FileMaker® Server 8 Advanced

カスタム Web 公開ガイド

FileMaker, Inc.
5201 Patrick Henry Drive
Santa Clara, California 95054

第 01 版

目次

第 5 章
FileMaker XSLT スタイルシートの開発 47

第 1 章
カスタム Web 公開の概要

FileMaker® Server Advanced では、次の方法で FileMaker データベースをインターネットまたはイントラネット上に公開できます。

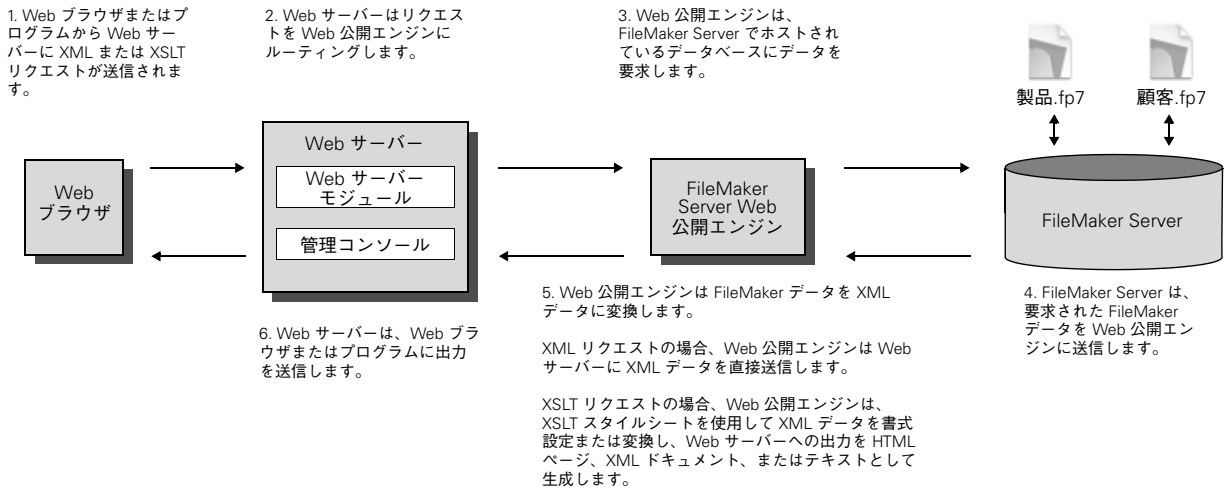
- XML（Extensible Markup Language）を使用したカスタム Web 公開
- XSLT（Extensible Stylesheet Language Transformations）を使用したカスタム Web 公開
- インスタント Web 公開。『FileMaker インスタント Web 公開ガイド』を参照してください。

XML および XSLT を使用したカスタム Web 公開では、Web ページのデザインと機能を選択および制御できます。公開されるデータベースは FileMaker Server でホストされ、カスタム Web 公開を利用可能にするために FileMaker Pro がインストールまたは実行されている必要はありません。

インスタント Web 公開と、XML および XSLT を使用したカスタム Web 公開をサポートするため、FileMaker Server では、FileMaker Server Web 公開エンジンと呼ばれるソフトウェアコンポーネントのセットが使用されます。Web 公開エンジンは、Web ユーザのブラウザ、Web サーバー、および FileMaker Server の間の通信を処理します。Web 公開エンジンは XSLT プロセッサとして機能し、出力を HTML、XML、またはテキスト（vCard など）として Web サーバーに提供します。Web 公開エンジンからの出力は、Web サーバーによって Web ブラウザに提供されます。

Web ユーザがカスタム Web 公開ソリューションにアクセスするには、HREF リンクをクリックするか、または Web サーバーのアドレスと FileMaker クエリー文字列リクエストを指定した URL（Uniform Resource Locator）を入力します。この URL で、XML データにアクセスするか、または XSLT スタイルシートを参照できます。Web 公開エンジンは、クエリー文字列リクエストで指定された XML データ、または参照されている XSLT スタイルシートの結果を返します。

XML または XSLT を使用したカスタム Web 公開のための FileMaker Server Web 公開エンジンの使用



重要　Web 上でデータを公開する場合は、セキュリティがさらに重要になります。『FileMaker セキュリティガイド』のセキュリティガイドラインを参照してください。このマニュアルは、www.filemaker.co.jp/downloads から入手することができます。

このガイドについて

このガイドでは、読者が、XML と XSLT、Web サイトの開発、および FileMaker Pro を使用したデータベースの作成の経験があることを想定しています。このガイドでは、FileMaker Server 上での XML および XSLT を使用したカスタム Web 公開に関する次の情報を説明します。

- XML または XSLT を使用してカスタム Web 公開ソリューションを開発するための必要条件
- XML または XSLT を使用してデータベースを公開する方法
- Web ユーザがカスタム Web 公開ソリューションにアクセスするための必要条件
- FileMaker Server でホストされているデータベースから XML データを取得する方法
- FileMaker XSLT スタイルシートを開発する方法
- CDML ソリューションを FileMaker XSLT に変換する方法

重要 FileMaker に関するドキュメントの PDF は、www.filemaker.co.jp/downloads からダウンロードすることができます。

FileMaker Server Advanced のドキュメントには、次の情報も含まれます。

必要な情報	参照先
FileMaker Server のインストールと設定	『FileMaker Server 管理者ガイド』 『FileMaker Server Admin ヘルプ』
インスタント Web 公開、および XML と XSLT を使用したカスタム Web 公開の FileMaker Server 用のインストールと設定	『FileMaker Server Advanced Web 公開インストールガイド』
インスタント Web 公開	『FileMaker インスタント Web 公開ガイド』
カスタム Web 公開	『FileMaker Server Advanced カスタム Web 公開ガイド』（このマニュアル）
ODBC との JDBC ドライバのインストールと設定	『FileMaker ODBC および JDBC クライアントドライバのインストール』
ODBC と JDBC	『FileMaker ODBC および JDBC デベロッパーズガイド』

Web 公開エンジンを使用した動的な Web サイトの作成

Web 公開エンジンは、XML データの公開およびサーバーによって処理される XSLT スタイルシートを使用して、FileMaker Server にカスタム Web 公開機能を提供します。カスタム Web 公開には、次のような多くの利点があります。

- カスタマイズ: Web ユーザが FileMaker データを操作する方法や、Web ブラウザにデータを表示する方法を決定できます。
- データの交換: FileMaker XML を使用することで、FileMaker データを他の Web サイトやアプリケーションと交換できます。
- データの統合: FileMaker XSLT スタイルシートを使用することで、FileMaker データのサブセットを、他の Web サイトに統合したり、他のミドルウェアやカスタムアプリケーションを使用して統合することができます。Web ブラウザに FileMaker のレイアウト全体を表示する代わりに、データが別の Web サイトに属するかのように表示できます。
- セキュリティ : Web 公開エンジンの管理者である場合は、サーバーでホストされているすべてのデータベースに対して、インスタント Web 公開、XML を使用した Web 公開、または XSLT を使用した Web 公開を個別に有効または無効にすることができます。FileMaker データベースの所有者である場合は、各データベースに対して、インスタント Web 公開、XML を使用した Web 公開、または XSLT を使用した Web 公開へのユーザアクセスを制御できます。
- サーバーサイドのスタイルシート: クライアントサイドのスタイルシートでは、機密性の高いデータベースの情報が不正に検証される可能性がありますが、XSLT スタイルシートはサーバーサイドで処理されるため、これを防ぐことができます。

- 公開されるデータの制御とフィルタ: XSLT スタイルシートを使用することで、公開するデータベース情報のデータやタイプを制御およびフィルタして、データベースの不正使用を防止できます。また、データベース名やフィールド名などのメタデータを隠すこともできます。
- オープンスタンダードへの準拠: カスタム Web 公開ソリューションに対しては、ツール、リソース、および熟練した技術者がより豊富に揃っています。標準的な XML または XSLT の知識がある場合、使用する URL 構文やクエリー引数など、XML を使用したカスタム Web 公開に特有の詳細事項をいくつか学べば、すぐにソリューションの開発に取りかかることができます。
- CDML ソリューションからの移行支援: FileMaker CDML Converter ツールは、CDML フォーマットファイルを XSLT スタイルシートに変換する場合に役立ち、XSLT の学習にも役立ちます。CDML フォーマットファイルと XSLT スタイルシートは類似しており、違いは簡単に理解できます。XSLT スタイルシートは、CDML フォーマットファイルよりも複雑で強力です。

XML を使用したカスタム Web 公開について

XML を使用したカスタム Web 公開では、FileMaker データベースのデータに対してクエリーを実行して、さまざまな方法でデータを使用できます。適切なクエリーコマンドと引数を指定した HTTP リクエストを使用して、FileMaker データを XML ドキュメントとして取得してから、XML データを他のアプリケーションで使用したり、XML データに XSLT スタイルシートを適用できます。第 3 章「Web 公開エンジンを使用した XML データへのアクセス」を参照してください。

XSLT を使用したカスタム Web 公開について

XSLT を使用したカスタム Web 公開では、XML データを変換、フィルタ、または書式設定して、Web ブラウザや他のアプリケーションで使用できます。XSLT スタイルシートを使用して、FileMaker XML 文法と他の XML 文法の間でデータを変換して、他のアプリケーションやデータベースで使用できます。データベースのどのフィールドを公開するかをスタイルシートで制御することにより、データをフィルタできます。また、Web ページにデータを表示する書式を設定したり、ユーザがデータを操作する方法を制御できます。第 4 章「XSLT を使用したカスタム Web 公開の概要」を参照してください。

Web ユーザがいずれかの XSLT スタイルシートを参照する HTTP リクエストと URL を送信すると、Web 公開エンジンは、スタイルシートを使用して FileMaker データベースから動的にデータを取得します。Web 公開エンジンは、スタイルシートを使用して XML データの変換と書式設定を行い、Web ユーザが操作できる最終的な HTML ページを生成します。

XML および XSLT を使用した FileMaker Server Advanced カスタム Web 公開の使用の詳細については、www.filemaker.co.jp にアクセスしてください。

XSLT スタイルシートを開発するためのツールについて

FileMaker Server Advanced には、XSLT スタイルシートを開発するための次の 2 つのツールが付属します。

- FileMaker Site Assistant は、XSLT を使用した FileMaker カスタム Web 公開の起点となる基本的な XSLT スタイルシートを作成する場合に使用できるアプリケーションです。Site Assistant は、FileMaker XSLT スタイルシートの構造を学ぶのに効果的です。必要に応じて、好みの XSLT スタイルシートオーサリングツールを使用してスタイルシートを変更することができます。39 ページの「FileMaker Site Assistant を使用した FileMaker XSLT スタイルシートの生成」を参照してください。
- FileMaker CDML Converter は、既存の CDML フォーマットファイルを、XSLT を使用したカスタム Web 公開と互換性がある XSLT スタイルシートに変換するアプリケーションです。CDML Web サイトの移行を開始する場合や、FileMaker XSLT スタイルシートの構造を理解する場合に便利です。42 ページの「FileMaker CDML Converter の使用」を参照してください。

Web 上でデータベースを公開する場合の必要条件

カスタム Web 公開を使用してデータベースを公開するための必要条件

XML または XSLT を使用したカスタム Web 公開を使用してデータベースを公開するための必要条件は、次のとおりです。

- FileMaker Server が動作する、カスタム Web 公開が有効な環境
- FileMaker Server でホストされている 1 つまたは複数の FileMaker Pro データベース
- FileMaker Server Web 公開エンジンのインストールと設定
- Microsoft IIS（Windows）または Apache（Mac OS X）のいずれかの Web サーバー
- Web サーバーが実行されているホストの IP アドレスまたはドメイン名
- カスタム Web 公開ソリューションを開発およびテストするための Web ブラウザと Web サーバーへのアクセス

Web ユーザがカスタム Web 公開ソリューションにアクセスするための必要条件

Web ユーザが XML または XSLT を使用したカスタム Web 公開ソリューションにアクセスするための必要条件は、次のとおりです。

- Web ブラウザソフトウェア
- インターネットまたはイントラネット、および Web サーバーへのアクセス
- Web サーバーが実行されているホストの IP アドレスまたはドメイン名
- データベースがパスワードで保護されている場合は、データベースアカウントのユーザ名とパスワードの入力

インターネットまたはイントラネットへの接続

インターネットまたはイントラネット上でデータベースを公開する場合、ホストコンピュータで FileMaker Server を起動し、共有するデータベースをホストして利用可能にする必要があります。また、次の点にも注意してください。

- データベースは、インターネットまたはイントラネットへの常時接続を確保したコンピュータで公開することをお勧めします。インターネットに常時接続していなくても Web 上でデータベースを公開することは可能ですが、Web ユーザはホストするコンピュータがインターネットまたはイントラネットに接続している場合にのみデータベースにアクセスすることができます。
- Web 公開エンジンを実行するホストコンピュータに、静的、つまり固有な専用の IP アドレスまたはドメイン名が設定されている必要があります。ISP（インターネットサービスプロバイダ）に接続してインターネットを使用する場合、IP アドレスは動的に割り当てられる可能性があります。つまり、接続するたびに IP アドレスが変更されることになります。動的な IP アドレスを使用すると、Web ユーザがデータベースを見つけることが困難になります。使用できるインターネットへのアクセスの種類がわからない場合は、ISP またはネットワーク管理者にお問い合わせください。

XML および XSLT を使用したカスタム Web 公開の主な機能

FileMaker Server Advanced のバージョン 7 と 8 では、XML および XSLT を使用したカスタム Web 公開はいくつかの重要な機能を提供します。

- データベースは FileMaker Server 上でホストされ、FileMaker Pro が実行されている必要はありません。
- XSLT スタイルシートをサーバーサイドで処理できるようになりました。これは、クライアントサイドでスタイルシートを処理するよりも安全です。
- FileMaker Server 8 Advanced では、XSLT スタイルシートによって JavaScript をサーバーサイドで処理できるようになりました。詳細については、67 ページの「スクリプト言語のサーバーサイド処理の使用」を参照してください。

- FileMaker XSLT スタイルシートを使用して、XML データの要求時に使用するクエリーコマンド、引数、および値を静的に定義しておくことで、クエリーコマンドとクエリー引数の不正使用を防止できます。50 ページの「静的に定義されたクエリーコマンドとクエリー引数の使用」を参照してください。
- FileMaker Pro と同様に、データ、レイアウト、およびフィールドへのアクセスは、データベースのアクセス権で定義されているユーザのアカウント設定に基づきます。また、Web 公開エンジンでは、他のセキュリティの強化点もいくつかサポートされています。18 ページの「公開されたデータベースの保護」を参照してください。
- Web ユーザが、複数のステップを使用した複雑なスクリプトを実行することができます。約 70 のスクリプトステップが Web 上でサポートされるようになりました。次のセクション「FileMaker スクリプトとカスタム Web 公開」を参照してください。
- FileMaker Server 8 Advanced では、FileMaker スクリプトに引数の値を渡すことができます。詳細については、84 ページの「-script.param（スクリプトに引数を渡す）クエリー引数」、85 ページの「-script.prefind.param（検索する前に引数をスクリプトに渡す）クエリー引数」、および86 ページの「-script.presort.param（ソートする前に引数をスクリプトに渡す）クエリー引数」を参照してください。
- 新しい fmresultset XML 文法では、名前でフィールドにアクセスして、関連セット（ポータル）のデータを操作できます。
- XSLT スタイルシートでセッション関数を使用して、サーバーによって維持されるセッションに Web ユーザの情報および処理を保存できます。
- 複数の新しいクエリーコマンドと引数が追加され、一部のクエリーコマンドと引数は使用されなくなりました。データベースのデータにアクセスするには、レイアウトを指定する必要があります。セキュリティ上の理由から、レイアウトを指定せずにデータにアクセスすることはできなくなりました。付録 A「クエリー文字列で使用される有効な名前」を参照してください。
- 各 Web ユーザは、セッションがアクティブである限り維持される固有のグローバルフィールド値を使用できます。グローバルフィールドの詳細については、FileMaker Pro ヘルプを参照してください。カスタム Web 公開でのグローバルフィールドの使用の詳細については、78 ページの「グローバルフィールドを指定するための構文について」を参照してください。

FileMaker スクリプトとカスタム Web 公開

頻繁に実行されるタスクを自動化したり、複数のタスクを組み合わせるには、FileMaker Pro の ScriptMaker™ 機能が便利です。ScriptMaker をカスタム Web 公開とともに使用すると、Web ユーザは FileMaker スクリプトを使用して、より多くのタスクや一連のタスクを実行できます。

カスタム Web 公開では、約 70 のスクリプトステップがサポートされています。URL のクエリー文字列内、または XSLT スタイルシート内で <?xslt–cwp–query?> 処理命令を使用すると、Web ユーザは、自動化されたさまざまなタスクを実行できます。サポートされていないスクリプトステップを参照するには、FileMaker Pro の [スクリプト編集] ダイアログボックスで [Web 互換を区別して表示] チェックボックスを選択します。グレー表示されるスクリプトステップは、Web 上ではサポートされません。スクリプトの作成の詳細については、FileMaker Pro ヘルプを参照してください。

スクリプトのヒントと考慮事項

多くのスクリプトステップは Web 上でも同じように動作しますが、次の表に示すように動作が異なるものもあります。データベースを共有する前に、Web ブラウザから実行されるスクリプトをすべて評価してください。また、異なるユーザアカウントでログインして、すべてのクライアントに対して正しく動作することを確認します。

次のヒントおよび考慮事項に注意してください。

- アカウントとアクセス権を使用して、Web ユーザが実行可能なスクリプトのセットを制限します。Web 互換のスクリプトステップのみがスクリプトに含まれることを確認し、Web ブラウザから使用する必要があるスクリプトへのアクセスのみを提供します。
- アクセス権によって制御されたステップの組み合わせを実行するスクリプトの影響を考慮します。たとえば、レコードを削除するステップがスクリプトに含まれていて、Web ユーザがレコードの削除を許可するアカウントでログインしていない場合、このスクリプトでは、[レコード削除] スクリプトステップは実行されません。ただし、スクリプトは引き続き実行される場合があり、予期しない結果になる可能性があります。

- スクリプトで [スクリプトを完全アクセス権で実行] を選択すると、個々のアクセスが付与されていないタスクをスクリプトで実行することができます。たとえば、アカウントとアクセス権を使用してユーザがレコードを削除できないようにしつつ、スクリプト内にあらかじめ定義された条件下で特定のタイプのレコードを削除するスクリプトの実行を許可することができます。
- サポートされていないステップ（Web 互換ではないステップなど）がスクリプトに含まれる場合は、[ユーザによる強制終了を許可] スクリプトステップを使用して、以降のステップの処理方法を決定します。
  - [ユーザによる強制終了を許可] スクリプトステップオプションが有効（オン）の場合、サポートされていないスクリプトステップが使用されていると、スクリプトの続行は停止されます。
  - [ユーザによる強制終了を許可] がオフの場合、サポートされていないスクリプトステップはスキップされ、スクリプトの実行が続行されます。
  - このスクリプトステップが含まれない場合、スクリプトは、この機能が有効な場合と同様に実行されるため、サポートされていないスクリプトステップが使用されていると、スクリプトは停止します。
- FileMaker Pro クライアントから 1 つのステップで動作する一部のスクリプトでは、追加の [レコード/検索条件確定] ステップを使用して、データをホストに保存しなければならない場合があります。Web ユーザはホストと直接接続していないので、データが変更されたときに通知されません。たとえば、条件付き値一覧などの機能では、値一覧フィールドに結果を表示するにはデータをホストに保存する必要があるため、Web ユーザに対しては高速に応答しません。
- 同様に、すべてのデータ変更は、データをサーバーに保存する（「送信する」）までブラウザに表示されないので、データを変更するスクリプトにも [レコード/検索条件確定] ステップを含める必要があります。これには、[切り取り]、[コピー]、[貼り付け] などのスクリプトステップが含まれます。単一ステップの処理の多くは、[レコード/検索条件確定] ステップに変換する必要があります。Web サーバーから実行されるスクリプトを設計する際は、スクリプトの最後に [レコード/検索条件確定] ステップを含めて、すべての変更が保存されるようにすることをお勧めします。
- クライアントのタイプに基づく条件付きスクリプトを作成するには、Get（アプリケーションバージョン）関数を使用します。返された値に「Web Publishing Engine 8.0v1」が含まれていれば、現在のユーザがカスタム Web 公開を使用してデータベースにアクセスしていることがわかります。関数の詳細については、FileMaker Pro ヘルプを参照してください。
- ファイルを変更した後に、Web ユーザが実行する可能性のある各スクリプトを開いて、[Web 互換を区別して表示] を選択し、カスタム Web 公開でスクリプトが正しく実行されることを確認する必要があります。
- XSLT スタイルシートで状態を設定または変更するスクリプトを使用する場合は、管理コンソールを使用して、Web 公開エンジンの [XSLT データベースセッション:] オプションを有効にする必要があります。このオプションが有効でない場合、状態はリクエスト間で維持されません。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

次のスクリプトステップは、Web 上と FileMaker Pro で機能が異なります。すべてのスクリプトステップの詳細については、FileMaker Pro ヘルプを参照してください。

スクリプトステップ	カスタム Web 公開ソリューションでの動作
スクリプト実行	ファイルが FileMaker Server 上でホストされていて、他のファイルでカスタム Web 効果が有効になっていない限り、スクリプトを他のファイルで実行することはできません。
アプリケーションを終了	Web ユーザをログオフしてすべてのウインドウを閉じますが、Web ブラウザアプリケーションは終了しません。
ユーザによる強制終了を許可	サポートされていないスクリプトステップの処理方法を決定します。スクリプトの続行を中止する場合は有効にし、サポートされていないステップをスキップする場合は無効にします。詳細については、このセクションの前の説明を参照してください。 注意　Web ユーザはカスタム Web 公開スクリプトを中止することはできませんが、このオプションによって、サポートされていないスクリプトステップの続行を中止することができます。
エラー処理	これは、カスタム Web 公開では常に有効です。Web ユーザはカスタム Web 公開スクリプトを中止することができません。

スクリプトステップ	カスタム Web 公開ソリューションでの動作
スクリプト一時停止/続行	このスクリプトステップはカスタム Web 公開でサポートされていますが、使用しないことをお勧めします。一時停止を行うステップが実行されると、スクリプトの実行が一時停止します。一時停止したスクリプトの実行は、続行スクリプトステップが含まれるスクリプトでのみ続行できます。セッションがタイムアウトするまでスクリプトの実行が一時停止された状態のままになっている場合、スクリプトは完了しません。
レコードのソート	カスタム Web 公開で実行するには、指定するソート順をこのスクリプトとともに保存しておく必要があります。
URL を開く	このスクリプトは、カスタム Web 公開ソリューションでは効果はありません。
フィールドへ移動	[フィールドへ移動] を使用して Web ブラウザで特定のフィールドをアクティブにすることはできませんが、このスクリプトステップを他のスクリプトステップと組み合わせて使用して、タスクを実行することができます。たとえば、フィールドに移動して内容をコピーし、別のフィールドに移動して値を貼り付けることができます。ブラウザに結果を表示するには、必ず [レコード/検索条件確定] スクリプトステップを使用してレコードを保存してください。
確定 レコード/検索条件	データベースにレコードを送信します。

旧バージョンのファイルメーカーPro からの Web 公開ソリューションの移行

既存のデータベースを FileMaker Pro 8 に更新する前に、『旧バージョンの FileMaker データベースの変換』を参照してください。このマニュアルは、www.filemaker.co.jp からダウンロードすることができます。

Web 公開ソリューションを移行する場合は、最初に、ファイルをバックアップして、変換を行うための独立したテスト環境を設定します。XML または XSLT ソリューションを開発したら、サイトの運用を開始する前に、ソリューションの機能とセキュリティ（アカウントとアクセス権）をテストします。

次に、XML または XSLT を使用して公開するソリューションを移行する際、特にバージョン 7 より前のバージョンで作成されたソリューションの際の、考慮事項を示します。

- アクセス権モデルは、FileMaker Pro 7 で強化されました。ユーザ名とパスワードを割り当て直して、アカウントとアクセス権を活用することを検討してください。FileMaker Pro ヘルプを参照してください。
- FileMaker Pro 用に設計されているプラグインは、FileMaker Server では自動的には有効になりません。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。
- Web セキュリティデータベースはサポートされなくなりました。Web ベースのセキュリティを確保するために Web セキュリティデータベースに依存している場合は、FileMaker Pro で、アカウント、パスワード、および関連するアクセス権を変換後のデータベースファイルに移動する必要があります。『旧バージョンの FileMaker データベースの変換』ガイドを参照してください。
- CDML ソリューションを移行する場合は、付録 C「CDML ソリューションの FileMaker XSLT への変換」を参照してください。
- データベースのオブジェクトフィールドに実際のオブジェクトではなくファイル参照が保存されている場合は、レコードを作成または編集するときに、参照されているオブジェクトを FileMaker Pro の「Web」フォルダに保存してから、Web サーバーソフトウェアのルートフォルダ内の同じ相対パスの場所にあるフォルダにコピーまたは移動する必要があります。18 ページの「Web 上でのオブジェクトフィールドの内容の公開について」を参照してください。
- Web 上で公開するファイルの URL 構文が FileMaker Pro Server 7 Advanced で変更されています。Web 上のデータベースにアクセスするためのリンクを FileMaker Pro Server 7 Advanced より前のバージョンで作成している場合は、ファイルを変換して FileMaker Server 上でホストした後に、新しい構文とデータベースの場所を使用してリンクを更新する必要があります。23 ページの「XML データとオブジェクトにアクセスするための URL 構文について」および 48 ページの「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。
- FileMaker Pro 7 では、Web に対するスクリプトのサポートが強化されています。
  - Web ユーザにアクセス権を設定して、特定の個々のスクリプトの実行は許可し、他のスクリプトの実行は防止することができます。

- ScriptMaker の [スクリプト編集] ダイアログボックスにある [Web 互換を区別して表示] オプションによって、スクリプトステップが「Web 互換」かどうかが明確に示されるようになり、Web ユーザ専用のスクリプトを作成することができます。ファイルの変換後、Web ユーザが実行する可能性のある各スクリプトを開き、[Web 互換を区別して表示] を選択して、Web ブラウザから実行した場合に予期しない結果になる可能性のあるステップがスクリプトに含まれているかどうかを確認してください。
- Web 上では、スクリプトは、常にエラー処理がオンの状態で実行されます。

この後の作業を開始するにあたって

次に、カスタム Web 公開ソリューションの開発を開始するために必要な作業に関する推奨事項と、情報が記載されている場所を示します。

- FileMaker Server Admin と Web 公開エンジン管理コンソールを使用してカスタム Web 公開を有効にしていない場合は、有効にします。FileMaker Server Admin ヘルプ、および『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。
- 公開する各 FileMaker データベースを FileMaker Pro で開き、データベースで、カスタム Web 公開に対して適切な拡張アクセス権が有効になっていることを確認します。17 ページの「データベースでのカスタム Web 公開の有効化」を参照してください。
- XML を使用して FileMaker データベースのデータにアクセスする方法を学ぶには、第 3 章「Web 公開エンジンを使用した XML データへのアクセス」を参照してください。
- FileMaker XSLT スタイルシートの開発を開始する方法を学ぶには、第 4 章「XSLT を使用したカスタム Web 公開の概要」を参照してください。

第 2 章
データベースのカスタム Web 公開の準備

データベースでカスタム Web 公開を使用する前に、データベースを準備して不正アクセスから保護するためには、いくつかの手順を実行する必要があります。

データベースでのカスタム Web 公開の有効化

公開する各データベースでカスタム Web 公開を有効にする必要があります。各データベースで、XML を使用したカスタム Web 公開または XSLT を使用したカスタム Web 公開のいずれかを個別に有効にするか、あるいは両方の技術を有効にできます。データベースでこれらの技術の一方または両方を有効にしないと、Web 公開エンジンをサポートするように設定されている FileMaker Server でデータベースがホストされていても、Web ユーザがカスタム Web 公開を使用してデータベースにアクセスすることはできません。

データベースに対してカスタム Web 公開を有効にするには、次の操作を行います。

1. FileMaker Pro で、[完全アクセス] アクセス権セットが割り当てられているアカウントを使用して、公開するデータベースを開きます。または、[拡張アクセス権の管理] アクセス権が割り当てられているアカウントを使用してデータベースを開くこともできます。
2. 1 つまたは両方の拡張アクセス権を、次の 1 つまたは両方のアクセス権セットに割り当てます。
   - XML を使用したカスタム Web 公開を許可するには、キーワード“fmxml”を入力します。
   - XSLT を使用したカスタム Web 公開を許可するには、キーワード“fmxslt”を入力します。

FileMaker Pro バージョン 8 では、“fmxml”と“fmxslt”のキーワードは [拡張アクセス権] タブで定義されます。

3. 1 つまたは複数のアカウント、あるいは Admin またはゲストアカウントに、カスタム Web 公開拡張アクセス権が含まれるアクセス権セットを割り当てます。

注意　カスタム Web 公開ソリューション用のアカウント名とパスワードを定義する場合は、表示可能な ASCII 文字（a から z、A から Z、および 0 から 9 など）を使用します。アカウント名とパスワードのセキュリティを高めるには、「!」や「%」などの記号を含めます。ただし、コロンは含めないでください。アカウントの設定の詳細については、FileMaker Pro ヘルプを参照してください。

Web ユーザがカスタム Web 公開を使用して保護されたデータベースにアクセスする場合

カスタム Web 公開ソリューションを使用してデータベースにアクセスする場合、Web ユーザに対して、アカウント情報を入力するようメッセージが表示される場合があります。データベースのゲストアカウントが無効になっているか、またはカスタム Web 公開拡張アクセス権が含まれるアクセス権セットが割り当てられていない場合、Web 公開エンジンは、HTTP 基本認証を使用して Web ユーザに認証を要求します。Web ユーザのブラウザによって、カスタム Web 公開拡張アクセス権が割り当てられているアカウントのユーザ名とパスワードをユーザが入力するための HTTP 基本認証のダイアログボックスが表示されます。

次に、Web ユーザがカスタム Web 公開ソリューションを使用してデータベースにアクセスする場合の処理の概要を説明します。

- アカウントにパスワードが割り当てられていない場合、Web ユーザはアカウント名のみを入力します。
- ゲストアカウントが無効な場合、データベースにアクセスするときに、アカウント名とパスワードを入力するようメッセージが表示されます。入力するアカウントでは、カスタム Web 公開拡張アクセス権が有効になっている必要があります。

- ゲストアカウントが有効で、カスタム Web 公開拡張アクセス権が含まれるアクセス権セットが有効な場合、すべての Web ユーザは、自動的に、ゲストアカウントに割り当てられているアクセス権でデータベースを開きます。ゲストアカウントにカスタム Web 公開拡張アクセス権が割り当てられている場合、次のように処理されます。
  - ファイルを開くときに、アカウント名とパスワードを入力するようメッセージは表示されません。
  - すべての Web ユーザは自動的にゲストアカウントでログインし、ゲストアカウントのアクセス権を持ちます。[再ログイン] スクリプトステップを使用すると、ユーザは Web ブラウザからログインアカウントを変更することができます（たとえば、ゲストアカウントからより多くのアクセス権を持つアカウントに切り替えることができます）。
  - ゲストアカウントのデフォルトのアクセス権セットは、「閲覧のみ」アクセスを提供します。このアカウントのデフォルトのアクセス権（拡張アクセス権を含む）を変更できます。FileMaker Pro ヘルプを参照してください。
- 一般的には、Web ユーザが Web ブラウザからアカウントのパスワードを変更することはできません。ただし、[パスワード変更] スクリプトステップを使用すると、この機能をデータベースに組み込み、Web ユーザがパスワードを変更できるようにすることができます。FileMaker Pro ヘルプを参照してください。

公開されたデータベースの保護

XML または XSLT を使用したカスタム Web 公開を使用する場合、公開されたデータベースにアクセス可能なユーザを制限できます。

- カスタム Web 公開に使用されるデータベースアカウントにパスワードを割り当てます。
- XML または XSLT を使用したカスタム Web 公開は、公開されたデータベースへのアクセスを許可するアカウントのアクセス権セットでのみ有効にします。
- 個々のデータベースに対して特定のタイプのカスタム Web 公開技術を有効または無効にするには、拡張アクセス権を設定します。
- Web 公開エンジンで、すべてのカスタム Web 公開ソリューションに対して特定のタイプのカスタム Web 公開技術を有効または無効にするには、管理コンソールを使用します。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。
- Web 公開エンジンを使用してデータベースにアクセスできる IP アドレスを制限するように Web サーバーを設定できます。たとえば、192.168.100.101 という IP アドレスの Web ユーザにのみデータベースへのアクセスを許可するように設定できます。IP アドレスの制限の詳細については、Web サーバーのマニュアルを参照してください。
- Web サーバーと Web ユーザのブラウザの間の通信に、SSL（Secure Sockets Layer）暗号化を使用できます。SSL 暗号化は、「暗号」と呼ばれる数式を使用して、サーバーとクライアントの間で交換される情報を判読不可能な情報に変換します。その後、これらの暗号を使用して、「暗号鍵」によって情報を判読可能なデータに再変換します。SSL の有効化と設定の詳細については、Web サーバーのマニュアルを参照してください。

公開されたデータベースのセキュリティの確保の詳細については、『FileMaker セキュリティガイド』を参照してください。

Web サーバーでのインターネットメディア タイプ（MIME）のサポート

インターネットに対して登録されている最新の MIME（Multipurpose Internet Mail Extensions）タイプがサポートされているかどうかは、Web サーバーによって判断されます。Web 公開エンジンによって、Web サーバーの MIME のサポートが変更されることはありません。詳細については、Web サーバーのマニュアルを参照してください。

Web 上でのオブジェクトフィールドの内容の公開について

イメージファイルなどのオブジェクトフィールドの内容は、FileMaker データベースの内部に保存するか、または相対パスを使用してファイル参照として保存できます。

注意　Web 公開エンジンでは、ムービーファイルのストリーミングはサポートされません。Web ユーザがムービーを表示するには、ムービーファイル全体をダウンロードする必要があります。

データベース内に保存されているオブジェクトフィールドのオブジェクトの公開

FileMaker データベースのオブジェクトフィールドに実際のファイルが保存されている場合は、データベースファイルが FileMaker Server 上で適切にホストされていてアクセス可能であれば、オブジェクトフィールドの内容を操作する必要はありません。24 ページの「XML ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」および 48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。

ファイル参照として保存されているオブジェクトフィールドのオブジェクトの公開

オブジェクトフィールドに、実際のファイルの代わりにファイル参照が保存されている場合、Web 公開エンジンを使用してオブジェクトフィールドのオブジェクトを公開するには、次の操作を行う必要があります。

注意　すべての QuickTime ムービーは、参照としてオブジェクトフィールドに保存されます。

ファイル参照として保存されているオブジェクトフィールドのオブジェクトを公開するには、次の操作を行います。

1. オブジェクトファイルを FileMaker Pro フォルダ内の「Web」フォルダに保存します。
2. FileMaker Pro で、オブジェクトフィールドにオブジェクトを挿入して、[ファイルの参照データのみ保存] オプションを選択します。
3. 「Web」フォルダ内の参照されているオブジェクトファイルを、Web サーバーソフトウェアのルートフォルダ内の同じ相対パスの場所にコピーまたは移動します。
   - IIS の場合は、「<ルートドライブ>\Inetpub\wwwroot」にファイルを移動します。
   - Apache の場合は、「/Library/WebServer/Documents」にファイルを移動します。

注意　ファイル参照として保存されているオブジェクトの場合、提供するファイルの種類（ムービーなど）の MIME タイプをサポートするように Web サーバーが設定されている必要があります。詳細については、Web サーバーのマニュアルを参照してください。

Web ユーザがオブジェクトフィールドのデータを使用する方法

Web 公開エンジンを使用して Web 上にデータベースを公開する場合、Web ユーザは、次のような限られた方法でオブジェクトフィールドのデータを操作することができます。

- オブジェクトフィールドのサウンドを再生したり、OLE オブジェクトを表示することはできません。代わりにグラフィックが表示されます。
- Web ユーザがオブジェクトフィールドの内容を変更または追加することはできません。
- GIF または JPEG 以外の形式のグラフィックがデータベースに格納されている場合は、Web ブラウザからそのグラフィックデータが要求されたときに、Web 公開エンジンによって一時的な JPEG イメージが作成されます。

第 3 章
Web 公開エンジンを使用した XML データへのアクセス

Web 公開エンジンを使用することで、FileMaker データを XML（Extensible Markup Language）形式で取得したり、更新することができます。HTML が World Wide Web 上での通信の標準表示言語になったのと同様に、XML も、構造化データを交換するための標準言語になっています。多くの個人、組織、および企業が、XML を使用して製品情報、取引、在庫データなどの業務データを転送しています。

XML を使用したカスタム Web 公開の使用

標準的な XML の知識がある場合、使用する URL 構文やクエリー引数など、XML を使用したカスタム Web 公開に特有の詳細事項をいくつか学べば、すぐに Web 公開エンジンを使い始めることができます。

HTTP URL リクエストを FileMaker に固有のクエリーコマンドと引数とともに使用することで、FileMaker Server によってホストされているデータベースに対してクエリーを実行して、結果のデータを XML 形式でダウンロードできます。たとえば、データベースに対して特定の郵便番号のレコードすべてを検索するクエリーを実行して、結果の XML データをさまざまな方法で使用できます。

また、Web 公開エンジンのサーバーサイド XSLT スタイルシートを使用して、XML データをフィルタしたり、データの書式を HTML または vCard などのテキストに再設定したり、データを SVG（Scalable Vector Graphics）などの他の XML 文法に変換することもできます。第 4 章「XSLT を使用したカスタム Web 公開の概要」および 第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

XML の一般情報、XML のその他の使用例、および XML に関する役立つ情報へのリンクについては、FileMaker の Web サイト www.filemaker.co.jp を参照してください。

注意　Web 公開エンジンによって生成される XML データは、整形式で、XML 1.0 仕様に準拠しています。整形式の XML の要件の詳細については、www.w3.org で入手できる XML の仕様を参照してください。

Web 公開エンジンと FileMaker Pro の XML インポート/エクスポート機能の違い

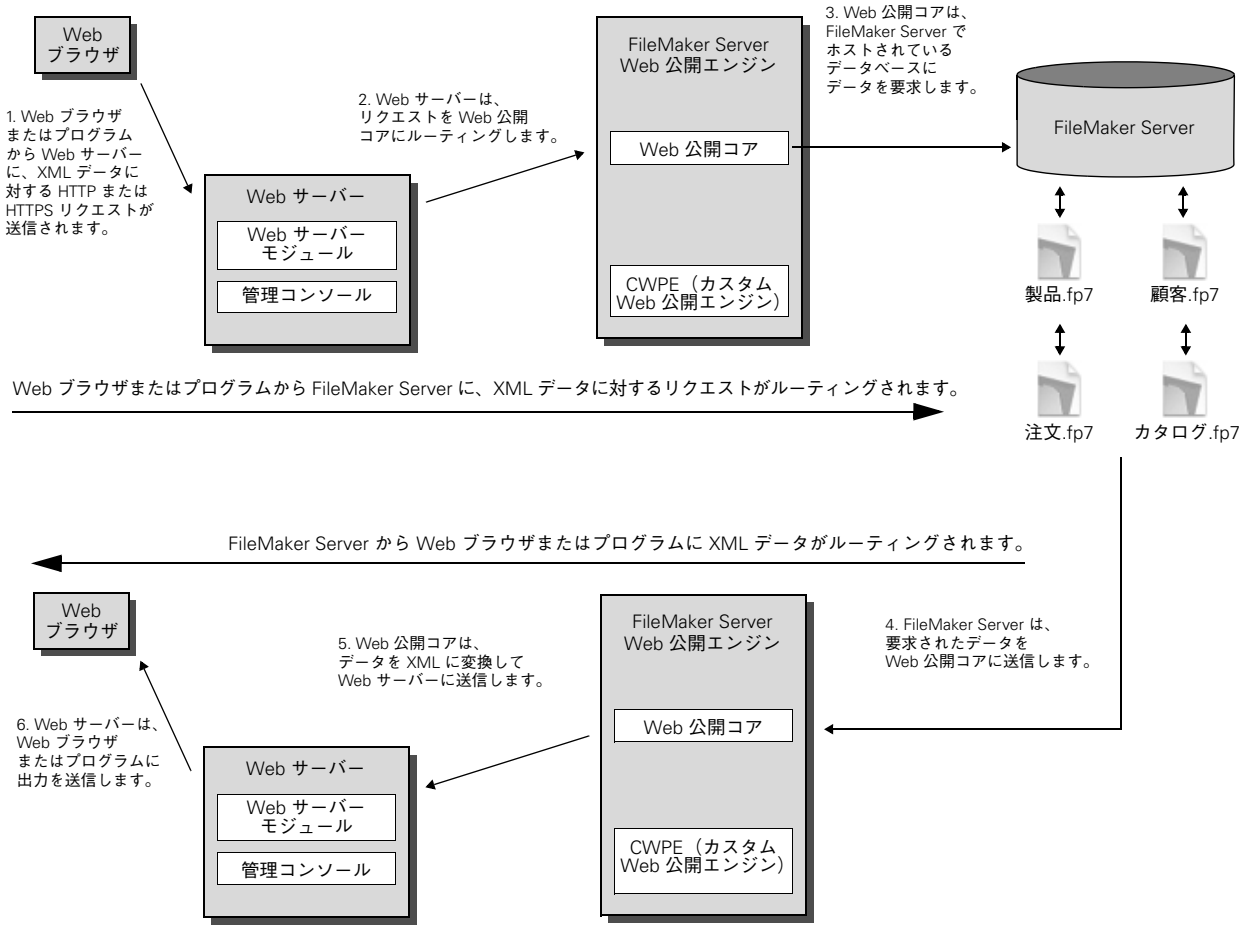
Web 公開エンジンと FileMaker Pro のどちらを使用しても、FileMaker データベースで XML データを使用できます。ただし、これらの２つの方法には、次に示すような重要な違いがあります。

- XML データおよびXSLT Web 公開にアクセスするために、Web 公開エンジンでは、fmresultset、FMPXMLRESULT、および FMPXMLLAYOUT 文法がサポートされています。FileMaker Pro では、XML のインポートには FMPXMLRESULT 文法、エクスポートには FMPXMLRESULT または FMPDSORESULT 文法が使用されます。25 ページの「Web 公開エンジンを使用した XML データへのアクセス」を参照してください。
- Web 公開エンジンで XML データにアクセスするには、URL で Web 公開エンジンのクエリー文字列を使用します。FileMaker Pro で XML をインポートおよびエクスポートするには、FileMaker Pro のメニューコマンドまたはスクリプトを使用します。
- Web 公開エンジンはサーバーベースで、FileMaker Server と同じホストにインストールするか、または異なるホストにインストールできます。FileMaker Pro の XML インポートおよびエクスポートの機能は、デスクトップベースです。
- Web 公開エンジンとともに URL リクエストを使用することで、FileMaker データベースから XML データに動的にアクセスできます。FileMaker Pro の XML エクスポート機能では、あらかじめ指定した XML データファイルが生成されます。
- Web 公開エンジンを使用した XML データの操作は、対話型の処理です。FileMaker Pro の XML インポートおよびエクスポートの機能は、バッチ処理です。
- Web 公開エンジンは FileMaker ポータルの XML データにアクセスできますが、FileMaker Pro ではできません。
- Web 公開エンジンはオブジェクトフィールド内のデータにアクセスできますが、FileMaker Pro ではできません。
- Web 公開エンジンは HTTP または HTTPS 上で FileMaker データにリアルタイムにアクセスできますが、FileMaker Pro ではできません。

注意　FileMaker Pro を使用して XML 形式でデータをインポートおよびエクスポートする操作の詳細については、FileMaker Pro ヘルプを参照してください。

Web 公開エンジンがリクエストから XML データを生成する方法

XML データに対するリクエストが Web サーバーに送信されると、Web 公開エンジンは、FileMaker データベースに対してクエリーを実行し、データを XML ドキュメントとして返します。



Web 公開エンジンから XML データにアクセスするための一般的な手順

次に、Web 公開エンジンを使用して FileMaker データベース内の XML データにアクセスする場合の手順の概要を示します。

1. Web　公開エンジン管理コンソールで、[XML　公開:]　が有効になっていることを確認します。『FileMaker　Server Advanced Web 公開インストールガイド』を参照してください。
2. 公開する各 FileMaker データベースを FileMaker Pro で開き、データベースで、XML カスタム Web 公開に対して fmxml 拡張アクセス権が有効になっていることを確認します。17 ページの「データベースでのカスタム Web 公開の有効化」を参照してください。

   ポータル内の XML データにアクセスするには、データベースレイアウトの表示形式を [フォーム形式] または [リスト形式] に設定します。ユーザまたはスクリプトによってデータベースレイアウトの表示形式が [表形式] に変更された場合は、最初の関連レコード（ポータルの最初の行）にのみ XML データとしてアクセスできます。

XML データは、フィールドオブジェクトがレイアウトに追加された順序に対応する順序で出力されます。XML データの順序を画面（上から下、左から右の順序）に表示されるフィールドの順序に一致させるには、すべてのフィールドを選択してグループ化した後で、グループ解除します。この操作を行うと、レイアウトの順序がリセットされて画面の順序と一致されます。

3. HTML フォーム、HREF リンク、またはプログラムや Web ページ内のスクリプトを使用して、FileMaker XML 文法、1 つのクエリーコマンド、および 1 つまたは複数の FileMaker クエリー引数を指定した URL の形式で、HTTP または HTTPS リクエストを Web 公開エンジンに送信します。Web ブラウザに URL を入力することもできます。

   URL の指定の詳細については、次のセクション「XML データとオブジェクトにアクセスするための URL 構文について」を参照してください。クエリーコマンドと引数の詳細については、32 ページの「FileMaker クエリー文字列を使用した XML データの要求」および付録 A「クエリー文字列で使用される有効な名前」を参照してください。

4. Web 公開エンジンは、URL で指定された文法を使用して、リクエストの結果（データベースからのレコードのセットなど）が含まれる XML データを生成し、プログラムまたは Web ブラウザに返します。

5. Web ブラウザに XML パーサが含まれる場合は、Web ブラウザによってデータが表示されます。または、指定した方法でプログラムによってデータが使用されます。

   クライアントサイドのスタイルシートが指定されている場合は、Web ブラウザのパーサによって、該当するスタイルシート命令も適用されます。35 ページの「サーバーサイドおよびクライアントサイドでのスタイルシートの処理の使用」を参照してください。

XML データとオブジェクトにアクセスするための URL 構文について

このセクションでは、Web 公開エンジンを使用して FileMaker データベースの XML データおよびオブジェクトにアクセスするための URL 構文について説明します。XSLT スタイルシートを使用するための URL 構文は、XML データにアクセスするための構文とは異なります。48 ページの「FileMaker XSLT スタイルシートの URL 構文について」および 48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。

XML データにアクセスするための新しい URL 構文

FileMaker Server バージョン 7 および 8 では、Web 公開エンジンは XML データにアクセスするために URL 構文を使用します。

- ファイルメーカー Pro 6 以前では、XML データに対する要求で次の構文が使用されていました。

  FMPro?<CGI リクエスト >

  この構文は変更されています。次のセクション「XML データにアクセスするための URL 構文について」を参照してください。

- -dbnames、-layoutnames、-scriptnames、および -process （XSLT リクエストのみ）以外のすべてのクエリーコマンドで、データベースレイアウトを指定するための -lay クエリー引数が必要です。付録 A「クエリー文字列で使用される有効な名前」を参照してください。

- -format 引数は使用されなくなりました。XML リクエストの XML 文法は、URL 構文でクエリー文字列の前に指定します。次のセクション「XML データにアクセスするための URL 構文について」を参照してください。

注意　XML リクエストと異なり、XSLT スタイルシートの文法は、-grammar クエリー引数を使用して指定します。49 ページの「FileMaker XSLT スタイルシートの XML 文法の指定」を参照してください。

XML データにアクセスするための URL 構文について

次に、Web 公開エンジンを使用して FileMaker データベースから XML データにアクセスするための URL 構文を示します。

<スキーム>://<ホスト>[:<ポート>]/fmi/xml/<XML 文法>.xml[?<クエリー文字列>]

各要素の意味は、次のとおりです。

- <スキーム> には、HTTP または HTTPS プロトコルを指定できます。
- <ホスト> には、Web サーバーがインストールされているホストの IP アドレスまたはドメイン名を指定します。

- <ポート> には、Web サーバーが使用するポートを指定します（オプション）。ポートが指定されていない場合は、プロトコルのデフォルトのポート（HTTP ではポート 80、HTTPS ではポート 443）が使用されます。
- <XML 文法> には、FileMaker XML 文法の名前を指定します。指定できる値は、fmresultset.xml、FMPXMLRESULT.xml、FMPXMLLAYOUT.xml、または FMPDSORESULT.xml です。26 ページの「fmresultset 文法の使用」および 29 ページの「他の FileMaker XML 文法の使用」を参照してください。
- <クエリー文字列> には、FileMaker XML に使用する 1 つのクエリーコマンドと 1 つまたは複数のクエリー引数の組み合わせを指定します（-dbnames コマンドでは引数は必要ありません）。32 ページの「FileMaker クエリー文字列を使用した XML データの要求」および 付録 A「クエリー文字列で使用される有効な名前」を参照してください。

注意　クエリーコマンドおよび引数を含め、URL 構文では、クエリー文字列の部分を除いて大文字と小文字が区別されます。URL のほとんどの部分は小文字ですが、FMPXMLRESULT、FMPXMLLAYOUT、および FMPDSORESULT の 3 つの文法名は大文字です。クエリー文字列の大文字と小文字の規則の詳細については、76 ページの「クエリーコマンドと引数の使用のガイドライン」を参照してください。

次に、Web 公開エンジンを使用して XML データにアクセスするための URL の例を 2 つ示します。

http://server.company.com/fmi/xml/fmresultset.xml?-db=products&-lay=sales&-findall

http://192.168.123.101/fmi/xml/FMPXMLRESULT.xml?-db=products&-lay=sales&-findall

XML ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について

XML ソリューションに対して生成された XML ドキュメントでは、オブジェクトへの参照が保存されているフィールドと、実際のオブジェクトがデータベース内に保存されているオブジェクトフィールドで、オブジェクトを参照するために使用される構文が異なります。

- オブジェクトフィールドで実際のオブジェクトがデータベース内に保存されている場合、オブジェクトフィールドの <data> エレメントは、次の相対 URL 構文を使用してオブジェクトを参照します。

  <data>/fmi/xml/cnt/data.< 拡張子 >?< クエリー文字列 ></data>

  <拡張子> には、.jpg など、オブジェクトのタイプを識別するファイル拡張子を指定します。このファイル拡張子によって MIME タイプが設定され、Web ブラウザで適切にオブジェクトデータを識別できます。<クエリー文字列> の詳細については、前のセクション「XML データにアクセスするための URL 構文について」を参照してください。

  次に例を示します。

  <data>/fmi/xml/cnt/data.jpg?-db=products&-lay=sales&-field=product_image(1)&-recid=2</data>

  注意　オブジェクトフィールドに対して生成された XML では、-field クエリー引数の値は完全修飾フィールド名になります。括弧内の数字は、オブジェクトフィールドの繰り返し数を示し、繰り返しフィールドと非繰り返しフィールドの両方に対して生成されます。77 ページの「完全修飾フィールド名の構文について」を参照してください。

  データベースからオブジェクトデータを取得するには、次の構文を使用します。

  < スキーム >://< ホスト >[:< ポート >]/fmi/xml/cnt/data.< 拡張子 >?< クエリー文字列 >

  <スキーム>、<ホスト>、または <ポート> の詳細については、前のセクション「XML データにアクセスするための URL 構文について」を参照してください。

  次に例を示します。

  http://www.company.com/fmi/xml/cnt/data.jpg?-db=products&-lay=sales&-field=product_image(1)&-recid=2

- オブジェクトフィールドに実際のオブジェクトではなくファイル参照が保存されている場合、オブジェクトフィールドの <data> エレメントには、オブジェクトを参照する相対パスが含まれます。次に例を示します。

  <data>/images/logo.jpg</data>

  注意　レコードを作成または編集するときに、参照されているオブジェクトを FileMaker Pro の「Web」フォルダに保存してから、Web サーバーソフトウェアのルートフォルダ内の同じ相対パスの場所にあるフォルダにコピーまたは移動する必要があります。18 ページの「Web 上でのオブジェクトフィールドの内容の公開について」を参照してください。

- オブジェクトフィールドが空の場合は、オブジェクトフィールドの <data> エレメントも空になります。

注意　XML を使用してオブジェクトにアクセスするための構文は、XSLT を使用してオブジェクトにアクセスするための構文とは異なります。48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。

URL のテキストエンコードについて

XML データおよびオブジェクトにアクセスするための URL は、UTF-8（Unicode Transformation 8 Bit）形式でエンコードされている必要があります。32 ページの「UTF-8 でエンコードされているデータについて」を参照してください。

たとえば、「info」フィールドの値を「fiancée」に設定するには、次の URL を使用することができます。

http://server.company.com/fmi/xml/fmresultset.xml?-db=members&-lay=relationships&-recid=2
&info= fianc%C3%A9e&-edit

この URL の例で、%C3%A9 は、UTF-8 で表した é 文字を URL エンコードしたものです。

URL のテキストエンコードの詳細については、www.w3.org で入手できる URL の仕様を参照してください。

Web 公開エンジンを使用した XML データへのアクセス

Web 公開エンジンを通じて XML データにアクセスするには、使用する FileMaker 文法の名前、1 つの FileMaker クエリーコマンド、および 1 つまたは複数の FileMaker クエリー引数を指定した URL を使用します。Web 公開エンジンによって、次のいずれかのタイプの XML 文法によって書式が設定された XML データがデータベースから生成されます。

- fmresultset: Web 公開エンジンには、この文法を使用することをお勧めします。この文法は、柔軟で XSLT スタイルシートオーサリングに最適化されており、名前によるフィールドアクセスや、関連セット（ポータル）データの操作をより簡単に行うことができます。また、この文法は、グローバル格納オプションや、集計および計算フィールドの識別など、FileMaker の用語や機能とより直接的なつながりを持ちます。この文法は、XML データへのアクセスと XSLT スタイルシートに使用できます。Web 公開を効率的に実行できるよう、この文法は、FMPXMLRESULT 文法よりも詳細になるように設計されています。26 ページの「fmresultset 文法の使用」を参照してください。
- FMPXMLRESULT および FMPXMLLAYOUT: XML データにアクセスしたり、XSLT スタイルシートを使用するには、FMPXMLRESULT および FMPXMLLAYOUT 文法も Web 公開エンジンとともに使用できます。XML エクスポートとカスタム Web 公開の両方に 1 つのスタイルシートを使用するには、FMPXMLRESULT 文法を使用する必要があります。レイアウト内の値一覧およびフィールド表示情報にアクセスするには、FMPXMLLAYOUT 文法を使用する必要があります。29 ページの「他の FileMaker XML 文法の使用」を参照してください。
- FMPDSORESULT: FMPDSORESULT 文法は、FileMaker Pro で XML をエクスポートするためにサポートされているもので、Web 公開エンジン経由で XML データにアクセスする目的では使用されません。また、FMPDSORESULT 文法は XSLT スタイルシートに対してはサポートされていません。FMPDSORESULT 文法の詳細については、FileMaker Pro ヘルプを参照してください。

URL リクエストで指定した文法に応じて、Web 公開エンジンにより、次の文法の 1 つを使用して XML ドキュメントが生成されます。各 XML ドキュメントには、使用する文法のデフォルトの XML ネームスペース宣言が格納されます。次のセクション「FileMaker XML のネームスペースについて」を参照してください。ドキュメントや Web ページでこれらの文法の 1 つを使用して、FileMaker のデータを XML 形式で表示および操作します。

注意　Web 公開エンジンによって生成される XML データは、UTF-8 形式（Unicode Transformation Format 8）を使用してエンコードされます。32 ページの「UTF-8 でエンコードされているデータについて」を参照してください。

FileMaker XML のネームスペースについて

XML タグが使用されるアプリケーションでタグを区別するには、固有の XML ネームスペースが役立ちます。たとえば、2 つの <DATABASE> エレメント（FileMaker XML データと Oracle XML データ用にそれぞれ 1 つずつ）が XML ドキュメントに含まれている場合、各 <DATABASE> エレメントをネームスペースで区別できます。

Web 公開エンジンによって、各文法に対してデフォルトのネームスペースが生成されます。

使用する文法	生成されるデフォルトのネームスペース
fmresultset	xmlns="http://www.filemaker.com/xml/fmresultset"
FMPXMLRESULT	xmlns="http://www.filemaker.com/ fmpxmlresult"
FMPXMLLAYOUT	xmlns="http://www.filemaker.com/fmpxmllayout"

FileMaker データベースのエラーコードについて

最後に実行されたクエリーコマンドの実行時にエラーが発生した場合、Web 公開エンジンは、各 XML ドキュメントの始めに、エラーを表すエラーコードをエラーコードエレメントで囲んで返します。エラーがない場合、値ゼロ（0）が返されます。

使用する文法	使用される構文
fmresultset	<error code="0"></error>
FMPXMLRESULT	<ERRORCODE>0</ERRORCODE>
FMPDSORESULT	<ERRORCODE>0</ERRORCODE>

XML ドキュメントのエラーコードエレメントは、データベースとクエリー文字列に関するエラーを示します。XSLT スタイルシートでは他のタイプのエラーが発生する可能性もあり、異なる方法で処理されます。付録 B「カスタム Web 公開のエラーコード」を参照してください。

FileMaker 文法の文書型定義の取得

FileMaker 文法の DTD（文書型定義）は、HTTP リクエストを使用して取得できます。

使用する文法	使用する HTTP リクエスト
fmresultset	http://<ホスト>[:<ポート]/fmi/xml/fmresultset.dtd
FMPXMLRESULT	http://<ホスト>[:<ポート]/fmi/xml/FMPXMLRESULT.dtd
FMPXMLLAYOUT	http://<ホスト>[:<ポート]/fmi/xml/FMPXMLLAYOUT.dtd
FMPDSORESULT	http://<ホスト>[:<ポート]/fmi/xml/FMPDSORESULT.dtd?-db=<データベース>&-lay=<レイアウト>

fmresultset 文法の使用

この文法では、XML エレメント名に FileMaker の用語が使用され、フィールドの保存内容がフィールドのタイプから分離されています。また、この文法には、集計、計算、およびグローバルフィールドを識別する機能も含まれています。

fmresultset 文法を使用するには、Web 公開エンジンに XML ドキュメントを要求する URL で、fmresultset 文法の次の名前を指定します。

fmresultset.xml

次に例を示します。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-findall

注意　fmresultset 文法を指定する場合は、小文字を使用してください。

Web 公開エンジンによって、fmresultset 文法を使用して XML ドキュメントが生成されます。この XML ドキュメントで、Web 公開エンジンは、ドキュメントの <?xml...?> 命令直後の２行目にある <!DOCTYPE> 命令で、fmresultset 文法の文書型定義を参照します。<!DOCTYPE> 命令によって、fmresultset 文法の DTD をダウンロードするための URL が指定されます。

fmresultset 文法のエレメントの説明

fmresultset 文法は主に <datasource> エレメント、<metadata> エレメント、および <resultset> エレメントで構成されています。

<datasource> エレメント

fmresultset 文法では、<datasource> エレメントに、table、layout、date-format、time-format、timestamp-format、total-count、および database 属性が含まれます。

- XML ドキュメントの日付の書式は、<datasource> エレメントの date-format 属性で指定されます。

  yyyy/MM/dd

各要素の意味は、次のとおりです。

- yyyy は、4 桁の年の値
- MM は、2 桁の月の値（01 から 12 までで、01 は 1 月、12 は 12 月）
- dd は、2 桁の日付の値（00 から 31 まで）

- XML ドキュメントの時刻の書式は、<datasource> エレメントの time-format 属性で指定されます。

  HH:mm:ss

各要素の意味は、次のとおりです。

- HH は、2 桁の時間の値（00 から 23 までの 24 時間形式）
- mm は、2 桁の分の値（00 から 59）
- ss は、2 桁の秒の値（00 から 59）

- <datasource> エレメントの timestamp-format 属性は、date-format と time-format の形式を 1 つのタイムスタンプに結合します。

  yyyy/MM/dd HH:mm:ss

<metadata> エレメント

fmresultset 文法の <metadata> エレメントには、1 つまたは複数の <field-definition> および <relatedset-definition> エレメントが含まれ、各エレメントには、結果セットのフィールドの 1 つの属性が含まれます。これらの属性によって、入力値の自動化が設定されたフィールドかどうか（「yes」または「no」）、繰り返し値の最大数（max-repeat 属性）、空欄不可のフィールドかどうか（「yes」または「no」）、グローバルフィールドかどうか（「yes」または「no」）、結果（「text」、「number」、「date」、「time」、「timestamp」、または「container」）、タイプ（「normal」、「calculation」、または「summary」）、およびフィールド名（必要に応じて完全修飾名）が指定されます。

<relatedset-definition> エレメントは、ポータルを表します。ポータル内の各関連フィールドは、<relatedset-definition> エレメントに含まれる <field-definition> エレメントで表されます。ポータル内に複数の関連フィールドがある場合、関連フィールドのフィールド定義は、単一の <relatedset-definition> エレメント内にまとめられます。

<resultset> エレメント

<resultset> エレメントには、クエリーの結果として返されたすべての <record> エレメントと、見つかったレコードの総数の属性が格納されます。各 <record> エレメントは、結果セット内の 1 つのレコードのフィールドデータ（レコードの mod-id および record-id 属性を含む）と、レコード内の 1 つのレコードのデータが含まれる <data> エレメントで構成されます。

ポータル内の各レコードは、<relatedset> エレメント内の <record> エレメントで表されます。<relatedset> エレメントの count 属性にはポータル内のレコードの数が指定され、table 属性にはポータルに関連付けられているテーブルが指定されます。

fmresultset 文法での XML データの例

次に、fmresultset 文法で生成される XML データの例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE fmresultset PUBLIC "-//FMI//DTD fmresultset//EN" "/fmi/xml/fmresultset.dtd">
```

```
<fmresultset xmlns="http://www.filemaker.com/xml/fmresultset" version="1.0">
    <error code="0"></error>
    <product build="06/15/2005" name="FileMaker Web Publishing Engine" version="8.0.1.32"/>
    <datasource database="art" date-format="MM/dd/yyyy" layout="web3" table="art" time-format="HH:mm:ss" timestamp-format="MM/
    dd/yyyy HH:mm:ss" total-count="16"/>
    <metadata>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="Title" not-empty="no" result="text" type="normal" />
        <field-definition auto-enter="no" global="no" max-repeat="1" name="Artist" not-empty="no" result="text" type="normal" />
        <relatedset-definition table="artlocation">
            <field-definition auto-enter="no" global="no" max-repeat="1" name="artlocation::Location"
            not-empty="no" result="text" type="normal" />
            <field-definition auto-enter="no" global="no" max-repeat="1" name="artlocation::Date"
            not-empty="no" result="date" type="normal" />
        </relatedset-definition>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="Style" not-empty="no" result="text" type="normal"/>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="length" not-empty="no" result="number"
        type="calculation"/>
    </metadata>
    <resultset count="1" fetch-size="1">
        <record mod-id="2" record-id="3">
            <field name="Title">
                <data>Still Life with Apples, Cup and Glass</data>
            </field>
            <field name="Artist">
                <data>Paul Cezanne</data>
            </field>
            <relatedset count="2" table="artlocation">
                <record mod-id="1" record-id="6">
                    <field name="artlocation::Location">
                        <data>Vault</data>
                    </field>
                    <field name="artlocation::Date">
                        <data>07/07/1997</data>
                    </field>
                </record>
                <record mod-id="0" record-id="18">
                    <field name="artlocation::Location">
                        <data>Home</data>
                    </field>
                    <field name="artlocation::Date">
                        <data>08/01/2001</data>
                    </field>
                </record>
            </relatedset>
            <field name="Style">
                <data>Impressionist</data>
            </field>
            <field name="length">
```

```
            <data>37</data>
        </field>
    </record>
</resultset>
</fmresultset>
```

他の FileMaker XML 文法の使用

他の FileMaker XML 文法には、フィールドタイプ、値一覧、およびレイアウトに関する情報が含まれます。FMPXMLRESULT は、fmresultset と機能的に同等です。レイアウト内の値一覧およびフィールド表示情報にアクセスするには、FMPXMLLAYOUT 文法を使用する必要があります。FMPXMLRESULT および FMPXMLLAYOUT 文法は、データ交換用としてより簡潔になっています。

FMPXMLRESULT 文法を使用するには、Web 公開エンジンに XML ドキュメントを要求する URL で、次の文法名を指定します。

FMPXMLRESULT.xml

次に例を示します。

http://192.168.123.101/fmi/xml/FMPXMLRESULT.xml?-db=employees&-lay=family&-findall

FMPXMLLAYOUT 文法を使用するには、Web 公開エンジンに XML ドキュメントを要求する URL で、次の文法名を -view クエリーコマンドとともに指定します。

FMPXMLLAYOUT.xml

次に例を示します。

http://192.168.123.101/fmi/xml/FMPXMLLAYOUT.xml?-db=employees&-lay=family&-view

注意　FMPXMLRESULT および FMPXMLLAYOUT 文法を指定する場合は、文法名を大文字で入力してください。

生成された XML ドキュメントで、Web 公開エンジンは、ドキュメントの <?xml...?> 命令直後の 2 行目にある <!DOCTYPE> 命令で指定されている文法の文書型定義を参照します。<!DOCTYPE> 命令によって、文法の DTD をダウンロードするための URL が指定されます。

FMPXMLRESULT 文法のエレメントの説明

FMPXMLRESULT 文法では、<DATABASE> エレメントに、NAME、RECORDS、DATEFORMAT、LAYOUT、および TIMEFORMAT 属性が含まれます。

XML ドキュメントの日付の書式は、<DATABASE> エレメントの DATEFORMAT 属性で指定されます。XML ドキュメントの時刻の書式は、<DATABASE> エレメントの TIMEFORMAT 属性で指定されます。FMPXMLRESULT 文法と fmresultset 文法の日付および時刻の書式は同じです。27 ページの「fmresultset 文法のエレメントの説明」の表を参照してください。

FMPXMLRESULT 文法の <METADATA> エレメントには 1 つまたは複数の FIELD エレメントが含まれ、各 <FIELD> エレメントには、結果セットのフィールド/列のうちの 1 つの情報が含まれます。この情報には、データベースで定義されているとおりのフィールドの名前、フィールドタイプ、空欄のフィールドの許可（Yes）/不許可（No）（EMPTYOK 属性）、および繰り返し値の最大数（MAXREPEAT 属性）が含まれます。フィールドタイプに対して有効な値は、TEXT、NUMBER、DATE、TIME、および CONTAINER です。

<RESULTSET> エレメントには、クエリーの結果として返されたすべての <ROW> エレメントと、見つかったレコードの総数の属性が格納されます。各 <ROW> エレメントには、結果セットの 1 つの行に対するフィールド/列のデータが含まれます。このデータには、行の RECORDID と MODID（84 ページの「-modid（修正 ID）クエリー引数」を参照）、および <COL> エレメントが含まれます。<COL> エレメントには、行内の 1 つのフィールド/列のデータが含まれ、複数の <DATA> エレメントは、繰り返しフィールドまたはポータルフィールドの値の 1 つを表します。

FMPXMLRESULT 文法での XML データの例

次に、FMPXMLRESULT 文法で生成される XML データの例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLRESULT PUBLIC "-//FMI//DTD FMPXMLRESULT//EN" "/fmi/xml/FMPXMLRESULT.dtd">
<FMPXMLRESULT xmlns="http://www.filemaker.com/fmpxmlresult">
    <ERRORCODE>0</ERRORCODE>
    <PRODUCT BUILD="06/15/2005" NAME="FileMaker Web Publishing Engine" VERSION="8.0.1.32"/>
    <DATABASE DATEFORMAT="MM/dd/yyyy" LAYOUT="web3" NAME="art" RECORDS="16" TIMEFORMAT="HH:mm:ss"/>
    <METADATA>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Title" TYPE="TEXT" />
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Artist" TYPE="TEXT" />
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="artlocation::Location" TYPE="TEXT"/>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="artlocation::Date" TYPE="DATE"/>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Style" TYPE="TEXT"/>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="length" TYPE="NUMBER"/>
    </METADATA>
    <RESULTSET FOUND="1">
        <ROW MODID="2" RECORDID="2">
            <COL>
                <DATA>The Dancers in Blue</DATA>
            </COL>
            <COL>
                <DATA>Edgar Degas</DATA>
            </COL>
            <COL>
                <DATA>Study</DATA>
            </COL>
            <COL>
                <DATA>01/08/1979</DATA>
            </COL>
            <COL>
                <DATA>Impressionist</DATA>
            </COL>
            <COL>
                <DATA>19</DATA>
            </COL>
        </ROW>
    </RESULTSET>
</FMPXMLRESULT>
```

<COL> エレメントの順序は、<METADATA> エレメント内の <FIELD> エレメントの順序に一致します。たとえば、<METADATA> エレメントに「Title」および「Artist」フィールドが一覧表示されている場合、「Village Market」および「Camille Pissarro」は、これと同じ順序で <RESULTSET> および <ROW> エレメントに一覧表示されます。

FMPXMLLAYOUT 文法のエレメントの説明

FMPXMLLAYOUT 文法では、レイアウト名、データベース名、およびデータベース側の対応するレイアウトで見つかった各フィールドの <FIELD> エレメントが <LAYOUT> エレメントに含まれます。各 <FIELD> エレメントでフィールドのスタイルタイプが説明され、フィールドの関連した値一覧の VALUELIST 属性がこのエレメントに含まれます。

<VALUELISTS> エレメントには、レイアウトにある各値一覧の <VALUELIST> エレメントが 1 つまたは複数含まれます（各エレメントには、値一覧の名前と、一覧の各値の <VALUE> エレメントが含まれます）。

日付、時刻、およびタイムスタンプのフィールドでは、値リストのデータはそのフィールドタイプの「fm」形式を使用して書式設定されます。「fm」形式では、MM/dd/yyyy が日付、HH:mm:ss が時刻、MM/dd/yyyy HH:mm:ss がタイムスタンプです。64 ページの「日付、時刻、および曜日拡張関数の使用」を参照してください。たとえば、「生年月日」の値リストがレイアウトの「生年月日」フィールドのポップアップメニューに使用されていて、「生年月日」フィールドが日付のタイプである場合は、その値リストに出力される値はすべて「fm」日付形式になります。

注意　レイアウトで異なるフィールドタイプを持った 2 つのフィールドが同じ値リストを共有する場合は、最初のフィールドのタイプが値リストデータの形式を決定します。

FMPXMLLAYOUT 文法での XML データの例

次に、FMPXMLLAYOUT 文法で生成される XML データの例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLLAYOUT PUBLIC "-//FMI//DTD FMPXMLLAYOUT//EN" "/fmi/xml/FMPXMLLAYOUT.dtd">
<FMPXMLLAYOUT xmlns="http://www.filemaker.com/fmpxmllayout">
    <ERRORCODE>0</ERRORCODE>
    <PRODUCT BUILD="06/15/2005" NAME="FileMaker Web Publishing Engine" VERSION="8.0.1.32"/>
    <LAYOUT DATABASE="art" NAME="web2">
        <FIELD NAME="Title">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="Artist">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="Image">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="artlocation::Location">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="artlocation::Date">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="Style">
            <STYLE TYPE="POPUPMENU" VALUELIST="style"/>
        </FIELD>
    </LAYOUT>
    <VALUELISTS>
        <VALUELIST NAME="style">
            <VALUE>Impressionist</VALUE>
            <VALUE>Modern</VALUE>
```

```
            <VALUE>Abstract</VALUE>
        </VALUELIST>
    </VALUELISTS>
</FMPXMLLAYOUT>
```

UTF-8 でエンコードされているデータについて

Web 公開エンジンによって生成されるすべての XML データは、UTF-8 形式（Unicode Transformation Format 8）を使用してエンコードされます。この形式では、データの ASCII 文字は、標準的な Unicode 形式の 16 ビットから 8 ビットに圧縮されます。XML パーサが Unicode と UTF-8 エンコードをサポートしている必要があります。

UTF-8 エンコードの場合、英語で使用される標準的な ASCII 文字セットは 0 から 127 の値で直接表され、それ以上の値の Unicode 文字については、マルチバイトのエンコードが使用されます。

注意　UTF-8 ファイルをサポートする Web ブラウザまたはテキストエディタプログラムを使用してください。

UTF-8 エンコード形式には次の特徴があります。

- ASCII 文字は、すべて 1 バイトのUTF-8 文字になります。したがって、ASCII で有効な文字列は、UTF-8 文字列としても有効です。
- ASCII 以外の文字（ビットの大きい文字セット）は、マルチバイト文字の一部です。
- UTF-8 文字の 1 バイト目は、その文字に含まれる追加バイトの数を示します。
- マルチバイト文字の 1 バイト目は、2 バイト目以降と容易に区別できるため、データストリームのどの位置でも、文字の開始位置を簡単に判断できます。
- UTF-8 と UTF-16 のような他の Unicode エンコーディングスキーマは簡単に相互変換できます。
- UTF-8 でエンコードしたテキストは比較的小さくなり、ASCII 文字の比率が大きいテキストの場合、Unicode よりも小さくなります。最も条件が悪い場合でも、UTF-8 文字列は、同じ UTF-16 文字列と比較して 50% 程度しか大きくなりません。

FileMaker クエリー文字列を使用した XML データの要求

FileMaker データベースに XML データを要求するには、クエリー文字列で FileMaker クエリーコマンドと引数を使用します。たとえば、URL 内の次のクエリー文字列で -findall クエリーコマンドを使用して、「products」という名前の FileMaker データベースに含まれるすべての製品の一覧を要求できます。

```
http://192.168.123.101/fmi/xml/fmresultset.xml?-db=products-lay=sales&-findall
```

クエリー文字列に含めるクエリーコマンドは、-new など、1 つだけにする必要があります。ほとんどのクエリーコマンドでは、対応するさまざまなクエリー引数もクエリー文字列で指定する必要があります。たとえば、-dbnames 以外のすべてのクエリーコマンドでは、クエリー対象のデータベースを指定する -db 引数が必要です。

クエリーコマンドと引数は、URL で使用したり、FileMaker XSLT スタイルシートの <?xslt-cwp-query?> 処理命令で使用することもできます。第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

このセクションでは、FileMaker クエリーコマンドと引数の概要を説明します。クエリー文字列でのクエリーコマンドと引数の使用の詳細については、付録 A「クエリー文字列で使用される有効な名前」を参照してください。

注意　Web 公開エンジンでは、FileMaker XSLT スタイルシート専用に定義されている追加のクエリーコマンド（-process）と 3 つのクエリー引数もサポートされています。49 ページの「FileMaker XSLT スタイルシートでのクエリー文字列の使用」を参照してください。

使用するクエリーコマンド名	実行するコマンド
-dbnames	ホストされているすべての Web 共有データベースの名前の取得
-delete	レコードの削除
-dup	レコードの複製
-edit	レコードの編集
-find	レコードの検索
-findall	すべてのレコードの検索
-findany	ランダムなレコードの検索
-layoutnames	ホストされている Web 共有データベースで利用可能なすべてのレイアウトの名前の取得
-new	新規レコードの作成
-scriptnames	ホストされている Web 共有データベースで利用可能なすべてのスクリプトの名前の取得
-view	FMPXMLLAYOUT 文法が指定されている場合は、データベースからのレイアウト情報の取得。fmresultset または FMPXMLRESULT 文法が指定されている場合は、XML ドキュメントの <metadata> セクションおよび空のレコードセットの取得。

使用するクエリー引数名	使用するクエリーコマンド
-db（データベース名）	-dbnames および -process（XSLT リクエストのみ）以外のすべてのクエリーコマンドで必要です。
-field	オブジェクトリクエストの URL でフィールドを指定するために必要です。24 ページの「XML ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。
フィールド名	-edit では、少なくとも 1 つのフィールド名が必要です。-find のオプションです。82 ページの「フィールド名（オブジェクトフィールド以外のフィールド名）クエリー引数」を参照してください。
フィールド名.op（演算子）	-find のオプションです。
-lay（レイアウト名）	-dbnames、-layoutnames、-scriptnames、および -process（XSLT リクエストのみ）以外のすべてのクエリーコマンドで必要です。
-lay.response（XML 応答に対するレイアウトの切り替え）	-dbnames、-layoutnames、-scriptnames、および -process（XSLT リクエストのみ）以外のすべてのクエリーコマンドのオプションです。
-lop（論理演算子）	-find のオプションです。
-max（最大レコード）	-find および -findall のオプションです。
-modid（修正 ID）	-edit のオプションです。
-recid（レコード ID）	-edit、-delete、および -dup で必要です。-find のオプションです。
-script（スクリプトの実行）	-find、-findall、-findany、-new、-edit、-delete、-dup、および -view のオプションです。
-script.param（-scriptで指定されたスクリプトに引数の値を渡します）	-script のオプションです。
-script.prefind（-find、-findany、および -findall の前にスクリプトを実行）	-find、-findany および -findall のオプションです。
-script.prefind.param（-script.prefindで指定されたスクリプトに引数の値を渡します）	-script.prefindのオプションです。

使用するクエリー引数名	使用するクエリーコマンド
-script.presort（ソートの前にスクリプト実行）	-find および -findall のオプションです。
-script.presort.param（-script.presortで指定されたスクリプトに引数の値を渡します）	-script.presortのオプションです。
-skip（レコードのスキップ）	-find および -findall のオプションです。
-sortfield.[1-9]（フィールドのソート）	-find および -findall のオプションです。
-sortorder.[1-9]（ソート順）	-find および -findall のオプションです。
-stylehref（スタイルシートの HREF）	すべてのクエリーコマンドでオプションです（-styletype に対してスタイルシートの URL を指定します）。
-styletype（スタイルシートの種類）	すべてのクエリーコマンドでオプションです（クライアントサイドのスタイルシートを指定します）。

XML 応答に対するレイアウトの切り替え

-lay クエリー引数には、XML データを要求する場合に使用するレイアウトを指定します。多くの場合、リクエストから生成されるデータの処理には、同じレイアウトが適しています。場合によっては、セキュリティ上の理由から結果の表示に使用するレイアウトには存在しないフィールドが含まれる別のレイアウトを使用して、データを検索できます。フィールド内のデータを検索するには、XML リクエストで指定したレイアウトにそのフィールドが配置されている必要があります。

XML 応答を表示するために、XML リクエストの処理に使用するレイアウトとは異なるレイアウトを指定するには、オプションの -lay.response クエリー引数を使用できます。

たとえば、次のリクエストは、「Budget」レイアウト上の「Salary」フィールドで 100,000 を超える値を検索します。結果のデータは「ExecList」レイアウトを使用して表示されます。このレイアウトには、「Salary」フィールドは含まれません。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=Budget&Salary=100000&Salary.op=gt&-find
&-lay.response=ExecList

XML リクエストの処理方法の理解

XML リクエストの処理と XML ドキュメントの生成を制御するクエリー引数は複数あります。

次に、FileMaker Server と Web 公開エンジンが XML リクエストを処理する順序を示します。

1. -lay クエリー引数を処理します。
2. クエリーに指定されたグローバルフィールドの値を設定します（「.global=」は URL の一部）。
3. -script.prefind クエリー引数を処理します（指定されている場合）。
4. -find や -new などのクエリーコマンドを処理します。
5. -script.presort クエリー引数を処理します（指定されている場合）。
6. 結果のデータをソートします（ソートが指定されていた場合）。
7. -lay.response クエリー引数を処理して別のレイアウトに切り替えます（指定されている場合）。
8. -script クエリー引数を処理します（指定されている場合）。
9. XML ドキュメントを生成します。

上のいずれかの手順でエラーコードが生成された場合、リクエストの処理は停止し、以降の手順は実行されません。ただし、リクエスト内の前の手順は引き続き実行されます。

たとえば、現在のレコードを削除し、レコードをソートしてからスクリプトを実行するリクエストがあるとします。-sortfield 引数で存在しないフィールドが指定されている場合、このリクエストでは、現在のレコードが削除され、エラーコード 102（「フィールドが見つかりません。」）が返されますが、スクリプトは実行されません。

サーバーサイドおよびクライアントサイドでのスタイルシートの処理の使用

Web 公開エンジンでは、サーバーサイドでの XSLT スタイルシートの処理がサポートされています。また、クライアントサイドでのスタイルシートの処理を指定するクエリー引数を指定することもできます。

スタイルシートを処理するこれら２つの方法の違いと、クライアントサイドの処理を使用した場合のセキュリティへの影響を理解することが重要です。サーバーサイドでの処理では、Web ユーザがフィルタされていない XML データにアクセスすることはできないため、サーバーサイドでの処理の方がクライアントサイドでの処理よりも安全です。サーバーサイドでの処理では、データは、データの所有者または XSLT スタイルシートの作成者が適切と判断した形式で提示されます。サーバーサイドの処理では、データベース名、フィールド名、および他の実装の詳細は Web ユーザから隠されます。さらに、サーバーサイドの処理を使用して、静的に定義されたクエリー引数を指定することもでき、データベース名など、権限のないクエリーコマンドとクエリー引数の使用を防止できます。第 4 章「XSLT を使用したカスタム Web 公開の概要」および 第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

ソリューションでクライアントサイドでのスタイルシートの処理が必要な場合は、FileMaker クエリー文字列リクエストに -styletype および -stylehref 引数を含めることで、Web 公開エンジンによって各文法で XML スタイルシート処理命令を生成できます。XML ドキュメントの表示には、CSS（Cascading Style Sheet）または XSLT スタイルシートを使用できます。

- -styletype 引数は、TYPE 属性（type=text/css または type=text/xsl）の値を設定するために使用されます。
- -stylehref 引数は、絶対パスを使用してスタイルシートの場所を指定する HREF 属性の値を設定するために使用されます。たとえば、href=/mystylesheet.css または href=/stylesheets/mystylesheet.xsl のように指定します。スタイルシートの名前には任意の名前を指定できますが、.css または .xsl のいずれかの拡張子を含める必要があります。

次に、クライアントサイドのスタイルシートの処理を生成する FileMaker クエリー文字列の例を示します。

```
http://localhost/fmi/xml/fmresultset.xml?-db=products-lay=sales&-findall&-styletype=text/xsl
&-stylehref=/mystylesheet.xsl
```

注意 この例では、「-stylehref=/document.xsl」の「/」は、スタイルシートが Web サーバーソフトウェアのルートフォルダにあるために使用されています。絶対パスを使用して Web サーバー上での場所を指定するスタイルシートには、URL を使用してください。スタイルシートは、別の Web サーバー上に配置することもできます。

Web 公開エンジンにより、このリクエストに基づいて、次の処理命令が XML ドキュメントに含められます。

```
<?xml–stylesheet type="text/xsl" href="/mystylesheet.xsl"?>
```

クライアントサイドでの処理に使用するスタイルシートは、HREF 属性の URL に絶対パスで指定された場所にある Web サーバーにコピーまたは配置します。

重要 クライアントサイドでの処理に使用するスタイルシートは、「xslt-template-files」フォルダ内には配置しないでください。このフォルダは、サーバーサイドでの XSLT スタイルシートの処理に使用されます。44 ページの「Web サイトまたはプログラムでの FileMaker XSLT スタイルシートの使用」を参照してください。

注意 一部のWeb ブラウザでは、クライアントサイドでの処理はサポートされていません。詳細については、Web ブラウザのマニュアルを参照してください。

XML ドキュメントへのアクセスに関するトラブルシューティング

Web 公開エンジンを使用して XML ドキュメントにアクセスできない場合は、次の点を確認してください。

- XML カスタム Web 公開用にデータベースの拡張アクセス権が設定されていて、ユーザアカウントに割り当てられている。17 ページの「データベースでのカスタム Web 公開の有効化」を参照してください。
- データベースが FileMaker Server によってホストされて開かれている。FileMaker Server Admin ヘルプを参照してください。
- 使用しているデータベースアカウント名とパスワードが正しい。
- Web サーバーおよび Web 公開エンジンが実行されている。
- Web 公開エンジンで [XML 公開:] が有効になっている。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

第 4 章
XSLT を使用したカスタム Web 公開の概要

FileMaker XSLT を使用すると、XML データの変換、フィルタ、または書式設定を行って、Web ブラウザ、または他のプログラムやアプリケーションで使用できます。この章では、FileMaker XSLT スタイルシートと、XSLT スタイルシートの作成を開始する場合に役立つ、Site Assistant および CDML Converter という 2 つのツールの概要を説明します。FileMaker XSLT スタイルシートの構造の詳細については、第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

FileMaker XSLT スタイルシートについて

FileMaker XSLT スタイルシートを使用して、次の処理を行うことができます。

- データベースのどのフィールドを公開するかをスタイルシートで制御することによってデータをフィルタする
- データベース名やフィールド名などのメタデータを隠す
- Web ページにデータを表示する書式を設定したり、ユーザがデータを操作する方法を制御する
- データを HTML またはテキスト（vCard やコンマ区切り値など）として出力する
- データを FileMaker XML 文法から別の XML 文法（SVG（Scalable Vector Graphics）など）に変換して、別のデータベースやアプリケーションで使用する
- FileMaker データのサブセットを他の Web サイトに統合したり、FileMaker データベースとは大きく異なる他のミドルウェアやアプリケーションを使用して統合する
- 公開されるフィールド名を変更して、データベースデザイン情報の不正使用を防止する

注意　FileMaker Server の XSLT を使用したカスタム Web 公開は、XSLT 1.0 の W3C 勧告に基づいています。XSLT 1.0 の詳細については、www.w3.org を参照してください。セッション管理、電子メールの送信、および Cookie とヘッダへのアクセスなどの追加の機能は、FileMaker XSLT 拡張関数によって提供されます。詳細については、53 ページの「FileMaker XSLT 拡張関数と引数の使用」を参照してください。Web 公開エンジンでは、XSL-FO（XSL Formatting Objects）はサポートされていません。

FileMaker XSLT スタイルシートの使用例

次に、FileMaker XSLT スタイルシートを使用可能なさまざまな例の一部を示します。

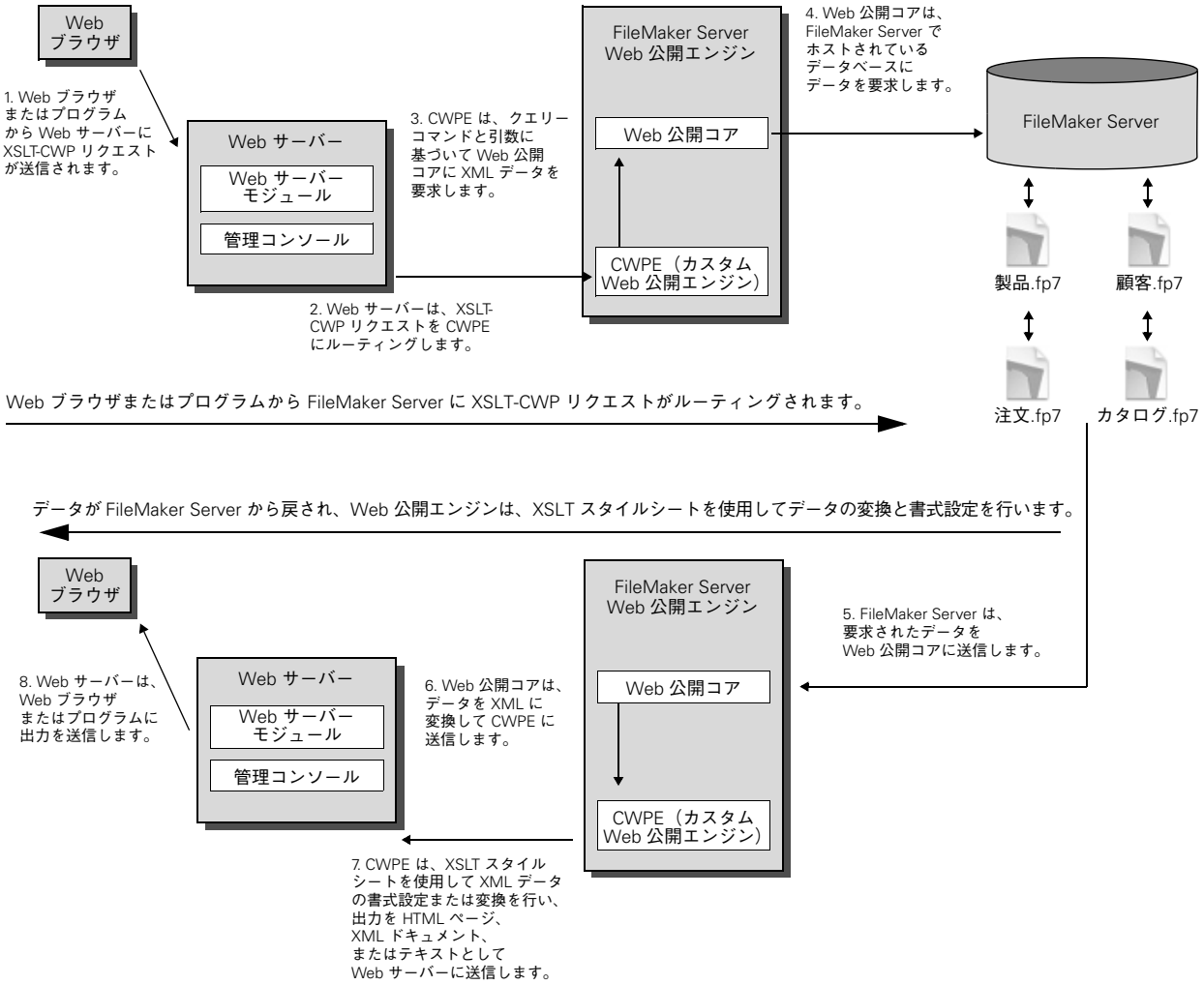
- FileMaker データベースのデータのサブセットが含まれるテーブルを Web ページに挿入して、Web ユーザが参照できます。たとえば、テーブルに名前と住所を含め、電話番号は含めないようにすることができます。不正アクセスを防止するために、Web ページでは、データに対して、FileMaker データベースの実際のフィールド名（「姓」など）ではなく、汎用のラベル（「名前」など）を表示できます。
- FileMaker ポータルのデータを他のデータソースの情報に統合する Web ページやアプリケーションを作成できます。
- FileMaker データベースに保存されている連絡先情報から vCard を作成するボタンを Web ページ上に作成できます。
- FileMaker データベースの XML データを、スプレッドシートやデータベースアプリケーションで開くことができる XML 文法に変換できます。

XSLT を使用したカスタム Web 公開の使用の開始

標準的な XML と XSLT の知識がある場合、FileMaker に固有の XSLT 拡張関数や、クエリーコマンドと引数など、FileMaker XML と XSLT に特有の詳細事項をいくつか学べば、すぐに Web 公開エンジンを使い始めることができます。Site Assistant と CDML Converter は、スタイルシートの作成を開始する場合や、スタイルシートの構成を学ぶ場合に役立つツールです。その後、好みの XML および XSLT オーサリングツールを使用して、スタイルシートの機能を向上させることができます。

Web 公開エンジンが XML データと XSLT スタイルシートに基づいてページを生成する方法

Web サーバーに XSLT-CWP（XSLT カスタム Web 公開）リクエストが送信されると、Web 公開エンジンは、スタイルシートおよび URL で定義されたクエリーコマンドと引数に基づいて FileMaker データベースに対してクエリーを実行し、XSLT スタイルシートの命令に従ってデータを出力します。



XSLT を使用したカスタム Web 公開を使用するための一般的な手順

次に、XSLT を使用したカスタム Web 公開を使用するための手順の概要を示します。

1. Web 公開エンジン管理コンソールで、[XSLT 公開:] が有効になっていることを確認します。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。
2. 公開する各 FileMaker データベースを FileMaker Pro を使用して開き、データベースで、XSLT を使用したカスタム Web 公開の fmxslt 拡張アクセス権が有効になっていることを確認します。17 ページの「データベースでのカスタム Web 公開の有効化」を参照してください。

注意　スタイルシートを開発する際は、エンドユーザに提供するアクセス権セットと同等の FileMaker データベースのアクセス権セットを使用してください。同等のアクセス権セットを使用しなかった場合、開発者は、エンドユーザが使用できない FileMaker データベースのレイアウトや機能にアクセスできることになり、同じ動作を実現できません。

3. FileMaker データベースからの XML データの書式設定と変換を行うために、FileMaker に固有の XSLT 拡張関数、クエリーコマンド、およびクエリー引数が含まれる XSLT スタイルシートを作成します。

   FileMaker Site Assistant を使用し、サイトの開始点として、1 つまたは複数の基本的な XSLT スタイルシートを作成できます。次のセクション「FileMaker Site Assistant を使用した FileMaker XSLT スタイルシートの生成」を参照してください。

   既存の CDML ソリューションを利用する場合は、CDML Converter を使用して、CDML フォーマットフィルを XSLT スタイルシートに変換できます。42 ページの「FileMaker CDML Converter の使用」を参照してください。

   また、必要に応じて、独自の XSLT オーサリングツールやテキスト編集ツールを使用して XSLT スタイルシートを変更したり、スタイルシートを最初から開発することができます。第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

4. XSLT スタイルシートを「xslt-template-files」フォルダにコピーまたは保存します。「xslt-template-files」フォルダは、Web 公開エンジンがインストールされているホスト上の FileMaker Server フォルダ内の「Web Publishing」フォルダにあります。

   スタイルシートは、「xslt-template-files」フォルダ内のオプションのフォルダまたはフォルダ階層に保存することもできます。

5. 静的ファイルを Web サーバーに配置します。44 ページの「Web サイトまたはプログラムでの FileMaker XSLT スタイルシートの使用」を参照してください。

6. XSLT スタイルシートを使用する Web サイトやプログラムを作成または変更します。

   たとえば、Web ユーザを XSLT スタイルシートに自動転送するか、または XSLT スタイルシートへのリンクが指定された index.html などの静的ページを Web サイトに使用できます。

7. サイトまたはプログラムのセキュリティメカニズムが設定されていることを確認します。『FileMaker セキュリティガイド』を参照してください。

8. Web ユーザ用に定義されているものと同じアカウントとアクセス権を使用して、XSLT スタイルシートを使用するサイトまたはプログラムをテストします。

9. サイトまたはプログラムを使用可能にし、ユーザに通知します。

FileMaker Site Assistant を使用した FileMaker XSLT スタイルシートの生成

FileMaker Site Assistant は、XSLT を使用した FileMaker カスタム Web 公開で起点として使用する基本的な XSLT スタイルシートを作成するために使用できるアプリケーションです。Site Assistant は、FileMaker XSLT スタイルシートの構造を学ぶのに効果的です。必要に応じて、独自の XSLT スタイルシートオーサリングツールやテキスト編集ツールを使用してスタイルシートを変更できます。Site Assistant を使用して既存のスタイルシートを編集または更新することはできませんが、サイト全体で使用する初期スタイルシートを生成したり、既存のサイトに基本的な機能（レコードの削除など）を追加するための単一のスタイルシートを作成することが可能です。

Site Assistant を使用して、カスタム Web 公開経由で FileMaker データベースを操作する場合に役立つあらゆるタイプのページに対して XSLT スタイルシートを生成できます。Site Assistant で選択するオプションに応じて、ユーザに次の操作を許可するサイトを作成できます。

- 一度に 1 つのレコードをブラウズする
- データベース内のすべてのレコードのリストを表示する
- データベースを検索して、結果をリストで表示する
- レコードをソートする
- レコードを追加する

- レコードを編集および複製する
- レコードを削除する
- 集計レポートを表示する

また、生成される他の XSLT スタイルシートページへのリンクを含むホームページを生成することもできます（オプション）。

Web ユーザがいずれかの XSLT スタイルシートを参照する HTTP リクエストと URL を送信すると、Web 公開エンジンは、各スタイルシートを使用して FileMaker データベースから動的にデータを取得します。Web 公開エンジンは、スタイルシートを使用して XML データの変換と書式設定を行い、Web ユーザが操作できる最終的な HTML ページを生成します。

注意　Site Assistant のスタイルシートは、fmresultset XML 文法に基づいて FileMaker XML データを HTML ページに変換するもので、FileMaker XML エクスポートなど、XML データの他の用途には使用できません。

Site Assistant のインストール

Site Assistant のインストールの詳細については、『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

Site Assistant を使用する前に

Site Assistant を使用してデータベース用の XSLT スタイルシートを生成する前に、次の作業を行ってください。

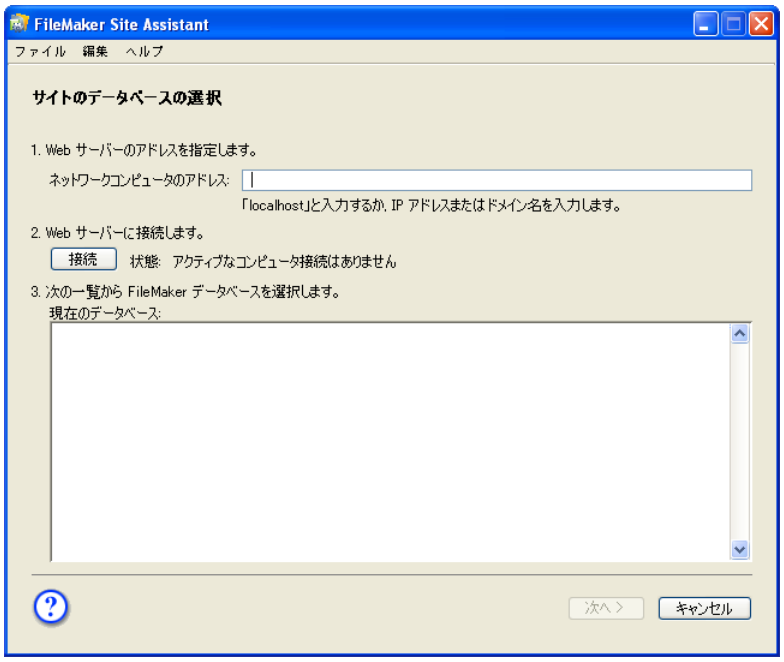
- データベースで拡張アクセス権「fmxslt」を設定する。Site Assistant を実行する際は、Web ユーザに許可するアクセス権セットと同等のアクセス権セットを使用します。17 ページの「データベースでのカスタム Web 公開の有効化」を参照してください。
- FileMaker Server でデータベースを開いてホストする。FileMaker Server Admin ヘルプを参照してください。
- Web サーバーおよび Web 公開エンジンが実行されていることを確認する。
- XSLT スタイルシートを使用およびテストするために、Web 公開エンジンで [XSLT 公開:] を有効にする。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

Site Assistant の起動

Site Assistant を起動するには、次のいずれかの操作を行います。

- FileMaker Site Assistant アプリケーションのアイコンをダブルクリックします。

- Windows の場合、[スタート] ボタンをクリックし、[プログラム] メニューから [FileMaker Site Assistant] を選択します。



FileMaker Site Assistant

Site Assistant の使用

Site Assistant を使用するための手順ごとの操作の詳細については、Site Assistant ヘルプを参照してください。Site Assistant によって生成されたスタイルシートの使用の詳細については、44 ページの「Web サイトまたはプログラムでの FileMaker XSLT スタイルシートの使用」を参照してください。

重要 Site Assistant を使用する際に、複数のテーブルが含まれるデータベースを選択する場合は、同じテーブルに関連付けられているレイアウトを選択してください。同じテーブルに関連付けられていないレイアウトを選択すると、生成されるサイトは予想外の結果を返します。たとえば、データベースに「製品」テーブルと「顧客」テーブルが含まれる場合に、検索用のページ、レコード編集用のページ、およびレコード追加用のページに使用するレイアウトを選択するときは、これらのレイアウトがすべて同じテーブルに関連付けられていることを確認します。

Site Assistant によって生成されるスタイルシートについて

Site Assistant によって生成される XSLT スタイルシートには、FileMaker に固有の複数の処理命令、エレメント、および引数が含まれます。次に、スタイルシートに含まれる処理命令、エレメント、および引数の例をいくつか示します。

- <?xslt-cwp-query params="クエリー文字列"?> 処理命令は、使用する XML 文法を指定し、Site Assistant で選択したデータベースの名前を静的に定義します。50 ページの「静的に定義されたクエリーコマンドとクエリー引数の使用」を参照してください。
- <xsl:param name="request-query"/> エレメントは、リクエストまたは HTML フォームデータ内のクエリー情報にアクセスするために使用されます。たとえば、Site Assistant のスタイルシートでこのエレメントを使用すると、現在のリクエストのクエリー情報にアクセスして、対象レコードで現在の場所を判断したり、前後のレコードへのリンクを作成することができます。54 ページの「リクエストのクエリー情報へのアクセス」を参照してください。
- <xsl:param name="authenticated-xml-base-uri"/> エレメントは、リクエスト内でさらに多くの XML データが必要になった場合に、リクエスト内の認証済みのベース URI にアクセスするために使用されます。このエレメントは、常にスタイルシートに含まれるわけではありません。55 ページの「認証済みベース URI 引数の使用」を参照してください。

さらに、エラーを定義するための「utilities.xsl」スタイルシート、および Site Assistant のスタイルシートによって呼び出される一般的な XSLT テンプレートも生成されます。

Site Assistant のスタイルシートの他のセクションの詳細については、第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

FileMaker CDML Converter の使用

FileMaker Server 7 Advanced では、FileMaker データベースのカスタム Web 公開の言語として、CDML（FileMaker 独自のマークアップ言語）の代わりに XSLT が採用され、CDML はサポートされなくなっています。CDML フォーマットファイルから FileMaker XSLT スタイルシートに Web サイトを移行するには、FileMaker CDML Converter を使用します。

FileMaker CDML Converter について

FileMaker CDML Converter は、既存の CDML フォーマットファイルを、XSLT を使用したカスタム Web 公開と互換性がある XSLT スタイルシートに変換するアプリケーションです。CDML Web サイトの移行を開始する場合や、FileMaker XSLT スタイルシートの構造を理解する場合に便利です。

CDML Converter を使用する前に、運用環境から一時作業ディレクトリに CDML フォーマットファイルをコピーすることをお勧めします。CDML Converter の使用後は、生成されたスタイルシートと変換ログを確認してください。変換された XSLT スタイルシートは、変更せずにそのまま FileMaker Server で使用できる場合もありますが、CDML Converter では一部の CDML タグを XSLT スタイルシートに変換できないことがあり、変換された XSLT スタイルシートを手動で編集しなければならない場合もあります。必要に応じて、独自の XSLT スタイルシートオーサリングツールやテキスト編集ツールを使用してスタイルシートを変更できます。また、データベースレコードで CDML フォーマットファイルが参照されている場合も、データベースへの変更が必要になることがあります。これは、変換された XSLT スタイルシートの名前には、.xsl というファイル拡張子が使用されるためです。

CDML Converter は、既存の CDML フォーマットファイルを新しいスタイルシートに変換する目的でのみ使用できます。変換された XSLT スタイルシートの編集には使用できません。CDML Converter を使用して、CDML Web サイトを XSLT に効果的に移行するには、CDML 開発者として十分な経験があり、XSLT を使用したカスタム Web 公開に精通している必要があります。

注意　生成される XSLT スタイルシートは、fmresultset XML 文法に基づいて、FileMaker データを HTML ページに変換します。

CDML Converter のインストール

CDML Converter のインストールの詳細については、『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

CDML Converter の起動と使用

重要　最初に、「cdml_format_files」フォルダ（このフォルダを使用するように選択した場合）のすべてのファイルと、FileMaker Pro の「Web」フォルダのすべてのファイルを一時作業ディレクトリにコピーし、コピーしたファイルに対して変換処理を行うことをお勧めします。

注意　ソースファイルを XSLT スタイルシートに変換する場合、CDML Converter では、同じファイル名が保持されますが、ファイル拡張子は .xsl に変更されます。変換元のフォルダに、名前が同じで拡張子が異なる複数のファイル（「Myfile.html」と「Myfile.cdml」など）が含まれる場合は、変換時にファイルが上書きされる問題が発生します。たとえば、CDML Converter は、「Myfile.html」を「Myfile.xsl」に変換してから、「Myfile.cdml」を「Myfile.xsl」に変換しようと試みます。「Myfile.xsl」はすでに存在するため、CDML Converter によって、上書きを確認するダイアログボックスが表示されます。変換前にこの問題を避けるには、変換元のフォルダ内のファイルに固有のファイル名が付いていることを確認してください。ファイル名を変更した場合は、必ず、変換前にそれらのファイルへの参照も変更してください。

CDML Converter を起動して使用するには、次の操作を行います。

1. CDML Converter を起動するために、次のいずれかの操作を行います。
   - FileMaker CDML Converter アプリケーションのアイコンをダブルクリックします。

- Windows の場合、[スタート] ボタンをクリックし、[プログラム] メニューから [FileMaker CDML Converter] を選択します。

FileMaker CDML Converter

2. [変換元のフォルダ] で、[選択...] をクリックして、CDML ソースファイルが保存されているフォルダを指定します。
3. CDML ソースファイルが含まれるフォルダを検索して選択し、[選択] をクリックします。
4. [ファイルのテキストエンコード:] で、CDML ソースフォーマットファイルのエンコードを選択します。51 ページの「リクエストのテキストエンコードの設定」を参照してください。

   注意　変換セッションで使用するすべてのソースフォーマットファイルは、同じテキストエンコードでなければなりません。

5. [変換先のフォルダ] で、[選択...] をクリックして、変換された XSLT スタイルシートおよび CDML Web サイトの他のファイルを保存するフォルダを指定します。

   CDML Converter によって、変換先のフォルダ内に、フォルダ階層と、ソースファイルに対応するファイルのセットが作成されます。「images」サブフォルダ内の GIF ファイルなど、CDML に変換する必要のないファイルは、変更されずに変換元のフォルダからコピーされます。変換されたファイルは、ファイル名は同じですが、ファイル拡張子は .xsl に変更されています。

6. XSLT スタイルシートを保存するハードディスク上のフォルダを検索して選択し、[選択] をクリックします。
7. [開始] をクリックします。

   CDML Converter によって CDML フォーマットファイルが変換され、指定した変換先のフォルダに、変換された XSLT スタイルシートとともに変換ログが保存されます。変換ログは、CDML Converter のウインドウにも表示されます。

8. [終了] をクリックします。

CDML Converter によって生成されたスタイルシートの確認と修正

CDML Converter の使用後に、CDML Converter のウインドウに表示される変換ログを確認するか、または変換先のフォルダ内にある変換ログを開くことで、変換処理に関するエラー情報を取得できます。変換ログファイルは「cdml2xsl_<日付と時刻>.log」という名前で、<日付と時刻> は、変換を開始した日付と時刻です。

変換時に警告またはエラーが発生した場合、CDML Converter によって変換ログにメッセージが追加され、変換された XSLT スタイルシートに、問題を説明する XSLT コメントタグが挿入されます。エラーと警告は、XSLT コメントタグまたは変換された XSLT スタイルシートに、次のいずれかの形式で示されます。

<!-- CDML Converter エラー : <エラーの説明> -->

<!-- CDML Converter 警告: <警告の説明> -->

CDML Converter は、CDML から XSLT へのマッピング規則の所定のセットを使用して、CDML フォーマットファイルを変換します。すべての変換エラーは、CDML Converter が CDML から XSLT への正しい変換方法を自動的には判断できなかった場合に発生します。すべての変換エラーに対して、CDML ファイルでエラーの原因を手動で修正してファイルをもう一度変換するか、またはテキストエディタや XSLT スタイルシートエディタを使用して、変換された XSLT スタイルシートで問題を手動で修正する必要があります。

次に、変換エラーの一般的なタイプを示します。

- XSLT を使用したカスタム Web 公開で CDML タグがサポートされていない。たとえば、–dbclose CDML アクションタグはサポートされていません。このタイプのエラーを修正するには、サポートされているタグを使用するように CDML タグを変更するか、CDML フォーマットファイルから該当する機能を削除するか、または XSLT スタイルシートで論理を修正します。
- CDML タグを認識できない。たとえば、CDML タグのスペルが間違っている場合があります。このタイプのエラーを修正するには、有効な構文を使用するように CDML タグを変更するか、タグを削除するか、または XSLT スタイルシートで論理を修正します。
- CDML 引数を認識できない。たとえば、XSLT スタイルシートでは、CDML タグ [FMP-ValueList:フィールド名, List=値一覧名] の CDML List 引数はサポートされません。このタイプのエラーを修正するには、有効な構文を使用するように CDML 引数を変更するか、引数を削除するか、または XSLT スタイルシートで論理を修正します。

CDML と XSLT のマッピング規則、および CDML タグの変換を完了できなかったため XSLT ステートメントを手動で変更しなければならない他の状況の詳細については、付録 C「CDML ソリューションの FileMaker XSLT への変換」を参照してください。

注意

- データベースフィールドから CDML フォーマットファイルを参照するクエリー引数がある場合は、データベース内の参照を手動で更新する必要があります。変換された XSLT スタイルシートでこのような参照が見つかった場合は、データベースで変更を行うよう通知するメモが XSLT スタイルシートに挿入されます。
- 変換された XSLT スタイルシートに FileMaker の日付または時刻の拡張関数（fmxslt:get_date() など）が含まれる場合、この関数は、FileMaker Server に対して設定されている文字列の「fm」書式を使用します。「fm」書式は、日付に対しては「MM/dd/yyyy」、時刻に対しては「HH:mm:ss」、タイムスタンプに対しては「MM/dd/yyyy HH:mm:ss」です。64 ページの「日付、時刻、および曜日拡張関数の使用」を参照してください。変換後に、これらの関数に渡される日付および時刻の書式の文字列を手動で変更およびローカライズする必要があります。たとえば、月/日/年の書式を年/月/日の書式に変更する必要があります。

CDML Converter によって生成されたスタイルシートの使用

変換された XSLT スタイルシートでエラーを修正したら、Web 公開エンジンで使用できます。次のセクション「Web サイトまたはプログラムでの FileMaker XSLT スタイルシートの使用」を参照してください。

CDML Converter によって生成されたスタイルシートのテスト

変換された XSLT スタイルシートは、運用環境で使用する前に十分にテストする必要があります。第 6 章「サイトのテストと監視」を参照してください。

Web サイトまたはプログラムでの FileMaker XSLT スタイルシートの使用

Web サイトやプログラムで Web 公開エンジンとともにスタイルシートを使用するための手順は、Site Assistant または CDML Converter を使用して XSLT スタイルシートを生成した場合も、独自のスタイルシートを最初から作成した場合も同じです。

Web サイトまたはプログラムで FileMaker XSLT スタイルシートを使用するには、次の操作を行います。

1. XSLT スタイルシートを「xslt-template-files」フォルダにコピーまたは保存します。「xslt-template-files」フォルダは、Web 公開エンジンがインストールされているホスト上の FileMaker Server フォルダ内の「Web Publishing」フォルダにあります。

スタイルシートは、「xslt-template-files」フォルダ内のオプションのフォルダまたはフォルダ階層に保存することもできます。

2. XSLT スタイルシートで静的イメージや HTML ファイルなどの静的ファイルが参照されている場合は、静的ファイルを、Web サーバーのルートフォルダ内に元のフォルダ階層で配置します。相対パスが保持されていることを確認します。

   たとえば、HTML タグ <img src="logo.jpg"> を使用して、XSLT スタイルシートで「logo.jpg」という名前のイメージファイルを参照している場合、「logo.jpg」ファイルは、Web サーバー上の次の場所に配置する必要があります。

   < ルートフォルダ >/fmi/xsl/logo.jpg

3. データベースのオブジェクトフィールドに実際のファイルではなくファイル参照が保存されている場合は、レコードを作成または編集するときに、参照されているオブジェクトを FileMaker Pro の「Web」フォルダに保存してから、Web サーバーソフトウェアのルートフォルダ内の同じ相対パスの場所にあるフォルダにコピーまたは移動する必要があります。18 ページの「Web 上でのオブジェクトフィールドの内容の公開について」を参照してください。

   注意 FileMaker データベースのオブジェクトフィールドに実際のファイルが保存されている場合は、データベースファイルが FileMaker Server 上で適切にホストされていてアクセス可能であれば、オブジェクトフィールドの内容を操作する必要はありません。

4. XSLT スタイルシートを要求および処理するには、次の URL 構文を使用します。

   < スキーム >://< ホスト >[:< ポート >]/fmi/xsl/< フォルダ >/< スタイルシート >.xsl[?< クエリー文字列 >]

   48 ページの「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。

注意 Web サイトに対しては、XSLT スタイルシートをホームページとして含めることをお勧めします。これにより、ユーザがスタイルシートにアクセスするためにクエリー文字列を入力する必要がなくなります。Site Assistant を使用すると、<?xslt-cwp-query?> 処理命令が使用されているためクエリー文字列の必要のない「home.xsl」を作成できます。たとえば、スタイルシート（「home.xsl」スタイルシートを含む）を「xslt-template-files」フォルダ内の「my_templates」フォルダにコピーした場合、Web ユーザは、次の URL を使用してスタイルシートを要求および処理することができます。

http://192.168.123.101/fmi/xsl/my_templates/home.xsl

重要 Web 公開エンジンでは、Web ユーザが「xslt-template-files」フォルダにインストールされている XSLT スタイルシートのソースを表示することは許可されません。Web ユーザがスタイルシートを処理するリクエストを送信した場合、Web 公開エンジンは、スタイルシートによる変換の結果のみを Web ブラウザまたはプログラムに送信します。

XSLT スタイルシートのトラブルシューティング

XSLT スタイルシートの使用に問題がある場合は、次の点を確認します。

- XSLT を使用したカスタム Web 公開用の拡張アクセス権がデータベースで設定されていて、ユーザアカウントに割り当てられている。17 ページの「データベースでのカスタム Web 公開の有効化」を参照してください。
- データベースが FileMaker Server によってホストされて開かれている。FileMaker Server Admin ヘルプを参照してください。
- 使用しているデータベースアカウント名とパスワードが正しい。
- Web サーバーおよび Web 公開エンジンが実行されている。
- Web 公開エンジンで [XSLT 公開:] が有効になっている。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

第 5 章
FileMaker XSLT スタイルシートの開発

この章では、FileMaker XSLT スタイルシートの構造と、FileMaker XSLT 拡張関数の使用方法を説明します。

Web 公開エンジンでの XSLT スタイルシートの使用

Web 公開エンジン経由で FileMaker XML データを要求するための XSLT スタイルシートを開発および使用する場合は、次の点に注意してください。

- Web 公開エンジンで XSLT スタイルシートを使用するには、URL で XSLT スタイルシートの名前を指定する必要があります。スタイルシートが指定されていない場合や、Web 公開エンジンがスタイルシートを見つけたり解釈できない場合は、Web 公開エンジンによってエラーページが表示されます。48 ページの「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。
- スタイルシートのファイル名、およびスタイルシートが保存されているフォルダ名は、URL エンコードされた UTF-8 にする必要があります。スタイルシートで古い Web ブラウザとの互換性を保つ必要がある場合は、名前を ASCII 文字に制限します。
- 使用する FileMaker XML 文法を、URL 内のクエリー引数、または <?xslt-cwp-query?> 処理命令で静的に定義されたクエリー引数として指定する必要があります。XML 文法が指定されていない場合、Web 公開エンジンによってエラーが表示されます。49 ページの「FileMaker XSLT スタイルシートの XML 文法の指定」を参照してください。
- 要求する FileMaker XML データを識別するクエリー引数を、URL 内、または <?xslt-cwp-query?> 処理命令で静的に定義されたクエリー引数として指定する必要があります。48 ページの「FileMaker XSLT スタイルシートの URL 構文について」および 50 ページの「静的に定義されたクエリーコマンドとクエリー引数の使用」を参照してください。
- -encoding クエリー引数を使用することで、XSLT リクエストのテキストエンコードを指定できます（オプション）。エンコードが指定されていない場合、Web 公開エンジンは、リクエストに対してデフォルトのテキストエンコード設定を使用します。51 ページの「リクエストのテキストエンコードの設定」を参照してください。
- <xsl:output> エレメントの method 属性を使用して、出力方法を指定できます（オプション）。出力方法が指定されていない場合、Web 公開エンジンは、出力として HTML を使用します。また、<xsl:output> エレメントの encoding 属性を使用して、出力ページのエンコードを指定することもできます（オプション）。エンコードが指定されていない場合、出力ページに対してデフォルトのテキストエンコード設定を使用します。52 ページの「出力方法とエンコードの指定」を参照してください。
- fmxslt:send_email() 拡張関数の関数引数を使用して、Web 公開エンジンから送信される電子メールメッセージのテキストエンコードを指定できます（オプション）。59 ページの「Web 公開エンジンからの電子メールメッセージの送信」を参照してください。

リクエストを構築するため、Web 公開エンジンは、最初に、オプションの <?xslt-cwp-query?> 処理命令で静的に定義されたクエリーコマンドとクエリー引数を使用します。静的に定義されたクエリーコマンドと引数は、基本リクエストになります。スタイルシートに <?xslt-cwp-query?> 処理命令は必須ではありませんが、ここで指定した基本リクエストは、URL クエリー文字列で指定された一致するクエリーコマンドや引数よりも優先されます。続いて、Web 公開エンジンは、<?xslt-cwp-query?> 処理命令で定義されていない URL クエリー文字列内のクエリーコマンドや追加の引数を基本リクエストに追加します。Web 公開エンジンは、このリクエストを使用して FileMaker XML データを取得し、Web ブラウザやプログラムに、指定した出力方法または HTML でデータを返します。

FileMaker XSLT 拡張関数リファレンスについて

FileMaker Server Web Publishing CD には、各 FileMaker XSLT 拡張関数の簡単な説明と例が含まれた「XSLT Reference.fp7」という FileMaker データベースが収録されています。FileMaker XSLT 拡張関数リファレンスは、FileMaker Server Web Publishing CD の「Custom Web Publishing Reference」フォルダにあります。

FileMaker XSLT スタイルシートの URL 構文について

次に、FileMaker XSLT スタイルシートを Web 公開エンジンで使用するための URL 構文を示します。

<スキーム>://<ホスト>[:<ポート>]/fmi/xsl/[<パス>/]<スタイルシート.xsl>[?<クエリー文字列>]

各要素の意味は、次のとおりです。

- <スキーム> には、HTTP または HTTPS プロトコルを指定できます。
- <ホスト> には、Web サーバーがインストールされているホストの IP アドレスまたはドメイン名を指定します。
- <ポート> には、Web サーバーが使用するポートを指定します（オプション）。ポートが指定されていない場合は、プロトコルのデフォルトのポート（HTTP ではポート 80、HTTPS ではポート 443）が使用されます。
- <パス> はオプションで、XSLT スタイルシートが保存されている「xslt-template-files」フォルダ内のフォルダを指定します。
- <スタイルシート.xsl> には、XSLT スタイルシートのファイル名を指定します。
- <クエリー文字列> には、XSLT を使用したカスタム Web 公開で使用する 1 つのクエリーコマンドと 1 つまたは複数のクエリー引数の組み合わせを指定することができます。49 ページの「FileMaker XSLT スタイルシートでのクエリー文字列の使用」および付録 A「クエリー文字列で使用される有効な名前」を参照してください。指定したスタイルシートに <?xslt-cwp-query?> 処理命令が含まれる場合は、URL クエリー文字内の一致するクエリーコマンドよりも、静的に定義されたクエリーコマンドと引数が優先されます。50 ページの「静的に定義されたクエリーコマンドとクエリー引数の使用」を参照してください。

注意　クエリーコマンドおよび引数を含め、URL 構文では、クエリー文字列の部分を除いて大文字と小文字が区別されます。URL のほとんどの部分は小文字ですが、文法名 FMPXMLRESULT および FMPXMLLAYOUT は大文字です。クエリー文字列の大文字と小文字の規則の詳細については、76 ページの「クエリーコマンドと引数の使用のガイドライン」を参照してください。

次に、FileMaker XSLT スタイルシートを Web 公開エンジンとともに使用するための URL の例を示します。

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?-grammar=fmresultset&-db=mydatabase
&-lay=mylayout&-findall

XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について

XSLT ソリューションに対して生成された XML ドキュメントでは、オブジェクトへの参照が保存されているフィールドと、実際のオブジェクトがデータベース内に保存されているオブジェクトフィールドで、オブジェクトを参照するために使用される構文が異なります。

- オブジェクトフィールドで実際のオブジェクトがデータベース内に保存されている場合、オブジェクトフィールドの <data> エレメントでは、次の URL 構文を使用してオブジェクトが参照されます。

  <data>/fmi/xsl/cnt/data.< 拡張子 >?< クエリー文字列 ></data>

  <拡張子> には、.jpg や .mov など、オブジェクトのタイプを識別するファイル拡張子を指定します。<クエリー文字列> の詳細については、前のセクション「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。

  次に例を示します。

  <data>/fmi/xsl/cnt/data.jpg?-db=products&-lay=sales&-field=product_image(1)&-recid=2</data>

  注意　オブジェクトフィールドに対して生成された XML では、-field クエリー引数の値は完全修飾フィールド名になります。括弧内の数字は、オブジェクトフィールドの繰り返し数を示し、繰り返しフィールドと非繰り返しフィールドの両方に対して生成されます。77 ページの「完全修飾フィールド名の構文について」を参照してください。

データベースからオブジェクトデータを取得するには、次の構文を使用します。

< スキーム >://< ホスト >[:< ポート >]/fmi/xsl/cnt/data.< 拡張子 >?< クエリー文字列 >

<スキーム>、<ホスト>、または <ポート> の詳細については、前のセクション「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。

次に例を示します。

http://www.company.com/fmi/xsl/cnt/data.jpg?-db=products&-lay=sales&-field=product_image(1)&-recid=2

- オブジェクトフィールドに実際のオブジェクトではなくファイル参照が保存されている場合、オブジェクトフィールドの <data> エレメントには、オブジェクトを参照する相対パスが含まれます。たとえば、「logo.jpg」が FileMaker Pro フォルダ内の「Web」フォルダにある場合、このオブジェクトフィールドの <data> エレメントは次のようになります。

  <data>/images/logo.jpg</data>

  注意　レコードを作成または編集するときに、参照されているオブジェクトを FileMaker Pro の「Web」フォルダに保存してから、Web サーバーソフトウェアのルートフォルダ内の同じ相対パスの場所にあるフォルダにコピーまたは移動する必要があります。18 ページの「Web 上でのオブジェクトフィールドの内容の公開について」を参照してください。

- オブジェクトフィールドが空の場合は、オブジェクトフィールドの <data> エレメントも空になります。

FileMaker XSLT スタイルシートでのクエリー文字列の使用

URL 内、または FileMaker XSLT スタイルシートの <?xslt-cwp-query?> 処理命令内でクエリー文字列を使用する場合、FileMaker データベースに XML データを要求するために定義されている任意のクエリーコマンドと引数を含めることができます。32 ページの「FileMaker クエリー文字列を使用した XML データの要求」を参照してください。

また、FileMaker XSLT スタイルシート専用に定義されている次のクエリーコマンドと引数を使用することもできます。

使用する XSLT クエリーコマンドまたは引数名	目的	コメント
-grammar	XSLT-CWP リクエストまたは XSLT スタイルシートの XML 文法を指定する。次のセクション「FileMaker XSLT スタイルシートの XML 文法の指定」を参照してください。	このクエリー引数は、すべての XSLT リクエストで必須です。
-encoding	リクエストのテキストエンコードを指定する。51 ページの「リクエストのテキストエンコードの設定」を参照してください。	このクエリー引数は、すべての XSLT リクエストでオプションです。
-process	データを要求せずにスタイルシートを処理する。53 ページの「FileMaker Server に対してクエリーを実行しない XSLT リクエストの処理」を参照してください。	このクエリーコマンドには、-grammar クエリー引数が必要です。
-token	セッションまたは Cookie を使用せずにページ間で値を渡す。53 ページの「トークンを使用したスタイルシート間での情報の受け渡し」を参照してください。	このクエリー引数は、すべての XSLT リクエストでオプションです。

FileMaker XSLT スタイルシートの XML 文法の指定

XSLT を使用したカスタム Web 公開で推奨される XML 文法は、fmresultset 文法です。この文法は、XSLT との連携が容易になるように設計されています。26 ページの「fmresultset 文法の使用」を参照してください。また、古い FMPXMLRESULT または FMPXMLLAYOUT 文法を使用することもできます。レイアウト内の値一覧およびフィールド表示情報にアクセスするには、FMPXMLLAYOUT 文法を使用する必要があります。29 ページの「他の FileMaker XML 文法の使用」を参照してください。XSLT を使用したカスタム Web 公開では、FMPDSORESULT 文法は使用できません。

FileMaker XSLT スタイルシートの文法を指定するには、URL、または <?xslt-cwp-query?> 処理命令で静的に定義されたクエリー引数で、-grammar クエリーコマンドを使用します。

次に例を示します。

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?-grammar=fmresultset&-db=mydatabase&-lay=mylayout&-findall

または

<?xslt-cwp-query params="-grammar=fmresultset&-db=mydatabase&-lay=mylayout&-findall"?>

重要　FileMaker XSLT スタイルシートの XML 文法が指定されていない場合、エラー「QUERY -ER0001」が表示されます。付録 B「カスタム Web 公開のエラーコード」を参照してください。

FileMaker XSLT スタイルシートのネームスペースと接頭語

XSLT タグが使用されるアプリケーションでタグを区別するには、固有の XSLT ネームスペースが役立ちます。スタイルシートで使用する FileMaker XSLT 拡張関数と特定の文法のネームスペースは、すべての FileMaker XSLT スタイルシートの先頭にある <xsl:stylesheet> エレメントで宣言します。

使用する文法	宣言するネームスペース	使用する接頭語
fmresultset XML 文法	xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"	fmrs
FMPXMLRESULT 文法	xmlns:fmp="http://www.filemaker.com/fmpxmlresult"	fmp
FMPXMLLAYOUT 文法	xmlns:fml="http://www.filemaker.com/fmpxmllayout"	fml
クエリー XML 文法	xmlns:fmq="http://www.filemaker.com/xml/query"	fmq
FileMaker XSLT 拡張関数	xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"	fmxslt

各 FileMaker XSLT スタイルシートでは、次の必須の引数も宣言する必要があります。

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

次に、ネームスペース宣言の例を示します。

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"
    xmlns:fml="http://www.filemaker.com/fmpxmllayout"
    xmlns:fmq="http://www.filemaker.com/xml/query"
    xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"
    exclude-result-prefixes="xsl fmrs fmq fml fmxslt">
```

静的に定義されたクエリーコマンドとクエリー引数の使用

FileMaker XSLT スタイルシートでは、XML データの要求時に使用するクエリーコマンドとクエリー引数を静的に定義しておくことで、クエリーコマンドとクエリー引数の不正使用を防止できます。これは必須ではありませんが、クエリーコマンドと引数をスタイルシートに静的に定義した場合、これらのクエリーコマンドと引数は、クライアントが URL クエリー文字列で指定する可能性がある、一致するクエリーコマンドまたは引数よりも優先されます。

Site Assistant および CDML Converter ツールによって生成されるスタイルシートでは、静的に定義されたクエリーコマンドと引数が使用されます。ソリューションのセキュリティを高める最も効果的な方法として、静的に定義されたクエリーコマンドと引数を使用することをお勧めします。

クエリーコマンドと引数を静的に定義するには、FileMaker XSLT スタイルシートの先頭で次の処理命令を使用します。

<?xslt-cwp-query params="クエリー文字列"?>

各要素の意味は、次のとおりです。

クエリー文字列には、名前と値の組が含まれる文字列を次の形式で指定します。

名前=値&名前２=値２....

各要素の意味は、次のとおりです。

名前には、クエリーコマンド、クエリー引数、またはデータベースフィールドの名前である文字列を指定します。

値には、任意の長さの文字列値を指定します。クエリー引数とフィールド名では、「-db=products」など、定義する特定の値を使用します。クエリーコマンドでは、-findall などのコマンド名の後に「=」記号や値を指定しないでください。付録 A「クエリー文字列で使用される有効な名前」を参照してください。

クエリー文字列で使用する文字列は、URL エンコードされている必要があります。25 ページの「URL のテキストエンコードについて」を参照してください。<xsl:output> タグの encoding 属性で指定されている文字エンコードと同じ文字エンコードを使用する必要があります。エンコードが指定されていない場合、Web 公開エンジンは、設定されているデフォルトのエンコードを使用します。

名前と値の２つの組を区切る区切り文字には、アンパサンド（&）を使用する必要があります。

たとえば、「my_stylesheet.xsl」という名前のスタイルシートで次の処理命令を使用するとします。

<?xslt-cwp-query params="-db=products&-lay=sales&-grammar=fmresultset&productname=the%20item&-find"?>

この処理命令の例は、「my_stylesheet.xsl」へのすべてのリクエストで、「products」データベースと「sales」レイアウトとともに強制的に fmresultset 文法を使用し、値が「the%20item」に設定されている「productname」フィールドに対して -find リクエストを実行するようにしています。

クライアントが、「my_stylesheet.xsl」を使用する次のリクエストを実行したとします。

http://server.company.com/fmi/xsl/my_stylesheet.xsl?-lay=revenue&city=London&-edit

この場合、次の XML リクエストが Web 公開エンジンによって処理されることになります。

http://server.company.com/fmi/xml/fmresultset.xml?-db=products&-lay=sales&productname=the%20item&city=London&-find

クライアントが入力した -lay=revenue クエリー引数と -edit クエリーコマンドよりも、静的に定義されたクエリーコマンドと引数が優先されます。「city」フィールドは処理命令で静的に定義されていないので、クライアントが入力した「city」フィールドの「London」という値は、Web 公開エンジンによって XML リクエストに含められます。

リクエストのテキストエンコードの設定

Web 公開エンジンは、XSLT リクエストのエンコードを判断できるまで、次の手順をこの順序で実行します。

1. Content-Type リクエストヘッダに charset 属性が設定されているかどうかを確認します。
2. -encoding クエリー引数にエンコードが指定されているかどうかを確認します。この引数は、URL で指定するか、または <?xslt-cwp-query?> 処理命令で静的に定義されたクエリー引数として指定できます。-encoding 引数の値は、リクエスト内の残りの引数に対して使用されるエンコードを示します。次の表は、この引数の有効な値を示します。次に例を示します。

   http://192.168.123.101/fmi/xsl/template/my_stylesheet.xsl?-db=products-lay=sales&-grammar=fmresultset&-encoding=Shift_JIS&-findall

3. Web 公開エンジンのデフォルトのテキストエンコードオプション [リクエストと出力ページ:] の現在の設定を使用します。Web 公開エンジンを初めてインストールした状態では、リクエストに対するデフォルトのテキストエンコードの初期状態の設定は、UTF-8 になっています。Web 公開エンジンのテキストエンコードの設定は、管理コンソールを使用して変更できます。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

Web 公開エンジンによってエンコードが判断された後は、そのエンコードが使用され、エンコードを判断するための以降の手順は実行されません。たとえば、Content-Type リクエストヘッダで charset 属性が設定されている場合、-encoding クエリー引数の値は使用されません。

これらのいずれかの方法で指定するテキストエンコードには、次のエンコードの 1 つを使用する必要があります。

エンコード	説明
US-ASCII	基本 ASCII 文字セット。一般的には、標準テキストの英語の電子メールに使用されます。
ISO-8859-1	Latin 1 文字セット。一般的には、上位 ASCII 文字が必要なローマ字ベースの Web ページや電子メールメッセージに使用されます。
ISO-8859-15	Latin 9 文字セット。Latin 1 文字セットとほぼ同じですが、ユーロ € 記号が追加されています。
ISO-2022-JP	ISO の日本語エンコード。一般的には、日本語の電子メールメッセージに使用されます。
Shift_JIS	日本語エンコード。一般的には、日本語の Web ページに使用されます。
UTF-8	Unicode の 8 ビットのエンコード。主要なブラウザと電子メールクライアントでサポートされるようになったため、電子メールメッセージや Web ページでの UTF-8 の使用は一般的になっています。UTF-8 では Unicode 文字の全範囲がサポートされているため、すべての言語のページを処理できます。

注意

- Web 公開エンジンを初めてインストールした状態では、出力ページに対するデフォルトのテキストエンコードの初期状態の設定は Shift_JIS になっています。次のセクション「出力方法とエンコードの指定」を参照してください。Web 公開エンジンでは、電子メールメッセージに対しては、デフォルトのテキストエンコードの初期状態の設定である ISO-2022-JP が使用されます。これらの設定は、管理コンソールを使用して変更できます。
- fmxslt:send_email(String SMTPフィールド, String 本文, String エンコード) 拡張関数を使用して、電子メールメッセージのエンコードを設定することもできます。59 ページの「Web 公開エンジンからの電子メールメッセージの送信」を参照してください。

出力方法とエンコードの指定

<xsl:output> エレメントの method および encoding 属性を使用することで、出力ページの出力方法とエンコードを指定できます。これらの属性はどちらもオプションです。

method 属性は出力のタイプを指定するもので、「html」、「text」、または「xml」を指定できます。他のタイプの方法はサポートされていません。方法が指定されていない場合は、「html」メソッドが使用されます。

encoding 属性には、出力ページのエンコードを指定します。前のセクションの表に記載されている任意のエンコードを指定できます。エンコードが指定されていない場合は、出力ページに使用するデフォルトのテキストエンコード設定が使用されます。

次に例を示します。

```
<xsl:output method="html" encoding="Shift_JIS"/>
```

スタイルシートで <xsl:output> エレメントを使用していない場合は、出力ページに使用するデフォルトのテキストエンコードの現在の設定を使用して、HTML ページが出力されます。

XSLT スタイルシートのエンコードについて

スタイルシートの先頭にある XML 宣言の encoding 属性で、リクエストと出力ページのエンコードの他に、XSLT スタイルシートのエンコードも指定する必要があります。52 ページの表に示されている任意のテキストエンコードを使用できます。

たとえば、次の宣言では、スタイルシートのエンコードとして UTF-8 を指定しています。

```
<?xml version="1.0" encoding="UTF-8"?>
```

スタイルシートのエンコードが指定されていない場合、Web 公開エンジンでは、エンコードが UTF-8 であると想定されます。

FileMaker Server に対してクエリーを実行しない XSLT リクエストの処理

スタイルシートで、レコード、フィールド名、レイアウト名など、データベースに固有の情報が必要ない場合は、-process クエリーコマンドを使用して、データベースのデータを必要としない XSLT リクエストを処理できます。このような場合、-process コマンドを使用することで、FileMaker Server の作業負荷を軽減することができます。

たとえば、-process コマンドを使用して、次のような処理を行うことができます。

- データベースの情報が必要ない場合に、静的なページを生成するスタイルシートをロードする
- スタイルシートでデータベースやレイアウトの情報（値一覧など）が必要ない場合に、新しいレコードを作成するスタイルシートをロードする
- データベースのデータを必要としない fmxslt:send_email() などの拡張関数を使用する
- データベースの情報が必要ない場合に、セッションに保存された情報にアクセスする

-process コマンドは、Web 公開エンジンの製品情報が含まれた XML ドキュメントを返します。

-process コマンドに必要な引数は -grammar のみで、fmresultset 文法または FMPXMLRESULT 文法を使用する必要があります。

次に例を示します。

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?-grammar=fmresultset&-process

トークンを使用したスタイルシート間での情報の受け渡し

-token クエリー引数を、URL 内で、または静的に定義されたクエリーコマンドとして使用すると、セッションや Cookie を使用せずに、スタイルシート間でユーザ定義の情報を渡すことができます。-token クエリー引数は、すべてのクエリーコマンドでオプションです。

ユーザ定義引数の値には、URL エンコードされた任意の文字列を使用できます。次に例を示します。

http://192.168.123.101/fmi/xsl/template/my_stylesheet.xsl?-db=products&-lay=sales&-grammar=fmresultset&-token.D100=Pending&-findall

88 ページの「-token.[文字列]（XSLT スタイルシート間での値の受け渡し）クエリー引数」を参照してください。

重要　-token クエリー引数は、個人データを渡す目的では使用しないでください。

-token クエリー引数の値を取得するには、<xsl:param name="request-query" /> ステートメントを使用します。54 ページの「リクエストのクエリー情報へのアクセス」を参照してください。

FileMaker XSLT 拡張関数と引数の使用

FileMaker XSLT 拡張関数は、fmxslt ネームスペースに存在するように定義されています。必ず、XSLT スタイルシートの先頭にある <xsl:stylesheet> エレメントに、fmxslt ネームスペースの宣言を含めてください。50 ページの「FileMaker XSLT スタイルシートのネームスペースと接頭語」を参照してください。

FileMaker XSLT 拡張関数は、XPath ステートメント内に関数呼び出しとして指定することによって、XSLT スタイルシートで使用できるように設計されています。XPath ステートメントは、多くの XSLT エレメントで select 属性および test 属性の値として使用されます。

たとえば、User-Agent ヘッダを確認して、使用されているブラウザを判断するとします。このためには、User-Agent ヘッダの値を格納する次のような変数を使用できます。

<xsl:variable name="user-agent" select="fmxslt:get_header('User-Agent')"/>

値を返す拡張関数では、値は、指定された XSLT タイプで返されます。多くの関数は文字列を返しますが、トラバースできるノードセットを返す関数もあります。

注意　このセクションでは、FileMaker XSLT 拡張関数と引数について説明し、いくつかの例を示しています。各関数の他の例については、FileMaker XSLT 拡張関数リファレンスを参照してください。47 ページの「FileMaker XSLT 拡張関数リファレンスについて」を参照してください。

Web 公開エンジンによって設定される FileMaker に固有の XSLT 引数について

リクエストを処理する際に、Web 公開エンジンによって、FileMaker に固有の次の XSLT 引数の値が動的に設定されます。<xsl:param> エレメントを使用して、スタイルシートでこれらの引数の値を使用することができます。

FileMaker に固有の XSLT 引数	詳細の参照先
<xsl:param name="request-query"/>	次のセクションの「リクエストのクエリー情報へのアクセス」
<xsl:param name="client-ip"/> <xsl:param name="client-user-name"/> <xsl:param name="client-password"/>	55 ページの「クライアント情報の取得」
<xsl:param name="xml-base-uri"/>	55 ページの「Web 公開エンジンのベース URI 引数の使用」
<xsl:param name="authenticated-xml-base-uri">	55 ページの「認証済みベース URI 引数の使用」

リクエストのクエリー情報へのアクセス

FileMaker XSLT 引数を使用して、URL または HTML フォームデータのリクエストに含まれるクエリー情報にアクセスできます。たとえば、現在のリクエストのクエリー情報にアクセスして、対象レコードの現在の場所を判断したり、前後のレコードへのリンクを作成することができます。

Web 公開エンジン経由で FileMaker XML データを要求するために使用されるすべてのクエリーコマンドとクエリー引数へのアクセスは、次の FileMaker XSLT 引数によって提供されます。

<xsl:param name="request-query"/>

フィールド名を除き、Web 公開エンジンでは、すべてのクエリーコマンドとクエリー引数の名前は小文字で返されます。フィールド名が大文字の場合は、大文字のまま返されます。

request-query 引数には、XML ドキュメントの一部が次の文法でロードされます。

```
<!DOCTYPE query [
    <!ELEMENT query (parameter)*>
        <!ATTLIST query action CDATA #REQUIRED>
        <!ELEMENT parameter (#PCDATA)>
        <!ATTLIST parameter name CDATA #REQUIRED>
        ]
```

注意　クエリー情報は、fmq="http://www.filemaker.com/xml/query" ネームスペースに存在するように定義されます。必ず、XSLT スタイルシートの先頭にある <xsl:stylesheet> エレメントに、fmq ネームスペースの宣言を含めてください。50 ページの「FileMaker XSLT スタイルシートのネームスペースと接頭語」を参照してください。

たとえば、次のリクエストのクエリーコマンドとクエリー引数にアクセスするとします。

http://192.168.123.101/fmi/xsl/my_stylesheet.xsl?-db=products&-lay=sales&-grammar=fmresultset&-token.1=abc123&-findall

<xsl:param name="request-query" /> ステートメントをテンプレートセクションの前に含めた場合、Web 公開エンジンによって、次の XML ドキュメントが引数に格納されます。

```
<query action="my_stylesheet.xsl" xmlns="http://www.filemaker.com/xml/query">
    <parameter name="-db">products</parameter>
    <parameter name="-db">sales</parameter>
    <parameter name="-grammar">fmresultset</parameter>
    <parameter name="-token.1">abc123</parameter>
    <parameter name="-findall"></parameter>
</query>
```

その後、XPath 式を使用することによって、request-query 引数を使用して、URL で渡されたトークンの値にアクセスできます。次に例を示します。

```
$request-query/fmq:query/fmq:parameter[@name = '-token.1']
```

クライアント情報の取得

次の FileMaker XSLT 引数を使用して、Web 公開エンジンから Web クライアントの IP アドレス、ユーザ名、およびパスワードを取得できます。

```
<xsl:param name="client-ip"/>

<xsl:param name="client-user-name"/>

<xsl:param name="client-password">
```

これらの引数ステートメントは、XSLT スタイルシートで最も上にある <xsl:template> エレメントの前に含めます。

スタイルシートがパスワードで保護された追加の XML ドキュメントをプログラムによってロードするとき、これらの引数は Web ユーザの資格情報を指定します。56 ページの「追加のドキュメントのロード」を参照してください。Web ユーザは、最初に HTTP 基本認証のダイアログボックスでユーザ名とパスワードを入力する必要があります。17 ページの「Web ユーザがカスタム Web 公開を使用して保護されたデータベースにアクセスする場合」を参照してください。

これら 3 つの FileMaker XSLT 引数の使用に関する詳細と他の例については、FileMaker XSLT 拡張関数リファレンスを参照してください。

Web 公開エンジンのベース URI 引数の使用

Web 公開エンジンでは、ベース URI（Uniform Resource Identifier）引数は、Web 公開エンジンがインストールされているホストとポートになるように定義されています。ベース URI により、FileMaker データベースの XML データへのリクエストを Web 公開エンジンホストに対して相対的に解決できます。

Web 公開エンジンのベース URI にアクセスするには、XSLT スタイルシートで最も上にある <xsl:template> エレメントの前に、次のステートメントを含めます。

```
<xsl:param name="xml-base-uri"/>
```

その後、FileMaker XML データへの追加のリクエストを実行する必要がある場合は、$xml-base-uri 変数を使用して、現在のスタイルシートのベース URI を使用できます。たとえば、次の追加の XML データに対する要求で、ベース URI を使用することができます。

```
<xsl:variable name="layout_information" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?
-db=products&-lay=sales&-view'))" />
```

認証済みベース URI 引数の使用

authenticated-xml-base-uri 引数では、client-user-name および client-password 引数の機能と xml-base-uri 引数が組み合わされます。

```
<xsl:param name="authenticated-xml-base-uri"/>
```

この引数は、現在処理中の元のリクエストで指定されているものと同じユーザ名とパスワードが必要な、パスワードで保護された追加の XML ドキュメントをロードする場合に使用します。例については、次のセクション「追加のドキュメントのロード」を参照してください。

この引数ステートメントは、XSLT スタイルシートで最も上にある <xsl:template> エレメントの前に含めます。

client-user-name および client-password 引数の値が空白でない場合、authenticated-xml-base-uri 引数の値は次のようになります。

```
http://ユーザ名:パスワード@ホスト名:ポート
```

client-user-name および client-password 引数の値が空白の場合、authenticated-xml-base-uri 引数の値は、xml-base-uri 引数の値と同じになります。

追加のドキュメントのロード

XSLT スタイルシートの処理中に追加の XML ドキュメントをロードするには、XML ドキュメントへの URI とともに、XSLT の標準の document() 関数を使用します。document() 関数は、要求された XML を、<xsl:variable> エレメントに格納できるノードセットとして返します。

FileMaker データベースのデータが含まれる XML ドキュメントをロードするには、document() 関数を FileMaker クエリーコマンドおよび引数とともに使用します。次に例を示します。

```
<xsl:variable name="other-data" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?
-db=products&-lay=sales&-view'))" />
```

現在処理中の元のリクエストで指定されているものと同じユーザ名とパスワードが必要な、パスワードで保護された追加の XML ドキュメントをロードするには、authenticated-xml-base-uri 引数を使用します。この引数を使用することで、document() 関数に渡される URI の一部として同じユーザ名とパスワードが指定されます。

次に例を示します。

```
<xsl:variable name="other-data" select="document(concat($authenticated-xml-base-uri,
'/fmi/xml/FMPXMLLAYOUT.xml?-db=products&-lay=sales&-view'))"/>
```

親リクエストで指定されているものとは異なるユーザ名とパスワードが必要な、パスワードで保護された XML ドキュメントをロードするには、次の構文を使用して、document() 関数に渡される URI の一部としてユーザ名とパスワードを指定します。

http://ユーザ名:パスワード@ホスト名/パス?クエリー文字列

FileMaker データベースに基づかない XML ドキュメントをロードするには、FileMaker クエリーコマンドまたは引数を設定せずに document() 関数を使用します。次に例を示します。

```
<xsl:variable name="other-data" select="document('http://server.company.com/data.xml')" />
```

document() 関数を相対 URL とともに使用した場合、Web 公開エンジンは、スタイルシートが保存されている場所に対して相対的な場所にある XML ドキュメントをローカルファイルシステムからロードしようとします。たとえば、「xslt-template-files」フォルダの「mystylesheets」フォルダ内にあるスタイルシートに、相対 URL を使用した次の document() 関数が含まれているとします。

```
<xsl:variable name="mydoc" select="document('mystylesheets/mydoc.xml')" />
```

この場合、Web 公開エンジンは、ローカルファイルシステムの「xslt-template-files」フォルダ内にある「mystylesheets」フォルダから「mydoc.xml」をロードしようとします。

注意 Web 公開エンジンでは、Web 公開エンジンのベース URI を使用してドキュメントをロードする場合は、HTTP のみがサポートされています。外部サーバーからドキュメントをロードする場合は、HTTP と HTTPS の両方がサポートされています。

スタイルシートでのデータベースのレイアウト情報の使用

FMPXMLLAYOUT 文法を使用してレイアウト情報を要求し、XSLT の document() 関数を使用して変数にロードすることで、FileMaker データベースのレイアウト情報をスタイルシートに組み込むことができます。

```
<xsl:variable name="layout" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?-view'))" />
```

たとえば、「Color」という名前のフィールドに対してプルダウンメニューを作成するとします。このフィールドには、FileMaker データベースのレイアウトに定義されている「shirts」という名前の値一覧の値が入力されています。次に、document() 関数を使用して、このレイアウト情報を XSLT の変数にロードする方法を示します。

```
<xsl:variable name="layout" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?-db=products
&-lay=sales&-view'))" />

<select size="1">

    <xsl:attribute name="name">Color</xsl:attribute>

    <option value="">Select One...</option>
```

```
<xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = 'shirts']/fml:VALUE">
        <option>
            <xsl:attribute name="value"><xsl:value-of select="."/></xsl:attribute>
            <xsl:value-of select="."/>
        </option>
    </xsl:for-each>
</select>
```

コンテンツバッファの使用

コンテンツバッファが無効の場合、Web 公開エンジンは、XSLT 変換の結果を直接クライアントに戻します。明示的に有効にしない限り、コンテンツバッファは常に無効です。コンテンツバッファを有効にすると、変換されたコンテンツは、変換全体が完了するまで、Web 公開エンジンによって保存されています。

ヘッダを操作する XSLT スタイルシートでは、コンテンツバッファが必要です。ヘッダは応答の本文より先に書き込まれるため、追加されたヘッダ情報を含めることができるように、本文をバッファする必要があります。

XSLT 変換の結果をバッファする必要がある FileMaker 拡張関数は、次の 4 つです。

- fmxslt:create_session(): 58 ページの「セッション拡張関数の使用」を参照してください。
- fmxslt:set_header(): 60 ページの「ヘッダ関数の使用」を参照してください。
- fmxslt:set_status_code (): 60 ページの「ヘッダ関数の使用」を参照してください。
- fmxslt:set_cookie(): 61 ページの「Cookie 拡張関数の使用」を参照してください。

これらの FileMaker 拡張関数を正常に機能させるには、リクエストに対する最上位のドキュメントに次の XSLT 処理命令を含める必要があります。

```
<?xslt-cwp-buffer buffer-content="true"?>
```

重要　他のスタイルシートが含まれるベーススタイルシートを使用している場合、<?xslt-cwp-buffer?> 処理命令は、ベーススタイルシートに含める必要があります。ベーススタイルシートに含まれるスタイルシートでこの命令を使用しても、無視されます。

この処理命令を使用して応答をバッファすると、Web 公開エンジンが応答の長さを判断して、応答に Content-Length ヘッダを設定できるという利点があります。応答をバッファすると、Web 公開エンジンのパフォーマンスが低下する場合があります。

Web 公開エンジンセッションを使用したリクエスト間での情報の保存

Web 公開エンジンのサーバーサイドセッションを使用して、リクエスト間で任意のタイプの情報を追跡および保存できます。セッションを使用すると、持続的な任意の情報をリクエスト間で使用することによって状態を維持できる Web アプリケーションを作成できます。たとえば、最初のフォームページで入力されたユーザクライアント情報をセッションに保存しておき、以降のページで値を入力するために使用できます。

デフォルトでは、Web 公開エンジンは、Cookie を使用してセッション ID を保存します。Cookie が許可されていないクライアントに対応するには、fmxslt:session_encode_url() 関数を使用して、セッション ID を URL に追加できます。すべての状況において互換性を保証するために、ページに書き込む URL は、fmxslt:session_encode_url() 関数を使用してすべてエンコードすることをお勧めします。この関数では、jsessionid というセミコロン区切りの引数が URL に追加されます。この引数は、該当するクライアントの親セッションの ID です。

たとえば、次のリンクがページに配置されているとします。

```
<a href="my_stylesheet.xsl?-db=products&-lay=sales&-grammar=fmresultset&-findall">hyperlinked text</a>
```

この場合、ページ上のすべてのリンクを次のようにエンコードすることをお勧めします。

```
<a href="{fmxslt:session_encode_url('my_stylesheet.xsl?-db=products&-lay=sales&-grammar=fmresultset
&-findall')}">hyperlinked text</a>
```

クライアントで Cookie が許可されていない場合は、次のリンクがページに含められます。

```
<a href="my_stylesheet.xsl;jsessionid=<session id>?-db=products&-lay=sales&-grammar=fmresultset&-findall">
hyperlinked text</a>
```

クライアントで Cookie が許可されていることが Web 公開エンジンによって検出された場合、fmxslt:session_encode_url() 関数は、URL ではなく Cookie にセッション ID を保存します。

注意　セッション情報は、Web 公開エンジンの再起動後には維持されません。

セッション拡張関数の使用

セッション変数を操作するには、次のセッション関数を使用します。セッションオブジェクトには、文字列、数字、論理値、またはノードセットを保存できます。ノードセットを使用すると、XML のデータ構造体を作成して、リクエスト間でセッションオブジェクトに保存できます。

セッション拡張関数	戻り値のデータタイプ	説明
fmxslt:session_exists(String セッション名)	論理値	指定した名前のセッションが存在するかどうかを確認します。
fmxslt:create_session(String セッション名)	論理値	指定したセッション名とデフォルトのタイムアウトでセッションを作成します。デフォルトのタイムアウトは、管理コンソールを使用して設定します。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。 注意　この関数には、<?xslt-cwp-buffer?> 処理命令が必要です。57 ページの「コンテンツバッファの使用」を参照してください。
fmxslt:invalidate_session(String セッション名)	論理値	セッションをただちに強制的にタイムアウトさせます。
fmxslt:set_session_timeout(String セッション名, Number タイムアウト)	論理値	セッションのタイムアウトを設定します（分単位）。 セッションのデフォルトのタイムアウトは、管理コンソールを使用して設定します。
fmxslt:session_encode_url(String URL)	文字列	クライアントで Cookie がサポートされていない場合は、セッション ID を含めて URL をエンコードします。その他の場合は、入力 URL を返します。
fmxslt:set_session_object(String セッション名, String 名前, Object 値)	XSLT オブジェクト（数字、文字列、論理値、またはノードセット）	セッション中に XSLT オブジェクト（数字、文字列、論理値、またはノードセット）を保存しておき、後で fmxslt:get_session_object () 関数を使用して取得できます。 この関数は、指定したセッションオブジェクト名ですでに保存されているオブジェクトも返します。指定した名前で保存されているオブジェクトがない場合は、ヌルオブジェクトを返します。
fmxslt:get_session_object(String セッション名, String 名前)	XSLT オブジェクト	セッションから XSLT オブジェクトを取得します。
fmxslt:remove_session_object(String セッション名, String 名前)	XSLT オブジェクト	セッションから XSLT オブジェクトを取得して削除します。

次に、セッションを作成して、お気に入りの色をセッションに保存する例を示します。

```
<xsl:variable name="session">
    <xsl:choose>
        <xsl:when test="not (fmxslt:session_exists(string($session-name)))">
            <xsl:value-of select="fmxslt:create_session(string($session-name))"/>
        </xsl:when>
        <xsl:otherwise>true</xsl:otherwise>
    </xsl:choose>
```

```
</xsl:variable>
<xsl:variable name="favorite-color" select="fmxslt:set_session_object(string($session-name), 'favorite-color', string($color))"/>
```

重要

- セッションの完了後にユーザをデータベースから確実にログアウトするには、fmxslt:invalidate_session () 関数を使用して、セッションをただちに強制的にタイムアウトさせます。
- 状態を設定または変更するグローバルフィールドやスクリプトを使用する場合は、管理コンソールを使用して、Web 公開エンジンの [XSLT データベースセッション:] オプションを有効にする必要があります。このオプションが有効でない場合、グローバルフィールドの値や状態は、リクエスト間で維持されません。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。
- Web 公開エンジンセッションの使用中に他のデータベースファイルに切り替えた場合、グローバルフィールドの値は保存されません。Web 公開エンジンは、新しい第２のファイルを開く前に、それまで開いていた第１のファイルを閉じてしまうからです。代替手段として、第１のデータベースファイル内のレイアウトを使用して、第２のデータベースファイルからデータにアクセスすることもできます。

Web 公開エンジンからの電子メールメッセージの送信

Web 公開エンジンを使用して電子メールメッセージを生成することができます。これは、カスタム Web ソリューションで便利です。Web 公開エンジンから電子メールメッセージを送信するには、XSLT スタイルシートで、次の３つの fmxslt:send_email () 拡張関数の１つを使用します。これらの関数を使用して、１つまたは複数の別個のメッセージを送信できます。fmxslt:send_email () 関数は、Web 公開エンジンのサーバーサイドの XSLT スタイルシートに含まれるため、クライアントが Web 公開エンジンを使用して不正な電子メールメッセージを送信することはできません。

電子メール拡張関数	戻り値のデータタイプ	説明
fmxslt:send_email(String SMTPフィールド, String 本文)	論理値	電子メールメッセージに使用する Web 公開エンジンのデフォルトのテキストエンコードを使用して、Web 公開エンジンから任意の長さのテキストの電子メールメッセージを送信します。
fmxslt:send_email(String SMTPフィールド, String 本文, String エンコード)	論理値	任意の長さのテキストの電子メールメッセージを、US-ASCII、ISO-8859-1、ISO-8859-15、ISO-2022-JP、Shift_JIS、UTF-8 のいずれかのテキストエンコードを使用して送信します。これらのエンコードの詳細については、51 ページの「リクエストのテキストエンコードの設定」を参照してください。
fmxslt:send_email(String SMTPフィールド, String XSLT ファイル, Node XML, boolean イメージの包含)	論理値	スタイルシートの <xsl:output> エレメントの encoding 属性によって指定されているエンコードを使用して、HTML ベースの電子メールメッセージを送信します。<xsl:output> エレメントに encoding 属性が含まれない場合は、電子メールメッセージに使用する Web 公開エンジンのデフォルトのテキストエンコードが使用されます。

注意

- これら３つの形式の各 fmxslt:send_email () 関数では、SMTPフィールド引数には、次の形式を使用するアドレスとトピックの情報が含まれる、URL エンコードされた任意の長さの文字列を指定します。この形式は、RFC 2368 の mailto URL スキームに基づきます。

  ユーザ名 @ ホスト ? 名前１ = 値１ & 値２ = 値２ ...

  ユーザ名@ホストには、受信者を指定します。名前/値の組を任意の順序で指定でき、次のように定義されます。

  - from=ユーザ名@ホスト（記述するのは１回だけにする必要があります）。from フィールドは必ず指定する必要があります。
  - to=ユーザ名@ホスト。追加の受信者には、この名前/値の組を使用します。
  - reply-to=ユーザ名@ホスト（１回だけ記述できます）。
  - cc=ユーザ名@ホスト。
  - bcc=ユーザ名@ホスト。

- subject=文字列（1 回だけ記述できます）。

  複数の from、reply-to、または subject フィールドが指定されている場合、電子メールメッセージは送信されず、関数によって値 false() が返され、該当するエラーコードが設定されます。

- 入力されたすべての電子メールアドレスの構文は、Web 公開エンジンによって確認されます。構文は次の形式でなければなりません。

  ユーザ @ ホスト .tld または " ダブルクォーテーションで囲まれた ID"< ユーザ @ ホスト .tld>

  tld には、com や net などの最上位のドメインを指定します。いずれかのフィールドに無効な電子メールアドレスが含まれる場合、電子メールメッセージは送信されず、該当するエラーステータスコードが設定されます。

- subject などの SMTPフィールド引数の個々の値は、URL エンコードされた文字列である必要があります。たとえば、「&」文字は「&」として、スペースは「%20」として指定する必要があります。SMTPフィールド引数の文字列全体が XML エンコードされている必要があります（このセクションの最後にある例を参照してください）。
- これらの各関数に対して、電子メールメッセージが正常に送信された場合は true() の値が返され、その他の場合は false() が返されます。
- 日本語の電子メールメッセージには、Web 公開エンジンによって、デフォルトのテキストエンコードの初期状態の設定である Shift_JIS が使用されます。この設定は、管理コンソールを使用して変更できます。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。
- fmxslt:send_email(String SMTPフィールド, String XSLTファイル, Node XML, boolean イメージの包含) 関数は、この関数で指定した電子メール用スタイルシートによって処理された XML データで構成される電子メールメッセージを送信します。
  - XSLTファイル引数には、リクエストに使用する主な処理用スタイルシートファイルに対して相対的な URL を入力して電子メール用スタイルシートの名前を指定します。
  - XML 引数には、電子メール用スタイルシートとともに使用する XML データの親ノードを指定します。ブラウザに表示されている XML データと同じ XML データを使用して電子メールメッセージを送信するには、単にドキュメントのルートの XPath「/」を指定します。その他の場合は、最初に document() 関数でドキュメントをロードしてから、そのドキュメントを fmxslt:send_email() 関数に渡すことで、異なる XML ドキュメントを使用できます。
  - イメージの包含引数には、電子メールメッセージの HTML で指定されているすべてのイメージを Web 公開エンジンで添付ファイルとして含める場合は、論理値 true() を指定します。この引数では、FileMaker データベース内にあるイメージと、データベース内にない他の場所のイメージの両方が含められます。イメージの URL は、Web 公開エンジンによって、添付ファイルを参照するように変更されます。イメージファイルが多い場合や、大きい場合は、パフォーマンスが低下する可能性があります。false () を指定した場合、イメージの URL は Web 公開エンジンによって変更されません。絶対 URL の場合、電子メールクライアントは、Web サーバーからイメージをロードしようとします。

次に、<xsl:if> エレメントの内部など、XPath ステートメント内で、fmxslt:send_email(String SMTPフィールド, String XSLTファイル, Node XML, boolean イメージの包含) 関数を使用する場合の例を示します。

```
fmxslt:send_email('tom_jones@company.com?subject=project%20status&amp;from=john_smith@company.com
&amp;cc=jane_doe@company.com','my_mail_template.xsl', /, true())
```

SMTP サーバーに接続するように Web 公開エンジンを設定する場合の詳細については、『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

ヘッダ関数の使用

fmxslt:get_header() 関数を使用して、HTTP リクエストおよび応答のヘッダから情報を読み取り、fmxslt:set_header() 関数を使用して、これらのヘッダに情報を書き込むことができます。これらの関数は、クライアントがヘッダ情報を使用して Web サーバーから情報を取得できる場合や、他の理由で HTTP ヘッダを設定する必要がある場合に便利です。

ヘッダ拡張関数	戻り値のデータタイプ	説明
fmxslt:get_header(String 名前)	文字列	指定したヘッダの値を返します。
fmxslt:set_header(String 名前, String 値)	void	指定したヘッダの値を設定します。
fmxslt:set_status_code(Number ステータスコード)	void	HTTP ステータスコードを設定します。

注意

- fmxslt:get_header() と fmxslt:set_header() 関数で使用する名前、および fmxslt:set_header() 関数の値には、任意の長さの文字列を指定できます。
- fmxslt:set_header() 関数および fmxslt:set_status_code() 関数には、<?xslt-cwp-buffer?> 処理命令が必要です。57 ページの「コンテンツバッファの使用」を参照してください。

次の例に、ヘッダの値を設定する方法を示します。ここでは、スタイルシートを使用して vCard を出力するとします。この場合、ブラウザがスタイルシートのページをロードするときに、.xsl ファイルがブラウザによって vCard ではなくスタイルシートとして解釈されるという問題が発生する可能性があります。Content-Disposition というヘッダを使用すると、拡張子 .vcf の付いた添付ファイルが存在することを指定できます。

<xsl:value-of select="fmxslt:set_header('Content-Disposition','attachment;filename=test.vcf')" />

Cookie 拡張関数の使用

Cookie 拡張関数を使用して、クライアントの Web ブラウザに保存される Cookie を取得または設定できます。

Cookie 拡張関数	戻り値のデータタイプ	説明
fmxslt:get_cookie(String 名前)	ノードセット	指定した Cookie 名を持つ COOKIES ノードリストを返します。
fmxslt:get_cookies()	ノードセット	クライアントによって提供された Cookie がすべて含まれる COOKIES ノードリストを返します。
fmxslt:set_cookie(String 名前, String 値)	void	指定した Cookie を、指定した値でクライアントのブラウザに保存します。
fmxslt:set_cookie(String 名前, String 値, Number 有効期限, String パス, String ドメイン)	void	指定した Cookie を、Cookie で使用可能なすべての値でクライアントのブラウザに保存します。有効期限引数には、Cookie の有効期限が切れるまでの秒数を指定します。

注意

- fmxslt:get_cookie() および fmxslt:get_cookies() 関数は、次の構造のノードセットを返します。

```
<!ELEMENT cookies (cookie)*>
    <!ATTLIST cookie xmlns CDATA #FIXED "http://www.filemaker.com/xml/cookie">
<!ELEMENT cookie (#PCDATA)>
    <!ATTLIST cookie name CDATA #REQUIRED>
```

- cookies ノードセットの XML ネームスペースは "http://www.filemaker.com/xml/cookie" です。必ずこのネームスペースを宣言し、ネームスペースには接頭語を指定してください。
- fmxslt:set_cookie 関数のすべての引数値が有効である必要があります。有効でない場合、Web ブラウザでは、fmxslt:set_cookie 関数のリクエストは無視されます。
- すべての Cookie 関数では、文字列引数は任意の長さにできます。
- fmxslt:set_cookie() 関数の両方の形式には、<?xslt-cwp-buffer?> 処理命令が必要です。57 ページの「コンテンツバッファの使用」を参照してください。

例: get_cookie

次の例は、preferences という名前の cookie とその値を検索します。

```
<xsl:variable name="pref_cookie" select="fmxslt:get_cookie('preferences')"/>
<xsl:value-of select="concat('Cookie Name = ', $pref_cookie/fmc:cookies/fmc:cookie/@name)"/> <br/>
<xsl:value-of select="concat('Cookie Name = ', $pref_cookie/fmc:cookies/fmc:cookie)"/>
```

例: set_cookie

次に、すべての値を使用して Cookie を設定する方法の例を示します。

```
<xsl:variable name="storing_cookie" select="fmxslt:set_cookie ('text1', 'text2', 1800, 'my_text', 'my.company.com')" />
```

文字列操作拡張関数の使用

文字列操作関数を使用して、任意の長さの文字列のエンコードを変更できます。

文字列操作 拡張関数	戻り値の データタイプ	説明
fmxslt:break_encode(String 値)	文字列	改行などが HTML エンコードされた文字列を返します。「&」（アンパサンド）などの文字は「&」などに、行送りや改行などの復帰改行文字は にそれぞれ置き換えられます。この関数は、<xsl:value-of> および <xsl:text> エレメントの disable-output-escaping 属性が「yes」に設定されている（disable-output-escaping="yes"）場合にのみ動作します。 注意　fmxslt:break_encode() 関数が適用される文字列に行送りまたは改行を含めるには、文字列内でエスケープ文字「
」（行送り）または「
」（改行）を使用する必要があります。テキストエディタで Enter キー（Window）または return キー（Mac OS）を押して文字列に行送りまたは改行を含めることはできません。
fmxslt:html_encode(String 値)	文字列	HTML エンコードされた文字列を返します。「&」（アンパサンド）などの文字は、「&」などに置き換えられます。
fmxslt:url_encode(String 値)	文字列	URL エンコードされた文字列を返します。URL エンコードは、インターネット上、特に URL で文字を転送するために使用されます。たとえば、URL エンコードされた形式の「&」（アンパサンド）は、%26 になります。予約済みの文字が href で使用されている場合は、この関数を使用して、文字列を URL エンコードします。
fmxslt:url_encode(String 値, String エンコード)	文字列	encoding 引数に指定した文字エンコードを使用して、URL エンコードされた文字列を返します。この引数には、US-ASCII、ISO-8859-1、ISO-8859-15、ISO-2022-JP、Shift_JIS、または UTF-8 を指定できます。 この関数は、Web サーバーで、現在のスタイルシートで使用されている文字コードと異なる文字エンコードが使用されることがわかっている場合に使用します。たとえば、Web サイトの最初のページは UTF-8 で表示されていても、ユーザがリンクをクリックして日本語のページに移動する場合があります。リクエストに日本語の文字が含まれていて、日本語のページで Shift_JIS エンコードが使用されている場合は、文字列を Shift_JIS でエンコードすることをお勧めします。
fmxslt:url_decode(String 値)	文字列	すでにエンコードされている URL 文字列から URL デコードされた文字列を返します。
fmxslt:url_decode(String 値, Stringエンコード)	文字列	encoding 引数に指定した文字エンコードを使用して、URL デコードされた文字列を返します。この引数には、US-ASCII、ISO-8859-1、ISO-8859-15、ISO-2022-JP、Shift_JIS、または UTF-8 を指定できます。 この関数は、URL エンコードされた文字列を正しくデコードするために、文字列で使用されている文字エンコードを指定する必要がある場合に使用します。たとえば、Web サイトで ISO-8859-1 が使用されていても、ユーザは、異なる文字エンコードを使用してフォームを送信できます。

Perl 5 の正規表現を使用した文字列の比較

fmxslt:regex_contains() 拡張関数を使用すると、Perl 5 の正規表現を使って文字列を比較できます。正規表現による比較は高度な種類のテキスト照合で、文字列が指定したパターンに一致するかどうかを判断できます。この関数の構文は次のとおりです。

fmxslt:regex_contains(String 入力, String パターン)

入力には文字列を、パターンには Perl 5 の正規表現をそれぞれ指定します。Perl 5 の正規表現の構文の詳細については、www.perldoc.com を参照してください。fmxslt:regex_contains() 関数は、論理値を返します。

この関数は、標準の XSLT によって提供されている機能よりも高度な文字列操作が必要な場合に便利です。たとえば、Perl 5 の正規表現で文字列を比較することで、フィールドの値に有効な電話番号や電子メールアドレスが含まれているかどうかを判断できます。

次に、この関数を使用して、フィールドの値に正しい形式の電子メールアドレスが含まれているかどうかを判断する場合の例を示します。

```
<xsl:variable name="email" select="'foo@bar.com'"/>
<xsl:if test="fmxslt:regex_contains($email,'^\w+[\w-\.]*\@\w+((-\w+)|(\w*))\.[a-z]{2,3}$')">Valid Email</xsl:if>
```

Web 公開エンジンがこのパターンを解析できない場合は、エラーステータスがエラーコード 10311 に設定されます。96 ページの「FileMaker XSLT 拡張関数のエラーコード番号」を参照してください。

チェックボックスとして書式設定されたフィールドの値の確認

次の拡張関数を使用して、チェックボックス形式の値一覧の特定の値が FileMaker データベースのフィールドに保存されているかどうかを判断します。

fmxslt:contains_checkbox_value(String 値の文字列, String 値一覧エントリ)

値の文字列にはフィールドを指定する XPath を、値一覧エントリには確認する値をそれぞれ指定します。

指定した値がフィールドに保存されている場合、この論理関数は true() を返します。その他の場合は、false() を返します。この関数を使用して、HTML フォームの checked 属性を設定し、チェックボックスを選択された状態で表示するかどうかを決定できます。

たとえば、FileMaker データベースのレイアウトのフィールドに、次のチェックボックスオプションが設定されているとします。

[ ] Red

[ ] Blue

[ ] Green

[ ] Small

[ ] Medium

[ ] Large

ユーザが [Red] のみを選択している場合、フィールドには文字列「Red」が含まれています。フィールドに「Blue」が含まれるかどうかを判断するために、次の関数呼び出しを使用できます。

fmxslt:contains_checkbox_value(<フィールド値のノード>,'Blue')

<フィールド値のノード> には、チェックボックスフィールドの <data> エレメントへの XPath を指定します。この例では、この関数は「false」を返します。

この関数の一般的な応用は、Web ページにチェックボックスの値一覧を表示して、データベースで選択されているチェックボックスを Web ページでも選択する場合です。たとえば、次の HTML および XSLT ステートメントを使用して、「color_size」という名前の値一覧を使用する「style」というフィールドに対して、チェックボックスのセットを作成します。

```
<xsl:variable name="field-value" select="fmrs:field[@name='style']/fmrs:data" />
<xsl:for-each select="$valuelists[@NAME = 'color_size']/fml:VALUE">
    <input type="checkbox">
        <xsl:attribute name="name">style</xsl:attribute>
        <xsl:attribute name="value"><xsl:value-of select="."/></xsl:attribute>
        <xsl:if test="fmxslt:contains_checkbox_value($field-value,.)">
            <xsl:attribute name="checked">checked</xsl:attribute>
        </xsl:if>
    </input><xsl:value-of select="."/><br/>
</xsl:for-each>
```

この例の HTML および XSLT ステートメントでは、次のチェックボックスが、[Red] および [Medium] が選択された状態で Web ページ上に出力されます。

[x] Red

[ ] Blue

[ ] Green

[ ] Small

[x] Medium

[ ] Large

日付、時刻、および曜日拡張関数の使用

現在の日付、時刻、または曜日を取得したり、２つの日付、時刻、または曜日を比較する拡張関数を使用することができます。

次の表に示す関数は、ロケールにかかわらず、「fm」書式を使います。「fm」書式とは、日付を MM/dd/yyyy、時刻を HH:mm:ss、タイムスタンプを MM/dd/yyyy HH:mm:ss で表す書式です。

出力値を他のもっと好ましい書式に変更したい場合は、計算関数または JavaScript を使用してください。

日付、時刻、および曜日拡張関数	戻り値のデータタイプ	説明
fmxslt:get_date()	文字列	現在の日付を「fm」書式で返します。
fmxslt:get_date(String 書式)	文字列	現在の日付を指定した書式で返します。書式引数には“short”、“long”、“fm”のいずれかを入力します。
fmxslt:get_time()	文字列	現在の時刻を「fm」書式で返します。
fmxslt:get_time(String 書式)	文字列	現在の時刻を指定した書式で返します。書式引数には“short”、“long”、“fm”のいずれかを入力します。
fmxslt:get_day()	文字列	現在の曜日を短い書式で返します。
fmxslt:get_day(String 書式)	文字列	現在の曜日を指定した書式で返します。書式引数には、“short”または“long”を入力します。
fmxslt:get_fm_date_format()	文字列	「fm」日付書式の形式文字列 「MM/dd/yyyy」を返します。
fmxslt:get_short_date_format()	文字列	短い日付書式の形式文字列 「yy/MM/dd」を返します。
fmxslt:get_long_date_format()	文字列	長い日付書式の形式文字列「yyyy/MM/dd」を返します。
fmxslt:get_fm_time_format()	文字列	「fm」時刻書式の形式文字列「HH:mm:ss」を返します。

日付、時刻、および曜日拡張関数	戻り値のデータタイプ	説明
fmxslt:get_short_time_format()	文字列	短い時刻書式の形式文字列「h:mm a」を返します。
fmxslt:get_long_time_format()	文字列	長い時刻書式の形式文字列「h:mm:ss a z」を返します。
fmxslt:get_short_day_format()	文字列	短い曜日書式の形式文字列「EEE」を返します。
fmxslt:get_long_day_format()	文字列	長い曜日書式の形式文字列「EEEE」を返します。
fmxslt:compare_date(String 日付１, String 日付２)	数字	この関数は、２つの日付の値を比較します。日付１が日付２より前の場合は負の数を返し、日付１が日付２より後の場合は正の数を返します。日付１と日付２が同一の場合は０を返します。両方の日付を「fm」日付書式で指定する必要があります。
fmxslt:compare_time(String 時刻１, String 時刻２)	数字	この関数は、２つの時刻の値を比較します。時刻１が時刻２より前の場合は負の数を返し、時刻１が時刻２より後の場合は正の数を返します。時刻１と時刻２が同一の場合は０を返します。両方の時刻を「fm」時刻書式で指定する必要があります。
fmxslt:compare_day(String 曜日１, String 曜日２)	数字	この関数は、２つの曜日の値を比較します。曜日１が曜日２より前の場合は負の数を返し、曜日１が曜日２より後の場合は正の数を返します。曜日１と曜日２が同一の場合は０を返します。両方の曜日を短い曜日書式で指定する必要があります。

次の表に示す関数は、日付と時刻の書式を指定するカスタム日付形式文字列を使用します。次のセクション「日付と時刻の形式文字列について」を参照してください。

日付、時刻、および曜日拡張関数	戻り値のデータタイプ	説明
fmxslt:get_datetime(String 日付書式)	文字列	日付および時刻の形式文字列を使用して、現在の日付と時刻を返します。
fmxslt:convert_datetime(String 古い書式, String 新しい書式, String 日付)	文字列	指定した古い書式の指定した日付を、指定した新しい書式に従った文字列に変換します。古い書式と新しい書式の文字列は、日付と時刻の形式文字列を使用して指定する必要があります。
fmxslt:compare_datetime(String 日付書式１, String 日付書式２, String 日付１, String 日付２)	数字	この関数は、日付１と日付２を各日付書式に従ってデコードすることによって、これらの２つの日付を比較します。日付１が日付２より前の場合は負の数を返し、日付１が日付２より後の場合は正の数を返します。日付１と日付２が同一の場合は０を返します。日付書式１と日付書式２の文字列は、日付と時刻の形式文字列を使用して指定する必要があります。

日付と時刻の形式文字列について

日付と時刻の書式は、日付と時刻のパターン文字列で指定します。日付と時刻のパターン文字列内では、クォートで囲まれていない「A」から「Z」および「a」から「z」の文字は、日付または時刻の文字列の構成部分を表すパターン文字として解釈されます。

次のパターン文字が定義されています（「A」から「Z」および「a」から「z」の他の文字もすべて予約されています）。

文字	日付または時刻の構成要素	表示	例
G	年号指定子	テキスト	西暦
y	年	年	1996; 96
M	年を基準とした月	月	7 月; 7; 07
w	年を基準とした週	数字	27
W	月を基準とした週	数字	2
D	年を基準とした日	数字	189
d	月を基準とした日	数字	10

文字	日付または時刻の構成要素	表示	例
F	月を基準とした週の日	数字	2
E	曜日	テキスト	火曜日; 火
a	AM/PM のマーカ	テキスト	午後
H	日を基準とした時間（0 から 23）	数字	0
k	日を基準とした時間（1 から 24）	数字	24
K	AM/PM での時間（0 から 11）	数字	0
h	AM/PM での時間（1 から 12）	数字	12
m	時を基準とした分	数字	30
s	分を基準とした秒	数字	55
S	ミリ秒	数字	978
z	時間帯	一般的な時間帯	太平洋夏時間; PST
Z	時間帯	RFC 822 の時間帯	-0800

パターン文字の数によって正確な表現が決まるため、パターン文字は、通常は繰り返し使用されます。

- テキスト: 書式設定時には、パターン文字の数字が４つ以上の場合、完全な書式が使用されます。パターン文字が３つ以下の場合、使用可能であれば、短い書式または短縮形の書式が使用されます。解析時には、パターン文字の数に応じて両方の書式を使用できます。
- 数字: 書式設定時には、パターン文字の数は最小の桁数になり、それよりも短い数字には、この量になるまでゼロが追加されます。解析時には、パターン文字の数は、連続する２つのフィールドを分離するために必要な場合以外は無視されます。
- 年: 書式設定時には、パターン文字の数が２つの場合、年は２桁に切り詰められます。その他の場合は、数字として解釈されます。

  解析時には、パターン文字の数が２より多い場合、年は、桁数にかかわらず文字どおりに解釈されます。したがって、「yyyy/MM/dd」というパターンを使用すると、「12/01/11」は、西暦 12 年１月 11 日に解釈されます。

  省略形の年のパターン（「y」または「yy」）を使用した解析時には、省略形の年は、日付書式のインスタンスが作成された時点の 80 年前または 20 年後に日付を調整することによって、特定の世紀に対して相対的に解釈されます。たとえば、「yy/MM/dd」というパターンと、1997 年１月１日に作成された日付書式のインスタンスを使用した場合、文字列「12/01/11」は 2012 年１月 11 日に解釈されますが、文字列「64/05/04」は 1964 年５月４日に解釈されます。解析時には、ちょうど２桁で構成される文字列のみがデフォルトの世紀に解析されます。１桁の文字列、３桁以上の文字列、または一部が数字ではない２桁の文字列（例:「-1」）など、他の数値文字列は、文字どおりに解釈されます。したがって、「3/01/02」または「003/01/02」は、同じパターンを使用して、西暦３年１月２日と解釈されます。同様に、「-3/01/02」は、紀元前４年１月２日と解釈されます。
- 月: パターン文字の数が３つ以上の場合、月はテキストとして解釈されます。その他の場合は、数字として解釈されます。
- 一般的な時間帯: 時間帯は、名前がある場合はテキストとして解釈されます。GMT からのオフセットの値を表す時間帯に対しては、次の構文が使用されます。
  - GMT からのオフセットの時間: GMT 記号 時:分
  - 記号: + または -
  - 時: 数字または数字 数字
  - 分: 数字 数字
  - 数字: 0 1 2 3 4 5 6 7 8 9 のいずれか

  時は 0 から 23 の数字、分は 00 から 59 の数字でなければなりません。書式はロケールに依存せず、数字は Unicode 標準の Basic Latin ブロックから使用する必要があります。

  解析時には、RFC 822 の時間帯も使用できます。

- RFC 822 の時間帯: 書式設定時には、RFC 822 の 4 桁の時間帯書式が使用されます。
  - RFC822 の時間帯: 記号 2 桁の時 分
  - 2 桁の時: 数字 数字

2 桁の時は 00 から 23 でなければなりません。他の定義は、一般的な時間帯用です。

解析時には、一般的な時間帯も使用できます。

次の例は、日本のロケールにおいて日付と時刻のパターンがどのように解釈されるかを示します。指定されている日付と時刻は、米国の太平洋標準時の時間帯のローカル時間 2001-07-04 12:08:56 です。

日付と時刻のパターン	結果
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 西暦 at 12:08:56 PDT
"EEE, MMM d, ' 'yy"	水, 7 4, '01
"h:mm a"	12:08 午後
"hh 'o' 'clock' a, zzzz"	12 o'clock 午後, 太平洋夏時間
"K:mm a, z"	0:08 午後, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.7 月.04 西暦 12:08 午後
"EEE, d MMM yyyy HH:mm:ss Z"	水, 4 7 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700

拡張関数のエラーステータスのチェック

XSLT スタイルシート内で次の拡張関数を使用して、直前に呼び出された FileMaker XSLT 拡張関数の現在のエラーステータスをチェックして、ページの処理中に発生したエラーを処理できます。

fmxslt:check_error_status()

fmxslt:check_error_status() 関数が呼び出されると、Web 公開エンジンは、直前に呼び出された関数の現在のエラーコードの値を数字タイプとして返し、エラーステータスを 0 (「エラーなし」) に設定します。エラーコードの値の詳細については、96 ページの「FileMaker XSLT 拡張関数のエラーコード番号」を参照してください。

ログの使用

標準の XSLT <xsl:message> エレメントを使用して、Web 公開エンジンのアプリケーションログファイルにログエントリを書き込むことができます。72 ページの「Web 公開エンジンのアプリケーションログの使用」を参照してください。

スクリプト言語のサーバーサイド処理の使用

Web 公開エンジンに埋め込まれた XSLT 変換処理系には、スクリプト言語のサーバーサイド処理機能が組み込まれています。したがって、JavaScript を使って、XSLT スタイルシートから直接呼び出せる、独自の拡張関数を開発することができます。

FileMaker Server 8 Advanced をインストールする際、この機能を実現する 2 つの Java ライブラリもインストールされます。

- bsf.jar - XSLT 変換処理系をスクリプト言語処理系に接続するためのライブラリ。
- js.jsr - Mozilla プロジェクトが開発した、完全な JavaScript 実装ライブラリ。

以上のライブラリを使って、XSLT スタイルシートコードに埋め込む形で、独自の拡張関数を記述することができます。この拡張関数にはどのような処理でも記述でき、XSLT や XPath だけで実装するよりも扱いやすくなっています。

XSLT 変換処理系の拡張機能については、次の Apache Xalan Extensions Web サイトを参照してください。

http://xml.apache.org/xalan-j/extensions.html

拡張関数の定義

スタイルシートに拡張関数を定義するには、次の操作を行います。

1. 拡張関数を記述するためのネームスペースを定義します。

   「xalan」ネームスペースを追加して、XSLT 変換処理系に対し、拡張コンポーネントをサポートするよう指示します。このとき、独自の拡張関数ネームスペースの名前も指定します。次の例では、拡張関数ネームスペースの接頭語として「fmp-ex」を指定しています。

   ```
   <xsl:stylesheet version="1.0"
   xmlns:xsl=http://www.w3.org/1999/XSL/Transform
   xmlns:xalan=http://xml.apache.org/xslt
   xmlns:fmp-ex="ext1"
   exclude-result-prefixes="xsl xalan fmp-ex">
   ```

2. 拡張コンポーネントと拡張関数を定義し、拡張関数の実装コードを記述します。

   ```
   <xalan:component prefix="fmp-ex" functions="getValueColor">
   <xalan:script lang="javascript">
           function getValueColor(value) {
           if (value > 0)
                   return ("#009900");
           else
                   return ("#CC0000");
           }
   </xalan:script>
   </xalan:component>
   ```

   この例は、入力値に基づいて色の値を返す関数です。入力値が正であれば緑（"#009900"）、そうでなければ赤（"#CC0000"）を返します。

   注意　<xalan:component> 要素は <xsl:stylesheet> 要素の子にする必要があります。

3. スタイルシートに埋め込んだ拡張関数を呼び出して使います。

   XPath ステートメントを使って拡張関数を呼び出す例を示します。

   最初の例では、フォントの色を緑（"#009900"）に設定しています。

   ```
   <font color="{fmp-ex:getValueColor(50)}">The value is 50</font>
   ```

   ２つめの例では、フォントの色を赤（"#CC0000"）に設定しています。

   ```
   <font color="{fmp-ex:getValueColor(-500)}">The value is -500</font>
   ```

拡張関数の例

先に紹介した JavaScript 関数は機能が単純なので、<xsl:choose> ステートメントで実装することも可能です。しかしスクリプト拡張が真に威力を発揮するのは、XSLT や XPath だけでは実装できないような機能の場合です。

たとえば自社のイントラネット用ポータルサイトを構築し、株価情報を随時表示したいとしましょう。XML 形式で株価を提供するサービスはありますが、その多くは商用で、利用するにはライセンスが必要です。一方、Yahoo からは株価データファイルを自由にダウンロードできますが、これはコンマ区切り値（CSV）形式です。XPath document() 関数で XML ドキュメントの内容を読み込むことは可能ですが、CSV 形式のファイルを読み込むためには、あらかじめ XML に変換しておかなければなりません。ひとつの手段として、JavaScript を使って CSV 形式の株価情報ファイルをダウンロードし、必要なデータを抽出する、という方法があります。

この CSV ファイルは次のような URL で取得できます。

```
http://quote.yahoo.com/d/quotes.csv?s=<ticker>&f=l1gh&e=.csv
```

ここで <ticker> は銘柄コード（チッカーシンボル）を表します。

その結果得られるデータは、次のように、３つの数字をコンマで区切った形をしています。

31.79,31.17,32.12

１番目の値は前取引日の終値、２番目は安値、３番目は高値を表します。

次の例は、Yahoo から現在の株価情報を取得する JavaScript XSLT 拡張関数です。この結果は、XPath の機能を使ってアクセスできます。

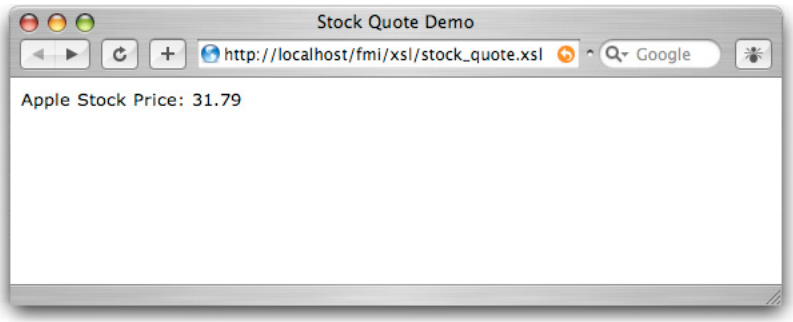
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
	exclude-result-prefixes="xsl fmxslt fmrs xalan fmp-ex"
	version="1.0"
	xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"
	xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"
	xmlns:xalan="http://xml.apache.org/xslt"
	xmlns:fmp-ex="ext1"
>

<?xslt-cwp-query params="-grammar=fmresultset&-process" ?>
<xsl:output method="html"/>
<xalan:component prefix="fmp-ex" functions="include get_quote" >
<xalan:script lang="javascript">
	function include(url) {
		var dest = new java.net.URL(url);
		var dis = new java.io.DataInputStream(dest.openStream());
		var res = "";
		while ((line = dis.readLine()) != null)
		{
			res += line + java.lang.System.getProperty("line.separator");
		}
		dis.close();
		return res;
	}
	function get_quote(ticker) {
		url = "http://quote.yahoo.com/d/quotes.csv?s="+
			"+ticker+"&amp;f=l1gh&amp;e=.csv";
		csv_file = include(url);
		var str_tokenizer = new java.util.StringTokenizer(csv_file, ',');
		// the first token is the last trade price
		var last = str_tokenizer.nextToken();
		return last;
	}
</xalan:script>
</xalan:component>
```

```
<xsl:template match="/fmrs:fmresultset">
    <html>
        <body>
        <font size="2" face="verdana, arial">
        Apple Stock Price:<xsl:value-of select="fmp-ex:get_quote('AAPL')"/>
        </font>
        </body>
    </html>
</xsl:template>

</xsl:stylesheet>
```

Web 公開エンジンは、このスタイルシートを処理すると、Yahoo に対して株価情報を要求します。get_quote() 関数は株価情報データを解析し、その結果をスタイルシートに返します。その結果、ブラウザには次のように表示されます。



第 6 章
サイトのテストと監視

カスタム Web 公開サイトを展開する前に、サイトをテストします。テスト中または展開後に、ログファイルを使用してサイトを監視できます。

カスタム Web 公開サイトのテスト

カスタム Web 公開サイトが使用可能であることをユーザに通知する前に、データベースが意図したとおりに表示され、機能することを確認することが重要です。

- レコードの検索、追加、削除、およびソートなどの機能を異なるアカウントとアクセス権セットでテストする。
- 異なるアカウントでログインして、さまざまなアクセス権セットが意図したとおりに動作することを確認する。権限のないユーザがデータにアクセスしたり、データを変更することができないようにしてください。
- すべてのスクリプトをチェックして、結果が意図したとおりであることを確認する。Web で安全に使用できるスクリプトの設計の詳細については、13 ページの「FileMaker スクリプトとカスタム Web 公開」を参照してください。
- 異なるオペレーティングシステムや Web ブラウザを使ってサイトをテストする。

注意　ネットワークに接続されていない場合でも、Web サーバー、Web 公開エンジン、および FileMaker Server が 1 つのコンピュータにインストールされていれば、URL に “http://localhost/” または “http://127.0.0.1/” と入力することで、カスタム Web 公開サイトをテストできます。URL 構文の詳細については、23 ページの「XML データとオブジェクトにアクセスするための URL 構文について」および 48 ページの「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。

XML 出力をテストするためのスタイルシートの例

次に、XML 出力をテストする場合に役立つ XSLT スタイルシートの例を 2 つ示します。

- 次のスタイルシートの例では、要求された XML データを、変換を行わずに出力します。このスタイルシートは、Web 公開エンジンによって使用される実際の XML データを表示する場合に便利です。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset">
    <xsl:output method="xml"/>
    <xsl:template match="/">
        <xsl:copy-of select="."/>
    </xsl:template>
</xsl:stylesheet>
```

- スタイルシートをデバッグする場合は、次の例の HTML <textarea> タグを使用して、スタイルシートによってアクセスされる XML ソースドキュメントを、スクロールするテキスト領域に表示することができます。同じページで、変換された XSLT 結果を変換前の XML ソースドキュメントに照らして比較できます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset">
    <xsl:output method="html"/>
```

```
<html>
    <body>
    <xsl:template match="/fmrs:fmresultset">
        <textarea rows="20" cols="100">
            <xsl:copy-of select="."/>
        </textarea><br/>
    </xsl:template>
    </body>
</html>
</xsl:stylesheet>
```

サイトの監視

次のタイプのログファイルを使用して、カスタム Web 公開サイトを監視し、サイトにアクセスした Web ユーザに関する情報を収集することができます。

- Web サーバーのアクセスログとエラーログ
- Web 公開エンジンのアプリケーションログ
- Web サーバーモジュールのエラーログ
- Web 公開コアの内部アクセスログ

Web サーバーのアクセスログとエラーログの使用

Apache Web サーバーでは、アクセスログファイルとエラーログファイルが生成されます。Apache アクセスログファイルは、デフォルトでは W3C Common Logfile Format の形式で、Web サーバーへのすべての着信 HTTP リクエストの記録です。Apache エラーログは、HTTP リクエストの処理に関係する問題の記録です。これらのログファイルの詳細については、Apache Web サーバーのマニュアルを参照してください。

Microsoft IIS Web サーバーではアクセスログファイルが生成され、エラーは、ログファイルに書き込まれるのではなく、Windows イベント ビューアに表示されます。アクセスログファイルは、デフォルトでは W3C Extended Log File Format の形式で、Web サーバーへのすべての着信 HTTP リクエストの記録です。アクセスログには W3C Common Logfile Format を使用することもできます。詳細については、Microsoft IIS Web サーバーのマニュアルを参照してください。

W3C Common Logfile Format および W3C Extended Log File Format の詳細については、World Wide Web Consortium の Web サイト www.w3.org を参照してください。

Web 公開エンジンのアプリケーションログの使用

Web 公開エンジンでは、Web 公開エンジンのエラー、スクリプト、およびユーザログ情報の記録であるアプリケーションログファイルがデフォルトで生成されます。

- エラーログ情報には、Web 公開エンジンで発生した一般的ではないエラーが記述されます。Web ユーザに通知された一般的なエラー（たとえば、「データベースが開いていません」など）は記録されません。
- スクリプトログ情報には、Web ユーザがスクリプトを実行したときに生成されたエラーが記録されます。たとえば、スクリプトステップが Web 互換でない場合に、スキップされたスクリプトステップの一覧が記述されます。
- ユーザログメッセージには、XSLT スタイルシートの XSLT <xsl:message> エレメントによって生成されたメッセージが含まれます。Web ユーザが XSLT スタイルシートにアクセスすると、<xsl:message> エレメント内に含めた情報がアプリケーションログファイルに記録されます。第 5 章「FileMaker XSLT スタイルシートの開発」を参照してください。

アプリケーションログは「pe_application_log.txt」という名前で、Web 公開エンジンホスト上の FileMaker Server フォルダにある「Web Publishing」フォルダ内の「Logs」フォルダにあります。

「pe_application_log.txt」ファイルは、Web 公開エンジンの [ログオプション:] で次のいずれかが有効な場合に生成されます。

- [エラーログ:]
- [スクリプトログ:]
- [ユーザログ:]

[ログオプション:] のこれらの３つのオプションは、すべてデフォルトで有効になっています。管理コンソールを使用してこれらのオプションを設定する場合の詳細については、『FileMaker Server Web 公開インストールガイド』を参照してください。

注意　アプリケーションログ内の項目は自動的には削除されないため、時間の経過とともにファイルのサイズが非常に大きくなる場合があります。ホストコンピュータ上のハードディスクのスペースを節約するために、定期的にアプリケーションログファイルのアーカイブを作成することを考慮してください。

Web サーバーモジュールのエラーログの使用

Web サーバーが Web 公開エンジンに接続できない場合は、Web サーバーモジュールによって、処理に関するエラーの記録であるログファイルが生成されます。このファイルは「web_server_module_log.txt」という名前で、Web サーバーホスト上の FileMaker Server フォルダにある「Web Publishing」フォルダ内の「Logs」フォルダにあります。

Web 公開コアの内部アクセスログの使用

Web 公開エンジンの Web 公開コアソフトウェアコンポーネントでは、Web 公開コアへの各アクセスの記録である２つの内部アクセスログファイルが生成されます。

- 「wpc_access_log.txt」アクセスログは、XML を生成したり、FileMaker Server インスタント Web 公開を使用するためのすべてのエンドユーザリクエストの記録です。これらのリクエストは、Web サーバーから Web 公開コアに直接ルーティングされます。
- 「pe_internal_access_log.txt」アクセスログは、XSLT リクエストの処理中に Web 公開エンジンの XSLT-CWP ソフトウェアコンピュータによって実行されるすべての内部 XML リクエストの記録です。これらのリクエストは、Web 公開エンジン内で、XSLT-CWP ソフトウェアコンポーネントから Web 公開コアソフトウェアコンポーネントに内部的にルーティングされます。

これらのログファイルは、Web 公開エンジンホスト上の FileMaker Server フォルダにある「Web Publishing」フォルダ内の「Logs」フォルダにあります。

内部アクセスログは、Web 公開エンジンの [アクセスログ:] オプションが有効な場合に生成されます。デフォルトの設定は [有効] です。管理コンソールを使用して [アクセスログ:] オプションを設定する場合の詳細については、『FileMaker Server Web 公開インストールガイド』を参照してください。

付録 A
クエリー文字列で使用される有効な名前

この付録では、Web 公開エンジンを使用して FileMaker データにアクセスする場合にクエリー文字列で使用できる、クエリーコマンドと引数の有効な名前を説明します。

クエリーコマンドと引数について

次に、すべてのクエリーコマンド名とクエリー引数名の一覧を示します。

クエリーコマンド名	クエリー引数名
-dbnames（79 ページを参照）	-db（81 ページを参照）
-delete（79 ページを参照）	-encoding（XSLT のみ）（81 ページを参照）
-dup（79 ページを参照）	-field（81 ページを参照）
-edit（79 ページを参照）	フィールド名（82 ページを参照）
-find、-findall、-findany（79 ページを参照）	フィールド名.op（82 ページを参照）
-layoutnames（80 ページを参照）	-grammar（XSLT のみ）（83 ページを参照）
-new（80 ページを参照）	-lay（83 ページを参照）
-process（XSLT のみ）（80 ページを参照）	-lay.response（83 ページを参照）
-scriptnames（80 ページを参照）	-lop（83 ページを参照）
-view（81 ページを参照）	-max（84 ページを参照）
	-modid（84 ページを参照）
	-recid（84 ページを参照）
	-script（84 ページを参照）
	-script.param（84 ページを参照）
	-script.prefind（85 ページを参照）
	-script.prefind.param（85 ページを参照）
	-script.presort（85 ページを参照）
	-script.presort.param（86 ページを参照）
	-skip（86 ページを参照）
	-sortfield.[1-9]（86 ページを参照）
	-sortorder.[1-9]（86 ページを参照）
	-stylehref（87 ページを参照）
	-styletype（87 ページを参照）
	-token.[文字列]（XSLT のみ）（88 ページを参照）

重要　-dbnames、-layoutnames、-scriptnames、および -process （XSLT リクエストのみ）以外のすべてのクエリーコマンドで、データベースレイアウトを指定するための -lay クエリー引数が必須です。

使用されなくなったリクエスト名と引数

次に、サポートされなくなったリクエスト名と引数を示します。

使用されなくなったリクエスト名	コメント
-dbopen	「Web」フォルダにある、[リモート管理] が有効なデータベースを開くために使用されていました。
-dbclose	「Web」フォルダにある、[リモート管理] が有効なデータベースを閉じるために使用されていました。
-img	指定されたイメージを取得するために使用されていました。代わりにオブジェクトの URL を使用してください。24 ページの「XML ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」および 48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。

使用されなくなった引数名	コメント
-password	-dbopen にデータベースのパスワードを指定するために使用されていました。Web 公開エンジンでは、HTTP 基本認証が使用されます。17 ページの「Web ユーザがカスタム Web 公開を使用して保護されたデータベースにアクセスする場合」を参照してください。
-format	CDML のフォーマットファイルまたは XML 文法を指定するために使用されていました。CDML はサポートされなくなっています。付録 C「CDML ソリューションの FileMaker XSLT への変換」を参照してください。
-fmtfield	データベース内のフィールドからフォーマットファイルの情報を取得するために使用されていました。
-error -errnum -errfmtfield	CDML のエラー処理に使用されていました。Web 公開エンジンでは、エラーはスタイルシートで処理されます。付録 B「カスタム Web 公開のエラーコード」および 102 ページの「CDML の -error および -errornum 変数タグの変換」を参照してください。
-mailto -mailfrom -mailhost -mailformat -mailcc -mailbcc -mailsub	Web コンパニオンからの電子メールメッセージに使用されていました。電子メールには FileMaker XSLT 拡張関数を使用します。59 ページの「Web 公開エンジンからの電子メールメッセージの送信」を参照してください。

クエリーコマンドと引数の使用のガイドライン

クエリー文字列でクエリーコマンドと引数を使用する場合は、次の点に注意してください。

- クエリー文字列に含めるクエリーコマンドは、1 つだけにする必要があります。クエリーコマンドをまったく指定しないことも、2 つ以上のクエリーコマンドを指定することもできません。たとえば、新しいレコードを追加するためにクエリー文字列に -new を含めることはできますが、同じ文字列に -new と -edit を含めることはできません。
- ほとんどのクエリーコマンドでは、対応するさまざまなクエリー引数をクエリー文字列で指定する必要があります。たとえば、-dbnames および -process 以外のすべてのクエリーコマンドでは、クエリー対象のデータベースを指定する -db 引数が必要です。必要な引数については、32 ページの「FileMaker クエリー文字列を使用した XML データの要求」の表を参照してください。
- クエリー引数とフィールド名には、-db=employees など、使用する特定の値を指定します。クエリーコマンドには、-findall などのコマンド名の後に「=」記号や値を指定しないでください。
- クエリーコマンドと引数の名前は、-delete や -lay のように小文字で指定する必要があります。
- クエリー文字列で使用されるデータベース名、レイアウト名、およびフィールド名では、大文字と小文字は区別されません。たとえば、MyLayout を指定するために -lay=mylayout を使用できます。

注意 クエリー文字列の外部の XSLT ステートメントで使用されるフィールドとデータベースの名前では大文字と小文字が区別され、データベースで使用されている実際の名前と完全に一致する必要があります。たとえば、次のステートメントがあるとします。

```
<xsl:value-of select="fmrs:field[@name='LastName']"/>
```

この場合、フィールド参照「LastName」は、データベースの「LastName」フィールドの名前に完全に一致する必要があります。

- ピリオドが含まれるフィールド名（「text.field」など）に、HTTP クエリを使用して XML または XSLT でアクセスすることはできません。ピリオドは、以下のセクション「完全修飾フィールド名の構文について」で説明するように、レコード ID に使用される予約済みの文字です。
- -find コマンドでは、フィールドの値の大文字と小文字は区別されません。たとえば、Field1=Blue または Field1=blue を使用することができます。-new および -edit コマンドでは、フィールドの値に使用した大文字と小文字は保持され、クエリー文字列で指定したとおりにデータベースに保存されます。たとえば、LastName=Doe などの大文字と小文字は保持されます。

FileMaker クエリー文字列リファレンスについて

FileMaker Server Web Publishing CD には、FileMaker の各クエリーコマンドとクエリー引数の簡単な説明と例が含まれた「Query Strings Reference.fp7」という FileMaker データベースが収録されています。FileMaker クエリー文字列リファレンスは、FileMaker Server Web Publishing CD の「Custom Web Publishing Reference」フォルダにあります。

完全修飾フィールド名の構文について

完全修飾フィールド名により、フィールドのインスタンスが正確に識別されます。一般的な名前を使用したフィールドは別のテーブルに基づく可能性もあるため、場合によっては、エラーを回避するために完全修飾名を使用する必要があります。

次に、完全修飾フィールド名を指定するための構文を示します。

テーブル名::フィールド名(繰り返し数).レコード ID

各要素の意味は、次のとおりです。

- テーブル名には、フィールドが含まれるテーブルの名前を指定します。テーブル名は、フィールドが、クエリー文字列で指定されているレイアウトの基本テーブルにない場合にのみ必要です。
- フィールド名(繰り返し数) には、繰り返しフィールドの特定の値を指定します。これは、繰り返しフィールドに対してのみ必要です。繰り返し数は数字の 1 から始まります。たとえば、フィールド名(2) は、繰り返しフィールドの 2 番目の値を参照します。繰り返しフィールドに対して繰り返し数を指定しなかった場合は、繰り返しフィールドの最初の値が使用されます。繰り返し数は、繰り返しフィールドが含まれる -new および -edit クエリーコマンドでは必要ですが、-find コマンドでは必要ありません。
- レコード ID には、レコード ID を指定します。クエリー文字列を使用して、ポータルフィールドにレコードを追加したり、ポータルフィールド内のレコードを編集する場合にのみ必要です。次のセクション「ポータルへのレコードの追加」および「ポータル内のレコードの編集」を参照してください。レコード IDは、ポータルフィールドが含まれる -new および -edit クエリーコマンドでは必要ですが、-find コマンドでは必要ありません。

注意　フィールドにアクセスできるようにするには、フィールドが、クエリー文字列で指定するレイアウト上に配置されている必要があります。

ポータルへのレコードの追加

親レコードを追加すると同時にポータルに新しいレコードを追加するには、-new クエリーコマンドを使用して、リクエストのクエリー文字列で次のように指定します。

- 関連するポータルフィールドに対して完全修飾フィールド名を使用する
- 関連するポータルフィールドの名前の後に、レコード ID として 0 を指定する
- 関連するポータルフィールドを指定する前に、親レコードの少なくとも 1 つのフィールドを指定する
- 親レコードの照合フィールド（キーフィールド）のデータを指定する

たとえば、次の URL では、「Employees」に John Doe の新しい親レコードを追加すると同時に、ポータルに Jane の新しい関連レコードを追加します。関連テーブルの名前は「Dependents」で、ポータル内の関連フィールドの名前は「Names」です。照合フィールド「ID」には、従業員の ID 番号が保存されています。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&FirstName=John&LastName=Doe&ID=9756&Dependents::Names.0=Jane&-new

注意　1 つのリクエストでポータルに追加できる関連レコードは 1 つだけです。

ポータル内のレコードの編集

ポータル内の 1 つまたは複数のレコードを編集するには、-edit コマンドとレコード ID を使用して、編集するポータルレコードが含まれる親レコードを指定します。そのレコード ID を完全修飾フィールド名で使用して、編集する特定のポータルレコードを指定します。レコード ID は、XML データの <relatedset> エレメント内にある <record> エレメントの record-id 属性から判断できます。26 ページの「fmresultset 文法の使用」を参照してください。

たとえば、次の URL では、親レコードのレコード ID が 1001 であるポータル内のレコードを編集します。「Dependents」は関連テーブルの名前、「Names」はポータル内の関連フィールドの名前、「Names.2」の「2」はポータルレコードのレコード ID です。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-recid=1001
&Dependents::Names.2=Kevin&-edit

次に、1 つのリクエストを使用して、親レコード経由で複数のポータルレコードを編集する方法の例を示します。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-recid=1001
&Dependents::Names.2=Kevin&Dependents::Names.5=Susan&-edit

-edit コマンドを使用してポータルのレコード ID として 0 を指定し、既存の親レコードに対してポータル内で新しい関連レコードを追加することもできます。次に例を示します。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-recid=1001
&Dependents::Names.0=Timothy&-edit

グローバルフィールドを指定するための構文について

次に、グローバルフィールドを指定するための構文を示します。

テーブル名::フィールド名(繰り返し数).global

.global により、フィールドは、グローバル格納を使用するものと識別されます。テーブル名およびフィールド名(繰り返し数) の詳細については、77 ページの「完全修飾フィールド名の構文について」を参照してください。グローバルフィールドの詳細については、FileMaker Pro ヘルプを参照してください。

クエリー文字列内でグローバルフィールドを識別するには、.global の構文を使用する必要があります。Web 公開エンジンでは、グローバルフィールドの引数値は、クエリーコマンドを実行する前、またはクエリー文字列内の他の引数値を設定する前に設定されます。直接の XML リクエスト、およびセッションを使用しない XSLT スタイルシートによって実行されるリクエストでは、グローバル値は、リクエストの実行後ただちに使用期限が切れます。セッションを使用する XSLT スタイルシートによって実行されるリクエストでは、スタイルシートで定義されたセッションの期間中、または他のリクエストで変更されるまで、グローバル値は維持されます。

.global 構文を使用してクエリー文字列でグローバルフィールドを識別しない場合、Web 公開エンジンは、最初にグローバルフィールドの値を設定せずに、クエリー文字列の残りの部分を使用してグローバルフィールドを評価します。

次に例を示します。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&Country.global=USA&-edit

注意 -edit コマンドを使用してグローバルフィールドの値を設定する場合、そのリクエストでグローバルフィールドの値を設定するだけのときは、-recid 引数を使用する必要はありません。

重要 XSLT スタイルシートでグローバルフィールドを使用する場合は、管理コンソールを使用して、Web 公開エンジンの [XSLT データベースセッション :] オプションを有効にする必要があります。このオプションが有効でない場合、グローバルフィールドの値は、リクエスト間で維持されません。『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

クエリーコマンドの使用

このセクションでは、XML および XSLT のリクエストで使用可能なクエリーコマンドについて説明します。

注意 XSLT リクエストの場合のみ、次のすべてのクエリーコマンドで -grammar クエリー引数が必要です。

-dbnames（データベース名）クエリーコマンド

FileMaker Server でホストされていて、XML または XSLT を使用したカスタム Web 公開が有効なすべてのデータベースの名前を取得します。

必須のクエリー引数:（なし）

例:

データベース名を取得する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-dbnames

-delete（レコード削除）クエリーコマンド

-recid 引数で指定されているレコードを削除します。

必須のクエリー引数: –db, –lay, –recid

オプションのクエリー引数: –script

例:

レコードを削除する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-recid=4&-delete

-dup（レコード複製）クエリーコマンド

-recid で指定されているレコードを複製します。

必須のクエリー引数: -db、-lay、-recid

オプションのクエリー引数: -script

例:

指定したレコードを複製する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-recid=14&–dup

-edit（レコード編集）クエリーコマンド

任意のフィールド名/値の組の内容をフィールドに入れて、-recid 引数で指定されているレコードを更新します。-recid 引数は編集されるレコードを示します。

必須のクエリー引数: -db、-lay、-recid、１つまたは複数のフィールド名

オプションのクエリー引数: -modid、-script

注意　ポータル内のレコードの編集の詳細については、78 ページの「ポータル内のレコードの編集」を参照してください。

例:

レコードを編集する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-recid=13&Country=USA&-edit

-find、-findall、または -findany（レコードの検索）クエリーコマンド

定義された条件を使用して検索リクエストを送信します。

必須のクエリー引数: -db、-lay

オプションのクエリー引数: -recid、-lop、-op、-max、-skip、-sortorder、-sortfield、-script、-script.prefind、-script.presort、フィールド名

例

レコードをフィールド名で検索する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&Country=USA&-find

注意　1つのリクエストでフィールド名を複数回指定することは、サポートされていません。FileMaker Server はすべての値を解析しますが、解析された最後の値のみを使用します。

レコードをレコード ID で検索する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-recid=427&–find

データベース内のすべてのレコードを検索する場合は、-findall を使用します。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-findall

ランダムなレコードを検索する場合は、-findany を使用します。

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family&-findany

-layoutnames（レイアウト名）クエリーコマンド

FileMaker Server でホストされていて、XML または XSLT を使用したカスタム Web 公開が有効な指定されたデータベースで使用可能なレイアウトすべての名前を取得します。

必須のクエリー引数: -db

例:

使用可能なレイアウトの名前を取得する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-layoutnames

-new（新規レコード）クエリーコマンド

新規レコードを作成し、そのレコードに任意のフィールド名/値の組の内容を入れます。

必須のクエリー引数: -db、-lay

オプションのクエリー引数: 1 つまたは複数のフィールド名、-script

注意　ポータルに新しいレコードを含める場合の詳細については、77 ページの「ポータルへのレコードの追加」を参照してください。

例:

新規レコードを追加する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&Country=Australia&-new

-process（XSLT スタイルシートの処理）

データベースのデータを要求せずに、XSLT スタイルシートを処理します。このクエリーコマンドは、XSLT スタイルシートとともにのみ使用できます。

必須のクエリー引数: -grammar。fmresultset または FMPXMLRESULT 文法を使用する必要があります。

例:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?-grammar=fmresultset&-process

53 ページの「FileMaker Server に対してクエリーを実行しない XSLT リクエストの処理」を参照してください。

-scriptnames（スクリプト名）クエリーコマンド

FileMaker Server でホストされていて、XML または XSLT を使用したカスタム Web 公開が有効な、指定されたデータベースで使用可能なスクリプトすべての名前を取得します。

必須のクエリー引数: -db

例:

すべてのスクリプトの名前を取得する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-scriptnames

-view（レイアウト情報の表示）クエリーコマンド

FMPXMLLAYOUT 文法が指定されている場合は、データベースからレイアウト情報を取得して、FMPXMLLAYOUT 文法で表示します。データ文法（fmresultset または FMPXMLRESULT）が指定されている場合は、XML ドキュメントの metadata セクションおよび空のレコードセットを取得します。

必須のクエリー引数: -db、-lay

オプションのクエリー引数: -script

例

レイアウト情報を取得する場合:

http://192.168.123.101/fmi/xml/FMPXMLLAYOUT.xml?-db=employees&-lay=departments&-view

メタデータ情報を取得する場合:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-view

クエリー引数の使用

このセクションでは、XML および XSLT リクエストで使用可能なクエリー引数について説明します。XSLT リクエストでのみ使用可能な引数の詳細については、49 ページの「FileMaker XSLT スタイルシートでのクエリー文字列の使用」を参照してください。

-db（データベース名）クエリー引数

クエリーコマンドを適用するデータベースを指定します。

値: データベースの名前（ファイル拡張子がある場合は、拡張子を含まない名前）

注意　クエリー引数で -db 引数にデータベースの名前を指定する場合は、ファイル拡張子を含めないでください。実際のデータベースファイル名にはオプションで拡張子を含めることができますが、-db 引数の値として拡張子は使用できません。

必須: -dbnames および -process 以外のすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-findall

-encoding（XSLT リクエストのエンコード）クエリー引数

XSLT リクエストのテキストエンコードを指定します。このクエリーコマンドは、XSLT を使用したカスタム Web 公開のリクエストでのみ使用できます。

値: US-ASCII、ISO-8859-1、ISO-8859-15、ISO-2022-JP、Shift_JIS、または UTF-8

オプション: XSLT リクエストのすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?-db=employees&-lay=departments
&-grammar=fmresultset&-encoding=Shift_JIS&-findall

51 ページの「リクエストのテキストエンコードの設定」を参照してください。

-field（オブジェクトフィールド名）クエリー引数

オブジェクトフィールドの名前を指定します。

必須: オブジェクトフィールドのデータに対するリクエスト

24 ページの「XML ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」および 48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。

フィールド名（オブジェクトフィールド以外のフィールド名）クエリー引数

フィールド名は、-find クエリーコマンドの条件の制御とレコードの内容の変更に使用されます。クエリーコマンドや引数にオブジェクトフィールド以外のフィールドの値を指定する必要がある場合は、名前/値の組の名前の部分としてハイフン（-）文字を付けずにフィールド名を使用します。

名前: FileMaker データベース内のフィールドの名前。フィールドが、クエリー文字列で指定されたレイアウトの基本テーブルにない場合、フィールド名は完全修飾されている必要があります。ピリオドが含まれるフィールド名（「text.field」など）に、HTTP クエリを使用して XML または XSLT でアクセスすることはできません。ピリオドは、完全修飾フィールド名でレコード ID に使用される予約済みの文字です。77 ページの「完全修飾フィールド名の構文について」を参照してください。

値: -new および -edit クエリーコマンドでは、現在のレコード内のフィールドに保存する値を指定します。-find クエリーコマンドでは、フィールドで検索する値を指定します。日付、時刻、またはタイムスタンプフィールドの値を指定する場合は、「fm」形式を使用してそのフィールドタイプに値を指定します。「fm」形式では、MM/dd/yyyy が日付、HH:mm:ss が時刻、MM/dd/yyyy HH:mm:ss がタイムスタンプです。

必須: -edit クエリーコマンド

オプション: -new および -find クエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-op=eq&FirstName=Sam&-max=1&-find

注意　1つのリクエストにフィールド名を複数回指定することは、サポートされていません。FileMaker Server はすべての値を解析しますが、解析された最後の値のみを使用します。

フィールド名.op（比較演算子）クエリー引数

演算子の前に指定したフィールド名に適用する比較演算子を指定します。比較演算子は、-find クエリーコマンドとともに使用します。

値: 使用する演算子。デフォルトの演算子は「begins with」です。次に、有効な演算子を示します。

キーワード	FileMaker Pro の演算子
eq	=値
cn	*値*
bw	値*
ew	*値
gt	> 値
gte	>=値
lt	< 値
lte	<=値
neq	除外, 値

オプション: -find クエリーコマンド

必須: フィールド名と値

次に、比較演算子を指定するための構文を示します。

テーブル名::フィールド名=値&テーブル名::フィールド名.op=演算子記号

各要素の意味は、次のとおりです。

- テーブル名には、フィールドが含まれるテーブルを指定します。フィールドが、クエリー文字列で指定されているレイアウトの基本テーブルにない場合にのみ必要です。
- 演算子記号には、cn など、前の表に示されているキーワードの 1 つを指定します。

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&name=Tim&name.op=cn&-find

bw キーワードを指定して、FileMaker Pro の任意の検索演算子を使用できます。たとえば、範囲の演算子（...）を使用して値の範囲を検索するには、bw キーワードを指定して、検索条件の前に「...」の文字を配置します。

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&IDnum=915...925&IDnum.op=bw
&-find

テキストの検索に使用できる演算子の詳細については、FileMaker Pro ヘルプを参照してください。

-grammar（XSLT スタイルシートの文法）クエリー引数

XSLT スタイルシートに対して使用する文法を指定します。このクエリーコマンドは、XSLT を使用したカスタム Web 公開のリクエストでのみ使用できます。

値: fmresultset、FMPXMLRESULT、または FMPXMLLAYOUT

必須: すべての XSLT リクエスト

例:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?-grammar=fmresultset&-db=mydatabase
&-lay=mylayout&-findall

49 ページの「FileMaker XSLT スタイルシートの XML 文法の指定」を参照してください。

-lay（レイアウト）クエリー引数

使用するデータベースのレイアウトを指定します。

値: レイアウトの名前

必須: -dbnames、-layoutnames、-scriptnames、および-process（XSLT リクエストのみ）以外のすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–view

-lay.response（応答のレイアウトの切り替え）クエリー引数

リクエストを処理する際には -lay 引数で指定されているレイアウトを使用し、XML 応答を処理する際には -lay.response 引数で指定されているレイアウトに切り替えるよう指定します。

-lay.response 引数が含まれていない場合は、リクエストの処理時も、応答の処理時も、-lay 引数で指定されているレイアウトが使用されます。

-lay.response 引数は、XML リクエストに対して、または XSLT スタイルシートリクエスト内のいずれかで使用できます。

値: レイアウトの名前

オプション: -dbnames、-layoutnames、-scriptnames、および-process（XSLT リクエストのみ）以外のすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=Budget&Salary=100000&Salary.op=gt&-find
&-lay.response=ExecList

-lop（論理演算子）クエリー引数

-find クエリーコマンドに含まれる複数の検索条件を「and」または「or」のいずれの検索として組み合わせるかを指定します。

値: 「and」または「or」（小文字で指定する必要があります）。-lop クエリー引数が含まれない場合、-find クエリーコマンドでは and の値が使用されます。

オプション: -find クエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&Last+Name=Smith
&Birthdate=2/5/1972&-lop=and&-find

-max（最大レコード）クエリー引数

返されるレコードの最大数を指定します。

値: 数字。すべてのレコードを返すには、値 all を使用します。値 all は小文字で指定する必要があります。-max が指定されていない場合は、すべてのレコードが返されます。

オプション: -find または -findall クエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-max=10&-findall

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-max=all&-findall

-modid（修正 ID）クエリー引数

修正 ID は、レコードの現在のバージョンを指定する増加するカウンタです。-edit クエリーコマンドを使用する際に修正 ID を指定することで、確実にレコードの現在のバージョンを編集できます。指定した修正 ID の値がデータベースの現在の修正 ID の値に一致しない場合、-edit クエリーコマンドは使用できず、エラーコードが返されます。

値: 修正 ID。修正 ID は、FileMaker データベースのレコードの現在のバージョンを指定する固有の ID です。

オプション: -edit クエリーコマンド

必須: -recid 引数

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-recid=22&-modid=6
&last_name=Jones&-edit

-recid（レコード ID）クエリー引数

処理するレコードを指定します。主に -edit および -delete クエリーコマンドで使用されます。

値: レコード ID。レコード ID は、FileMaker データベースのレコードの固有の識別子です。

必須: -edit、-delete、および -dup クエリーコマンド

オプション: -find クエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-recid=22&-delete

-script（スクリプト）クエリー引数

クエリーコマンドとソートの実行後に実行する FileMaker スクリプトを指定します。34 ページの「XML リクエストの処理方法の理解」を参照してください。

値: スクリプト名

オプション: -dbnames、-layoutnames、-process、および-scriptnames 以外のすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-script=myscript&–findall

-script.param（スクリプトに引数を渡す）クエリー引数

引数を-script に指定された FileMaker スクリプトに渡します。

値: 1つのテキスト引数

- 複数の引数を渡す場合は、引数を区切ってストリングを作成すると、個々の引数がスクリプト内で解析されます。たとえば、「|」文字を使用して「param1|param2|param3」をリストとして渡す場合は、次のように URL エンコードされます。param1%7Cparam2%7Cparam3
- テキスト引数をテキストでない値として処理する場合は、スクリプトでテキスト値を変換します。たとえば、テキスト値を数字に変換する場合は、スクリプトに以下を含めることができます。GetAsNumber(Get(スクリプト引数))
- クエリーに -script が含まれずに -script.param が含まれている場合は、-script.param は無視されます。
- クエリーに複数の –script.param が含まれている場合は、Web 公開エンジンは最後の値を使用して解析します。

オプション: –script

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-script=myscript
&-script.param=Smith%7CChatterjee%7CSu&-findall

-script.prefind（検索前のスクリプト）クエリー引数

-find クエリーコマンドの処理時に、レコードの検索とソート（指定されている場合）の前に実行する FileMaker スクリプトを指定します。

値: スクリプト名

オプション: -dbnames、-layoutnames、-process、および-scriptnames以外のすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-script.prefind=myscript&-findall

-script.prefind.param（検索する前に引数をスクリプトに渡す）クエリー引数

引数を –script.prefind に指定された FileMaker スクリプトに渡します

値: 1つのテキスト引数

- 複数の引数を渡す場合は、引数を区切ってストリングを作成し、個々の引数をスクリプト内で解析します。たとえば、「|」文字を使用して「param1|param2|param3」をリストとして渡す場合は、次のように URL エンコードされます。param1%7Cparam2%7Cparam3
- テキスト引数をテキストでない値として処理する場合は、スクリプトでテキスト値を変換します。たとえば、テキスト値を数字に変換する場合は、スクリプトに以下を含めることができます。GetAsNumber(Get(スクリプト引数))
- クエリーに -script.prefind が含まれずに -script.prefind.param が含まれている場合は、-script.prefind.param は無視されます。
- クエリーに複数の -script.prefind.param が含まれている場合は、Web 公開エンジンは最後の値を使用して解析します。

オプション: -script.prefind

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-script.prefind=myscript
&-script.prefind.param=payroll&-findall

-script.presort（ソート前のスクリプト）クエリー引数

-find クエリーコマンドの処理時に、レコードの検索（指定されている場合）後、レコードのソート前に実行する FileMaker スクリプトを指定します。

オプション: -dbnames、-layoutnames、-process、および-scriptnames以外のすべてのクエリーコマンド

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-script.presort=myscript
&-sortfield.1=dept&-sortfield.2=rating&-findall

-script.presort.param（ソートする前に引数をスクリプトに渡す）クエリー引数

引数を -script.presort に指定された FileMaker スクリプトに渡します

値: 1つのテキスト引数

- 複数の引数を渡す場合は、引数を区切ってストリングを作成し、個々の引数をスクリプト内で解析します。たとえば、「|」文字を使用して「param1|param2|param3」をリストとして渡す場合は、次のように URL エンコードされます。param1%7Cparam2%7Cparam3
- テキスト引数をテキストでない値として処理する場合は、スクリプトでテキスト値を変換します。たとえば、テキスト値を数字に変換する場合は、スクリプトに以下を含めることができます。GetAsNumber(Get(スクリプト引数))
- クエリーに -script.presort が含まれずに -script.presort.param が含まれている場合は、-script.presort.param は無視されます。
- クエリーに複数の -script.presort.param が含まれている場合は、Web 公開エンジンは最後の値を使用して解析します。

オプション: -script.presort

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-script.presort=myscript
&–script.presort.param=18%7C65&-sortfield.1=dept&-sortfield.2=rating&-findall

-skip（レコードのスキップ）クエリー引数

対象レコード内のスキップするレコードの数を指定します。

値: 数字。値が対象レコード内のレコード数より大きい場合、レコードは表示されません。デフォルト値は 0 です。

オプション: -find クエリーコマンド

次の例では、結果の最初の 10 レコードがスキップされ、11 番目から 15 番目のレコードが返されます。

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-skip=10&-max=5&–findall

-sortfield（ソートフィールド）クエリー引数

ソートに使用するフィールドを指定します。

値: フィールド名

オプション: -find または -findall クエリーコマンド

-sortfield クエリー引数を使用して、複数のフィールドソートを複数回実行できます。次に、ソートフィールドの優先順位を指定するための構文を示します。

-sortfield.優先順位番号=完全修飾フィールド名

-sortfield.優先順位番号クエリー引数の優先順位番号には、複数のソートフィールドを使用する場合の優先順位を指定する、1 から始まる数字を指定します。

次の例では、最初に「dept」フィールドでソートされ、続いて「rating」フィールドでソートされます。-sortorder クエリー引数が指定されていないため、両方のフィールドは昇順でソートされます。

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=performance&-sortfield.1=dept
&-sortfield.2=rating&-findall

-sortorder（ソート順）クエリー引数

ソート順を指定します。

値: ソート順。次に、有効なソート順を示します。<値一覧名> には、Custom などの値一覧名を指定します。

キーワード	FileMaker Pro の演算子
ascend	a から z、-10 から 10 の昇順のソート
descend	z から a、10 から -10 の降順のソート
<値一覧名>	レイアウト上のフィールドに割り当てられた、指定した値一覧を使用したソート

オプション: -find または -findall クエリーコマンド

必須: -sortfield クエリー引数

-sortorder クエリー引数を -sortfield クエリー引数とともに使用して、複数のソートフィールドのソート順を指定できます。次に、ソートフィールドのソート順を指定するための構文を示します。

-sortorder.優先順位番号=ソート方法

各要素の意味は、次のとおりです。

- -sortorder.優先順位番号引数の優先順位番号には、-sortorder クエリー引数の適用先の -sortfield クエリー引数を指定する 1 から 9 の数字を指定します。
- ソート方法には、ascend など、前の表に示されているソート順を指定するためのキーワードの 1 つを指定します。

次の例では、最も優先順位の高いソートフィールド（「dept」）のソート順は ascend で、2 番目に優先順位の高いソートフィールド（「rating」）のソート順は descend になっています。-sortorder.2 の優先順位番号 2 により、クエリー引数 -sortorder.2=descend が -sortfield.2=rating クエリー引数に適用されるように指定されています。

例:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=performance&-sortfield.1=dept
&-sortorder.1=ascend&-sortfield.2=rating&-sortorder.2=descend&-findall

注意 ソートフィールドに対して -sortorder クエリー引数が指定されていない場合は、デフォルトの昇順ソートが使用されます。

-stylehref（スタイル href）クエリー引数

XML ドキュメントでクライアントサイドの CSS（Cascading Style Sheet）または XSLT スタイルシートを使用できるように、出力ドキュメント内に XML スタイルシート処理命令を生成して、href 属性の値（href=/mystylesheet.css または href=/stylesheets/mystylesheet.xsl）を設定します。-stylehref 引数の値には、絶対パスを使用する必要があります。スタイルシートの名前には任意の名前を指定できますが、.css または .xsl のいずれかの拡張子を含める必要があります。35 ページの「サーバーサイドおよびクライアントサイドでのスタイルシートの処理の使用」を参照してください。この引数は -styletype 引数とともに使用します。

オプション: すべてのクエリーコマンド

必須: -styletype 引数

例（「mystylesheet.xsl」が Web サーバーソフトウェアのルートフォルダにあると想定しています）:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-styletype=text/xsl
&-stylehref=/mystylesheet.xsl&-findall

-styletype（スタイルタイプ）クエリー引数

XML ドキュメントでクライアントサイドの CSS（Cascading Style Sheet）または XSLT スタイルシートを使用できるように、出力ドキュメント内に XML スタイルシート処理命令を生成して、type 属性の値（type=text/css または type=text/xsl）を設定します。35 ページの「サーバーサイドおよびクライアントサイドでのスタイルシートの処理の使用」を参照してください。この引数は -stylehref 引数とともに使用します。

オプション: すべてのクエリーコマンド

必須: -stylehref 引数

例（「mystylesheet.css」が Web サーバーソフトウェアのルートフォルダにあると想定しています）:

http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=departments&-styletype=text/css
&-stylehref=/mystylesheet.css&-findall

-token.[文字列]（XSLT スタイルシート間での値の受け渡し）クエリー引数

セッションまたは Cookie を使用せずに、XSLT スタイルシート間でユーザ定義の任意の情報を渡します。このクエリー引数は、XSLT を使用したカスタム Web 公開のリクエストでのみ使用できます。

-token.[文字列] の文字列: ０から９の数字、a から z の小文字の英字、または A から Z の大文字の英字を含む、任意の長さの任意の英数字（空白を除く）の文字列を指定します。

ユーザ定義引数の値: URL エンコードされた任意の文字列

オプション: すべての XSLT リクエスト

例:

http://192.168.123.101/fmi/xsl/template/my_stylesheet.xsl?-db=employees&-lay=departments
&-grammar=fmresultset&-token.D100=Active&-findall

53 ページの「トークンを使用したスタイルシート間での情報の受け渡し」を参照してください。

付録 B
カスタム Web 公開のエラーコード

Web 公開エンジンでは、カスタム Web 公開で発生する可能性がある次の３種類のエラーコードがサポートされています。

- データベースおよびクエリー文字列のエラー : XML データリクエストが発生すると、常に、公開されているデータベースのエラーコードが Web 公開エンジンによって生成されます。詳細については、次のセクション「FileMaker データベースのエラーコード番号」を参照してください。
- Web 公開エンジンのエラー : Web 公開エンジンが開発モードの場合に、Web 公開エンジン自体でエラーが発生したときは、特定のエラーページが生成されます。実作業モードの場合は、一般的なテキストメッセージが表示されます。94 ページの「Web 公開エンジンのエラーコード番号」を参照してください。
- FileMaker XSLT 拡張関数のエラー : XSLT スタイルシート内で fmxslt:check_error_status() 拡張関数を使用して、拡張関数が呼び出されたときに拡張関数のエラーステータスをチェックできます。96 ページの「FileMaker XSLT 拡張関数のエラーコード番号」を参照してください。

FileMaker データベースのエラーコード番号

データが要求されると、常に、XML 形式で公開されているデータベースのエラーコードが Web 公開エンジンによって生成されます。このタイプのエラーコードの値は、XML ドキュメントの先頭に、fmresultset 文法の場合は <error code> エレメント、FMPXMLRESULT または FMPDSORESULT 文法の場合は <ERRORCODE> エレメントでそれぞれ挿入されます。エラーコード０は、エラーが発生していないことを示します。

次に、fmresultset 文法のデータベースエラーコードの例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE fmresultset PUBLIC "-//FMI//DTD fmresultset//EN" "/fmi/xml/fmresultset.dtd">
<fmresultset xmlns="http://www.filemaker.com/xml/fmresultset" version="1.0">
    <error code="0"></error>
```

次に、FMPXMLRESULT 文法のデータベースエラーコードの例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLRESULT PUBLIC "-//FMI//DTD FMPXMLRESULT//EN" "/fmi/xml/FMPXMLRESULT.dtd">
<fmpxmlresult xmlns="http://www.filemaker.com/fmpxmlresult">
    <ERRORCODE>0</ERRORCODE>
```

<error code> または <ERRORCODE> エレメントの値をチェックして適切に処理することは、カスタム Web 公開ソリューションの開発者の責任です。Web 公開エンジンによってデータベースエラーが処理されることはありません。

エラー番号	説明
-1	原因不明のエラー
0	エラーなし
1	ユーザによるキャンセル
2	メモリエラー
3	コマンドが使用できません（たとえば誤ったオペレーティングシステム、誤ったモードなど）
4	コマンドが見つかりません
5	コマンドが無効です（たとえば、[フィールド設定] スクリプトステップに計算式が指定されていない場合など）
6	ファイルが読み取り専用です
7	メモリ不足

エラー番号	説明
8	空白の結果
9	アクセス権が不十分です
10	要求されたデータが見つかりません
11	名前が有効ではありません
12	名前がすでに存在します
13	ファイルまたはオブジェクトが使用中です
14	範囲外
15	0 で割ることができません
16	処理に失敗したため、再試行が必要です（たとえば、ユーザクエリーなど）
17	外国語の文字セットの UTF-16 への変換に失敗しました
18	続行するには、クライアントはアカウント情報を指定する必要があります
19	文字列に A から Z、a から z、0 から 9（ASCII）以外の文字が含まれています
100	ファイルが見つかりません
101	レコードが見つかりません
102	フィールドが見つかりません
103	リレーションシップが見つかりません
104	スクリプトが見つかりません
105	レイアウトが見つかりません
106	テーブルが見つかりません
107	索引が見つかりません
108	値一覧が見つかりません
109	アクセス権セットが見つかりません
110	関連テーブルが見つかりません
111	フィールドの繰り返しが無効です
112	ウインドウが見つかりません
113	関数が見つかりません
114	ファイル参照が見つかりません
130	ファイルが損傷しているか見つからないため、再インストールする必要があります
131	言語パックファイルが見つかりません（テンプレートなど）
200	レコードアクセスが拒否されました
201	フィールドを変更できません
202	フィールドアクセスが拒否されました
203	ファイルに印刷するレコードがないか、入力したパスワードでは印刷できません
204	ソート優先順位に指定されたフィールドにアクセスできません
205	ユーザに新規レコードを作成するアクセス権がありません。既存のデータはインポートしたデータで上書きされます
206	ユーザにパスワードの変更アクセス権がないか、変更可能なファイルではありません
207	ユーザにデータベーススキーマを変更する十分なアクセス権がないか、変更可能なファイルではありません
208	パスワードに十分な文字が含まれていません
209	既存のパスワードと新規パスワードを同一にすることはできません
210	ユーザアカウントが非アクティブです
211	パスワードが期限切れです

エラー番号	説明
212	ユーザアカウントまたはパスワードが無効です。再試行してください
213	ユーザアカウントまたはパスワードが存在しません
214	ログイン試行回数が多すぎます
215	管理者権限は複製できません
216	ゲストアカウントは複製できません
217	ユーザに管理者アカウントを変更する十分なアクセス権がありません
300	ファイルがロックされているか、使用中です
301	別のユーザがレコードを使用中です
302	別のユーザがテーブルを使用中です
303	別のユーザがデータベーススキーマを使用中です
304	別のユーザがレイアウトを使用中です
306	レコード修正 ID が一致しません
400	検索条件が空です
401	検索条件に一致するレコードがありません
402	選択したフィールドはルックアップの照合フィールドではありません
403	評価版の FileMaker Pro に設定されている最大レコード数の制限を超過しています
404	ソート優先順位が無効です
405	指定したレコード数が除外可能なレコード数を超過しています
406	全置換またはシリアル番号の再入力に指定された条件が無効です
407	片方または両方の照合フィールドが欠けています（無効なリレーションシップ)
408	指定されたフィールドのデータが不適切なため、この処理を実行できません
409	インポート順が無効です
410	エクスポート順が無効です
412	ファイルの修復に、誤ったバージョンの FileMaker Pro が使用されました
413	指定されたフィールドのフィールドタイプが不適切です
414	レイアウトに結果を表示できません
415	必要な 1 つまたは複数の関連テーブルが使用できません
500	日付の値が入力値の制限を満たしていません
501	時刻の値が入力値の制限を満たしていません
502	数字の値が入力値の制限を満たしていません
503	フィールドの値が入力値の制限オプションに指定されている範囲内に入っていません
504	フィールドの値が入力値の制限オプションで要求されているようにユニークな値になっていません
505	フィールドの値が入力値の制限オプションで要求されているようにデータベースファイル内の既存値になっていません
506	フィールドの値が入力値の制限オプションに指定されている値一覧に含まれていません
507	フィールドの値が入力値の制限オプションに指定されている計算式を満たしません
508	検索モードに無効な値が入力されました
509	フィールドに有効な値が必要です
510	関連する値が空であるか、使用できません
511	フィールド内の値が最大文字数を超過しました
600	印刷エラーが発生しました
601	ヘッダとフッタの高さを加算するとページの高さを超えます

エラー番号	説明
602	現在の段数設定ではボディ部分がページ内に収まりません
603	印刷接続が遮断されました
700	インポートできないファイルタイプです
706	EPSF ファイルにプレビューイメージがありません
707	グラフィックの変換ファイルが見つかりません
708	ファイルをインポートできないか、ファイルをインポートするにはカラーのコンピュータが必要です
709	QuickTime ムービーのインポートに失敗しました
710	データベースファイルが読み取り専用になっているため QuickTime ファイルの参照を更新できません
711	インポートの変換ファイルが見つかりません
714	入力したパスワードでは設定されている権限が不足しているためこの操作は認められていません
715	指定された Excel ワークシートまたは名前の付いた範囲がありません
716	ODBC インポートでは、DELETE、INSERT、または UPDATE を使用する SQL クエリーは使用できません
717	インポートまたはエクスポートを続行するための十分な XML/XSLT 情報がありません
718	（Xerces からの）XML ファイルの解析エラーです
719	（Xalan からの）XSL を使用した XML 変換エラーです
720	エクスポート時のエラー。対象のドキュメントフォーマットでは繰り返しフィールドはサポートされていません
721	パーサまたはトランスフォーマで原因不明のエラーが発生しました
722	フィールドのないファイルにデータをインポートすることはできません
723	インポート先のテーブルでレコードを追加または変更する権限がありません
724	インポート先のテーブルにレコードを追加する権限がありません
725	インポート先のテーブルでレコードを変更する権限がありません
726	インポートファイルのレコードの方がインポート先のテーブルのレコードよりも多くなっています。一部のレコードはインポートされません
727	インポート先のテーブルのレコードの方がインポートファイルのレコードよりも多くなっています。一部のレコードは更新されません
729	インポート中にエラーが発生しました。レコードをインポートすることができません
730	サポートされていない Excel のバージョンです。ファイルを Excel 7.0（Excel 95）、Excel 97、2000、または XP のフォーマットに変換して、もう一度実行してください
731	インポート元のファイルにデータが含まれていません
732	このファイルには内部に他のファイルが含まれているため、挿入できません
733	テーブルをテーブル自体にインポートすることはできません
734	このファイルタイプをピクチャとして表示することはできません
735	このファイルタイプをピクチャとして表示することはできません。ファイルとして挿入および表示されます
800	ファイルをディスク上に作成できません
801	システムディスクにテンポラリファイルを作成できません
802	ファイルを開くことができません
803	ファイルが単独使用に設定されているか、またはホストが見つかりません
804	ファイルは現在の状態では読み取り専用として開くことができません
805	ファイルが損傷しています。修復コマンドを使用してください
806	このバージョンの FileMaker Pro ではファイルを開くことができません
807	ファイルが FileMaker Pro のファイルではないか、重大な損傷があります
808	アクセス権情報が壊れているため、ファイルを開くことができません

エラー番号	説明
809	ディスク/ボリュームがいっぱいです
810	ディスク/ボリュームがロックされています
811	テンポラリファイルを FileMaker Pro ファイルとして開くことができません
813	ネットワーク上でレコードの同期エラーが発生しました
814	最大数のファイルがすでに開いているため、ファイルを開くことができません
815	ルックアップファイルを開くことができません
816	ファイルを変換できません
817	このソリューションに属していないため、ファイルを開くことができません
819	リモートファイルのローカルコピーを保存できません
820	ファイルを閉じる途中です
821	ホストによって接続解除されました
822	FMI ファイルが見つかりません。見つからないファイルを再インストールしてください
823	ファイルをシングルユーザに設定できません。ゲストが接続しています
824	ファイルが損傷しているか、FileMaker のファイルではありません
900	スペルチェックのエンジンにエラーが発生しています
901	スペルチェック用のメイン辞書がインストールされていません
902	ヘルプシステムを起動できませんでした
903	共有ファイルではコマンドを使用できません
904	コマンドは、FileMaker Server がホスト管理しているファイル内でのみ使用できます
905	アクティブなフィールドが選択されていません。アクティブなフィールドが存在する場合のみコマンドを使用することができます
920	スペルチェックエンジンを初期化できません
921	編集するユーザ辞書をロードできません
922	ユーザ辞書が見つかりません
923	ユーザ辞書が読み取り専用です
951	予期しないエラーが発生しました
954	サポートされていない XML 文法です
955	データベース名がありません
956	データベースセッションが最大数を超過しました
957	コマンドが競合しています
958	クエリーに引数がありません
1200	一般的な計算エラーです
1201	関数の引数が足りません
1202	関数の引数が多すぎます
1203	計算式が未完了です
1204	数字、テキスト、フィールド名、または「(」を入れてください
1205	コメントは「*/」で終了できません
1206	テキストは半角のダブルクォーテーションマークで終わらなければなりません
1207	カッコが一致していません
1208	演算子または関数が見つからないか、「(」は指定できません
1209	名前（フィールド名またはレイアウト名）が見つかりません

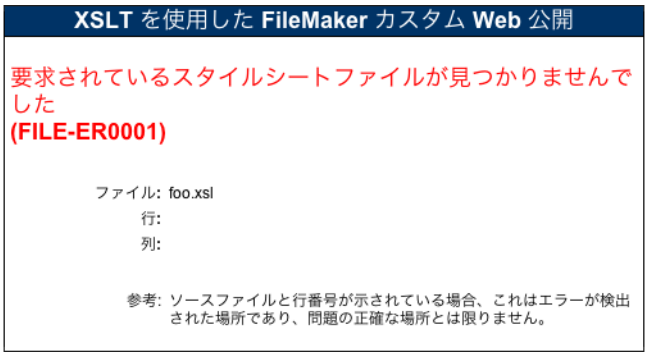
エラー番号	説明
1210	プラグイン関数はすでに登録されています
1211	この関数では一覧を使用できません
1212	演算子（+、-、* など）を入れてください
1213	この変数はすでに Let 関数で定義されています
1214	AVERAGE、COUNT、EXTEND、GETREPETITION、MAX、MIN、NPV、STDEV、SUM、および GETSUMMARY 関数で、フィールドの値を指定できない部分に式が使われています
1215	この引数は Get 関数の無効な引数です
1216	GetSummary 関数の 1 番目の引数は、集計フィールドのみに限られます
1217	区分けフィールドが無効です
1218	数字を評価できません
1219	フィールド固有の式にフィールドは使用できません
1220	フィールドタイプは標準にするか、計算する必要があります
1221	データタイプは数字、日付、時刻、またはタイムスタンプでなければなりません
1222	計算式を保存できません
1223	指定された関数は存在しません
1400	ODBC クライアントドライバの初期化に失敗しました。ODBC クライアントドライバが適切にインストールされていることを確認してください。 注意　ODBC 経由でデータを共有するプラグインコンポーネントは、FileMaker Server によって自動的にインストールされます。ODBC クライアントドライバは、FileMaker Server Web Publishing CD を使用してインストールされます。詳細については、『FileMaker ODBC および JDBC クライアントドライバのインストール』を参照してください。
1401	環境の割り当てに失敗しました（ODBC）
1402	環境の解放に失敗しました（ODBC）
1403	切断に失敗しました（ODBC）
1404	接続の割り当てに失敗しました（ODBC）
1405	接続の解放に失敗しました（ODBC）
1406	SQL API のチェックに失敗しました（ODBC）
1407	ステートメントの割り当てに失敗しました（ODBC）
1408	拡張エラー（ODBC）

Web 公開エンジンのエラーコード番号

Web 公開エンジンが開発モードの場合に、Web 公開エンジン自体でエラーが発生したときは、特定のエラーページが生成されます。このタイプのエラーは、次のように Web 公開エンジンが処理を実行できない場合など、さまざまな原因から発生することがあります。

- 要求されたスタイルシートファイル、または <xsl:include> によってネストされたスタイルシートファイルが見つからない
- ファイルに XML エラーがあるため、要求されたスタイルシートファイルまたはネストされたスタイルシートファイルを解析できない
- ファイルの XSLT または XPath エラーのため、ファイルからスタイルシートを生成できない
- CGI で XML 文法が正しく指定されていないため、リクエストを処理できない
- Web 公開コアと通信して XML を取得できない

Web 公開エンジンが開発モードで動作している場合は、このタイプのエラーのエラーページには、エラーメッセージと、カッコで囲まれたエラー番号が含まれます。次に例を示します。



Web 公開エンジンが開発モードの場合のエラーページの例

次に、Web 公開エンジンの一部のエラーコードの値を示します。

エラーコード	説明
QUERY-ER0001	「-grammar」クエリー引数で XML 文法が指定されていませんでした
QUERY-ER0002	「xxx」は FileMaker XSLT で有効な XML 文法ではありません
FILE-ER0001	要求されているスタイルシートファイルが見つかりませんでした
FILE-ER0002	要求されているファイルが見つかりませんでした
UNKNOWN	予期せぬエラーが発生しました
MCS-000 から MCS-600	予期せぬエラーが発生しました
MCS-601	タイプ「x」のリソースはサポートされていないため、リソース「x」をロードできませんでした
MCS-602	URL「x」を解決できませんでした
MCS-603	「x」に対する HTTP リクエストがタイプ「x」のエラーを返しました
MCS-604	予期せぬエラーのため、リソース「x」をロードできませんでした
MCS-605	content-type が無効なため、リソース「x」をロードできませんでした
MCS-606	ドキュメントの XML エラーのため、リソース「x」をロードできませんでした
MCS-607	認証の問題のため、リソース「x」をロードできませんでした
MCS-700	予期せぬエラーが発生しました
MCS-800	予期せぬエラーが発生しました

Web 公開エンジンが実作業モードの場合は、Web 公開エンジンのエラーに対して、「pe_server_error.html」に記述された次のデフォルトの一般的なテキストメッセージが表示されます。

FileMaker カスタム Web 公開を XSLT とともに使用中に、予期せぬエラーが発生しました。

デフォルトの「pe_server_error.html」ファイルには、このテキストメッセージが 6 つの言語で記述されています。

開発者は、ソリューションの必要に応じて、「pe_server_error.html」エラーページのテキストを編集できます。「pe_server_error.html」ファイルは、Web 公開エンジンをインストールしたホストの「publishing-engine」フォルダ内の「cwpe」フォルダにあります。

開発モードまたは実作業モードでの Web 公開エンジンの設定方法の詳細については、『FileMaker Server Advanced Web 公開インストールガイド』を参照してください。

FileMaker XSLT 拡張関数のエラーコード番号

fmxslt:check_error_status() 拡張関数（67 ページの「拡張関数のエラーステータスのチェック」を参照）は、次の表に示すエラーの 1 つを返します。

エラーコード	説明
-1	原因不明のエラー
0	エラーなし
	一般的なエラー
10000	無効なヘッダ名
10001	無効な HTTP ステータスコード
	セッションエラー
10100	原因不明のセッションエラー
10101	要求されたセッション名はすでに使用されています
10102	セッションにアクセスできませんでした。存在しない可能性があります
10103	セッションがタイムアウトしました
10104	指定されたセッションオブジェクトは存在しません
	メッセージングエラー
10200	原因不明のメッセージングエラー
10201	メッセージの書式設定エラー
10202	メッセージの SMTP フィールドのエラー
10203	メッセージの「To」フィールドのエラー
10204	メッセージの「From」フィールドのエラー
10205	メッセージの「CC」フィールドのエラー
10206	メッセージの「BCC」フィールドのエラー
10207	メッセージの「Subject」フィールドのエラー
10208	メッセージの「Reply-To」フィールドのエラー
10209	メッセージの本文のエラー
10210	再帰メールエラー - 電子メール用の XSLT スタイルシート内で send_email() を呼び出そうとしました
10211	SMTP 認証エラー - ログインに失敗したか、間違ったタイプの認証が指定されています
10212	無効な関数の使用法 - 電子メール用の XSLT スタイルシート内で set_header()、set_status_code()、または set_cookie() を呼び出そうとしました
10213	SMTP サーバーが無効か、動作していません。
	書式設定エラー
10300	原因不明の書式設定エラー
10301	無効な日付または時刻書式
10302	無効な日付書式
10303	無効な時刻書式
10304	無効な曜日書式
10305	不適切な形式の日付または時刻文字列
10306	不適切な形式の日付文字列

エラーコード	説明
10307	不適切な形式の時刻文字列
10308	不適切な形式の曜日文字列
10309	サポートされていないテキストエンコード
10310	無効な URL エンコード
10311	正規表現パターンのエラー

付録 C
CDML ソリューションの FileMaker XSLT への変換

この付録では、FileMaker CDML Converter を使用して CDML フォーマットファイルを FileMaker XSLT スタイルシートに変換した結果について説明します。FileMaker CDML Converter の使用の詳細については、42 ページの「FileMaker CDML Converter の使用」を参照してください。

注意　この付録、および変換された XSLT スタイルシート内に生成されるコメントでは、「XSLT-CWP」という用語は、XSLT を使用した FileMaker カスタム Web 公開を指しています。

CDML ソリューションの FileMaker XSLT ソリューションへの変換の処理について

CDML Converter は、変換元のフォルダにある CDML ソリューションファイルを次のように変換、名前変更、またはコピーします。

- CDML タグまたはクエリー引数が含まれる CDML フォーマットファイルは、XSLT スタイルシートに変換されて、変換先のフォルダに保存されます。
- HTML ファイルは、CDML タグが含まれるかどうかに関係なく、ファイル拡張子が .xsl に変更され、変換先のフォルダにコピーされます。たとえば、「myfile.html」の名前は「myfile.xsl」に変更されます。
- 変換されたファイル参照ではすべて小文字が使用され、XSLT では大文字と小文字が区別されるため、大文字のファイル名はすべて小文字に変更されます。
- CDML タグまたはクエリー引数が含まれない HTML 以外のファイルは、変更されずに変換先のフォルダにコピーされます。
- 変換元のフォルダ階層内にあるすべてのフォルダは、変換先のフォルダ内に自動的に作成されます。

各 CDML フォーマットファイルを XSLT スタイルシートに変換するために、CDML Converter によって次の処理が行われます。

- <!DOCTYPE> タグが含まれる場合は削除されます。
- CDML のすべての論理式は XPath 式にマップされます。
- -format CDML タグが記述されているすべての部分で、ファイル名参照のファイル拡張子が .xsl に変換されます。
- CDML のすべての置換タグとイントラタグ引数が XSLT-CWP ステートメントにマップされます。
- 変換されたすべてのスタイルシートに　<?xslt-cwp-query　params="-grammar=fmresultset"?>　処理命令が挿入され、fmresultset 文法が指定されます。50 ページの「静的に定義されたクエリーコマンドとクエリー引数の使用」を参照してください。
- <?xml ...?> 処理命令および <xsl:output> エレメントに、変換時に CDML Converter で選択したテキストエンコードオプションに一致する encoding 属性の値が挿入されます。
- 入力ファイルが <?xml ... ?> 処理命令から始まる場合は、<xsl:output> エレメントの method 属性に値「xml」が挿入されます。入力ファイルが HTML ファイルの場合は method 属性の値として「html」が挿入され、その他の場合は「text」が挿入されます。
- スタイルシートで使用されるネームスペース、XSLT 引数、および変数に対して、次の宣言が挿入されます。これらの宣言は、変換された XSLT-CWP ステートメントの前後に挿入されます。

  <?xml version="1.0" encoding="Shift_JIS"?>

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"
    xmlns:fml="http://www.filemaker.com/fmpxmllayout"
    xmlns:fmq="http://www.filemaker.com/xml/query"
    xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"
    xmlns:xalan="http://xml.apache.org/xalan"
    exclude-result-prefixes="xsl fmrs fml fmq fmxslt xalan">
    <xsl:param name="authenticated-xml-base-uri"/>
    <xsl:param name="request-query"/>
    <xsl:variable name="layout" select="document(concat($authenticated-xml-base-uri,
    '/fmi/xml/FMPXMLLAYOUT.xml?','-db=', /fmrs:fmresultset/fmrs:database/@name,'&amp;
    -lay=', /fmrs:fmresultset/fmrs:database/@layout, '&amp;-view='))"/>
    <xsl:variable name="current-find"><xsl:call-template name="get-current-find"/></xsl:variable>
    <xsl:variable name="current-sort"><xsl:call-template name="get-current-sort"/></xsl:variable>
    <xsl:variable name="current-action"><xsl:call-template name="get-current-action"/></xsl:variable>
    <xsl:variable name="current-lop"><xsl:call-template name="get-current-lop"/></xsl:variable>
    <xsl:variable name="current-max"><xsl:call-template name="get-current-max"/></xsl:variable>
    <xsl:variable name="current-skip"><xsl:call-template name="get-current-skip"/></xsl:variable>
    <xsl:include href="cdml2xsl_utilities.xsl"/>
    <xsl:output method="html" encoding="Shift_JIS"/>
    <xsl:template match="/fmrs:fmresultset">
        ... CDML/HTML から変換された XSLT-CWP ステートメントがここに挿入されます...
    </xsl:template>
</xsl:stylesheet>
```

XML および HTML ファイルを含むすべての SGML（Standard Generalized Markup Language）入力ファイルでは、CDML Converter により、必要に応じて改行が挿入され、ネストされたエレメントがインデントされます。また、CDML Converter によって次の処理が行われ、ファイルは整形式の XML ドキュメントに変換されます。

- 終了タグが見つからない場合は、親エレメントの終了タグの直前に挿入されます。ただし、空の HTML エレメントは例外で、HTML タイプのドキュメントに記述されているとおりに終了されます。これらの HTML エレメントには、<area />、<base />、<basefont />、
、<col />、<frame />、<hr />、<img />、<input />、<isindex />、<link />、<meta />、および <param /> があります。
- すべての終了タグが正しくネストされるように修正されます。
- すべてのエレメント名と属性名は小文字に変換されますが、属性値は変更されません。
- すべての属性値がダブルクォーテーションマークで囲まれていることが確認されます。
- 属性値が指定されていない場合は、属性名="属性名" のように、属性値に属性名が割り当てられます。

CDML のアクションタグ、変数タグ、および URL の変換

CDML のアクションタグ、変数タグ、および URL を変換するために、CDML Converter によって次の処理が行われます。

- CDML Converter では、先頭にダッシュ（-）の付いた識別されないタグ名は変更されません。URL 内の識別されないタグ名は、クエリー文字列の末尾に移動されます。CDML Converter では、先頭にダッシュの付いていないタグ名は、<フィールド名> 変数であると想定され、URL 内の適切な位置に残されます。

- URL を識別するために、CDML Converter によってテキスト文字列「fmpro?」が検索されます。SGML 以外のドキュメントと SGML ドキュメントでは、すべての場所にある URL が識別されます。SGML エレメント内では、エレメントの属性値内にある URL のみが識別されます。
- CDML Converter では、URL の外部にある -format 変数タグは、HTML ドキュメントの input エレメント内にある場合にのみ識別されます。select や textarea エレメントなどの他のフォームエレメントは、すべて無視されます。
- CDML の -format タグが含まれるすべての <input> エレメントは、XSLT-CWP ステートメントに置換されます。たとえば、次の <input> エレメントと -format タグがあるとします。

  <input type="hidden" name="–format" value="results.htm">

  これらは、次のように置換されます。

  <xsl:attribute name="action">results.xsl</xsl:attribute>

- リクエスト内の URL および <form> の <input> エレメントは、XSLT 用の新しい構文に変換されます。48 ページの「FileMaker XSLT スタイルシートの URL 構文について」を参照してください。
- -format で -fmp_xml または -dso_xml の値が使用されている、FileMaker XML データを要求する URL は、XML データ用の新しい URL 構文に変換されます。23 ページの「XML データとオブジェクトにアクセスするための URL 構文について」を参照してください。
- -img を使用して FileMaker イメージデータを要求する URL は、XSLT ソリューション内のオブジェクトを要求するための URL 構文に変換されます。48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。クエリーリクエストに img-key のフォーマットが含まれる場合は、[FMP-Image] 変数タグの変換と同じ方法で変換されます。たとえば、次の CDML リクエストがあるとします。

  <img src="fmpro?–db=employees&key=[FMP-Field: fieldname, url]&–img=">

  このリクエストは、次のタグと同じ方法で変換されます。

  <img src="[FMP-Image:fieldname]">

[FMP-Image] の変換の詳細については、126 ページの「CDML タグ名: 画像」を参照してください。

- タグ名は、XSLT-CWP の小文字のクエリーコマンドと引数名に変換されます。ただし、検索条件内の <フィールド名> CDML 変数タグは除きます。クエリーコマンドと引数の値は変更されません。
- URL に含まれる変換された XSLT-CWP クエリーコマンドと引数は、-db、-grammar、-lay、他のクエリー引数、クエリーコマンド（-findall など）の順序に並べ替えられます。クエリーコマンドと引数が含まれる <form> の <input> エレメントの順序は変更されません。
- CDML リクエストで -find または -findall クエリーコマンドに -max クエリー引数が指定されていない場合は、変換された XSLT リクエストに -max=25 が追加されます。CDML Converter では、25 という値を使用して、-max クエリー引数が指定されていない場合は 25 レコードが返される元の CDML リクエストと同じ動作を実現します。この動作は XML または XSLT リクエストでは異なっており、XML または XSLT リクエストで -max クエリー引数が指定されていない場合は、すべてのレコードが返されます。84 ページの「-max（最大レコード）クエリー引数」を参照してください。
- 変換エラーが発生した場合は、変換されたスタイルシートのエラーが発生したエレメントの直後にエラーコメントが挿入されます。スタイルシートでエラーを手動で変更する必要があります。エラーコメントの形式は次のとおりです。

  <!-- CDML Converter エラー：< エラーの説明 > -->

- 変換に関する警告が発生した場合は、変換されたスタイルシートの警告が発生したエレメントの直後に警告コメントが挿入されます。スタイルシートで警告を修正する必要がありませんが、他のエラーを防止するために、警告を調べることをお勧めします。警告コメントの形式は次のとおりです。

  <!-- CDML Converter 警告 : < 警告の説明 > -->

- 使用されなくなった CDML のアクションタグおよび変数タグは、URL から削除され、スタイルシート内の削除された場所にエラーコメントが挿入されます。「使用されなくなった CDML アクションタグの変換」および「使用されなくなった CDML 変数タグの変換」を参照してください。

- CDML Converter では、使用されなくなった -mail 変数タグ（-mailto や -mailfrom など）は変換されません。103 ページの「使用されなくなった CDML 変数タグの変換」を参照してください。-mail 変数タグで提供されていた電子メール機能を置き換えるには、fmxslt:send_email () 拡張関数を使用します。59 ページの「Web 公開エンジンからの電子メールメッセージの送信」を参照してください。
- fmpro で終わるアクション引数が指定された <form> エレメントに -format <input> エレメントが含まれない場合は、次のエラーコメントが挿入されます。

  <!-- CDML Converter エラー : 引数「-format」は見つかりません -->

CDML の -error および -errornum 変数タグの変換

CDML の -error および -errornum タグは、CDML Converter によってトークンの値に変換されます。たとえば、次のような元の URL があるとします。

fmpro?–db=employees&–format=format.htm&–error=error.htm&–errnum=401&–view

この場合、-error タグは -token.error に変換され、-errnum タグは -token.errnum に変換されます。変換された XSLT-CWP URL は、次のようになります。

format.xsl?–db=employees&–token.error=error.xsl&–token.errnum=401&–view

変換された「format.xsl」スタイルシートには、次のステートメントが含まれます。

```
<xsl:include href="cdml2xsl_includes.xsl"/>
<xsl:variable name="_errorcode" select="/fmrs:fmresultset/fmrs:error/@code"/>
<xsl:variable name="token.error" select="$request-query/fmq:query/fmq:parameter[@name = '-token.error']"/>
<xsl:variable name="token.errnum" select="$request-query/fmq:query/fmq:parameter[@name = '-token.errnum']"/>
<xsl:choose>
    <xsl:when test="$token.error != '' and ($_errorcode = $token.errnum)">
        <xsl:call-template name="error"/>
    </xsl:when>
    <xsl:otherwise>
        [... 変換された「format.xsl」のコードがここに記述されます ...]
    </xsl:otherwise>
</xsl:choose>
```

「error.html」ファイルは、次の名前付きのテンプレートが含まれる「error.xsl」スタイルシートに変換されます。

```
<xsl:template name="err">
    [... 変換された「error.xsl」のコードがここに記述されます ... ]
</xsl:template>
```

すべてのエラーファイルは、CDML Converter によって、「cdml2xsl_includes.xsl」という名前のグローバルエラーファイルにまとめられます。次のステートメントにより、エラー処理を行うすべてのスタイルシートにグローバルエラーファイルがインクルードされます。

```
<xsl:include href="error.xsl"/>
...
```

「format.xsl」スタイルシートの処理時にエラーが発生し、そのエラーコードが -token.errnum クエリー引数に指定されたエラー番号と一致する場合、「format.xsl」が error という名前のテンプレートを呼び出したときに、「error.xsl」からのコードが実行されます。この動作は、元の CDML の -error および -errornum の機能と一致します。

使用されなくなった CDML アクションタグの変換

次に、使用されなくなった CDML アクションタグが CDML Converter によってどのように変換されるかを示します。

使用されなくなった CDML アクションタグ	処理方法
-dbclose	次のコメントによって置換されます。このコメントは、元のアクションタグがあった場所の後に挿入されます。 <!-- CDML Converter エラー：「-dbclose」は XSLT-CWP ではサポートされていません。-->
-dbopen	次のコメントによって置換されます。このコメントは、元のアクションタグがあった場所の後に挿入されます。 <!-- CDML Converter エラー：「-dbopen」は XSLT-CWP ではサポートされていません。-->
-img	削除され、指定されたイメージの URL は、オブジェクトにアクセスするための新しい URL 構文に変換されます。「XML ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」および 48 ページの「XSLT ソリューション内の FileMaker オブジェクトにアクセスするための URL 構文について」を参照してください。

サポートされている CDML アクションタグの変換

次に、サポートされている CDML アクションタグが CDML Converter によってどのように変換されるかを示します。

CDML アクションタグ	変換後の XSLT-CWP クエリー引数（構文の変更なし）
-delete	-delete
-dup	-dup
-edit	-edit
-find	-find
-findall	-findall
-findany	-findany
-new	-new
-view	-view
-dbnames	-dbnames
-layoutnames	-layoutnames
-scriptnames	-scriptnames

使用されなくなった CDML 変数タグの変換

次に、使用されなくなった CDML 変数タグが CDML Converter によってどのように変換されるかを示します。

使用されなくなった CDML 変数タグ	元の変数タグがあった場所の後に置換されるコメント
–errorfmtfield	<!-- CDML Converter エラー：「–errorfmtfield」は XSLT-CWP ではサポートされていません。-->
-fmtfield	<!-- CDML Converter エラー：「-fmtfield」は XSLT-CWP ではサポートされていません。-->
-format	コメントは挿入されません。代わりに、フォーマットファイルへの URL は、変換された XSLT スタイルシートの名前を参照する新しい URL 構文に変換されます。
-mailbcc	<!-- CDML Converter エラー：「-mailbto」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
-mailcc	<!-- CDML Converter エラー：「-mailcc」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
–mailfmtfield	<!-- CDML Converter エラー：「-mailfmtfield」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->

使用されなくなった CDML 変数タグ	元の変数タグがあった場所の後に置換されるコメント
-mailformat	<!-- CDML Converter エラー：「-mailformat」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
-mailfrom	<!-- CDML Converter エラー：「-mailformat」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
-mailhost	<!-- CDML Converter エラー：「-mailhost」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
-mailsub	<!-- CDML Converter エラー：「-mailsub」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
-mailto	<!-- CDML Converter エラー：「-mailbto」は XSLT-CWP ではサポートされていません。fmxslt:send_email() 関数を使用してください。-->
-password	<!-- CDML Converter エラー：「-password」は XSLT-CWP ではサポートされていません。-->

サポートされている CDML 変数タグの変換

注意　次に、サポートされている CDML 変数タグが CDML Converter によってどのように変換されるかを示します。

CDML 変数タグ	変換後の XSLT-CWP クエリー引数	説明
-db	-db	構文は変更されません。指定されているファイル拡張子 .fp5 はすべて削除されます。
–lay	–lay	構文は変更されません。
–lop	–lop	構文は変更されません。「and」および「or」キーワードは小文字に変更されます。
–max	–max	構文は変更されません。CDML で -max が指定されていなかった場合は、変換された XSLT に -max=25 が挿入されます。
–modid	–modid	構文は変更されません。
–op	フィールド.op	検索演算子の新しい構文（フィールド.op）をサポートするように変更されます。82 ページの「フィールド名.op（比較演算子）クエリー引数」を参照してください。
–recid	–recid	構文は変更されません。
-script	-script	エレメントの後に、コメント <!-- CDML2XSLT 警告:please verify '-script' functionality with XSLT-CWP --> が追加されます。
–script.prefind	–script.prefind	構文は変更されません。XSLT-CWP で「-script.prefind」の機能を確認してください。
–script.presort	–script.presort	構文は変更されません。XSLT-CWP で「-script.presort」の機能を確認してください。
–skip	–skip	構文は変更されません。
–sortfield	–sortfield	ソート演算子の新しい構文（-sortfield.優先順位）をサポートするように変更されます。86 ページの「-sortfield（ソートフィールド）クエリー引数」を参照してください。
–sortorder	–sortorder	ソート演算子の新しい構文（-sortorder.優先順位）をサポートするように変更されます。86 ページの「-sortorder（ソート順）クエリー引数」を参照してください。
–styletype	–styletype	構文は変更されません。
–stylehref	–stylehref	構文は変更されません。
-token	-token	構文は変更されません。
-token.番号	-token.番号	構文は変更されません。
<フィールド名>	<フィールド名>	構文は変更されません。

CDML 論理値引数の XPath 論理値引数への変換

CDML 論理式は、[FMP-If] および [FMP-ElseIf] イントラタグ引数内で使用され、引数と演算子で構成されます。これらの引数と演算子は、CDML Converter によって XPath 式に変換されます。

CDML 論理値引数	変換後の XPath 論理値引数	コメント
CanDelete	false()	セキュリティ上の理由から、この引数は false() に変換されますが、サポートされません。タグの後に、コメント<!-- CDML Converter エラー：「CanDelete」は XSLT-CWP ではサポートされていません。--> が追加されます。
CanEdit	false()	セキュリティ上の理由から、この引数は false() に変換されますが、サポートされません。タグの後に、コメント<!-- CDML Converter エラー：「CanEdit'」は XSLT-CWP ではサポートされていません。--> が追加されます。
CanNew	false()	セキュリティ上の理由から、この引数は false() に変換されますが、サポートされません。タグの後に、コメント<!-- CDML Converter エラー：「CanNew」は XSLT-CWP ではサポートされていません。--> が追加されます。
IsSorted	boolean($request–query/fmq:query/fmq:parameter[starts-with(@name, '–sortfield')]) = true())	
True	true()	
False	false()	
Checked	false()	
文字列	'文字列'	
数字	数字	
<見つからない引数>	''	[FMP-If xyz .eq.]...[/FMP-If] は、次のように変更されます。 <xsl:choose><xsl:when test="'xyz' = ''"> ...</xsl:when></xsl:choose>

注意　論理式でイントラタグ引数 CurrentDate、CurrentDay、または CurrentTime が使用されている場合は、CDML Converter によって、XSLT-CWP の適切な日付比較拡張関数（fmxslt:compare_date () など）に置き換えられます。64 ページの「日付、時刻、および曜日拡張関数の使用」を参照してください。

CDML 論理演算子の XPath への変換

CDML 論理演算子	変換後の XPath 論理演算子
x .and. y	x and y
x .or. y	x or y
x .xor. y	(x or y) and not (x and y)
x .eq. y	x = y
x .neq. y	x != y
x .gt. y	x > y
x .gte. y	x >= y
x .lt. y	x < y

CDML 論理演算子	変換後の XPath 論理演算子
x .lte. y	x <= y
x .cn. y	contains(x, y)
x .ncn. y	not (contains(x, y))

CDML イントラタグ引数の XSLT-CWP への変換

CDML イントラタグ引数は、別の置換タグ内で使用されている CDML の置換タグです。イントラタグ引数が引数値として使用されている場合、中カッコで囲まれています。次に例を示します。

-db={CurrentDatabase}

これは、次のタグの代わりに使用されます。

[FMP-CurrentDatabase]

イントラタグ引数は、CDML Converter によって、置換タグの変換と同じ方法で変換されます。110 ページの「CDML 置換タグの XSLT-CWP への変換」を参照してください。次に、引数値として使用されているイントラタグ引数が CDML Converter によってどのように変換されるかの例を示します。

引数値（中カッコ）内にあるイントラタグ引数の変換例

元の **CDML:**	<!-- Log page hits in another database --> [FMP-InlineAction:-db=log.fp5, -lay=web, time="**{CurrentTime}**", date="**{CurrentDate}**", page="This page!", browser="**{ClientType}**", ip="**{ClientIP}**", -new] [/FMP-InlineRequest]
変換された **XSLT-CWP:**	<!-- Log page hits in another database --> <xsl:variable select="document(concat('/fmi/xml/fmresultset.xml?-db=', fmxslt:url_encode('log'), '&-lay=', fmxslt:url_encode('web'), '&time=', **fmxslt:url_encode(fmxslt:get_time())**, '&date=', **fmxslt:url_encode(fmxslt:get_date())**, '&page=', fmxslt:url_encode('This Page!'), '&browser=', **fmxslt:url_encode(fmxslt:get_header('User-Agent'))**, '&ip=', **fmxslt:url_encode($client-ip)**, '&-new='))" /> '&–new='))" />

CDML 変換エラーの手動での修正

CDML Converter が CDML から XSLT への正しい変換方法を自動的には判断できない場合があります。このような状況では、テキストエディタまたは XSLT スタイルシートエディタを使用して、変換された XSLT スタイルシートで、問題を手動で修正する必要があります。

- FileMaker Pro では、スクリプト名は最大 100 文字に制限されています。データベースの変換時に、この制限を超えるスクリプト名はデータベース内で切り詰められます。この制限を超えるスクリプト名が CDML Converter によって切り詰められることはないため、データベース内の切り詰められた名前と一致するように、XSLT スタイルシートでスクリプト名を変更する必要があります。
- CDML Converter では、Web 公開エンジンがインストールされたホストコンピュータで設定されているロケールにかかわらず、fmxslt:get_date() や fmxslt:get_time() などのサーバーサイドの関数を XSLT スタイルシートに挿入する際に、「fm」書式が使用されます。「fm」形式では、日付は MM/dd/yyyy、時刻は HH:mm:ss、タイムスタンプは MM/dd/yyyy HH:mm:ss です。64 ページの「日付、時刻、および曜日拡張関数の使用」を参照してください。CDML フォーマットファイルに、これらの関数に渡される日付および時刻書式の文字列が含まれる場合は、変換後にそれらの文字列を手動でローカライズする必要があります。たとえば、月/日/年の書式を年/月/日の書式に変更する必要があります。出力された値を他の好ましい書式に変更するには、計算関数または JapaScript を使用します。
- CDML ソリューションに、ユーザが選択可能な -format タグの値（フォーマットファイルの選択肢など）を提供するフォームが含まれる場合、CMDL Converter では、機能的に同等の XSLT ソリューションを自動的に作成することはできません。Web ユーザに提供する各 XSLT スタイルシートに対して別個のリクエストを使用して、変換された XSLT ソリューションを手動で編集する必要があります。

- CDML ソリューションに、-lay タグを使用してレイアウトを指定するクエリーリクエストが含まれる場合は、指定されたレイアウトに、リクエストで参照されるフィールドがすべて含まれることを確認してください。指定されたレイアウトに一部のフィールドが含まれない場合は、フィールドをレイアウトに追加するか、またはすべてのフィールドが含まれるレイアウトを参照するように -lay クエリー引数を手動で変更する必要があります。または、他のレイアウトに存在しないフィールドが含まれるレイアウトを使用してリクエストを送信する場合は、-lay.response クエリー引数を使用して、レイアウトを切り替えることができます。34 ページの「XML 応答に対するレイアウトの切り替え」を参照してください。
- CDML ソリューションに、-lay タグを使用してレイアウトを指定していないクエリーリクエストが含まれる場合は、CDML Converter によって、リクエストに自動的に -lay クエリー引数が追加され、AllFieldsLayout の値が指定されます。使用するデータベース内のレイアウトに一致するように、変換されたスタイルシートの -lay 引数の値を手動で変更するか、または「AllFieldsLayout」という名前のレイアウトをデータベースに追加する必要があります。
- CDML ソリューションに、-script、-script.prefind、または -script.presort 変数タグが含まれる場合は、変換された XSLT スタイルシートでスクリプトの機能を確認してください。
- CDML では、フィールド名やデータベース名を比較する際に大文字と小文字は区別されないため、[FMP-Field: myfield] のようなタグを使用して、「MyField」または「myField」という名前のフィールドを参照できていました。XSLT-CWP では、クエリー文字列で使用されている場合以外は、フィールド名やデータベース名を比較する際に大文字と小文字が区別されます。変換されたスタイルシートで、大文字と小文字の区別など、データベースソリューションで使用されている名前に完全に一致するように、XSLT ステートメント内（クエリー文字列を除く）のフィールド名とデータベース名を手動で修正する必要があります。

  たとえば、次のステートメントがあるとします。

  <xsl:value-of select="fmrs:field[@name='LastName']"/>

  この場合、フィールド参照「LastName」は、データベースの「LastName」フィールドの名前、および大文字と小文字に完全に一致する必要があります。

  注意 XSLT-CWP では、クエリー文字列で使用されるフィールド名とデータベース名の大文字と小文字が区別されます。76 ページの「クエリーコマンドと引数の使用のガイドライン」を参照してください。

- CDML では、フィールド属性を含めずにフィールドを比較できます。たとえば、[FMP-If: myfield.eq.10] または [FMP-If:field:myfield.eq.10] のどちらも使用できます。この例では、フィールド属性は比較に含まれないため、CDML Converter によって、myfield は、フィールド名ではなくテキスト文字列として変換されます。

  たとえば、次の CDML ステートメントがあるとします。

  [FMP-If:myfield.eq.10]

  このステートメントは、変換後には次の XSLT-CWP ステートメントに変換されます。

```
<xsl:choose>
    <xsl:when test="'myfield' = '10'">Ten</xsl:when>
</xsl:choose>
```

このような問題を修正するには、変換されたスタイルシートで、比較ステートメントに適切なフィールド名が設定されるように、ステートメントを手動で修正する必要があります。または、必要に応じて、CDML ファイルに「field:」を追加して、ファイルを再変換することもできます。

- クラリスホームページで生成される不正な形式の HTML の多くは、CDML Converter によって、HTML ページの先頭にメタタグが含まれると想定して修正されます。メタタグが削除されている場合は、不正な形式の HTML は修正されません。CDML Converter では修正できず、HTML よりも厳密な構造の XHTML に正しく変換できない、クラリスホームページの不正な形式の HTML や、それ以外の HTML ファイルは、他にも存在する可能性があります。不正な形式の HTML が正しく変換されなかった場合は、変換されたスタイルシートで XHTML を手動で修正する必要があります。
- すべてのイメージのファイル名参照には、-img アクションタグを変換する際に、CDML Converter によってファイル拡張子 .jpg が追加されます。たとえば、次の CDML リクエストがあるとします。

  /fmpro?–db=products.fp5&–format=format_file.html&–lay=sales&–recid=123&–img

このリクエストは、次の XSLT-CWP リクエストに変換されます。

/fmi/xsl/data.jpg?–db=products&–lay=sales&–recid=123

ファイル拡張子 .jpg がソリューションに対して正しくない場合は、変換されたスタイルシートのファイル名参照の拡張子を手動で変更する必要があります。

- CDML Converter では、変換された（埋め込み）フォームは変換できません。フォームをネストする場合は、CDML ソリューションを変更するか、または生成された .xsl ファイルを修正する必要があります。
- CDML Converter では、SGML タグ内に埋め込まれた CDML タグは変換できません。次に例を示します。

```
<input type="text" NAME="StateProvince" SIZE="16" [FMP-If:currentdatabase.eq.Customers.fp5]
VALUE="[FMP-Field:StateProvince]" [/FMP-If]>
        <SELECT NAME="–max">
            <OPTION [FMP-If:currentmax.eq.5] SELECTED[/FMP-If]>5
            <OPTION [FMP-If:currentmax.eq.10] SELECTED[/FMP-If]>10
        </SELECT>
```

この例から発生する変換の問題を避けるには、CDML のソースコードを次のように変更します。

```
[FMP-If:currentdatabase.eq.Customers.fp5]
        <input type="text" NAME="StateProvince" SIZE="16" VALUE="[FMP-Field:StateProvince]">
[FMP-Else]
        <input type="text" NAME="StateProvince" SIZE="16" VALUE="">
[/FMP-If]
        <SELECT NAME="–max">
            [FMP-If:currentmax.eq.5]
                <OPTION SELECTED>5 </OPTION>
            [FMP-Else]
                <OPTION>5 </OPTION>
            [FMP-If:currentmax.eq.10]
                <OPTION SELECTED>10 </OPTION>
            [FMP-Else]
                <OPTION>10 </OPTION>
            [/FMP-If]
        </SELECT>
```

- フォーム内の -format タグは CDML Converter によって削除され、その値は form タグの属性に変換されます。その結果、変換されたスタイルシートでは、フォームの入力引数が 1 つ少なくなります。CDML ソリューションで JavaScript を使用してフォームのフィールドを参照している場合は、これによって問題が発生することがあります。

  たとえば、次のフォームがあるとします。

```
<FORM METHOD="POST" ACTION="FMPro" NAME="checkoutform">
        <INPUT TYPE="hidden" NAME="–db" VALUE="Orders.FP5">
        <INPUT TYPE="hidden" NAME="–format" VALUE="thanks.htm">
        <INPUT TYPE="hidden" NAME="–lay" VALUE="CGI">
        <INPUT TYPE="text" SIZE="50" NAME="Account Number">
</FORM>
```

このフォームは、次の XSLT ステートメントに変換されます。

```
<form method="POST" name="checkoutform"><xsl:attribute name="action">thanks.xsl</xsl:attribute>
        <input type="hidden" name="–db" value="Orders"/>
        <input type="hidden" name="–lay" value="CGI"/>
        <input type="text" size="50" name="Account Number"/>
</form>
```

ここでの問題は、ユーザが「Account Number」フィールドの値を送信したことを確認するために、次の JavaScript を使用していることです。

```
<P>
<CENTER>
        <A HREF="javascript:document.checkoutform.elements[3].value == '' ?
        alert('You must fill out the \'Account number\' field!'): document.checkoutform.submit()">
        <IMG SRC="images/continue1.gif" NAME="cont" ALT="Continue"></A>
</CENTER>
</P>
```

この例の変換後は、-format タグが削除されるため、checkoutform のエレメントが少なくなります。この問題を修正するには、document.checkoutform.elements[3] ではなく、document.checkoutform.elements[2] を確認するように、手動で JavaScript を変更する必要があります。

- CDML における式の評価論理は、XSLT における論理とは異なります。たとえば、次の CDML の式では、userchoice が空の場合、「userchoice less than 1」が出力されます。

```
[fmp-if:currentcookie:userchoice .lt. 1 ]
        userchoice less than 1
[fmp-else]
        userchoice not less than 1
[/fmp-if]
```

XLST では、変換された同じ式により、userchoice が空の場合、「userchoice not less than 1」が出力されます。

```
<xsl:choose>
        <xsl:when test="fmxslt:get_cookie('userchoice') &lt; '1'">
                userchoice less than 1
        </xsl:when><xsl:otherwise>
                userchoice not less than 1
        </xsl:otherwise>
</xsl:choose>
```

この違いの理由は、CDML の式では数字の比較が実行されるのに対し、XSLT の式では文字列の比較が実行されるためです。XSLT で元の CDML の式と同じ結果を返すには、空の文字列が考慮されるように XSLT ステートメントを変更します。次に例を示します。

```
<xsl:variable name="userchoice" select="fmxslt:get_cookie('userchoice')"/>
<xsl:choose>
        <xsl:when test="string-length($userchoice) = 0 or $userchoice &lt; '1'">
                userchoice less than 1
        </xsl:when><xsl:otherwise>
                userchoice not less than 1
        </xsl:otherwise>
</xsl:choose>
```

CDML 置換タグの XSLT-CWP への変換

このセクションでは、すべての CDML 置換タグの XSLT-CWP ステートメントへの変換について説明します。すべての場合において、次のような処理が行われます。

- 変換された XSLT-CWP ステートメントでは、fmresultset 文法が使用されます。
- XSLT-CWP ステートメントが SGML エレメントの属性値内の値を返す場合は、{名前} の形式が使用されます。その他の場合は、<xsl:value-of select="名前" /> の形式が使用されます。
- CDML タグに エンコード引数が指定されている場合、変換された XSLT-CWP の値は、fmxslt:break_encode()、fmxslt:html_encode ()、または fmxslt:url_encode () のいずれかの関数を通じて渡されてエンコードされます。62 ページの「文字列操作拡張関数の使用」を参照してください。エンコードの値が「Raw」の場合、エンコードは指定されません。エンコードが「Raw」に設定されている場合は、CDML Converter によって、<xsl:value-of> の「disable-output-escaping」属性が設定され、<xsl:text> エレメントが「yes」に設定されます。
- HTML タイプのドキュメント内にある場合、fmxslt:html_encode () 拡張関数は使用されません。

CDML タグ名 : クライアントアドレス

このタグは、現在の Web クライアントの IP アドレスで置換されます。

CDML 構文: [FMP-ClientAddress]

XSLT-CWP への変換:

- SGML エレメントの属性値内にある場合: {$client-ip}
- その他の場合: <xsl:value-of select="$client-ip"/>

変換例

元の **CDML:**	アドレス : **[FMP-ClientAddress]**
変換された **XSLT-CWP:**	アドレス : **<xsl:value-of select="$client-ip"/>**
変換された結果 **:**	アドレス : **192.168.123.101**

CDML タグ名 : クライアント IP アドレス

このタグは、現在のクライアントの IP アドレスで置換されます。

CDML 構文: [FMP-ClientIP]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$client-ip}
- その他の場合: <xsl:value-of select="$client-ip"/>

変換例

元の **CDML:**	IP アドレス : **[FMP-ClientIP]**
変換された **XSLT-CWP:**	IP アドレス : **<xsl:value-of select="$client-ip"/>**
変換された結果 **:**	IP アドレス : **192.168.123.101**

CDML タグ名 : クライアントパスワード

このタグは、HTTP 認証済みの現在のクライアントパスワードで置換されます。

CDML 構文: [FMP-ClientPassword]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$client-password}
- その他の場合: <xsl:value-of select="$client-password"/>

変換例

元の **CDML:**	パスワード : **[FMP-ClientPassword]**
変換された **XSLT-CWP:**	パスワード : **<xsl:value-of select="$client-password"/>**
変換された結果 **:**	パスワード : **my-password**

CDML タグ名 : クライアントタイプ

このタグは、現在のブラウザのクライアントタイプで置換されます。

CDML 構文: [FMP-ClientType]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {fmxslt:get_header('User-Agent')}
- その他の場合: <xsl:value-of select="fmxslt:get_header('User-Agent')" />

変換例

元の **CDML:**	ブラウザタイプ : **[FMP-ClientType]**
変換された **XSLT-CWP:**	ブラウザタイプ : **<xsl:value-of select="fmxslt:get_header('User-Agent')" />**
変換された結果 **:**	ブラウザタイプ : **Mozilla/3.01 (Macintosh; I; PPC)**

CDML タグ名 : クライアントユーザ名

このタグは、HTTP 認証のクライアントユーザ名で置換されます。

CDML 構文: [FMP-ClientUserName]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$client-user-name}
- その他の場合: <xsl:value-of select="$client-user-name"/>

変換例

元の **CDML:**	名前 : **[FMP-ClientUserName]**
変換された **XSLT-CWP:**	名前 : **<xsl:value-of select="$client-user-name"/>**
変換された結果 **:**	名前 : **my-user-name**

CDML タグ名 : MIME タイプ

このタグは、HTML の内容では置換されず、ブラウザに返される MIME タイプを変更します。

CDML 構文: [FMP-ContentMimeType: MIME タイプ]

XSLT-CWP への変換

- <xsl:variable name="header1" select="fmxslt:set_header('Content-Type', 'MIME タイプ')" />
- SGML エレメントの属性値内にある場合: エレメントの外または後に挿入されます。

注意　変換後は fmxslt:set_header() 拡張関数が使用されるため、コンテンツバッファが有効になっている必要があります。コンテンツバッファを有効にするために、CDML Converter によって、自動的に <?xslt-cwp-buffer?> 処理命令が挿入されます。57 ページの「コンテンツバッファの使用」を参照してください。

変換例

元の **CDML:**	サンプルテキスト **[FMP-ContentMIMEType:text/plain]**
変換された **XSLT-CWP:**	サンプルテキスト **<xsl:variable name="header1" select="fmxslt:set_header('Content-Type', 'text/plain')" />**
変換された結果 **:**	サンプルテキスト （HTTP 応答のヘッダにも、**Content-Type:text/plain** が含まれます。）

CDML タグ名 : Cookie

このタグは、指定された Cookie の現在の値で置換されます。

CDML 構文: [FMP-Cookie: Cookie 名, エンコード]。エンコードは、「Raw」（デフォルト）または「URL」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {fmxslt:get_cookie('Cookie 名')}
- その他の場合: <xsl:value-of select="fmxslt:get_cookie('Cookie 名')" />

変換例

元の **CDML:**	最新の **[FMP-Cookie:ColorChoice]** の製品は以下のとおりです。
変換された **XSLT-CWP:**	最新の **<xsl:value-of select="fmxslt:get_cookie('ColorChoice')" />** の製品は以下のとおりです。
変換された結果 **:**	最新の **緑色** の製品は以下のとおりです。

CDML タグ名 : アクション

このタグは、現在のアクションの名前で置換されます。たとえば、find、findall、new、edit、delete、view、dup などで置換されます。返される名前には、ダッシュ（-）文字は含まれません。

CDML 構文: [FMP-CurrentAction:エンコード]。エンコードは、「HTML」（デフォルト）または「Display」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$current-action}
- その他の場合: <xsl:value-of select="$current-action" />
- 書式が「Display」の場合: <!-- CDML2XSLT 警告:[FMP-CurrentAction] 「Display」のエンコードは XSLT-CWP ではサポートされていません。-->
- ドキュメント内で使用されている場合: $current-action 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	最終アクション : **[FMP-CurrentAction]**
変換された **XSLT-CWP:**	最終アクション : **<xsl:value-of select="$current-action" />**
変換された結果 **:**	最終アクション : **view**

CDML タグ名 : データベース

このタグは、処理中のデータベース名で置換されます。

CDML 構文: [FMP-CurrentDatabase:エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {fmrs:fmresultset/fmrs:datasource/@database}
- その他の場合: <xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@database"/>

変換例

元の **CDML:**	データベース : **[FMP-CurrentDatabase]**
変換された **XSLT-CWP:**	データベース : **<xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@database" />**
変換された結果 :	データベース : **住所録**

CDML タグ名 : 日付

このタグは、現在の日付で置換されます。

CDML 構文: [FMP-CurrentDate: 書式]。書式は、「Short」（デフォルト）または「Long」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にあり、「Short」の場合: {fmxslt:get_date('short')}
- または「Long」の場合: {fmxslt:get_date('long')}
- その他の場所にあり、「Short」の場合: <xsl:value-of select="fmxslt:get_date('short')" />
- または「Long」の場合: <xsl:value-of select="fmxslt:get_date('long')" />
- 書式が「Abbrev」の場合: <!-- CDML CONVERTER 警告:短縮形の日付書式はサポートされなくなりました。追加の書式については、fmxslt:get_datetime() を使用してください。-->

変換例

元の **CDML:**	現在の日付 : **[FMP-CurrentDate]**
変換された **XSLT-CWP:**	現在の日付 : **<xsl:value-of select="fmxslt:get_date()" />**
変換された結果 :	現在の日付 : **04/2/2**

CDML タグ名 : 曜日

このタグは、現在の曜日で置換されます。

CDML 構文: [FMP-CurrentDay:書式]。書式は、「Short」（デフォルト）または「Long」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にあり、「Short」の場合: {fmxslt:get_day('short')}
- または「Long」の場合: {fmxslt:get_day('long')}
- その他の場所にあり、「Short」の場合: <xsl:value-of select="fmxslt:get_day('short')" />
- または「Long」の場合: <xsl:value-of select="fmxslt:get_day('long')" />

変換例

元の **CDML:**	現在の曜日 : **[FMP-CurrentDay:Long]**
変換された **XSLT-CWP:**	現在の曜日 : **<xsl:value-of select="fmxslt:get_day('long')" />**
変換された結果 **:**	現在の曜日 : **Monday**

CDML タグ名 : エラー

このタグは、現在の操作の FileMaker エラー番号で置換されます。

CDML 構文: [FMP-CurrentError]

XSLT-CWP への変換

SGML エレメントの属性値内にある場合: {fmrs:fmresultset:/fmrs:error/@code}

その他の場合: <xsl:value-of select="fmrs:fmresultset:/fmrs:error/@code" />

変換例

元の **CDML:**	操作はエラー番号 **[FMP-CurrentError]** で失敗しました。
変換された **XSLT-CWP:**	操作はエラー番号 **<xsl:value-of select="fmrs:fmresultset:/fmrs:error/@code" />** で失敗しました。
変換された結果 **:**	操作はエラー番号 **500** で失敗しました。

CDML タグ名 : 検索条件

ページを作成するリクエストの一部である各検索条件に対して、[FMP-CurrentFind] タグと [/FMP-CurrentFind] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-CurrentFind]...[/FMP-CurrentFind]

XSLT-CWP への変換

- <xsl:for-each select="$current-find">...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-CurrentFind] not in a valid location -->
- ドキュメント内で使用されている場合: $current-find 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	現在の検索条件 : **[FMP-CurrentFind]** Field:[FMP-FindFieldItem], Op:[FMP-FindOpItem], Value:[FMP-FindValueItem] [/FMP-CurrentFind]
変換された **XSLT-CWP:**	現在の検索条件 : **<xsl:for-each select="$current-find/find-field">** Field:<xsl:value-of select="@name" />, Op:<xsl:value-of select="@long-operator" />, Value:<xsl:value-of select="text()" /> </xsl:for-each>
変換された 結果 **:**	現在の検索条件 : Field:First Name, Op:begins with, Value:John Field:Last Name, Op:equals, Value:Doe

CDML タグ名 : フォーマットファイル

このタグは、現在処理中のフォーマットファイルの名前で置換されます。

CDML 構文: [FMP-CurrentFormat:エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$request-query/@action}
- その他の場合: <xsl:value-of select="$request-query/@action"/>

変換例

元の **CDML:**	フォーマットファイル : **[FMP-CurrentFormat]**.
変換された **XSLT-CWP:**	フォーマットファイル : **<xsl:value-of select="$request-query/@action"/>**.
変換された結果 **:**	フォーマットファイル : **Detail.xsl**.

CDML タグ名 : 対象レコード数

このタグは、現在の対象レコードの総数で置換されます。

CDML 構文: [FMP-CurrentFoundCount]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {fmrs:fmresultset/fmrs:resultset/@count}
- その他の場合: <xsl:value-of select="fmrs:fmresultset/fmrs:resultset/@count" />

変換例

元の **CDML:**	対象レコード数 : **[FMP-CurrentFoundCount]**
変換された **XSLT-CWP:**	対象レコード数 : **<xsl:value-of select="fmrs:fmresultset/fmrs:resultset/@count" />**
変換された結果 **:**	対象レコード数 : **12**

CDML タグ名 : レイアウト

このタグは、ページを処理するために使用されたレイアウトの名前に置換されます。

CDML 構文: [FMP-CurrentLayout:エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {fmrs:fmresultset/fmrs:datasource/@layout}
- その他の場合: <xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@layout" />

変換例

元の **CDML:**	レイアウト : **[FMP-CurrentLayout]** レイアウト
変換された **XSLT-CWP:**	レイアウト : **<xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@layout" />** レイアウト
変換された結果 **:**	レイアウト : **Detail** レイアウト

CDML タグ名 : 論理演算子

このタグは、現在の検索に使用されている論理演算子で置換されます。

CDML 構文: [FMP-CurrentLOP]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$current-lop}

- その他の場合: <xsl:value-of select="$current-lop" />
- ドキュメント内で使用されている場合: $current-lop 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	論理演算子 "**[FMP-CurrentLOP]**" 検索
変換された **XSLT-CWP:**	論理演算子 "**<xsl:value-of select="$current-lop" />**" 検索
変換された結果 **:**	論理演算子 "**or**" 検索

CDML タグ名 : 表示するレコードの最大数

このタグは、指定されたレコードの最大数で置換されます。

CDML 構文: [FMP-CurrentMax]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$current-max}
- その他の場合: <xsl:value-of select="$current-max" />
- ドキュメント内で使用されている場合: $current-max 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	次の **[FMP-CurrentMax]** レコードを表示
変換された **XSLT-CWP:**	次の **<xsl:value-of select="$current-max" />** レコードを表示
変換された結果 **:**	次の **10** レコードを表示

CDML タグ名 : 内部修正 ID

このタグは、現在編集中のレコードの修正 ID で置換されます。

CDML 構文: [FMP-CurrentModID]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合:
  - 現在のコンテキストがレコードの場合: {@mod-id}
  - その他の場合: {/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@mod-id}
- その他の場合:
  - 現在のコンテキストがレコードの場合: <xsl:value-of select="@mod-id" />
  - その他の場合: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@mod-id" />

変換例

元の **CDML:**

```
<FORM ACTION="FMPro" METHOD="POST">
  <INPUT TYPE="HIDDEN" NAME="–DB" VALUE="contacts.fp5">
  <INPUT TYPE="HIDDEN" NAME="–Format" VALUE="results.htm">
  <INPUT TYPE="HIDDEN" NAME="–RecID" VALUE="[FMP-CurrentRecID]">
  <INPUT TYPE="HIDDEN" NAME="–ModID" VALUE="[FMP-CurrentModID]">
  <INPUT TYPE="TEXT" NAME="Country">
  <INPUT TYPE="SUBMIT" NAME="–Edit" VALUE="Edit This Record">
</FORM>
```

変換例

変換された **XSLT-CWP:**	<form action="/fmi/xsl/results.xsl" method="POST"> <input type="HIDDEN" name="–DB" value="contacts"></input> <input type="HIDDEN" name="–grammar" value="fmresultset"></input> <input type="HIDDEN" name="–RecID" value="{@record-id}"></input> <input type="HIDDEN" name="–ModID" value="**{@mod-id}**"></input> <input type="TEXT" name="Country"></input> <input type="SUBMIT" name="–Edit" value="Edit This Record"></input> </form>
変換された結果 :	<form action="/fmi/xsl/results.xsl" method="POST"> <input type="HIDDEN" name="–DB" value="contacts"> <input type="HIDDEN" name="–grammar" value="fmresultset"> <input type="HIDDEN" name="–RecID" value="1032"> <input type="HIDDEN" name="–ModID" value="**3**"> <input type="TEXT" name="Country"> <input type="SUBMIT" name="–Edit" value="Edit This Record"> </form>

CDML タグ名 : ポータルの行番号

このタグは、現在処理中のポータルの行番号で置換されます。このタグは、常に [FMP-Portal] ループの内側にあります。

CDML 構文: [FMP-CurrentPortalRowNumber]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {position()}
- その他の場合: <xsl:value-of select="position()"/>
- [FMP-Portal] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-CurrentPortalRowNumber] outside of [FMP-Portal] -->

変換例

元の **CDML:**	[FMP-Portal:lineitems] **[FMP-CurrentPortalRowNumber]**:[FMP-Field:lineitems::name] [/FMP-Portal]
変換された **XSLT-CWP:**	<xsl:for-each select="fmrs:relatedset[@table='lineitems']/fmrs:record"> **<xsl:value-of select="position()"/>:**<xsl:value-of select="fmrs:field[@name = 'name']" /> </xsl:for-each>
変換された結果 :	**1**: Red

CDML タグ名 : ファイル内のレコード数

このタグは、データベース内のレコードの総数で置換されます。

CDML 構文: [FMP-CurrentRecordCount]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {fmrs:fmresultset/fmrs:datasource/@total-count}
- その他の場合: <xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@total-count"/>

変換例

元の **CDML:**	データベースのレコード数 : **[FMP-CurrentRecordCount]**
変換された **XSLT-CWP:**	データベースのレコード数 : **<xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@total-count"/>**
変換された結果 **:**	データベースのレコード数 : **1123**

CDML タグ名 : レコード ID

このタグは、レコード ID で置換されます。

CDML 構文: [FMP-CurrentRecID]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合:
  - 現在のコンテキストがレコードの場合: {@record-id}
  - その他の場合: {/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@record-id}
- その他の場合:
  - 現在のコンテキストがレコードの場合: <xsl:value-of select="@record-id" />
  - その他の場合: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@record-id" />

変換例

元の **CDML:**	<form action="FMPro" method="post"> <input type="hidden" name="–DB" value="name.fp5"> <input type="hidden" name="–Format" value="results.htm"> <input type="hidden" name="–RecID" value="**[FMP-CurrentRecID]**"> <input type="submit" name="-Delete" value="Delete This Record"> </form>
変換された **XSLT-CWP:**	<form action="/fmi/xsl/results.xsl" method="post"> <input type="hidden" name="–DB" value="name"></input> <input type="hidden" name="–grammar" value="fmresultset"></input> <input type="hidden" name="–RecID" value="**{@record-id}**"></input> <input type="submit" name="–Delete" value="Delete This Record"></input> </form>
変換された結果 **:**	<form action="/fmi/xsl/results.xsl" method="post"> <input type="hidden" name="–DB" value="name"> <input type="hidden" name="–grammar" value="fmresultset"> <input type="hidden" name="–RecID" value="**1023**"> <input type="submit" name="-Delete" value="Delete This Record"> </form>

CDML タグ名 : レコード番号

このタグは、現在の対象レコードのレコードの位置で置換されます。

CDML 構文: [FMP-CurrentRecordNumber]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {position()}
- その他の場合: <xsl:value-of select="position() + $current-skip" />

- [FMP-Record] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-CurrentRecordNumber] outside of [FMP-Record] -->
- ドキュメント内で使用されている場合: $current-skip 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	対象レコード内の現在のレコード : **[FMP-CurrentRecordNumber]**
変換された **XSLT-CWP:**	対象レコード内の現在のレコード : **<xsl:value-of select="position() + $current-skip" />**
変換された結果 :	対象レコード内の現在のレコード : **3**

CDML タグ名 : 繰り返し位置番号

このタグは、現在処理中の繰り返しで置換されます。このタグは、常に [FMP-Repeating][/FMP-Repeating] ループの内側にあります。

CDML 構文: [FMP-CurrentRepeatNumber]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {position()}
- その他の場合: <xsl:value-of select="position()"/>
- [FMP-Repeating] ループの外側にある場合: <!-- CDML Converter エラー: [FMP-CurrentRepeatNumber] outside of [FMP-Repeating] -->

変換例

元の **CDML:**	[FMP-Repeating:extensions] **[FMP-CurrentRepeatNumber]**:[FMP-RepeatingItem] [/FMP-Repeating]
変換された **XSLT-CWP:**	<xsl:for-each select="fmrs:field[@name = 'extensions']/fmrs:data"> **<xsl:value-of select="position()" />**:<xsl:value-of select="."/> </xsl:for-each>
変換された結果 :	**3**: Green

CDML タグ名 : スキップ設定

このタグは、対象レコードの最初からスキップされたレコード数で置換されます。

CDML 構文: [FMP-CurrentSkip]

XSLT-CWP への変換<xsl:call-template name="$get-current-skip" />

- SGML エレメントの属性値内にある場合: {$current-skip}
- その他の場合: <xsl:value-of select="$current-skip"/>
- ドキュメント内で使用されている場合: $current-skip 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	現在のスキップ値 : **[FMP-CurrentSkip]**
変換された **XSLT-CWP:**	現在のスキップ値 : **<xsl:value-of select="$current-skip"/>**
変換された結果 :	現在のスキップ値 : **10**

CDML タグ名 : ソート順

ページを作成するリクエストの一部である各検索条件に対して、[FMP-CurrentSort] タグと [/FMP-CurrentSort] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-CurrentSort]...[/FMP-CurrentSort]

XSLT-CWP への変換

- <xsl:for-each select="$current-sort/sort-field">...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-CurrentSort] not in a valid location -->
- ドキュメント内で使用されている場合: $current-sort 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	現在のソート順 : **[FMP-CurrentSort]** Field:[FMP-SortFieldItem], Order:[FMP-SortOrderItem] [/FMP-CurrentSort]
変換された **XSLT-CWP:**	現在のソート順 : **<xsl:for-each select="$current-sort/sort-field">** Field:<xsl:value-of select="@name" />, Order:<xsl:value-of select="@order" /> </xsl:for-each>
変換された結果 :	現在のソート順 : Field:First Name, Order:descend Field:Last Name, Order:descend

CDML タグ名 : 時刻

このタグは、現在の時刻で置換されます。

CDML 構文: [FMP-CurrentTime:書式]。書式は、「Short」（デフォルト）または「Long」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にあり、「Short」の場合: {fmxslt:get_time('short')}
- または「Long」の場合: {fmxslt:get_time('long')}
- その他の場所にあり、「Short」の場合: <xsl:value-of select="fmxslt:get_time('short')" />
- または「Long」の場合: <xsl:value-of select="fmxslt:get_time('long')" />

変換例

元の **CDML:**	現在の時刻 : **[FMP-CurrentTime:Short]**
変換された **XSLT-CWP:**	現在の時刻 : **<xsl:value-of select="fmxslt:get_time('short')" />**
変換された結果 :	現在の時刻 : **10:12 AM**

CDML タグ名 : トークン

このタグは、現在のページを作成するために使用された -Token 変数タグの値で置換されます。

CDML 構文: [FMP-CurrentToken:番号, エンコード]。番号は「0」...「9」（オプション）、エンコードは「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$request-query/fmq:query/fmq:parameter[@NAME='-token']}

- その他の場合: <xsl:value-of select="$request-query/fmq:query/fmq:parameter[@NAME='-token']" />
- トークンに番号が指定されている場合、クエリー引数は「-token.番号」になります（「-token.5」など）。

変換例

元の **CDML:**	token.5 の値は **[FMP-CurrentToken: 5, HTML]** です。
変換された **XSLT-CWP:**	token.5 の値は **<xsl:value-of select="$request-query/fmq:query/fmq:parameter[@NAME='-token.5']" />** です。
変換された結果 **:**	token.5 の値は **MyValue** です。

CDML タグ名 : Else

このタグは、前の [FMP-If] 論理式が偽の場合に、指定されたデータで置換されます。

CDML 構文: [FMP-If:論理式1]...[FMP-Else]...[/FMP-If]

XSLT-CWP への変換

- <xsl:choose><xsl:when test="論理式1">...</xsl:when><xsl:otherwise>...</xsl:otherwise></xsl:choose>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-Else] not in a valid location -->

変換例

元の **CDML:**	[FMP-If:(Field:country.Eq.us).or.(Field:country.Eq.usa)] United States of America **[FMP-Else]** Other country [/FMP-If]
変換された **XSLT-CWP:**	<xsl:choose> <xsl:when test="'fmrs:field[@name = 'country']/fmrs:data[1] = 'us' or fmrs:field[@name = 'country']/fmrs:data[1] = 'usa'">United States of America**</xsl:when>** **<xsl:otherwise>**Other country**</xsl:otherwise>** </xsl:choose>
変換された結果 **:**	Other country

CDML タグ名 : Else If

このタグは、論理式が真で、前の [FMP-If] 論理式が偽の場合に、指定されたデータで置換されます。

CDML 構文: [FMP-If:論理式1]...[FMP-ElseIf 論理式2]...[/FMP-If]

XSLT-CWP への変換

- <xsl:choose><xsl:when test="論理式1">...</xsl:when><xsl:when test="論理式2">...</xsl:when></xsl:choose>
- 論理式 2 は XPath に変換されます。
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-ElseIf] not in a valid location -->

変換例

元の **CDML:**	[FMP-If:(Field:country.Eq.us).or.(Field:country.Eq.usa)] United States of America **[FMP-ElseIf:Field:country .Eq.Italy]** Italy [/FMP-If]

変換例

変換された **XSLT-CWP:**	<xsl:choose> <xsl:when test="'fmrs:field[@name = 'country']/fmrs:data[1] = 'us' or fmrs:field[@name = 'country']/fmrs:data[1] = 'usa'">United States of America**</xsl:when>** **<xsl:when test="fmrs:field[@name = 'country']/fmrs:data[1] = 'Italy'">**Italy</xsl:when> </xsl:choose>
変換された結果 :	Italy

CDML タグ名 : フィールド

このタグは、指定されたフィールドの内容で置換されます。

CDML 構文: [FMP-Field:フィールド名, エンコード]。エンコードは、「Raw」、「URL」、「HTML」（デフォルト）、または「Break」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合:
  - 現在のコンテキストがレコードの場合: {fmrs:field[@name = 'フィールド名']/fmrs:data[1]}
  - その他の場合: {/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = 'フィールド名']/fmrs:data[1]}
- その他の場合:
  - 現在のコンテキストがレコードの場合: <xsl:value-of select="fmrs:field[@name = 'フィールド名']/fmrs:data[1]" />
  - その他の場合: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = 'フィールド名']/fmrs:data[1]" />

変換例

元の **CDML:**	名 : **[FMP-Field:First Name]**
変換された **XSLT-CWP:**	名 : **<xsl:value-of select="fmrs:field[@name = 'First Name']/fmrs:data[1]" />**
変換された結果 :	名 : **John**

CDML タグ名 : フィールド名

このタグは、現在のフィールドの名前で置換されます。このタグは、常に [FMP-LayoutFields] ループの内側にあります。

CDML 構文: [FMP-FieldName:エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {@name}
- その他の場合: <xsl:value-of select="@name" />
- [FMP-LayoutFields] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-FieldName] outside of [FMP-LayoutFields] -->

変換例

元の **CDML:**	<select name="-SortField"> <option value="">-None- [FMP-LayoutFields] <option>**[FMP-FieldName:Raw]** [/FMP-LayoutFields] </select>

変換例

変換された **XSLT-CWP:**	<select name="-SortField"> <option value="" />-None- <xsl:for-each select="fmrs:fmresultset/fmrs:metadata/fmrs:field-definition" /> <option>**<xsl:value-of select="@name"/>** </option> </xsl:for-each> </select>
変換された結果 :	<select name="-SortField"> <option value="">-None-</option> <option **First Name**</option> <option **Last Name**</option> <option **Employee Number**</option> </select>

CDML タグ名 : 検索条件フィールド

このタグは、ページを作成する検索条件の一部であるフィールド名で置換されます。このタグは、常に [FMP-CurrentFind] ループの内側にあります。

CDML 構文: [FMP-FindFieldItem: エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {@name}
- その他の場合: <xsl:value-of select="@name" />
- ドキュメント内で使用されている場合: $current-find 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。
- [FMP-CurrentFind] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-FindFieldItem] outside of [FMP-CurrentFind] -->

変換例

元の **CDML:**	現在の検索条件 : [FMP-CurrentFind] Field: **[FMP-FindFieldItem]**, Op:[FMP-FindOpItem], Value:[FMP-FindValueItem] [/FMP-CurrentFind]
変換された **XSLT-CWP:**	現在の検索条件 : <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field:**<xsl:value-of select="@name" />**, Op:<xsl:value-of select="@long-operator" />, Value:<xsl:value-of select="text()" /> </xsl:for-each>
変換された結果 :	現在の検索条件 : Field:**First Name**, Op:begins with, Value:John Field:**Last Name**, Op:equals, Value:Doe

CDML タグ名 : 検索条件演算子

このタグは、ページを作成する検索条件の一部である検索演算子で置換されます。このタグは、常に [FMP-CurrentFind] ループの内側にあります。

CDML 構文: [FMP-FindOpItem:書式]。書式は、「Short」、「Long」（デフォルト）、または「Display」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {@short-operator} または {@long-operator}
- その他の場合: <xsl:value-of select="@short-operator" /> または <xsl:value-of select="@long-operator" />
- 書式が「Display」の場合:「Long」が使用され、<!-- CDML2XSLT 警告:[FMP-SortOrderItem]「Display」書式は XSLT-CWP ではサポートされていません。--> が挿入されます。
- ドキュメント内で使用されている場合: $current-find 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。
- [FMP-CurrentFind] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-FindOpItem] outside of [FMP-CurrentFind] -->

変換例

元の **CDML:**	現在の検索条件 : [FMP-CurrentFind] Field:[FMP-FindFieldItem], Op: **[FMP-FindOpItem]**, Value:[FMP-FindValueItem] [/FMP-CurrentFind]
変換された **XSLT-CWP:**	現在の検索条件 : <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field:<xsl:value-of select="@name" />, Op:**<xsl:value-of select="@long-operator" />**, Value:<xsl:value-of select="text()" /> </xsl:for-each>
変換された結果 :	現在の検索条件 : Field:First Name, Op:**begins with**, Value:John Field:Last Name, Op:**equals**, Value:Doe

CDML タグ名 : 検索条件値一覧

このタグは、ページを作成する検索条件の一部である値で置換されます。このタグは、常に [FMP-CurrentFind] ループの内側にあります。

CDML 構文: [FMP-FindValueItem: エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {text()}
- その他の場合: <xsl:value-of select="text()" />
- ドキュメント内で使用されている場合: $current-find 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。
- [FMP-CurrentFind] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-FindValueItem] outside of [FMP-CurrentFind] -->

変換例

元の **CDML:**	現在の検索条件 : [FMP-CurrentFind] Field:[FMP-FindFieldItem], Op:[FMP-FindOpItem], Value:**[FMP-FindValueItem]** [/FMP-CurrentFind]

変換例

変換された **XSLT-CWP:**	現在の検索条件 : <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field:<xsl:value-of select="@name" />, Op:<xsl:value-of select="@long-operator" />, Value:**<xsl:value-of select="text()" />** </xsl:for-each>
変換された結果 :	現在の検索条件 : Field:First Name, Op:begins with, Value:**John** Field:Last Name, Op:equals, Value:**Doe**

CDML タグ名 : ヘッダ

ブラウザに送信されたページの HTTP ヘッダが、[FMP-Header] タグと [/FMP-Header] タグの間にあるテキストで置換されます。これらのタグの間のテキストは、ページの HTML 部分には表示されません。

CDML 構文: [FMP-Header]...[/FMP-Header]

XSLT-CWP への変換

- ステータスコードがヘッダコンテンツの中にある場合: <xsl:variable name="header-status-code1" select="set_status_code('<コンテンツのステータスコード>')" />
- ヘッダコンテンツ内の値と名前の各組に対する置換: <xsl:variable name="header-param番号" select="fmxslt:set_header('<引数名>', '<引数値>')" />
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-Header] not in a valid location -->

変換例

元の **CDML:**	ヘッダコード : **[FMP-Header]** **HTTP/1.0 302 Moved Temporary** **Location:http://www.FileMaker.com** [/FMP-Header]
変換された **XSLT-CWP:**	ヘッダコード : **<xsl:variable name="header-status-code1" select="set_status_code(302)" />** **<xsl:variable name="header-param1" select="fmxslt:set_header('location', 'http://www.FileMaker.com')" />**
変換された結果 :	ヘッダコード : (HTTP 応答のヘッダにも、**Location:http://www.filemaker.com** が含まれます。)

CDML タグ名 : If

[FMP-If] は、[FMP-Else]、[FMP-ElseIf]、および [/FMP-If] タグとともに、ブラウザによって表示される HTML を制御します。

CDML 構文: [FMP-If BooleanExpression]...[/FMP-If]

XSLT-CWP への変換

- <xsl:choose><xsl:when test="論理式">...</xsl:when></xsl:choose>
- 論理式は XPath に変換されます。
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-If] not in a valid location -->

変換例

元の **CDML:**	**[FMP-If:First Name .eq. field:Nick Name]** Your nick name is the same as your name. **[/FMP-If]**
変換された **XSLT-CWP:**	**<xsl:choose><xsl:when test="'First Name' = fmrs:field[@name = 'Nick Name']/ fmrs:data[1]">** Your nickname is the same as your name. **</xsl:when></xsl:choose>**
変換された結果 **:**	Your nick name is the same as your name.

CDML タグ名 : 画像

このタグは、指定されたフィールド内のイメージへのオブジェクト URL 参照で置換されます。

CDML 構文: [FMP-Image: フィールド名]

XSLT-CWP への変換

- 現在のコンテキストがレコードの場合: <xsl:call-template name="get-image"><xsl:with-param name="fieldname" select="'fieldname'"/><xsl:with-param name="recid" select="$record/@record-id"/></xsl:call-template>
- その他の場合: <xsl:call-template name="get-image"><xsl:with-param name="fieldname" select="'fieldname'"/><xsl:with-param name="recid" select="$default-record/@record-id"/></xsl:call-template>

変換例

元の **CDML:**	<img src="**[FMP-Image:pictures]**">
変換された **XSLT-CWP:**	<img> **<xsl:attribute name="src">** **<xsl:call-template name="get-image">** **<xsl:with-param name="fieldname" select="'pictures'"/>** **<xsl:with-param name="recid" select="$default-record/@record-id"/>** **</xsl:call-template>** **</xsl:attribute>** </img>
変換された結果 **:**	**

CDML タグ名 : 他のファイルを含める

このタグは、他のファイルの内容で置換されます。通常は HTML フォーマットファイルで置換されます。

CDML 構文: [FMP-Include:ファイル名]

XSLT-CWP への変換

- <!-- CDML Converter 警告:ファイル「ファイル名」はこのドキュメントにインラインで組み込まれ、元のファイルへの参照は削除されています。元のファイルは不要になった可能性があります。--> ファイルの変換後の内容
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-Include] not in a valid location -->

変換例

元の **CDML:**	[FMP-Include:requirefield.htm]
変換された **XSLT-CWP:**	<!-- CDML Converter 警告 : ファイル「requirefield.htm」はこのドキュメントにインラインで組み込まれ、元のファイルへの参照は削除されています。元のファイルは不要になった可能性があります。--> ファイルの変換後の内容
変換された結果 :	ファイルの変換後の内容

CDML タグ名 : フォーマットファイルのフィールド

このタグは、フィールドの内容で置換されます。通常は、HTML フォーマットファイルが含まれるテキストフィールドで置換されます。

CDML 構文: [FMP-IncludeField:フィールド名]

XSLT-CWP への変換

- <!-- CDML Converter エラー : [FMP-IncludeField] は XSLT-CWP ではサポートされていません。-->

変換例

元の **CDML:**	[FMP-IncludeField:errorPage]
変換された **XSLT-CWP:**	<!-- CDML Converter エラー : [FMP-IncludeField] は XSLT-CWP ではサポートされていません。-->
変換された結果 :	<!-- CDML Converter エラー : [FMP-IncludeField] は XSLT-CWP ではサポートされていません。-->

CDML タグ名 : インラインアクション

このタグを使用することで、1 つのフォーマットファイルを処理する際に複数の CDML リクエストを処理できます。[FMP-InlineAction] タグでは、CDML リクエストの名前と値を組とした URL のような形式で引数を指定します。フォーマットファイルの以降の処理は、すべてインラインリクエストによって処理が開始されたかのように続行されます。

CDML 構文: [FMP-InlineAction:イントラタグ]...[/FMP-InlineAction]

XSLT-CWP への変換

- <xsl:variable name="inline-action" select="document(concat($authenticated-xml-base-uri, '/fmi/xml/fmresultset.xml?', '-db=', /fmrs:fmresultset/fmrs:datasource/@database, '&-lay=web3&title=s&-find'))"/><xsl:for-each select="$inline-action/fmrs:fmresultset/fmrs:resultset/fmrs:record"><xsl:variable name="inline-action-record" select="current()"/>...</xsl:for-each>
- 開始タグと終了タグの間にスペース以外の内容がある場合、応答ドキュメントのレコードがループ処理されます。
- リクエスト URL の引数の値は、URL エンコードされます。

- リクエスト URL はイントラタグから作成され、「-format」引数は無視されます。
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-InlineAction] not in a valid location -->

変換例

元の **CDML:**	**[FMP-InlineAction:-db={currentdatabase}, title="s", -lay=web3, find=]** Title:[FMP-Field:Title], Artist:[FMP-Field:Artist] [/FMP-InlineAction]
変換された **XSLT-CWP:**	**<xsl:variable name="inline-action" select="document(concat($authenticated-xml-base-uri, '/fmi/xml/fmresultset.xml?', '-db=', /fmrs:fmresultset/fmrs:datasource/@database, '&-lay=web3&title=s&-find'))"/>** **<xsl:for-each select="$inline-action/fmrs:fmresultset/fmrs:resultset/fmrs:record">** **<xsl:variable name="inline-action-record" select="current()"/>** Title:<xsl:value-of select="**$inline-action-record**/fmrs:field[@name = 'Title']/fmrs:data[1]"/>, Artist:<xsl:value-of select="**$inline-action-record**/fmrs:field[@name = 'Artist']/fmrs:data[1]"/> </xsl:for-each>
変換された結果 :	Title:Mona Lisa, Artist:Leonardo da Vinci Title:Sunflowers, Artist:Vincent Van Gogh

CDML タグ名 : レイアウト内のフィールド

ページを作成するリクエストの一部である指定されたレイアウト上のすべてのフィールドに対して、[FMP-LayoutFields] タグと [/FMP-LayoutFields] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-LayoutFields]...[/FMP-LayoutFields]

XSLT-CWP への変換

- <xsl:for-each select="fmrs:fmresultset/fmrs:metadata/fmrs:field-definition" />...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー :[FMP-LayoutFields] not in a valid location -->

変換例

元の **CDML:**	<select name="-SortField"> <option value="">-None- **[FMP-LayoutFields]** <option>[FMP-FieldName:Raw] **[/FMP-LayoutFields]** </select>
変換された **XSLT-CWP:**	<select name="-SortField.1"> <option value="" />-None-</option> **<xsl:for-each select="fmrs:fmresultset/fmrs:metadata/fmrs:field-definition" />** <option /><xsl:value-of select="@name" /> **</xsl:for-each>** </select>
変換された結果 :	<select name="-SortField.1"> <option value="">-None- <option >First Name</option> <option >Last Name</option> <option >Employee Number</option> </select>

CDML タグ名 : リンク

このタグは、タグが存在するページを参照する URL で置換されます。引数を使用して、生成された URL の一部を削除することができます。

CDML 構文: <sgml_tag sgml_attr="[FMP-Link:文字コード]&-format=ファイル名.htm&名前1=値1&名前2=値2...">

XSLT-CWP への変換

- 挿入される内容:<xsl:call-template name="get-link"><xsl:with-param name="filter-codes" select="'文字コード'"/><xsl:with-param name="stylesheet" select="'ファイル名.xsl'"/><xsl:with-param name="other-params" select="'&名前1=値1&名前2=値2...'"/></xsl:call-template>
- SGML 属性内にあり、続いて「-format」引数が指定されている場合:「stylesheet」引数が「get-link」テンプレートに渡されます。
- SGML 属性内にあり、続いて追加のクエリー引数が指定されている場合:「other-params」引数が「get-link」テンプレートに渡されます。

変換例

元の **CDML:**	Alternate hit list
変換された **XSLT-CWP:**	<a> **<xsl:attribute name="href">** **<xsl:call-template name="get-link">** **<xsl:with-param name="filter-codes" select="'adr'"/>** **<xsl:with-param name="stylesheet" select="'AltHitList.xsl'"/>** **<xsl:with-param name="other-params" select="'&-db=art'"/>** **</xsl:call-template>** **</xsl:attribute>**Alternate hit list
変換された結果 :	Alternate hit list

CDML タグ名 : 最初へのリンク

[FMP-LinkFirst] タグと [/FMP-LinkFirst] タグの間にある HTML が、現在のページを作成するために使用された -Max の値に基づくレコードの最初の範囲へのリンクで置換されます。

CDML 構文: [FMP-LinkFirst]...[/FMP-LinkFirst]

XSLT-CWP への変換

- ...
- 最上位のドキュメントに挿入される内容:<xsl:variable name="link-first"><xsl:call-template name="get-link-first"/></xsl:variable>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-LinkFirst] not in a valid location -->

変換例

元の **CDML:**	**[FMP-LinkFirst]**First set of records**[/FMP-LinkFirst]**
変換された **XSLT-CWP:**	****First set of records****
変換された結果 :	****First set of records****

CDML タグ名 : 最後へのリンク

[FMP-LinkLast] タグと [/FMP-LinkLast] タグの間にある HTML が、現在のページを作成するために使用された -Max の値に基づくレコードの最後の範囲へのリンクで置換されます。

CDML 構文: [FMP-LinkLast]...[/FMP-LinkLast]

XSLT-CWP への変換

- ...
- 最上位のドキュメントに挿入される内容: <xsl:variable name="link-last"><xsl:call-template name="get-link-last"/></xsl:variable>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-LinkLast] not in a valid location -->

変換例

元の **CDML:**	**[FMP-LinkLast]**Last set of records**[/FMP-LinkLast]**
変換された **XSLT-CWP:**	****Last set of records****
変換された結果 :	****Last set of records****

CDML タグ名 : 次へのリンク

[FMP-LinkNext] タグと [/FMP-LinkNext] タグの間にある HTML が、現在のページを作成するために使用された -Max および -Skip の値に基づくレコードの次の範囲へのリンクで置換されます。

CDML 構文: [FMP-LinkNext]...[/FMP-LinkNext]

XSLT-CWP への変換

- ...
- 最上位のドキュメントに挿入される内容: <xsl:variable name="link-next"><xsl:call-template name="get-link-next"/> </xsl:variable>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-LinkNext] not in a valid location -->

変換例

元の **CDML:**	**[FMP-LinkNext]**Next set of records**[/FMP-LinkNext]**
変換された **XSLT-CWP:**	****Next set of records****
変換された結果 :	****Next set of records****

CDML タグ名 : 前へのリンク

[FMP-LinkPrevious] タグと [/FMP-LinkPrevious] タグの間にある HTML が、現在のページを作成するために使用された -Max および -Skip の値に基づくレコードの前の範囲へのリンクで置換されます。

CDML 構文: [FMP-LinkPrevious]...[/FMP-LinkPrevious]

XSLT-CWP への変換

- ...
- 最上位のドキュメントに挿入される内容: <xsl:variable name="link-previous"> <xsl:call-template name="get-link-previous"/></xsl:variable>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-LinkPrevious] not in a valid location -->

変換例

元の **CDML:**	**[FMP-LinkPrevious]**Previous set of records**[/FMP-LinkPrevious]**
変換された **XSLT-CWP:**	****Previous set of records****
変換された結果 :	****Previous set of records****

CDML タグ名 : レコード ID へのリンク

このタグは、データベース内の特定のレコードの URL で置換されます。生成されたリンクには、このタグが存在するページを生成するために使用された検索条件とソート条件がすべて含まれます。Layout 引数はオプションです。

CDML 構文: [FMP-LinkRecID: Format=ファイル名, Layout=レイアウト名]

XSLT-CWP への変換

- <xsl:call-template name="get-link-rec-id"><xsl:with-param name="rec-id" select="@record-id"/><xsl:with-param name="stylesheet" select="'/ファイル名.xsl'"/></xsl:call-template>
- 後に「-format」引数が指定されている場合: 「stylesheet」引数が「get-link-rec-id」テンプレートに渡されます。
- 後に「-lay」引数が指定されている場合: 「layout」引数が「get-link-rec-id」テンプレートに渡されます。
- SGML 属性内にない場合: <!-- CDML Converter エラー : [FMP-LinkRecID] not in a valid location -->

変換例

元の **CDML:**	More detail
変換された **XSLT-CWP:**	<a> <xsl:attribute name="href"> **<xsl:call-template name="get-link-rec-id">** **<xsl:with-param name="rec-id" select="$record/@record-id"/>** **<xsl:with-param name="stylesheet" select="'FormatFile.xsl'"/>** **<xsl:with-param name="layout" select="'detail'"/>** **</xsl:call-template>** </xsl:attribute>More Detail
変換された結果 :	More detail

CDML タグ名 : ログ

このタグは、引数として指定したテキストで置換され、アプリケーションログファイル内にユーザログメッセージとして書き込まれます。72 ページの「Web 公開エンジンのアプリケーションログの使用」を参照してください。

CDML 構文: [FMP-Log:テキスト]

XSLT-CWP への変換

- <xsl:message>テキスト</xsl:message>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-Log] not in a valid location -->

変換例

元の **CDML:**	ログコード : **[FMP-Log:the search page was accessed]**
変換された **XSLT-CWP:**	ログコード :**<xsl:message>the search page was accessed</xsl:message>**
変換された結果 :	ログコード : (ユーザログが有効な場合は、「**the search page was accessed**」というテキストがアプリケーションログに書き込まれます。)

CDML タグ名 : オプション

このタグは、レイアウト内のフィールドの値一覧に含まれるすべての値で置換されます。

CDML 構文: [FMP-Option: フィールド名, List=値一覧名]

注意　適切に変換するには、「[FMP-Option]」CDML タグに、指定された値一覧を参照するレイアウト上のフィールドが指定されている必要があります。

XSLT-CWP への変換

- <xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'フィールド３']/fml:STYLE/@VALUELIST"/><xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name-4]/fml:VALUE"><option value="{current()}"><xsl:if test="current() = $default-record/fmrs:field[@name = 'フィールド３']/fmrs:data[1]"><xsl:attribute name="selected">selected</xsl:attribute></xsl:if><xsl:value-of select="current()"/></option>
</xsl:for-each>
  - 現在のコンテキストがレコードの場合: 上の fmrs:field[@name = 'フィールド名']/fmrs:data[1] が使用されます。
  - その他の場合: 上の /fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = 'フィールド名']/fmrs:data[1] が使用されます。

- 「List」引数が無視された場合: <!-- CDML2XSLT 警告:[FMP-Option]「List」引数は XSLT-CWP ではサポートされていません。-->
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-Option] not in a valid location -->
- ドキュメント内で使用されている場合: ドキュメントの最上位に $layout 変数が作成されます。

変換例

元の **CDML:**	<select name="Groups"> **[FMP-option:Groups]** </select>
変換された **XSLT-CWP:**	<select> **<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/>** **<xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE">** **<option value="{current()}"><xsl:if test="current() = $default-record/fmrs:field[@name = 'Groups']/fmrs:data[1]"><xsl:attribute name="selected">selected</xsl:attribute></xsl:if><xsl:value-of select="current()"/></option></xsl:for-each>** </select>
変換された結果 :	<select name="Groups"> **<option> Production** **<option selected> Sales** **<option> Support** </select>

CDML タグ名 : ポータル

指定されたポータル内の各レコードに対して、[FMP-Portal] タグと [/FMP-Portal] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-Portal:リレーションシップ名]...[/FMP-Portal]

XSLT-CWP への変換

- 現在のコンテキストがレコードの場合: <xsl:for-each select="fmrs:relatedset[@table = 'リレーションシップ名']/fmrs:record">...</xsl:for-each>
- その他の場合: <xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record/fmrs:relatedset[@table='リレーションシップ名']/fmrs:record">...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-Portal] not in a valid location -->

変換例

元の **CDML:**	**[FMP-Portal:lineitems]** [FMP-CurrentPortalRowNumber]:[FMP-Field:lineitems::name] [/FMP-Portal]
変換された **XSLT-CWP:**	**<xsl:for-each select="fmrs:relatedset[@table='lineitems']/fmrs:record">** <xsl:value-of select="position()" />:<xsl:value-of select="fmrs:field[@name = 'name']" /> </xsl:for-each>
変換された結果 :	1: Red

CDML タグ名 : 表示範囲の終わり

このタグは、表示されている最後のレコードのレコード番号で置換されます。

CDML 構文: [FMP-RangeEnd]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size}
- その他の場合: <xsl:value-of select="$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size"/>
- ドキュメント内で使用されている場合: $current-skip 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	Records [FMP-RangeStart] through **[FMP-RangeEnd]**
変換された **XSLT-CWP:**	Records <xsl:value-of select="$current-skip + 1"/> through **<xsl:value-of select="$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size"/>**
変換された結果 :	Records 6 through **10**

CDML タグ名 : 表示範囲のサイズ

このタグは、ページに表示されるレコードの数で置換されます。

CDML 構文: [FMP-RangeSize]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {/fmrs:fmresultset/fmrs:resultset/@fetch-size}
- その他の場合: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/@fetch-size" />

変換例

元の **CDML:**	You are viewing **[FMP-RangeSize]** records.
変換された **XSLT-CWP:**	You are viewing **<xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/@fetch-size" />** records.
変換された結果 :	You are viewing **8** records.

CDML タグ名 : 表示範囲の始め

このタグは、表示されている最初のレコードのレコード番号で置換されます。

CDML 構文: [FMP-RangeStart]

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {$current-skip + 1}
- その他の場合: <xsl:value-of select="$current-skip + 1"/>
- ドキュメント内で使用されている場合: $current-skip 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

変換例

元の **CDML:**	Records **[FMP-RangeStart]** through [FMP-RangeEnd]
変換された **XSLT-CWP:**	Records **<xsl:value-of select="$current-skip + 1"/>** through <xsl:value-of select="$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size"/>
変換された結果 :	Records **6** through 10

CDML タグ名 : レコード

-Skip で指定されたレコードをスキップした後、-Max で指定された数までのすべてのレコードに対して、[FMP-Record] タグと [/FMP-Record] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-Record]...[/FMP-Record]

XSLT-CWP への変換

- <xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record">...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-Record] not in a valid location -->

変換例

元の **CDML:**	**[FMP-Record]** [FMP-Field:Country] - [FMP-Field:Capital] **[/FMP-Record]**
変換された **XSLT-CWP:**	**<xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record">** <xsl:value-of select="fmrs:field[@name = 'Country']/fmrs:data[1]" /> - <xsl:value-of select="fmrs:field[@name = 'Capital']/fmrs:data[1]" /> **</xsl:for-each>**
変換された結果 :	Great Britain - London France - Paris USA - Washington D.C.

CDML タグ名 : 繰り返しフィールド

指定されたフィールドの繰り返しに対して、[FMP-Repeating] タグと [/FMP-Repeating] タグの間にある HTML を繰り返します。タグ [FMP-RepeatingItem] が、特定の繰り返しの内容で置換されます。

CDML 構文: [FMP-Repeating:フィールド名]...[/FMP-Repeating]

XSLT-CWP への変換

- 現在のコンテキストがレコードの場合: <xsl:for-each select="fmrs:field[@name = 'フィールド名']/fmrs:data">...</xsl:for-each>
- その他の場合: <xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = 'フィールド名']/fmrs:data">...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-Repeating] not in a valid location -->

変換例

元の **CDML:**	**[FMP-Repeating:extensions]** [FMP-CurrentRepeatNumber]:[FMP-RepeatingItem] **[/FMP-Repeating]**
変換された **XSLT-CWP:**	**<xsl:for-each select="fmrs:field[@name = 'extensions']/fmrs:data">** <xsl:value-of select="position()" />:<xsl:value-of select="."/> **</xsl:for-each>**
変換された結果 :	3: Green

CDML タグ名 : 繰り返し項目

このタグは、次の繰り返しの内容で置換されます。このタグは、常に [FMP-Repeating] ループの内側にあります。

CDML 構文: [FMP-RepeatingItem: エンコード]。エンコードは、「Raw」、「URL」、「HTML」（デフォルト）、または「Break」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {.}
- その他の場合: <xsl:value-of select="." />

変換例

元の **CDML:**	[FMP-Repeating:extensions] [FMP-CurrentRepeatNumber]:**[FMP-RepeatingItem]** [/FMP-Repeating]
変換された **XSLT-CWP:**	<xsl:for-each select="fmrs:field[@name = 'extensions']/fmrs:data"> <xsl:value-of select="position()" />:**<xsl:value-of select="."/>** </xsl:for-each>
変換された結果 :	3: **Green**

CDML タグ名 : Cookie を設定

このタグは、HTML の内容では置換されず、ユーザのブラウザに変数を保存するために使用されます。Expires、Path、および Domain 引数はオプションです。

CDML 構文: [FMP-SetCookie:名前=値, Expires=有効期限, Path=パス, Domain=ドメイン]

XSLT-CWP への変換

- <xsl:variable name="cookie-名前" select="fmxslt:set_cookie('名前', '値', '有効期限', 'パス', 'ドメイン')" />
- オプションの引数である有効期限、パス、およびドメインが定義されていない場合: 空の文字列として渡されます。
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-SetCookie] not in a valid location -->

変換例

元の **CDML:**	Set-Cookie here: **[FMP-SetCookie:ColorChoice=Green, Expires=43200]**
変換された **XSLT-CWP:**	Set-Cookie here:**<xsl:variable name="cookie-ColorChoice" select="fmxslt:set_cookie('ColorChoice', 'Value', '43200', '', '')" />**
変換された結果 :	Set-Cookie here:

CDML タグ名 : ソートフィールド項目

このタグは、ページを作成するリクエストの一部であるフィールド名で置換されます。このタグは、常に [FMP-CurrentSort] ループの内側にあります。

CDML 構文: [FMP-SortFieldItem: エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {@name}
- その他の場合: <xsl:value-of select="@name" />
- ドキュメント内で使用されている場合: $current-sort 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。

- [FMP-CurrentSort] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-SortFieldItem] outside of [FMP-CurrentSort] -->

変換例

元の **CDML:**	Current sort order is: [FMP-CurrentSort] Field:**[FMP-SortFieldItem]**, Order:[FMP-SortOrderItem] [/FMP-CurrentSort]
変換された **XSLT-CWP:**	Current sort order is: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field:**<xsl:value-of select="@name" />**, Order:<xsl:value-of select="@order" /> </xsl:for-each>
変換された結果 :	Current sort order is: Field:**First Name**, Order:descend Field:**Last Name**, Order:descend

CDML タグ名 : ソート優先順位

このタグは、ページを作成するリクエストの一部であるソート優先順位で置換されます。このタグは、常に [FMP-CurrentSort] ループの内側にあります。

CDML 構文: [FMP-SortOrderItem:エンコード]。エンコードは、「Raw」、「URL」、「HTML」（デフォルト）、または「Display」です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {@order}
- その他の場合: <xsl:value-of select="@order" />
- エンコードが「Display」の場合: HTML が使用され、<!-- CDML2XSLT 警告:[FMP-SortOrderItem]「Display」のエンコードは XSLT-CWP ではサポートされていません。--> が挿入されます。
- ドキュメント内で使用されている場合: $current-sort 変数は、「cdml2xsl_utilities.xsl」スタイルシートから指定されたテンプレートを使用して、最上位に作成されます。
- [FMP-CurrentSort] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-SortOrderItem] outside of [FMP-CurrentSort] -->

変換例

元の **CDML:**	Current sort order is: [FMP-CurrentSort] Field:[FMP-SortFieldItem], Order:**[FMP-SortOrderItem]** [/FMP-CurrentSort]
変換された **XSLT-CWP:**	Current sort order is: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field:<xsl:value-of select="@name" />, Order:**<xsl:value-of select="@order" />** </xsl:for-each>
変換された結果 :	Current sort order is: Field:First Name, Order:**descend** Field:Last Name, Order:**descend**

CDML タグ名 : 値一覧項目

指定した値一覧のすべての値に対して、[FMP-ValueList] タグと [/FMP-Valuelist] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-ValueList:フィールド名, List=値一覧名]...[/FMP-ValueList]

注意　適切に変換するには、「[FMP-ValueList]」タグに、指定された値一覧を参照するレイアウト上のフィールドが指定されている必要があります。

XSLT-CWP への変換

- <xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'フィールド名']/fml:STYLE/@VALUELIST"/><xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE">...</xsl:for-each>
- 「List」引数が無視された場合: <!-- CDML2XSLT 警告:[FMP-ValueList] 「List」引数は XSLT-CWP ではサポートされていません。-->
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-ValueList] not in a valid location -->
- ドキュメント内で使用されている場合: ドキュメントの最上位に $layout 変数が作成されます。

変換例

元の **CDML:**	**[FMP-ValueList:Groups, List=GroupList]** <input type="radio" name="Groups" value="[FMP-ValueListItem]" [FMP-ValueListChecked]>[FMP-ValueListItem] [/FMP-ValueList]
変換された **XSLT-CWP:**	**<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/>** **<xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE">** **<!-- CDML2XSLT 警告 :[FMP-ValueList] 「List」引数は XSLT-CWP ではサポートされていません。-->** <input type="radio" name="Groups" value="{current()}"><xsl:if test=".= $current-record/fmrs:field[@name = 'Groups']/fmrs:data[1]"><xsl:attribute name="checked">checked</xsl:attribute></xsl:if></input><xsl:value-of select="{current()}" />
変換された結果 :	<input type="radio" name="Groups" value="Production">Production <input type="radio" name="Groups" value="Sales" checked>Sales <input type="radio" name="Groups" value="Support">Support

CDML タグ名 : 選択された値

このタグは、指定したフィールドで選択されている値一覧のすべての項目に対して、「checked」という単語で置換されます。このタグは、常に [FMP-ValueList] ループの内側にあります。

CDML 構文: <input name="フィールド名" value="フィールド値" [FMP-ValueListChecked]>

XSLT-CWP への変換

- <xsl:if test="current() = fmrs:field[@name = 'フィールド名']/fmrs:data[1]"><xsl:attribute name="checked">checked</xsl:attribute></xsl:if>
- [FMP-ValueList] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-ValueListChecked] outside of [FMP-ValueList] -->
- SGML の「input」タグ内にない場合: <<!-- CDML Converter エラー : [FMP-ValueListChecked] not in a valid location -->

- ドキュメント内で使用されている場合: ドキュメントの最上位に $layout 変数が作成されます。

変換例

元の **CDML:**	[FMP-ValueList:Groups, List=GroupList] <input type="radio" name="Groups" value="[FMP-ValueListItem]" **[FMP-ValueListChecked]**>[FMP-ValueListItem] [/FMP-ValueList]
変換された **XSLT-CWP:**	<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/> <xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE"> <!-- CDML2XSLT 警告 :[FMP-ValueList] 「List」引数は XSLT-CWP ではサポートされていません。--> <input type="radio" name="Groups" value="{current()}"**><xsl:if test="current() = fmrs:field[@name = 'Groups']/fmrs:data[1]"> <xsl:attribute name="checked">checked</xsl:attribute></xsl:if>**</input><xsl:value-of select="{current()}" /> </xsl:for-each>
変換された結果 :	<input type="radio" name="Groups" value="Production">Production <input type="radio" name="Groups" value="Sales" **checked**>Sales <input type="radio" name="Groups" value="Support">Support

CDML タグ名 : 値一覧内の項目

このタグは、値一覧の次のエレメントで置換されます。このタグは、常に [FMP-ValueList] ループの内側にあります。

CDML 構文: [FMP-ValueListItem: Checked, エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {current()}
- その他の場合: <xsl:value-of select="current()" />
- [FMP-ValueList] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-ValueListItem] outside of [FMP-ValueList] -->
- ドキュメント内で使用されている場合: ドキュメントの最上位に $layout 変数が作成されます。

変換例

元の **CDML:**	[FMP-ValueList:Groups, List=GroupList] <input type="radio" name="Groups" value="**[FMP-ValueListItem]**" [FMP-ValueListChecked]> **[FMP-ValueListItem]** [/FMP-ValueList]
変換された **XSLT-CWP:**	<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/> <xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE"> <!-- CDML2XSLT 警告 :[FMP-ValueList] 「List」引数は XSLT-CWP ではサポートされていません。--> <input type="radio" name="Groups" value="**{current()}**"><xsl:if test=".= $current-record/fmrs:field[@name = 'Groups']/fmrs:data[1]"><xsl:attribute name="checked">checked</xsl:attribute></xsl:if></input>**<xsl:value-of select="{current()}" />**

変換例

変換された結果 :	<input type="radio" name="Groups" value="**Production**">**Production** <input type="radio" name="Groups" value="**Sales**" checked>**Sales** <input type="radio" name="Groups" value="**Support**">**Support**

CDML タグ名 : 値一覧名

このタグは、値一覧の名前で置換されます。このタグは、常に [FMP-ValueNames] ループの内側にあります。

CDML 構文: [FMP-ValueNameItem:エンコード]。エンコードは、「Raw」、「URL」、または「HTML」（デフォルト）です。

XSLT-CWP への変換

- SGML エレメントの属性値内にある場合: {@NAME}
- その他の場合: <xsl:value-of select="@NAME" />
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-ValueNames] not in a valid location -->
- [FMP-ValueNames] ループの外側にある場合: <!-- CDML Converter エラー : [FMP-ValueNameItem] outside of [FMP-ValueNames] -->
- ドキュメント内で使用されている場合: ドキュメントの最上位に $layout 変数が作成されます。

変換例

元の **CDML:**	<select name="–sortorder"> <option>Ascending <option>Descending [FMP-ValueNames] <option value="Custom=[FMP-ValueNameItem]">**[FMP-ValueNameItem]** [/FMP-ValueNames] </select>
変換された **XSLT-CWP:**	<select name="–sortorder.1"> <option />Ascending <option />Descending <xsl:for-each select="$layout/fml:fmpxmllayout/fml:VALUELISTS/fml:VALUELIST"> <option value="Custom=**{@NAME}**" />**<xsl:value-of select="@NAME" />** /xsl:for-each> </select>
変換された結果 :	<select name="–sortorder.1"> <option />Ascending <option />Descending <option value="Custom=Colors">**Colors** <option value="Custom=Sizes">**Sizes** </select>

CDML タグ名 : 値一覧

データベース内にあるすべての値一覧に対して、[FMP-ValueNames] タグと [/FMP-ValueNames] タグの間にある HTML を繰り返します。

CDML 構文: [FMP-ValueNames]...[/FMP-ValueNames]

注意　適切に変換するには、「[FMP-ValueNames]」CDML タグに、指定された値一覧を参照するレイアウト上のフィールドが指定されている必要があります。

XSLT-CWP への変換

- <xsl:for-each select="$layout/fml:fmpxmllayout/fml:VALUELISTS/fml:VALUELIST">...</xsl:for-each>
- SGML タグまたは属性内にある場合: <!-- CDML Converter エラー : [FMP-ValueNames] not in a valid location -->
- ドキュメント内で使用されている場合: ドキュメントの最上位に $layout 変数が作成されます。

変換例

元の **CDML:**	<select name="–sortorder"> <option>Ascending <option>Descending **[FMP-ValueNames]** <option value="Custom=[FMP-ValueNameItem]">[FMP-ValueNameItem] **[/FMP-ValueNames]** </select>
変換された **XSLT-CWP:**	<select name="–sortorder.1"> <option />Ascending <option />Descending **<xsl:for-each select="$layout/fml:fmpxmllayout/fml:VALUELISTS/ fml:VALUELIST">** <option value="Custom={@NAME}" /><xsl:value-of select="@NAME" /> **</xsl:for-each>** </select>
変換された結果 :	<select name="–sortorder.1"> <option />Ascending <option />Descending <option value="Custom=Colors">Colors <option value="Custom=Sizes">Sizes </select>

索引

G

H

I

J

L

M

P

Q

S

U

V

W

X

あ

い

う

え

お

か

く

と

に

ね

は

ひ

ふ

へ

ほ

ま

め

も

ゆ

よ

れ

ろ