Sterling Selling and Fulfillment Foundation

データベースの拡張

バージョン 9.1

Sterling Selling and Fulfillment Foundation

データベースの拡張

バージョン 9.1

お願い

本書および本書で紹介する製品をご使用になる前に、 53 ページの『特記事項』に記載されている情報をお読みください。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： Sterling Selling and Fulfillment Foundation
Extending the Database
Version 9.1

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.2

目次

第 1 章 カスタマイズ・プロジェクトのチェックリスト

カスタマイズ・プロジェクト

Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales をカスタマイズまたは拡張するプロジェクトは、必要な変更のタイプによってさまざまです。ただし、ほとんどのプロジェクトは、特定の順序で最適に実行される、相互接続された変更の連続が関係します。チェックリストは、カスタマイズ・タスクの最も一般的な順序を特定し、文書セットのどのガイドが各ステージの詳細を提供するかを示します。

開発環境の準備

WebLogic、WebSphere®、または JBoss アプリケーション・サーバーにアプリケーションをデプロイするかどうかなど、実稼働環境をミラーリングする開発環境を設定します。これを行うことによって、拡張をリアルタイム環境でテストできるようになります。

実稼働環境にアプリケーションをインストールしてデプロイする手順と同じ手順で開発環境にアプリケーションをインストールしてデプロイします。詳細については、システム要件およびインストール文書を参照してください。

Microsoft COM+ でアプリケーションをカスタマイズするオプションがあります。Microsoft COM+ を使用すると、セキュリティーの向上、パフォーマンスの向上、サーバー・アプリケーションの管理の容易性の向上、混合環境のクライアントのサポートなどの利点を得られます。これを選択する場合、インストールの指示について詳しくは、*カスタマイズ基本ガイド*を参照してください。

カスタマイズの計画

新しいメニュー項目を追加していますか。または、サインイン画面またはロゴをカスタマイズしていますか。または、表示またはウィザードをカスタマイズしていますか。または新しいテーマまたは新しい画面を作成していますか。カスタマイズのそれぞれのタイプの範囲と複雑さはさまざまです。

背景については、実行できる変更のタイプを要約し、ファイル名、キーワード、およびその他の一般的な規則に関するガイドラインを提供している*カスタマイズ基本ガイド*を参照してください。

データベースの拡張

多くのカスタマイズ・プロジェクトにおいて、最初のタスクは、データベースを拡張し、後に行う UI または API の他の変更をデータベースがサポートするようにすることです。この指示については、以下のトピックについて説明しているデータベースの*拡張ガイド*を参照してください。

- データベースで変更できるものおよび変更できないものに関する重要なガイドライン。
- API の変更に関する情報。任意の API が影響を受けるようなデータベース表の変更を行った場合、これらの API のテンプレートを拡張する必要があります。そうしない場合、データベースへのデータの格納またはデータベースからのデータの取り出しを実行できません。この手順は、テーブルの変更が API に影響する場合に必要になります。
- エンティティー・レベルでレコードをトラッキングしてレコード管理を向上させるために監査参照を生成する方法。この手順はオプションです。

API に対するその他の変更の実施

アプリケーションは、標準 API またはカスタム API の呼び出しまたは起動を実行できます。API に関する背景およびサービス・タイプ、動作、ならびにセキュリティーのサービス・アーキテクチャーについては、*API のカスタマイズ・ガイド*を参照してください。このガイドでは、以下のタイプの変更について説明します。

- UI でのデータの表示および UI で行われた変更のデータベースへの保存を行う標準 API の呼び出し。
- 拡張サービス定義およびパイプライン構成でカスタム・ロジックを実行するためのカスタマイズ API の呼び出し。
- API は、入力および出力の XML を使用し、データベースにデータを格納し、またデータベースからデータを取り出します。これらの API 入力および出力の XML ファイルを拡張しない場合、ビジネス・ロジック実行時に UI で必要な結果を取得できない場合があります。
- それぞれの API 入力および出力の XML ファイルには、このファイルに関連付けられた DTD および XSD があります。入力および出力の XML を変更したときには必ず、対応する DTD および XSD を生成し、データ保全性を確保する必要があります。拡張 XML に対して DTD および XSD を生成しないと、不整合データを取得する場合があります。

UI のカスタマイズ

IBM® アプリケーションは、いくつかの UI フレームワークをサポートしています。アプリケーションおよび実行するカスタマイズに応じて、これらのフレームワークの 1 つのみまたはいくつかで作業できます。各フレームワークには、メニュー項目、ロゴ、テーマなどのコンポーネントをカスタマイズする独自のプロセスがあります。

必要なフレームワークに応じて、以下のガイドのいずれかを参照してください。

- *カスタマイズ・ガイド (コンソール JSP インターフェース) (Customizing the Console JSP Interface Guide)*
- *カスタマイズ・ガイド (Swing インターフェース) (Customizing the Swing Interface Guide)*
- *カスタマイズ・ガイド (モバイル・デバイス向けユーザー・インターフェース) (Customizing User Interfaces for Mobile Devices Guide)*

- *カスタマイズ・ガイド (リッチ・クライアント・プラットフォーム) (Customizing the Rich Client Platform Guide)* および *RCP 拡張性ツール使用ガイド (Using the RCP Extensibility Tool Guide)*
- *カスタマイズ・ガイド (Web UI フレームワーク) (Customizing the Web UI Framework Guide)*

必要なフレームワークに応じて、以下のガイドのいずれかを参照してください。

- *カスタマイズ・ガイド (コンソール JSP インターフェース) (Customizing the Console JSP Interface Guide)*
- *カスタマイズ・ガイド (Swing インターフェース) (Customizing the Swing Interface Guide)*
- *カスタマイズ・ガイド (モバイル・デバイス向けユーザー・インターフェース) (Customizing User Interfaces for Mobile Devices Guide)*
- *カスタマイズ・ガイド (リッチ・クライアント・プラットフォーム) (Customizing the Rich Client Platform Guide)* および *RCP 拡張性ツール使用ガイド (Using the RCP Extensibility Tool Guide)*
- *カスタマイズ・ガイド (Web UI フレームワーク) (Customizing the Web UI Framework Guide)*

トランザクションの拡張

条件ビルダーを拡張し、外部システムと統合することによって、アプリケーションの標準機能を拡張できます。トランザクション・タイプの背景、セキュリティー、動的変数、および条件ビルダーの拡張については、*トランザクションの拡張ガイド* および *条件ビルダーの拡張ガイド* を参照してください。これらのガイドでは、以下のタイプの変更について説明します。

- 条件ビルダーを拡張して、カスタム・ビジネス・ロジックを実行し、属性の静的セットを使用するための複雑で動的な条件の定義。
- 変数を定義して、アクション、エージェント、およびサービスの構成に属するプロパティーの動的構成。
- 誰がどのデータにアクセスできるか、どれだけの量を表示できるか、およびデータで何を実行できるかを制御するトランザクション。データ・セキュリティーの設定。
- カスタムの時間トリガー・トランザクションの作成。ご使用のアプリケーションが提供する時間トリガー・トランザクションの呼び出しおよびスケジューリングとほぼ同じ方法でカスタムの時間トリガー・トランザクションの呼び出しおよびスケジューリングを実行できます。
- カスタムの時間トリガー・トランザクションを外部トランザクションと調整し、イベントの起動、外部プログラムの呼び出し、またはカスタム API またはサービスの呼び出しのいずれかによってカスタムの時間トリガー・トランザクションを実行します。

カスタマイズまたは拡張のビルドおよびデプロイ

必要なカスタマイズを実行した後、カスタマイズまたは拡張をビルドしてデプロイする必要があります。

1. カスタマイズまたは拡張を確認できるように、テスト環境でこれらをビルドしてデプロイします。
2. 準備ができたら、同じプロセスを繰り返して、実稼働環境でカスタマイズおよび拡張をビルドしてデプロイします。

このプロセスの指示については、以下のトピックについて説明しているカスタマイズ基本ガイドを参照してください。

- 標準リソース、データベース拡張、およびその他の拡張 (テンプレート、外部プログラム、および Java インターフェースなど) のビルドおよびデプロイ。
- エンタープライズ・レベルの拡張のビルドおよびデプロイ。

第 2 章 データベースを拡張するためのガイドライン

データベースを拡張するためのガイドライン

シーケンスの変更はサポートされていません。エンティティー・リポジトリー内の sequences.xml ファイルを編集しても、効果はありません。さらに、IBM は、データベース・ツールを介してシーケンスを変更することを推奨していません。ほとんどの環境では、そのような変更は不要であり、逆効果です。

データベースの拡張について

Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales データベースの一部の機能は、変更することができません。それらを変更しようとした場合、データに悪影響はありませんが、試みた変更はデータベースに取り込まれません。アプリケーションは、以下の変更を許可しません。

- テーブルの既存の列
- テーブルの主要キー
- テーブルの固有キー
- ビュー

データベースを拡張しようとする場合、変更による影響と、他の領域に与える影響の程度を考慮してください。

注: テーブルを変更し、デプロイメントで Sterling Business Center コンポーネントを使用する場合、そのテーブルに関連付けられたビューも変更する必要があります。

エンティティー・リレーションシップ・ダイアグラム

Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales データベースについて詳しくは、*INSTALL_DIR*/xapidocs/erd/html/erd.html ファイルを使用してエンティティー・リレーションシップ・ダイアグラム (ERD) を参照してください。それらの ERD は、以下の情報を提供します。

- 列を追加して拡張できるテーブルを示します。
- ハングオフ関係を持つことができるテーブルを示します。
- テーブル間の関係 (オーダー、出荷、支払いなどの論理エンティティー間の関係を理解するのに役立てるため)。
- インデックスの詳細。各テーブルは、主要キーによってインデックスが付けられています。ほとんどのテーブルには、論理固有キーを構成する列から成る固有インデックスも付けられています。さらに、一部のテーブルには、クエリーをサポートするための代替インデックスが付けられています。
- 複数のテーブル間の相互関係を示すビュー。

エンティティー・データベースの XML ファイル

アプリケーション・データベースに付属する標準テーブルは、一連のエンティティー XML ファイルで定義されており、データベース定義 XML ファイルとも呼ばれています。各エンティティー XML ファイルには、複数のテーブル定義が含まれる場合があります。それらのテーブルについて詳しくは、*INSTALL_DIR*/repository/entity ディレクトリー内のファイルを参照してください。それらのエンティティー XML ファイル内で、エンティティーはテーブルを表し、属性は列を表します。

SQLProxy ツールとは

SQLProxy は、JDBC ドライバーを検索しながら SQL ステートメントをデータベースにキャプチャーし、JDBC トレースを生成します。SQLProxy は、使用されているデータベースのタイプに関係なく、すべての SQL ステートメントをキャプチャーします。別のツールである SQLProxy Analyzer は、SQLProxy トレース・レコードに基づいてレポートを作成します。SQLProxy ツールを使用すると、クエリーに対するデータベース応答時間を分析するのに役立ちます。これにより、キャパシティー・プランニングを促進し、データベース・サーバーを最大限に有効使用することができます。これらのツールは、IBM お客様サポートで指示された場合のみ、使用するようにしてください。

SQLProxy ツールのセットアップ

このタスクについて

SQLProxy ツールをセットアップするには、アプリケーション・サーバーまたは非アプリケーション・サーバーを開始する前に、customer_overrides.properties ファイルにプロパティーを追加する必要があります。これにより、jdbc.properties にあるこれらのプロパティーの設定がオーバーライドされます。

注: yfs.properties ファイル (yfs.enable.proxy.sql.logging、yfs.enable.source.logging、yfs.proxy.log.dir など) にある SQLProxy プロパティーは非推奨です。サード・パーティー製のアプリケーション・サーバー (WebSphere、JBoss、WebLogic) または非アプリケーション・サーバーのどちらを使用していても、すべてのユーザーが、次の JDBC プロパティーを customer_overrides.properties ファイルに追加して、ツールを構成する必要があります。

手順

1. *install_dir*\properties\ フォルダーで、customer_overrides.properties ファイルを見つけます。見つからない場合は、作成してください。
2. customer_overrides.properties で次のプロパティーを設定します。

   ```
   jdbcService.proxyLoggingEnabled=Y
   ```

   (ロギングが必要かどうかを判断します)

   ```
   jdbcService.proxySourceLogging=Y
   ```

   (追加の呼び出し元情報を提供します)

```
jdbcService.proxyLogDir=<INSTALL_DIR>\logs
```

(トレース・ファイルが生成されるディレクトリー)

SQLProxy ツールの開始および使用

このタスクについて

ログ・レベルが TIMER 以上に設定されていると、すべてのデータベース操作に対してログが生成されます。トレースを開始するには、ログ・レベル設定後に API またはエージェントを実行します。proxyLogDir プロパティーで指定されたロケーションにトレース・ファイルが生成されます。各ファイルには、次のような詳細情報が含まれます。

- *.log ファイルは、コンポーネント名、システム始動および終了時刻、操作実行時刻 (個別および累積)、操作名、SQL ステートメント、Java クラス名を提供
- *.tail ファイルは、長時間実行中のクエリーを把握する際に有用 (主にパフォーマンス・テストで使用)

SQLProxy の停止

このタスクについて

ロギングを停止するには、TIMER トレースを削除します。プロキシーを完全に無効にするには、enabled プロパティーを N に設定します。「Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Salesシステム*管理ガイド*」の「*コンポーネント・トレースの停止 (Stopping a Component Trace)*」を参照してください。

結果の分析

ファイルのコンテンツを .csv ファイルにエクスポートし、XSL で開くことができます。JDBC トレース・レコードの分析および要約を行う、SQLProxy Analyzer というサポート・ツールもあります。このツールは、Eclipse で MTCP の一部として利用することができます。

注意:
このツールをご使用の際は、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales お客様サポートに連絡してください。

SQLProxy Analyzer について

SQLProxy Analyzer は、SQLProxy ログのデータ分析に使用されるツールです。SQLProxy Analyzer には、次のような機能があります。

- ディレクトリー内の複数のログ・ファイルの処理や単一のログ・ファイルの処理が可能
- 各ログ・ファイル (データベース接続を示す) に対して、作業単位 (UOW) のサマリーを作成 (1 つの接続に複数の作業単位が存在する場合もあります)
- 実行数および SQL の総コストの計算が可能
- 多くの SQL インスタンスや高平均 SQL 時間がある場合に有用な応答時間バケットによって、どのバケットが SQL 応答時間に該当するかを表示

- 実行計画 (Analyzer によって自動的に生成される) の提供によって、任意の Oracle データベースが指定できるようになり (SQL がキャプチャーされたデータベースである必要はなし)、これによって、Analyzer に最新の Oracle 統計を持つ非常に大規模なデータベースを指定することが可能
- 各作業単位に対して、以下を提供
  - 全 SQL のサマリー (非常に大容量のファイルから少量のサマリーを取得するために SQL はグループ化されます)
  - LOCK HOLDING 時間 (これにより、その他のワークロードに影響を与える可能性があるロックをシステムが取り込むかどうかを確認できます)
  - Oracle EXPLAIN 計画 (データベースが大規模になるにつれて、あるクエリーが低速化する可能性があるかどうかを見極めるのに役立ちます)

標準テーブルに列を追加するためのガイドライン

標準テーブルの列を拡張する場合、以下の考慮事項に留意してください。

- テーブルへの列の追加は、ERD で指定されているテーブルに対してのみ可能です。
- どの列も削除または変更することはできません。
- アプリケーションをインストールする前か後のいずれかで列を追加することができます。
- アプリケーション・データベース・テーブルに追加したすべての列に対して、データベース・フレームワークに関するデフォルト値を指定する必要があります。
- 以下のフィールドに、NULL 可能な列を使用することはできません。
  - 主要キー属性
  - エンティティー・リレーションシップ

  したがって XML エンティティーでは、上記に示す列以外のすべての列に対して、`Nullable="true"` を設定することができます。
- `Long` データ・タイプの列を追加することはできません。
- マップを読み取ったりマップを公開したりする (GetOrderNoUE 外部プログラムなどの) アプリケーション・コンポーネント (イベントや外部プログラムなど) を使用する場合、マップ内の拡張フィールドには Extn_ の接頭辞を付ける必要があります。

注: Oracle データベースでは、新規インストールの場合、LONG データ・タイプの代わりに CLOB データ・タイプが生成されます。ただし、アップグレードの場合、既存の LONG 列は変更されません。

注: DB2® データベースでは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales フレームワークによって、タイム・スタンプとして日付データ・タイプが生成されます。

列のデータ圧縮のサポートを追加するためのガイドライン

標準テーブル、カスタム・テーブル、またはハングオフ・テーブルについて、新規または既存の列のデータ圧縮を使用可能にできます。データ圧縮が実装されている場合、テーブルのサイズを縮小することができます。

- 新しく追加された列の場合、エンティティー XML に `CompressionSupported="true"` と `UseCompression="true"` の両方を追加することによって、データ圧縮を使用可能にできます。
- 標準テーブル内の既存の列の場合、UseCompression 属性の値をオーバーライドすることによってデータ圧縮を使用可能にできます。

  注: CompressionSupported 属性が `true` に設定されている既存の列に対してのみ、データ圧縮を使用可能にできます。

データ圧縮のサポートを標準テーブルの列に追加する場合、以下の考慮事項に留意してください。

- データ圧縮は、通常は直接照会されない大規模なテキスト・データ (監査レコード、エラー・スタック・トレース、XML テンプレート・データなど) を含む列に対してのみ使用する必要があります。これらのレコードは、その性質上、データベース内で大量のスペースを使用する可能性があります。
- CompressionSupported="true" を使用する列は、リスト API からの照会操作をサポートしません。この設定は、エンティティー XML 内で QueryAllowed 属性を true に設定することによってオーバーライドできます。
- デフォルトでは、`CompressionSupported="true"` を使用する列は、インデックス制約または固有制約の一部として使用することはできません。この設定は、エンティティー XML 内のインデックス要素で AllowCompressedColumns 属性を true に設定することによってオーバーライドできます。
- 主要キー列に、圧縮可能な列としてマークを付けることはできません。
- データ圧縮は、列の最大サイズが 500 バイト以上の場合にのみ使用可能にする必要があります。500 バイト未満の列にデータ圧縮のマークが付けられている場合、警告が発生します。
- デフォルトでは、データ圧縮は GZip アルゴリズムを使用して実行されます。このデフォルトのデータ圧縮ロジックは、customer_overrides.properties ファイル内の「yfs.db.compression.class=クラス名」プロパティーを設定することによってオーバーライドできます。クラス名には、データ圧縮ロジックを含むカスタム・クラスの名前を指定します。

データ圧縮ロジック

圧縮可能な列のデータは、GZip アルゴリズムを使用して圧縮されます。SCICustomDataCompressor インターフェースを実装することによって、このデフォルトのデータ圧縮ロジックをオーバーライドし、カスタム圧縮ロジックを指定することができます。

SCICustomDataCompressor インターフェースを実装するカスタム・クラスの名前を、customer_overrides.properties ファイル内の yfs.db.compression.class=*class name* プロパティーに入力する必要があります。

非固有インデックスを標準テーブルに追加するためのガイドライン

非固有インデックスを追加する場合、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales の規則 (すなわち *tablename*_i<1+n>) とは異なる命名規則を使用します。独自の命名規則を使用することで、アップグレード時にインデックスが誤って削除されるのを防ぎます。また、以下の点について考慮することをお勧めします。

- Y で始まらない接頭辞を追加する。
- 識別を容易にするために、非固有インデックスに EXTN_ という接頭辞を追加する。
- テーブルに対して固有インデックスは許可されない。
- インデックスの列名が必ず有効である。
- インデックス名が 18 文字を超えない。

外部キー要素を標準テーブルに追加するためのガイドライン

現在、拡張されたテーブルの列における外部キー関係は、YFS_PERSON_INFO テーブルのみに制限されています。外部キー要素を公開する場合、以下の検証が実行されます。

- 親テーブル名が、必ず YFS_PERSON_INFO となっている。
- 親列名が、必ず YFS_PERSON_INFO テーブルの主要キーとなっている。

テキスト検索インデックス要素を標準テーブルに追加するためのガイドライン

テキスト検索インデックス要素を標準テーブルに追加する場合、以下の考慮事項について留意してください。

- テキスト検索要素の追加は、非トランザクション・テーブルに対してのみ可能です。
- テキスト検索インデックスに対して行われる検索は、大/小文字を区別しません。
- エンティティーごとに複数のテキスト検索インデックスを定義できます。
- テキスト・インデックスに対して複数の列を定義することはできません。
- IBM は、Oracle 上で CTXCAT および CONTEXT の各テキスト検索インデックス・タイプをサポートしています。
- 拡張エンティティーとテキスト検索インデックスの両方を、同じ拡張 XML ファイル内に定義することをお勧めします。

デッドロックを回避するためのコーディング・ガイドライン

デッドロックとは、2 つのプロセスが、互いに他のプロセスによってリソースが解放されるのを待っている、データベースの固有の状態のことです。例えば、1 番目のクライアント・アプリケーションがテーブルをロックしており、2 番目のクライアント・アプリケーションによってロックされている別のテーブルのロックの取得を試みているとします。このとき、2 番目のアプリケーションが、1 番目のアプリ

ケーションによってロックされているテーブルのロックを取得しようとした場合に、デッドロックが発生する可能性があります。

デッドロック問題を回避するには、アクセスする情報を特定の順序でソートしてからロックを取得することをお勧めします。これには、単一トランザクションの境界内で、複数の在庫アイテムのロックを取得する必要があるような状況が該当します。ただし、API を呼び出して、トランザクションのコミットにつき 1 つのアイテムを処理する場合は、ソートする必要はありません。

DB2 データベース内のコミットされていないデータの読み取り。

DB2 では、テーブルからレコードを選択するときに、レコードに対する読み取りロックが取得されます。選択されているレコードが更新されているがコミットされていない場合、スレッドはその変更がコミットされるまで待機します。あるいは、未コミット読み取り (UR) を使用してレコードを読み取ることもできます。その場合ユーザーには、最後に更新された値が提供されます。

リスト API の入力 XML 内の ReadUncommitted 属性を Y に設定して有効化することにより、コミットされていないデータを任意のリスト API から読み取ることができます。これを行うには、個々の JSP をカスタマイズして、ReadUncommitted 属性を hidden 属性として渡す必要があります。次に例を示します。

```
<input type="hidden" name="xml:/Order/@ReadUnCommitted" value="Y"/>
```

この結果、DB2 データベースでロックが発生する状況は回避されます。DB2 では、ロックすることがデフォルトになっています。

このフラグを渡すことは、必須ではありません。しかし、このフラグを Y に設定すると、システムはコミットされていないデータを強制的に読み取ります。例えば、トランザクション T1 がテーブル TAB-1 を更新したが、トランザクションのデータはコミットされていないとします。ReadUncommitted フラグが Y に設定されていれば、テーブル TAB-1 内のコミットされていないデータを、他のトランザクションが読み取ることができます。

このフラグを設定する前に、同時に実行するトランザクションを評価して、デッドロックが発生する状況が存在するかどうかを判断してください。そのような状況が発生しないのであれば、このフラグはデフォルト設定のままにしてください。

この動作は、Oracle とは異なります。そのため、DB2 でカスタム・コードを記述する場合、この動作を理解してロックの拡大を防ぐ必要があります。

第 3 章 データベース表の拡張

標準テーブルへの列の追加

このタスクについて

エンティティー・データベースの拡張 XML ファイルを変更することによってテーブルに列を追加してから、アプリケーション・データベースと JAR ファイルを再ビルドします。アプリケーションが再ビルドされると、API は追加されたそれらの列を認識し、データの格納と検索を行う際に使用するようになります。

列を標準テーブルに追加するには、以下の手順に従います。

手順

1. *install_dir*/repository/entity/extensions/Extensions.xml.sample ファイルを *install_dir*/extensions/global/entities/*your_filename*.xml ファイルとしてコピーするか、既存の拡張 XML ファイルを変更します。
2. 以下の例で示すように、*your_filename*.xml ファイルを編集して、拡張するテーブルごとに新しいエンティティー・タグを追加します。タグが既に存在する場合は、既存のタグを使用します。 XML 属性の説明については、例の下に示す表を参照してください。

```
<!-- element exposed to create a column -->
<DBSchema>
  <Entities>
    <Entity TableName="REQUIRED">
    <Attributes>
      <Attribute ColumnName="REQUIRED" DataType="" DecimalDigits=""
       CompressionSupported="false"
       UseCompression="false" QueryAllowed="false" DefaultValue="" Description=""
       Nullable="false" Size="1" Type="REQUIRED" XMLName="" XMLGroup=""
       SqlServerDataType="" />
    </Attributes>
    </Entity>
  </Entities>
</DBSchema>
```

属性	説明
ColumnName	必須。このテーブルに追加される列の名前。 ColumnName は、EXTN_ で始まる必要があります。
DataType	オプション。有効な値は、*install_dir*/repository/datatypes/datatypes.xml ファイルから得られます。
DecimalDigits	オプション。小数点以下で必要な精度の桁数。数値フィールドの場合にのみ必要です。

属性	説明
CompressionSupported	オプション。この列でデータ圧縮がサポートされるかどうかを指定するために使用される属性。有効な値は、`true` または `false` です。 true を指定した場合、圧縮のサポートが有効になります。 **注:** 列に挿入されるデータの CompressionSupported 属性と UseCompression 属性の両方が true に設定されている場合、この属性を false に設定しないでください。false に設定すると、すべての圧縮済みデータが解凍されないで取り出されることになります。
UseCompression	オプション。この列のデータを圧縮する場合に使用される属性。有効な値は、`true` または `false` です。 true を指定した場合、データが圧縮されます。 **注:** CompressionSupported 属性を true に設定した場合にのみ、この属性の値を true に設定する必要があります。
QueryAllowed	オプション。リスト API のクエリーで、圧縮可能な列を使用可能にするための属性。有効な値は、`true` または `false` です。 true を指定した場合、この列は、リスト API のクエリーで使用可能になります。 **注:** CompressionSupported 属性の値を true に設定した場合、この属性の値も true に設定する必要があります。
DefaultValue	必須。データベース内のデフォルト句に、そのまま使用されます。
説明	オプション。列の使用法の説明。
Nullable	オプション。フィールド値に NULL が可能であることを指定するために使用される属性。デフォルトは false です。主要キー属性およびエンティティー・リレーションシップ以外のすべての列に対して、`Nullable=true` を指定することができます。
サイズ (Size)	データベース列のサイズ。
型	必須。データベース列のデータ・タイプ。この属性は、生成される Java クラス内の属性のタイプおよび XML 内の属性の形式も決定します。有効な値は、 CHAR、VARCHAR2、NUMBER、DATE、および TIMESTAMP です。 Microsoft SQL Server® を使用していて、データベース内で TEXT のデータ・タイプを指定する場合、SqlServerDataType 属性を使用して TEXT として属性値を指定することも必要です。 **注:** DATE を指定した場合、カレンダー日付のみが格納されます。TIMESTAMP を指定した場合、カレンダー日付と時刻が格納されます。

属性	説明
XMLName	属性の XML 名。属性名と異なる場合に指定します。 ベースになる拡張と競合しない名前を選択します。接頭辞として Extn を使用することをお勧めします。また、XMLName への変換規則には、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales ベース製品で使用されているものと同じ規則を使用することを、強くお勧めします。つまり、列名内のアンダースコアーに続く各文字を大文字にし、それ以外を小文字にします。次に、アンダースコアーを削除します。したがって、Extn_Item_Id は、`ExtnItemId` となります。
XMLGroup	もしあれば、属性が含まれる子タグを指定します。XML に属性がなければ、NOT_SHOWN ストリングを使用します。 XMLGroup は、`Extn` である必要があります。したがって、拡張列のデータは、API の XML 出力内の別の要素に含まれます。
VirtualDefaultValue	オプション。この属性は、NULL 列に適用できます。データベースから NULL が返された場合、それは仮想デフォルト値としてメモリーに格納されます。
SqlServerDataType	オプション。Microsoft SQL Server データベースのみに関連します。行サイズが大きすぎるという警告が表示された場合、1 つ以上のより大きな列を「TEXT」として指定します。 TEXT タイプの列は、テーブルの最大行サイズ計算には含まれません。
ForceUpperCase	オプション。テキスト列で「大/小文字を区別しない」検索が必要な場合、このフィールドを「true」に設定する必要があります。 これを「true」に設定すると、システムはこのフィールドに入力されたデータを大文字に変換します。

3. テーブルに追加する列ごとに、Attribute タグを新規作成します。
4. スクリプト生成用のデータベース検証ツール dbverify を使用して、データベースに列を追加します。

   注: Microsoft SQL Server では、すべての拡張列の長さの合計は、900 バイトを超えることはできません。行サイズが最大長を超えたという警告が Microsoft SQL Server によってスローされた場合、1 つ以上の列のデータ・タイプを TEXT に変更します。次に、上記の表に記載されているように、SqlServerDataType 属性に対して TEXT を指定します。
5. 「API テンプレートの拡張について」に記載された手順に従って、対応する API テンプレートを拡張します。
6. 拡張をビルドして、デプロイします。

タスクの結果

固有タグ ID または固有記述子を追加するために列を拡張する特殊ケースについては、 に記載しています。.

標準テーブルに列を追加する場合に複数の拡張が必要な機能

標準テーブルの拡張に加えて、指定されたガイドラインに従って、以下の機能を拡張することができます。

- 分類の継承 - YFS_ITEM テーブルに列を追加して属性を拡張し、それらの属性を分類の継承で使用可能にする場合、エンティティー・タグの複製を YFS_CLASS_ITEM_ATTR テーブルの XML に追加する必要があります。さらに、Nullable を `true` に設定し、それらの属性にデフォルト値を渡さないようにする必要があります。分類レベルでのアイテム属性の定義について詳しくは、『*カタログ管理構成ガイド (Catalog Management Configuration Guide)*』を参照してください。
- アイテムの資格 - YFS_CUSTOMER テーブルに列を追加して属性を拡張し、それらの属性を資格ルールの割り当てで使用可能にする場合、エンティティー・タグの複製を YFS_ENTITLE_RULE_ASSIGNMENT テーブルの XML に追加する必要があります。さらに、Nullable を `true` に設定し、それらの属性にデフォルト値を渡さないようにする必要があります。資格ルールの割り当てについて詳しくは、『*Business Center: アイテム管理ガイド*』を参照してください。
- 価格リストの割り当て - YFS_CUSTOMER テーブルに列を追加して属性を拡張し、それらの属性を価格リストの割り当てで使用可能にする場合、エンティティー・タグの複製を YFS_PRICELIST_ASSIGNMENT テーブルの XML に追加する必要があります。さらに、Nullable を `true` に設定し、それらの属性にデフォルト値を渡さないようにする必要があります。価格リストの割り当てについて詳しくは、『*Business Center: 価格設定管理ガイド*』を参照してください。
- 価格設定ルールの割り当て - YFS_CUSTOMER テーブルに列を追加して属性を拡張し、それらの属性を価格設定ルールの割り当てで使用可能にする場合、エンティティー・タグの複製を YFS_PRICING_RULE_ASSIGNMENT テーブルの XML に追加する必要があります。さらに、Nullable を `true` に設定し、それらの属性にデフォルト値を渡さないようにする必要があります。価格設定ルールの割り当てについて詳しくは、『*Business Center: 価格設定管理ガイド*』を参照してください。

標準列のサイズの拡大

このタスクについて

yfsdatatype XML ファイルを拡張し、データベースと JAR ファイルを再ビルドすることによって、テーブル列のサイズを増やすことができます。アプリケーションが再ビルドされると、API はサイズ変更されたそれらの列を認識し、データの格納と検索を行う際に使用するようになります。サイズ変更は、VARCHAR2 データ・タイプの列に対してのみ行えます。列のサイズは増やせますが、減らすことはできません。

テーブルの標準列のサイズを増やすには、以下の手順を実行します。

手順

1. *install_dir*/extensions/global/etc/datatypes.xml ファイルがまだ存在しない場合、新規作成します。

   /global/etc/ ディレクトリー構造が存在しない場合、必要なディレクトリー構造を作成します。
2. 既存の列定義を、*install_dir*/repository/datatypes/datatypes.xml ファイルから、新規作成された *install_dir*/extensions/global/etc/datatypes.xml ファイルにコピーし、次に、必要な変更を適用します。
3. 以下の例で示すように、サイズ属性を所望のサイズに増やすことによって、データ・タイプを拡張します。このファイルには、対象となる変更のみを記述する必要があります。変更が不要なデータ・タイプは、記述しないでください。

   ```
   <DataTypes>
   <DataType Name="TagNumber" Size="140"/>
   </DataTypes>
   ```
4. データベース内の列を手動で変更します。スクリプト生成用のデータベース検証ツール dbverify を使用して、データベースに列を追加することもできます。

   注: Microsoft SQL Server では、すべての拡張列の長さの合計は、900 バイトを超えることはできません。行サイズが最大長を超えたという警告が Microsoft SQL Server によってスローされた場合、1 つ以上の列のデータ・タイプを TEXT に変更します。「標準テーブルへの列の追加」の属性の表に記載されているように、SqlServerDataType 属性に対して TEXT を指定します
5. データベースの拡張をビルドして、デプロイします。

カタログ検索の拡張

このタスクについて

カタログ検索インデックス・ファイル内の情報を更新することによってカタログ検索を拡張し、検索機能を拡大することができます。インデックス・ファイルを更新するには、対応する拡張 XML 構成ファイル変更します。カタログ検索の以下のコンポーネントは、拡張可能です。

- 検索システム構成
- ロケールおよび対応するアナライザー
- クエリー・パーサー
- 属性

カタログ検索を拡張するには、以下の手順を実行します。

手順

1. *install_dir*/repository/xapi/template/merged/resource/extn/ExtnCatalogSearchConfigProperties.xml.sample ファイルを *install_dir*/extensions/global/template/resource/extn/ExtnCatalogSearchConfigProperties.xml ファイルとしてコピーするか、既存の拡張 XML ファイルを変更します。

   ExtnCatalogSearchConfigProperties.xml ファイルの例を以下に示します。

```
<SearchConfigurations>
  <SearchSystemConfigurations>
    <MergeFactor Value="2"/>
      <MaxMergeDocs Value="2147483647"/>
   </SearchSystemConfigurations>
    <IndexSets>
    <IndexSet Name="CatalogIndex">
      <Locales>
       <Locale LocaleCode="en_US"
        SynonymFile="/properties/EnglishSynonym.properties"/>
        <Locale LocaleCode="fr_FR"
          QueryParserClass="package.FrenchQueryParser.class"
          AnalyzerClass="package.FrenchAnalyzer.class"
          SynonymFile="/properties/FrenchSynonym.properties"/>
       </Locales>
       <Entities>
          <Entity Name="Item">
             <Attributes>
               <Attribute XMLName="MyExtendedDescription"
                 DefaultWeight="1.0" Index="ANALYZED"
                 Store="Y" UseSynonyms="N" Searchable="Y" Sortable="N"/>
               <Attribute IndexFieldName="CustomerItemDescription"
                 DefaultWeight="1.0" Index="ANALYZED" Store="Y" UseSynonyms="N"
                 Searchable="Y" Sortable="N" GetExternalValue="Y"/>
             </Attributes>
              <Entity Name="CategoryItem" RelationshipName="Category_ItemList">
              <Entity Name="Category" RelationshipName="Category">
             <Attributes>
               <Attribute XMLName="MyCategoryExtendedDescription"
                 DefaultWeight="1.0" Index="ANALYZED" Store="Y"
                 UseSynonyms="N" Searchable="Y" Sortable="N"/>
             </Attributes>
            </Entity>
            </Entity>
            <Entity Name="Asset" RelationshipName="AssetList">
             <Attributes>
               <Attribute Type="MyManual" DefaultWeight="1.0" Index="ANALYZED"
                Store="N" UseSynonyms="N" Searchable="Y" Sortable="N"/>
              </Attributes>
             </Entity>
            </Entity>
       </Entities>
     </IndexSet>
 </IndexSets>
</SearchConfigurations>
```

2. インデックス・ファイルのシステム構成情報を変更するには、ExtnCatalogSearchConfigProperties.xml の SearchSystemConfigurations セクション内の要素を変更します。ExtnCatalogSearchConfigProperties.xml 内の SearchSystemConfiguration 要素について、以下の表で説明します。

要素	説明
MergeFactor Value	Apache Lucene オープン・ソース検索エンジンがメモリーに格納する、ドキュメントの数を指定します。それらのドキュメントは、その後 1 つのセグメントとしてディスクに書き込まれます。この値の設定について詳しくは、http://lucene.apache.org を参照してください。
MaxMergeDocs Value	1 つのセグメントに含めることができるドキュメントの数を指定します。この値の設定について詳しくは、http://lucene.apache.org を参照してください。

3. インデックス・ファイルのロケール情報を追加または変更するには、ExtnCatalogSearchConfigProperties.xml の Locales セクション内の要素を追加または変更します。上記で示した ExtnCatalogSearchConfigProperties.xml ファイルの例では、同義語ファイルが米国英語のロケールに追加されており、フランス語のロケールも追加されています。ExtnCatalogSearchConfigProperties.xml 内の Locale 要素について、以下の表で説明します。

要素	説明
LocaleCode	インデックス・ファイルのロケール・コードを指定します。ロケールを追加する場合、対応するアナライザーを指定していることを確認してください。デフォルト・ロケールは、米国英語です。
QueryParserClass	クエリー・パーサーへのパスを指定します。デフォルトでは、アプリケーションは Lucene クエリー・パーサーを提供します。
AnalyzerClass	アナライザーへのパスを指定します。アプリケーションは、対応するデフォルト・ロケールのアナライザーを提供します。
SynonymFile	対応するロケールの同義語ファイルへのパスを指定します。同義語ファイルを使用して、キーワード検索用の関連用語を構成します。

4. インデックス・ファイルのアイテム情報を追加するために、その属性を ExtnCatalogSearchConfigProperties.xml の Item セクションに追加します。上記で示した ExtnCatalogSearchConfigProperties.xml ファイルの例では、MyExtendedDescription 属性と CustomerItemDescription 属性が追加されています。ExtnCatalogSearchConfigProperties.xml 内の Item 属性要素について、以下の表で説明します。

   注: 属性をカタログ検索インデックスの拡張 XML 構成ファイルに追加することはできますが、属性およびエンティティーを変更することはできません。デフォルトでサポートされる属性のリストについては、*install_dir*/repository/xapi/template/merged/resource/CatalogSearchConfigProperties.xml を参照してください。
5. インデックス・ファイルに関連するカタログ情報を追加するために、その属性を ExtnCatalogSearchConfigProperties.xml のカテゴリー・セクションに追加します。上記で示した ExtnCatalogSearchConfigProperties.xml ファイルの例では、MyCategoryExtendedDescription カテゴリー属性が追加されています。ExtnCatalogSearchConfigProperties.xml 内のカテゴリー属性要素について、以下の表で説明します。
6. インデックス・ファイルのアセット情報を追加するために、その属性を ExtnCatalogSearchConfigProperties.xml の Asset セクションに追加します。上記で示した ExtnCatalogSearchConfigProperties.xml ファイルの例では、MyManual アセット・タイプのアセットが追加されています。CatalogSearchConfigProperties.xml の属性について、以下の表で説明します。

   注: アプリケーションは、YCMParseAssetUE 外部プログラムを呼び出して、対応するアセット・タイプのアセットの内容を構文解析します。

属性	説明
XMLName	アイテム・エンティティーのアイテム属性およびカテゴリー属性の場合、フィールドの XML 属性名を指定します。

属性	説明
IndexFieldName	インデックスに格納される属性のフィールド名を指定します。IndexFieldName の値は、構成ファイル内で一意的である必要があります。XML 構成ファイル内で IndexFieldName が構成されていない場合、システムは、エンティティー名.*XMLName* の形式に基づいてこの値を生成します。
型	アセット属性の場合、データベース内のアセット・タイプを指定します。
DefaultWeight	用語に付与するウェイトを指定します。デフォルト値は 1 です。
Index	フィールド値を格納するために、以下のいずれかのオプションを指定します。 analyzed - 値を検索可能なセグメントとして格納します。例えば、ユーザーが「Desktop Computer」を検索して、フィールドのインデックス・パラメーターを解析する場合、検索結果には、「Desktop Computer」、「Desktop」、および「Computer」の各用語を持つアイテムが含まれます。 non_analyzed - 表示されているとおりの値をデータベースに格納します。この値を取得するには、完全一致検索が必要です。上記の例では、non_analyzed フィールドに対して「Desktop Computer」を検索した場合、「Desktop Computer」の用語を持つアイテムのみの検索結果が得られます。
Store	フィールド値を格納するために、以下のいずれかのオプションを指定します。 Y - 属性値をインデックスに格納します。そのため、属性値を検索結果として返すことができます。GetExternalValue 属性を Y に設定した場合、Store は Y として扱われます。 N - 属性値をインデックスに格納せず、専用の形式でデータとしてのみ保持し、検索中にのみ使用します。
UseSynonyms	検索に同義語を含めるために、以下のいずれかのオプションを指定します。 Y - 検索クエリーに同義語を含めることを指定します。 N - 検索クエリーに同義語を含めないことを指定します。

属性	説明
Searchable	フィールドに対して、以下のいずれかの検索オプションを指定します。 Y - 検索クエリーを作成するときに、このフィールドが選択可能な検索条件に含まれることを指定します。 N - 検索クエリーを作成するときに、このフィールドが選択可能な検索条件に含まれないことを指定します。 例えば、デフォルト XML ファイル内の Is_Superseded フィールドの Searchable パラメーターを N に設定し、Store パラメーターを Y に設定したとします。その場合ユーザーは、クエリーで Is_Superseded フィールドを検索することはできません。しかし、廃止されて取り替えられたアイテムに対するクエリーによって、取り替え後のアイテムを取得することができます。
Sortable	フィールドに対して、以下のいずれかのソート・オプションを指定します。 Y - このフィールドによって検索結果をソートすることを指定します。 N - このアイテムによって検索結果をソートしないことを指定します。
GetExternalValue	フィールドに対して、以下のいずれかのオプションを指定します。 Y - この属性の値を、外部プログラムを介して外部ソースから取得することを指定します。 N - 外部プログラムを呼び出さないことを指定します。

クエリー用の列の事前定義とグループ化

エンティティー定義に QueryGroup パラメーターを追加することによって、クエリー (選択またはリスト) 検索で使用される列を選択してグループ化できます。各 QueryGroup には、クエリーで使用される列が含まれます。各 QueryGroup は、固有名によって指定します。その下に、各列を指定する必要があります。

Entity クラスで指定されるグローバル・マップの QueryGroupMap には、固有クエリー・グループ名およびクエリー・グループの列に対応する属性が含まれています。このマップは、エンティティー・リポジトリーがロードされるときに取り込まれます。

注: QueryGroup 名を空欄にすることはできません。クエリー・グループ名をマップに追加する場合、その名前が既にマップに存在するかどうかを確認してください。

各エンティティーは、そのようなクエリー・グループを複数持つことができます。それらは、以下のように QueryGroups パラメーターの下で定義されます。

```
<Entity>
  <QueryGroups>
    <QueryGroup Name="PasswordQueryGroup">
```

```
        <Column Name="PASSWORD"/>
        <Column Name="IS_PASSWORD_ENCRYPTED"/>
        <Column Name="SALT"/>
      </QueryGroup>
      <QueryGroup Name="BusinessQueryGroup">
        <Column Name="BUSINESS_KEY"/>
        <Column Name="CONTACTADDRESS_KEY"/>
        <ColumnName="BILLINGADDRESS_KEY"/>
       </QueryGroup>
   </QueryGroups>
</Entity>
```

クエリー・グループの下には、任意の数の列を指定できます。ただし、1 つのクエリー・グループに対して、列を繰り返すことはできません。

注: QueryGroup 内の列を、別の QueryGroups に含めることができます。クエリー・グループには、少なくとも 1 つの有効な列が含まれる必要があります。また、クエリー・グループに対して定義される列は、エンティティー内に存在している必要があります。仮想列をクエリー・グループに追加することはできません。

特定の QueryGroup に含まれるすべての列のリストを取得するには、getColumnSetForQueryGroup メソッドを以下の形式で使用します。

```
public Set getColumnSetForQueryGroup(String queryGroupName)
```

以下のメソッドを特定のパラメーターとともに使用して、クエリー・グループ内の列のリストを取得することもできます。

- selectWithWhereForQueryGroup
- listWithWhereForQueryGroup

例:

```
public YFS_User selectWithWhereForQueryGroup(YFCDBContext ctx,
String aWhereClause, String queryGroupName) throws YFCDBException{
Set columns = getColumnSet(queryGroupName);
return selectWithWhere(ctx, aWhereClause, columns);
}
```

標準テーブルへの固有タグ ID と固有タグ記述子の追加

Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales のデフォルトのタグ ID は、バッチ番号、改訂番号、およびロット番号です。固有タグ ID と固有タグ記述子を定義するために、アプリケーション・データベースの拡張が必要な場合があります。

追加する固有タグ ID または固有タグ記述子のデータ・タイプは、CHAR または VARCHAR とすることをお勧めします。

注: タグ属性を拡張する場合、常にコンソールも拡張する必要があります。これは、API のテンプレートに、それらの拡張されたタグ属性が含まれないためです。

例えば、金属業界で働いているのであれば、Mill 属性と Garde 属性の両方を含む Steel というカスタム・タグ ID を使用したい場合があります。それらはデフォルトでは提供されないため、以下に示す一連のテーブルを拡張して、各テーブルに Steel タグ ID の列を含める必要があります。

固有タグ ID を追加する場合に拡張するテーブル

固有タグ ID をアプリケーション・データベースに追加する場合、常に以下の各テーブルを拡張する必要があります。

- YFS_COUNT_RESULT_TAG
- YFS_COUNT_TAG
- YFS_INVENTORY_AUDIT
- YFS_INVENTORY_TAG
- YFS_ITEM_TAG - このテーブルに使用するデータ・タイプは CHAR(2) です。
- YFS_MOVE_REQUEST_LINE_TAG
- YFS_ORDER_KIT_LINE_SCHEDULE
- YFS_ORDER_KIT_LINE_SCHEDULE_H
- YFS_ORDER_LINE_REQ_TAG
- YFS_ORDER_LINE_REQ_TAG_H
- YFS_ORDER_LINE_SCHEDULE
- YFS_ORDER_LINE_SCHEDULE_H
- YFS_ORDER_LINE_RESERVATION
- YFS_RECEIPT_LINE
- YFS_RECEIPT_LINE_H
- YFS_SHIPMENT_LINE_REQ_TAG
- YFS_SHIPMENT_LINE_REQ_TAG_H
- YFS_SHIPMENT_TAG_SERIAL
- YFS_SHIPMENT_TAG_SERIAL_H
- YFS_WORK_ORDER_COMP_TAG
- YFS_WORK_ORDER_COMP_TAG_H
- YFS_WORK_ORDER_TAG
- YFS_WORK_ORDER_TAG_H

固有タグ記述子を追加する場合に拡張するテーブル

固有タグ記述子をアプリケーション・データベースに追加する場合、常に以下の各テーブルを拡張する必要があります。

- YFS_COUNT_RESULT_TAG
- YFS_COUNT_TAG
- YFS_INVENTORY_TAG
- YFS_ITEM_TAG - このテーブルに使用するデータ・タイプは CHAR(2) です
- YFS_ORDER_LINE_REQ_TAG
- YFS_ORDER_LINE_REQ_TAG_H
- YFS_RECEIPT_LINE
- YFS_RECEIPT_LINE_H
- YFS_SHIPMENT_LINE_REQ_TAG
- YFS_SHIPMENT_LINE_REQ_TAG_H

- YFS_SHIPMENT_TAG_SERIAL
- YFS_SHIPMENT_TAG_SERIAL_H
- YFS_WORK_ORDER_COMP_TAG
- YFS_WORK_ORDER_TAG
- YFS_WORK_ORDER_TAG_H

標準テーブルへの非固有インデックスの追加

このタスクについて

非固有インデックスをエンティティーに追加することができます。インデックス要素を Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales データベースの標準テーブルの拡張 XML に追加することによって、そのテーブルにインデックスを追加します。

非固有インデックスを標準テーブルに追加するには、以下の手順を実行します。

手順

1. *install_dir*/installed_data/repository/entity/extensions/Extensions.xml.sample ファイルを *install_dir*/extensions/global/entities/*your_filename*.xml ファイルとしてコピーするか、既存の拡張 XML ファイルを変更します。
2. 以下の例で示すように、*your_filename*.xml ファイルを編集して、拡張するテーブルごとに非固有インデックスを追加します。XML 属性の説明については、例の下に示す表を参照してください。

```
<!-- element exposed to create index -->
<DBSchema>
<Entities>
  <Entity TableName="REQUIRED">
  .
  .
    <Indices>
      <Index Name="REQUIRED" AllowCompressedColumns="false">
         <Column Name="REQUIRED" />
         .
         .
      </Index>
        .
        .
    </Indices>
     .
     .
   </Entity>
</Entities>
</DBSchema>
```

属性	説明
Entity	
TableName	必須。インデックスを追加するテーブルの名前。例えば、YFS_ITEM など。
Entity/Index	
Name	必須。カスタム・インデックスの名前。この名前は、接頭辞 EXTN_ で始まる必要があります。
AllowCompressedColumns	オプション。これを true に設定すると、CompressionSupported 属性が true に設定された列をインデックスに含めることができます。

属性	説明
Entity/Index/Column	
Name	必須。インデックスを追加する列の名前。インデックスを追加する列ごとに、列名を新規作成します。

3. 列に追加するインデックスごとに、Index タグを新規作成します。
4. 「API テンプレートの拡張」の手順に従って、非固有インデックスを含むように、対応する API テンプレートを拡張します。
5. 拡張をビルドして、デプロイします。

標準テーブルへの外部キー要素の追加

このタスクについて

外部キー関係は、任意のアプリケーション・データベース・テーブル内の拡張列と YFS_PERSON_INFO テーブルとの間の関係です。外部キー要素を作成して、拡張列と YFS_PERSON_INFO テーブルとの間の関係を確立することができます。

注: 現在、YFS_PERSON_INFO は、アプリケーション・データベース内の外部キー拡張との関係をサポートする唯一のテーブルです。

外部キー要素を標準テーブルに追加するには、以下の手順を実行します。

手順

1. *install_dir*/repository/entity/extensions/Extensions.xml.sample ファイルを *install_dir*/extensions/global/entities/*your_filename*.xml ファイルとしてコピーするか、既存の拡張 XML ファイルを変更します。
2. 以下の例で示すように、*your_filename*.xml ファイルを編集して、拡張するテーブルごとに外部キー要素を追加します。XML 属性の説明については、例の下に示す表を参照してください。

```
<!-- element exposed to create foreign key relationship -->
<DBSchema>
 <Entities>
    <Entity TableName="REQUIRED">
     .
     .
      <!-- element exposed to create relationship with PERSON_INFO table -->
         <ForeignKeys>
           <ForeignKey ParentTableName="YFS_PERSON_INFO"
             XMLName="YFSName1" >
               <Attribute ColumnName="REQUIRED"
                ParentColumnName="PERSON_INFO_KEY" />
             </ForeignKey>
           <ForeignKey ParentTableName="YFS_PERSON_INFO"
             XMLName="YFSName2" >
               <Attribute ColumnName="REQUIRED"
                ParentColumnName="PERSON_INFO_KEY" />
           </ForeignKey>
           .
           .
         </ForeignKeys>
       .
       .
    </Entity>
   </Entities>
</DBSchema>
```

属性	説明
Entity	
TableName	必須。外部キー要素を追加するテーブルの名前。例えば、YFS_ITEM 。
Entity/ForeignKeys/ForeignKey	
ParentTableName	この外部キー要素の親テーブルの名前。 **注:** この値は、YFS_PERSON_INFO にする必要があります。これは、現在外部キー関係をサポートする唯一のテーブルです。
XMLName	要素名の XML 表記を指定することができます。この名前は、親エンティティーの接頭辞で始まる必要があります。例えば、ParentTableName の接頭辞が YFS である場合、XMlName は YFS で始まる必要があります。 デフォルトでは、親テーブル名が仮定されます。
Entity/ForeignKeys/ForeignKey/Attribute	
ColumnName	Entity の拡張列名を指定します。
ParentColumnName	外部キー要素関係を持つ YFS_PERSON_INFO の列名。

3. 追加する外部キー関係ごとに、ForeignKey タグを新規作成します。
4. 複数の外部キー要素を、同じ親テーブルに関連付けることができます。
5. 「API テンプレートの拡張」の手順に従って、外部キー要素を含むように、対応する API テンプレートを拡張します。
6. 拡張をビルドして、デプロイします。

標準テーブルへのテキスト検索インデックスの追加

このタスクについて

テキスト検索インデックスをエンティティーに追加することができます。TSIndex 要素をアプリケーション・データベースの標準テーブルの拡張 XML に追加することによって、テキスト検索インデックスをそのテーブルに追加します。

テキスト検索インデックスを標準テーブルに追加するには、以下の手順を実行します。

手順

1. *install_dir*/repository/entity/extensions/Extensions.xml.sample ファイルを *install_dir*/extensions/global/entities/*your_filename*.xml ファイルとしてコピーするか、既存の拡張 XML ファイルを変更します。
2. 以下の例で示すように、*your_filename*.xml ファイルを編集して、拡張するテーブルごとにテキスト検索インデックスを追加します。XML 属性の説明については、例の下に示す表を参照してください。

```
<!-- element exposed to create index -->
<DBSchema>
  <Entities>
    <Entity TableName="REQUIRED">
      .
      .
```

```
        <TSIndices>
          <TSIndex Name="REQUIRED" >
           <Column Name="USERNAME" />
          </TSIndex>
         .
         .
        </TSIndices>
       .
       .
      </Entity>
    </Entities>
  </DBSchema>
```

属性	説明
Entity	
TableName	必須。テキスト検索インデックスを追加するテーブルの名前。例えば、YFS_USER など。
Entity/TSIndex	
Name	必須。テキスト検索インデックスの名前。例えば、YFS_TS_USER_Name など。 **注:** この値は、18 文字を超えることはできません。
Entity/TSIndex/Column	
Name	必須。テキスト検索インデックスを追加する列の名前。テキスト・インデックスに対して複数の列を定義することはできません。

3. 列に追加するテキスト検索インデックスごとに、TSIndex タグを新規作成します。
4. 拡張をビルドして、デプロイします。

ケース・インセンシティブ検索の使用可能化

このタスクについて

必要なエンティティーの列に対してケース・インセンシティブ検索を使用可能にすることによって、アプリケーションの検索操作で大/小文字を区別しないようにすることができます。これは、対象のエンティティー XML ファイルに CaseInsensitiveSearch 属性を追加することによって行います。

ケース・インセンシティブ検索を使用可能にするには、以下の手順を実行します。

手順

1. 対象のエンティティー XML を編集して、CaseInsensitiveSearch=Y という属性を含めます。

   シャドー列属性 ShadowColumnName をエンティティー XML に含め、シャドー列の名前を指定します。シャドー列名を空欄のままにすると、システムは自動的に名前を生成します。ただし、シャドー列にインデックスを作成する場合は、シャドー列名を指定する必要があります。

   これによって、ケース・インセンシティブのマークが付けられた元の列にリンクするシャドー列が生成されます。

2. 実稼働環境でエージェントが実行中の場合、customer_overrides.properties ファイル内の CaseInsensitiveSearch.Mode プロパティーのエンティティーまたは列に対して、構成モードを指定します。CaseInsensitiveSearch.Mode プロパティーの値を、MIXED または DISABLED に設定します。

   注: デフォルトでは、ケース・インセンシティブ検索が使用可能になっています。
3. 「大/小文字を区別しないデータ・ローダー」エージェントの実行対象となるエンティティーのキャッシュを無効にします。

   注: キャッシュを無効化するとともに、対象のエンティティーの監査もオフにすることをお勧めします。
4. 「大/小文字を区別しないデータ・ローダー」エージェントを実行して、データをシャドー列に取り込みます。

   注: ステップ 2 をスキップした場合は、このステップも同様にスキップします。
5. customer_overrides.properties ファイル内の CaseInsensitiveSearch.Mode プロパティーの値を ENABLED に設定します。

   注: ステップ 3 で監査を無効化した場合、ここで監査を有効化します。
6. ステップ 3 で「大/小文字を区別しないデータ・ローダー」エージェントを実行したエンティティーまたは列のキャッシュを有効化します。

   以下の API は、システム定義列に対するケース・インセンシティブ検索をサポートしています。

   - getOrderList
   - getItemListForOrdering
   - getExceptionListForOrder
   - getCustomerList

   拡張されたカスタム列の場合、それらの列に対して通常の検索を実行すると、ケース・インセンシティブ検索が実行されます。

   注: この機能は、アプリケーションによって生成される XAPI を使用する検索に対してのみ適用できます。カスタム・クエリーを使用する検索では、大/小文字が区別されます。

   注: ケース・インセンシティブ検索は、テキスト・データ・タイプ (CHAR/VARCHAR 列) の場合にのみ使用可能にできます。

エンティティー XML の変更

手順

1. 以下に示すように、対象となるエンティティー XML を編集して、必要な列の <Attributes> タグの下に CaseInsensitiveSearch 属性を含めます。

```
<Entity Description="This table stores all the exceptions raised
   by the system."
   EntityType="TRANSACTION" Extensible="Y" Module="ycp"
   Name="Inbox" Prefix="YFS_" TableName="YFS_INBOX"  XMLName="Inbox">
```

```
<Attributes>
 ........
  <Attribute ColumnName="EXCEPTION_TYPE"
   DataType="Text-40" DefaultValue="' ' "
   Description="The type of exception."
   Name="Exception_Type" Nullable="false"
   XMLName="ExceptionType" CaseInsensitiveSearch ="Y"
   ShadowColumnName="ExceptionType_LC"/>
   ......
 <Indices>
    <Index Name="EXCEPTION_TYPE_I1">
        <Column Name="ExceptionType_LC"/>
  </Index>
</Indices>
```

2. 上記の例で示すように、ShadowColumnName 属性を含めて、シャドー列の名前を指定します。

カスタム・テーブルとハングオフ・テーブルの作成

データベース・フレームワークを使用して、カスタム・テーブルまたはハングオフ・テーブルを作成することによって、アプリケーション・データベースを拡張できます。

カスタム・テーブルは独立したテーブルであり、標準アプリケーション・データベース・テーブルの拡張としてモデル化することはできません。

ハングオフ・テーブルは、標準アプリケーション・データベース・テーブルと多対 1 の関係を持つテーブルです。

カスタム・エンティティーまたはハングオフ・エンティティーを作成することで、以下が可能になります。

- 標準テーブルとハングオフ・テーブルとの間の関係を作成する。
- 拡張可能 API を呼び出して、ハングオフ・テーブルにデータを格納したり取り出したりする。
- dbverify を呼び出して適切な SQL スクリプトを生成し、カスタム・エンティティーまたはハングオフ・エンティティー用のテーブルを作成または変更する。
- アイテム・テーブルと組織テーブルを監査する。

カスタム・テーブルまたはハングオフ・テーブルの作成に適用される以下の点に留意してください。

- *install_dir*/xapidocs/ERD ディレクトリーにある関連エンティティー・リレーションシップ・ダイアグラム (ERD) を参照することによってのみ、ハングオフでエンティティーを使用できるかどうかを決定できます。
- アプリケーションは、Extensions.xml ファイルに基づいて外部キー制約を EFrame_TableChanges.sql に作成しませんが、外部キー関係は強制されます。
- 現在、ハングオフが使用可能とのマークが付けられているのは、オーダー、オーダー明細、作業オーダー、出荷、アイテム、および組織の各テーブルのみです。
- カスタム・テーブル名とハングオフ・テーブル名を、Y で始めることはできません。
- カスタム・テーブルとハングオフ・テーブルの XML 名からは、「Extn」の部分が切り取られます。

- 主要キー名を、Y で始めることはできません。
- 主要キーを、数値データ・タイプにすることができます。
- エンティティー名は、エンティティー定義で指定された接頭辞で始まる必要があります。
- YIFApi インターフェースは、カスタム・テーブルおよびハングオフ・テーブル用の API を拡張しません。したがって、これらのテーブル用の API を、サービスとして構成する必要があります。
- カスタム・テーブルとハングオフ・テーブルをサポートするためにインフラストラクチャーによって作成された API 用の Javadoc は、作成されません。
- カスタム・テーブルとハングオフ・テーブルに対する XSD の生成と検証は実行されません。
- すべてのカスタム・エンティティーとハングオフ・エンティティーには、主要キーを含める必要があります。

列名	データ・タイプ	デフォルト値
Key-Column	キーまたは任意の数値データ・タイプ	' ' (スペース)

- (オプション) カスタム・エンティティーまたはハングオフ・エンティティーには、以下の表に記載された列を含めることができます。

列名	データ・タイプ	デフォルト値
CREATETS	タイム・スタンプ	sysdate
MODIFYTS	タイム・スタンプ	sysdate
CREATEUSERID	ユーザー ID	' ' (スペース)
MODIFYUSERID	ユーザー ID	' ' (スペース)
CREATEPROGID	プログラム ID	' ' (スペース)
MODIFYPROGID	プログラム ID	' ' (スペース)
LOCKID	ロック ID	0 (ゼロ)

注: DB2 データベースでは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales フレームワークによって、タイム・スタンプとして日付データ・タイプが生成されます。

カスタム・テーブルの作成手順

このタスクについて

カスタム・テーブルを作成するには、以下の手順を実行します。

手順

1. *install_dir*/repository/entity/extensions/Extensions.xml.sample ファイルを *install_dir*/extensions/global/entities/*your_filename*.xml ファイルとしてコピーするか、既存の拡張 XML ファイルを変更します。例えば、ABC_CUSTOMER_ORDER_LINE をカスタム・テーブルと仮定します。

2. 以下の例で示すように、*your_filename*.xml ファイルを編集してカスタム・テーブルを作成します。XML 属性の説明については、例の下に示す表を参照してください。

```
<DBSchema>
  <Entities>
     <Entity ApiNeeded="Y/N" AuditRequired="Y" Description=""
      HasHistory="Y/N" Prefix="ABC"
      TableName="ABC_CUSTOMER_ORDER_LINE" >
      <!-- table columns -->
      <Attributes>
         <Attribute ColumnName="CREATETS" DataType="TimeStamp"
           DefaultValue="sysdate" Description="Create TimeStamp" />
         <Attribute ColumnName="MODIFYTS" DataType="TimeStamp"
           DefaultValue="sysdate" Description="Modify TimeStamp" />
         <Attribute ColumnName="CREATEUSERID" DataType="UserId"
           DefaultValue="&apos; &apos;" Description="Creating User ID" />
         <Attribute ColumnName="MODIFYUSERID" DataType="UserId"
           DefaultValue="&apos; &apos;" Description="Modifying User ID" />
         <Attribute ColumnName="CREATEPROGID" DataType="ProgramID"
           DefaultValue="&apos; &apos;" Description="Creating Program ID" />
         <Attribute ColumnName="MODIFYPROGID" DataType="ProgramID"
           DefaultValue="&apos; &apos;" Description="Modifying Program ID" />
         <Attribute ColumnName="LOCKID" DataType="Lockid"
           DefaultValue="0" Description="Lock ID" />
         <Attribute ColumnName="TABLE_KEY" DataType="Key" DefaultValue=" "
           Description="" Nullable="True/False" XMLName="TableKey" />
         .
          .
      </Attributes>
      <!-- PrimaryKey is a mandatory attribute in entity definition.
        This element can have ONLY ONE attribute element -->
      <PrimaryKey Name="TABLE_NAME_PK">
           <Attribute ColumnName="TABLE_KEY" />
      </PrimaryKey>
      <!-- Indices -->
      <Indices>
          <Index Name="INDEX_I1" Unique="True/False">
              <Column Name="Attribute2" />
              .
              .
          </Index>
          .
          .
       </Indices>
       <!-- Relationship -->
       <Parent ParentTableName="YFS_ORDER_LINE" XMLName="YFSOrderLine" >
       <Attribute ColumnName="CUSTOM_ORDER_KEY"
         ParentColumnName="ORDER_LINE_KEY" />
       .
       .
       </Parent>
       <!-- ForeignKeys -->
       <ForeignKeys>
         <ForeignKey ParentTableName="PARENT_ORDER_LINE"
           XMLName="PARENTName1" >
             <Attribute ColumnName="CUSTOM_ORDER_KEY"
               ParentColumnName="PARENT_COLUMN_KEY" />
             .
             .
         </ForeignKey>
         .
         .
       </ForeignKeys>
       <!-- AuditReferences -->
       <AuditReferences>
         <Reference ColumnName="TABLE_KEY" />
```

```
            .
            .
        </AuditReferences>
      </Entity>
    </Entities>
  </DBSchema>
```

3. 以下の表で、エンティティー XML 内の属性を説明します。

属性	説明
Entity	
ApiNeeded	API を生成するかどうかを指定します。有効な値は、Y または N です。Y を渡した場合、デフォルトの一連の API が生成されます。 例えば、ABC_CUSTOMER_ORDER_LINE テーブルの場合、データベース拡張 JAR ファイルを生成するときにアプリケーションは以下の API を作成します。 • getABCCustomerOrderLine() • getABCCustomerOrderLineList() • createABCCustomerOrderLine() • changeABCCustomerOrderLine() • deleteABCCustomerOrderLine() サービス定義フレームワークを使用して、サービスとしてこれらの API にアクセスできます。
AuditRequired	これを Y に設定した場合、このエンティティーの監査レコードが作成されます。
HasHistory	このフラグは、このカスタム・テーブルに履歴テーブルを関連付けることができるかどうかを指定します。 デフォルト値は、N です。 このフラグを Y に設定した場合、履歴テーブルを作成および変更するデータベース・スクリプトを生成するための適切なスクリプトが、dbverify によって生成されます。 カスタム・テーブルの場合、履歴テーブルを生成するには、HasHistory フラグを Y に設定する必要があります。ただし、エンティティー XML 内で Parent 関係を定義している場合、このフラグは親テーブル定義からコピーされます。すべての子エンティティーは、このフラグをオーバーライドすることはできません。
「プレフィックス」	カスタム・テーブルに追加される接頭辞。Y で始まる接頭辞を使用しないことをお勧めします。
TableName	カスタム・テーブルに与えられる名前。
Entity/Attributes/Attribute	
ColumnName	テーブルを構成する列の名前。

属性	説明
DataType	列のデータ・タイプ。有効なデータ・タイプは、*install_dir*/repository/datatypes/datatypes.xml ファイルで指定されます。
CompressionSupported	オプション。この列でデータ圧縮がサポートされるかどうかを指定するために使用される属性。有効な値は、`true` または `false` です。 `true` を指定した場合、圧縮のサポートが有効になります。 **注:** 列に挿入されるデータの CompressionSupported 属性と UseCompression 属性の両方が `true` に設定されている場合、この属性を `false` に設定しないでください。false に設定すると、すべての圧縮済みデータが解凍されないで取り出されることになります。
UseCompression	オプション。この列のデータを圧縮する場合に使用される属性。有効な値は、`true` または `false` です。 `true` を指定した場合、データが圧縮されます。 **注:** CompressionSupported 属性を `true` に設定した場合にのみ、この属性の値を `true` に設定する必要があります。
QueryAllowed	オプション。リスト API のクエリーで、圧縮可能な列を使用可能にするための属性。有効な値は、`true` または `false` です。 `true` を指定した場合、この列は、リスト API のクエリーで使用可能になります。 **注:** CompressionSupported 属性の値を true に設定した場合、この属性の値も `true` に設定する必要があります。
DefaultValue	列のデフォルト値。
説明	Javadoc または ERD で使用可能な、列についての説明。
Nullable	オプション。フィールド値に NULL が可能であることを指定するために使用される属性。デフォルトは false です。主要キー属性およびエンティティー・リレーションシップ以外のすべての列に対して、`Nullable=true` を指定することができます。

属性	説明
XMLName	オプション。属性の XML 名。属性名と異なる場合に指定します。 ベースになる拡張と競合しない名前を選択します。接頭辞として Extn を使用することをお勧めします。また、XMLName への変換規則には、ベース製品で使用されているものと同じ規則を使用することを、強くお勧めします。つまり、列名内のアンダースコアーに続く各文字を大文字にし、それ以外を小文字にします。次に、アンダースコアーを削除します。したがって、Extn_Item_Id は、ExtnItemId となります。
Entity/PrimaryKey	
Name	主要キーのために作成される固有インデックスの名前。この値は、18 文字を超えることはできません。 **注:** 拡張 XML 内の主要キーの名前は、_PK で終わる必要があります。
ColumnName	主要キーとして識別されるテーブル列の名前。
Entity/Indices/Index	
Name	インデックス名。この値は、18 文字を超えることはできません。
Unique	このキーは、カスタム・エンティティーの場合にのみ存在します。有効な値は、true または false です。 true を指定した場合、固有インデックスが作成されます。
AllowCompressedColumns	オプション。これを true に設定すると、CompressionSupported 属性が true に設定された列をインデックスに含めることができます。
Column/ Name	インデックスに関連付けられたテーブル列名。
Entity/Parent	
ParentTableName	このエンティティーが外部キー関係を持つ他のテーブルの名前。
XMLName	親属性の XML 名。親テーブルで使用されている接頭辞で始まる必要があります。 デフォルトでは、親テーブル名が仮定されます。
Parent/Attribute Level	
ParentColumnName	親テーブル内の列名。 注意: エンティティー間に関係を作成するには、親列のデータ・タイプは、CHAR タイプまたは VARCHAR タイプである必要があります。
ColumnName	このカスタム・エンティティー内の列名。
Entity/ForeignKeys/ForeignKey	

属性	説明
ParentTableName	エンティティーが外部キー関係を持つテーブルの名前。
XMLName	要素名の XML 表記。 デフォルトでは、親テーブル名が仮定されます。
Entity/ForeignKeys/ForeignKey/Attribute	
ParentColumnName	親テーブルの列名。 注意: エンティティー間に外部キーを作成するには、親列のデータ・タイプは、CHAR タイプまたは VARCHAR タイプである必要があります。
ColumnName	このカスタム・エンティティー内の列名。
Entity/AuditReferences/Reference	
ColumnName	監査テーブル内の参照列名。

注: エンティティー定義では、Parent 要素と ForeignKey 要素の下に関係を定義できます。

4. ForeignKey 要素の下に定義される関係は、以下を指定します。
   a. 外部テーブルがアプリケーション・データベースのテーブルである場合、外部テーブルの 1 つのレコードに対して、カスタム・テーブル内にゼロ個または複数のレコードが存在することができます。
   b. これは、読み取り専用の関係です。したがって、外部テーブルからレコードを削除しても、このカスタム・テーブルから対応するレコードは削除されません。
5. Parent 要素の下に定義される関係は、以下を指定します。
   a. 親テーブル内の 1 つのレコードに対して、複数の子レコードが存在することができます。
   b. 親テーブルからレコードが削除されると、対応するレコードが子テーブルに存在する場合、それが削除されます。
6. 「API テンプレートの拡張」の手順に従って、対応する API テンプレート (例えば、getOrderDetails() API) を拡張します。

   注: アプリケーションによってカスタム・テーブル用に生成された API を、複数 API のラッパー・コンポーネントを介してサービスとして呼び出すことができます。
7. 拡張をビルドして、デプロイします。

ハングオフ・テーブルの作成手順

このタスクについて

ハングオフ・テーブルを作成するには、以下の手順を実行します。

手順

1. *install_dir*/repository/entity/extensions/Extensions.xml.sample ファイルを *install_dir*/extensions/global/entities/*your_filename*.xml ファイルとしてコピーする

か、既存の拡張 XML ファイルを変更します。例えば、ABC_CUSTOMER_ORDER_LINE をハングオフ・テーブルと仮定します。

2. *your_filename*.xml ファイルを編集して、以下の例で示すようなハングオフ・テーブルを作成します。XML 属性の説明については、例の下に示す表を参照してください。

```
<DBSchema>
   <Entities>
     <Entity ApiNeeded="Y/N" AuditRequired="Y" Description=""
        HasHistory="Y/N" Prefix="ABC"
        TableName="ABC_CUSTOMER_ORDER_LINE" >
        <!-- table columns -->
        <Attributes>
            <Attribute ColumnName="CREATETS" DataType="TimeStamp"
             DefaultValue="sysdate" Description="Create TimeStamp" />
            <Attribute ColumnName="MODIFYTS" DataType="TimeStamp"
             DefaultValue="sysdate" Description="Modify TimeStamp" />
            <Attribute ColumnName="CREATEUSERID" DataType="UserId"
             DefaultValue="&apos; &apos;" Description="Creating User ID" />
            <Attribute ColumnName="MODIFYUSERID" DataType="UserId"
             DefaultValue="&apos; &apos;" Description="Modifying User ID" />
            <Attribute ColumnName="CREATEPROGID" DataType="ProgramID"
             DefaultValue="&apos; &apos;" Description="Creating Program ID" />
            <Attribute ColumnName="MODIFYPROGID" DataType="ProgramID"
             DefaultValue="&apos; &apos;" Description="Modifying Program ID" />
            <Attribute ColumnName="LOCKID" DataType="Lockid" DefaultValue="0"
             Description="Lock ID" />
            <Attribute ColumnName="TABLE_KEY" DataType="Key" DefaultValue=" "
             Description="" Nullable="True/False" XMLName="TableKey" />
             .
             .
        </Attributes>
        <!-- PrimaryKey is a mandatory attribute in entity definition. This
          element can have ONLY ONE attribute element -->
        <PrimaryKey Name="TABLE_NAME_PK">
           <Attribute ColumnName="TABLE_KEY" />
        </PrimaryKey>
        <!-- Indices -->
        <Indices>
            <Index Name="INDEX_I1" Unique="True/False">
                <Column Name="Attribute2" />
                .
                .
            </Index>
            .
            .
         </Indices>
         <!-- Relationship -->
         <Parent ParentTableName="YFS_ORDER_LINE" XMLName="YFSOrderLine" >
            <Attribute ColumnName="CUSTOM_ORDER_KEY"
           ParentColumnName="ORDER_LINE_KEY" />
           .
           .
         </Parent>
         <ForeignKeys>
            <ForeignKey ParentTableName="PARENT_ORDER_LINE"
              XMLName="PARENTName1" >
               <Attribute ColumnName="CUSTOM_ORDER_KEY"
                 ParentColumnName="PARENT_COLUMN_KEY" />
                .
                .
            </ForeignKey>
            .
            .
            </ForeignKeys>
         <!-- AuditReferences -->
```

```
            <AuditReferences>
              <Reference ColumnName="TABLE_KEY" />
              .
              .
            </AuditReferences>
        </Entity>
    </Entities>
</DBSchema>
```

3. 以下の表で、エンティティー XML 内の属性を説明します。

属性	説明
Entity	
ApiNeeded	API を生成するかどうかを指定します。有効な値は、Y または N です。 Y を渡した場合、デフォルトの一連の API が生成されます。 例えば、ABC_CUSTOMER_ORDER_LINE テーブルの場合、データベース拡張 JAR ファイルを生成するときにアプリケーションは以下の API を作成します。 • listABCCustomerOrderLine() • getABCCustomerOrderLine() • createABCCustomerOrderLine() • modifyABCCustomerOrderLine() • deleteABCCustomerOrderLine() サービス定義フレームワークを使用して、サービスとしてこれらの API にアクセスできます。
AuditRequired	これを Y に設定した場合、このエンティティーの監査レコードが作成されます。 注意: オーダー関連テーブルのハングオフを作成する場合は、この属性を渡さないでください。その場合 YFS_ORDER_AUDIT テーブルには、自動的に監査が挿入されます。
説明	Javadoc または ERD で使用可能な、エンティティーの説明。
HasHistory	このフラグは、親テーブルから自動的に継承されます。例えば、YFS_ORDER_HEADER に対するハングオフ・テーブルとして ABC_ORDER_HEADER テーブルを作成し、それに関連履歴テーブルを持たせると仮定します。その場合、データベース・フレームワークによって、自動的に ABC_ORDER_HEADER_H が生成されます。
「プレフィックス」	カスタム・テーブルに追加される接頭辞。Y で始まる接頭辞を使用しないことをお勧めします。
TableName	ハングオフ・テーブルに与えられる名前。
Entity/Attributes/Attribute	
ColumnName	テーブルを構成する列の名前。

属性	説明
DataType	列のデータ・タイプ。有効なデータ・タイプは、*install_dir*/repository/datatypes/datatypes.xml ファイルで指定されます。
CompressionSupported	オプション。この列でデータ圧縮がサポートされるかどうかを指定するために使用される属性。有効な値は、`true` または `false` です。 `true` を指定した場合、圧縮のサポートが有効になります。 **注:** 列に挿入されるデータの CompressionSupported 属性と UseCompression 属性の両方が `true` に設定されている場合、この属性を `false` に設定しないでください。false に設定すると、すべての圧縮済みデータが解凍されないで取り出されることになります。
UseCompression	オプション。この列のデータを圧縮する場合に使用される属性。有効な値は、`true` または `false` です。 `true` を指定した場合、データが圧縮されます。 **注:** CompressionSupported 属性を `true` に設定した場合にのみ、この属性の値を `true` に設定する必要があります。
QueryAllowed	オプション。リスト API のクエリーで、圧縮可能な列を使用可能にするための属性。有効な値は、`true` または `false` です。 `true` を指定した場合、この列は、リスト API のクエリーで使用可能になります。 **注:** CompressionSupported 属性の値を true に設定した場合、この属性の値も `true` に設定する必要があります。
DefaultValue	列のデフォルト値。
説明	Javadoc または ERD で使用可能な、列についての説明。
Nullable	オプション。フィールド値に NULL が可能であることを指定するために使用される属性。デフォルトは false です。主要キー属性およびエンティティー・リレーションシップ以外のすべての列に対して、`Nullable=true` を指定することができます。
XMLName	オプション。属性の XML 名。属性名と異なる場合に指定します。 ベースになる拡張と競合しない名前を選択します。接頭辞として `Extn` を使用することをお勧めします。また、XMLName への変換規則には、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales ベース製品で使用されているものと同じ規則を使用することを、強くお勧めします。つまり、列名内のアンダースコアーに続く各文字を大文字にし、それ以外を小文字にします。次に、アンダースコアーを削除します。したがって、`Extn_Item_Id` は、`ExtnItemId` となります。
Entity/PrimaryKey	

属性	説明
Name	主要キーのために作成される固有インデックスの名前。この値は、18 文字を超えることはできません。 **注:** 拡張 XML 内の主要キーの名前は、_PK で終わる必要があります。
ColumnName	主要キーとして識別されるテーブル列の名前。
Entity/Indices/Index	
Name	インデックス名。この値は、18 文字を超えることはできません。
Unique	このキーは、カスタム・エンティティーの場合にのみ存在します。有効な値は、true または false です。true を指定した場合、固有インデックスが作成されます。
AllowCompressedColumns	オプション。これを true に設定すると、CompressionSupported 属性が true に設定された列をインデックスに含めることができます。
Column/ Name	インデックスに関連付けられたテーブル列名。
Entity/Parent	
ParentTableName	このエンティティーが外部キー関係を持つ他のテーブルの名前。
XMLName	親属性の XML 名。親テーブルで使用されている接頭辞で始まる必要があります。 デフォルトでは、親テーブル名が仮定されます。
Parent/Attribute Level	
ParentColumnName	親テーブル内の列名。 注意: エンティティー間に関係を作成するには、親列のデータ・タイプは、CHAR タイプまたは VARCHAR タイプである必要があります。
ColumnName	このカスタム・エンティティー内の列名。
Entity/ForeignKeys/ForeignKey	
ParentTableName	エンティティーが外部キー関係を持つテーブルの名前。
XMLName	要素名の XML 表記。 デフォルトでは、親テーブル名が仮定されます。
Entity/ForeignKeys/ForeignKey/Attribute	
ParentColumnName	親テーブルの列名。 注意: エンティティー間に外部キーを作成するには、親列のデータ・タイプは、CHAR タイプまたは VARCHAR タイプである必要があります。
ColumnName	このハングオフ・エンティティー内の列名。
Entity/AuditReferences/Reference	
ColumnName	監査テーブル内の参照列名。

注: エンティティー定義では、ForeignKey 要素の下に関係を定義することができます。

4. ForeignKey 要素の下に定義される関係は、以下を指定します。
   a. 外部テーブルが Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales のテーブルである場合、外部テーブルの 1 つのレコードに対して、このハングオフ・テーブル内にゼロ個または複数のレコードが存在することができます。
   b. これは、読み取り専用の関係です。したがって、外部テーブルからレコードを削除しても、このハングオフ・テーブルから対応するレコードは削除されません。
5. 「API テンプレートの拡張」の手順に従って、対応する API テンプレート (例えば、getOrderDetails() API) を拡張します。

   注: ハングオフ・テーブル用に生成された API を、複数 API のラッパー・コンポーネントを介してサービスとして呼び出すことができます。
6. 拡張をビルドして、デプロイします。

ハングオフ・テーブルからのデータのパージ

現在、パージ・エージェントは、レコードを履歴テーブルに移動します。カスタム・エンティティーまたはハングオフ・エンティティーが使用可能な場合、パージ・エージェントはハングオフ・テーブルからもレコードを削除します。ただし、ハングオフ・テーブルのデータは、その親要素もパージされている場合にのみパージすることができます。履歴テーブルが存在する場合、レコードは履歴テーブルに追加されます。履歴テーブルのレコードは、履歴パージ・エージェントを使用して削除されます。

カスタム・エンティティーとハングオフ・エンティティーをパージするには、entities.jar ファイルをエージェント・サーバーのクラスパスに含める必要があります。

第 4 章 データ・タイプ・ファイルの拡張

データ・タイプ・ファイルの拡張

独自の XML 属性と抽象データ・タイプを datatypes.xml ファイルに追加することによって、使用可能な属性を拡張することができます。

以下の場合に、datatypes.xml ファイルを変更する必要があります。

- 既存のデータ・タイプの実際のサイズを変更する場合。
- UI 上の特定のフィールドに入力する長さを制限する場合。例えば、データ・タイプは長さ 20 の入力を許容するが、UI では入力の長さをちょうど 10 に制限する場合です。
- 既存のデータ・タイプが要件を満たさないため、まったく新しいデータ・タイプを作成する場合。

以下の場合に、yfsdatatypemap.xml ファイルを変更する必要があります。

- datatypes.xml ファイルでまったく新しいデータ・タイプを定義しており、その新しいデータ・タイプのマッピングを yfsdatatypemap.xml に定義する場合。

データ・タイプ・マップ・ファイルの拡張

このタスクについて

データ・タイプ・マップ XML ファイルを拡張するには、以下の手順を実行します。

手順

1. *install_dir*/extensions/global/template/resource/yfsdatatypemap.xml ファイルを新規作成します。

   /global/template/resource/ ディレクトリー構造が存在しない場合、必要なディレクトリー構造を作成します。
2. *install_dir*/repository/xapi/template/merged/resource/yfsdatatypemap.xml ファイルに記述されているのと同じ方法で、XML ルート・ノードを追加します。
3. マップする必要のある任意の属性を yfsdatatypemap.xml ファイルに追加します。
4. 拡張をビルドして、デプロイします。

データ・タイプ・ファイルの拡張

このタスクについて

データ・タイプ XML ファイルを拡張するには、以下の手順を実行します。

手順

1. *install_dir*/extensions/global/etc/datatypes.xml ファイルがまだ存在しない場合、新規作成します。

/global/etc/ ディレクトリー構造が存在しない場合、必要なディレクトリー構造を作成します。

注: *install_dir*/extensions/global/etc/datatypes.xml ファイルによってデータ・タイプを拡張している場合、resources.jar を再ビルドしてから dbverify ツールを実行する必要があります。

2. *install_dir*/repository/datatypes/datatypes.xml ファイルに記述されているのと同じ方法で、XML ルート・ノードを追加します。
3. 以下の作業のいずれかを行って、異なるデータ・タイプの値を追加します。
   - 新しいテータ・タイプのパラメーターを追加して定義する。
   - 既存のデータ・タイプのパラメーターを変更する。

   注: 既存のデータ・タイプの場合、datatypes.xml ファイル内の、UI Size や UITableSize などの UI 関連属性のみを変更することができます。

   注: ボード全体のコンソール内の日付入力フィールドをサイズ変更することはできません。対象の日付形式が、アプリケーションが使用するデフォルトの日付形式よりも大きい場合でも、変更できません。

   注: アプリケーションは、内部で使用するために Type 属性を予約しています。そのため、この属性をオーバーライドすることはできません。その他の属性は、すべてオーバーライドが可能です。

   注: 既存のデータ・タイプの場合、文字タイプを数値タイプに変更することはできません。例えば、CHAR タイプを NUMBER に変更することはできません。そのような不正な変換が行われた場合、例外がスローされます。不正として扱われる変換を以下に示します。
   - NVARCHAR、NCHAR、VARCHAR2、BLOB、CLOB、TIME、DATE、DATETIME の各タイプから NUMBER タイプへの変換。
   - NVARCHAR、NCHAR、VARCHAR2、BLOB、CLOB、TIME、DATE、DATETIME の各タイプから SCI_LONG タイプへの変換。
   - NVARCHAR、NCHAR、VARCHAR2、BLOB、CLOB、TIME、DATE、DATETIME の各タイプから SCI_INT タイプへの変換。
   - 任意のタイプから BLOB、CLOB の各タイプへの変換。
   - BLOB、CLOB の各タイプから任意のタイプへの変換。
   - 任意のタイプから DATE、DATETIME、TIME の各タイプへの変換。

   注: 不正な変換は、どのデータベースにも固有ではありません。不正な変換は、古いデータ・タイプの列が保持できたデータを新しいデータ・タイプが保持できるかどうかという基準に基づいています。

   注: 新しいテーブル列の拡張を追加して SQLServer を使用する場合、文字ベースのデータ・タイプとして、NCHAR および NVARCHAR を必ず使用してください。SQLServer では、CHAR データ・タイプおよび VARCHAR データ・タイプを使用しないでください。
4. 拡張をビルドして、デプロイします。

第 5 章 カスタム・ビューの作成

カスタム・ビューを作成して複数のテーブルを結合する

このタスクについて

カスタム・ビューを作成して、複数の DB2 テーブルを結合できます。この手法は、出来合いのテーブルまたはカスタム・テーブルに適用できます。

その後、製品の API を使用して、カスタム・ビューを照会できます。カスタム・クエリーは不要です。

手順

1. 必要に応じて、カスタム・テーブルを作成します。
2. ガイドラインに従って、以下の例を使用してビュー・エンティティーを記述します。

   DB フレームワークは、レコードを取り出すために、ビューに対して DB のクラスと API を生成します。

   制約事項: この方法で生成されるクラスまたは API は、取得およびリスト用の API 関数のみをサポートし、データの修正に使用することはできません。

   dbverify は、ビューの処理または操作を行いません。ビューの XML ファイル内で行われたどの変更も、dbverify によって処理されません。
3. ビューに対応するデータベース API を呼び出すために、該当するコンソールをカスタマイズします。
4. カスタマイズをビルドして、デプロイします。

在庫アイテム検索の例

ビジネス・ケースでは、yfs_item の拡張属性に基づく在庫アイテム検索 (在庫コンソール) が必要になります。そのような検索では、the yfs_item テーブルと yfs_inventory_item テーブルを結合してビューを作成する必要がありますが、yfs_inventory_item テーブルは拡張不能です。

以下の ExtnCatalogSearchConfigProperties.xml ファイルに示すように、ビュー・エンティティーが定義され、ビューに対応するデータベース API を呼び出すように在庫コンソールがカスタマイズされています。

ExtnCatalogSearchConfigProperties.xml ファイルの例を以下に示します。

```
<DBSchema>
  <Entities>
    <Entity TableName="EXTN_INV_ITEM_VW"
            Description="This view joins YFS_ITEM and YFS_INVENTORY_ITEM
        tables to enable querying based on webclass and subclass
        attributes"
            View="true"
            EntityType="VIEW"
```

```
            HasHistory="False"
            AuditRequired="N"
            ApiNeeded="Y"
            Prefix="EXTN" >
        <Attributes>
          <Attribute ColumnName="ITEM_INV_VW_KEY"
                     DataType="Key"
                     DefaultValue="' '"
                     Description="Primary key for this view."
                     Nullable="false"
                     XMLName="ItemInVwKey"/>
          <Attribute ColumnName="ITEM_ID"
                     DataType="ItemID"
                     Description="Identifer for this inventory item."
                     Name="Item_Id"
                     Nullable="false"
                     XMLName="ItemID"
                     DefaultValue="' '" />
          <Attribute ColumnName="UOM"
                     DataType="UOM"
                     Description="Unit of measure for this inventory item."
                     Name="Uom"
                     Nullable="false"
                     XMLName="UnitOfMeasure"
                     DefaultValue="' '" />
          <Attribute ColumnName="EXTN_SUBCLASS"
                     DataType="VARCHAR2-24"
                     Type="VARCHAR2"
                     Size="24"
                     DefaultValue="' '"
                     Nullable="false"
                     XMLName="ExtnSubclass"/>
          <Attribute ColumnName="DEFAULT_PRODUCT_CLASS"
                     DataType="ProductClass"
                     DefaultValue="' '"
                     Description="Default product class of an item."
                     Name="Default_Product_Class"
                     Nullable="false"
                     XMLName="DefaultProductClass"/>
          <Attribute ColumnName="PRODUCT_CLASS"
                     DataType="ProductClass"
                     DefaultValue="' '"
                     Description="Product class for the item of this
                       inventory audit."
                     Name="Product_Class"
                     Nullable="false"/>
          <Attribute ColumnName="DESCRIPTION"
                     DataType="ItemDesc"
                     DefaultValue="' '"
                     Description="Description of the item."
                     Name="Description"
                     Nullable="false"
                     XMLName="Description"/>
        </Attributes>
        <PrimaryKey>
          <Attribute ColumnName="ITEM_INV_VW_KEY"
                     Name="Item_Inv_Vw_Key"/>
        </PrimaryKey>
      </Entity>
    </Entities>
  </DBSchema>
```

第 6 章 エンティティーの監査参照の生成

監査参照の拡張について

このタスクについて

エンティティー XML 内で AuditRequired フラグが有効な場合、監査レコードが YFS_AUDIT テーブルに追加されます。アイテム・テーブルと組織テーブルの場合、このフラグのデフォルトは Y です。ただし、監査フラグと監査参照は、拡張 XML ファイルによってオーバーライドすることができます。

注: キャッシュ・テーブルおよび AuditRequired フラグの値が Y に設定されているテーブルに関連するすべてのレコードは、YFS_AUDIT テーブルに記録されます。

一部の特定のエンティティーについて監査参照の生成をオフに切り替える場合は、そのエンティティーの AuditRequired フラグの値を N に変更します。

注: 新しい監査参照を、拡張 XML ファイルに追加することができます。新しい参照を追加した場合、それらは既存の監査参照よりも優先され、既存の監査参照全体をオーバーライドします。

追加可能な監査参照は、6 つまで限定されています。

アイテムと組織のヘッダー・レベルの監査レコードのみが、YFS_AUDIT_HEADER テーブルに挿入されます。監査参照は、監査されるエンティティーの列を参照します。

エンティティーのテーブル名と監査参照列名を変更することによって、ハングオフ・テーブルおよびカスタム・テーブル用の監査を生成することができます。

注: 複数の親を持つハングオフ・テーブルの場合、監査はサポートされません。

エンティティーに監査参照を生成するには、以下の手順を実行します。

手順

1. *install_dir*/repository/entity/extensions ディレクトリー内の *your_filename*.xml ファイルを編集して、対象となるエンティティーについて監査レコード生成を有効にします。以下の例で、データベース・スキーマに追加する要素について説明します。

```
<DBSchema>
   <Entities>
     <Entity TableName="YFS_ITEM" AuditRequired="Y" >
        .
        .
        <AuditReferences>
            <Reference ColumnName="ItemId" />
            .
            .
        </AuditReferences>
        .
```

```
        .
      </Entity>
    </Entities>
  </DBSchema>
```

属性	説明
Entity	
TableName	監査対象のテーブル名。
AuditRequired	このフラグを Y に設定すると、監査参照が YFS_AUDIT テーブルに入力されます。 **注:** オーダー関連のテーブルのハングオフを作成する場合は、この属性を渡さないでください。その場合監査は、関連するオーダーの監査テーブルに自動的に挿入されます。
Entity/AuditReferences/Reference	
ColumnName	監査参照を含むこのエンティティー内の列名。この名前は、エンティティーに対して有効である必要があります。

2. 追加する監査参照ごとに、Reference タグを新規作成します。
3. オーダー・テーブルのハングオフの監査は、関連付けられたオーダーの監査とともに表示することができます。

第 7 章 API テンプレートの拡張

API テンプレートの拡張について

このタスクについて

テンプレートに基づく各 API は、渡されたテンプレートに応じて異なる出力を提供します。API がテンプレートに基づくかどうかを確認するには、Javadoc を参照してください。

テーブルの変更が API に影響を与える場合、それらの API のテンプレートを拡張する必要があります。拡張した API テンプレートは、*install_dir*/extensions/global/template/api ディレクトリーに配置します。

テーブルの変更によってどの API が影響を受けるかを調べるには、以下の手順を実行します。

手順

1. データベース・エンティティー XML ファイル (すべてのテーブル定義が含まれます) 内で、Entity タグ内の変更するテーブルの XMLName 属性に注目します。それらのデータベース・エンティティー XML ファイルは、*install_dir*/repository/entity ディレクトリーにあります。
2. *install_dir*/extensions/global/template/api ディレクトリー内で、XMLName 属性のパターンを検索します。テーブルの変更または拡張によって影響を受ける公開済み内部 API が、この検索によって見つかります。

   例えば、YFS_CHARGE_CATEGORY テーブル内の属性を拡張する場合を考えます。*install_dir*/repository/entity/omp_tables.xml 内で指定されているこのテーブルの XMLName は、ChargeCategory です。そこで、ChargeCategory 属性を *install_dir*/extensions/global/template/api ディレクトリー内で検索し、この拡張によって影響を受ける API を見つけます。

拡張属性を API テンプレートに含める

拡張属性は、1 次要素の下に、別の <Extn> 要素として現れます。

例えば、getItemDetails() API のデフォルトの出力 XML テンプレートでは、アイテム属性は以下のような構造をしています。

```
<?xml version="1.0" encoding="UTF-8"?>
 <Item .. Item attributes >
    <PrimaryInformation .... PrimaryInformation attributes />
    <ItemServiceSkillList .. ItemServiceSkillList attributes/>
    <ItemAliasList ... ItemAliasList attributes />
   .
   .
 </Item>
```

アイテム・ヘッダーを拡張すると、getItemDetails() API は以下のような XML を出力できるようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
 <Item .. Item attributes >
   <PrimaryInformation .... PrimaryInformation attributes />
   <Extn ExtnAltQty="200408201034469490" ..... extnded attributes />
   <YFSPersonInfo .... PersonInfoKey="200408201034469490" ...../>
   <ItemServiceSkillList .. ItemServiceSkillList attributes/>
   <ItemAliasList ... ItemAliasList attributes />
   .
   .
 </Item>
```

注: 拡張列の外部キー変数は、YFSPersonInfo 要素の PersonInfoKey 属性として現れています。拡張列と PersonInfoKey が同じ値を持つ場合に、関係は有効になります。

拡張属性は、*your_filename*.xml ファイルの XMLName 属性から取り出されます。このファイルは、前のセクションで標準テーブルを拡張した際に編集されています。拡張テンプレートは、*install_dir*/extensions/global/template/api ディレクトリーに配置します。

カスタム・エンティティーとハングオフ・エンティティーを API テンプレートに含める

このタスクについて

カスタム・テーブルまたはハングオフ・テーブルの情報を提供するために、標準 API を拡張することができます。テンプレート XML の templateXmlGen.xml を生成するために特に提供されているツールが、*install_dir*/bin ディレクトリーにあります。

手順

1. 以下のコマンドを使用して、*install_dir* ディレクトリーからテンプレート XML 生成ツールを実行します。

   ```
   sci_ant.sh -Dtable=TABLE_NAME -f bin/templateXmlGen.xml
   ```

2. コマンドを実行すると、サンプル XML ファイルが、*TABLE_NAME*_sample.xml. の名前で *install_dir*/extn/sampleXML ディレクトリーに置かれます。

   例えば、HF_Order_Header が YFS_Order_Header テーブルのハングオフであるとします。その場合、以下のような HF_Order_Header_sample.xml が生成されます。

   ```
   <HFOrderHeader Createprogid=" " Description=" " DocumentType=" "
   EnterpriseKey=" " OrderHeaderKey=" " OrderName=" " OrderNo=" " .... >
   </HFOrderHeader>
   ```

3. 対象のハングオフ・テーブルと関係を持つ YFS テーブルを渡すことによって、上記の属性を標準 API に含めるためのサンプル XML を生成することができます。

   例えば、HF_Order_Header が、YFS_Order_Header テーブルとの関係を持つハングオフ・テーブルであると仮定します。TABLE_NAME=YFS_Order_Header を渡した場合にツールによって生成される XML テンプレートは以下のとおりです。

```
<Order>
  <OrderLines>
     <OrderLine ........>
       <Extn extended attributes >
         <HFOrderHeaderList>
           <HFOrderHeader Createprogid=" " Description=" " .........>
           </HFOrderHeader>
         </HFOrderHeaderList>
       </Extn>
     </OrderLine>
  </OrderLines>
</Order>
```

注: カスタム要素内またはハングオフ要素内の属性のみが変更可能です。

カスタム属性を getOrderDetails 出力テンプレートなどの API テンプレートに含めるために、このサンプル XML を修正することができます。ただし、YFS 要素または YFS 属性を変更することはできません。

注: データベース拡張 JAR ファイルを作成するときにも自動的にサンプル XML がされ、*install_dir*/xapidocs/sampleXML ディレクトリーに置かれます。ただし、サンプル・テンプレートの作成が必要な場合は、対応する YFS テーブル名を指定して、テンプレート XML 生成ツールを別途実行する必要があります。

4. 変更 API 内で Operation 属性を渡すことによって、ハングオフ・テーブルを削除することができます。例えば、changeOrder API 内で以下のようにして HF_Order_Header 要素を削除できます。

```
<Order>
  <OrderLines>
    <OrderLine ........>
        <Extn extended attributes >
           <HFOrderHeaderList>
             <HFOrderHeader Operation="Delete" Createprogid=" " ....... >
             </HFOrderHeader>
           </HFOrderHeaderList>
        </Extn>
    </OrderLine>
  </OrderLines>
</Order>
```

作成および変更の操作は、デフォルトで実行されます。要素のエントリーが存在する場合、API はそのエントリーを最新の値に変更します。要素が存在しない場合は、エントリーが新規作成されます。

5. XML のリスト・レベル要素で、Reset 属性に「true」の値を割り当てることによって、ハングオフ・テーブル内のレコードをリセットすることができます。レコードがリセットされると、親テーブルに対応するハングオフ・テーブルの既存のレコードはすべて削除され、リスト要素の下に含まれていたすべての要素が挿入されます。例えば、HF_Order_Header_list 要素内のハングオフ・レコードは、以下を使用してリセットできます。

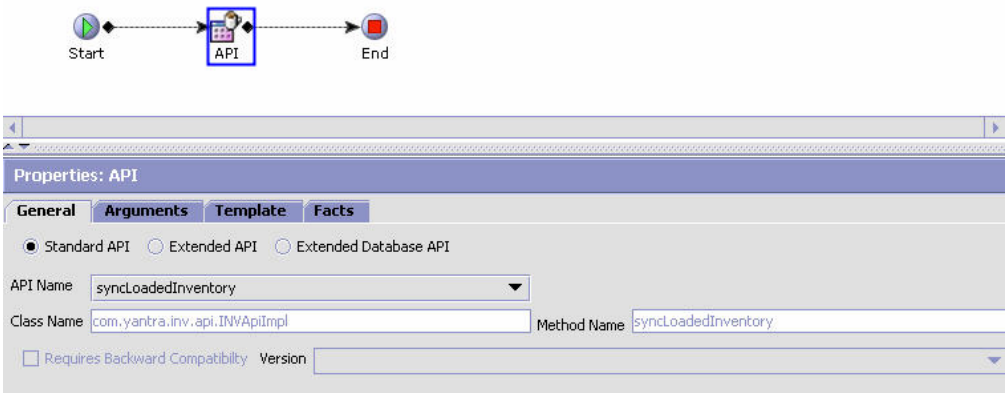
```
<Order>
   <OrderLines>
     <OrderLine ........>
         <Extn extended attributes >
            <HFOrderHeaderList Reset="true">
               <HFOrderHeader>
               </HFOrderHeader>
            </HFOrderHeaderList>
```

```
        </Extn>
      </OrderLine>
    </OrderLines>
  </Order>
```

6. 拡張をビルドして、デプロイします。

カスタム API とハングオフ API のサービスの構成

アプリケーションによってカスタム・エンティティーまたはハングオフ・エンティティー用に生成された API を、複数 API のラッパー・コンポーネントを介してサービスとして呼び出すことができます。複数 API コンポーネントを介した API の呼び出しについて詳しくは、Javadoc を参照してください。これらの API を構成するには、サービス構成ユーザー・インターフェースを使用可能にする必要があります。



カスタム API を含めるために、図に示すようにサービス定義を作成できます。各構成フィールドについて、以下の表で説明します。

フィールド名	説明
「一般」タブ	
標準 API	標準アプリケーション API を呼び出す場合、このオプションを選択します。これを選択した場合、標準 API 名のドロップダウン・リストが表示されます。API ごとにクラス名とメソッド名が表示され、編集することはできません。
拡張 API	カスタム Java コードを呼び出す場合、このオプションを選択します。

フィールド名	説明
拡張データベース API	サービスがカスタム API またはハングオフ API を呼び出す場合、このオプションを選択します。これを選択した場合、カスタム API 名のドロップダウン・リストが表示されます。API ごとにクラス名とメソッド名が表示され、編集することはできません。 **注:** カスタム・テーブルのレコードをロックする場合、SelectMethod 属性を、入力 XML の一部としてカスタム・エンティティー API に渡します。ロックは、カスタム API 呼び出しのトランザクション境界内で発生します。 SelectMethod 属性には、以下の値を設定できます。 • WAIT • NO_WAIT • NONE
「API 名」	呼び出される API を選択または入力します。 **注:** このフィールドは、統合の目的にのみ使用されます。
クラス名	呼び出されるクラスを指定します。
メソッド名	呼び出されるメソッドを指定します。
下位互換性が必要	API に渡される入力データが以前のバージョンからのものであることを示すには、このフィールドを選択します (システム API にのみ適用可能)。
Version	「下位互換性が必要」を選択した場合、API が対応するアプリケーションのバージョンを選択します。個々の API に適用可能なバージョンのみが表示されます。
「引数」タブ	
引数名	「引数」タブに値を入力することで、名前と値のペアを API に渡すことができます。 カスタム API がカスタム値にアクセスするには、カスタム API は com.yantra.interop.japi.YIFCustomApi インターフェースを実装する必要があります。 これを入力すると、これらの名前と値のペアが、Properties オブジェクトとして CustomApi に渡されます。
引数値	引数値を入力します。
「テンプレート」タブ	
	システム API を呼び出すときに、API が使用する出力テンプレートを指定できます。リソース階層ツリー内のサービス定義またはリソース定義 (あるいはその両方) の構成プロパティーで、テンプレートを指定できます。ただし、両方の定義レベルでテンプレートが指定された場合、サービス定義で指定されたテンプレートが使用されます。
XML テンプレート	このラジオ・ボタンを選択して、API 出力に使用される XML を構成します。テンプレートのルート要素名を入力し、「OK」をクリックします。これで、XML を構成できます。

フィールド名	説明
ファイル名	このラジオ・ボタンを選択して、API 出力テンプレートとして使用される XML ファイルのファイル名を入力します。このファイルは、ご使用の CLASSPATH にも存在している必要があります。
「ファクト」タブ	
	ファクトは、接続してデータを取り出すコロニーを特定するために API またはエージェントが使用する属性です。入力されたファクト名とファクト値に基づいて、対応するコロニーが決定されます。
ファクト名	XML 属性のファクト名を入力します。
ファクト値	XML 属性のファクト値を入力します。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510

東京都中央区日本橋箱崎町19番21号

日本アイ・ビー・エム株式会社

法務・知的財産

知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© IBM 2011. このコードの一部は、IBM Corp. のサンプル・プログラムの派生物です。© Copyright IBM Corp. 2011.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript、および PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

IT Infrastructure Library は、英国 Office of Government Commerce の一部である the Central Computer and Telecommunications Agency の登録商標です。

Intel、Intel (ロゴ)、Intel Inside、Intel Inside (ロゴ)、 Intel Centrino、Intel Centrino (ロゴ)、Celeron、Intel Xeon、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

ITIL は英国 Office of Government Commerce の登録商標および共同体登録商標であって、米国特許商標庁にて登録されています。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

Cell Broadband Engine, Cell/B.E は、米国およびその他の国における Sony Computer Entertainment, Inc. の商標であり、同社の許諾を受けて使用しています。

Linear Tape-Open、LTO、LTO ロゴ、Ultrium および Ultrium ロゴは、米国およびその他の国における HP、IBM Corp. および Quantum の商標です。

Connect Control Center®、Connect:Direct®、Connect:Enterprise™、Gentran®、Gentran®:Basic®、Gentran:Control®、Gentran:Director®、Gentran:Plus®、Gentran:Realtime®、Gentran:Server®、Gentran:Viewpoint®、Sterling Commerce™、Sterling Information Broker®、および Sterling Integrator® は、Sterling Commerce™, Inc.、IBM Company の商標です。

プログラム番号: xxxx-xxx

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21