Claris FileMaker Server 19

カスタム Web 公開ガイド

Claris

Claris International Inc.
5201 Patrick Henry Drive
Santa Clara, California 95054

第 01 版

目次

第 8 章
FileMaker API for PHP の使用 72

はじめに

このガイドについて

このガイドでは、Claris™ 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。FileMaker Pro データベースの設計の基礎、ならびにフィールド、リレーションシップ、レイアウト、ポータル、およびオブジェクトについてご理解いただく必要があります。FileMaker Pro の詳細については、FileMaker Pro ヘルプを参照してください。

このガイドでは、FileMaker データを Web サイトおよび Web アプリケーションに統合することを目的として、XML または PHP などのテクノロジーを使用した Web サイトの開発経験があることも想定しています。

このガイドでは、Claris FileMaker Server でのカスタム Web 公開に関する次の情報を説明します:

- 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 を使用してデータベースを公開する方法
- FileMaker Server で共有されているデータベースから XML データを取得する方法
- PHP を使用してデータベースを公開する方法
- FileMaker Server で共有されているデータベースからデータを取得するために FileMaker API for PHP を使用する方法
- 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 プロダクトドキュメンテーションの場所

- FileMaker Server ヘルプは FileMaker Server Admin Console の各ページで使用できます。ページの下までスクロールして [**ヘルプ**] をクリックしてください。
- FileMaker Server のマニュアルには Admin Console の各ページからアクセスできます。ページの下までスクロールして [**マニュアル**] をクリックしてください。
- Web 上でプロダクトドキュメンテーションセンターにアクセスします。

第 1 章
カスタム Web 公開の概要

FileMaker Server では、次の方法で FileMaker Pro 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。

FileMaker WebDirect: Claris FileMaker WebDirect™ を使用すると、データベースのレイアウトを素早く簡単に Web 上で公開することができます。互換性のある Web ブラウザソフトウェアを所有し、インターネットまたはイントラネットにアクセス可能な Web ユーザは、他のソフトウェアをインストールしなくても、FileMaker WebDirect ソリューションに接続してレコードを表示、編集、ソート、および検索することができます。ただし、その場合にはこれらの操作を行うためのアクセス権が必要となります。

FileMaker WebDirect を使用するには、ホストコンピュータで FileMaker Server を実行する必要があります。ユーザインターフェースは、FileMaker Pro デスクトップアプリケーションに似ています。Web ユーザが操作する Web ページおよびフォームは、FileMaker Pro データベースで定義されたレイアウトおよび表示形式によって変わります。FileMaker WebDirect ガイドを参照してください。

静的公開: データがあまり変更されない場合、または稼動中のデータベースにユーザが接続しないようにする場合には、静的な公開方法を使用します。静的な公開方法では、FileMaker Pro データベースからデータをエクスポートして Web ページを作成します。Web ページは、HTML を使用してさらにカスタマイズすることができます。データベースの情報が変更されても Web ページは変更されません。ユーザはデータベース自体には接続しません。(FileMaker WebDirect を使用すれば、データベースでデータが更新されると同時に、Web ブラウザ内のデータも更新されます)。FileMaker Pro ヘルプを参照してください。

FileMaker Data API: REST (Representational State Transfer) アーキテクチャの使用経験がある場合、Claris FileMaker プラットフォームに実装されている REST API を使用して共有ソリューションのデータに Web サービスでアクセスすることができます。Web サービスにより FileMaker Data API を呼び出して共有ソリューションにアクセスするための認証トークンを取得し、以降の呼び出しでそのトークンを使用してレコードの作成、更新、削除、および検索の実行を行います。FileMaker Data API は JSON (JavaScript Object Notation) でデータを返します。FileMaker Data API ガイドを参照してください。

カスタム Web 公開: FileMaker Pro データベースをカスタム Web サイトに統合するには、FileMaker Server で使用できるカスタム Web 公開テクノロジーを使用します。公開されるデータベースは FileMaker Server で共有され、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。

カスタム Web 公開では、次の操作を行うことができます:

- データベースを他の Web サイトに統合する
- ユーザによるデータの操作方法を決定する
- Web ブラウザでのデータの表示方法を制御する

FileMaker Server には、次の 2 つのカスタム Web 公開テクノロジーが備わっています:

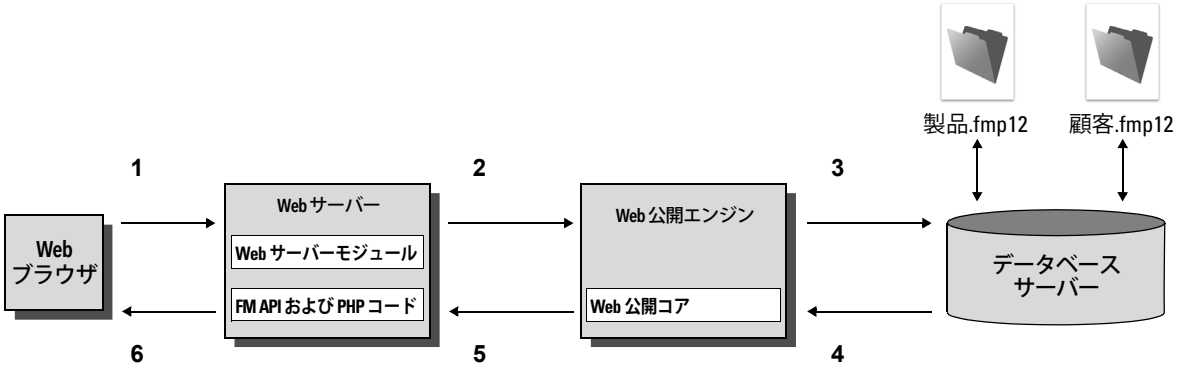
- カスタム Web 公開 with XML: XML データ公開を使用して、FileMaker データを他の Web サイトやアプリケーションと交換できます。FileMaker クエリーコマンドと引数とともに HTTP URL を使用することにより、FileMaker Server で共有されているデータベースに問い合わせて結果データを XML 形式でダウンロードし、結果として生成された XML データを任意の用途に使用できます。
- カスタム Web 公開 with PHP: FileMaker Pro データベースへのオブジェクト指向 PHP インターフェースを提供する FileMaker API for PHP を使用して、FileMaker データを PHP Web アプリケーションに統合することができます。PHP Web ページを自分でコーディングすることにより、ユーザインターフェースとユーザがデータと交信する方法を完全に管理できます。

Web 公開エンジンについて

FileMaker WebDirect およびカスタム 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が使用されています。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。

カスタム Web 公開 with XML: 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。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 データを返します。

カスタム Web 公開 with PHP: Web ユーザがカスタム Web 公開ソリューションにアクセスしている場合、FileMaker Server 上の PHP が Web 公開エンジンに接続し、FileMaker API for PHP を使用して応答します。



カスタム Web 公開のための FileMaker Server Web 公開エンジンの使用

Web 公開エンジンのリクエストの処理

1. リクエストが、Web ブラウザまたはアプリケーションから Web サーバーに送信されます。
2. Web サーバーが、FileMaker Web サーバーモジュールを使用してリクエストを Web 公開エンジンにルーティングします。
3. Web 公開エンジンが、データベースサーバーで共有されているデータベースにデータをリクエストします。
4. FileMaker Server が、リクエストされた FileMaker データを Web 公開エンジンに送信します。
5. Web 公開エンジンが、FileMaker データを変換してリクエストへの応答を行います。
   - PHP リクエストの場合、FileMaker API for PHP によって PHP リクエストが XML リクエストに変換されます。Web 公開エンジンでは XML リクエストを処理して、XML データを FileMaker API for PHP に戻します。次に、FileMaker API for PHP によって XML データが PHP アプリケーションで使用できる PHP オブジェクトに変換されます。
   - XML リクエストの場合、Web 公開エンジンは Web サーバーに XML データを直接送信します。
6. Web サーバーが、Web ブラウザまたはプログラムに出力を送信します。

重要 Web 上にデータを公開する場合は、セキュリティが重要になります。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。

カスタム Web 公開用のデータベースの準備については、第 2 章「データベースのカスタム Web 公開の準備」を参照してください。

カスタム Web 公開 with XML

XML を使用した FileMaker カスタム Web 公開では、FileMaker Server によって共有されている FileMaker Pro データベースに対してクエリーリクエスト送信して、結果のデータの表示、変更、または操作を行うことができます。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にエクスポートできます。

カスタム Web 公開 with PHP

FileMaker API for PHP には、FileMaker Pro データベースへのオブジェクト指向 PHP インターフェースが備わっています。FileMaker API for PHP を使用すると、FileMaker Pro データベースに保存されているロジックおよびデータの両方に対し、Web 上にアクセスして公開、または他のアプリケーションにエクスポートすることができます。また、API は、FileMaker Pro データベースに保存されているデータの抽出やフィルタを行うために、複雑で複合の検索コマンドをサポートしています。

PHP は元々、手続き型プログラミング言語として設計されており、オブジェクト指向の Web 開発言語として強化されています。PHP には、サイトのページ内でのロジックのほぼすべてのタイプを構築するためのプログラミング言語機能が備わっています。たとえば、条件付きロジック構築を使用して、ページ生成やデータルーティング、ワークフローを制御することができます。また、PHP はサイト管理とセキュリティも提供します。

XML と PHP の比較

以降のセクションでは、ユーザのサイトに最適なソリューションを決定するためのガイドラインの一部について説明します。

XML を選択する理由

- FileMaker XML リクエスト引数構文は、データベース操作用に設計され、ソリューション開発を簡略化します。
- XML は W3C スタンダードです。
- XML は、Unicode をサポートするコンピュータおよび人間が読み込み可能な形式であり、書き込まれた任意の言語でのデータ通信を可能にします。
- XML は、レコード、一覧、およびツリー構造データの表示に適しています。
- カスタム Web 公開を使用した XML データへのアクセス、および FileMaker Pro データベースからの XML のエクスポートには、FMPXMLRESULT を使用できます。

メモ　カスタム Web 公開 with XML の詳細については、第 3 章「カスタム Web 公開 with XML について」を参照してください。

PHP を選択する理由

- PHP はオブジェクト指向手続き型スクリプト言語として優れていますが、学習は比較的容易です。トレーニング、開発、およびサポート用に数多くのリソースを使用できます。
- FileMaker API for PHP を使用すると、FileMaker Pro データベースに保存されているロジックとデータに対し、Web 上にアクセスして公開、または他のアプリケーションにエクスポートすることができます。
- PHP では、条件付きロジックを使用して、ページ構築やフローを制御することができます。
- PHP には、サイトのページ上でさまざまなタイプのロジックを構築するためのプログラミング言語機能が備わっています。
- PHP は、最も知られている Web スクリプト言語の 1 つです。
- PHP はオープンソースの言語であり、php.net から利用できます。
- PHP を使用すると、さまざまな種類のサードパーティ製コンポーネントにアクセスして、ユーザのソリューションを統合することができます。

メモ　カスタム Web 公開 with PHP の詳細については、第 6 章「カスタム Web 公開 with PHP について」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必要があります。

データベースのカスタム Web 公開の有効化

公開する各データベースでカスタム Web 公開拡張アクセス権を有効にする必要があります。データベースでカスタム Web 公開拡張アクセス権を有効にしなかった場合、Web 公開エンジンをサポートするように構成されている FileMaker Server でデータベースが共有されていても、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. 使用するカスタム Web 公開拡張アクセス権を割り当てます:
   - カスタム Web 公開 with XML の場合は、fmxml を使用します。
   - カスタム Web 公開 with PHP の場合は、fmphp を使用します。
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を含むアクセス権セットを割り当てます。

メモ　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。

保護されたデータベースへのアクセス

カスタム Web 公開を使用すると、データベースのパスワード保護、データベースの暗号化、セキュリティ保護された接続によって公開したデータベースへのアクセスを制限できます。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 ユーザに認証をリクエストします。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は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サインインし、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 ブラウザからサイン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、より多くの機能を使用できる別の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。
- Web ユーザが有効なアカウント情報を入力すると、そのアカウント情報はブラウザセッションがタイムアウトしない限り再利用されます。ブラウザセッションがタイムアウトすると、Web ユーザは有効なアカウントの入力を再度求められます。

メモ デフォルトでは、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。[パスワード変更] スクリプトステップを使用してこの機能をデータベースに構築して、Web ユーザがブラウザから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

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。
- カスタム Web 公開拡張アクセス権は、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。
- 特定のデータベースのすべてのアクセス権セットの fmxml または fmphp 拡張アクセス権の選択を解除して、データベースのカスタム Web 公開拡張アクセス権を無効にします。FileMaker Pro ヘルプを参照してください。

- CLI (コマンドラインインターフェース) を使用して、すべてのカスタム Web 公開ソリューションのカスタム Web 公開を有効または無効にすることができます。また、FileMaker Server Admin Console を使用して、Web 公開エンジンを起動および停止することができます。FileMaker Server ヘルプ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 サーバーを構成します。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指定できます。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。

FileMaker Server では、ディスクに書き込むデータとクライアントに転送するデータの暗号化がサポートされています。

- FileMaker Pro のデータベース暗号化機能を使用してデータベースを暗号化します。暗号化によって、FileMaker Pro データベースファイルとディスクに書き込まれる一時ファイルが保護されます。FileMaker Server インストールおよび構成ガイドおよび FileMaker Pro ヘルプを参照してください。
  - FileMaker Server 上で共有されている暗号化されたデータベースは、Admin Console または CLI を使用して開きます。FileMaker Server 管理者としてデータベース暗号化パスワードを使用してファイルを開いて FileMaker クライアントが暗号化されたデータベースを使用できるようにします。
  - 暗号化された FileMaker Pro データベースが FileMaker Server 管理者によって暗号化パスワードを使用して開かれると、FileMaker クライアントは暗号化パスワードを入力することなく暗号化されたデータベースにアクセスできます。暗号化されたデータベースを開く方法については、FileMaker Server ヘルプを参照してください。
- Web サーバーと Web ブラウザの間の通信に SSL (Secure Sockets Layer) 暗号化を使用します。SSL 接続は HTTPS 接続でアクセスされます。FileMaker Server は、Claris International Inc. によって署名された標準の SSL 証明書を提供しますが、サーバー名の検証は行われません。この FileMaker デフォルト証明書はテスト用にのみ利用できます。実際に運用環境で使用する場合はカスタム SSL 証明書が必要です。FileMaker Server インストールおよび構成ガイドを参照してください。

  カスタム SSL 証明書をインポートすると、データベースサーバークライアント接続で SSL が使用され、HTTP 接続が HTTPS にルーティングされます。ご使用のカスタム SSL 証明書とともに、サイトの PHP ファイルを共有するための HTTPS ディレクトリを使用してください。第 7 章「カスタム Web 公開 with PHP の一般手順」を参照してください。

データベースのセキュリティ保護については、FileMaker セキュリティガイドを参照してください。

メモ セキュリティ上の理由から、他の Web サーバーがホストする Web ページでは、カスタム Web 公開コンテンツの埋め込みに <iframe> タグを使用することはできません。別の Web ページの <iframe> タグにカスタム Web 公開コンテンツを埋め込む場合は、それらの Web ページが FileMaker Server Web サーバーによってホストされている必要があります。

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。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

オブジェクトフィールドの内容は、データベースに埋め込むか、相対パスを使用した参照でリンクさせるか、または外部に保存できます。

データベースに埋め込まれたオブジェクトフィールドのオブジェクト

FileMaker Pro 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 上で適切に共有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。27 ページの「XML ソリューション内の FileMaker Pro 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。

保存されたファイル参照を含むオブジェクトフィールド

オブジェクトフィールドにファイル参照が保存されている場合は、次の手順に従って Web 公開エンジンを使用して参照先ファイルを公開する必要があります。

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 (Windows):
     `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥HTTPServer¥conf`
     [ドライブ] は展開した FileMaker Server の Web 公開エンジンコンポーネントが格納されているドライブです。
   - Apache (macOS): /ライブラリ/`FileMaker Server/HTTPServer/htdocs`

メモ 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 サーバーが構成されている必要があります。インターネットに対して登録されている最新の MIME 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。Web サーバーのマニュアルを参照してください。

外部に保存されたデータを含むオブジェクトフィールド

オブジェクトフィールドがオブジェクトを外部に保存している場合 (FileMaker Pro のフィールドのオプションダイアログボックスで [**オブジェクトデータを外部に保存**] を選択した場合)、FileMaker Pro を使用して、クライアントファイルシステムから FileMaker Server へデータベースファイルを移動します。FileMaker Pro を使用してデータベースをアップロードする場合、外部に保存されたオブジェクトフィールドデータは、プロセスの一環として FileMaker Server にアップロードされます。FileMaker Server へのデータベースファイルの転送については、FileMaker Pro ヘルプを参照してください。

外部に保存されたオブジェクトを含むオブジェクトフィールドを使用しているデータベースを手動でアップロードする場合、次の操作を行って外部に保存されたオブジェクトを Web 公開エンジンを使用して公開する必要があります。

データベースを手動でアップロードするには:

1. データベースファイルをサーバー上の適切な場所に配置します。FileMaker Server で開く FileMaker Pro データベースファイルか、またはそれらのファイルへのショートカット (Windows) またはエイリアス (macOS) を次のフォルダに配置します:
   - Windows:
     `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥Data¥Databases¥`
     `[ドライブ]` はシステムが起動されるプライマリドライブです。
   - macOS: `/ライブラリ/FileMaker Server/Data/Databases/`

   または、オプションで指定した追加のデータベースフォルダにファイルを配置することもできます。
2. データベースを配置したフォルダ内に、「RC_Data_FMS」という名前のフォルダを作成します (存在していない場合)。
3. 「RC_Data_FMS」フォルダの中に、データベース名と同じ名前のフォルダを作成します。たとえば、データベース名が Customers の場合は、Customers というフォルダを作成します。作成した新しいフォルダに、外部に保存されたオブジェクトを配置します。

   メモ データベースが FileMaker Server 上で共有されている場合は、複数のデータベース間で共通のオブジェクトのフォルダを共有する方法はありません。各データベースのオブジェクトは、データベース名と同じ名前で識別されたフォルダにある必要があります。
4. macOS から共有するファイルでは、**fmsadmin** グループに属するようにファイルを変更します。

データベースの手動によるアップロードについては、FileMaker Server ヘルプを参照してください。

オブジェクトフィールドとプログレッシブダウンロード

Web 公開エンジンは、インタラクティブオブジェクトのオーディオファイル (.mp3)、ビデオファイル (.mov、.mp4、.avi を推奨) および PDF ファイルのプログレッシブダウンロードをサポートしています。たとえば、Web ユーザは、ムービーファイルが完全にダウンロードされる前にムービーの再生を開始できます。プログレッシブダウンロードを可能にするには、ストリーミングをサポートするか、Web での表示に最適化するオプションを使用してファイルを作成する必要がある場合があります。たとえば、PDF ファイルは、Web 表示用に最適化するオプションを使用して作成します。

カスタム SSL 証明書をインポートすると、データベースサーバークライアント接続で SSL が使用され、HTTP 接続が HTTPS にルーティングされます。FileMaker Server はセキュア接続を使用して HTTPS でデータを送信します。

- インタラクティブオブジェクトデータは HTTPS を使用してダウンロードされます。
- 一時キャッシュファイルは作成されず、データは転送中に暗号化されるため、データは共有されたソリューションがローカルデータベースであるかのように安全です。

カスタム SSL 証明書がない場合、FileMaker Server でデータ転送に使用する接続は転送中に暗号化されず、データは HTTP を使用して送信されます。

- FileMaker クライアントは、最小の遅延の後インタラクティブオブジェクトデータを認識します。
- FileMaker Server は、FileMaker Pro、Claris FileMaker Go®、または Web クライアントがデータをリクエストしたときに、サーバー上のキャッシュフォルダにオブジェクトフィールドデータをキャッシュします。データは、FileMaker Server が定期的にキャッシュフォルダを空にするまで、サーバーのキャッシュフォルダに 2 時間キャッシュされたままになることがあります。データはクライアント上でローカルにはキャッシュされません。

Web ユーザがオブジェクトフィールドのデータを表示する方法

Web 公開エンジンを使用してデータベースを公開する場合、オブジェクトフィールドのオブジェクトには次の制限が適用されます:

- 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。Web ユーザがオブジェクトフィールドを使用してオブジェクトをデータベースにアップロードすることはできません。
- サムネールを有効にしたオブジェクトフィールドを使用するデータベースの場合、Web 公開エンジンはサムネールではなく完全なファイルをダウンロードします。

FileMaker スクリプトとカスタム Web 公開

FileMaker Pro のスクリプトの管理機能を使用すると、頻繁に実行されるタスクの自動化や、複数のタスクの結合が可能となります。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。

FileMaker プラットフォームは多くのスクリプトステップをカスタム Web 公開でサポートしています。URL にクエリー文字列でスクリプトを使用すると、Web ユーザはさまざまな自動化タスクを実行できます。カスタム Web 公開がサポートするスクリプトステップを表示するには、FileMaker Pro のスクリプトワークスペースウインドウで [**互換性**] ボタンをクリックして [**カスタム Web 公開**] を選択します。グレー表示されていないスクリプトステップがカスタム Web 公開でサポートされています。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、動作が異なるものもあります。20 ページの「カスタム Web 公開ソリューションでのスクリプト動作」を参照してください。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サインインして、すべてのクライアントに対して期待どおりに動作することを確認します。スクリプト関連のエラーについて、Web 公開エンジンのログファイル (wpe.log) を確認します。97 ページの「Web 公開エンジンのログの使用」を参照してください。

次のヒントおよび考慮事項に注意してください:

- スクリプトが返す可能性がある値を考慮してください。返されるすべてのデータを処理できるように準備します。FileMaker Pro では、スクリプトがテーブルまたは現在の対象レコードからすべてのレコードを返す場合があります。しかし、スクリプトがテーブルからすべてのレコードを返すと、Web アプリケーションではこれらのレコード処理によってメモリが不足する場合があります。返されるレコード数を制限するには、XML クエリーでの `-max` クエリー引数、または PHP クエリーでの `setRange()` メソッドの使用を検討してください。
- 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サイン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。
- スクリプトワークスペースウインドウで、完全なアクセス権をスクリプトに付与して、個人ユーザにアクセスを付与しないタスクの実行を許可します。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。
- スクリプトでカスタム Web 公開および FileMaker WebDirect ソリューションのプラグインをインストールできるようにするには、FileMaker Server Admin Console を使用して必要な設定を有効にします。[**Web 公開プラグイン**] で [**Web 公開プラグイン**] および [**[プラグインファイルのインストール] スクリプトステップ**] を [**有効**] に設定します。スクリプトが Web 公開ソリューションのプラグインをインストールできないようにするには、この設定を [**無効**] にします。
- 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] スクリプト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、条件付きの値一覧のような機能では、値一覧フィールドに結果を表示する前に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。

- データの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ため、データを変更するスクリプトには [レコード/検索条件確定] スクリプト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] ステップを含むスクリプト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します。
- データを変更しても確定していない場合、データに依存するスクリプトは FileMaker Pro クライアントの操作と連動しない場合があります。たとえば、フィールドに Get (変更されたフィールド) 計算式が含まれている場合、XML クエリーまたは PHP 呼び出しでレコードが編集されると、Get (変更されたフィールド) 計算式は空の一覧を返します。ただし、カスタム Web 公開ソリューションでレコードを編集する FileMaker スクリプトが動作している場合、Get (変更されたフィールド) 計算式はスクリプトによって編集されたフィールドの一覧を返します。
- クライアントのタイプに基づく条件付きの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」という文字列が含まれる場合、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。
- 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、カスタム Web 公開ソリューションとしてデータベースを共有するときにスクリプトが適切に実行されることを確認します。上記で説明するように、カスタム Web 公開用にサポートされているスクリプトステップだけがスクリプトで使用されることを確認します。

カスタム Web 公開ソリューションでのスクリプト動作

カスタム Web 公開ソリューションと FileMaker Pro では、一部のスクリプトステップの機能が異なる場合があります。互換性の情報については、FileMaker Pro ヘルプを参照してください。

他の FileMaker Pro ファイルが同じ FileMaker Server のインストールで共有されていない場合、および他の FileMaker Pro ファイルで同じカスタム Web 公開拡張アクセス権が有効になっていない場合、カスタム Web 公開ソリューションのスクリプトはこれらのファイルのスクリプトを実行できません。

スクリプトトリガとカスタム Web 公開ソリューション

FileMaker Pro では、スクリプトとユーザの操作 (ユーザによるフィールドのクリックなど) の両方でスクリプトトリガをアクティブにすることができます。ただし、カスタム Web 公開では、スクリプトでのみアクティブにすることができます。

スクリプトトリガの詳細については、FileMaker Pro ヘルプを参照してください。

メモ カスタム Web 公開ソリューションでは、OnFirstWindowOpen スクリプトトリガはアクティブになりません。OnLastWindowClose スクリプトトリガはスクリプトで最後の仮想ウインドウが閉じられた場合にのみアクティブになります。XML –script クエリー引数または PHP の `newPerformScriptCommand()` メソッドを使用してスクリプトを手動で実行することができます。

第 3 章
カスタム Web 公開 with XML について

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 データを他の Web サイトと他のミドルウェア、およびカスタムアプリケーションと統合することができます。Web ブラウザに FileMaker Pro 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:** FileMaker Server の管理者は、サーバーで共有されているすべてのデータベースに対して XML Web 公開を有効または無効にすることができます。FileMaker Pro データベースの所有者として、各データベースに対して XML Web 公開への Web ユーザアクセスを制御できます。
- **公開されるデータの制御とフィルタ:** 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 の知識がある場合、使用する URL 構文やクエリー引数など、カスタム Web 公開 with XML に特有の詳細事項をいくつか学べば、ソリューションの開発に取りかかることができます。

カスタム Web 公開 with XML では、FileMaker Pro データベースからデータを取得して、そのデータを別の出力形式で簡単に使用できます。適切なクエリーコマンドと引数を指定した HTTP リクエストを使用することで、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することができます。29 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。

カスタム Web 公開 with XML の主な機能

XML を使用した FileMaker Server カスタム Web 公開では、多くの重要な機能が提供されています:

- データベースは FileMaker Server 上で共有され、FileMaker Pro が実行されている必要はありません。
- JavaScript を使用した XML のサーバーサイドでの処理を使用することができます。
- 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。14 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 ユーザは、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。FileMaker プラットフォームは多くのスクリプトステップをカスタム Web 公開でサポートしています。18 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。
- FileMaker スクリプトには、引数値を渡すことができます。60 ページの「-script.param (スクリプトに引数を渡す) クエリー引数」、61 ページの「-script.prefind.param (検索前にスクリプトに引数を渡す) クエリー引数」、および62 ページの「-script.presort.param (ソート前にスクリプトに引数を渡す) クエリー引数」を参照してください。
- `fmresultset` XML 文法では、名前でフィールドにアクセスして、`relatedset` (ポータル) のデータを操作できます。
- 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。第 5 章「XML 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。

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

カスタム Web 公開 with XML 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:

- 次を含む FileMaker Server 展開:
  - Microsoft IIS (Windows) または Apache (macOS) のいずれかの Web サーバー
  - CLI を使用してカスタム Web 公開用に有効にされた FileMaker データベースサーバー
  - インストールおよび構成されている Web 公開エンジン
- FileMaker Server で共有されている 1 つ以上の FileMaker Pro データベース
- 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
- 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

FileMaker Server インストールおよび構成ガイドを参照してください。

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 ユーザがカスタム Web 公開 with XML 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
- 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が必要です。

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、データベースを共有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してください。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。
- FileMaker Server 展開の一部である Web サーバー用のホストコンピュータには、固有の静的 (不変) 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 アドレスでは、データベースの検索が困難になります。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確認してください。

この後の作業を開始するにあたって

カスタム Web 公開ソリューションの開発を開始するための推奨事項は次のとおりです:

- カスタム Web 公開を有効にするには、CLI を使用します。FileMaker Server ヘルプを参照してください。
- 公開する各 FileMaker Pro 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。13 ページの「データベースのカスタム Web 公開の有効化」を参照してください。
- XML を使用して FileMaker Pro データベースのデータにアクセスする方法については、29 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。

第 4 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) 形式で取得または更新することができます。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。

カスタム Web 公開 with XML 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、カスタム Web 公開 with XML に特有の詳細事項をいくつか習得することで、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 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 によって共有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。

ナレッジベースを参照してください。

メモ　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 XML インポートおよびエクスポートの比較

Web 公開エンジンと FileMaker Pro のどちらを使用しても、FileMaker Pro 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:

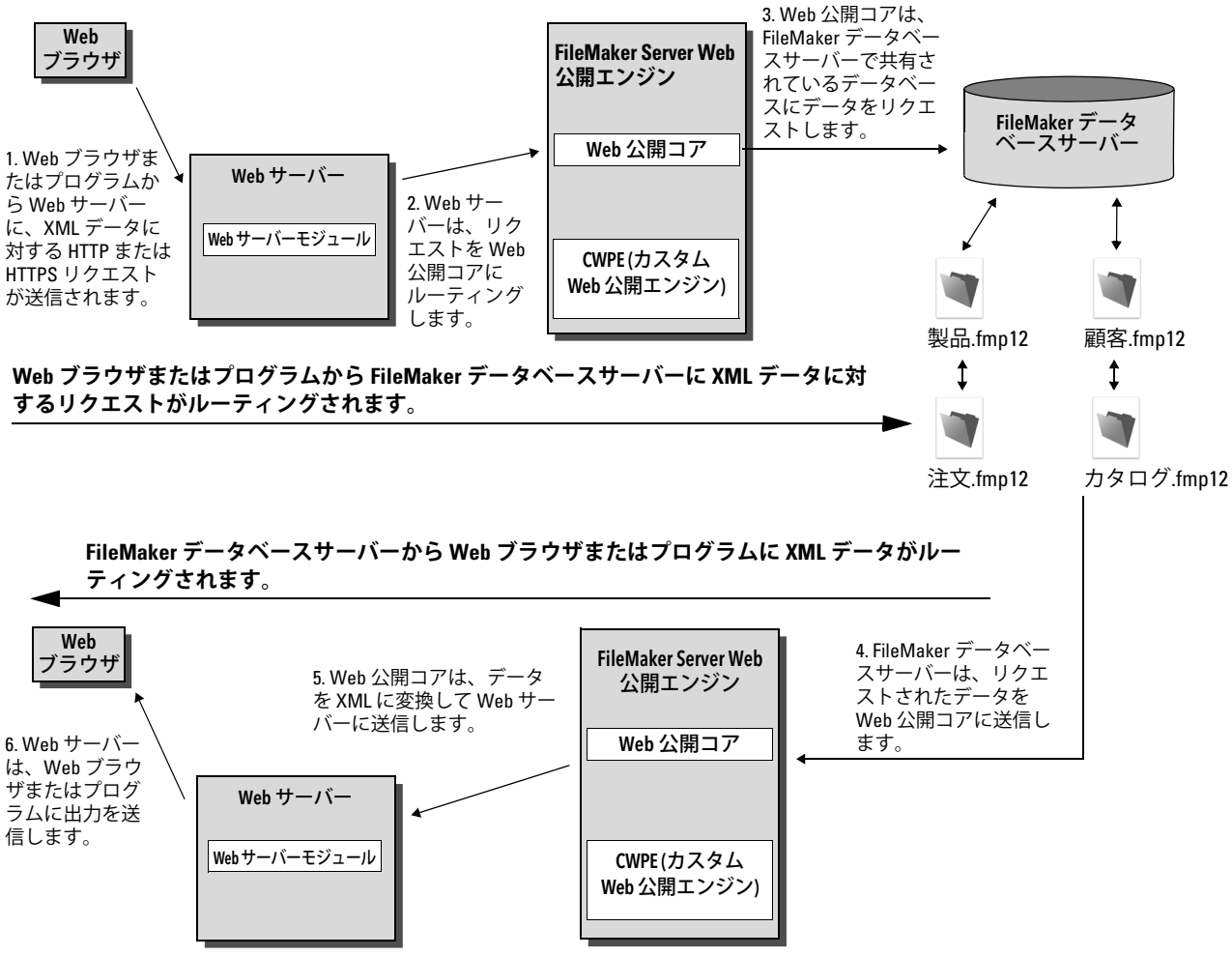
- XML データにアクセスするために、Web 公開エンジンでは、`fmresultset`、`FMPXMLRESULT`、および `FMPXMLLAYOUT` 文法がサポートされています。FileMaker Pro では、XML のインポートには `FMPXMLRESULT` 文法、エクスポートには `FMPXMLRESULT` 文法が使用されます。29 ページの「Web 公開エンジンを使用した XML データへのアクセス」を参照してください。
- Web 公開エンジンで XML データにアクセスするには、URL で Web 公開エンジンのクエリー文字列を使用します。FileMaker Pro 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 Pro 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 Pro 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 を使用して 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。

メモ 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 Pro 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 Pro 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. CLI を使用して XML 公開が有効になっていることを確認します。FileMaker Server ヘルプを参照してください。

2. 公開する各 FileMaker Pro 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 公開に対する 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。13 ページの「データベースのカスタム 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して出力されます。XML データ順序を画面上に表示されるフィールドの順序 (上から下、左から右) と一致させるには、すべてのフィールドを選択し、グループ化してからグループ解除します。この手順によって、画面の順序と一致したレイアウト順序にリセットされます。

3. FileMaker Server Admin Console で、公開する各 FileMaker Pro データベースの 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。
   - Admin Console で、[**データベース**] ページをクリックします。
   - [**すべてのデータベース**] の横のメニューセレクタから [**拡張アクセス権を表示**] を選択します。fmxml 拡張アクセス権が有効になっているデータベースには **FMXML** という文字が表示されます。

4. 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については、39 ページの「FileMaker クエリー文字列を使用した XML データリクエスト」および第 5 章「XML 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。

5. 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。

6. 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。

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 Pro 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。

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 Pro 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。可能な値は `fmresultset`、`FMPXMLRESULT`、または `FMPXMLLAYOUT` です。30 ページの「fmresultset 文法の使用」および 34 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 公開に使用する 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。) 39 ページの「FileMaker クエリー文字列を使用した XML データリクエスト」および第 5 章「XML 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。

メモ 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` と `FMPXMLLAYOUT` の 2 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、44 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。

例

```
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 Pro 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をデータベースに格納する場合

オブジェクトフィールドの <data> 要素は、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。

例

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

メモ 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。カッコ内の数字は、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。46 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。

例

```
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で実際のオブジェクトの代わりにファイル参照を格納する場合

オブジェクトフィールドの <data> 要素は、オブジェクトを参照する相対パスを含みます。

例

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

メモ 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。16 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

オブジェクトフィールドの <data> 要素は空になります。

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。39 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。

例

「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 の例では、%C3%A9 は「é」文字を UTF-8 で URL エンコードしたものです。

URL の仕様については、www.w3.org 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 公開エンジンを使用して 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:** これは、Web 公開エンジンが XML データにアクセスする際に推奨される文法です。この文法は柔軟で、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えるように最適化されています。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。Web 公開を効率的に実行できるように、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。30 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。34 ページの「他の FileMaker XML 文法の使用」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。

メモ 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。39 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> 要素 (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> 要素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 Pro 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要素で囲んで返します。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>
FMPXMLLAYOUT	<ERRORCODE>0</ERRORCODE>

XML ドキュメントのエラーコード要素は、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。付録 A「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

fmresultset 文法の使用

この文法では、XML 要素名に 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 ドキュメントをリクエストする URL で、fmresultset 文法の次の名前を指定します:

fmresultset.xml

例

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

メモ `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 文法の要素の説明

`fmresultset` 文法は主に、`<datasource>` 要素、`<metadata>` 要素、および `<resultset>` 要素からなります。

<datasource> 要素

`fmresultset` 文法では、`<datasource>` 要素に、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>` 要素の date-format 属性で指定されます:
  `MM/dd/yyyy`

各要素の意味は次のとおりです:

- `MM` - 月を表す 2 桁の値 (01 から 12 まで。たとえば 01 は 1 月で 12 は 12 月です)。
- `dd` - 日を表す 2 桁の値 (00 から 31 まで)。
- `yyyy` - 年を表す 4 桁の値。

- XML ドキュメントの時刻の書式は、`<datasource>` 要素の 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` - 時間を表す 2 桁の値 (24 時間形式の 00 から 23 まで)。
- `mm` - 分を表す 2 桁の値 (00 から 59 まで)。
- `ss` - 秒を表す 2 桁の値 (00 から 59 まで)。

- `<datasource>` 要素の timestamp-format 属性によって、date-format と time-format の形式が 1 つのタイムスタンプに結合されます:
  `MM/dd/yyyy HH:mm:ss`

<metadata> 要素

`fmresultset` 文法の `<metadata>` 要素には、1 つまたは複数の `<field-definition>` および `<relatedset-definition>` 要素が含まれ、各要素には、結果セットのフィールドの 1 つの属性が含まれます。

<field-definition> 属性では、次が指定されます:

- フィールドが「auto-enter」フィールドであるかどうか (「yes」または「no」)
- フィールドが「four-digit-year」フィールドであるかどうか (「yes」または「no」)
- フィールドが「global」フィールドであるかどうか (「yes」または「no」)
- 繰り返し値の最大数 (max-repeat 属性)
- 入力を許可する最大文字数 (max-characters 属性)
- 「not-empty」フィールドであるかどうか (「yes」または「no」)
- 「numeric-only」データであるかどうか (「yes」または「no」)
- 結果 (「text」、「number」、「date」、「time」、「timestamp」、または「container」)
- 「time-of-day 」フィールドであるかどうか (「yes」または「no」)
- タイプ (「normal」、「calculation」、または「summary」)
- フィールド名 (必要に応じて完全修飾名)

<relatedset-definition> 要素は、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> 要素に含まれる <field-definition> 要素で表されます。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> 要素内にまとめられます。

<resultset> 要素

<resultset> 要素には、クエリーの結果として返された <record> 要素と、見つかったレコードの総数の属性が格納されます。各 <record> 要素は、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> 要素で構成されます。

ポータル内の各レコードは、<relatedset> 要素内の <record> 要素で表されます。<relatedset> 要素の 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

例

```
<fmresultset xmlns="http://www.filemaker.com/xml/fmresultset" version="1.0">
  <error code="0"/>
  <product build="03/29/2019" name="FileMaker Web Publishing Engine"
   version="18.0.1.0"/>
  <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="12"/>
  <metadata>
    <field-definition auto-enter="no" four-digit-year="no" global="no" max-
     repeat="1" name="Title" not-empty="no" numeric-only="no" result="text"
     time-of-day="no" type="normal"/>
    <field-definition auto-enter="no" four-digit-year="no" global="no" max-
     repeat="1" name="Artist" not-empty="no" numeric-only="no" result="text"
     time-of-day="no" type="normal"/>
    <field-definition auto-enter="no" four-digit-year="no" global="no" max-
     repeat="1" name="Style" not-empty="no" numeric-only="no" result="text"
     time-of-day="no" type="normal"/>
    <field-definition auto-enter="no" four-digit-year="no" global="no" max-
     repeat="1" name="length" not-empty="no" numeric-only="no"
     result="number" time-of-day="no" type="calculation"/>
    <relatedset-definition table="artlocations">
      <field-definition auto-enter="no" four-digit-year="no" global="no" max-
       repeat="1" name="artlocations::Location" not-empty="no" numeric-
       only="no" result="text" time-of-day="no" type="normal"/>
      <field-definition auto-enter="no" four-digit-year="no" global="no" max-
       repeat="1" name="artlocations::Date" not-empty="no" numeric-only="no"
       result="date" time-of-day="no" type="normal"/>
    </relatedset-definition>
  </metadata>
  <resultset count="1" fetch-size="1">
    <record mod-id="6" record-id="14">
      <field name="Title">
        <data>Spring in Giverny 3</data>
      </field>
      <field name="Artist">
        <data>Claude Monet</data>
      </field>
      <field name="Style">
        <data/>
      </field>
      <field name="length">
        <data>19</data>
      </field>
      <relatedset count="0" table="artlocations"/>
    </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 ドキュメントをリクエストする URL で、次の文法名を指定します:

FMPXMLRESULT.xml

例

```
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 ドキュメントをリクエストする 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

例

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

メモ 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 文法の要素の説明

FMPXMLRESULT 文法では、<DATABASE> 要素に、NAME、RECORDS、DATEFORMAT、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> 要素の 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> 要素の 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。31 ページの「fmresultset 文法の要素の説明」を参照してください。

FMPXMLRESULT 文法の <METADATA> 要素には 1 つまたは複数の <FIELD> 要素が含まれ、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、TIMESTAMP、および CONTAINER です。

<RESULTSET> 要素には、クエリーの結果として返されたすべての <ROW> 要素と、見つかったレコードの総数の属性が格納されます。各 <ROW> 要素には、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 (57 ページの「-modid (修正 ID) クエリー引数」を参照)、および <COL> 要素が含まれます。<COL> 要素には、行内の 1 つのフィールド/列のデータが含まれ、複数の <DATA> 要素は、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

例

```
<FMPXMLRESULT xmlns="http://www.filemaker.com/fmpxmlresult">
  <ERRORCODE>0</ERRORCODE>
  <PRODUCT BUILD="03/29/2019" NAME="FileMaker Web Publishing Engine"
 VERSION="18.0.1.0"/>
  <DATABASE DATEFORMAT="MM/dd/yyyy" LAYOUT="web" NAME="art" RECORDS="12"
 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="Image" TYPE="CONTAINER"/>
  </METADATA>
  <RESULTSET FOUND="1">
    <ROW MODID="7" RECORDID="4">
      <COL>
        <DATA>Village Market</DATA>
      </COL>
      <COL>
        <DATA>Camille Pissarro</DATA>
      </COL>
      <COL>
        <DATA>/fmi/xml/cnt/Untitled.pct?-db=art&-lay=web&-recid=4
         &-field=Image(1)
        </DATA>
      </COL>
    </ROW>
  </RESULTSET>
</FMPXMLRESULT>
```

<COL> 要素の順序は、<METADATA> 要素内の <FIELD> 要素の順序に一致します。たとえば、<METADATA> 要素に「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> 要素に一覧表示されます。

FMPXMLLAYOUT 文法の要素の説明

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> 要素が <LAYOUT> 要素に含まれます。各 <FIELD> 要素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 属性がこの要素に含まれます。

<VALUELISTS> 要素には、レイアウトにある各値一覧の <VALUELIST> 要素が 1 つまたは複数含まれます (各要素には、値一覧の名前と、一覧の各値の <VALUE> 要素が含まれます)。

FileMaker Pro データベースの [**値一覧に使用するフィールドの指定**] ダイアログボックスで選択したオプションに応じて <VALUE> 要素は、最初のフィールドのみ、2 番目のフィールドのみ、または値一覧の両方のフィールドを含む DISPLAY 属性を含みます。たとえば、値一覧内の最初のフィールドがアートスタイルの ID 番号 ("100"など) を格納し、2 番目のフィールドがアートスタイルの関連付けられた名前 ("Impressionism" など) を表示すると仮定します。[**値一覧に使用するフィールドの指定**] ダイアログで各種組み合わせのオプションが選択されたときの DISPLAY 属性についての概要は次のようになります:

- [**2 番目のフィールドの値も表示**] が選択されなかった場合、DISPLAY 属性は、値一覧の最初のフィールドの値のみを含みます。

例

DISPLAY 属性はアートスタイルの ID 番号のみを含みます。

```
<VALUELISTS>
  <VALUELIST NAME="style">
    <VALUE DISPLAY="100">100</VALUE>
    <VALUE DISPLAY="101">101</VALUE>
    <VALUE DISPLAY="102">102</VALUE>
  </VALUELIST>
</VALUELISTS>
```

- [**2 番目のフィールドの値も表示**] と [**2 番目のフィールドの値のみを表示**] の両方が選択された場合、DISPLAY 属性は、2 番目のフィールドの値のみを含みます。

例

DISPLAY 属性はアートスタイルの名前のみを含みます。

```
<VALUELISTS>
    <VALUELIST NAME="style">
      <VALUE DISPLAY="Impressionism">100</VALUE>
      <VALUE DISPLAY="Cubism">101</VALUE>
      <VALUE DISPLAY="Abstract">102</VALUE>
    </VALUELIST>
</VALUELISTS>
```

- [**2 番目のフィールドの値も表示**] が選択され、[**2 番目のフィールドの値のみを表示**] が選択されなかった場合、DISPLAY 属性は、値一覧の両方のフィールドの値を含みます。

例

DISPLAY 属性はアートスタイルの ID 番号と名前の両方を含みます。

```
<VALUELISTS>
  <VALUELIST NAME="style">
      <VALUE DISPLAY="100 Impressionism">100</VALUE>
      <VALUE DISPLAY="101 Cubism">101</VALUE>
      <VALUE DISPLAY="102 Abstract">102</VALUE>
  </VALUELIST>
</VALUELISTS>
```

日付、時刻、およびタイムスタンプのフィールドの場合は、そのフィールドタイプに「fm」の書式を使用して、値一覧のデータを書式設定します。「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。たとえば、「birthdays」値一覧をレイアウトの「birthdate」フィールドでのポップアップメニューに使用し、その「birthdate」フィールドが日付タイプである場合には、その値一覧の値出力はすべて「fm」日付の書式になります。

メモ レイアウト上で異なるフィールドタイプである 2 つのフィールドが同じ値一覧を共有している場合には、その値一覧データの書式は、1 番目のフィールドのタイプによって決まります。

FMPXMLLAYOUT 文法での XML データ

例

```
<FMPXMLLAYOUT xmlns="http://www.filemaker.com/fmpxmllayout">
  <ERRORCODE>0</ERRORCODE>
  <PRODUCT BUILD="03/29/2019" NAME="FileMaker Web Publishing Engine"
 VERSION="18.0.1.0"/>
  <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="artlocations::Location">
      <STYLE TYPE="EDITTEXT" VALUELIST=""/>
    </FIELD>
    <FIELD NAME="artlocations::Date">
      <STYLE TYPE="EDITTEXT" VALUELIST=""/>
    </FIELD>
    <FIELD NAME="Style">
      <STYLE TYPE="POPUPMENU" VALUELIST="style"/>
    </FIELD>
  </LAYOUT>
  <VALUELISTS>
    <VALUELIST NAME="style">
      <VALUE DISPLAY="Impressionist">Impressionist</VALUE>
      <VALUE DISPLAY="Modern">Modern</VALUE>
      <VALUE DISPLAY="Abstract">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 (Unicode Transformation 8 Bit) 形式で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。

メモ 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 Unicode 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 文字列は適合する Unicode 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 データリクエスト

FileMaker Pro データベースに XML データをリクエストするには、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。

例

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 Pro データベースに含まれるすべての製品の一覧をリクエストできます。

```
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。

また、URL でクエリーコマンドと引数を使用することもできます。

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については、43 ページの「XML 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。

使用するクエリーコマンド名	実行するコマンド
-dbnames	すべての共有データベースおよび Web 共有データベースの名前の取得
-delete	レコード削除
-dup	レコード複製
-edit	レコードの編集
-find	1 つまたは複数のレコードの検索
-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
-findquery	複雑または複合検索条件の実行
-layoutnames	共有データベースおよび 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	新規レコードの追加
-scriptnames	共有データベースおよび 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 以外のすべてのクエリーコマンドで必須です。
-delete.related	-edit のオプションです。
-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。27 ページの「XML ソリューション内の FileMaker Pro 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 ではオプションです。53 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、および -scriptnames を除くすべてのクエリーコマンドで必須です。
-lay.response（XML 応答に対するレイアウトの切り替え）	-dbnames、-layoutnames、および -scriptnames を除く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、-findall、および -findquery のオプションです。
-modid（修正 ID）	-edit のオプションです。
-query	複合検索条件 -findquery で必須です。
-recid（レコード ID）	-edit、-delete、および -dup で必須です。-find 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
-relatedsets.filter	-find、-findall、-findany、-edit、-new、-dup、および -findquery のオプションです。
-relatedsets.max	-find、-edit、-new、-dup、および -findquery のオプションです。
-script（スクリプトの実行）	-find、-findall、-findany、-new、-edit、-delete、-dup、-view、および -findquery のオプションです。
-script.param（-script によって指定されたスクリプトに引数値を渡します）	-script および -findquery 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、-findall、および -findquery のオプションです。
-script.prefind.param（-script.prefind によって指定されたスクリプトに引数値を渡します）	-script.prefind および -findquery のオプションです。
-script.presort（ソートの前にスクリプト実行）	-find、-findall、および -findquery のオプションです。
-script.presort.param（-script.presort によって指定されたスクリプトに引数値を渡します）	-script.presort および -findquery のオプションです。
-skip（レコードのスキップ）	-find、-findall、および -findquery のオプションです。
-sortfield.[1-9]（フィールドのソート）	-find、-findall、および -findquery 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。

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 データをリクエスト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。

例

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。これには「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. クエリーで指定されたグローバルフィールド値を指定します (URL の「`.global=`」部分)。
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. `-script` クエリー引数を処理します (指定されている場合)。
8. `-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)。
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。

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。13 ページの「データベースのカスタム Web 公開の有効化」を参照してください。
- データベースは、FileMaker Server 展開のデータベースサーバーコンポーネントで共有され、FileMaker Server によって開かれている。FileMaker Server 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。
- FileMaker Server 展開の Web サーバーコンポーネントが実行されている。
- FileMaker Server 展開の Web 公開エンジンコンポーネントが実行されている。
- CLI を使用して XML 公開が有効にされている。FileMaker Server ヘルプを参照してください。

第 5 章
XML クエリー文字列で使用される有効な名前

この章では、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 XML 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 (49 ページを参照)	-db (52 ページを参照)
-delete (49 ページを参照)	-field (53 ページを参照)
-dup (49 ページを参照)	フィールド名 (53 ページを参照)
-edit (49 ページを参照)	フィールド名.op (54 ページを参照)
-find、-findall、-findany (50 ページを参照)	-lay (55 ページを参照)
-findquery (50 ページを参照)	-lay.response (55 ページを参照)
-layoutnames (51 ページを参照)	-lop (56 ページを参照)
-new (51 ページを参照)	-max (56 ページを参照)
-scriptnames (52 ページを参照)	-modid (57 ページを参照)
-view (52 ページを参照)	-query (57 ページを参照)
	-recid (58 ページを参照)
	-relatedsets.filter (59 ページを参照)
	-relatedsets.max (60 ページを参照)
	-script (60 ページを参照)
	-script.param (60 ページを参照)
	-script.prefind (61 ページを参照)
	-script.prefind.param (61 ページを参照)
	-script.presort (62 ページを参照)
	-script.presort.param (62 ページを参照)
	-skip (62 ページを参照)
	-sortfield.[1-9] (63 ページを参照)
	-sortorder.[1-9] (63 ページを参照)

重要 -dbnames、-layoutnames、および -scriptnames 以外のすべてのクエリーコマンドでは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。

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` を含めることができますが、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` 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、39 ページの「FileMaker クエリー文字列を使用した XML データリクエスト」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。
- Web 公開エンジンは予約語をすべて小文字に変換します。その中には、特殊な値が予想されるクエリーコマンド、クエリー引数およびコマンド値が含まれます (例: `-lop=and`, `-lop=or`, `-sortorder=ascend`, `-sortorder=descend`, `-max=all`)。
- 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、レイアウト名 `MyLayout` を指定するために `-lay=mylayout` を使用できます。
- フィールド名にピリオドやカッコを使用しないことをお勧めします。ピリオドを含むフィールド名が機能することもありますが、次の例外を含むフィールド名を使用することはできません:
  - ピリオドは、数字の前に置くことはできません。たとえば、「`myfield.9`」のフィールド名は無効です。
  - ピリオドは、文字列「`op`」 (2 文字の「op」) の前に置くことはできません。たとえば、「`myfield.op`」のフィールド名は無効です。
  - ピリオドは、文字列「`global`」 (「global」という文字) の前に置くことはできません。たとえば、「`myfield.global`」のフィールド名は無効です。

    これらの例外のいずれかが含まれるフィールド名に、HTTP クエリーを使用して XML でアクセスすることはできません。これらの構造は、46 ページの「完全修飾フィールド名の構文について」で記述されているとおり、レコード ID に予約されています。
- `-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。

クエリーコマンド解析

Web 公開エンジンは次の順番でクエリーコマンドを解析し、最初のエラーに遭遇した時点で XML クエリーの解析を終了します。エラーコードが返された場合、そのコードは特定した最初のエラーと一致します。

1. クエリーにコマンドがあり、そのコマンドが有効か?
   クエリーにコマンドがない場合、または不明なコマンドを使用した場合はエラーになります。

   例

   ```
   -database
   ```

2. クエリーに 2 つのコマンドがあるか?

   例

   ```
   -find&-edit
   ```

3. クエリーにコマンドまたは引数の無効な値があるか?

   例

   ```
   -lop=amd
   ```

4. クエリーに必須のデータベース名引数 (`-db` 引数) が欠けているか?
5. クエリーに必須のレイアウト名引数 (`-lay` 引数) が欠けているか?
6. クエリーに無効なソートがあるか?
7. クエリーに無効なフィールド引数があるか?

メモ クエリーに有効でも異質な情報が含まれている場合、クエリーはエラーなしで処理されます。たとえば、`-lop` 引数を `-delete` コマンドで指定した場合、クエリーが無効であるか、またはあいまいな場合は実行できないので、`-lop` 引数は無視されます。

返される特定のエラーコードについては、付録 A「カスタム Web 公開のエラーコード」を参照してください。

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).record-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。
- record-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、または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。record-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。

メモ 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。

ポータルフィールドでのクエリーコマンドの使用

次の各セクションでは、ポータルフィールドでのクエリーコマンドの操作方法について説明します。

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

例

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
```

メモ 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>` 要素内にある `<record>` 要素の record-id 属性から判断できます。30 ページの「fmresultset 文法の使用」を参照してください。

例

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
```

1 つのリクエストを使用して、親レコードから複数のポータルレコードを編集する方法:

```
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。

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

ポータルレコードの削除

ポータルレコードを削除するには、`-delete` コマンドではなく `-edit` コマンドで `-delete.related` 引数を使用します。

例

次の URL では、「employees」から「1001」レコードが削除されます。

```
http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family
&-recid=1001&-delete
```

ただし、次の URL では、「Dependents」という関連テーブルから親レコード「1001」とともに、レコード ID が「3」であるポータルレコードが削除されます。

```
http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees&-lay=family
&-recid=1001&-delete.related=Dependents.3&-edit
```

53 ページの「-delete.related (ポータルレコードを削除) クエリー引数」を参照してください。

ポータルレコードのクエリーを実行

関連レコードが多くあるソリューションでは、ポータルレコードのクエリーを実行してソートすると、時間がかかる可能性があります。関連セットで表示するレコードと行の数を制限するには、`-relatedsets.filter` 引数および `-relatedsets.max` 引数を使用してリクエストを検索します。59 ページの「-relatedsets.filter (ポータルレコードのフィルタ) クエリー引数」および 60 ページの「-relatedsets.max (ポータルレコードの制限) クエリー引数」を参照してください。

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、46 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。ダイレクト XML リクエストでは、リクエストの直後にグローバル値の期限が切れます。

グローバルフィールドの識別にクエリー文字列で `.global` 構文を使用しない場合、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。

例

```
http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees
&-lay=departments&Country.global=USA&-recid=1&-edit
```

クエリーコマンドリファレンス

このセクションでは、XML 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。

-dbnames (データベース名) クエリーコマンド

FileMaker Server で共有されていて、カスタム Web 公開 with XML 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` 引数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、フィールド名

メモ -edit コマンドは、ポータルのレコードの編集に使用できます。47 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
```

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には、-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には、-findany を使用します:

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

メモ

- 1 回のリクエストでのフィールド名の複数回指定はサポートされていません。FileMaker Server ではすべての値を解析しますが、解析された最後の値のみが使用されます。
- -findall コマンドを使用する場合、-max クエリー引数を使用して 1 ページにつき返すデフォルトの最大レコード数を指定することでコンピュータメモリの過負荷問題を回避します。

-findquery (複合検索) クエリーコマンド

複数の検索レコードおよびレコード除外リクエストを使用して、検索リクエストを送信します。

必須のクエリー引数: -db、-lay、-query

オプションのクエリー引数: -max、-skip、-sortorder、-sortfield、-script、-script.prefind、-script.presort

例

「Fluffy」という名前ではない猫または犬のレコードの検索:

```
http://host/fmi/xml/fmresultset.xml?-db=vetclinic&-lay=animals
&-query=(q1);(q2);!(q3)&-q1=typeofanimal&-q1.value=Cat&-q2=typeofanimal
&-q2.value=Dog&-q3=name&-q3.value=Fluffy&-findquery
```

複合検索での -findquery コマンドの使用

`-findquery` ステートメントは、次の順序での 4 つのパートからなります:

- `-query` 引数
- クエリー識別子宣言とリクエスト処理からなるクエリーリクエスト宣言
- 各識別子における検索フィールドと値の定義
  - クエリー識別子を定義します。クエリー識別子では、「q」という文字の後に数字が付きます。例: `-q1`
  - 引数でクエリー識別子値を定義します。例: `-q1.value=fieldvalue`
  - クエリー識別子の演算子を `fieldvalue` 式の一部として含めるように定義します。たとえば、アスタリスクを「begins with」演算子として使用します。例: `-q1.value=fieldvalue*`
- 構文全体の最後にある `-findquery` コマンド

`-query` 引数の使用方法については、57 ページの「-query (複合検索条件) クエリー引数」を参照してください。

-layoutnames (レイアウト名) クエリーコマンド

FileMaker Server で共有されていて、カスタム Web 公開 with XML 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`

メモ 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、46 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
```

-scriptnames (スクリプト名) クエリーコマンド

FileMaker Server で共有されていて、カスタム Web 公開 with XML 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
```

クエリー引数リファレンス

このセクションでは、XML 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。

-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)

メモ 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 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
```

-delete.related (ポータルレコードを削除) クエリー引数

ポータルフィールドからレコードを削除します。

オプション: -edit クエリーコマンド

必須: 関連テーブル名とレコード ID

例

次の例では、「jobtable」という関連テーブルから、親レコード「7」とともに、レコード ID が「20」であるポータルレコードが削除されます。

```
http://host/fmi/xml/fmresultset.xml?-db=career&-lay=applications&-recid=7
&-delete.related=jobtable.20&-edit
```

-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

27 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。

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 Pro 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。

フィールド名にピリオドやカッコを使用しないことをお勧めします。ピリオドを含むフィールド名が機能することもありますが、次の例外を含むフィールド名を使用することはできません:

- ピリオドは、数字の前に置くことはできません。たとえば、「myfield.9」のフィールド名は無効です。
- ピリオドは、文字列「op」 (2 文字の「op」) の前に置くことはできません。たとえば、「myfield.op」のフィールド名は無効です。
- ピリオドは、文字列「global」 (「global」という文字) の前に置くことはできません。たとえば、「myfield.global」のフィールド名は無効です。

これらの例外のいずれかが含まれるフィールド名に、HTTP クエリーを使用して XML でアクセスすることはできません。フィールド名でのピリオドの使用は、46 ページの「完全修飾フィールド名の構文について」の説明のとおり、レコード ID に予約されています。

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、およびタイムスタンプのフィールドの値を指定する場合、そのフィールドタイプに「fm」の書式を使用して、値を指定する必要があります。「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。

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
```

メモ 1 回のリクエストでのフィールド名の複数回指定はサポートされていません。FileMaker Server ではすべての値を解析しますが、解析された最後の値のみが使用されます。

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。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
```

メモ `bw` キーワードは、日付、時刻、タイムスタンプ文字列、および現在の日付 (//) 検索演算子では機能しません。

フィールド名.op 演算子キーワードを指定する代わりに、検索条件の一部として含めることによって FileMaker Pro の任意の検索演算子を使用できます。たとえば値の範囲を検索するには、範囲 (...) 検索演算子を使用して、演算子のキーワードは指定しません。代わりに、検索条件で範囲値の間に「...」文字列を使用します。

例

```
http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees
&-lay=departments&IDnum=915...925&-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。

-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`、および `-scriptnames` を除く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に指定します。

`-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。

XML リクエストに -lay.response 引数を使用できます。

値: レイアウトの名前

オプション: -dbnames、-layoutnames、および -scriptnames を除く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 -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

メモ -findquery クエリーコマンドではサポートされません。

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。-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、-findall、および -findquery クエリーコマンド

メモ -maxクエリー引数は、ポータルレコードで返された値に影響しません。ポータルレコードで返された列数を制限するには、60 ページの「-relatedsets.max (ポータルレコードの制限) クエリー引数」を参照してください。

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 Pro 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
```

-query (複合検索条件) クエリー引数

複合検索条件における、クエリー名と検索基準を指定します。50 ページの「-findquery (複合検索) クエリーコマンド」を参照してください。

値: クエリー式

必須: `-findquery` クエリーコマンド

複合検索条件用の構文は、次のようになります:

`-query=<リクエスト宣言><リクエスト定義>&-findquery`

各要素の意味は次のとおりです:

<リクエスト宣言> は、2 つ以上のリクエスト宣言です。

- 各リクエスト宣言は、コンマで区切られた 1 つまたは複数のクエリー識別子からなり、カッコで囲まれます。クエリー識別子では、「q」という文字の後に数字が付きます。例: `q1`
- カッコで囲まれ、複数のクエリーは、対象レコードを絞り込む論理式 AND での検索として動作します。たとえば、`(q1, q2)` で返されるレコードは、`q1` および `q2` と一致します。

  メモ 複数の `q` 変数を、「and」検索条件を用いた同じフィールドに使用することはお勧めしません。

- FileMaker Pro の場合のように、各リクエストは検索リクエストまたは除外リクエストのいずれかにできます。検索リクエストでは一致するレコードが対象レコードに追加されます。除外リクエストでは一致するレコードが対象レコードから除外されます。デフォルトは、検索リクエストです。除外リクエストの場合、リクエストの前に感嘆符 (!) を付けます。

  例

  ```
  (q1);!(q2)
  ```

  `q1` が検索リクエストで、感嘆符が前に付いている `q2` は除外リクエストです。

- リクエストは、セミコロンで区切られます。複数のリクエストは、対象レコードを拡大する論理式 OR での検索として動作します。たとえば、`(q1);(q2)` で返されるレコードは、`q1` or `q2` と一致します。除外リクエストは、検索条件からレコードを取り除くため論理式 OR 検索としては動作しません。
- リクエストは、指定された順序で実行されます。対象レコードには、複合検索条件全体の結果が含まれます。

<リクエスト定義> は、各リクエスト宣言におけるリクエスト定義です。各リクエスト定義は、検索フィールドと値の定義からなります。マイナス (-) 記号は、リクエスト定義の開始です。

構文:

`-<クエリー ID>=<フィールド名>&-<クエリー ID>.value=<値>`

例

```
-q1=typeofanimal&-q1.value=Cat
-q2=name&-q2.value=Fluffy
```

「Fluffy」という名前ではない「Gray」の猫の検索:

```
http://host/fmi/xml/fmresultset.xml?-db=petclinic&-lay=Patients
&-query=(q1, q2);!(q3)&-q1=typeofanimal&-q1.value=Cat&-q2=color
&-q2.value=Gray&-q3=name&-q3.value=Fluffy&-findquery
```

-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。`view` コマンドによって使用され、関連する値一覧データを FMPXMLLAYOUT 文法で取得します。

値: レコード ID は、FileMaker Pro 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` クエリーおよび `-view` コマンド

例

```
http://192.168.123.101/fmi/xml/fmresultset.xml?-db=employees
&-lay=departments&-recid=22&-delete
http://localhost/fmi/xml/FMPXMLLAYOUT.xml?-db=test&-lay=empty&-view
&-recid=9
```

-relatedsets.filter (ポータルレコードのフィルタ) クエリー引数

このクエリーの結果で返すポータルレコードを制限するかどうかを指定します。

値: `layout` または `none`

- `-relatedsets.filter` が `layout` に設定されている場合、FileMaker Pro の [ポータル設定] ダイアログボックスで指定された [**最初の行**] の設定が優先されます。
  - [ポータル設定] ダイアログボックスの [**垂直スクロールを許可**] 設定が有効になっている場合、`-relatedsets.max` オプションを使用して返されるレコードの最大数を指定します。「-relatedsets.max (ポータルレコードの制限) クエリー引数」を参照してください。
  - [**垂直スクロールを許可**] 設定が無効な場合、または `-relatedsets.max` オプションを使用しない場合、返されるポータルレコードの数は [ポータル設定] ダイアログボックスの [**行数**] 設定に基づいて決定されます。
- この引数が指定されていない場合、デフォルトの値は「`none`」です。`-relatedsets.filter` が「`none`」に設定されている場合、Web 公開エンジンによってポータル内のすべてのレコードが返されます。[ポータル設定] ダイアログボックスで指定されている [**最初の行**] および [**行数**] の値は無視されます。

メモ

- `-relatedsets.filter` 引数は、ポータルレコードが XML クエリーに保存される方法には影響しません。`-relatedsets.filter` 引数の値が「`layout`」または「`none`」であるかを問わず、FileMaker Pro で指定されているソートが尊重されます。
- [ポータル設定] ダイアログボックスの [**ポータルレコードのフィルタ**] 設定は、XML クエリーでサポートされていません。[**ポータルレコードのフィルタ**] 設定に指定されている計算は無視されます。

オプション: `-find`、`-edit`、`-new`、`-dup`、および `-findquery`

例

```
http://localhost/fmi/xml/fmresultset.xml?-db=FMPHP_Sample&-lay=English
&-relatedsets.filter=none&-findany
http://localhost/fmi/xml/fmresultset.xml?-db=FMPHP_Sample
&-lay=English&relatedsets.filter=layout&-relatedsets.max=all&-findany
http://localhost/fmi/xml/fmresultset.xml?-db=FMPHP_Sample&-lay=English
&-relatedsets.filter=layout&-relatedsets.max=10&-findany
```

-relatedsets.max (ポータルレコードの制限) クエリー引数

このクエリーの結果で返すポータルレコードの最大数を指定します。

値: 整数または `all`

- `-relatedsets.max` 引数は、[**垂直スクロールを許可**] 設定が FileMaker Pro の [ポータル設定] ダイアログボックスで有効な場合、および `-relatedsets.filter` 引数が「`layout`」である場合にのみ尊重されます。
  - `-relatedsets.max` 引数で整数が指定される場合、Web 公開エンジンによって、最初の行からその数のポータルレコードが返されます。
  - `-relatedsets.max` 引数で「`all`」を指定する場合、Web 公開エンジンによって、すべてのポータルレコードが返されます。

メモ ポータルレコードのフィルタの詳細については、上記の「-relatedsets.filter (ポータルレコードのフィルタ) クエリー引数」を参照してください。

オプション: `-find`、`-edit`、`-new`、`-dup`、および `-findquery`

例

```
http://localhost/fmi/xml/fmresultset.xml?-db=FMPHP_Sample
&-lay=English&relatedsets.filter=layout&-relatedsets.max=all&-findany
http://localhost/fmi/xml/fmresultset.xml?-db=FMPHP_Sample&-lay=English
&-relatedsets.filter=layout&-relatedsets.max=10&-findany
```

-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。42 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`、および `-scriptnames` を除く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

`-script` によって指定された FileMaker スクリプトに引数を渡します。

値: 1 つのテキスト引数

- 複数の引数を渡すには、それらの引数を区切る文字列を作成し、スクリプトが個々の引数を解析するようにします。たとえば、「`param1|param2|param3`」は「`|`」文字を「`param1%7Cparam2%7Cparam3`」のように URL エンコードした一覧として渡します。
- テキスト引数をテキストではない値として処理するには、スクリプトでテキスト値を変換できます。たとえば、テキスト値を数字に変換する場合には、スクリプトに次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 がなく -script.param が含まれている場合、-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、Web 公開エンジンによって、解析される最後の値が使用されます。

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、および -scriptnames を除く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 (検索前にスクリプトに引数を渡す) クエリー引数

-script.prefind によって指定された FileMaker スクリプトに引数を渡します。

値: 1 つのテキスト引数

- 複数の引数を渡すには、それらの引数を区切る文字列を作成し、スクリプトが個々の引数を解析するようにします。たとえば、「param1|param2|param3」は「|」文字を「param1%7Cparam2%7Cparam3」のように URL エンコードした一覧として渡します。
- テキスト引数をテキストではない値として処理するには、スクリプトでテキスト値を変換できます。たとえば、テキスト値を数字に変換する場合には、スクリプトに次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 がなく -script.prefind.param が含まれている場合、-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、Web 公開エンジンによって、解析される最後の値が使用されます。

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) の後、レコードの検索の前に実行する FileMaker スクリプトを指定します。

オプション: -dbnames、-layoutnames、および -scriptnames を除く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 (ソート前にスクリプトに引数を渡す) クエリー引数

-script.presort によって指定された FileMaker スクリプトに引数を渡します。

値: 1 つのテキスト引数

- 複数の引数を渡すには、それらの引数を区切る文字列を作成し、スクリプトが個々の引数を解析するようにします。たとえば、「param1|param2|param3」は「|」文字を「param1%7Cparam2%7Cparam3」のように URL エンコードした一覧として渡します。
- テキスト引数をテキストではない値として処理するには、スクリプトでテキスト値を変換できます。たとえば、テキスト値を数字に変換する場合には、スクリプトに次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 がなく -script.presort.param が含まれている場合、-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、Web 公開エンジンによって、解析される最後の値が使用されます。

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

例

結果の最初の 10 レコードをスキップして、11 番目から 15 番目のレコードを返します。

```
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、複数のフィールドの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数字を指定します。precedence-number の値は:

- 1 から開始する必要があります。
- 連続してインクリメントする必要があります。
- 9 より大きい数を指定することはできません。

例

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。

```
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。

例

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。

```
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
```

メモ 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。

第 6 章
カスタム Web 公開 with PHP について

カスタム Web 公開 with PHP では、PHP スクリプト言語を使用して FileMaker Pro データベースからのデータをカスタマイズした Web ページレイアウトと統合できます。カスタム Web 公開 with PHP は、FileMaker API for PHP を提供します。これは PHP クラスで、FileMaker Server が共有するデータベースにアクセスします。この PHP クラスは、FileMaker Server の Web 公開エンジンに接続し、ご使用の Web サーバーの PHP エンジンに対してデータを利用可能にします。

カスタム Web 公開 with PHP の主な機能

- オープンソース PHP スクリプト言語を使用する Web アプリケーションを作成します。FileMaker Server でサポートされているバージョンの PHP を使用するか、独自のバージョンの PHP を使用します。(独自の PHP バージョンの使用を選択した場合は、67 ページの「FileMaker API for PHP の手動によるインストール」を参照してください。)
- FileMaker Server 上でデータベースを共有します。FileMaker Server がデータベースを共有するので、FileMaker Pro はカスタム Web 公開には必要ありません。
- 共有されている FileMaker Pro データベース内のレコードを作成、削除、編集、または複製できる PHP コードを記述します。記述したコードは、共有データベースに変更を確定する前に、フィールドおよびレコードの入力値の制限のチェックを実行できます。
- レイアウト、ポータル、値一覧、および関連フィールドにアクセスする PHP コードを記述します。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。14 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。
- 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 PHP コードを記述します。FileMaker プラットフォームは多くのスクリプトステップをカスタム Web 公開でサポートしています。18 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。
- 複雑な検索条件を実行する PHP コードを記述します。

カスタム Web 公開の必要条件

このセクションでは、PHP 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に必要な事項、カスタム Web 公開ソリューションにアクセスするために Web ユーザに必要なこと、および Web 公開ソリューションを共有することによるサーバーに与える影響について説明します。

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

カスタム Web 公開 with PHP 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:

- 3 つのコンポーネントを含む FileMaker Server の展開:
  - Microsoft IIS (Windows) または Apache (macOS) のいずれかの Web サーバー。FileMaker Web サーバーモジュールは Web サーバー上にインストールされます。
  - FileMaker Web 公開エンジン
  - FileMaker データベースサーバー
- FileMaker Server で共有されている 1 つ以上の FileMaker Pro データベース
- 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
- 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 サーバー上にインストールされた PHP。FileMaker Server では、サポートされているバージョンの PHP をインストールするか、ユーザ独自のバージョンを使用できます。
  - 最低限必要な PHP のバージョンについては、FileMaker Server の技術仕様を参照してください。
  - PHP の詳細については、php.net を参照してください。
  - Web サーバーにインストールする PHP のバージョンは、cURL (クライアント URL ライブラリ) 関数をサポートする必要があります。cURL については、php.net/curl を参照してください。

  重要 FileMaker Server でサポートされているバージョンの PHP をインストールする場合、macOS Server Admin ツールには表示されません (一覧されないようになっています)。macOS Server Admin ツールを使用して PHP をオンにする場合は、FileMaker Server でサポートされているバージョンの PHP を無効にして独自の PHP バージョンを有効にします。

FileMaker Server インストールおよび構成ガイドを参照してください。

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 ユーザがカスタム Web 公開 with PHP 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
- 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が必要です。

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、データベースを共有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してください。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。
- FileMaker Server 展開の一部である Web サーバー用のホストコンピュータには、固有の静的 (不変) 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 アドレスでは、データベースの検索が困難になります。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確認してください。

FileMaker API for PHP の手動によるインストール

FileMaker Server をインストールする場合、FileMaker でサポートされているバージョンの PHP がインストールされます。別の PHP エンジンのインストールおよび構成が終了していて FileMaker API for PHP のみを追加する場合は、FileMaker API for PHP クラスを手動でインストールして PHP スクリプトで利用できるようにします。CLI を使用して FileMaker でサポートされているバージョンの PHP を無効にしてください。CLI ヘルプを参照してください。

独自の PHP エンジンを使用している場合は、ご使用のバージョンの PHP エンジン上で次の構成タスクを実行してください:

- php.ini 内の cURL モジュールを有効にする。
- php.ini 内の include_path 変数にある FileMaker API for PHP の場所を指定する。
- 日付と時刻を含むデータベースにアクセスしている場合、PEAR Date パッケージをインストールします。

メモ 最低限必要な PHP のバージョンについては、FileMaker Server の技術仕様を参照してください。PHP の機能を最大限に活用するためには適切なバージョンの使用をお勧めします。

FileMaker API for PHP を PHP スクリプトからアクセスできるようにする方法

FileMaker Server をインストールすると、FileMaker API for PHP パッケージが .zip ファイルとして次の場所に含まれます:

- IIS (Windows):
  `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥Web Publishing¥FM_API_for_PHP_Standalone.zip`
  [ドライブ] は展開した FileMaker Server の Web サーバーコンポーネントが格納されているドライブです。
- Apache (macOS):
  `/ライブラリ/FileMaker Server/Web Publishing/FM_API_for_PHP_Standalone.zip`

「FM_API_for_PHP_Standalone.zip」ファイルには「FileMaker.php」という名前のファイルおよび「FileMaker」という名前のフォルダが含まれます。ファイルを解凍して「FileMaker.php」ファイルおよび「FileMaker」フォルダを次の場所のいずれかにコピーします:

- PHP スクリプトが存在するフォルダ。
  - HTTP または HTTPS での IIS (Windows):
    `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥HTTPServer¥Conf`
    [ドライブ] は展開した FileMaker Server の Web 公開エンジンコンポーネントが格納されているドライブです。
  - HTTP での Apache (macOS):
    `/ライブラリ/FileMaker Server/HTTPServer/htdocs`
- HTTPS での Apache (macOS):
  `/ライブラリ/FileMaker Server/HTTPServer/htdocs/httpsRoot`

  メモ カスタム SSL 証明書をインポートすると、データベースサーバークライアント接続で SSL が使用され、HTTP 接続が HTTPS にルーティングされます。ご使用のサーバー上のカスタム SSL 証明書とともに、サイトの PHP ファイルを共有するための HTTPS ディレクトリを使用してください。

- PHP インストール内の include_path ディレクトリの 1 つ。macOS のデフォルトの場所は、`/usr/lib/php` です。

この後の作業を開始するにあたって

- CLI を使用してカスタム Web 公開を有効にします。FileMaker Server ヘルプを参照してください。
- 公開する各 FileMaker Pro 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。13 ページの「データベースのカスタム Web 公開の有効化」を参照してください。
- FileMaker API for PHP を使用して FileMaker Pro データベースのデータにアクセスする方法は、第 8 章「FileMaker API for PHP の使用」を参照してください。

第 7 章
カスタム Web 公開 with PHP の概要

FileMaker API for PHP は、FileMaker Pro データベースから PHP ソリューションへデータを統合するのに役立ちます。この章では、PHP が FileMaker Server のカスタム Web 公開エンジンと連携する方法について説明します。FileMaker API for PHP の詳細については、第 8 章「FileMaker API for PHP の使用」を参照してください。

Web 公開エンジンと PHP ソリューションの連携方法

FileMaker Server は、Web サーバー、Web 公開エンジン、およびデータベースサーバーという 3 つのコンポーネントから構成されます。FileMaker Server インストールおよび構成ガイドを参照してください。PHP ソリューションをサポートするために、Web サーバーとともに PHP エンジンがプライマリマシンにインストールされます。PHP ファイルをプライマリマシンの Web サーバー上に配置すると、FileMaker Server で PHP ソリューションが共有されます。

- Web ユーザが PHP ソリューションを開くと、Web サーバーはリクエストを PHP コードが処理される PHP エンジンにルーティングします。
- PHP コードが FileMaker API for PHP への呼び出しを含む場合は、それらの呼び出しは解釈され、Web 公開エンジンへのリクエストとして送信されます。
- Web 公開エンジンが、データベースサーバーで共有されているデータベースにデータをリクエストします。
- データベースサーバーが、リクエストされたデータを Web 公開エンジンに送信します。
- Web 公開エンジンは、API の呼び出しに応えて、Web サーバー上の PHP エンジンへデータを送信します。
- PHP ソリューションはデータを処理して Web ユーザに表示します。

カスタム Web 公開 with PHP の一般手順

1. FileMaker Pro を使用してデータベースのカスタム Web 公開を有効にします。第 2 章「データベースのカスタム Web 公開の準備」を参照してください。

   メモ　PHP ソリューションを開発する際は、エンドユーザに提供するアクセス権セットと同等の FileMaker Pro 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 Pro 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。

2. FileMaker Server CLI を使用してカスタム Web 公開 with PHP を有効にします。FileMaker Server ヘルプを参照してください。

3. Admin Console で、公開する各 FileMaker Pro データベースの fmphp 拡張アクセス権が有効になっていることを確認します。
   - Admin Console で、[**データベース**] ページをクリックします。
   - [**すべてのデータベース**] の横のメニューセレクタから [**拡張アクセス権を表示**] を選択します。fmphp 拡張アクセス権が有効になっているデータベースには **FMPHP** という文字が表示されます。
4. PHP オーサリングツールを使用して FileMaker API 関数を PHP コードに統合して FileMaker データにアクセスし、PHP ソリューションを作成します。第 8 章「FileMaker API for PHP の使用」を参照してください。
5. プライマリマシンの Web サーバーの次のフォルダにご使用のサイトのディレクトリ構造およびファイルをコピーまたは移動します。
   - HTTP または HTTPS での IIS (Windows):
     `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥HTTPServer¥Conf`
     [ドライブ] は FileMaker Server を展開したプライマリマシンのドライブです。
   - HTTP での Apache (macOS):
     `/ライブラリ/FileMaker Server/HTTPServer/htdocs`
   - HTTPS での Apache (macOS):
     `/ライブラリ/FileMaker Server/HTTPServer/htdocs/httpsRoot`

   メモ カスタム SSL 証明書をインポートすると、データベースサーバークライアント接続で SSL が使用され、HTTP 接続が HTTPS にルーティングされます。ご使用のカスタム SSL 証明書とともに、サイトの PHP ファイルを共有するための HTTPS ディレクトリを使用してください。
6. 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、レコードを作成または編集するときに、その参照されているオブジェクトが FileMaker Pro の「Web」フォルダに保存されている必要があります。オブジェクトを、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。

   16 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。
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。
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、サイトをテストします。
9. サイトを使用可能にしてユーザに通知します。Web ユーザが入力する URL には次の形式が使用されます:

   `http://<サーバー>/<サイトパス>`
   - <サーバー> は FileMaker Server が存在しているコンピュータです。
   - <サイトパス> は上記手順 5 で使用したディレクトリ構造によって決定される、サイトのホームページの相対パスです。

例

ご使用の Web サーバーのアドレスが 192.168.123.101 で、サイトのホームページが c:¥Inetpub¥wwwroot¥customers¥index.php の Web サーバー上にある場合、Web ユーザは次のように URL を入力します:

```
http://192.168.123.101/customers/index.php
```

メモ PHP では Latin-1 (ISO-8859-1) エンコードを使用します。FileMaker Server は Unicode (UTF-8) データを返します。CLI を使用してサイト用にデフォルトの文字エンコードを指定します。CLI ヘルプを参照してください。PHP サイトには、UTF-8 または ISO-8859-1 のいずれかを指定できますが、UTF-8 が推奨されます。サイトの PHP ファイルの `<HEAD>` セクションにある `charset` 属性に同じ設定を指定します。

PHP ソリューションの展開と使用の詳細については、第 9 章「サイトのステージング、テスト、および監視」を参照してください。

第 8 章
FileMaker API for PHP の使用

FileMaker API for PHP には、FileMaker Pro データベースに対するオブジェクト指向インターフェースを提供する PHP クラス (FileMaker クラス) が実装されています。FileMaker API for PHP を使用すると、FileMaker Pro データベースに保存されているロジックおよびデータの両方に対し、Web 上にアクセスして公開、または他のアプリケーションにエクスポートすることができます。

FileMaker API for PHP は、FileMaker Pro データベース内ですでに使用可能な次の機能を PHP コードで実行できるようにします:

- レコードの作成、削除、編集、または複製
- 検索条件の実行
- フィールドおよびレコードの入力値の制限のチェックの実行
- レイアウトの使用
- FileMaker スクリプトの実行
- ポータルおよび関連レコードの表示
- 値一覧の使用

この章では、FileMaker クラスオブジェクトの使用方法、およびこれらの一般的な機能を PHP ソリューションに追加するメソッドを説明します。この章は、FileMaker API for PHP 全体をカバーするものではありませんが、主要なオブジェクトおよびメソッドを紹介します。

追加情報の入手場所

FileMaker API for PHP の詳細については、次のリソースを参照してください。

すでに PHP エンジンのインストールおよび構成が終了し、FileMaker API for PHP を追加するだけの場合は、67 ページの「FileMaker API for PHP の手動によるインストール」を参照してください。

FileMaker API for PHP リファレンス

FileMaker API for PHP をインストールしている場合は、展開した FileMaker Server の Web サーバーコンポーネントでリファレンス情報を参照できます。

- IIS (Windows):
  `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥Documentation¥PHP API Documentation¥index.html`
  [ドライブ] は展開した FileMaker Server の Web サーバーコンポーネントが格納されているドライブです。
- Apache (macOS): /ライブラリ/`FileMaker Server/Documentation/PHP API Documentation/index.html`

FileMaker API for PHP に関するサポート

FileMaker API for PHP の追加情報については、FileMaker のサポートページを参照してください。

FileMaker クラスの使い方

PHP ソリューションで FileMaker クラスを使用するには、PHP コードに次の文を追加します:

```
require_once ('FileMaker.php');
```

FileMaker クラスオブジェクト

FileMaker クラスは FileMaker Pro のデータベースからデータを取得するのに使用できるクラスオブジェクトを定義します。

クラスオブジェクト	オブジェクトを使用して行う処理
FileMaker Pro データベース	データベースのプロパティの定義 FileMaker Pro データベースファイルへの接続 FileMaker API for PHP の情報の取得
コマンド	レコード追加、レコード削除、レコード複製、レコード編集、検索条件実行、およびスクリプト実行コマンドの作成
レイアウト	データベースレイアウトの使用
レコード	レコードデータの使用
フィールド	フィールドデータの使用
関連セット	ポータルレコードの使用
結果	検索条件から返されたレコードの処理
エラー	エラーが発生したかどうかの確認 エラーの処理

FileMaker のコマンドオブジェクト

FileMaker クラスは特定のコマンドのインスタンスを作成し、コマンドの引数を指定するのに使用する基本コマンドオブジェクトを定義します。コマンドを実行するには、`execute()` メソッドを呼び出す必要があります。

FileMaker クラスは次の特定のコマンドを定義します:

- Add コマンド
- Compound Find コマンド
- Delete コマンド
- Duplicate コマンド
- Edit コマンド
- Find コマンド、Find All コマンド、Find Any コマンド
- Find Request コマンド (Compound Find コマンドに追加される)
- Perform Script コマンド

重要 コマンドには `FileMaker.php` クラスに定義されている戻り値があります。たとえば、一部のコマンドは TRUE などの論理値または FileMaker_Error オブジェクトを返します。その他のコマンドはレイアウトの「対象レコード」全体を含む場合がある FileMaker_Result オブジェクトを返します。コンピュータのメモリの過負荷の問題を回避するには、使用するコマンドの要求する戻り値に注意してください。各コマンドの戻り値の詳細については、72 ページの「FileMaker API for PHP リファレンス」を参照してください。

ほとんどの PHP アプリケーションが実行する必要がある基本的なタスクについては、次を参照してください:

- 75 ページの「レコードの使用」
- 77 ページの「FileMaker スクリプトの実行」
- 84 ページの「検索条件の実行」

FileMaker API で使用するデータのデコード

PHP アプリケーションが Web サイトからデータを取得している場合、そのデータは URL エンコードされている可能性があります。FileMaker API for PHP では、データが URL エンコードされた文字列ではなく、デコードされた文字列であるものとして処理します。通常は、PHP アプリケーションでデータを取得する場合は urldecode() 関数を呼び出すことをお勧めします。

例

```
$user = urldecode($_GET['user']);
$event = urldecode($_GET['event']);
```

メモ FileMaker API for PHP では、アンパサンド (&) 文字を含む文字列を使用することはできません。FileMaker API for PHP に渡す文字列に含まれる特殊文字の前にはエスケープ文字としてバックスラッシュを使用します。

FileMaker Pro データベースへの接続

FileMaker クラスはサーバーまたはデータベースに接続するためにインスタンスを作成するデータベースオブジェクトを定義します。クラスコンストラクタを使用するか、`setProperty()` メソッドを呼び出してオブジェクトのプロパティを定義します。

例

サーバーに接続し、データベースの一覧を表示:

```
$fm = new FileMaker();
$databases = $fm->listDatabases();
```

サーバー上の特定のデータベースへ接続:

```
$fm = new FileMaker();
$fm->setProperty('database', 'questionnaire');
$fm->setProperty('hostspec', 'http://192.168.100.110');
$fm->setProperty('username', 'web');
$fm->setProperty('password', 'web');
```

ユーザ名とパスワードのプロパティによって、この接続用のアクセス権セットが決まります。

メモ `hostspec` プロパティは、デフォルトで `http://localhost` という値になります。PHP エンジンはプライマリマシンの Web サーバーコンポーネントとともにインストールされるため、hostspec プロパティを指定する必要はありません。

レコードの使用

FileMaker クラスはレコードを使用するためにインスタンスを作成するレコードオブジェクトを定義します。レコードオブジェクトのインスタンスは、FileMaker Pro データベースの 1 つのレコードを表します。レコードオブジェクトを、Add、Delete、Duplicate、および Edit コマンドと使用して、レコード内のデータを変更します。検索コマンド (Find、Find All、Find Any、および Compound Find) は、レコードオブジェクトの配列を返します。

レコードの作成

レコードを作成するには、次の 2 つの方法があります:

- `createRecord()` メソッドを使用します (レイアウト名を指定、およびフィールド値の配列をオプションで指定)。新規レコードオブジェクトでは個別に値を設定することもできます。

  `createRecord()` メソッドは、新規レコードをデータベースに保存しません。レコードをデータベースに保存するには、`commit()` メソッドを呼び出します。

  例

  ```
  $rec = $fm->createRecord('Form View', $values);
  $result = $rec->commit();
  ```

  FileMaker_Record の `commit()` メソッドを使用すると、エラーがない場合は `$result` 変数に論理値 TRUE が割り当てられ、FileMaker Pro データベースに新しいレコードが作成されます。

  エラーが発生した場合は、変数 `$result` に FileMaker_Error オブジェクトが含まれます。`commit()` メソッドの実行後にエラーチェックを行ってください。

- Add コマンドを使用します。`newAddCommand()` メソッドを使用し、レイアウト名およびレコードデータを持つ配列を指定して FileMaker_Command_Add オブジェクトを作成します。レコードをデータベースに保存するには、`execute()` メソッドを呼び出します。

  例

  ```
  $newAdd = $fm->newAddCommand('Respondent', $respondent_data);
  $result = $newAdd->execute();
  ```

  FileMaker_Command の `execute()` メソッドを使用すると、エラーがない場合は `$result` 変数に FileMaker_Result オブジェクトが含まれます。このオブジェクトには作成したレコードに関するすべての情報が含まれます。

  エラーが発生した場合は、変数 `$result` に FileMaker_Error オブジェクトが含まれます。`execute()` メソッドの実行後にエラーチェックを行ってください。

レコードの複製

Duplicate コマンドを使用して既存のレコードを複製します。`newDuplicateCommand()` メソッドを使用し、レイアウト名および複製するレコードのレコード ID を指定して、FileMaker_Command_Duplicate オブジェクトを作成します。その後、`execute()` メソッドを呼び出してレコードを複製します。

例

```
$newDuplicate = $fm->newDuplicateCommand('Respondent', $rec_ID);
$result = $newDuplicate->execute();
```

レコードの編集

レコードを編集するには、次の 2 つの方法があります:

- Edit コマンドを使用します。`newEditCommand()` メソッドを使用し、レイアウト名、編集するレコードのレコード ID、および更新する値の配列を指定して、FileMaker_Command_Edit オブジェクトを作成します。その後、`execute()` メソッドを呼び出してレコードを編集します。

  例

  ```
  $newEdit = $fm->newEditCommand('Respondent', $rec_ID, $respondent_data);
  $result = $newEdit->execute();
  ```

- レコードオブジェクトを使用します。データベースからレコードを取得し、フィールド値を変更し、`commit()` メソッドを呼び出してレコードを編集します。

例

```
$rec = $fm->getRecordById('Form View', $rec_ID);
$rec->setField('Name', $nameEntered);
$result = $rec->commit();
```

レコードの削除

レコードを削除するには、次の 2 つの方法があります:

- データベースからレコードを取得して `delete()` メソッドを呼び出します。

例

```
$rec = $fm->getRecordById('Form View', $rec_ID);
$rec->delete();
```

- Delete コマンドを使用して既存のレコードを削除します。`newDeleteCommand()` メソッドを使用し、レイアウト名および削除するレコードのレコード ID を指定して、FileMaker_Command_Delete オブジェクトを作成します。その後、`execute()` メソッドを呼び出してレコードを削除します。

例

```
$newDelete = $fm->newDeleteCommand('Respondent', $rec_ID);
$result = $newDelete->execute();
```

FileMaker スクリプトの実行

FileMaker のスクリプトは、スクリプトステップの名前付きのセットです。FileMaker クラスは FileMaker Pro のデータベースで定義された FileMaker スクリプトを使用可能にするためのいくつかのメソッドを定義します。Web 互換のスクリプトステップ (Web ソリューションの中で実行できるスクリプトステップ) については、18 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。

利用可能なスクリプト一覧の取得

`listScripts()` メソッドを使用して、現在接続されているデータベースから利用可能なスクリプトの一覧を取得します。`listScripts()` メソッドは、データベース接続が定義された際に指定されたユーザ名およびパスワードで実行できるスクリプトの配列を返します。(74 ページの「FileMaker Pro データベースへの接続」を参照してください。)

例

```
$scripts = $fm->listScripts();
```

FileMaker スクリプトの実行

`newPerformScriptCommand()` メソッドを使用し、レイアウト、スクリプト名、および必要なスクリプト引数を指定して、FileMaker_Command_PerformScript オブジェクトを作成します。その後、`execute()` メソッドを呼び出してスクリプトを実行します。

重要 FileMaker スクリプトを実行する場合、返される FileMaker_Result オブジェクトのサイズは FileMaker スクリプトの動作によって異なります。たとえば FileMaker スクリプトによって特定のレイアウトに切り替える場合、そのレイアウトのテーブルのすべてのレコードが対象レコードに含まれ、その対象レコード内のすべてのレコードが FileMaker_Result オブジェクトに返される場合があります。コンピュータのメモリの過負荷の問題を回避するには、PHP アプリケーションで FileMaker スクリプトを実行する前にその FileMaker スクリプトによって返されるデータに注意してください。

例

```
$newPerformScript = $fm->newPerformScriptCommand('Order Summary',
'ComputeTotal');
$result = $newPerformScript->execute();
```

コマンド実行前のスクリプトの実行

`setPreCommandScript()` メソッドを使用して、コマンドが実行される前に実行するスクリプトを指定します。次の例では、検索コマンドを使用していますが、任意のコマンドとともに `setPreCommandScript()` メソッドを使用できます。

例

```
$findCommand = $fm->newFindCommand('Students');
$findCommand->addFindCriterion('GPA', $searchValue);
$findCommand->setPreCommandScript('UpdateGPA');
$result = $findCommand->execute();
```

結果セットをソートする前のスクリプトの実行

`setPreSortScript()` メソッドを使用して、検索の結果セットが生成された後、結果セットをソートする前に実行するスクリプトを指定します。85 ページの「Find コマンドの使用」を参照してください。

例

```
$findCommand = $fm->newFindCommand('Students');
$findCommand->setPreSortScript('RemoveExpelled');
```

結果セットが生成された後のスクリプトの実行

`setScript()` メソッドを使用して、検索の結果セットが生成された後に実行するスクリプトを指定します。85 ページの「Find コマンドの使用」を参照してください。

例

```
$findCommand = $fm->newFindCommand('Students');
$findCommand->setScript('myScript','param1|param2|param3');
```

スクリプトの実行順序

`setPreCommandScript()`、`setPreSortScript()`、および `setScript()` メソッドを `setResultLayout()` および `addSortRule()` メソッドとともに単一のコマンドに指定できます。

次に、FileMaker Server と Web 公開エンジンがこれらのメソッドを処理する順序を示します:

1. `setPreCommandScript()` メソッドで指定されているスクリプトを実行します (指定されている場合)。
2. 検索またはレコードの削除コマンドのような、コマンド自体を処理します。
3. `setPreSortScript()` メソッドで指定されているスクリプトを実行します (指定されている場合)。
4. `addSortRule()` メソッドが指定されている場合は、検索の結果セットをソートします。
5. `setResultLayout()` メソッド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)。
6. `setScript()` メソッドで指定されているスクリプトを実行します (指定されている場合)。
7. 最終的な検索の結果セットが返されます。

上のいずれかの手順でエラーコードが生成された場合、コマンドの実行は停止し、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コマンドがあるとします。`addSortRule()` メソッド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。

`newFindCommand()` メソッドに指定されたレイアウトは、検索条件を処理するときに使用されます。`setResultLayout()` メソッドで別のレイアウトに切り替えると、元のレイアウトに基づく検索条件のエラーオブジェクトは使用できなくなります。元のレイアウトに基づく検索条件のエラーオブジェクトをテストする場合は、レイアウトを変更する前にエラーオブジェクトをチェックします。

例

```
request = $fm->newFindCommand('Students');
$request->addFindCriterion('Day', 'Wednesday');

// 検索実行
$result = $request->execute();

if (FileMaker::isError($result)) {
  if ($result->code = 401) {
    $findError = 'There are no Records that match that request: '. ' (' .
     $result->code . ')';
  } else {
    $findError = 'Find Error: '. $result->getMessage(). ' (' . $result->code
     . ')';
  }
}
$request->setResultLayout('Teachers');
// 結果レイアウトに切り替え
$result = $request->execute();
```

FileMaker Pro レイアウトの使用

レイアウトとは、ユーザがレコードをブラウズ、プレビュー、または印刷する時に、フィールド、オブジェクト、グラフィック、レイアウトパートなどがどのように配置されるかを決める情報です。FileMaker クラスは FileMaker Pro データベースで定義されたレイアウトを使用可能にするためのいくつかのメソッドを定義します。レイアウトについての情報は複数の FileMaker クラスオブジェクトから取得できます。

クラスオブジェクト	次のメソッドを使用
データベース	▪ `listLayouts()` は、利用可能なレイアウトの名前の一覧を取得します。 ▪ `getLayout()` は、レイアウト名を指定してレイアウトオブジェクトを取得します。
レイアウト	▪ `getName()` は、特定のレイアウトオブジェクトのレイアウト名を取得します。 ▪ `listFields()` は、レイアウト内で使用されるすべてのフィールド名の配列を取得します。 ▪ `getFields()` は、すべてのフィールドを配列のキーとして持ち、関連する FileMaker_Field オブジェクトを配列の値として持つ関連配列を取得します。 ▪ `listValueLists()` は、値一覧名の配列を取得します。 ▪ `listRelatedSets()` は、関連セットの名前の配列を取得します。 ▪ `getDatabase()` は、データベース名を返します。
レコード	▪ `getLayout()` は、特定のレコードに関連するレイアウトオブジェクトを返します。
フィールド	▪ `getLayout()` は、特定のフィールドを含むレイアウトオブジェクトを返します。
コマンド	▪ `setResultLayout()` は、現在のレイアウトとは異なるレイアウトでコマンドの結果を返します。

ポータルの使用

ポータルとは、1 つ以上の関連レコードのデータ行を表示するテーブルです。FileMaker クラスは FileMaker Pro データベースで定義されたポータルを使用可能にするための関連セットオブジェクト、およびいくつかのメソッドを定義します。

関連セットオブジェクトとは、関連ポータルのレコードオブジェクトの配列で、各レコードオブジェクトがポータル内の 1 データ行を表します。

特定のレイアウト上に定義されたポータルの一覧

特定のレイアウトオブジェクトには、`listRelatedSets()` メソッドを使用して、このレイアウト内で定義されたすべてのポータルのテーブル名の一覧を取得します。

例

```
$tableNames = $currentLayout->listRelatedSets();
```

特定の結果オブジェクト用のポータル名の取得

特定の FileMaker_Result オブジェクトには、`getRelatedSets()` メソッドを使用して、このレコード内のすべてのポータルの名前を取得します。

例

```
$relatedSetsNames = $result->getRelatedSets();
```

特定レイアウト用のポータルの情報の取得

特定のレイアウトオブジェクトには、`getRelatedSets()` メソッドを使用して、レイアウト内のポータルについて記述する FileMaker_RelatedSet オブジェクトの配列を取得します。返された配列は、テーブル名を配列のキーとして持ち、関連する FileMaker_RelatedSet オブジェクトを配列の値として持つ関連配列です。

例

```
$relatedSetsArray = $currentLayout->getRelatedSets();
```

特定ポータルの情報の取得

特定のレイアウトオブジェクトには、`getRelatedSet()` メソッドを使用して、特定のポータルについて記述する FileMaker_RelatedSet オブジェクトを取得します。

例

```
$relatedSet = $currentLayout->getRelatedSet('customers');
```

ポータルのテーブル名の取得

関連セットオブジェクトには、`getName()` メソッドを使用してポータル用のテーブル名を取得します。

例

```
$tableName = $relatedSet->getName();
```

特定レコード用のポータルレコードの取得

特定のレコードオブジェクトには、`getRelatedSet()` メソッドを使用して、そのレコードに関する特定のポータル用の関連レコードの配列を取得します。

例

```
$relatedRecordsArray = $currentRecord->getRelatedSet('customers');
```

ポータル内で新規レコードを作成

`newRelatedRecord()` メソッドを使用して、特定の関連セット内で新規レコードを作成し、`commit()` メソッドを呼び出して、変更をデータベースに確定します。

例

```
// 'customer' ポータルに新規ポータル行を作成
$new_row = $currentRecord->newRelatedRecord('customer');

// 新規ポータル行にフィールド値を設定
$new_row->setField('customer::name', $newName);
$new_row->setField('customer::company', $newCompany);

$result = $new_row->commit();
```

ポータルからレコードを削除

`delete()` メソッドを使用して、ポータル内のレコードを削除します。

例

```
$relatedSet = $currentRecord->getRelatedSet('customers');
/* 各ポータル行に対して実行 */
foreach ($relatedSet as $nextRow) {
 $nameField = $nextRow->getField('customer::name')
 if ($nameField == $badName ) {
  $result = $newRow->delete();
 }
}
```

値一覧の使用

値一覧とは、事前定義された選択値のセットです。FileMaker クラスは FileMaker Pro のデータベースで定義された値一覧を使用可能にするためのいくつかのメソッドを定義します。

特定レイアウト用のすべての値一覧名の取得

特定のレイアウトオブジェクトには、`listValueLists()` メソッドを使用して、値一覧名を含む配列を取得します。

例

```
$valueListNames = $currentLayout->listValueLists();
```

特定レイアウト用のすべての値一覧の配列の取得

特定のレイアウトオブジェクトには、`getValueListsTwoFields()` メソッドを使用して、すべての値一覧からの値を含む配列を取得します。返される配列は関連配列です。配列のキーは値一覧名で、配列の値は表示名およびそれぞれの値一覧からの選択値の一覧である関連配列です。

例

```
$valueListsArray = $currentLayout->getValueListsTwoFields();
```

メモ `getValueLists()` メソッドは、現在 FileMaker API for PHP で引き続き使用できますが、推奨されません。代わりに `getValueListsTwoFields()` メソッドの使用を推奨しています。

名前付きの値一覧の値の取得

特定のレイアウトオブジェクトには、`getValueListTwoFields()` メソッドを使用して、名前付きの値一覧向けに定義された選択値の配列を取得します。返された配列は関連配列で、キーである値一覧の 2 番目のフィールドと、配列の値である最初のフィールドの関連格納値からの表示値を含みます。

FileMaker Pro データベースの [値一覧に使用するフィールドの指定] ダイアログボックスで選択したオプションに応じて `getValueListTwoFields()` メソッドは、最初のフィールドの値のみ、2 番目のフィールドの値のみ、または値一覧の両方のフィールドの値の何れかを格納または表示された値として返します。

- **[2 番目のフィールドの値も表示]** が選択されなかった場合、`getValueListTwoFields()` メソッドは、格納および表示された値として値一覧の最初のフィールドから値を返します。
- **[2 番目のフィールドの値も表示]** と **[2 番目のフィールドの値のみを表示]** の両方が選択された場合、`getValueListTwoFields()` メソッドは、格納された値として最初のフィールドから値を、表示された値として 2 番目のフィールドから値を返します。
- **[2 番目のフィールドの値も表示]** が選択され、**[2 番目のフィールドの値のみを表示]** が選択されなかった場合、`getValueListTwoFields()` メソッドは、格納された値として最初のフィールドから値を、表示された値として最初と 2 番目の両方のフィールドから値を返します。

`getValueListTwoFields()` メソッドでイテレータを使用して表示および格納された値を検索します。

例

```
$layout = $fm->getLayout('customers');
$valuearray = $layout->getValueListTwoFields("region", 4);
foreach ($valuearray as $displayValue => $value) {
 ....
}
```

メモ

- `getValueList()` メソッドは、現在 FileMaker API for PHP で引き続き使用できますが推奨されません。代わりに `getValueListTwoFields()` メソッドの使用を推奨しています。
- `getValueListTwoFields()` メソッドを使用する場合、必ず `foreach` ループを使用して関連配列を扱います。`for` ループでは予想外の結果が返りますので使用しないでください。

検索条件の実行

FileMaker クラスは 4 種類の検索コマンドオブジェクトを定義します。

- Find All コマンド。85 ページの「Find All コマンドの使用」を参照してください。
- Find Any コマンド。85 ページの「Find Any コマンドの使用」を参照してください。
- Find コマンド。85 ページの「Find コマンドの使用」を参照してください。
- Compound Find コマンド。86 ページの「Compound Find コマンドの使用」を参照してください。

また、FileMaker クラスは 4 種類すべての検索コマンドに使用できるメソッドをいくつか定義します:

- `addSortRule()` メソッドを使用して、結果セットのソート方法を定義するルールを追加します。`clearSortRules()` メソッドを使用して、定義されたソートルールすべてをクリアします。
- `setLogicalOperator()` メソッドを使用して、論理積による検索から論理和による検索に切り替えます。
- `setRange()` メソッドを使用して、結果セットの一部のみをリクエストします。`getRange()` メソッドを使用して、現在の範囲定義を取得します。

  `setRange()` メソッドを使用すると、検索条件によって返されるレコードの数が減るので、ソリューションのパフォーマンスが向上します。たとえば、検索条件が 100 レコードを返す場合、100 レコードを一度に処理する代わりに、結果セットを 20 レコードずつの 5 グループに分けることができます。
- 検索コマンドとともに FileMaker スクリプトを実行できます。
  - 検索コマンドを実行する前にスクリプトを実行するには、`setPreCommandScript()` メソッドを使用します。
  - 結果セットをソートする前にスクリプトを実行するには、`setPreSortScript()` メソッドを使用します。
  - 結果セットの生成後のソート前にスクリプトを実行するには、`setScript()` メソッドを使用します。

Find All コマンドの使用

Find All コマンドを使用して、特定のレイアウトからすべてのレコードを取得します。`newFindAllCommand()` メソッドを使用し、特定のレイアウトを指定して FileMaker_Command_FindAll オブジェクトを作成します。その後、`execute()` メソッドを呼び出して検索条件を実行します。

例

```
$findCommand = $fm->newFindAllCommand('Form View');
$result = $findCommand->execute;
```

メモ Find All コマンドを使用する場合、1 ページにつき返すデフォルトの最大レコード数を指定することでコンピュータメモリの過負荷問題を回避します。

Find Any コマンドの使用

Find Any コマンドを使用して、特定のレイアウトからレコードをランダムに 1 つ取得します。`newFindAnyCommand()` メソッドを使用し、特定のレイアウトを指定して FileMaker_Command_FindAny オブジェクトを作成します。その後、`execute()` メソッドを呼び出して検索条件を実行します。

例

```
$findCommand = $fm->newFindAnyCommand('Form View');
$result = $findCommand->execute;
```

Find コマンドの使用

`newFindCommand()` メソッドを使用し、特定のレイアウトを指定して FileMaker_Command_Find オブジェクトを作成します。その後、`execute()` メソッドを呼び出して検索条件を実行します。

メモ レイアウトには必ず固有の名前を付けてください。同じ名前のレイアウトがデータベースに 2 つある場合、FileMaker API for PHP では区別ができません。また、API では大文字と小文字が区別されません。たとえば、Websites という名前のレイアウトと WebSites という名前の別レイアウトがデータベースにある場合、API でこの 2 つを区別することはできません。

`addFindCriterion()` メソッドを使用して、検索条件に基準を追加します。`clearFindCriteria()` メソッドを使用して、定義済みのすべての検索条件をクリアします。

例

フィールド名によるレコード検索:

```
$findCommand = $fm->newFindCommand('Form View');
$findCommand->addFindCriterion('Questionnaire ID',
$active_questionnaire_id);
$result = $findCommand->execute();
```

ソート順の追加:

```
$findCommand = $fm->newFindCommand('Customer List');
$findCommand->addSortRule('Title', 1, FILEMAKER_SORT_ASCEND);
$result = $findCommand->execute();
```

Compound Find コマンドの使用

Compound Find コマンドを使用すると、複数の検索条件オブジェクトを 1 つのコマンドにまとめることができます。Compound Find コマンドを作成するにはいくつかの方法があります:

- `newCompoundFindCommand()` メソッドを呼び出して、FileMaker_Command_CompoundFind オブジェクトを作成します。
- `newFindRequest()` メソッドを呼び出して、1 つ以上の FileMaker_Command_FindRequest オブジェクトを作成します。
- `add()` メソッドを使用して、Compound Find コマンドオブジェクトへ検索条件オブジェクトを追加します。
- `execute()` メソッドを呼び出して、Compound Find コマンドを実行します。

例

Compound Find コマンド:

```
// Compound Find コマンドオブジェクトを作成
$compoundFind = $fm->newCompoundFindCommand('Form View');

// 1 番目の検索条件を作成
$findreq1 = $fm->newFindRequest('Form View');

// 2 番目の検索条件を作成
$findreq2 = $fm->newFindRequest('Form View');

// 3 番目の検索条件を作成
$findreq3 = $fm->newFindRequest('Form View');

// 1 番目の検索条件の検索基準を指定
$findreq1->addFindCriterion('Quantity in Stock', '<100');

// 2 番目の検索条件の検索基準を指定
$findreq2->addFindCriterion('Quantity in Stock', '0');

// 3 番目の検索条件の検索基準を指定
$findreq3->addFindCriterion('Cover Photo Credit', 'The London Morning
News');

// Compound Find コマンドに検索条件を追加
$compoundFind->add(1,$findreq1);
$compoundFind->add(2,$findreq2);
$compoundFind->add(3,$findreq3);

// ソート順を設定
$compoundFind->addSortRule('Title', 1, FILEMAKER_SORT_DESCEND);

// Compound Find コマンドを実行
$result = $compoundFind->execute();

// 対象レコードからレコードを取得
$records = $result->getRecords();

// 対象レコード数を表示
echo 'Found '. count($records) . " results.<br><br>";
```

結果セット内のレコードの処理

- `getRecords()` メソッドを呼び出して、結果セット内の各レコードを含む配列を取得します。配列の各メンバーは、FileMaker_Record オブジェクトか、レコードのインスタンスを作成するための API 内のクラス名セットのインスタンスです。結果セットにレコードが含まれない場合、配列は空の可能性があります。
- `getFields()` メソッドを呼び出して、結果セット内のすべてのフィールドの名前の一覧を取得します。メソッドはフィールド名のみ返します。フィールドに関する追加情報が必要な場合は、関連するレイアウトオブジェクトを使用します。
- `getFoundSetCount()` メソッドを呼び出して、対象レコード全体のレコード数を取得します。
- `getFetchCount()` メソッドを呼び出して、フィルタ済みの対象セットのレコード数を取得します。検索コマンド上で範囲の引数を指定しない場合、この値は `getFoundSetCount()` メソッドの結果と同じになります。これは常に `count($response->getRecords())` の値と等しくなります。
- 特定のレコードには、フィールドの内容を文字列として返す `getField()` メソッドを使用します。
- 特定のレコードには、フィールドの内容を Unix のタイムスタンプ (日付の PHP 内部表現) として返す `getFieldAsTimestamp()` メソッドを使用します。
  - フィールドが日付フィールドの場合、タイムスタンプは、午前零時のフィールド日付を表します。
  - フィールドが時刻フィールドの場合、タイムスタンプは 1970 年 1 月 1 日のその時刻を表します。
  - フィールドがタイムスタンプフィールドの場合、FileMaker タイムスタンプ値が Unix のタイムスタンプに直接マップされます。
  - 指定されたフィールドが日付または時刻フィールドではない場合、または生成されたタイムスタンプが範囲外である場合は、`getFieldAsTimestamp()` メソッドは FileMaker_Error オブジェクトを返します。
- 特定のレコードには、バイナリデータとしてオブジェクトフィールドのオブジェクトを返す `getContainerData()` メソッドを使用します:

```
<IMG src="img.php?-url=<?php echo urlencode($record->getField('Cover
Image')); ?>">
echo $fm->getContainerData($_GET['-url']);
```

- 特定のレコードには、オブジェクトフィールドのオブジェクトに完全修飾 URL を返す `getContainerDataURL()` メソッドを使用します:

```
// イメージには HTML img タグを使用
echo '<img src="'.$fm->
getContainerDataURL($record->getField('container')) .'">';
// ムービーおよび PDF ファイルには HTML embed タグを使用
//echo '<embed src="'.$fm->
getContainerDataURL($record->getField('container')) .'">';
```

検索条件によって返されたポータルの行の制限

関連レコードが多くあるソリューションでは、ポータルレコードのクエリーを実行してソートすると、時間がかかる可能性があります。関連セットで表示するレコードの数を制限するには、検索条件とともに `setRelatedSetsFilters()` メソッドを使用します。`setRelatedSetsFilters()` メソッドには次の 2 つの引数を指定できます:

- 関連セットのフィルタ値: `layout` または `none`
  - 値 `none` を指定する場合、Web 公開エンジンによって、ポータル内のすべての行が返され、ポータルレコードは事前にソートされません。
  - 値 `layout` を指定する場合、FileMaker Pro の [ポータル設定] ダイアログボックスで指定された設定が優先されます。レコードは、[ポータル設定] ダイアログボックスで定義されたソートに基づいてソートされ、レコードセットは、指定された最初の行から開始するようにフィルタされます。
- 返されるポータルレコードの最小数: 整数値または `all`。
  - この値は、[ポータル設定] ダイアログボックスで [**垂直スクロールを許可**] の設定が有効になっている場合のみ使用されます。整数値を指定する場合、最初の行より後の行の数が返されます。`all` を指定する場合、Web 公開エンジンによって、すべての関連レコードが返されます。
  - [**垂直スクロールを許可**] 設定が無効になっている場合、[ポータル設定] ダイアログボックスの [**行数**] 設定によって、返される関連レコードの最大数が決定されます。

メモ [ポータル設定] ダイアログボックスの [**ポータルレコードのフィルタ**] 設定は、PHP クエリーでサポートされていません。[**ポータルレコードのフィルタ**] 設定に指定されている計算は無視されます。

コマンド、レコード、およびフィールドの入力値の制限の事前チェック

FileMaker クラスを使用すると、データをデータベースに確定する前に、Web サーバー上の PHP ソリューションのフィールドデータの入力値の制限を事前にチェックすることができます。

入力値の制限の事前チェックは、デフォルトでは無効になっています。有効にするには CLI を使用します。CLI ヘルプを参照してください。

入力値の制限の事前チェックを使用するかどうかを決定する際は、Web ユーザが入力しているデータの値の数を考慮します。ユーザが更新するフィールド数が少ない場合は、入力値の制限の事前チェックを行わないことでパフォーマンスを向上させることができます。ただし、ユーザが複数のフィールドにデータを入力する場合は、入力値の制限の事前チェックを使用して、入力値の制限エラーでデータベースによりレコードを拒否することでユーザの不便さを防ぐことができます。

FileMaker クラスを使用する場合、PHP エンジンは次のフィールドの入力値の制限を事前にチェックします:

- 空欄不可

  有効なデータは、空ではない文字列です。データは少なくとも 1 文字含んでいる必要があります。
- 数字

  有効なデータには、数字のみが含まれます。

- 最大文字数

  有効なデータには、多くとも指定された最大文字数しか含まれません。

- 西暦 4 桁の日付

  有効なデータは、M/D/YYYY の形式で西暦 4 桁の日付を表す文字列です。M は 1 から 12 までの数で、D は 1 から 31 までの数で、YYYY は 0001 から 4000 までの間の 4 桁の数です。たとえば、`1/30/3030` は、有効な西暦 4 桁の日付の値です。ただし、`4/31/2019` は、4 月には 31 日目がないので、無効な西暦 4 桁の日付の値です。日付の入力値の制限では、フォワードスラッシュ (/)、バックスラッシュ (\)、およびハイフン (-) を区切り文字としてサポートします。ただし、文字列には区切り文字を混ぜ合わせて使用することはできません。たとえば、`1\30-2019` などは無効です。

- 時刻

  有効なデータは、次の形式の 1 つを使用して 12 時間の時間の値を表示する文字列です:

  - h
  - h:m
  - h:m:s
  - h:m:s AM/PM
  - h:m AM/PM

  h は、1 から 12 までの数で、m および s は 1 から 60 までの数です。

PHP エンジンの入力値の制限の事前チェックでは、フィールドの種類に基づいてフィールドデータを暗黙にチェックする機能をサポートしています:

- 日付

  日付フィールドとして定義されているフィールドは、年の値には 0〜4 桁の値を含めることができるという点 (年の値は空欄可) を除いて、「西暦 4 桁の日付」の入力値の制限のルールに従ってチェックされます。たとえば、`1/30` は、年が指定されていませんが、有効な日付です。

- 時刻

  時刻フィールドとして定義されているフィールドは、時 (H) を表す部分は 24 時間の値をサポートするために 1 から 24 までの数字にすることができるという点を除いて、「時刻」の入力値の制限のルールに従ってチェックされます。

- タイムスタンプ

  タイムスタンプフィールドとして定義されているフィールドは、時刻の部分は「時刻」の入力値の制限のルールに従ってチェックされ、日付の部分は「日付」の入力値の制限のルールに従ってチェックされます。

FileMaker クラスでは、FileMaker Pro で利用可能なフィールドの入力値の制限オプションのすべての入力値の制限を事前にチェックすることはできません。次の入力値の制限オプションは、データが確定される際にデータベースに存在するすべてのデータの状態に依存しているため入力値の制限は事前にチェックできません:

- ユニークな値
- 既存値
- 下限値/上限値
- 値一覧名
- 計算式で制限

コマンド内のレコードの入力値の制限の事前チェック

コマンドオブジェクトには、`validate()` メソッドを使用して、PHP エンジンが強制できる入力値の制限の事前チェックのルールに対して 1 つのフィールドまたはコマンド全体をチェックします。オプションのフィールド名の引数を渡した場合、そのフィールドのみの入力値の制限が事前にチェックされます。

入力値の制限の事前チェックをパスした場合、`validate()` メソッドは TRUE を返します。入力値の制限の事前チェックをパスできなかった場合、`validate()` メソッドは、何が入力値の制限をパスできなかったかについての詳細を含む FileMaker_Error _Validation オブジェクトを返します。

レコードの入力値の制限の事前チェック

レコードオブジェクトには、`validate()` メソッドを使用して、PHP エンジンが強制できる入力値の制限の事前チェックのルールに対して 1 つのフィールドまたはレコード内のすべてのフィールドをチェックします。オプションのフィールド名の引数を渡した場合、そのフィールドのみの入力値の制限が事前にチェックされます。

入力値の制限の事前チェックをパスした場合、`validate()` メソッドは TRUE を返します。入力値の制限の事前チェックをパスできなかった場合、`validate()` メソッドは、何が入力値の制限をパスできなかったかについての詳細を含む FileMaker_Error _Validation オブジェクトを返します。

フィールドの入力値の制限の事前チェック

フィールドオブジェクトには、`validate()` メソッドを使用して、与えられた値がフィールドに対して有効かどうかを決定します。

入力値の制限の事前チェックをパスした場合、`validate()` メソッドは TRUE を返します。入力値の制限の事前チェックをパスできなかった場合、`validate()` メソッドは、何が入力値の制限をパスできなかったかについての詳細を含む FileMaker_Error _Validation オブジェクトを返します。

入力値の制限エラーの処理

入力値の制限の事前チェックが失敗すると、返される FileMaker_Error_Validation オブジェクトには入力値の制限の失敗ごとに 3 つの要素の配列が含まれます:

1. 入力値の制限の事前チェックに失敗したフィールドオブジェクト

2. 失敗した入力値の制限ルールを示す入力値の制限の定数値:

1 - FILEMAKER_RULE_NOTEMPTY

2 - FILEMAKER_RULE_NUMERICONLY

3 - FILEMAKER_RULE_MAXCHARACTERS

4 - FILEMAKER_RULE_FOURDIGITYEAR

5 - FILEMAKER_RULE_TIMEOFDAY

6 - FILEMAKER_RULE_TIMESTAMP_FIELD

7 - FILEMAKER_RULE_DATE_FIELD

8 - FILEMAKER_RULE_TIME_FIELD

3. 入力値の制限の事前チェックに失敗したフィールドに入力された実際の値

次のオブジェクトを FileMaker_Error_Validation オブジェクトとともに使用することもできます:

- `isValidationError()` メソッドを使用して、エラーが入力値の制限エラーかどうかをテストします。
- `numErrors()` メソッドを使用して、失敗した入力値の制限ルールの数を取得します。

例

```
// Add リクエストを作成
$addrequest = $fm->newAddCommand('test', array('join' => 'added', 'maxchars'
=> 'abcx', 'field' => 'something' , 'numericonly' => 'abc'));

// すべてのフィールドを検証
$result = $addrequest->validate();

// validate() メソッドがエラーを返した場合、失敗したフィールド名、エラーコード番号、お
よび値を表示
if (FileMaker::isError($result)) {
 echo 'Validation failed:' ."\n";
 $validationErrors= $result->getErrors();
 foreach ($validationErrors as $error) {
  $field =  $error[0];
  echo 'Field Name: ' .  $field->getName()."\n";
  echo 'Error Code: ' .  $error[1] ."\n";
  echo 'Value: ' .  $error[2] ."\n";
 }
}
```

結果

```
Validation failed:
Field Name: numericonly
Error Code: 2
Value: abc
Field Name: maxchars
Error Code: 3
Value: abcx
```

エラー処理

FileMaker クラスは PHP ソリューション内で発生するエラーを処理するのに役立つ FileMaker_Error オブジェクトを定義します。

コマンドを実行するとエラーが発生する可能性があります。エラーが発生すると、コマンドが FileMaker_Error オブジェクトを返します。コマンドを実行した際に返されるエラーは毎回チェックするようにしてください。

次のメソッドを使用して、FileMaker_Error オブジェクト内で示されるエラーの詳細を調べます。

- `isError()` メソッドを呼び出して、変数が FileMaker Error オブジェクトかどうかテストします。
- `numErrors()` メソッドを呼び出して、発生したエラーの数を取得します。
- `getErrors()` メソッドを呼び出して、発生したエラーを説明する配列の中から配列を 1 つ取得します。
- `getMessage()` メソッドを呼び出して、エラーメッセージを表示します。

例

```
$result = $findCommand->execute();
if (FileMaker::isError($result)) {
 echo "<p>Error: " .$result->getMessage() ."</p>";
 exit;
}
```

FileMaker Error オブジェクトとともに返されるエラーコードについては、付録 A「カスタム Web 公開のエラーコード」を参照してください。

第 9 章
サイトのステージング、テスト、および監視

この章では、カスタム Web 公開サイトを運用環境に展開する前にステージングおよびテストを行う手順について説明します。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する手順についても説明します。

カスタム Web 公開サイトのステージング

サイトを正しくテストする前に、必要なファイルをステージングサーバーの正しい場所に移動またはコピーする必要があります。

1. 第 2 章「データベースのカスタム Web 公開の準備」にあるすべての手順を実行します。
2. カスタム Web 公開が CLI を使用して有効に設定されて正しく構成されていることを確認します。FileMaker Server ヘルプを参照してください。
3. 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します。
4. 展開した FileMaker Server の Web サーバーコンポーネントにサイトのファイルをコピーまたは移動します。

   次のディレクトリにサイトファイルをコピーまたは移動します。

   - HTTP または HTTPS での IIS (Windows):
     `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥HTTPServer¥Conf`
     [ドライブ] は FileMaker Server プライマリマシンのドライブです。
   - HTTP での Apache (macOS):
     `/ライブラリ/FileMaker Server/HTTPServer/htdocs`
   - HTTPS での Apache (macOS):
     `/ライブラリ/FileMaker Server/HTTPServer/htdocs/httpsRoot`

   メモ　カスタム SSL 証明書をインポートすると、データベースサーバークライアント接続で SSL が使用され、HTTP 接続が HTTPS にルーティングされます。ご使用のカスタム SSL 証明書とともに、サイトの PHP ファイルを共有するための HTTPS ディレクトリを使用してください。

5. まだ操作を行っていない場合は、参照先オブジェクトフィールドをプライマリマシン上の適切な場所にコピーまたは移動します。
   - データベースファイルが FileMaker Server 展開のデータベースサーバーコンポーネントに適切に共有されていてアクセス可能であり、FileMaker Pro 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には、オブジェクトフィールドの内容を移動する必要はありません。
   - 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、レコードを作成または編集するときに、その参照されているオブジェクトが FileMaker Pro の「Web」フォルダに保存されている必要があります。サイトをステージングするには、参照されているオブジェクトを、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。

- FileMaker Pro を使用してデータベースをアップロードする場合、外部に保存されたオブジェクトフィールドデータは、プロセスの一環として FileMaker Server にアップロードされます。FileMaker Server へのデータベースファイルの転送については、FileMaker Pro ヘルプを参照してください。
- オブジェクトを外部に保存したオブジェクトフィールドを使用するデータベースを手動でアップロードするには、17 ページの「外部に保存されたデータを含むオブジェクトフィールド」の説明のとおりに「RC_Data_FMS」フォルダのサブフォルダに参照されているオブジェクトをコピーするか移動する必要があります。

6. Web アプリケーションのコンポーネントをプライマリマシンにコピーします。カスタム Web 公開 with XML の場合、ご使用の Web アプリケーションは、XML データを処理してから別のアプリケーションまたはクライアントに送ります。

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、そのサイトが意図どおりに表示され、機能することを確認してください。

- 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サインインして、さまざまなアクセス権セットが期待どおりに動作することを確認する。権限のないユーザによるデータへのアクセスやデータの変更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、18 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 ブラウザを使用してサイトをテストする。
- FileMaker API for PHP を使用してソリューションを作成する場合は、Cookie のサポートを有効にしてソリューションを構築することをお勧めします。FileMaker API for PHP の応答時間は、Cookie を有効にすると向上します。カスタム Web 公開の機能を使用するために Cookie は必要ありませんが、Cookie を有効にすると Web 公開エンジンがセッション情報をキャッシュできます。

  メモ URL に http://127.0.0.1/ を使用することによって、ネットワークに接続せずにプライマリマシンでサイトを表示およびテストできます。

- PHP ソリューションの場合は、http://127.0.0.1/<サイトパス> を使用します。<サイトパス> は使用しているサイトのホームページへの相対パスです。
- XML ソリューションでの URL 構文の詳細については、27 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。

XML 出力をテストするためのスタイルシート

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、リクエストされた 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。

```
<?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 公開エンジンログ
- Web サーバーモジュールのエラーログ
- Tomcat ログ

Web サーバーのアクセスログとエラーログの使用

IIS (Windows): 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。Microsoft IIS Web サーバーのマニュアルを参照してください。

Apache (macOS): 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。Apache Web サーバーのマニュアルを参照してください。

メモ 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 公開エンジンのログの使用

Web 公開エンジンは、アプリケーションエラー、使用状況エラー、システムエラーを含むあらゆる Web 公開エンジンエラーが含まれる「wpe.log」というログファイルを生成します。

「wpe.log」ファイルは Web 公開エンジンが動作している場合に生成されます。(Web 公開エンジンが動作しているかを確認するには Admin Console を参照してください。)

「wpe.log」ファイルは、展開した FileMaker Server の Web 公開エンジンコンポーネントに格納されます:

- Windows:
  `[ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥Logs¥wpe.log`
  [ドライブ] はシステムが起動されるプライマリドライブです。
- macOS: /ライブラリ/`FileMaker Server/Logs/wpe.log`

Web 公開エンジンログの形式

「wpe.log」ファイルは各エントリに次の形式を使用します:

```
[TIMESTAMP_GMT] [WPC_HOSTNAME] [CLIENT_IP:PORT] [ACCOUNT_NAME] [MODULE_TYPE]
[SEVERITY] [FM_ERRORCODE] [RETURN_BYTES] [MESSAGE]
```

各要素の意味は次のとおりです:

- `[TIMESTAMP_GMT]` は、グリニッジ標準時 (GMT) のエントリの日付と時刻です。
- `[WPC_HOSTNAME]` は、プライマリマシンのマシン名です。
- `[CLIENT_IP:PORT]` は、XML リクエストが生成されたクライアントの IP アドレスとポートです。
- `[ACCOUNT_NAME]` は、共有されている FileMaker Pro データベースにサインインするために使用されるアカウント名です。
- `[MODULE_TYPE]` は: カスタム Web 公開 with XML のリクエストに使用する XML、またはカスタム Web 公開 with PHP のリクエストに使用する PHP です。
- `[SEVERITY]` は、情報メッセージを示す INFO、またはエラーメッセージを示す ERROR です。
- `[FM_ERROR_CODE]` は、エラーメッセージを返すエラー番号です。エラー番号は、FileMaker Pro データベースのエラーコードの場合があります (102 ページの「FileMaker Pro データベースのエラーコード番号」を参照してください)。

  また、エラー番号は先頭に文字列「HTTP:」の付いた HTTP エラー番号の場合もあります。
- `[RETURN_BYTES]` は、リクエストによって返されたバイト数です。
- `[MESSAGE]` は、ログエントリに関する追加情報を提供します。

Web 公開エンジンログメッセージ

例

次の例は、「wpe.log」ファイルに含まれるメッセージの種類を示しています。

Web 公開エンジンの起動および停止:

```
2019-06-02 15:15:31 -0700   -   -   -   -   INFO   -   -   FileMaker Server
Web Publishing Engine started.
2019-06-02 15:46:52 -0700   -   -   -   -   INFO   -   -   FileMaker Server
Web Publishing Engine stopped.
```

XML クエリーリクエストの成功または失敗:

```
2019-06-02 15:21:08 -0700  WPC_SERVER  192.168.100.101:0  jdoe  XML  INFO
0   3964   "/fmi/xml/fmresultset.xml?-db=Contacts&-lay=Contact_Details&-
findall"
2019-06-02 15:26:31 -0700   WPC_SERVER   192.168.100.101:0   jdoe   XML
ERROR   5   596   "/fmi/xml/fmresultset.xml?-db=Contacts&-
layout=Contact_Details&-findall"
```

スクリプトエラー :

```
2019-06-02 17:33:12 -0700  WPC_SERVER  192.168.100.101:0  jdoe  -  ERROR
4  -  Web Scripting Error: 4, File: "10b_MeetingsUpload", Script: "OnOpen",
Script Step: "Show Custom Dialog"
```

カスタム Web 公開設定の変更:

```
2019-06-09 10:59:49 -0700  WPC_SERVER  192.168.100.101:0  jdoe  -  INFO
-   -   XML Web Publishing Engine is enabled.
```

システムエラー :

```
2019-06-02 15:30:42 -0700   WPC_SERVER   192.168.100.101:0   jdoe   XML
ERROR   -   - Communication failed
```

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を記録す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」フォルダ内の「Logs」フォルダに格納されます。

Tomcat ログの使用

内部 Web サーバーエラーが原因で FileMaker Server に問題が発生した場合は、Tomcat ログを参照することをお勧めします。Tomcat ログは、FileMaker Server 展開の Web サーバーコンポーネントに格納されます:

- Windows: [ドライブ]:¥Program Files¥FileMaker¥FileMaker Server¥Web Publishing¥publishing-engine¥jwpc-tomcat¥logs
  [ドライブ] はシステムが起動されるプライマリドライブです。
- macOS: /ライブラリ/FileMaker Server/Web Publishing/publishing-engine/jwpc-tomcat/logs

付録 A
カスタム Web 公開のエラーコード

Web 公開エンジンは、データベースのエラーコードと、XML データリクエスト中に発生するクエリー文字列エラーを生成します。

更新されたエラーコードについては、ナレッジベースを参照してください。

XML 形式におけるエラーコード番号

データがリクエストされると、Web 公開エンジンは XML 形式で公開されているデータベースのエラーコードを生成します。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ある `fmresultset` 文法の `<error code>` 要素、または `FMPXMLRESULT` や `FMPXMLLAYOUT` 文法の `<ERRORCODE>` 要素に挿入されます。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。

例

`fmresultset` 文法のデータベースエラーコード:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE fmresultset PUBLIC "-//FMI//DTD fmresultset//EN"
"http://192.168.123.101/fmi/xml/fmresultset.dtd">
<fmresultset xmlns="http://www.filemaker.com/xml/fmresultset"
version="1.0">
 <error code="0"></error>
```

`FMPXMLRESULT` 文法のデータベースエラーコード:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLRESULT PUBLIC "-//FMI//DTD FMPXMLRESULT//EN"
"http://192.168.123.101/fmi/xml/FMPXMLRESULT.dtd">
<fmpxmlresult xmlns="http://www.filemaker.com/fmpxmlresult">
 <ERRORCODE>0</ERRORCODE>
```

`<error code>` または `<ERRORCODE>` 要素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。

FileMaker Pro データベースのエラーコード番号

FileMaker Pro エラーコードについては、FileMaker Pro ヘルプを参照してください。

FileMaker Server には、テクノロジーが無効であることを示すエラーコード 959 があります。

たとえば、サーバー管理者が CLI を使用してカスタム Web 公開を無効にすると、XML クエリーはエラーコード 959 を返します。

例

FMPXMLLAYOUT 文法のエラーコード 959:

```
<FMPXMLLAYOUT>
 <ERRORCODE>959</ERRORCODE>
 <LAYOUT DATABASE="" NAME=""/>
 <VALUELISTS/>
</FMPXMLLAYOUT>
```

索引

H

I

L

M

N

P

Q

R

S

T

U

V

W

X

あ

え

お

か

き

く

け

こ

さ

し

す

せ

た

て

と

に

は

ひ

ふ

ほ

ま

め

ゆ

よ

り

れ

ろ