

Composer サービスのトリガについては、Params (URL/Form)、XML (MIME multipart)、XML (HTML form field)、SOAP、および XML (HTTP POST) という 5 つのサーブレットタイプがあり、Composer サービスでデータを送受信する方法を表しています。それぞれのサーブレットタイプは、入力データを受け入れる方法が異なります。相違点について、次の表で簡単に説明します。

サーブレットタイプ	データの受け入れ方法	メモ
Params (URL/Form)	URL エンコード形式で URI に追加されたURIパラメータ	このサーブレットにより、ノードの名前としてHTTP URI形式のパラメータ、テキストとしてその値を使用するメモリ内 XML ドキュメントが作成されます。1 つのパラメータに対して複数の値を持つことができますが、複数の入力ドキュメントが作成されるわけではありません。
XML (HTTP POST)	ポストされた XML ドキュメントのコンテンツ (XML Interchange アクション - Post など)	このサーブレットにより、HTTP POSTメソッドで送信されたXML ドキュメントが取得されます。HTML Form POST は parameter name/value の対を含む点がこのサーブレットと異なります。この種類の HTTP 接続の実際のペイロードは、加工されていない XML ドキュメントです。取引パートナーと XML ドキュメントを交換する場合に便利なメソッドです。
XML (MIME multipart)	HTTP multipart/form データポストメソッドからの XML ファイル (これにより、ユーザはファイルシステムを検索したり、XML ファイルを選択して送信したりできます)。	このサーブレットにより、「file」入力タイプのフィールドを含むマルチパートのエンコード形式のデータからサービスの入力ドキュメントが抽出されます。サーブレットは、「xmlfile」と呼ばれる XML ファイルを含むフィールド名を予期し、このパラメータが最初に発生した場合にドキュメントを抽出します。

XML (HTML form field)	XML が内部にポストされている形式のファイル。XML が「xmlfile」とラベルのついたフィールドの中に存在する必要があります。	このサーブレットでは、ポストされた形式のフィールドからサービスの入力ドキュメントが抽出されます。サーブレットは、「xmlfile」と呼ばれる XML ファイルを含むフィールド名を予期し、このパラメータが最初に発生した場合にドキュメントを抽出します。
SOAP トリガ	HTTP POST から着信する SOAP 要求にラップされた XML ペイロード。	このサーブレットによりSOAP要求がアンラップされ、XML の本文 (つまり、SOAP-ENV:Body 要素すべて) が Composer サービスに渡されます。

出力タイプ

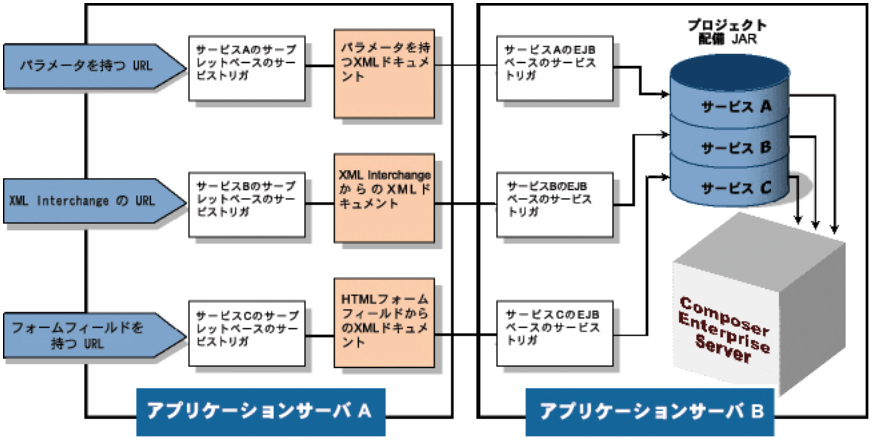
Composer サービスでは、もともと XML または HTML ドキュメントを出力したり、メッセージをメッセージキュー (JMS サービスの場合) に送信したりできます。HTML は、処理命令に応じて XML 出力ドキュメントの XSL 変換を通じ、Composer により即時に作成されます。

注記： SOAP ベースの Web サービスの例は、Composer の **\samples** ディレクトリにインストールされた TutorialEnd プロジェクト内にいくつかあります。

EJB ベースのサービストリガ

サービストリガの 2 つ目のタイプは、EJB (Enterprise Java Bean) です。EJB は、配備の柔軟性およびコンテナ管理によるトランザクション制御に優れているなど、さまざまな理由でサービストリガとして使用されます。生成された EJB サービストリガは、あらゆる Java アプリケーションで使用して exteNd Composer サービスを実行できます。

EJB の実行は、サーブレットの実行ほど直接的ではありません。そのため、Composer では EJB ベースのサービスをトリガするためにその前のセクションで参照されるタイプの Java サーブレットを生成できます。サーブレット -EJB ベースのサービストリガでは、1 つではなく、2 つのオブジェクトの間でこれらのタスクを分割しています。サーブレットが URI を関連付け、入力 HTTP データを XML に変換し、EJB を実行します。そして、EJB がサービスを実行します (EJB はサーブレットと別のサーバにも存在できますが、両者は Composer プロジェクトの JAR と同じマシンに存在する必要があります)。次の図にこのシナリオを示します。



アプリケーションベースのサービストリガ

Composer で生成されたサービストリガオブジェクトを使用する以外の方法として、Composer の外で独自のサービストリガオブジェクトを作成するか、Composer サービスの使用が必要な別のアプリケーションにサービストリガの機能性を統合する方法があります。このようなアプリケーション指向のいずれのトリガオプションでも、exteNd オブジェクトにインタフェースを記述する exteNd Composer Deployment Framework API が必要です。詳細については、『*Composer Enterprise Server ユーザガイド*』を参照してください。

JMS サービストリガ

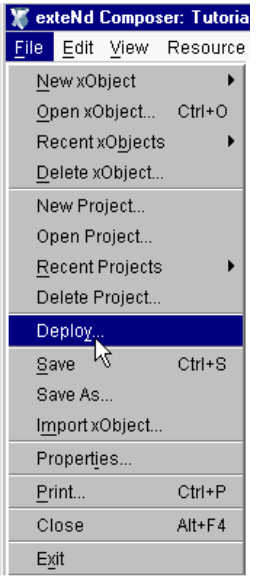
配備の際 JMS サービスにより、メッセージキューまたはメッセージトピックを持つ `onMessage()` ハンドラを登録する *MessageListener* オブジェクトのインスタンスが生成されます。JMS サービスのリスナオブジェクトは exteNd Composer Enterprise Server が実行されるたびに (つまりアプリケーションサーバが起動されるたびに) ロードされ、ブラウザコンソールからの管理制御の対象になります。また、JMS サービスは指定のサービスが複数のリスナオブジェクトに依存するように配備することもき、結果として性能および帯域幅が向上します。ただし、どのような場合でもサービスそのものはメッセージがキューに着信するまで、つまり `onMessage()` ハンドラが起動するまで実行されません。したがって、メッセージの着信は、サービスが応答するトリガイベントとして機能します。

本来 JMS サービスの寿命は無限で、サーバがアクティブな限りサービスもアクティブであるため、他の Composer サービスとは別に管理する必要があります。ブラウザベースの Administrative Console は、そのために提供されています。詳細については、『*JMS Connect ユーザガイド*』を参照してください。

注記： この説明は、exteNd Composer JMS Connect がインストールされている場合にのみ適用されます。

Composer からの配備プロセスの開始

次の図に示すように、Composer の［**ファイル**］メニューから［**Deploy**］を選択すると、プロジェクトの配備を開始できます。



この操作により、配備プロセスを補助する Deployment Wizard の最初の画面が表示されます。Deployment Wizard では、サーバ名およびポート、適当なステージングディレクトリ、JAR ファイルに指定する名前、JAR ファイル内の配備コンテキスト、SOAP バインドスタイル (SOAP サービスを配備する場合) などの情報を入力するプロンプトが表示されます。

ウィザードの最後には、Composer により配備オブジェクトが選択したステージング領域に配置されるか、実際にプロジェクトがアプリケーションサーバに配備されます (いずれかを指定します)。

exteNd Developer Workbench からの配備プロセスの開始

必要な場合は、Workbench を使用するとプロジェクトを EAR アーカイブにパッケージできます。Workbench を起動して、Workbench プロジェクトを開き、サブプロジェクトとして Composer の **.spf** ファイルを追加します。それから、必要に応じて Workbench EAR/WAR 配備オプションを使用します。詳細については、Novell exteNd Developer Workbench に関するマニュアルを参照してください。

詳細情報

次の点を含め、配備の詳細については『*Composer Enterprise Server ユーザガイド*』を参照してください。

- Composer の Deployment Wizard の使用方法
- SOAP 配備オプションの指定方法
- 実行するサービスの管理方法
- Composer Enterprise Server Framework APIを使用するカスタムトリガオブジェクトの作成方法

A XCCLASSPATH を使用した exteNd Composer への Java 拡張の追加

場合によっては、新しい Java クラスを exteNd 環境に統合することが便利であったり、必要性であったりすることがあります。たとえば、カスタムスクリプトから呼び出す重要な計算やデータ操作を実行する Java クラスがある場合や、JDBC Connect によって使用される新しいデータベースドライバを使いたい場合などがあります。

exteNd Composer で他の Java クラスを使用するには、exteNd の CLASSPATH で Java クラスにアクセスできなければなりません。この目的のために、XCCLASSPATH という名前の環境変数を使用することができます。

exteNd Composer への Java 拡張の追加

Composer の起動プログラムである xc.exe では、exteNd Composer ソフトウェアインストールの Bin ディレクトリに保存されている **xconfig.xml** ファイルのエントリから CLASSPATH 環境変数を作成します。この CLASSPATH に追加されるのが、環境変数の XCCLASSPATH です。Java クラスを使用する前に、XCCLASSPATH を更新し、この変数で示されている場所にクラスまたは jar ファイルをコピーする必要があります。

CLASSPATH を更新するには、次の 2 つの方法があります。

1 XCCLASSPATH 変数を設定します。

a. XCCLASSPATH 変数を設定します。

- Windows で［**スタート**］、［**設定**］、［**コントロールパネル**］、［**システム**］、［**詳細**］、［**環境変数**］の順に選択し、Java クラスにエントリを追加します。
- Windows で Autoexec.bat ファイルを編集し、必要な jar クラスを含むよう XCCLASSPATH 変数を設定します。

または

b. xc.exe を実行する前に、バッチファイルで XCCLASSPATH 変数を設定します。たとえば、バッチファイルには、次のような 2 行が含まれています。

- SET XCCLASSPATH=..lib\my.jar
- xc.exe

2 XCCLASSPATH で示されている場所に jar ファイルをコピーします (例 : xCopy my.jar **c:\exteNd\Composer\lib**)。

Composer Enterprise Server への Java 拡張の追加

各サーバのガイドには、クラスをクラスパスに追加する方法について詳しく説明したセクションがあります。(配備先のアプリケーションサーバの) 詳細については、『*exteNd Server ガイド*』を参照してください。

exteNd Composer に対する Java 仮想マシンの変更

exteNd に付属しているものとは異なる Java 仮想マシンで Composer を起動する場合は、**exteNd\Composer\Bin** ディレクトリの **xconfig.xml** ファイルを変更する必要があります。この場合、「VM」というタイトルの要素を RUNTIME のチャイルド要素として単に追加します。要素のデータには、使用する JVM へのパスが含まれていなければなりません。

```
<VM>d:\jdk1.2.2\jre\bin\java</VM>
```

JVM 起動パラメータを追加する場合は、「VM_PARAMS」という追加の要素を希望の JVM コマンドラインオプションとともに追加します。

```
<VM_PARAMS>-Xms64m -Xms128m</VM_PARAMS>
```

VM パラメータが **xconfig.xml** に追加され、Composer が起動すると、Java コンソールウィンドウは Composer のバックグラウンドで開いたままの状態になります。エラーメッセージが表示されますが、これらのほとんどは、Composer のメッセージログに表示されるものと同じです。

B 予約語

次の用語は、exteNd Composer の予約語であるため、ユーザ作成用語またはユーザ作成オブジェクトでは使用しないでください。

- Input、Temp、Output
- Input1、Temp1
- Input(n)、Temp(n)
- ERROR
- Math、Date、String、Array など (ECMAScript 予約語)
- theComponent
- SQLCODE、SQLSTATE、UPDATECOUNT、LASTSQL
- USERID PASSWORD

これらの用語以外にも、配備コンテキスト文字列、ユーザ変数名などでは「Java 言語キーワード」を使用しないことが推奨されます。Java の予約語は、次のとおりです。

「**Java キーワード**」

abstract	boolean	break
byte	case	catch
char	class	const
continue	default	do
double	else	extends
final	finally	float
for	goto	if
implements	import	instanceof

int	interface	long
native	new	package
private	protected	public
return	short	static
strictfp	super	switch
synchronized	this	throw
throws	transient	try
void	volatile	while

C 用語集

CDATA

CDATA セクション内の文字データがいずれも XML マークアップ言語として解釈されないようにする、XML ドキュメント内の宣言。これにより、各括弧 (< >) などの文字は、開始タグまたは終了タグの一部として解釈されることなく、XML ドキュメント内で使用できます。

Composer

Composer とは、XML 変換や外部データソース接続を実行する exteNd のサービスとコンポーネントを作成するために使用される視覚的生産ツールです。Composer では、非 XML 情報ソースや環境を有効にして XML エンコードデータの交換により相互動作するアプリケーションを作成します。

Connect

企業コネクタとは、インストール可能な Composer コンポーネントエディタ (および関連リソース) です。これを使用すると、ユーザに環境を視覚的に表示することにより、XML データを企業データソースと統合したり、XML をサポートしないレガシープラットフォームと統合したりできます。JDBC コンポーネントエディタは、企業コネクタの一例です。

Document Type Definition (DTD)

DTD は、XML ドキュメント内の要素が互いにどのように関連するかを指定します。これにより、ドキュメントに関するセマンティックルールだけでなく、有効なドキュメントのタイプであると見なされるためにその XML ドキュメントが準拠しなければならない要素も定義されます。

DOM

DOM (ドキュメントオブジェクトモデル) とは、ソフトウェアプログラムのメモリ内でオブジェクトとして作成された XML ドキュメントです。これにより、オブジェクトを操作するための標準的な方法が提供されます。Composer では、DOM は、XML ドキュメントと同義の場合があります。DOM は、単一のルートノードがある階層的なツリーとして表されます。

ECMAScript

(JavaScript に基づく) ECMAScript は、ホスト環境でオブジェクトを操作するための、オブジェクト指向のスクリプト言語です。ホスト環境として、Composer では、ECMAScript にさまざまなオブジェクト

(主に XML ドキュメント) へのアクセスを提供して処理できるようにします。代わりに、ECMAScript では、これらのオブジェクトで動作できる Java のような言語を提供します。

ERROR

サービスのコンテキストで実行しているすべてのコンポーネントがアクセスできるグローバル変数。ERROR の値は、RAISE ERROR アクションのエラー式によって設定されます。

Expression Builder

Expression Builder とは、有効な ECMAScript や XPath 構文を作成できるようにする、Composer 内のインタフェースです。

GET

XML Interchange アクションで使用される HTTP リクエストメソッド。GET メソッドによる方法では、Request-URI によって識別される情報 (エンティティの形式) であれば何でも取得します。

Group

Group は、XML ドキュメントで複数回発生する 1 つの要素をはさんだ固有な値のリストを表します。Groups は、Repeat ループが反復する回数 (つまり、固有な各値に対して 1 回) を決定することによって、Group Repeat ループを制御するために使用されます。グループに基づく Group Repeat 内の Map アクションは、固有な各値に対して 1 回のみ実行されます。Group は、Declare Group アクションを使用して作成されます。Groups は、繰り返す要素の XPath に関連付けられている別名によって参照されます。

Group(Detail)

Group(Detail) は、XML ドキュメントで複数回発生する 1 つの要素をはさんだすべての値のリストを表します。Group(Detail)s は、Repeat ループが反復する回数 (つまり、各値に対して 1 回) を決定することによって、Group Repeat ループを制御するために使用されます。Group(Detail) に基づく Group Repeat 内の Map アクションは、グループ内の各値に対して 1 回実行されます。Group(Detail) は、Declare Group アクションを使用して自動的に作成されます。Group(Detail)s は、テキスト「(Detail)」が付いているグループと同じ別名によって参照されます。Group(Detail) Repeat ループは、Group Repeat ループ内で常に使用される必要があります。

In 値

別のコードテーブルの異なる値に変換することを希望する、コードテーブルマップ内のデータ。

JDBC

Java Database Connectivity: リレーショナルデータベースデータにアクセスするための、Sun 設計の Java API。

JMS

Java Messaging Service: メッセージングの操作と構成要素から成る標準のセットを実装するための、Sun 定義の Java API。MOM (メッセージ指向ミドルウェア) 製品の中で最も普及しているものは、JMS 対応またはピュア JMS の実装です。

Map

ソースからターゲットにデータをコピーすることを目的として、データのソースとデータのターゲットの関連付けを示すために使用される一般用語。たとえば、XML Map コンポーネントは、ソース XML ドキュメントとターゲット XML ドキュメント間でデータを関連付けたり、コピーしたりします。Map アクションは、ソース要素とターゲット要素間、またはソース属性とターゲット属性間でデータを関連付けたり、コピーしたりします。

MessageListener

JMS サービスの配備時に作成されるオブジェクト。MessageListener オブジェクトは、キューに着信したメッセージによって JMS サービスが自動的に「発生」するよう、(前から存在する) メッセージのキューまたはトピックに登録されます。これにより、Composer サービスに対するメッセージングベースのトリガメカニズムが提供されます。

NodeList

1 つまたは複数のノードを含む明示的な XPath 式 (例 : Input.XPath ("INVOICEBATCH/INVOICE/INVOICEDATE")) によって返されるオブジェクト。NodeList は、ECMAScript 式で通常は使用されます。ノードリストに適用できるのは、ノードリストメソッドおよびノードリストプロパティのみです。任意のノードメソッドまたは要素メソッドをノードリストに適用するには、nodelist method item() を使用して単一のノードをまず選択する必要があります。

Out 値

関連付けられている In 値 (「In 値」を参照) に対して新しい値となる、コードテーブルマップ内のデータ。

POST

XML Interchange アクションで使用される HTTP リクエストメソッド。POST メソッドは、Request-Line の Request-URI によって識別されるリソースの新しいサブオーディネートとしてリクエスト内で囲まれたエンティティを発信元のサーバが受け付けるよう要求するために使用されます。POST メソッドによって実行される実際の機能はサーバによって決定され、通常は Request-URI に依存しています。

POST with Response

XML Interchange アクションで使用される HTTP リクエストメソッド。発信元のサーバから応答 XML オブジェクトが返されることを XML Interchange アクションで期待する点以外は、POST と同じです。

PUBLIC

XML DOCTYPE 命令に対するバリアント仕様。PUBLIC は、広範囲に及ぶ使用とアクセスを対象とした DTD を指定するために使用されます。

PUT

XML Interchange アクションで使用される HTTP リクエストメソッド。PUT メソッドは、囲まれたエンティティが指定した Request-URI に保存されることを要求します。

ROW TARGET

SQL ステートメントによって返された行のペアレント要素として機能する XPath の場所。返される各行に対して ROW TARGET 要素が作成され、行の各列は ROW TARGET のチャイルド要素になります。

SYSTEM

XML DOCTYPE 命令に対するバリアント仕様。SYSTEM は、XML ドキュメントによる私的使用を対象とした DTD を指定するために使用されます。

UDDI

Universal Description, Discovery and Integration: 独自のサービスを説明したり、他の会社のサービスをオンラインで見つけたりする方法を企業に提供する、パブリックレジストリ標準。

Unicode

世界中からの何千もの有用な文字が含まれた、2 バイト文字セット。「UTF-8」も参照してください。

URI

Uniform Resource Identifier: URL 内の情報に細部までアクセスすることを許可する、URL の拡張。

URL

URL とは、インターネットでリソースの検索やリソースへのアクセスを行うために使用されるテキスト文字列の構文およびセマンティックを指定する Uniform Resource Locator です。基本的な URL は、通信プロトコルを識別するスキームと、リソースを識別するスキーム固有の部分から構成されます (つまり、< スキーム >:< スキーム固有の部分 >)。

Userfunc

次に来る用語が XPath の一部として評価されないカスタムスクリプト関数であることを示すために XPath 式内で使用されるキーワード。

UTF-8

多くの一般的なテキストエディタで XML を作成できるようにするために最初の 128 文字が 7 ビット ASCII 文字と互換性がある Unicode 文字セットの文字エンコードスキーム。

W3C

http://www.w3.org の World Wide Web Consortium。進化を促進して相互運用を確実にする共通プロトコルを開発することによって、World Wide Web の可能性を最大限まで導くために組織化された標準化団体。

WSDL

Web Services Definition Language: 企業のサービスを XML で記述するための標準。WSDL によって、Web サービスプロバイダは、リモートパートナーと自動化または半自動化された方法によりオンラインで e ビジネスを行うために必要な手段を理解できるようになります。

XML カテゴリ

XML カテゴリには、XML テンプレートが含まれます。XML カテゴリは、プロジェクト内で使用される XML テンプレートを整理するために作成します。

XML サンプルドキュメント

XML サンプルドキュメントとは、運用環境でアプリケーションが処理するデータの代表的なモデルです。サンプルドキュメントは、正確なアクションモデルを作成できるようにするために使用されます。

XML テンプレート

XML テンプレートには、サンプルドキュメント、ドキュメント定義、および特定のドキュメントタイプに関連付けられている XML スタイルシートが含まれます。XML テンプレートは Composer で作成し、作成するコンポーネントの入力および出力を記述するために使用します。

XML ドキュメント定義

XML ドキュメント定義とは、W3C によって作成された標準的な検証方法です。これによって、ドキュメントのルール (どの要素が含まれているか、どんな構造上のリレーションシップが要素間に存在するかなど) が定義されます。 ドキュメント定義により、受信するアプリケーションに受信データの記述が組み込まれていない場合に、データを検証することができます。

XML メタデータ

exteNd Composer で作成された xObject は、XML ファイルとしてディスクに保存されます。これらのファイルは、プロジェクトのメタデータと呼ばれることがあります。

xObject

xObject とは、すべての exteNd データ統合サービスの構成要素です。xObject には、サービス、コンポーネント、リソース、および XML テンプレートが含まれます。

XPath

XPath とは、XML ドキュメントの部分をアドレッシングするための言語です。これは W3C 推奨の標準で、exteNd のプライマリ XML アドレッシング言語として使用されます。

XPointer

XPointer とは、インターネットメディアタイプのリソースを検索する URI-reference 内でフラグメントを識別するために使用される言語です。

XSL

XSL とは、XML ドキュメントを他の XML ドキュメントに変換するためのスタイルシート言語です。

XSL スタイルシート

XSL スタイルシートでは、XML ドキュメントの表示プロパティが定義されます。スタイルシートは、Composer の外部で作成または取得します。スタイルシートは、Web ブラウザに表示するページを作成しているコンポーネントに対して役立ちます。

アクション

アクションは、プログラミングステートメントに類似しており、パラメータの形式で入力を受け付け、特定のタスクを実行します。

アクションモデル

アクションモデルとは、連続するアクションの視覚的な表示です。アクションモデルは、コンポーネントエディタ内にあります。

アニメーション表示

問題をデバッグしたり、新しい入力をテストしたりするために、Composer でコンポーネントを視覚的に実行する段階的なプロセス。

エンティティ

XML ドキュメントのエンティティとは、何か他のものを表す、特別にフォーマットされたプレースホルダです。つまり、エンティティへの参照は、それらのエンティティのコンテンツで置き換えられます。XML および HTML には、「>」に対する `>` や、「<」に対する `<` など、特定の定義済みエンティティがあります。また、ユーザ定義エンティティを使用することも可能です。

カスタムスクリプト

Composer プロジェクトのユーザ定義 ECMAScript 関数の集合。

グループ名

Repeat アクションに対してグループを定義するために使用される XPath 式の別名。

コードテーブル

コードテーブルでは、一般的に使用されるビジネスコードとそれらに関連付けられている説明が保存されます。2 つのコードテーブルは 1 つのコードテーブルマップと組み合わさって機能し、あるセット (値) から別のセット (コード) への変換が生成されます。

コンテンツエディタ

データの XML 要素レベル変換を実行するために設計された Map アクションで使用可能なダイアログボックス。コンテンツエディタでは、文字または文字の位置によってデータを接続または再接続したり、定数を挿入したり、関数を適用したりできます。

コンポーネント

コンポーネントとは、1 つまたは複数の XML ドキュメントを入力として受け付け、入力で動作するアクションの集合を使用し、XML ドキュメントを出力として返すオブジェクトです。さまざまなタイプのコンポーネント (「Connect」を参照) は、リレーショナルデータベースのような外部の非 XML データソース、3270/CICS トランザクションなどと動作することもできます。

サービス

サービスは、作成するさまざまなコンポーネントを結合して、アプリケーションサーバ環境内に作業の論理単位を作成するために使用されます。これは、要求 XML ドキュメントを使用して開始され、応答 XML ドキュメントを必要とします。実行される作業と、各コンポーネントの責任は、アプリケーションの設計によって異なります。サービスでは、さまざまなコンポーネントが連続的または条件的、あるいはその両方で通常は実行され、他のサービスを実行することもできます。その他のサービスレベルタスクには、一般的なエラー処理や実行時のログ記録の操作が含まれる場合があります。

サービストリガ

サービストリガとは、Composer からプロジェクトを配備する際に作成される Java サーブレットまたは Enterprise Java Bean です。これにより、サービスが exteNd Server に送信され、実行されます。また、サービストリガは URL と関連付けられており、サービスへの入力 (このサービスによってトリガされる) として着信データを XML ドキュメントに変換します。

システム

{ANY} などの不変の XML テンプレートに対する予約された XML テンプレートカテゴリ。

出力

出力とは、コンポーネントがどのようにデータを返すかを記述するために使用される用語です。コンポーネントを作成するときに XML テンプレートを選択することによって、そのコンポーネントに対する出力の形式を指定します。すべてのコンポーネントでは、出力として単一の XML ドキュメントを返します。

出力 DOM

コンポーネント (またはサービス) の出力 DOM とは、そのコンポーネントで実行された変換の結果を含む XML ドキュメントです。出力 DOM は、コンポーネントを呼び出したサービス (またはコンポーネント) に返される XML ドキュメントです。コンポーネントおよびサービスでは、1 つの出力 DOM のみを返すことができます。

スキーマ

スキーマは、データの検証に役立つ XML ドキュメント定義に類似しています。しかし、ドキュメント定義とは違って、スキーマは、拡張可能な言語 (つまり、XML) で作成されます。スキーマを使用すると、どの要素名がドキュメントで許可され、また、各要素内でどのサブ要素、属性、およびリレーションが許可されるかを正確に定義できます。

接続

接続とは、外部データソースとの通信、または HTTP 認証を使用するサーバとの通信を確立するために使用されるリソースです。

属性

属性とは、要素に関連付けられている XML ドキュメントの一部で、要素に関する説明的な情報が提供されます。また、属性は、DOM 仕様のオブジェクトタイプでもあります。

ドキュメント

XML ドキュメントは、一般にドキュメントと呼ばれます。また、ドキュメントは、DOM 仕様のオブジェクトタイプでもあります。ドキュメントは、DOM と同意語として使用されることがあります。

ドキュメントハンドル

XML ドキュメントの DOM に割り当てられる名前。デフォルトのドキュメントハンドルは、入力 XML テンプレートに対しては Input、Input1、Input(n)、Temp ドキュメントに対しては Temp、Temp1、Temp(n)、すべてのサービスおよびコンポーネントの結果に対しては Output です。カスタムドキュメントハンドル名は、Component アクション (［Returned ID］フィールド)、Temp ドキュメント (［Identifier］フィールド)、および XML Interchange アクションを使用して作成できます。

入力

入力とは、コンポーネントがどのようにデータを受け入れるかを記述するために使用される用語です。コンポーネントを作成するときに 1 つまたは複数の XML テンプレートを選択することによって、そのコンポーネントに対する入力の形式を指定します。すべてのコンポーネントでは、1 つまたは複数の XML ドキュメントを入力として受け付けます。

入力 DOM

コンポーネントの入力 DOM とは、外部データソースにマップしたり、別の XML ドキュメントタイプに変換したりすることを希望する XML エンコード情報を含む XML ドキュメントです。入力 DOM は、コンポーネントを呼び出すサービス (またはコンポーネント) によってそのコンポーネントに渡されます。コンポーネントでは、1 つまたは複数の入力 DOM を受け付けることができます。サービスでは、サービストリガからの入力 DOM を 1 つだけ受け入れることができますが、別のコンポーネントからは複数受け付けることが可能です。

ネームスペース

異なる DTD からの名前を同じドキュメント内で結合できるよう、XML DTD で使用される名前が固有になるようにするメカニズム。

ノード

ノードとは、DOM を作成するために使用される基本的なオブジェクトです。DOM は、接続されているノードの集合で構成され、XML 要素、属性、コメントなどにすることができます。また、ノードは、DOM 仕様のオブジェクトタイプでもあります。属性、ドキュメント、および要素は、すべてノードであることに注意してください。

配備

exteNd Composer プロジェクトを運用のためにアプリケーションサーバ /exteNd Server 環境にパッケージ化してインストールするプロセス。

パブリック

カスタムスクリプト関数の属性。パブリックとは、パラメータとして式を受け入れるアクションから機能に直接アクセスできることを意味し、Expression Builder 選択リストに機能名を表示させます。非パブリック機能は、他の機能によってのみ呼び出すことができます。

プロジェクト

プロジェクトとは、XML 統合サービスを実行するために設計された exteNd オブジェクトの集合です。プロジェクトでは、作成しているアプリケーションのオブジェクトがすべて保持されます。

プロジェクト JAR

exteNd Composer プロジェクトを配備すると、プロジェクト内のすべてのオブジェクトは単一の JAR (Java Archive) ファイルに保存され、運用のためにアプリケーションサーバにインストールされます。

プロジェクトファイル

プロジェクトを作成すると、Composer によって、選択したフォルダに < *プロジェクト名* >.*spf* として保存されるプロジェクトファイルが作成されます。プロジェクトファイルには、プロジェクトの開始情報が含まれます。

プロジェクト変数

Composer サービスのプロジェクトレベルで置換可能なパラメータを追加するために Composer で作成された、名前と値のペア。プロジェクト変数は、配備されたアプリケーション内で置き換えると、プロジェクト全体を再配備することなく動作を変更できる別のファイル (project.xml) で維持されます。プロジェクト変数は、PROJECT/USERCONFIG と呼ばれるシステム DOM からアクセスされます。

別名

Repeat アクションで使用するための XPath 式によって識別される要素に与えられる名前。別名は、XPath 式に一致する次の繰り返し要素が Repeat ループの各反復で別個に処理されるようにします。

マークアップ

XML ドキュメントのマークアップは、予約されたメタデータの記号や構成要素 (start-tag、end-tag、empty-element tag、comment、processing instruction など) で構成されます。XML は、角括弧のような特定のトークンに依存して、特殊な意味を持ちます。このような記号が XML データに表示されている場合、XML ドキュメントは一般的に無効となります。記号をエンティティ形式 (「エンティティ」を参照) に変換することは、XML データとして「予約された文字」を渡す 1 つの方法です。別の方法は、CDATA セクション (「CDATA」を参照) でマークアップを折り返すことです。

マッピングペイン

マッピングペインは、コンポーネントの Map アクションを使用して情報を交換できるデータのソースを表します。マッピングペインには、現在のコンポーネントのサンプル入力 / 出力ドキュメントに関連付けられている DOM が表示され、リレーショナルデータベースや 3270 画面トランザクションなどの外部データソースの表記も表示されます。

文字データ

XML ドキュメント内に含まれるデータ。文字データは、マークアップ以外のすべてのデータです。XML ドキュメントの文字データは、Unicode 文字セットの文字から構成されます。「CDATA」も参照してください。

要素

要素とは、ドキュメントのデータの大部分を含む、XML ドキュメントの基本的な部分です。また、要素は、DOM 仕様のオブジェクトタイプでもあります。

リソース

リソースとは、専用の操作を実行してサービスとコンポーネントがタスクを実行できるようにする xObject です。リソースのタイプには、コードテーブル、コードテーブルマップ、接続、およびカスタムスクリプトがあります。配備時に、リソースは、プロジェクト JAR から別々に配備することのできる XML DTD/ スキーマファイルおよび XSL スタイルシートも参照します。

索引

記号

数字

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

の

は

ひ

ふ

へ

ほ

ま

む

め

も

ゆ

よ

ら

り

れ

ろ

わ